RCAC Biocontainers

Release v1.0

Yucheng Zhang

Oct 17, 2022

FREQUENTLY ASKED QUESTIONS

1 Frequently Asked Questions **3**

2 Singularity **5**
- 2.1 What is Singularity? . . . 5
- 2.2 Features . . . 5
- 2.3 Example . . . 5
- 2.4 Purdue Cluster Specific Notes . . . 6
- 2.5 Creating Singularity Images . . . 6

3 Abacas **9**
- 3.1 Introduction . . . 9
- 3.2 Versions . . . 9
- 3.3 Commands . . . 9
- 3.4 Module . . . 9
- 3.5 Example job . . . 10

4 Abismal **11**
- 4.1 Introduction . . . 11
- 4.2 Versions . . . 11
- 4.3 Commands . . . 11
- 4.4 Module . . . 11
- 4.5 Example job . . . 12

5 Abricate **13**
- 5.1 Introduction . . . 13
- 5.2 Versions . . . 13
- 5.3 Commands . . . 13
- 5.4 Module . . . 13
- 5.5 Example job . . . 13

6 Abyss **15**
- 6.1 Introduction . . . 15
- 6.2 Versions . . . 15
- 6.3 Commands . . . 15
- 6.4 Module . . . 17
- 6.5 Example job . . . 17

7 Actc **19**
- 7.1 Introduction . . . 19
- 7.2 Versions . . . 19
- 7.3 Commands . . . 19

7.4 Module . . . 19
7.5 Example job . . . 19

8 Advntr **21**
8.1 Introduction . . . 21
8.2 Versions . . . 21
8.3 Commands . . . 21
8.4 Module . . . 21
8.5 Example job . . . 21

9 Afplot **23**
9.1 Introduction . . . 23
9.2 Versions . . . 23
9.3 Commands . . . 23
9.4 Module . . . 23
9.5 Example job . . . 23

10 Afterqc **25**
10.1 Introduction . . . 25
10.2 Versions . . . 25
10.3 Commands . . . 25
10.4 Module . . . 25
10.5 Example job . . . 25

11 Agat **27**
11.1 Introduction . . . 27
11.2 Versions . . . 27
11.3 Commands . . . 27
11.4 Module . . . 29
11.5 Example job . . . 29

12 Alfred **31**
12.1 Introduction . . . 31
12.2 Versions . . . 31
12.3 Commands . . . 31
12.4 Module . . . 31
12.5 Example job . . . 32

13 Alien-hunter **33**
13.1 Introduction . . . 33
13.2 Versions . . . 33
13.3 Commands . . . 33
13.4 Module . . . 33
13.5 Example job . . . 33

14 Alignstats **35**
14.1 Introduction . . . 35
14.2 Versions . . . 35
14.3 Commands . . . 35
14.4 Module . . . 35
14.5 Example job . . . 36

15 Allpathslg **37**
15.1 Introduction . . . 37
15.2 Versions . . . 37

15.3 Commands . . . 37
15.4 Module . . . 37
15.5 Example job . . . 38

16 Alphafold **39**
16.1 Introduction . . . 39
16.2 Versions . . . 39
16.3 Commands . . . 39
16.4 Module . . . 39
16.5 Usage . . . 40
16.6 AlphaDB . . . 40
16.7 Example job using CPU . . . 40
16.8 Example job using GPU . . . 41

17 Amptk **43**
17.1 Introduction . . . 43
17.2 Versions . . . 43
17.3 Commands . . . 43
17.4 Module . . . 43
17.5 Example job . . . 43

18 Ananse **45**
18.1 Introduction . . . 45
18.2 Versions . . . 45
18.3 Commands . . . 45
18.4 Module . . . 45
18.5 Example job . . . 46

19 Anchorwave **47**
19.1 Introduction . . . 47
19.2 Versions . . . 47
19.3 Commands . . . 47
19.4 Module . . . 47
19.5 Example job . . . 48

20 ANGSD **49**
20.1 Introduction . . . 49
20.2 Versions . . . 49
20.3 Commands . . . 49
20.4 Module . . . 49
20.5 Example job . . . 50

21 Annogesic **51**
21.1 Introduction . . . 51
21.2 Versions . . . 51
21.3 Commands . . . 51
21.4 Module . . . 51
21.5 Example job . . . 52

22 ANNOVAR **53**
22.1 Introduction . . . 53
22.2 Versions . . . 53
22.3 Commands . . . 53
22.4 Module . . . 53
22.5 Example job . . . 54

23 Antismash **55**
23.1 Introduction . . . 55
23.2 Versions . . . 55
23.3 Commands . . . 55
23.4 Module . . . 55
23.5 Example job . . . 56

24 Anvio **57**
24.1 Introduction . . . 57
24.2 Versions . . . 57
24.3 Commands . . . 57
24.4 Module . . . 61
24.5 Example job . . . 61

25 Any2fasta **63**
25.1 Introduction . . . 63
25.2 Versions . . . 63
25.3 Commands . . . 63
25.4 Module . . . 63
25.5 Example job . . . 64

26 Arcs **65**
26.1 Introduction . . . 65
26.2 Versions . . . 65
26.3 Commands . . . 65
26.4 Module . . . 65
26.5 Example job . . . 66

27 ASGAL **67**
27.1 Introduction . . . 67
27.2 Versions . . . 67
27.3 Commands . . . 67
27.4 Module . . . 67
27.5 Example job . . . 67

28 Assembly-stats **69**
28.1 Introduction . . . 69
28.2 Versions . . . 69
28.3 Commands . . . 69
28.4 Module . . . 69
28.5 Example job . . . 69

29 Atac-seq-pipeline **71**
29.1 Introduction . . . 71
29.2 Versions . . . 71
29.3 Commands . . . 71
29.4 Module . . . 78
29.5 Example job . . . 78

30 Ataqv **79**
30.1 Introduction . . . 79
30.2 Versions . . . 79
30.3 Commands . . . 79
30.4 Module . . . 79
30.5 Example job . . . 79

31 aTRAM **81**
31.1 Introduction . . . 81
31.2 Versions . . . 81
31.3 Commands . . . 81
31.4 Module . . . 81
31.5 Example job . . . 81

32 Atropos **83**
32.1 Introduction . . . 83
32.2 Versions . . . 83
32.3 Commands . . . 83
32.4 Module . . . 83
32.5 Example job . . . 83

33 Augur **85**
33.1 Introduction . . . 85
33.2 Versions . . . 85
33.3 Commands . . . 85
33.4 Module . . . 85
33.5 Example job . . . 85

34 AUGUSTUS **87**
34.1 Introduction . . . 87
34.2 Versions . . . 87
34.3 Commands . . . 87
34.4 Module . . . 88
34.5 Example job . . . 88

35 Bactopia **89**
35.1 Introduction . . . 89
35.2 Versions . . . 89
35.3 Commands . . . 89
35.4 Module . . . 89
35.5 Example job . . . 90

36 Bali-phy **91**
36.1 Introduction . . . 91
36.2 Versions . . . 91
36.3 Commands . . . 91
36.4 Module . . . 91
36.5 Example job . . . 92

37 Bamgineer **93**
37.1 Introduction . . . 93
37.2 Versions . . . 93
37.3 Commands . . . 93
37.4 Module . . . 93
37.5 Example job . . . 94

38 Bamliquidator **95**
38.1 Introduction . . . 95
38.2 Versions . . . 95
38.3 Commands . . . 95
38.4 Module . . . 95
38.5 Example job . . . 96

39 Bam-readcount **97**
39.1 Introduction . . . 97
39.2 Versions . . . 97
39.3 Commands . . . 97
39.4 Module . . . 97
39.5 Example job . . . 97

40 Bamsurgeon **99**
40.1 Introduction . . . 99
40.2 Versions . . . 99
40.3 Commands . . . 99
40.4 Module . . . 99
40.5 Example job . . . 100

41 BamTools **101**
41.1 Introduction . . . 101
41.2 Versions . . . 101
41.3 Commands . . . 101
41.4 Module . . . 101
41.5 Example job . . . 101

42 Bamutil **103**
42.1 Introduction . . . 103
42.2 Versions . . . 103
42.3 Commands . . . 103
42.4 Module . . . 103
42.5 Example job . . . 103

43 Barrnap **105**
43.1 Introduction . . . 105
43.2 Versions . . . 105
43.3 Commands . . . 105
43.4 Module . . . 105
43.5 Example job . . . 105

44 Basenji **107**
44.1 Introduction . . . 107
44.2 Versions . . . 107
44.3 Commands . . . 107
44.4 Module . . . 109
44.5 Example job . . . 110

45 Bbmap **111**
45.1 Introduction . . . 111
45.2 Versions . . . 111
45.3 Commands . . . 111
45.4 Module . . . 115
45.5 Example job . . . 116

46 Bbtools **117**
46.1 Introduction . . . 117
46.2 Versions . . . 117
46.3 Commands . . . 117
46.4 Module . . . 123
46.5 Example job . . . 123

47 Bcftools **125**
47.1 Introduction . . . 125
47.2 Versions . . . 125
47.3 Commands . . . 125
47.4 Module . . . 126
47.5 Example job . . . 126

48 Bcl2fastq **127**
48.1 Introduction . . . 127
48.2 Versions . . . 127
48.3 Commands . . . 127
48.4 Module . . . 127
48.5 Example job . . . 128

49 Beagle **129**
49.1 Introduction . . . 129
49.2 Versions . . . 129
49.3 Commands . . . 129
49.4 Module . . . 129
49.5 Example job . . . 129

50 BEAST 2 **131**
50.1 Introduction . . . 131
50.2 Versions . . . 131
50.3 Commands . . . 131
50.4 Module . . . 132
50.5 Example job . . . 132

51 Bedops **133**
51.1 Introduction . . . 133
51.2 Versions . . . 133
51.3 Commands . . . 133
51.4 Module . . . 138
51.5 Example job . . . 138

52 Bedtools **139**
52.1 Introduction . . . 139
52.2 Versions . . . 139
52.3 Commands . . . 139
52.4 Module . . . 140
52.5 Example job . . . 140

53 Bioawk **143**
53.1 Introduction . . . 143
53.2 Versions . . . 143
53.3 Commands . . . 143
53.4 Module . . . 143
53.5 Example job . . . 143

54 Biobambam **145**
54.1 Introduction . . . 145
54.2 Versions . . . 145
54.3 Commands . . . 145
54.4 Module . . . 147
54.5 Example job . . . 147

55 Bioconvert **149**
55.1 Introduction . . . 149
55.2 Versions . . . 149
55.3 Commands . . . 149
55.4 Module . . . 149
55.5 Example job . . . 150

56 Biopython **151**
56.1 Introduction . . . 151
56.2 Versions . . . 151
56.3 Commands . . . 151
56.4 Module . . . 152
56.5 Interactive job . . . 152
56.6 Batch job . . . 152

57 Bismark **155**
57.1 Introduction . . . 155
57.2 Versions . . . 155
57.3 Commands . . . 155
57.4 Dependencies . . . 156
57.5 Module . . . 156
57.6 Example job . . . 156

58 Blasr **157**
58.1 Introduction . . . 157
58.2 Versions . . . 157
58.3 Commands . . . 157
58.4 Module . . . 157
58.5 Example job . . . 157

59 BLAST **159**
59.1 Introduction . . . 159
59.2 Versions . . . 159
59.3 Commands . . . 159
59.4 Module . . . 160
59.5 BLAST Databases . . . 160
59.6 Example job . . . 160

60 BlobTools **163**
60.1 Introduction . . . 163
60.2 Versions . . . 163
60.3 Commands . . . 163
60.4 Module . . . 163
60.5 Example job . . . 163

61 Bmge **165**
61.1 Introduction . . . 165
61.2 Versions . . . 165
61.3 Commands . . . 165
61.4 Module . . . 165
61.5 Example job . . . 165

62 Bowtie **167**
62.1 Introduction . . . 167
62.2 Versions . . . 167

62.3 Commands . . . 167
62.4 Module . . . 167
62.5 Example job . . . 168

63 Bowtie 2 **169**
63.1 Introduction . . . 169
63.2 Versions . . . 169
63.3 Commands . . . 169
63.4 Module . . . 169
63.5 Example job . . . 170

64 Bracken **171**
64.1 Introduction . . . 171
64.2 Versions . . . 171
64.3 Commands . . . 171
64.4 Module . . . 172
64.5 Example job . . . 172

65 BRAKER **173**
65.1 Introduction . . . 173
65.2 Versions . . . 173
65.3 Commands . . . 173
65.4 Helper command . . . 173
65.5 Module . . . 174
65.6 Example job . . . 174

66 Brass **175**
66.1 Introduction . . . 175
66.2 Versions . . . 175
66.3 Commands . . . 175
66.4 Module . . . 176
66.5 Example job . . . 176

67 Breseq **177**
67.1 Introduction . . . 177
67.2 Versions . . . 177
67.3 Commands . . . 177
67.4 Module . . . 177
67.5 Example job . . . 177

68 BUSCO **179**
68.1 Introduction . . . 179
68.2 Versions . . . 179
68.3 Commands . . . 179
68.4 Helper command . . . 179
68.5 Module . . . 180
68.6 Example job for prokaryotic genomes . . . 180
68.7 Example job for eukaryotic genomes . . . 180

69 Bustools **183**
69.1 Introduction . . . 183
69.2 Versions . . . 183
69.3 Commands . . . 183
69.4 Module . . . 183
69.5 Example job . . . 183

70 BWA **185**
70.1 Introduction . . . 185
70.2 Versions . . . 185
70.3 Commands . . . 185
70.4 Module . . . 185
70.5 Example job . . . 186

71 Bwameth **187**
71.1 Introduction . . . 187
71.2 Versions . . . 187
71.3 Commands . . . 187
71.4 Module . . . 187
71.5 Example job . . . 188

72 Cactus **189**
72.1 Introduction . . . 189
72.2 Versions . . . 189
72.3 Commands . . . 189
72.4 Module . . . 190
72.5 Example job . . . 190

73 Cafe **193**
73.1 Introduction . . . 193
73.2 Versions . . . 193
73.3 Commands . . . 193
73.4 Module . . . 193
73.5 Example job . . . 193

74 Canu **195**
74.1 Introduction . . . 195
74.2 Versions . . . 195
74.3 Commands . . . 195
74.4 Module . . . 195
74.5 Example job . . . 195

75 Ccs **197**
75.1 Introduction . . . 197
75.2 Versions . . . 197
75.3 Commands . . . 197
75.4 Module . . . 197
75.5 Example job . . . 198

76 Cdbtools **199**
76.1 Introduction . . . 199
76.2 Versions . . . 199
76.3 Commands . . . 199
76.4 Module . . . 199
76.5 Example job . . . 200

77 Cd-hit **201**
77.1 Introduction . . . 201
77.2 Versions . . . 201
77.3 Commands . . . 201
77.4 Module . . . 202
77.5 Example job . . . 202

78 Cegma **205**
78.1 Introduction . . . 205
78.2 Versions . . . 205
78.3 Commands . . . 205
78.4 Module . . . 205
78.5 Example job . . . 206

79 Cellbender **207**
79.1 Introduction . . . 207
79.2 Versions . . . 207
79.3 Commands . . . 207
79.4 Module . . . 207
79.5 Example job . . . 207

80 Cellphonedb **209**
80.1 Introduction . . . 209
80.2 Versions . . . 209
80.3 Commands . . . 209
80.4 Module . . . 209
80.5 Example job . . . 210

81 Cellranger **211**
81.1 Introduction . . . 211
81.2 Versions . . . 211
81.3 Commands . . . 211
81.4 Module . . . 211
81.5 Example job . . . 212

82 Cellranger-arc **213**
82.1 Introduction . . . 213
82.2 Versions . . . 213
82.3 Commands . . . 213
82.4 Module . . . 213
82.5 Example job . . . 214

83 Cellranger-atac **215**
83.1 Introduction . . . 215
83.2 Versions . . . 215
83.3 Commands . . . 215
83.4 Module . . . 215
83.5 Example job . . . 216

84 CellRank **217**
84.1 Introduction . . . 217
84.2 Versions . . . 217
84.3 Commands . . . 217
84.4 Module . . . 218
84.5 Interactive job . . . 218
84.6 Batch job . . . 218

85 CellRank-krylov **221**
85.1 Introduction . . . 221
85.2 Versions . . . 221
85.3 Commands . . . 221
85.4 Module . . . 222

85.5 Interactive job . . . 222
85.6 Batch job . . . 222

86 cellSNP **225**
86.1 Introduction . . . 225
86.2 Versions . . . 225
86.3 Commands . . . 225
86.4 Module . . . 225
86.5 Example job . . . 226

87 Celltypist **227**
87.1 Introduction . . . 227
87.2 Versions . . . 227
87.3 Commands . . . 227
87.4 Module . . . 227
87.5 Example job . . . 228

88 Centrifuge **229**
88.1 Introduction . . . 229
88.2 Versions . . . 229
88.3 Commands . . . 229
88.4 Module . . . 230
88.5 Example job . . . 230

89 Checkm-genome **231**
89.1 Introduction . . . 231
89.2 Versions . . . 231
89.3 Commands . . . 231
89.4 Module . . . 231
89.5 Example job . . . 232

90 Chewbbaca **233**
90.1 Introduction . . . 233
90.2 Versions . . . 233
90.3 Commands . . . 233
90.4 Module . . . 233
90.5 Example job . . . 234

91 Chromap **235**
91.1 Introduction . . . 235
91.2 Versions . . . 235
91.3 Commands . . . 235
91.4 Module . . . 235
91.5 Example job . . . 236

92 CICERO **237**
92.1 Introduction . . . 237
92.2 Versions . . . 237
92.3 Commands . . . 237
92.4 Module . . . 237
92.5 Example job . . . 237

93 Circexplorer2 **239**
93.1 Introduction . . . 239
93.2 Versions . . . 239

93.3 Commands . . . 239
93.4 Module . . . 239
93.5 Example job . . . 240

94 Circlator **241**
94.1 Introduction . . . 241
94.2 Versions . . . 241
94.3 Commands . . . 241
94.4 Module . . . 241
94.5 Example job . . . 242

95 Circompara2 **243**
95.1 Introduction . . . 243
95.2 Versions . . . 243
95.3 Commands . . . 243
95.4 Module . . . 246
95.5 Example job . . . 246

96 Circos **247**
96.1 Introduction . . . 247
96.2 Versions . . . 247
96.3 Commands . . . 247
96.4 Module . . . 247
96.5 Example job . . . 247

97 CIRIquant **249**
97.1 Introduction . . . 249
97.2 Versions . . . 249
97.3 Commands . . . 249
97.4 Module . . . 249
97.5 config.yml . . . 249
97.6 Example job . . . 250

98 Clair3 **251**
98.1 Introduction . . . 251
98.2 Versions . . . 251
98.3 Commands . . . 251
98.4 Module . . . 251
98.5 Model_path . . . 252
98.6 Example job . . . 252

99 Clairvoyante **253**
99.1 Introduction . . . 253
99.2 Versions . . . 253
99.3 Commands . . . 253
99.4 Module . . . 253
99.5 Example job . . . 254

100Clearcnv **255**
100.1 Introduction . . . 255
100.2 Versions . . . 255
100.3 Commands . . . 255
100.4 Module . . . 255
100.5 Example job . . . 255

101Clever-toolkit **257**
101.1 Introduction . . . 257
101.2 Versions . . . 257
101.3 Commands . . . 257
101.4 Module . . . 258
101.5 Example job . . . 258

102Clustalw **259**
102.1 Introduction . . . 259
102.2 Versions . . . 259
102.3 Commands . . . 259
102.4 Module . . . 259
102.5 Example job . . . 259

103CNVkit **261**
103.1 Introduction . . . 261
103.2 Versions . . . 261
103.3 Commands . . . 261
103.4 Module . . . 261
103.5 Example job . . . 262

104Cnvnator **263**
104.1 Introduction . . . 263
104.2 Versions . . . 263
104.3 Commands . . . 263
104.4 Module . . . 263
104.5 Example job . . . 264

105Coinfinder **265**
105.1 Introduction . . . 265
105.2 Versions . . . 265
105.3 Commands . . . 265
105.4 Module . . . 265
105.5 Example job . . . 266

106CONCOCT **267**
106.1 Introduction . . . 267
106.2 Versions . . . 267
106.3 Commands . . . 267
106.4 Module . . . 267
106.5 Example job . . . 268

107Control-freec **269**
107.1 Introduction . . . 269
107.2 Versions . . . 269
107.3 Commands . . . 269
107.4 Module . . . 269
107.5 Example job . . . 269

108Cooler **271**
108.1 Introduction . . . 271
108.2 Versions . . . 271
108.3 Commands . . . 271
108.4 Module . . . 271
108.5 Interactive job . . . 272

108.6 Batch job . . . 272

109Coverm **273**
109.1 Introduction . . . 273
109.2 Versions . . . 273
109.3 Commands . . . 273
109.4 Module . . . 273
109.5 Example job . . . 273

110CRISPRCasFinder **275**
110.1 Introduction . . . 275
110.2 Versions . . . 275
110.3 Commands . . . 275
110.4 Module . . . 275
110.5 Example job . . . 275

111Crispresso2 **277**
111.1 Introduction . . . 277
111.2 Versions . . . 277
111.3 Commands . . . 277
111.4 Module . . . 278
111.5 Example job . . . 278

112Crispritz **279**
112.1 Introduction . . . 279
112.2 Versions . . . 279
112.3 Commands . . . 279
112.4 Module . . . 279
112.5 Example job . . . 279

113Crossmap **281**
113.1 Introduction . . . 281
113.2 Versions . . . 281
113.3 Commands . . . 281
113.4 Module . . . 281
113.5 Example job . . . 281

114cross_match **283**
114.1 Introduction . . . 283
114.2 Versions . . . 283
114.3 Commands . . . 283
114.4 Module . . . 283
114.5 Example job . . . 283

115Csvtk **285**
115.1 Introduction . . . 285
115.2 Versions . . . 285
115.3 Commands . . . 285
115.4 Module . . . 285
115.5 Example job . . . 285

116Cufflinks **287**
116.1 Introduction . . . 287
116.2 Versions . . . 287
116.3 Commands . . . 287

116.4 Module . . . 288
116.5 Example job . . . 288

117Cutadapt **289**
117.1 Introduction . . . 289
117.2 Versions . . . 289
117.3 Commands . . . 289
117.4 Module . . . 289
117.5 Example job . . . 290

118Cyvcf2 **291**
118.1 Introduction . . . 291
118.2 Versions . . . 291
118.3 Commands . . . 291
118.4 Module . . . 291
118.5 Interactive job . . . 292
118.6 Batch job . . . 292

119Dbg2olc **293**
119.1 Introduction . . . 293
119.2 Versions . . . 293
119.3 Commands . . . 293
119.4 Module . . . 294
119.5 Example job . . . 294

120Deepbgc **295**
120.1 Introduction . . . 295
120.2 Versions . . . 295
120.3 Commands . . . 295
120.4 Module . . . 295
120.5 Example job . . . 295

121Deepconsensus **297**
121.1 Introduction . . . 297
121.2 Versions . . . 297
121.3 Commands . . . 297
121.4 Module . . . 297
121.5 Example job . . . 298

122Deepsignal2 **299**
122.1 Introduction . . . 299
122.2 Versions . . . 299
122.3 Commands . . . 299
122.4 Module . . . 300
122.5 Example job . . . 300

123DeepTools **301**
123.1 Introduction . . . 301
123.2 Versions . . . 301
123.3 Commands . . . 301
123.4 Module . . . 302
123.5 Example job . . . 302

124Deepvariant **303**
124.1 Introduction . . . 303

124.2 Versions . . . 303
124.3 Commands . . . 303
124.4 Module . . . 304
124.5 Example job . . . 304

125Delly **305**
125.1 Introduction . . . 305
125.2 Versions . . . 305
125.3 Commands . . . 305
125.4 Module . . . 305
125.5 Example job . . . 306

126Diamond **307**
126.1 Introduction . . . 307
126.2 Versions . . . 307
126.3 Commands . . . 307
126.4 Module . . . 308
126.5 Example job . . . 308

127Dnaio **309**
127.1 Introduction . . . 309
127.2 Versions . . . 309
127.3 Commands . . . 309
127.4 Module . . . 309
127.5 Example job . . . 309

128Dragonflye **311**
128.1 Introduction . . . 311
128.2 Versions . . . 311
128.3 Commands . . . 311
128.4 Module . . . 311
128.5 Example job . . . 312

129Drep **313**
129.1 Introduction . . . 313
129.2 Versions . . . 313
129.3 Commands . . . 313
129.4 Module . . . 313
129.5 Example job . . . 313

130Dropest **315**
130.1 Introduction . . . 315
130.2 Versions . . . 315
130.3 Commands . . . 315
130.4 Module . . . 315
130.5 Example job . . . 316

131Dsuite **317**
131.1 Introduction . . . 317
131.2 Versions . . . 317
131.3 Commands . . . 317
131.4 Module . . . 317
131.5 Example job . . . 318

132easySFS **319**

132.1 Introduction . . . 319
132.2 Versions . . . 319
132.3 Commands . . . 319
132.4 Module . . . 319
132.5 Example job . . . 319

133Edta **321**
133.1 Introduction . . . 321
133.2 Versions . . . 321
133.3 Commands . . . 321
133.4 Module . . . 323
133.5 Example job . . . 323

134Eggnog-mapper **325**
134.1 Introduction . . . 325
134.2 Versions . . . 325
134.3 Commands . . . 325
134.4 Module . . . 325
134.5 Example job . . . 326

135Emboss **327**
135.1 Introduction . . . 327
135.2 Versions . . . 327
135.3 Commands . . . 327
135.4 Module . . . 335
135.5 Example job . . . 335

136Ensembl-vep **337**
136.1 Introduction . . . 337
136.2 Versions . . . 337
136.3 Commands . . . 337
136.4 Module . . . 337
136.5 Example job . . . 338

137Epic2 **339**
137.1 Introduction . . . 339
137.2 Versions . . . 339
137.3 Commands . . . 339
137.4 Module . . . 339
137.5 Example job . . . 340

138Evidencemodeler **341**
138.1 Introduction . . . 341
138.2 Versions . . . 341
138.3 Commands . . . 341
138.4 Module . . . 342
138.5 Example job . . . 342

139Exonerate **343**
139.1 Introduction . . . 343
139.2 Versions . . . 343
139.3 Commands . . . 343
139.4 Module . . . 343
139.5 Example job . . . 343

140Fasta3 **345**
140.1 Introduction . . . 345
140.2 Versions . . . 345
140.3 Commands . . . 345
140.4 Module . . . 346
140.5 Example job . . . 346

141FastANI **347**
141.1 Introduction . . . 347
141.2 Versions . . . 347
141.3 Commands . . . 347
141.4 Module . . . 347
141.5 Example job . . . 347

142Fastp **349**
142.1 Introduction . . . 349
142.2 Versions . . . 349
142.3 Commands . . . 349
142.4 Module . . . 349
142.5 Example job . . . 349

143FastQC **351**
143.1 Introduction . . . 351
143.2 Versions . . . 351
143.3 Commands . . . 351
143.4 Module . . . 351
143.5 Example job . . . 352

144Fastq_pair **353**
144.1 Introduction . . . 353
144.2 Versions . . . 353
144.3 Commands . . . 353
144.4 Module . . . 353
144.5 Example job . . . 353

145Fastq-scan **355**
145.1 Introduction . . . 355
145.2 Versions . . . 355
145.3 Commands . . . 355
145.4 Module . . . 355
145.5 Example job . . . 356

146Fastspar **357**
146.1 Introduction . . . 357
146.2 Versions . . . 357
146.3 Commands . . . 357
146.4 Module . . . 357
146.5 Example job . . . 358

147fastStructure **359**
147.1 Introduction . . . 359
147.2 Versions . . . 359
147.3 Commands . . . 359
147.4 Module . . . 360
147.5 Example job . . . 360

148FastTree **361**
148.1 Introduction . . . 361
148.2 Versions . . . 361
148.3 Commands . . . 361
148.4 Module . . . 361
148.5 Example job using single CPU . . . 362
148.6 Example job using multiple CPUs . . . 362

149FASTX-Toolkit **363**
149.1 Introduction . . . 363
149.2 Versions . . . 363
149.3 Commands . . . 363
149.4 Module . . . 364
149.5 Example job . . . 364

150Filtlong **365**
150.1 Introduction . . . 365
150.2 Versions . . . 365
150.3 Commands . . . 365
150.4 Module . . . 365
150.5 Example job . . . 365

151Flye **367**
151.1 Introduction . . . 367
151.2 Versions . . . 367
151.3 Commands . . . 367
151.4 Module . . . 367
151.5 Example job . . . 368

152Fraggenescan **369**
152.1 Introduction . . . 369
152.2 Versions . . . 369
152.3 Commands . . . 369
152.4 Module . . . 369
152.5 Example job . . . 370

153Fraggenescanrs **371**
153.1 Introduction . . . 371
153.2 Versions . . . 371
153.3 Commands . . . 371
153.4 Module . . . 371
153.5 Example job . . . 372

154Freebayes **373**
154.1 Introduction . . . 373
154.2 Versions . . . 373
154.3 Commands . . . 373
154.4 Module . . . 373
154.5 Example job . . . 374

155Fseq **375**
155.1 Introduction . . . 375
155.2 Versions . . . 375
155.3 Commands . . . 375
155.4 Module . . . 375

155.5 Example job . . . 375

156Funannotate **377**
156.1 Introduction . . . 377
156.2 Versions . . . 377
156.3 Commands . . . 377
156.4 Module . . . 377
156.5 Example job . . . 377

157Fwdpy11 **379**
157.1 Introduction . . . 379
157.2 Versions . . . 379
157.3 Commands . . . 379
157.4 Module . . . 379
157.5 Interactive job . . . 380
157.6 Batch job . . . 380

158Gadma **381**
158.1 Introduction . . . 381
158.2 Versions . . . 381
158.3 Commands . . . 381
158.4 Module . . . 381
158.5 Interactive job . . . 382
158.6 Batch job . . . 382

159Gambit **383**
159.1 Introduction . . . 383
159.2 Versions . . . 383
159.3 Commands . . . 383
159.4 Module . . . 383
159.5 Example job . . . 384

160Gamma **385**
160.1 Introduction . . . 385
160.2 Versions . . . 385
160.3 Commands . . . 385
160.4 Module . . . 385
160.5 Example job . . . 386

161GATK **387**
161.1 Introduction . . . 387
161.2 Versions . . . 387
161.3 Commands . . . 387
161.4 Module . . . 387
161.5 Example job . . . 387

162GATK4 **389**
162.1 Introduction . . . 389
162.2 Versions . . . 389
162.3 Commands . . . 389
162.4 Module . . . 389
162.5 Example job . . . 390

163Gemma **391**
163.1 Introduction . . . 391

163.2 Versions . . . 391
163.3 Commands . . . 391
163.4 Module . . . 391
163.5 Example job . . . 391

164Gemoma **393**
164.1 Introduction . . . 393
164.2 Versions . . . 393
164.3 Commands . . . 393
164.4 Module . . . 393
164.5 Example job . . . 394

165GeneMark-ES/ET/EP **395**
165.1 Introduction . . . 395
165.2 Versions . . . 395
165.3 Commands . . . 395
165.4 Academic license . . . 396
165.5 Module . . . 396
165.6 Example job . . . 396

166Genemarks-2 **399**
166.1 Introduction . . . 399
166.2 Versions . . . 399
166.3 Commands . . . 399
166.4 Module . . . 399
166.5 Example job . . . 400

167Genmap **401**
167.1 Introduction . . . 401
167.2 Versions . . . 401
167.3 Commands . . . 401
167.4 Module . . . 401
167.5 Example job . . . 402

168Genomepy **403**
168.1 Introduction . . . 403
168.2 Versions . . . 403
168.3 Commands . . . 403
168.4 Module . . . 403
168.5 Example job . . . 403

169Genomescope2 **405**
169.1 Introduction . . . 405
169.2 Versions . . . 405
169.3 Commands . . . 405
169.4 Module . . . 405
169.5 Example job . . . 405

170Genomicconsensus **407**
170.1 Introduction . . . 407
170.2 Versions . . . 407
170.3 Commands . . . 407
170.4 Module . . . 407
170.5 Example job . . . 407

171Genrich **409**
171.1 Introduction . . . 409
171.2 Versions . . . 409
171.3 Commands . . . 409
171.4 Module . . . 409
171.5 Example job . . . 409

172Gfastats **411**
172.1 Introduction . . . 411
172.2 Versions . . . 411
172.3 Commands . . . 411
172.4 Module . . . 411
172.5 Example job . . . 412

173Gffcompare **413**
173.1 Introduction . . . 413
173.2 Versions . . . 413
173.3 Commands . . . 413
173.4 Module . . . 413
173.5 Example job . . . 413

174Gffread **415**
174.1 Introduction . . . 415
174.2 Versions . . . 415
174.3 Commands . . . 415
174.4 Module . . . 415
174.5 Example job . . . 415

175Gimmemotifs **417**
175.1 Introduction . . . 417
175.2 Versions . . . 417
175.3 Commands . . . 417
175.4 Module . . . 417
175.5 Example job . . . 418

176Glimmer **419**
176.1 Introduction . . . 419
176.2 Versions . . . 419
176.3 Commands . . . 419
176.4 Module . . . 420
176.5 Example job . . . 420

177Glimmerhmm **421**
177.1 Introduction . . . 421
177.2 Versions . . . 421
177.3 Commands . . . 421
177.4 Module . . . 421
177.5 Example job . . . 422

178Glnexus **423**
178.1 Introduction . . . 423
178.2 Versions . . . 423
178.3 Commands . . . 423
178.4 Module . . . 423
178.5 Example job . . . 424

179Gmap **425**
179.1 Introduction . . . 425
179.2 Versions . . . 425
179.3 Commands . . . 425
179.4 Module . . . 427
179.5 Example job . . . 427

180goatools **429**
180.1 Introduction . . . 429
180.2 Versions . . . 429
180.3 Commands . . . 429
180.4 Module . . . 430
180.5 Interactive job . . . 430
180.6 Batch job . . . 430

181Graphlan **431**
181.1 Introduction . . . 431
181.2 Versions . . . 431
181.3 Commands . . . 431
181.4 Module . . . 431
181.5 Example job . . . 432

182Graphmap **433**
182.1 Introduction . . . 433
182.2 Versions . . . 433
182.3 Commands . . . 433
182.4 Module . . . 433
182.5 Example job . . . 433

183Gridss **435**
183.1 Introduction . . . 435
183.2 Versions . . . 435
183.3 Commands . . . 435
183.4 Module . . . 436
183.5 Example job . . . 436

184Gseapy **437**
184.1 Introduction . . . 437
184.2 Versions . . . 437
184.3 Commands . . . 437
184.4 Module . . . 437
184.5 Example job . . . 438

185GTDB-Tk **439**
185.1 Introduction . . . 439
185.2 Versions . . . 439
185.3 Commands . . . 439
185.4 Module . . . 439
185.5 Example job . . . 439

186Gubbins **441**
186.1 Introduction . . . 441
186.2 Versions . . . 441
186.3 Commands . . . 441
186.4 Module . . . 441

186.5 Example job . . . 442

187Guppy **443**
187.1 Introduction . . . 443
187.2 Versions . . . 443
187.3 Commands . . . 443
187.4 Module . . . 444
187.5 Example job . . . 444

188Hail **445**
188.1 Introduction . . . 445
188.2 Versions . . . 445
188.3 Commands . . . 445
188.4 Module . . . 445
188.5 Interactive job . . . 446
188.6 Batch job . . . 446

189Hap.py **447**
189.1 Introduction . . . 447
189.2 Versions . . . 447
189.3 Commands . . . 447
189.4 Module . . . 448
189.5 Example job . . . 448

190Helen **449**
190.1 Introduction . . . 449
190.2 Versions . . . 449
190.3 Commands . . . 449
190.4 Module . . . 449
190.5 Example job . . . 449

191Hicexplorer **451**
191.1 Introduction . . . 451
191.2 Versions . . . 451
191.3 Commands . . . 451
191.4 Module . . . 453
191.5 Example job . . . 453

192Hifiasm **455**
192.1 Introduction . . . 455
192.2 Versions . . . 455
192.3 Commands . . . 455
192.4 Module . . . 455
192.5 Example job . . . 455

193HISAT2 **457**
193.1 Introduction . . . 457
193.2 Versions . . . 457
193.3 Commands . . . 457
193.4 Module . . . 458
193.5 Example job . . . 458

194Hmmer **459**
194.1 Introduction . . . 459
194.2 Versions . . . 459

194.3 Commands . . . 459
194.4 Module . . . 460
194.5 Example job . . . 461

195HOMMER **463**
195.1 Introduction . . . 463
195.2 Versions . . . 463
195.3 Commands . . . 463
195.4 Database . . . 467
195.5 Module . . . 467
195.6 Example job . . . 467

196How_are_we_stranded_here **469**
196.1 Introduction . . . 469
196.2 Versions . . . 469
196.3 Commands . . . 469
196.4 Module . . . 469
196.5 Example job . . . 469

197HTSeq **471**
197.1 Introduction . . . 471
197.2 Versions . . . 471
197.3 Commands . . . 471
197.4 Module . . . 472
197.5 Example job . . . 472

198Htslib **473**
198.1 Introduction . . . 473
198.2 Versions . . . 473
198.3 Commands . . . 473
198.4 Module . . . 473
198.5 Example job . . . 474

199Htstream **475**
199.1 Introduction . . . 475
199.2 Versions . . . 475
199.3 Commands . . . 475
199.4 Module . . . 476
199.5 Example job . . . 476

200HUMAnN 3 **477**
200.1 Introduction . . . 477
200.2 Versions . . . 477
200.3 Commands . . . 477
200.4 Database . . . 478
200.5 Module . . . 478
200.6 Example job . . . 478

201Hyphy **479**
201.1 Introduction . . . 479
201.2 Versions . . . 479
201.3 Commands . . . 479
201.4 Module . . . 479
201.5 Example job . . . 479

202Idba **481**
202.1 Introduction . . . 481
202.2 Versions . . . 481
202.3 Commands . . . 481
202.4 Module . . . 482
202.5 Example job . . . 482

203IGV **485**
203.1 Introduction . . . 485
203.2 Versions . . . 485
203.3 Commands . . . 485
203.4 Module . . . 485
203.5 Interactive job . . . 486

204Impute2 **487**
204.1 Introduction . . . 487
204.2 Versions . . . 487
204.3 Commands . . . 487
204.4 Module . . . 487
204.5 Example job . . . 487

205Instrain **489**
205.1 Introduction . . . 489
205.2 Versions . . . 489
205.3 Commands . . . 489
205.4 Module . . . 489
205.5 Example job . . . 490

206Intarna **491**
206.1 Introduction . . . 491
206.2 Versions . . . 491
206.3 Commands . . . 491
206.4 Module . . . 491
206.5 Example job . . . 491

207InterProScan **493**
207.1 Introduction . . . 493
207.2 Versions . . . 493
207.3 Commands . . . 493
207.4 Database . . . 493
207.5 Module . . . 493
207.6 Example job . . . 494

208IQ-TREE **495**
208.1 Introduction . . . 495
208.2 Versions . . . 495
208.3 Commands . . . 495
208.4 Module . . . 495
208.5 Example job . . . 496

209Isoseq3 **497**
209.1 Introduction . . . 497
209.2 Versions . . . 497
209.3 Commands . . . 497
209.4 Module . . . 497

209.5 Example job . . . 497

210Ivar **499**
210.1 Introduction . . . 499
210.2 Versions . . . 499
210.3 Commands . . . 499
210.4 Module . . . 499
210.5 Example job . . . 499

211Jcvi **501**
211.1 Introduction . . . 501
211.2 Versions . . . 501
211.3 Commands . . . 501
211.4 Module . . . 501
211.5 Example job . . . 502

212Kaiju **503**
212.1 Introduction . . . 503
212.2 Versions . . . 503
212.3 Commands . . . 503
212.4 Module . . . 504
212.5 Example job . . . 504

213Kallisto **505**
213.1 Introduction . . . 505
213.2 Versions . . . 505
213.3 Commands . . . 505
213.4 Module . . . 505
213.5 Example job . . . 505

214Khmer **507**
214.1 Introduction . . . 507
214.2 Versions . . . 507
214.3 Commands . . . 507
214.4 Module . . . 508
214.5 Example job . . . 508

215Kma **509**
215.1 Introduction . . . 509
215.2 Versions . . . 509
215.3 Commands . . . 509
215.4 Module . . . 509
215.5 Example job . . . 510

216Kmc **511**
216.1 Introduction . . . 511
216.2 Versions . . . 511
216.3 Commands . . . 511
216.4 Module . . . 511
216.5 Example job . . . 512

217Jellyfish **513**
217.1 Introduction . . . 513
217.2 Versions . . . 513
217.3 Commands . . . 513

217.4 Module . . . 513
217.5 Example job . . . 513

218KneadData **515**
218.1 Introduction . . . 515
218.2 Versions . . . 515
218.3 Commands . . . 515
218.4 Module . . . 515
218.5 Example job . . . 516

219Kover **517**
219.1 Introduction . . . 517
219.2 Versions . . . 517
219.3 Commands . . . 517
219.4 Module . . . 517
219.5 Example job . . . 518

220Kraken2 **519**
220.1 Introduction . . . 519
220.2 Versions . . . 519
220.3 Commands . . . 519
220.4 Module . . . 519
220.5 Example job . . . 519

221KrakenTools **521**
221.1 Introduction . . . 521
221.2 Versions . . . 521
221.3 Commands . . . 521
221.4 Module . . . 522
221.5 Example job . . . 522

222Lambda **523**
222.1 Introduction . . . 523
222.2 Versions . . . 523
222.3 Commands . . . 523
222.4 Module . . . 523
222.5 Example job . . . 523

223Last **525**
223.1 Introduction . . . 525
223.2 Versions . . . 525
223.3 Commands . . . 525
223.4 Module . . . 526
223.5 Example job . . . 526

224Ldsc **527**
224.1 Introduction . . . 527
224.2 Versions . . . 527
224.3 Commands . . . 527
224.4 Module . . . 527
224.5 Example job . . . 528

225Liftoff **529**
225.1 Introduction . . . 529
225.2 Versions . . . 529

225.3 Commands . . . 529
225.4 Module . . . 529
225.5 Example job . . . 530

226Lima **531**
226.1 Introduction . . . 531
226.2 Versions . . . 531
226.3 Commands . . . 531
226.4 Module . . . 531
226.5 Example job . . . 531

227Links **533**
227.1 Introduction . . . 533
227.2 Versions . . . 533
227.3 Commands . . . 533
227.4 Module . . . 533
227.5 Example job . . . 534

228Lofreq **535**
228.1 Introduction . . . 535
228.2 Versions . . . 535
228.3 Commands . . . 535
228.4 Module . . . 535
228.5 Example job . . . 535

229Longqc **537**
229.1 Introduction . . . 537
229.2 Versions . . . 537
229.3 Commands . . . 537
229.4 Module . . . 537
229.5 Example job . . . 538

230Lra **539**
230.1 Introduction . . . 539
230.2 Versions . . . 539
230.3 Commands . . . 539
230.4 Module . . . 539
230.5 Example job . . . 539

231Ltr_finder **541**
231.1 Introduction . . . 541
231.2 Versions . . . 541
231.3 Commands . . . 541
231.4 Module . . . 541
231.5 Example job . . . 542

232Ltrpred **543**
232.1 Introduction . . . 543
232.2 Versions . . . 543
232.3 Commands . . . 543
232.4 Module . . . 543
232.5 Example job . . . 544

233Lumpy-sv **545**
233.1 Introduction . . . 545

233.2 Versions . . . 545
233.3 Commands . . . 545
233.4 Module . . . 545
233.5 Example job . . . 546

234Lyveset **547**
234.1 Introduction . . . 547
234.2 Versions . . . 547
234.3 Commands . . . 547
234.4 Module . . . 548
234.5 Example job . . . 549

235MACS2 **551**
235.1 Introduction . . . 551
235.2 Versions . . . 551
235.3 Commands . . . 551
235.4 Module . . . 551
235.5 Example job . . . 551

236Macs3 **553**
236.1 Introduction . . . 553
236.2 Versions . . . 553
236.3 Commands . . . 553
236.4 Module . . . 553
236.5 Example job . . . 553

237MAFFT **555**
237.1 Introduction . . . 555
237.2 Versions . . . 555
237.3 Commands . . . 555
237.4 Module . . . 556
237.5 Example job . . . 556

238Mageck **557**
238.1 Introduction . . . 557
238.2 Versions . . . 557
238.3 Commands . . . 557
238.4 Module . . . 557
238.5 Example job . . . 558

239MAKER **559**
239.1 Introduction . . . 559
239.2 Versions . . . 559
239.3 Commands . . . 559
239.4 Module . . . 560
239.5 Prerequisites . . . 560
239.6 Example job non-mpi . . . 561
239.7 Example job mpi . . . 561

240Manta **563**
240.1 Introduction . . . 563
240.2 Versions . . . 563
240.3 Commands . . . 563
240.4 Module . . . 563
240.5 Example job . . . 563

241Mapcaller **565**
241.1 Introduction . . . 565
241.2 Versions . . . 565
241.3 Commands . . . 565
241.4 Module . . . 565
241.5 Example job . . . 565

242Marginpolish **567**
242.1 Introduction . . . 567
242.2 Versions . . . 567
242.3 Commands . . . 567
242.4 Module . . . 567
242.5 Example job . . . 568

243Mash **569**
243.1 Introduction . . . 569
243.2 Versions . . . 569
243.3 Commands . . . 569
243.4 Module . . . 569
243.5 Example job . . . 569

244Mashmap **571**
244.1 Introduction . . . 571
244.2 Versions . . . 571
244.3 Commands . . . 571
244.4 Module . . . 571
244.5 Example job . . . 571

245Mashtree **573**
245.1 Introduction . . . 573
245.2 Versions . . . 573
245.3 Commands . . . 573
245.4 Module . . . 573
245.5 Example job . . . 574

246Mauve **575**
246.1 Introduction . . . 575
246.2 Versions . . . 575
246.3 Commands . . . 575
246.4 Module . . . 575
246.5 Example job . . . 576

247Maxbin2 **577**
247.1 Introduction . . . 577
247.2 Versions . . . 577
247.3 Commands . . . 577
247.4 Module . . . 577
247.5 Example job . . . 578

248Maxquant **579**
248.1 Introduction . . . 579
248.2 Versions . . . 579
248.3 Commands . . . 579
248.4 Module . . . 579
248.5 GUI . . . 580

248.6 CMD job . . . 580

249Mcl **583**
249.1 Introduction . . . 583
249.2 Versions . . . 583
249.3 Commands . . . 583
249.4 Module . . . 584
249.5 Example job . . . 584

250Mcscanx **585**
250.1 Introduction . . . 586
250.2 Versions . . . 586
250.3 Commands . . . 586
250.4 Module . . . 586
250.5 Helper command . . . 587
250.6 Example job . . . 587

251Medaka **589**
251.1 Introduction . . . 589
251.2 Versions . . . 589
251.3 Commands . . . 589
251.4 Module . . . 589
251.5 Example job . . . 590

252Megadepth **591**
252.1 Introduction . . . 591
252.2 Versions . . . 591
252.3 Commands . . . 591
252.4 Module . . . 591
252.5 Example job . . . 591

253Megahit **593**
253.1 Introduction . . . 593
253.2 Versions . . . 593
253.3 Commands . . . 593
253.4 Module . . . 593
253.5 Example job . . . 593

254Megan **595**
254.1 Introduction . . . 595
254.2 Versions . . . 595
254.3 Commands . . . 595
254.4 Module . . . 596
254.5 GUI . . . 596
254.6 Example job . . . 597

255Meme **599**
255.1 Introduction . . . 599
255.2 Versions . . . 599
255.3 Commands . . . 599
255.4 Module . . . 600
255.5 Example job . . . 600

256Merqury **601**
256.1 Introduction . . . 601

256.2 Versions . . . 601
256.3 Commands . . . 601
256.4 Module . . . 601
256.5 Example job . . . 602

257Meryl **603**
257.1 Introduction . . . 603
257.2 Versions . . . 603
257.3 Commands . . . 603
257.4 Module . . . 603
257.5 Example job . . . 604

258Metabat **605**
258.1 Introduction . . . 605
258.2 Versions . . . 605
258.3 Commands . . . 605
258.4 Module . . . 606
258.5 Example job . . . 606

259MetaPhlAn 3 **607**
259.1 Introduction . . . 607
259.2 Versions . . . 607
259.3 Commands . . . 607
259.4 Database . . . 608
259.5 Module . . . 608
259.6 Example job . . . 608

260Methyldackel **609**
260.1 Introduction . . . 609
260.2 Versions . . . 609
260.3 Commands . . . 609
260.4 Module . . . 609
260.5 Example job . . . 610

261Metilene **611**
261.1 Introduction . . . 611
261.2 Versions . . . 611
261.3 Commands . . . 611
261.4 Module . . . 611
261.5 Example job . . . 612

262Mhm2 **613**
262.1 Introduction . . . 613
262.2 Versions . . . 613
262.3 Commands . . . 613
262.4 Module . . . 613
262.5 Example job . . . 614

263MicrobeDMM **615**
263.1 Introduction . . . 615
263.2 Versions . . . 615
263.3 Commands . . . 615
263.4 Module . . . 615
263.5 Example job . . . 615

264Minialign **617**
264.1 Introduction . . . 617
264.2 Versions . . . 617
264.3 Commands . . . 617
264.4 Module . . . 617
264.5 Example job . . . 617

265Miniasm **619**
265.1 Introduction . . . 619
265.2 Versions . . . 619
265.3 Commands . . . 619
265.4 Module . . . 619
265.5 Example job . . . 619

266Minimap2 **621**
266.1 Introduction . . . 621
266.2 Versions . . . 621
266.3 Commands . . . 621
266.4 Module . . . 621
266.5 Example job . . . 622

267Minipolish **623**
267.1 Introduction . . . 623
267.2 Versions . . . 623
267.3 Commands . . . 623
267.4 Module . . . 623
267.5 Example job . . . 624

268Miniprot **625**
268.1 Introduction . . . 625
268.2 Versions . . . 625
268.3 Commands . . . 625
268.4 Module . . . 625
268.5 Example job . . . 626

269miRDeep2 **627**
269.1 Introduction . . . 627
269.2 Versions . . . 627
269.3 Commands . . . 627
269.4 Module . . . 628
269.5 Example job . . . 628

270Mirtop **631**
270.1 Introduction . . . 631
270.2 Versions . . . 631
270.3 Commands . . . 631
270.4 Module . . . 631
270.5 Example job . . . 631

271Mitofinder **633**
271.1 Introduction . . . 633
271.2 Versions . . . 633
271.3 Commands . . . 633
271.4 Module . . . 633
271.5 Example job . . . 633

272Mlst **635**
272.1 Introduction . . . 635
272.2 Versions . . . 635
272.3 Commands . . . 635
272.4 Module . . . 635
272.5 Example job . . . 636

273Mmseqs2 **637**
273.1 Introduction . . . 637
273.2 Versions . . . 637
273.3 Commands . . . 637
273.4 Module . . . 637
273.5 Example job . . . 637

274Mothur **639**
274.1 Introduction . . . 639
274.2 Versions . . . 639
274.3 Commands . . . 639
274.4 Module . . . 639
274.5 Interactive job . . . 640
274.6 Batch job . . . 641

275MrBayes **643**
275.1 Introduction . . . 643
275.2 Versions . . . 643
275.3 Commands . . . 643
275.4 Module . . . 644
275.5 Example job . . . 644

276Multiqc **645**
276.1 Introduction . . . 645
276.2 Versions . . . 645
276.3 Commands . . . 645
276.4 Module . . . 645
276.5 Example job . . . 645

277Mummer4 **647**
277.1 Introduction . . . 647
277.2 Versions . . . 647
277.3 Commands . . . 647
277.4 Module . . . 648
277.5 Example job . . . 648

278Muscle **649**
278.1 Introduction . . . 649
278.2 Versions . . . 649
278.3 Versions . . . 649
278.4 Commands . . . 649
278.5 Module . . . 649
278.6 Example job . . . 650

279Mutmap **651**
279.1 Introduction . . . 651
279.2 Versions . . . 651
279.3 Commands . . . 651

279.4 Module . . . 651
279.5 Example job . . . 652

280Mykrobe **653**
280.1 Introduction . . . 653
280.2 Versions . . . 653
280.3 Commands . . . 653
280.4 Module . . . 653
280.5 Example job . . . 653

281Nanofilt **655**
281.1 Introduction . . . 655
281.2 Versions . . . 655
281.3 Commands . . . 655
281.4 Module . . . 655
281.5 Example job . . . 655

282Nanolyse **657**
282.1 Introduction . . . 657
282.2 Versions . . . 657
282.3 Commands . . . 657
282.4 Module . . . 657
282.5 Example job . . . 657

283Nanoplot **659**
283.1 Introduction . . . 659
283.2 Versions . . . 659
283.3 Commands . . . 659
283.4 Module . . . 659
283.5 Example job . . . 659

284Nanopolish **661**
284.1 Introduction . . . 661
284.2 Versions . . . 661
284.3 Commands . . . 661
284.4 Module . . . 661
284.5 Example job . . . 662

285Ncbi-amrfinderplus **663**
285.1 Introduction . . . 663
285.2 Versions . . . 663
285.3 Commands . . . 663
285.4 Module . . . 663
285.5 Example job . . . 664

286Ncbi-genome-download **665**
286.1 Introduction . . . 665
286.2 Versions . . . 665
286.3 Commands . . . 665
286.4 Module . . . 665
286.5 Example job . . . 665

287Neusomatic **667**
287.1 Introduction . . . 667
287.2 Versions . . . 667

287.3 Commands . . . 667
287.4 Module . . . 668
287.5 Example job . . . 668

288Nextalign **669**
288.1 Introduction . . . 669
288.2 Versions . . . 669
288.3 Commands . . . 669
288.4 Module . . . 669
288.5 Example job . . . 669

289Nextclade **671**
289.1 Introduction . . . 671
289.2 Versions . . . 671
289.3 Commands . . . 671
289.4 Module . . . 671
289.5 Example job . . . 671

290Nextflow **673**
290.1 Introduction . . . 673
290.2 Versions . . . 673
290.3 Commands . . . 673
290.4 Module . . . 673
290.5 Example job . . . 673

291Ngs-bits **675**
291.1 Introduction . . . 675
291.2 Versions . . . 675
291.3 Commands . . . 675
291.4 Module . . . 678
291.5 Example job . . . 678

292Ngsutils **679**
292.1 Introduction . . . 679
292.2 Versions . . . 679
292.3 Commands . . . 679
292.4 Module . . . 679
292.5 Example job . . . 680

293OrthoFinder **681**
293.1 Introduction . . . 681
293.2 Versions . . . 681
293.3 Commands . . . 681
293.4 Module . . . 681
293.5 Example job . . . 681

294Paml **683**
294.1 Introduction . . . 683
294.2 Versions . . . 683
294.3 Commands . . . 683
294.4 Module . . . 684
294.5 Example job . . . 684

295Panacota **685**
295.1 Introduction . . . 685

295.2 Versions . . . 685
295.3 Commands . . . 685
295.4 Module . . . 685
295.5 Example job . . . 685

296Panaroo **687**
296.1 Introduction . . . 687
296.2 Versions . . . 687
296.3 Commands . . . 687
296.4 Module . . . 688
296.5 Example job . . . 688

297Pandaseq **689**
297.1 Introduction . . . 689
297.2 Versions . . . 689
297.3 Commands . . . 689
297.4 Module . . . 689
297.5 Example job . . . 689

298Pandora **691**
298.1 Introduction . . . 691
298.2 Versions . . . 691
298.3 Commands . . . 691
298.4 Module . . . 691
298.5 Example job . . . 692

299Pangolin **693**
299.1 Introduction . . . 693
299.2 Versions . . . 693
299.3 Commands . . . 693
299.4 Module . . . 693
299.5 Example job . . . 694

300PanPhlAn **695**
300.1 Introduction . . . 695
300.2 Versions . . . 695
300.3 Commands . . . 695
300.4 Module . . . 695
300.5 Example job . . . 695

301Clara Parabricks **697**
301.1 Introduction . . . 697
301.2 Versions . . . 697
301.3 Commands . . . 697
301.4 Module . . . 697
301.5 Example job . . . 698

302Parallel-fastq-dump **699**
302.1 Introduction . . . 699
302.2 Versions . . . 699
302.3 Commands . . . 699
302.4 Module . . . 699
302.5 Example job . . . 699

303Parliament2 **701**

303.1 Introduction . . . 701
303.2 Versions . . . 701
303.3 Commands . . . 701
303.4 Module . . . 701
303.5 Example job . . . 702

304Parsnp **703**
304.1 Introduction . . . 703
304.2 Versions . . . 703
304.3 Commands . . . 703
304.4 Module . . . 703
304.5 Example job . . . 703

305Pbmm2 **705**
305.1 Introduction . . . 705
305.2 Versions . . . 705
305.3 Commands . . . 705
305.4 Module . . . 705
305.5 Example job . . . 705

306Pbptyper **707**
306.1 Introduction . . . 707
306.2 Versions . . . 707
306.3 Commands . . . 707
306.4 Module . . . 707
306.5 Example job . . . 708

307PCAngsd **709**
307.1 Introduction . . . 709
307.2 Versions . . . 709
307.3 Commands . . . 709
307.4 Module . . . 709
307.5 Example job . . . 709

308Peakranger **711**
308.1 Introduction . . . 711
308.2 Versions . . . 711
308.3 Commands . . . 711
308.4 Module . . . 711
308.5 Example job . . . 711

309Pepper_deepvariant **713**
309.1 Introduction . . . 713
309.2 Versions . . . 713
309.3 Commands . . . 713
309.4 Module . . . 713
309.5 Example job . . . 714

310BioPerl **717**
310.1 Introduction . . . 717
310.2 Versions . . . 717
310.3 Commands . . . 717
310.4 Module . . . 722
310.5 Example job . . . 722

311Phd2fasta **723**
311.1 Introduction . . . 723
311.2 Versions . . . 723
311.3 Commands . . . 723
311.4 Module . . . 723
311.5 Example job . . . 723

312Phg **725**
312.1 Introduction . . . 725
312.2 Versions . . . 725
312.3 Commands . . . 725
312.4 Module . . . 726
312.5 Example job . . . 726

313phrap **727**
313.1 Introduction . . . 727
313.2 Versions . . . 727
313.3 Commands . . . 727
313.4 Module . . . 727
313.5 Example job . . . 727

314phred **729**
314.1 Introduction . . . 729
314.2 Versions . . . 729
314.3 Commands . . . 729
314.4 Module . . . 729
314.5 Example job . . . 729

315Picard Tools **731**
315.1 Introduction . . . 731
315.2 Versions . . . 731
315.3 Commands . . . 731
315.4 Module . . . 731
315.5 Example job . . . 731

316Picrust2 **733**
316.1 Introduction . . . 733
316.2 Versions . . . 733
316.3 Commands . . . 733
316.4 Module . . . 734
316.5 Example job . . . 734

317Pilon **737**
317.1 Introduction . . . 737
317.2 Versions . . . 737
317.3 Commands . . . 737
317.4 Module . . . 737
317.5 Example job . . . 737

318Pindel **739**
318.1 Introduction . . . 739
318.2 Versions . . . 739
318.3 Commands . . . 739
318.4 Module . . . 739
318.5 Example job . . . 740

319Pirate **741**
319.1 Introduction . . . 741
319.2 Versions . . . 741
319.3 Commands . . . 741
319.4 Module . . . 745
319.5 Example job . . . 745

320Pixy **747**
320.1 Introduction . . . 747
320.2 Versions . . . 747
320.3 Commands . . . 747
320.4 Module . . . 747
320.5 Example job . . . 747

321Plasmidfinder **749**
321.1 Introduction . . . 749
321.2 Versions . . . 749
321.3 Commands . . . 749
321.4 Module . . . 749
321.5 Example job . . . 750

322Platypus **751**
322.1 Introduction . . . 751
322.2 Versions . . . 751
322.3 Commands . . . 751
322.4 Module . . . 751
322.5 Example job . . . 751

323Plink **753**
323.1 Introduction . . . 753
323.2 Versions . . . 753
323.3 Commands . . . 753
323.4 Module . . . 753
323.5 Example job . . . 754

324Plink2 **755**
324.1 Introduction . . . 755
324.2 Versions . . . 755
324.3 Commands . . . 755
324.4 Module . . . 755
324.5 Example job . . . 755

325Plotsr **757**
325.1 Introduction . . . 757
325.2 Versions . . . 757
325.3 Commands . . . 757
325.4 Module . . . 757
325.5 Example job . . . 758

326Pomoxis **759**
326.1 Introduction . . . 759
326.2 Versions . . . 759
326.3 Commands . . . 759
326.4 Module . . . 760
326.5 Example job . . . 760

327Popscle **761**
327.1 Introduction . . . 761
327.2 Versions . . . 761
327.3 Commands . . . 761
327.4 Module . . . 761
327.5 Example job . . . 761

328Prinseq **763**
328.1 Introduction . . . 763
328.2 Versions . . . 763
328.3 Commands . . . 763
328.4 Module . . . 763
328.5 Example job . . . 764

329Prodigal **765**
329.1 Introduction . . . 765
329.2 Versions . . . 765
329.3 Commands . . . 765
329.4 Module . . . 765
329.5 Example job . . . 765

330Prokka **767**
330.1 Introduction . . . 767
330.2 Versions . . . 767
330.3 Commands . . . 767
330.4 Module . . . 768
330.5 Example job . . . 768

331Proteinortho **769**
331.1 Introduction . . . 769
331.2 Versions . . . 769
331.3 Commands . . . 769
331.4 Module . . . 770
331.5 Example job . . . 770

332ProtHint **771**
332.1 Introduction . . . 771
332.2 Versions . . . 771
332.3 Commands . . . 771
332.4 Academic license . . . 772
332.5 Module . . . 772
332.6 Example job . . . 772

333Pullseq **773**
333.1 Introduction . . . 773
333.2 Versions . . . 773
333.3 Commands . . . 773
333.4 Module . . . 773
333.5 Example job . . . 774

334Pvactools **775**
334.1 Introduction . . . 775
334.2 Versions . . . 775
334.3 Commands . . . 775
334.4 Module . . . 776

334.5 Example job . . . 776

335Pyani **777**
335.1 Introduction . . . 777
335.2 Versions . . . 777
335.3 Commands . . . 777
335.4 Module . . . 777
335.5 Example job . . . 778

336Pybedtools **779**
336.1 Introduction . . . 779
336.2 Versions . . . 779
336.3 Commands . . . 779
336.4 Module . . . 779
336.5 Example job . . . 780

337Pybigwig **781**
337.1 Introduction . . . 781
337.2 Versions . . . 781
337.3 Commands . . . 781
337.4 Module . . . 781
337.5 Interactive job . . . 782
337.6 Batch job . . . 782

338Pychopper **783**
338.1 Introduction . . . 783
338.2 Versions . . . 783
338.3 Commands . . . 783
338.4 Module . . . 783
338.5 Example job . . . 784

339Pycoqc **785**
339.1 Introduction . . . 785
339.2 Versions . . . 785
339.3 Commands . . . 785
339.4 Module . . . 785
339.5 Example job . . . 786

340Pyensembl **787**
340.1 Introduction . . . 787
340.2 Versions . . . 787
340.3 Commands . . . 787
340.4 Module . . . 787
340.5 Example job . . . 787

341Pyfaidx **789**
341.1 Introduction . . . 789
341.2 Versions . . . 789
341.3 Commands . . . 789
341.4 Module . . . 789
341.5 Example job . . . 789

342Pygenometracks **791**
342.1 Introduction . . . 791
342.2 Versions . . . 791

342.3 Commands . . . 791
342.4 Module . . . 791
342.5 Example job . . . 792

343Pygenomeviz **793**
343.1 Introduction . . . 793
343.2 Versions . . . 793
343.3 Commands . . . 793
343.4 Module . . . 793
343.5 Example job . . . 794

344Pyranges **795**
344.1 Introduction . . . 795
344.2 Versions . . . 795
344.3 Commands . . . 795
344.4 Module . . . 795
344.5 Example job . . . 796

345Pysam **797**
345.1 Introduction . . . 797
345.2 Versions . . . 797
345.3 Commands . . . 797
345.4 Module . . . 797
345.5 Example job . . . 798

346QIIME 2 **799**
346.1 Introduction . . . 799
346.2 Versions . . . 799
346.3 Commands . . . 799
346.4 Module . . . 799
346.5 Example job . . . 800

347Qualimap **801**
347.1 Introduction . . . 801
347.2 Versions . . . 801
347.3 Commands . . . 801
347.4 Module . . . 801
347.5 Example job . . . 802

348Quast **803**
348.1 Introduction . . . 803
348.2 Versions . . . 803
348.3 Commands . . . 803
348.4 Module . . . 803
348.5 Example job . . . 804

349QuickMIRSeq **805**
349.1 Introduction . . . 805
349.2 Versions . . . 805
349.3 Commands . . . 805
349.4 Module . . . 805
349.5 Example job . . . 806

350R **807**
350.1 Introduction . . . 807

350.2 Versions . . . 807
350.3 Commands . . . 807
350.4 Module . . . 807
350.5 Example job . . . 808

351Racon **809**
351.1 Introduction . . . 809
351.2 Versions . . . 809
351.3 Commands . . . 809
351.4 Module . . . 809
351.5 Example job . . . 809

352Ragout **811**
352.1 Introduction . . . 811
352.2 Versions . . . 811
352.3 Commands . . . 811
352.4 Module . . . 811
352.5 Example job . . . 811

353Ragtag **813**
353.1 Introduction . . . 813
353.2 Versions . . . 813
353.3 Commands . . . 813
353.4 Module . . . 813
353.5 Example job . . . 813

354Rapmap **815**
354.1 Introduction . . . 815
354.2 Versions . . . 815
354.3 Commands . . . 815
354.4 Module . . . 815
354.5 Example job . . . 816

355Rasusa **817**
355.1 Introduction . . . 817
355.2 Versions . . . 817
355.3 Commands . . . 817
355.4 Module . . . 817
355.5 Example job . . . 818

356Raven-assembler **819**
356.1 Introduction . . . 819
356.2 Versions . . . 819
356.3 Commands . . . 819
356.4 Module . . . 819
356.5 Example job . . . 819

357Raxml **821**
357.1 Introduction . . . 821
357.2 Versions . . . 821
357.3 Commands . . . 821
357.4 Module . . . 821
357.5 Example job . . . 822

358Raxml-ng **823**

358.1 Introduction . . . 823
358.2 Versions . . . 823
358.3 Commands . . . 823
358.4 Module . . . 823
358.5 Example job . . . 824

359Rebaler **825**
359.1 Introduction . . . 825
359.2 Versions . . . 825
359.3 Commands . . . 825
359.4 Module . . . 825
359.5 Example job . . . 825

360Reciprocal Smallest Distance **827**
360.1 Introduction . . . 827
360.2 Versions . . . 827
360.3 Commands . . . 827
360.4 Module . . . 827
360.5 Example job . . . 827

361Recycler **829**
361.1 Introduction . . . 829
361.2 Versions . . . 829
361.3 Commands . . . 829
361.4 Module . . . 829
361.5 Example job . . . 830

362RepeatMasker **831**
362.1 Introduction . . . 831
362.2 Versions . . . 831
362.3 Commands . . . 831
362.4 Database . . . 831
362.5 Species name . . . 831
362.6 Module . . . 832
362.7 Example job . . . 832

363RepeatModeler **833**
363.1 Introduction . . . 833
363.2 Versions . . . 833
363.3 Commands . . . 833
363.4 Module . . . 833
363.5 Example job . . . 834

364RepeatScout **835**
364.1 Introduction . . . 835
364.2 Versions . . . 835
364.3 Commands . . . 835
364.4 Module . . . 835
364.5 Example job . . . 836

365Resfinder **837**
365.1 Introduction . . . 837
365.2 Versions . . . 837
365.3 Commands . . . 837
365.4 Module . . . 837

365.5 Example job . . . 838

366Revbayes **839**
366.1 Introduction . . . 839
366.2 Versions . . . 839
366.3 Commands . . . 839
366.4 Module . . . 839
366.5 Example job . . . 840

367rMATS **841**
367.1 Introduction . . . 841
367.2 Versions . . . 841
367.3 Commands . . . 841
367.4 Module . . . 841
367.5 Example job . . . 841

368rmats2sashimiplot **843**
368.1 Introduction . . . 843
368.2 Versions . . . 843
368.3 Commands . . . 843
368.4 Module . . . 843
368.5 Example job . . . 843

369RNAIndel **845**
369.1 Introduction . . . 845
369.2 Versions . . . 845
369.3 Commands . . . 845
369.4 Module . . . 845
369.5 Example job . . . 845

370RNApeg **847**
370.1 Introduction . . . 847
370.2 Versions . . . 847
370.3 Commands . . . 847
370.4 Module . . . 847
370.5 Example job . . . 847

371Rnaquast **849**
371.1 Introduction . . . 849
371.2 Versions . . . 849
371.3 Commands . . . 849
371.4 Module . . . 849
371.5 Example job . . . 849

372Roary **851**
372.1 Introduction . . . 851
372.2 Versions . . . 851
372.3 Commands . . . 851
372.4 Module . . . 851
372.5 Example job . . . 852

373r-rnaseq **853**
373.1 Introduction . . . 853
373.2 Versions . . . 854
373.3 Commands . . . 854

373.4 Module . . . 854
373.5 Install packages . . . 854
373.6 Interactive job . . . 854
373.7 Batch job . . . 855

374RStudio **857**
374.1 Introduction . . . 857
374.2 Versions . . . 857
374.3 Commands . . . 857
374.4 Module . . . 857
374.5 Example job . . . 858

375r-scrnaseq **859**
375.1 Introduction . . . 859
375.2 Versions . . . 860
375.3 Commands . . . 860
375.4 Module . . . 860
375.5 Install packages . . . 861
375.6 Interactive job . . . 861
375.7 Batch job . . . 862

376RSEM **863**
376.1 Introduction . . . 863
376.2 Versions . . . 863
376.3 Commands . . . 863
376.4 Dependencies . . . 864
376.5 Module . . . 864
376.6 Example job . . . 864

377Rseqc **867**
377.1 Introduction . . . 867
377.2 Versions . . . 867
377.3 Commands . . . 867
377.4 Module . . . 871
377.5 Example job . . . 871

378run-dbCAN **873**
378.1 Introduction . . . 873
378.2 Versions . . . 873
378.3 Commands . . . 873
378.4 Database . . . 873
378.5 Module . . . 873
378.6 Example job . . . 874

379rush **875**
379.1 Introduction . . . 875
379.2 Versions . . . 875
379.3 Commands . . . 875
379.4 Module . . . 875
379.5 Example job . . . 875

380Salmon **877**
380.1 Introduction . . . 877
380.2 Versions . . . 877
380.3 Commands . . . 877

380.4 Module . . . 877
380.5 Example job . . . 878

381Sambamba **879**
381.1 Introduction . . . 879
381.2 Versions . . . 879
381.3 Commands . . . 879
381.4 Module . . . 879
381.5 Example job . . . 879

382Samblaster **881**
382.1 Introduction . . . 881
382.2 Versions . . . 881
382.3 Commands . . . 881
382.4 Module . . . 881
382.5 Example job . . . 881

383Samclip **883**
383.1 Introduction . . . 883
383.2 Versions . . . 883
383.3 Commands . . . 883
383.4 Module . . . 883
383.5 Example job . . . 884

384Samplot **885**
384.1 Introduction . . . 885
384.2 Versions . . . 885
384.3 Commands . . . 885
384.4 Module . . . 885
384.5 Example job . . . 885

385Samtools **887**
385.1 Introduction . . . 887
385.2 Versions . . . 887
385.3 Commands . . . 887
385.4 Module . . . 887
385.5 Example job . . . 888

386Scanpy **889**
386.1 Introduction . . . 889
386.2 Versions . . . 889
386.3 Commands . . . 889
386.4 Module . . . 889
386.5 Interactive job . . . 890
386.6 Batch job . . . 890

387Scarches **891**
387.1 Introduction . . . 891
387.2 Versions . . . 891
387.3 Commands . . . 891
387.4 Module . . . 891
387.5 Example job . . . 892

388Scgen **893**
388.1 Introduction . . . 893

388.2 Versions . . . 893
388.3 Commands . . . 893
388.4 Module . . . 893
388.5 Example job . . . 894

389Scirpy **895**
389.1 Introduction . . . 895
389.2 Versions . . . 895
389.3 Commands . . . 895
389.4 Module . . . 895
389.5 Example job . . . 896

390scVelo **897**
390.1 Introduction . . . 897
390.2 Versions . . . 897
390.3 Commands . . . 897
390.4 Module . . . 897
390.5 Interactive job . . . 897
390.6 Batch job . . . 898

391Scvi-tools **899**
391.1 Introduction . . . 899
391.2 Versions . . . 899
391.3 Commands . . . 899
391.4 Module . . . 899
391.5 Example job . . . 900

392Seidr **901**
392.1 Introduction . . . 901
392.2 Versions . . . 901
392.3 Commands . . . 901
392.4 Module . . . 902
392.5 Example job . . . 902

393Sepp **903**
393.1 Introduction . . . 903
393.2 Versions . . . 903
393.3 Commands . . . 903
393.4 Module . . . 903
393.5 Example job . . . 904

394Seqkit **905**
394.1 Introduction . . . 905
394.2 Versions . . . 905
394.3 Commands . . . 905
394.4 Module . . . 905
394.5 Example job . . . 905

395Seqyclean **907**
395.1 Introduction . . . 907
395.2 Versions . . . 907
395.3 Commands . . . 907
395.4 Module . . . 907
395.5 Example job . . . 908

396Shasta **909**
396.1 Introduction . . . 909
396.2 Versions . . . 909
396.3 Commands . . . 909
396.4 Module . . . 909
396.5 Example job . . . 909

397Shorah **911**
397.1 Introduction . . . 911
397.2 Versions . . . 911
397.3 Commands . . . 911
397.4 Module . . . 911
397.5 Example job . . . 912

398Shortstack **913**
398.1 Introduction . . . 913
398.2 Versions . . . 913
398.3 Commands . . . 913
398.4 Module . . . 913
398.5 Example job . . . 913

399Shovill **915**
399.1 Introduction . . . 915
399.2 Versions . . . 915
399.3 Commands . . . 915
399.4 Module . . . 915
399.5 Example job . . . 916

400Sicer **917**
400.1 Introduction . . . 917
400.2 Versions . . . 917
400.3 Commands . . . 917
400.4 Module . . . 917
400.5 Example job . . . 918

401Sicer2 **919**
401.1 Introduction . . . 919
401.2 Versions . . . 919
401.3 Commands . . . 919
401.4 Module . . . 919
401.5 Example job . . . 920

402SignalP **921**
402.1 Introduction . . . 921
402.2 Versions . . . 921
402.3 Commands . . . 921
402.4 Module . . . 921
402.5 Example job . . . 921

403Signalp6 **923**
403.1 Introduction . . . 923
403.2 Versions . . . 923
403.3 Commands . . . 923
403.4 Module . . . 923
403.5 Example job for fast mode . . . 924

403.6 Example job for slow mode . . . 924

404Simug **927**
404.1 Introduction . . . 927
404.2 Versions . . . 927
404.3 Commands . . . 927
404.4 Module . . . 927
404.5 Example job . . . 927

405Skewer **929**
405.1 Introduction . . . 929
405.2 Versions . . . 929
405.3 Commands . . . 929
405.4 Module . . . 929
405.5 Example job . . . 929

406Slamdunk **931**
406.1 Introduction . . . 931
406.2 Versions . . . 931
406.3 Commands . . . 931
406.4 Module . . . 931
406.5 Example job . . . 932

407Smoove **933**
407.1 Introduction . . . 933
407.2 Versions . . . 933
407.3 Commands . . . 933
407.4 Module . . . 933
407.5 Example job . . . 933

408Snakemake **935**
408.1 Introduction . . . 935
408.2 Versions . . . 935
408.3 Commands . . . 935
408.4 Module . . . 935
408.5 Example job . . . 935

409Snap **937**
409.1 Introduction . . . 937
409.2 Versions . . . 937
409.3 Commands . . . 937
409.4 Module . . . 937
409.5 Example job . . . 937

410Snap-aligner **939**
410.1 Introduction . . . 939
410.2 Versions . . . 939
410.3 Commands . . . 939
410.4 Module . . . 939
410.5 Example job . . . 939

411Snaptools **941**
411.1 Introduction . . . 941
411.2 Versions . . . 941
411.3 Commands . . . 941

411.4 Module . . . 941
411.5 Example job . . . 941

412Snippy **943**
412.1 Introduction . . . 943
412.2 Versions . . . 943
412.3 Commands . . . 943
412.4 Module . . . 943
412.5 Example job . . . 944

413Snp-dists **945**
413.1 Introduction . . . 945
413.2 Versions . . . 945
413.3 Commands . . . 945
413.4 Module . . . 945
413.5 Example job . . . 946

414Snpeff **947**
414.1 Introduction . . . 947
414.2 Versions . . . 947
414.3 Commands . . . 947
414.4 Module . . . 947
414.5 Example job . . . 947

415Snpgenie **949**
415.1 Introduction . . . 949
415.2 Versions . . . 949
415.3 Commands . . . 949
415.4 Module . . . 950
415.5 Example job . . . 950

416Snphylo **951**
416.1 Introduction . . . 951
416.2 Versions . . . 951
416.3 Commands . . . 951
416.4 Module . . . 952
416.5 Example job . . . 952

417Snpsift **953**
417.1 Introduction . . . 953
417.2 Versions . . . 953
417.3 Commands . . . 953
417.4 Module . . . 953
417.5 Example job . . . 953

418Snp-sites **955**
418.1 Introduction . . . 955
418.2 Versions . . . 955
418.3 Commands . . . 955
418.4 Module . . . 955
418.5 Example job . . . 956

419Soapdenovo2 **957**
419.1 Introduction . . . 957
419.2 Versions . . . 957

419.3 Commands . . . 957
419.4 Module . . . 957
419.5 Example job . . . 958

420SortMeRNA **959**
420.1 Introduction . . . 959
420.2 Versions . . . 959
420.3 Commands . . . 959
420.4 Module . . . 959
420.5 Example job . . . 959

421Souporcell **961**
421.1 Introduction . . . 961
421.2 Versions . . . 961
421.3 Commands . . . 961
421.4 Module . . . 962
421.5 Example job . . . 962

422Sourmash **963**
422.1 Introduction . . . 963
422.2 Versions . . . 963
422.3 Commands . . . 963
422.4 Module . . . 963
422.5 Example job . . . 964

423Spaceranger **965**
423.1 Introduction . . . 965
423.2 Versions . . . 965
423.3 Commands . . . 965
423.4 Module . . . 965
423.5 Example job . . . 966

424SPAdes **967**
424.1 Introduction . . . 967
424.2 Versions . . . 967
424.3 Commands . . . 967
424.4 Module . . . 968
424.5 Example job . . . 968

425Sprod **969**
425.1 Introduction . . . 969
425.2 Versions . . . 969
425.3 Commands . . . 969
425.4 Module . . . 969
425.5 Example job . . . 970

426Squeezemeta **971**
426.1 Introduction . . . 971
426.2 Versions . . . 971
426.3 Commands . . . 971
426.4 Module . . . 973
426.5 Example job . . . 973

427SRA-Toolkit **975**
427.1 Introduction . . . 975

427.2 Versions . . . 975
427.3 Commands . . . 975
427.4 Module . . . 976
427.5 Configuring SRA-Toolkit . . . 977
427.6 Example job . . . 977

428Srst2 **979**
428.1 Introduction . . . 979
428.2 Versions . . . 979
428.3 Commands . . . 979
428.4 Module . . . 979
428.5 Example job . . . 980

429Stacks **981**
429.1 Introduction . . . 981
429.2 Versions . . . 981
429.3 Commands . . . 981
429.4 Module . . . 982
429.5 Example job . . . 982

430STAR **983**
430.1 Introduction . . . 983
430.2 Versions . . . 983
430.3 Commands . . . 983
430.4 Module . . . 983
430.5 Example job . . . 983

431Staramr **985**
431.1 Introduction . . . 985
431.2 Versions . . . 985
431.3 Commands . . . 985
431.4 Module . . . 985
431.5 Example job . . . 986

432STAR-Fusion **987**
432.1 Introduction . . . 987
432.2 Versions . . . 987
432.3 Commands . . . 987
432.4 Module . . . 987
432.5 Example job . . . 987

433STREAM **989**
433.1 Introduction . . . 989
433.2 Versions . . . 989
433.3 Commands . . . 989
433.4 Module . . . 989
433.5 Example job . . . 990

434StringTie **991**
434.1 Introduction . . . 991
434.2 Versions . . . 991
434.3 Commands . . . 991
434.4 Module . . . 991
434.5 Example job . . . 992

435Strique **993**
435.1 Introduction . . . 993
435.2 Versions . . . 993
435.3 Commands . . . 993
435.4 Module . . . 993
435.5 Example job . . . 994

436Structure **995**
436.1 Introduction . . . 995
436.2 Versions . . . 995
436.3 Commands . . . 995
436.4 Module . . . 995
436.5 Example job . . . 995

437Subread **997**
437.1 Introduction . . . 997
437.2 Versions . . . 997
437.3 Commands . . . 997
437.4 Module . . . 998
437.5 Example job . . . 998

438Survivor **999**
438.1 Introduction . . . 999
438.2 Versions . . . 999
438.3 Commands . . . 999
438.4 Module . . . 999
438.5 Example job . . . 1000

439Svaba **1001**
439.1 Introduction . . . 1001
439.2 Versions . . . 1001
439.3 Commands . . . 1001
439.4 Module . . . 1001
439.5 Example job . . . 1002

440Svtools **1003**
440.1 Introduction . . . 1003
440.2 Versions . . . 1003
440.3 Commands . . . 1003
440.4 Module . . . 1003
440.5 Example job . . . 1003

441Svtyper **1005**
441.1 Introduction . . . 1005
441.2 Versions . . . 1005
441.3 Commands . . . 1005
441.4 Module . . . 1006
441.5 Example job . . . 1006

442swat **1007**
442.1 Introduction . . . 1007
442.2 Versions . . . 1007
442.3 Commands . . . 1007
442.4 Module . . . 1007
442.5 Example job . . . 1007

443Syri **1009**
443.1 Introduction . . . 1009
443.2 Versions . . . 1009
443.3 Commands . . . 1009
443.4 Module . . . 1009
443.5 Example job . . . 1010

444Talon **1011**
444.1 Introduction . . . 1011
444.2 Versions . . . 1011
444.3 Commands . . . 1011
444.4 Module . . . 1012
444.5 Example job . . . 1012

445Targetp **1013**
445.1 Introduction . . . 1013
445.2 Versions . . . 1013
445.3 Commands . . . 1013
445.4 Module . . . 1013
445.5 Example job . . . 1014

446Tassel **1015**
446.1 Introduction . . . 1015
446.2 Versions . . . 1015
446.3 Commands . . . 1015
446.4 Module . . . 1015
446.5 Example job . . . 1016

447Taxonkit **1017**
447.1 Introduction . . . 1017
447.2 Versions . . . 1017
447.3 Commands . . . 1017
447.4 Module . . . 1017
447.5 Example job . . . 1017

448T-coffee **1019**
448.1 Introduction . . . 1019
448.2 Versions . . . 1019
448.3 Commands . . . 1019
448.4 Module . . . 1019
448.5 Example job . . . 1019

449Tetranscripts **1021**
449.1 Introduction . . . 1021
449.2 Versions . . . 1021
449.3 Commands . . . 1021
449.4 Module . . . 1021
449.5 Example job . . . 1022

450Tiara **1023**
450.1 Introduction . . . 1023
450.2 Versions . . . 1023
450.3 Commands . . . 1023
450.4 Module . . . 1023
450.5 Example job . . . 1023

451Tigmint **1025**
451.1 Introduction . . . 1025
451.2 Versions . . . 1025
451.3 Commands . . . 1025
451.4 Module . . . 1026
451.5 Example job . . . 1026

452Tobias **1027**
452.1 Introduction . . . 1027
452.2 Versions . . . 1027
452.3 Commands . . . 1027
452.4 Module . . . 1027
452.5 Example job . . . 1027

453Tombo **1029**
453.1 Introduction . . . 1029
453.2 Versions . . . 1029
453.3 Commands . . . 1029
453.4 Module . . . 1029
453.5 Example job . . . 1029

454TopHat **1031**
454.1 Introduction . . . 1031
454.2 Versions . . . 1031
454.3 Commands . . . 1031
454.4 Module . . . 1031
454.5 Example job . . . 1032

455TPMCalculator **1033**
455.1 Introduction . . . 1033
455.2 Versions . . . 1033
455.3 Commands . . . 1033
455.4 Module . . . 1033
455.5 Example job . . . 1033

456Transabyss **1035**
456.1 Introduction . . . 1035
456.2 Versions . . . 1035
456.3 Commands . . . 1035
456.4 Module . . . 1035
456.5 Example job . . . 1036

457TransDecoder **1037**
457.1 Introduction . . . 1037
457.2 Versions . . . 1037
457.3 Commands . . . 1037
457.4 Module . . . 1038
457.5 Example job . . . 1038

458Transrate **1041**
458.1 Introduction . . . 1041
458.2 Versions . . . 1041
458.3 Commands . . . 1041
458.4 Module . . . 1041
458.5 Example job . . . 1042

459Transvar **1043**
459.1 Introduction . . . 1043
459.2 Versions . . . 1043
459.3 Commands . . . 1043
459.4 Module . . . 1043
459.5 Example job . . . 1043

460tRAX **1045**
460.1 Introduction . . . 1045
460.2 Versions . . . 1045
460.3 Commands . . . 1045
460.4 Module . . . 1045
460.5 Example job . . . 1046

461Treetime **1047**
461.1 Introduction . . . 1047
461.2 Versions . . . 1047
461.3 Commands . . . 1047
461.4 Module . . . 1047
461.5 Example job . . . 1047

462Trimal **1049**
462.1 Introduction . . . 1049
462.2 Versions . . . 1049
462.3 Commands . . . 1049
462.4 Module . . . 1049
462.5 Example job . . . 1050

463Trim-galore **1051**
463.1 Introduction . . . 1051
463.2 Versions . . . 1051
463.3 Commands . . . 1051
463.4 Module . . . 1051
463.5 Example job . . . 1051

464Trimmomatic **1053**
464.1 Introduction . . . 1053
464.2 Versions . . . 1053
464.3 Commands . . . 1053
464.4 Module . . . 1053
464.5 Example job . . . 1053

465Trinity **1055**
465.1 Introduction . . . 1055
465.2 Versions . . . 1055
465.3 Commands . . . 1055
465.4 Module . . . 1056
465.5 Example job . . . 1057

466Trinotate **1059**
466.1 Introduction . . . 1059
466.2 Versions . . . 1059
466.3 Commands . . . 1059
466.4 Module . . . 1061
466.5 Example job . . . 1061

467Trnascan-se **1063**
467.1 Introduction . . . 1063
467.2 Versions . . . 1063
467.3 Commands . . . 1063
467.4 Module . . . 1063
467.5 Example job . . . 1063

468Trust4 **1065**
468.1 Introduction . . . 1065
468.2 Versions . . . 1065
468.3 Commands . . . 1065
468.4 Module . . . 1066
468.5 Example job . . . 1066

469Trycycler **1067**
469.1 Introduction . . . 1067
469.2 Versions . . . 1067
469.3 Commands . . . 1067
469.4 Module . . . 1067
469.5 Example job . . . 1068

470UCSC Executables **1069**
470.1 Introduction . . . 1069
470.2 Versions . . . 1069
470.3 Commands . . . 1069
470.4 Module . . . 1078
470.5 Example job . . . 1078

471Unicycler **1079**
471.1 Introduction . . . 1079
471.2 Versions . . . 1079
471.3 Commands . . . 1079
471.4 Module . . . 1079
471.5 Example job . . . 1079

472Vadr **1081**
472.1 Introduction . . . 1081
472.2 Versions . . . 1081
472.3 Commands . . . 1081
472.4 Module . . . 1082
472.5 Example job . . . 1082

473Vardict-java **1083**
473.1 Introduction . . . 1083
473.2 Versions . . . 1083
473.3 Commands . . . 1083
473.4 Module . . . 1083
473.5 Example job . . . 1084

474Varlociraptor **1085**
474.1 Introduction . . . 1085
474.2 Versions . . . 1085
474.3 Commands . . . 1085
474.4 Module . . . 1085
474.5 Example job . . . 1085

475Varscan **1087**
475.1 Introduction . . . 1087
475.2 Versions . . . 1087
475.3 Commands . . . 1087
475.4 Module . . . 1087
475.5 Example job . . . 1087

476Vartrix **1089**
476.1 Introduction . . . 1089
476.2 Versions . . . 1089
476.3 Commands . . . 1089
476.4 Module . . . 1089
476.5 Example job . . . 1089

477Vatools **1091**
477.1 Introduction . . . 1091
477.2 Versions . . . 1091
477.3 Commands . . . 1091
477.4 Module . . . 1092
477.5 Example job . . . 1092

478Vcf2maf **1093**
478.1 Introduction . . . 1093
478.2 Versions . . . 1093
478.3 Commands . . . 1093
478.4 Module . . . 1093
478.5 Example job . . . 1094

479Vcf2phylip **1095**
479.1 Introduction . . . 1095
479.2 Versions . . . 1095
479.3 Commands . . . 1095
479.4 Module . . . 1095
479.5 Example job . . . 1096

480Vcf-kit **1097**
480.1 Introduction . . . 1097
480.2 Versions . . . 1097
480.3 Commands . . . 1097
480.4 Module . . . 1097
480.5 Example job . . . 1097

481VCFtools **1099**
481.1 Introduction . . . 1099
481.2 Versions . . . 1099
481.3 Commands . . . 1099
481.4 Module . . . 1099
481.5 Example job . . . 1099

482Velocyto.py **1101**
482.1 Introduction . . . 1101
482.2 Versions . . . 1101
482.3 Commands . . . 1101
482.4 Module . . . 1101
482.5 Interactive job . . . 1101

482.6 Batch job . . . 1102

483Velvet **1103**
483.1 Introduction . . . 1103
483.2 Versions . . . 1103
483.3 Commands . . . 1103
483.4 Module . . . 1103
483.5 Example job . . . 1104

484Vg **1105**
484.1 Introduction . . . 1105
484.2 Versions . . . 1105
484.3 Commands . . . 1105
484.4 Module . . . 1105
484.5 Example job . . . 1106

485Viennarna **1107**
485.1 Introduction . . . 1107
485.2 Versions . . . 1107
485.3 Commands . . . 1107
485.4 Module . . . 1108
485.5 Example job . . . 1108

486Vsearch **1111**
486.1 Introduction . . . 1111
486.2 Versions . . . 1111
486.3 Commands . . . 1111
486.4 Module . . . 1111
486.5 Example job . . . 1111

487Weblogo **1113**
487.1 Introduction . . . 1113
487.2 Versions . . . 1113
487.3 Commands . . . 1113
487.4 Module . . . 1113
487.5 Example job . . . 1113

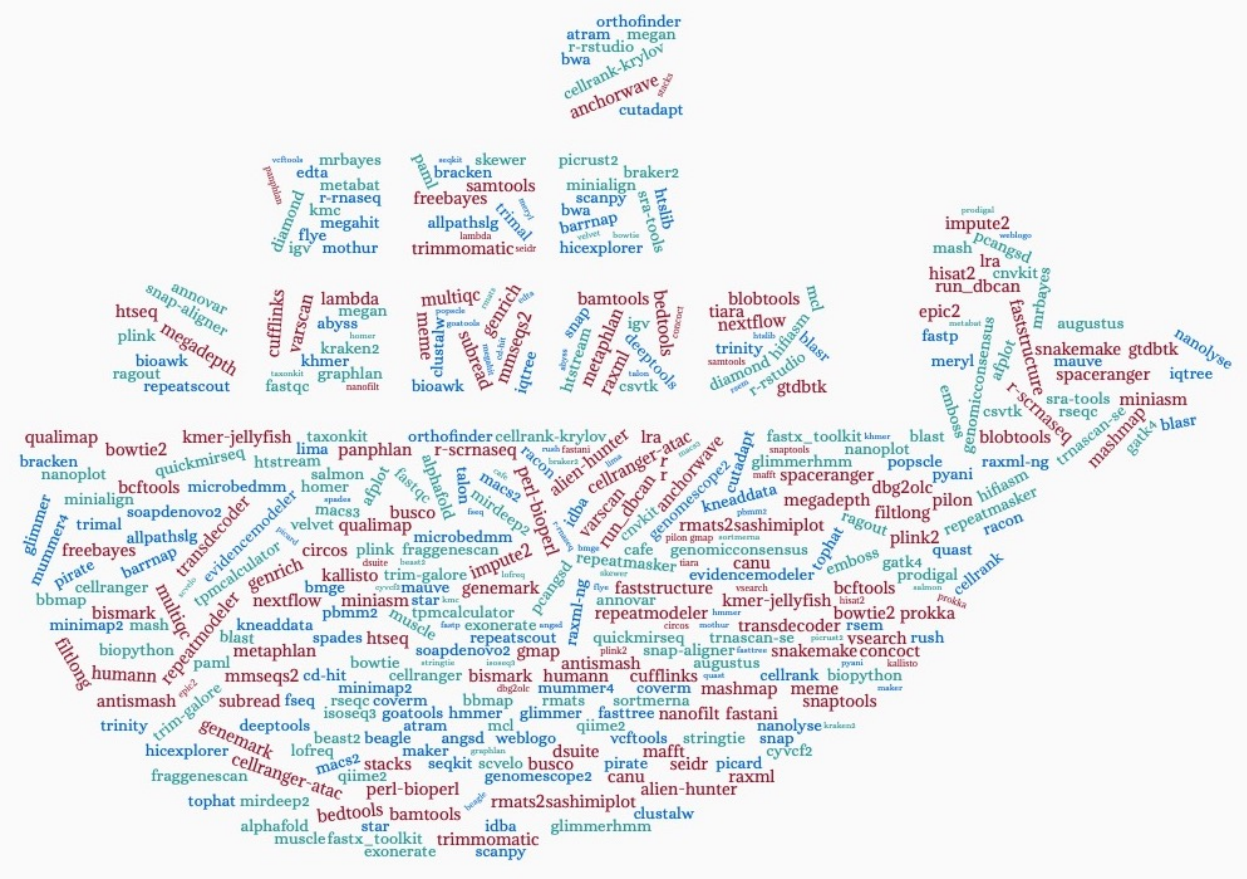
488Whatshap **1115**
488.1 Introduction . . . 1115
488.2 Versions . . . 1115
488.3 Commands . . . 1115
488.4 Module . . . 1115
488.5 Example job . . . 1116

489Wiggletools **1117**
489.1 Introduction . . . 1117
489.2 Versions . . . 1117
489.3 Commands . . . 1117
489.4 Module . . . 1117
489.5 Example job . . . 1118

490Winnowmap **1119**
490.1 Introduction . . . 1119
490.2 Versions . . . 1119
490.3 Commands . . . 1119

490.4 Module 1119
490.5 Example job 1119

491Wtdbg2 **1121**
491.1 Introduction 1121
491.2 Versions 1121
491.3 Commands 1121
491.4 Module 1121
491.5 Example job 1122



This is the user guide for biocontainer modules deployed in Purdue High Performance Computing clusters. More information about our center is avaiable here (https://www.rcac.purdue.edu).

If you have any question, contact me(Yucheng Zhang) at: zhan4429@purdue.edu

Warning: Do not use both `bioinfo` and `biocontainers` in your job script, because loading `bioinfo` will cause the failure of loading many modules including `biocontainers` in `Brown`, `Halstead`, `Scholar`, `Workbench`, and `Gilbreth`. Since RCAC will not provide support to `bioinfo` in the future clusters, we recommend users to just use `biocontainers`.

CHAPTER

ONE

FREQUENTLY ASKED QUESTIONS

What are the advantages of using biocontainers

Biocontainers are based on the popular container techonlogy. Due to their ease of deployment and portability, RCAC can deploy a large number of bioinformatic applications on our clusters, and keep adding newer versions. In addition, containerized applications can help improve reproductivity of scientists' research. Using biocontainers, you can generate the same results no matter which cluster you are using, and no matter whether you run the program today or 10 years later.

Can we use both bioinfo and biocontainers in our job script?

No. If you load bioinfo, you will find that you cannot load biocontainers. This is a legacy issue, and all clusters are affected except Bell. So you can use either bioinfo or biocontainers in your job script, just do not use both.

How should I load biocontainers after I load bioinfo? The error message shows "biocontainers" is unknown.

Run below commands:

- module purge
- module load modtree/new
- module load biocontainers

I cannot find the path to executables by `which` ?

Biocontainers' exectuables are located inside containers instead of the host system of cluster. The commands we provide are actually alias to `singularity exec /apps/biocontainers/images/image.sif command`. For example, the `blastp` command you use is actually `singularity exec /apps/biocontainers/images/blast.sif blastp`. For applications requiring users to provide exectuable path such as `RSEM` and `MAKER`, please check their specific user guides we provide.

CHAPTER

TWO

SINGULARITY

Note: Singularity was originally a project out of Lawrence Berkeley National Laboratory. It has now been spun off into a distinct offering under a new corporate entity under the name Sylabs Inc. This guide pertains to the open source community edition, *SingularityCE*.

2.1 What is Singularity?

Singularity is a new feature of the Community Clusters allowing the portability and reproducibility of operating system and application environments through the use of Linux containers. It gives users complete control over their environment.

Singularity is like Docker but tuned explicitly for HPC clusters. More information is available from the project's website.

2.2 Features

- Run the latest applications on an Ubuntu or Centos userland
- Gain access to the latest developer tools
- Launch MPI programs easily
- Much more

Singularity's user guide is available at: sylabs.io/guides/3.8/user-guide

2.3 Example

Here is an example using an Ubuntu 16.04 image on Weber:

```
singularity exec /depot/itap/singularity/ubuntu1604.img cat /etc/lsb-release
DISTRIB_ID=Ubuntu
DISTRIB_RELEASE=16.04
DISTRIB_CODENAME=xenial
DISTRIB_DESCRIPTION="Ubuntu 16.04 LTS"
```

Here is another example using a Centos 7 image:

```
singularity exec /depot/itap/singularity/centos7.img cat /etc/redhat-release
CentOS Linux release 7.2.1511 (Core)
```

2.4 Purdue Cluster Specific Notes

All service providers will integrate Singularity slightly differently depending on site. The largest customization will be which default files are inserted into your images so that routine services will work.

Services we configure for your images include DNS settings and account information. File systems we overlay into your images are your home directory, scratch, Data Depot, and application file systems.

Here is a list of paths:

- /etc/resolv.conf
- /etc/hosts
- /home/$USER
- /apps
- /scratch
- /depot

This means that within the container environment these paths will be present and the same as outside the container. The `/apps`, `/scratch`, and `/depot` directories will need to exist *inside* your container to work properly.

2.5 Creating Singularity Images

Due to how singularity containers work, you must have root privileges to *build* an image. Once you have a singularity container image built on your own system, you can copy the image file up to the cluster (you do not need root privileges to *run* the container).

You can find information and documentation for how to install and use singularity on your system:

- Install Singularity on Windows
- Install Singularity on macOS
- Install Singularity on Linux

We have version `3.8.0-1.el7` on the cluster. You will most likely not be able to run any container built with any singularity past that version. So be sure to follow the installation guide for version 3.8 on your system:

```
singularity --version
singularity version 3.8.0-1.el7
```

Everything you need on how to build a container is available from their user-guide. Below are merely some quick tips for getting your own containers built for Weber.

You can use a Definition File to both build your container and share its specification with collaborators (for the sake of reproducibility). Here is a simplistic example of such a file:

```
# FILENAME: Buildfile

Bootstrap: docker
From: ubuntu:18.04

%post
    apt-get update && apt-get upgrade -y
    mkdir /apps /depot /scratch
```

To build the image itself:

```
sudo singularity build ubuntu-18.04.sif Buildfile
```

The challenge with this approach however is that it must start from scratch if you decide to change something. In order to create a container image iteratively and interactively, you can use the `--sandbox` option:

```
sudo singularity build --sandbox ubuntu-18.04 docker://ubuntu:18.04
```

This will not create a flat image file but a directory tree (i.e., a folder), the contents of which are the container's filesystem. In order to get a shell inside the container that allows you to modify it, user the `--writable` option:

```
sudo singularity shell --writable ubuntu-18.04
Singularity: Invoking an interactive shell within container...

Singularity ubuntu-18.04.sandbox:~>
```

You can then proceed to install any libraries, software, etc. within the container. Then to create the final image file, `exit` the shell and call the `build` command once more on the *sandbox*:

```
sudo singularity build ubuntu-18.04.sif ubuntu-18.04
```

Finally, copy the new image to Weber and run it.

CHAPTER

THREE

ABACAS

3.1 Introduction

`Abacas` is a tool for algorithm based automatic contiguation of assembled sequences.

For more information, please check its website: https://biocontainers.pro/tools/abacas and its home page: http://abacas.sourceforge.net.

3.2 Versions

- 1.3.1

3.3 Commands

- abacas.pl
- abacas.1.3.1.pl

3.4 Module

You can load the modules by:

```
module load biocontainers
module load abacas
```

3.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Abacas on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=abacas
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers abacas

abacas.pl -r cmm.fasta -q Cm.contigs.fasta -p nucmer -o out_prefix
```

CHAPTER

FOUR

ABISMAL

4.1 Introduction

Another Bisulfite Mapping Algorithm (abismal) is a read mapping program for bisulfite sequencing in DNA methylation studies.

For more information, please check:
BioContainers: https://biocontainers.pro/tools/abismal
Home page: https://github.com/smithlabcode/abismal

4.2 Versions

- 3.0.0

4.3 Commands

- abismal
- abismalidx
- simreads

4.4 Module

You can load the modules by:

```
module load biocontainers
module load abismal
```

4.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run abismal on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=abismal
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers abismal

abismalidx  ~/.local/share/genomes/hg38/hg38.fa hg38
```

CHAPTER

FIVE

ABRICATE

5.1 Introduction

`Abricate` is a tool for mass screening of contigs for antimicrobial resistance or virulence genes.

For more information, please check its website: https://biocontainers.pro/tools/abricate and its home page on Github.

5.2 Versions

- 1.0.1

5.3 Commands

- abricate

5.4 Module

You can load the modules by:

```
module load biocontainers
module load abricate
```

5.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Abricate on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 8
#SBATCH --job-name=abricate
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers abricate

abricate --threads 8 *.fasta
```

CHAPTER

SIX

ABYSS

6.1 Introduction

ABySS is a de novo sequence assembler intended for short paired-end reads and genomes of all sizes.

For more information, please check its website: https://biocontainers.pro/tools/abyss and its home page on Github.

6.2 Versions

- 2.3.2
- 2.3.4

6.3 Commands

- ABYSS
- ABYSS-P
- AdjList
- Consensus
- DAssembler
- DistanceEst
- DistanceEst-ssq
- KAligner
- MergeContigs
- MergePaths
- Overlap
- ParseAligns
- PathConsensus
- PathOverlap
- PopBubbles

- SimpleGraph
- abyss-align
- abyss-bloom
- abyss-bloom-dbg
- abyss-bowtie
- abyss-bowtie2
- abyss-bwa
- abyss-bwamem
- abyss-bwasw
- abyss-db-txt
- abyss-dida
- abyss-fac
- abyss-fatoagp
- abyss-filtergraph
- abyss-fixmate
- abyss-fixmate-ssq
- abyss-gapfill
- abyss-gc
- abyss-index
- abyss-junction
- abyss-kaligner
- abyss-layout
- abyss-longseqdist
- abyss-map
- abyss-map-ssq
- abyss-mergepairs
- abyss-overlap
- abyss-paired-dbg
- abyss-paired-dbg-mpi
- abyss-pe
- abyss-rresolver-short
- abyss-samtoafg
- abyss-scaffold
- abyss-sealer
- abyss-stack-size
- abyss-tabtomd

- abyss-todot
- abyss-tofastq
- konnector
- logcounter

6.4 Module

You can load the modules by:

```
module load biocontainers
module load abyss
```

6.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run abyss on our our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 4
#SBATCH --job-name=abyss
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers abyss

abyss-pe np=4 k=25 name=test B=1G \
    in='test-data/reads1.fastq test-data/reads2.fastq'
```

CHAPTER

SEVEN

ACTC

7.1 Introduction

Actc is used to align subreads to ccs reads.

For more information, please check:
Home page: https://github.com/PacificBiosciences/actc

7.2 Versions

- 0.2.0

7.3 Commands

- actc

7.4 Module

You can load the modules by:

```
module load biocontainers
module load actc
```

7.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run actc on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=actc
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers actc

actc subreads.bam ccs.bam subreads_to_ccs.bam
```

CHAPTER

EIGHT

ADVNTR

8.1 Introduction

`Advntr` is a tool for genotyping Variable Number Tandem Repeats (VNTR) from sequence data.

For more information, please check its website: https://biocontainers.pro/tools/advntr and its home page on Github.

8.2 Versions

- 1.4.0

8.3 Commands

- advntr

8.4 Module

You can load the modules by:

```
module load biocontainers
module load advntr
```

8.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Advntr on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=advntr
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers advntr

advntr addmodel -r chr21.fa -p CGCGGGGCGGGG -s 45196324 -e 45196360 -c chr21
advntr genotype --vntr_id 1 --alignment_file CSTB_2_5_testdata.bam --working_directory␣
↪working_dir
```

CHAPTER

NINE

AFPLOT

9.1 Introduction

`Afplot` is a tool to plot allele frequencies in VCF files.

For more information, please check its website: https://biocontainers.pro/tools/afplot and its home page on Github.

9.2 Versions

- 0.2.1-py36

9.3 Commands

- afplot

9.4 Module

You can load the modules by:

```
module load biocontainers
module load afplot
```

9.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run afplot on our our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=afplot
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers afplot

afplot whole-genome histogram -v my_vcf.gz -l my_label -s my_sample -o mysample.
↪histogram.png
```

CHAPTER

TEN

AFTERQC

10.1 Introduction

`Afterqc` is a tool for quality control of FASTQ data produced by HiSeq 2000/2500/3000/4000, Nextseq 500/550, MiniSeq, and Illumina 1.8 or newer.

For more information, please check its website: https://biocontainers.pro/tools/afterqc and its home page on Github.

10.2 Versions

- 0.9.7

10.3 Commands

- after.py

10.4 Module

You can load the modules by:

```
module load biocontainers
module load afterqc
```

10.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run blobtools on our our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=afterqc
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers afterqc

after.py -1 SRR11941281_1.fastq.paired.fq  -2 SRR11941281_2.fastq.paired.fq
```

CHAPTER

ELEVEN

AGAT

11.1 Introduction

`Agat` is a suite of tools to handle gene annotations in any GTF/GFF format.

For more information, please check its website: https://biocontainers.pro/tools/agat and its home page on Github.

11.2 Versions

- 0.8.1

11.3 Commands

- agat_convert_bed2gff.pl
- agat_convert_embl2gff.pl
- agat_convert_genscan2gff.pl
- agat_convert_mfannot2gff.pl
- agat_convert_minimap2_bam2gff.pl
- agat_convert_sp_gff2bed.pl
- agat_convert_sp_gff2gtf.pl
- agat_convert_sp_gff2tsv.pl
- agat_convert_sp_gff2zff.pl
- agat_convert_sp_gxf2gxf.pl
- agat_sp_Prokka_inferNameFromAttributes.pl
- agat_sp_add_introns.pl
- agat_sp_add_start_and_stop.pl
- agat_sp_alignment_output_style.pl
- agat_sp_clipN_seqExtremities_and_fixCoordinates.pl
- agat_sp_compare_two_BUSCOs.pl

- agat_sp_compare_two_annotations.pl
- agat_sp_complement_annotations.pl
- agat_sp_ensembl_output_style.pl
- agat_sp_extract_attributes.pl
- agat_sp_extract_sequences.pl
- agat_sp_filter_by_ORF_size.pl
- agat_sp_filter_by_locus_distance.pl
- agat_sp_filter_by_mrnaBlastValue.pl
- agat_sp_filter_feature_by_attribute_presence.pl
- agat_sp_filter_feature_by_attribute_value.pl
- agat_sp_filter_feature_from_keep_list.pl
- agat_sp_filter_feature_from_kill_list.pl
- agat_sp_filter_gene_by_intron_numbers.pl
- agat_sp_filter_gene_by_length.pl
- agat_sp_filter_incomplete_gene_coding_models.pl
- agat_sp_filter_record_by_coordinates.pl
- agat_sp_fix_cds_phases.pl
- agat_sp_fix_features_locations_duplicated.pl
- agat_sp_fix_fusion.pl
- agat_sp_fix_longest_ORF.pl
- agat_sp_fix_overlaping_genes.pl
- agat_sp_fix_small_exon_from_extremities.pl
- agat_sp_flag_premature_stop_codons.pl
- agat_sp_flag_short_introns.pl
- agat_sp_functional_statistics.pl
- agat_sp_keep_longest_isoform.pl
- agat_sp_kraken_assess_liftover.pl
- agat_sp_list_short_introns.pl
- agat_sp_load_function_from_protein_align.pl
- agat_sp_manage_IDs.pl
- agat_sp_manage_UTRs.pl
- agat_sp_manage_attributes.pl
- agat_sp_manage_functional_annotation.pl
- agat_sp_manage_introns.pl
- agat_sp_merge_annotations.pl
- agat_sp_prokka_fix_fragmented_gene_annotations.pl

- agat_sp_sensitivity_specificity.pl
- agat_sp_separate_by_record_type.pl
- agat_sp_statistics.pl
- agat_sp_webApollo_compliant.pl
- agat_sq_add_attributes_from_tsv.pl
- agat_sq_add_hash_tag.pl
- agat_sq_add_locus_tag.pl
- agat_sq_count_attributes.pl
- agat_sq_filter_feature_from_fasta.pl
- agat_sq_list_attributes.pl
- agat_sq_manage_IDs.pl
- agat_sq_manage_attributes.pl
- agat_sq_mask.pl
- agat_sq_remove_redundant_entries.pl
- agat_sq_repeats_analyzer.pl
- agat_sq_rfam_analyzer.pl
- agat_sq_split.pl
- agat_sq_stat_basic.pl

11.4 Module

You can load the modules by:

```
module load biocontainers
module load agat
```

11.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Agat on our our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=agat
```

(continues on next page)

(continued from previous page)

```
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers agat

agat_convert_sp_gff2bed.pl  --gff genes.gff -o genes.bed
```

CHAPTER

TWELVE

ALFRED

12.1 Introduction

`Alfred` is an efficient and versatile command-line application that computes multi-sample quality control metrics in a read-group aware manner.

For more information, please check its website: https://biocontainers.pro/tools/alfred and its home page on Github.

12.2 Versions

- 0.2.5
- 0.2.6

12.3 Commands

- alfred

12.4 Module

You can load the modules by:

```
module load biocontainers
module load alfred
```

12.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Alfred on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=alfred
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers alfred

alfred qc -r genome.fasta -o qc.tsv.gz sorted.bam
```

CHAPTER

THIRTEEN

ALIEN-HUNTER

13.1 Introduction

`Alien-hunter` is an application for the prediction of putative Horizontal Gene Transfer (HGT) events with the implementation of Interpolated Variable Order Motifs (IVOMs).

For more information, please check its website: https://biocontainers.pro/tools/alien-hunter and its home page: https://www.sanger.ac.uk/tool/alien-hunter/.

13.2 Versions

- 1.7.7

13.3 Commands

- alien_hunter

13.4 Module

You can load the modules by:

```
module load biocontainers
module load alien_hunter
```

13.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Alien_hunter on our our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=alien_hunter
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers alien_hunter

alien_hunter genome.fasta output
```

CHAPTER

FOURTEEN

ALIGNSTATS

14.1 Introduction

AlignStats produces various alignment, whole genome coverage, and capture coverage metrics for sequence alignment files in SAM, BAM, and CRAM format.

For more information, please check:
BioContainers: https://biocontainers.pro/tools/alignstats
Home page: https://github.com/jfarek/alignstats

14.2 Versions

- 0.9.1

14.3 Commands

- alignstats

14.4 Module

You can load the modules by:

```
module load biocontainers
module load alignstats
```

14.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run alignstats on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=alignstats
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers alignstats

alignstats -C -i input.bam -o report.txt
```

CHAPTER

FIFTEEN

ALLPATHSLG

15.1 Introduction

`Allpathslg` is a whole-genome shotgun assembler that can generate high-quality genome assemblies using short reads.

For more information, please check its website: https://biocontainers.pro/tools/allpathslg and its home page: https://software.broadinstitute.org/allpaths-lg/blog/.

15.2 Versions

- 52488

15.3 Commands

- PrepareAllPathsInputs.pl
- RunAllPathsLG
- CacheLibs.pl
- Fasta2Fastb

15.4 Module

You can load the modules by:

```
module load biocontainers
module load allpathslg
```

15.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Allpathslg on our our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 24
#SBATCH --job-name=allpathslg
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers allpathslg

PrepareAllPathsInputs.pl \
                       DATA_DIR=data \
                       PLOIDY=1 \
                       IN_GROUPS_CSV=in_groups.csv\
                       IN_LIBS_CSV=in_libs.csv\
                       OVERWRITE=True\

RunAllPathsLG PRE=allpathlg REFERENCE_NAME=test.genome \
              DATA_SUBDIR=data  RUN=myrun TARGETS=standard \
              SUBDIR=test OVERWRITE=True
```

~

CHAPTER

SIXTEEN

ALPHAFOLD

16.1 Introduction

`Alphafold` is a protein structure prediction tool developed by DeepMind (Google). It uses a novel machine learning approach to predict 3D protein structures from primary sequences alone. The source code is available on Github. It has been deployed in all RCAC clusters, supporting both CPU and GPU.

It also relies on a huge database. The full database (~2.2TB) has been downloaded and setup for users.

16.2 Versions

- 2.1.1
- 2.2.0
- 2.2.3

16.3 Commands

run_alphafold.sh

16.4 Module

You can load the modules by:

```
module load biocontainers
module load alphafold/2.1.1
```

16.5 Usage

The usage of Alphafold on our cluster is very straightford:

```
run_alphafold.sh --flagfile=$AlphaDB --fasta_paths=XX --output_dir=XX ...
```

`$AlphaDB` (/depot/itap/datasets/alphafold/full_db.ff) is a configuration file passed to AlphaFold containing the location of the database. Typically it should not be edited. Users can add other parameters based on your needs.

Users can check its detaied user guide in its Github.

16.6 AlphaDB

Contents of $AlphaDB:

```
--db_preset=full_dbs
--bfd_database_path=/depot/itap/datasets/alphafold/db/bfd/bfd_metaclust_clu_complete_
↪id30_c90_final_seq.sorted_opt
--data_dir=/depot/itap/datasets/alphafold/db/
--uniref90_database_path=/depot/itap/datasets/alphafold/db/uniref90/uniref90.fasta
--mgnify_database_path=/depot/itap/datasets/alphafold/db/mgnify/mgy_clusters_2018_12.fa
--uniclust30_database_path=/depot/itap/datasets/alphafold/db/uniclust30/uniclust30_2018_
↪08/uniclust30_2018_08
--pdb70_database_path=/depot/itap/datasets/alphafold/db/pdb70/pdb70
--template_mmcif_dir=/depot/itap/datasets/alphafold/db/pdb_mmcif/mmcif_files
--max_template_date=2022-01-29
--obsolete_pdbs_path=/depot/itap/datasets/alphafold/db/pdb_mmcif/obsolete.dat
--hhblits_binary_path=/usr/bin/hhblits
--hhsearch_binary_path=/usr/bin/hhsearch
--jackhmmer_binary_path=/usr/bin/jackhmmer
--kalign_binary_path=/usr/bin/kalign
```

16.7 Example job using CPU

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run alphafold using CPU:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 20:00:00
#SBATCH -N 1
#SBATCH -n 24
#SBATCH --job-name=alphafold
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out
```

(continues on next page)

(continued from previous page)

```
module --force purge
ml biocontainers alphafold/2.1.1

run_alphafold.sh  --flagfile=$AlphaDB --fasta_paths=/scratch/bell/zhan4429/Containers4/
↪alphafold/sample.fasta --max_template_date=2022-02-01 \
--output_dir=/scratch/bell/zhan4429/Containers4/alphafold/af2_full --model_preset=monomer
```

16.8 Example job using GPU

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run alphafold using GPU:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 20:00:00
#SBATCH -N 1
#SBATCH -n 24
#SBATCH --gres=gpu:1
#SBATCH --job-name=alphafold
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers alphafold/2.1.1

run_alphafold.sh  --flagfile=$AlphaDB --fasta_paths=/scratch/bell/zhan4429/Containers4/
↪alphafold/sample.fasta --max_template_date=2022-02-01 \
--output_dir=/scratch/bell/zhan4429/Containers4/alphafold/af2_full --model_preset=monomer
```

CHAPTER

SEVENTEEN

AMPTK

17.1 Introduction

`Amptk` is a series of scripts to process NGS amplicon data using USEARCH and VSEARCH, it can also be used to process any NGS amplicon data and includes databases setup for analysis of fungal ITS, fungal LSU, bacterial 16S, and insect COI amplicons.

For more information, please check its website: https://biocontainers.pro/tools/amptk and its home page on Github.

17.2 Versions

- 1.5.4

17.3 Commands

- amptk

17.4 Module

You can load the modules by:

```
module load biocontainers
module load amptk
```

17.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Amptk on our our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 4
#SBATCH --job-name=amptk
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers amptk

amptk illumina -i test_data/illumina_test_data -o miseq -f fITS7 -r ITS4  --cpus 4
```

CHAPTER

EIGHTEEN

ANANSE

18.1 Introduction

ANANSE is a computational approach to infer enhancer-based gene regulatory networks (GRNs) and to identify key transcription factors between two GRNs.

For more information, please check:
BioContainers: https://biocontainers.pro/tools/ananse
Home page: https://github.com/vanheeringen-lab/ANANSE

18.2 Versions

- 0.4.0

18.3 Commands

- ananse

18.4 Module

You can load the modules by:

```
module load biocontainers
module load ananse
```

18.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run ananse on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=ananse
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers ananse

mkdir -p ANANSE.REMAP.model.v1.0
wget https://zenodo.org/record/4768075/files/ANANSE.REMAP.model.v1.0.tgz
tar xvzf ANANSE.REMAP.model.v1.0.tgz -C ANANSE.REMAP.model.v1.0
rm ANANSE.REMAP.model.v1.0.tgz

wget https://zenodo.org/record/4769814/files/ANANSE_example_data.tgz
tar xvzf ANANSE_example_data.tgz
rm ANANSE_example_data.tgz

ananse binding -H ANANSE_example_data/H3K27ac/fibroblast*bam -A ANANSE_example_data/ATAC/
↪fibroblast*bam -R ANANSE.REMAP.model.v1.0/ -o fibroblast.binding
ananse binding -H ANANSE_example_data/H3K27ac/heart*bam -A ANANSE_example_data/ATAC/
↪heart*bam -R ANANSE.REMAP.model.v1.0/ -o heart.binding

ananse network -b  fibroblast.binding/binding.h5 -e ANANSE_example_data/RNAseq/
↪fibroblast*TPM.txt -n 4 -o fibroblast.network.txt
ananse network -b  heart.binding/binding.h5 -e ANANSE_example_data/RNAseq/heart*TPM.txt -
↪n 4 -o heart.network.txt

ananse influence -s fibroblast.network.txt -t heart.network.txt -d ANANSE_example_data/
↪RNAseq/fibroblast2heart_degenes.csv -p -o fibroblast2heart.influence.txt
```

CHAPTER

NINETEEN

ANCHORWAVE

19.1 Introduction

`Anchorwave` is used for sensitive alignment of genomes with high sequence diversity, extensive structural polymorphism and whole-genome duplication variation.

For more information, please check its website: https://biocontainers.pro/tools/anchorwave and its home page on Github.

19.2 Versions

- 1.0.1

19.3 Commands

- anchorwave
- gmap_build
- gmap
- minimap2

19.4 Module

You can load the modules by:

```
module load biocontainers
module load anchorwave
```

19.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Anchorwave on our our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 4
#SBATCH --job-name=anchorwave
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers anchorwave

anchorwave gff2seq -i Zea_mays.AGPv4.34.gff3 -r Zea_mays.AGPv4.dna.toplevel.fa -o cds.fa
```

CHAPTER

TWENTY

ANGSD

20.1 Introduction

ANGSD is a software for analyzing next generation sequencing data. Detailed usage can be found here: http://www.popgen.dk/angsd/index.php/ANGSD.

20.2 Versions

- 0.935
- 0.937
- 0.939

20.3 Commands

- angsd
- realSFS
- msToGlf
- thetaStat
- supersim

20.4 Module

You can load the modules by:

```
module load biocontainers
module load angsd/0.937
```

20.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run angsd on our our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 20:00:00
#SBATCH -N 1
#SBATCH -n 24
#SBATCH --job-name=angsd
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers angsd/0.937

angsd -b bam.filelist -GL 1 -doMajorMinor 1 -doMaf 2 -P 5 -minMapQ 30 -minQ 20 -minMaf 0.
↪05
```

CHAPTER

TWENTYONE

ANNOGESIC

21.1 Introduction

ANNOgesic is the swiss army knife for RNA-Seq based annotation of bacterial/archaeal genomes.

For more information, please check:
Docker hub: https://hub.docker.com/r/silasysh/annogesic
Home page: https://github.com/Sung-Huan/ANNOgesic

21.2 Versions

- 1.1.0

21.3 Commands

- annogesic

21.4 Module

You can load the modules by:

```
module load biocontainers
module load annogesic
```

21.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run annogesic on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=annogesic
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers annogesic

ANNOGESIC_FOLDER=ANNOgesic
annogesic \
    update_genome_fasta \
    -c $ANNOGESIC_FOLDER/input/references/fasta_files/NC_009839.1.fa \
    -m $ANNOGESIC_FOLDER/input/mutation_tables/mutation.csv \
    -u NC_test.1 \
    -pj $ANNOGESIC_FOLDER
```

CHAPTER

TWENTYTWO

ANNOVAR

22.1 Introduction

ANNOVAR is an efficient software tool to utilize update-to-date information to functionally annotate genetic variants detected from diverse genomes (including human genome hg18, hg19, hg38, as well as mouse, worm, fly, yeast and many others).

For more information, please check its website: https://annovar.openbioinformatics.org/en/latest/.

22.2 Versions

- 2022-01-13

22.3 Commands

- annotate_variation.pl
- coding_change.pl
- convert2annovar.pl
- retrieve_seq_from_fasta.pl
- table_annovar.pl
- variants_reduction.pl

22.4 Module

You can load the modules by:

```
module load biocontainers
module load annovar
```

22.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run ANNOVAR on our our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 4
#SBATCH --job-name=annovar
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers annovar

annotate_variation.pl --buildver hg19 --downdb seq humandb/hg19_seq
convert2annovar.pl -format region -seqdir humandb/hg19_seq/ chr1:2000001-2000003
```

CHAPTER

TWENTYTHREE

ANTISMASH

23.1 Introduction

`Antismash` Antismash allows the rapid genome-wide identification, annotation and analysis of secondary metabolite biosynthesis gene clusters in bacterial and fungal genomes.

For more information, please check its website: https://biocontainers.pro/tools/antismash and its home page: https://docs.antismash.secondarymetabolites.org.

23.2 Versions

- 5.1.2
- 6.0.1

23.3 Commands

- antismash

23.4 Module

You can load the modules by:

```
module load biocontainers
module load antismash
```

23.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Antismash on our our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 4
#SBATCH --job-name=antismash
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers antismash

antismash --cb-general --cb-knownclusters --cb-subclusters --asf --pfam2go --smcog-trees␣
↪seq.gbk
```

CHAPTER

TWENTYFOUR

ANVIO

24.1 Introduction

`Anvio` is an analysis and visualization platform for 'omics data.

For more information, please check its website: https://biocontainers.pro/tools/anvio and its home page on Github.

24.2 Versions

- 7.0

24.3 Commands

- anvi-analyze-synteny
- anvi-cluster-contigs
- anvi-compute-ani
- anvi-compute-completeness
- anvi-compute-functional-enrichment
- anvi-compute-gene-cluster-homogeneity
- anvi-compute-genome-similarity
- anvi-convert-trnaseq-database
- anvi-db-info
- anvi-delete-collection
- anvi-delete-hmms
- anvi-delete-misc-data
- anvi-delete-state
- anvi-dereplicate-genomes
- anvi-display-contigs-stats
- anvi-display-metabolism

- anvi-display-pan
- anvi-display-structure
- anvi-estimate-genome-completeness
- anvi-estimate-genome-taxonomy
- anvi-estimate-metabolism
- anvi-estimate-scg-taxonomy
- anvi-estimate-trna-taxonomy
- anvi-experimental-organization
- anvi-export-collection
- anvi-export-contigs
- anvi-export-functions
- anvi-export-gene-calls
- anvi-export-gene-coverage-and-detection
- anvi-export-items-order
- anvi-export-locus
- anvi-export-misc-data
- anvi-export-splits-and-coverages
- anvi-export-splits-taxonomy
- anvi-export-state
- anvi-export-structures
- anvi-export-table
- anvi-gen-contigs-database
- anvi-gen-fixation-index-matrix
- anvi-gen-gene-consensus-sequences
- anvi-gen-gene-level-stats-databases
- anvi-gen-genomes-storage
- anvi-gen-network
- anvi-gen-phylogenomic-tree
- anvi-gen-structure-database
- anvi-gen-variability-matrix
- anvi-gen-variability-network
- anvi-gen-variability-profile
- anvi-get-aa-counts
- anvi-get-codon-frequencies
- anvi-get-enriched-functions-per-pan-group
- anvi-get-sequences-for-gene-calls

- anvi-get-sequences-for-gene-clusters
- anvi-get-sequences-for-hmm-hits
- anvi-get-short-reads-from-bam
- anvi-get-short-reads-mapping-to-a-gene
- anvi-get-split-coverages
- anvi-help
- anvi-import-collection
- anvi-import-functions
- anvi-import-items-order
- anvi-import-misc-data
- anvi-import-state
- anvi-import-taxonomy-for-genes
- anvi-import-taxonomy-for-layers
- anvi-init-bam
- anvi-inspect
- anvi-interactive
- anvi-matrix-to-newick
- anvi-mcg-classifier
- anvi-merge
- anvi-merge-bins
- anvi-meta-pan-genome
- anvi-migrate
- anvi-oligotype-linkmers
- anvi-pan-genome
- anvi-profile
- anvi-push
- anvi-refine
- anvi-rename-bins
- anvi-report-linkmers
- anvi-run-hmms
- anvi-run-interacdome
- anvi-run-kegg-kofams
- anvi-run-ncbi-cogs
- anvi-run-pfams
- anvi-run-scg-taxonomy
- anvi-run-trna-taxonomy

- anvi-run-workflow
- anvi-scan-trnas
- anvi-script-add-default-collection
- anvi-script-augustus-output-to-external-gene-calls
- anvi-script-calculate-pn-ps-ratio
- anvi-script-checkm-tree-to-interactive
- anvi-script-compute-ani-for-fasta
- anvi-script-enrichment-stats
- anvi-script-estimate-genome-size
- anvi-script-filter-fasta-by-blast
- anvi-script-fix-homopolymer-indels
- anvi-script-gen-CPR-classifier
- anvi-script-gen-distribution-of-genes-in-a-bin
- anvi-script-gen-help-pages
- anvi-script-gen-hmm-hits-matrix-across-genomes
- anvi-script-gen-programs-network
- anvi-script-gen-programs-vignette
- anvi-script-gen-pseudo-paired-reads-from-fastq
- anvi-script-gen-scg-domain-classifier
- anvi-script-gen-short-reads
- anvi-script-gen_stats_for_single_copy_genes.R
- anvi-script-gen_stats_for_single_copy_genes.py
- anvi-script-gen_stats_for_single_copy_genes.sh
- anvi-script-get-collection-info
- anvi-script-get-coverage-from-bam
- anvi-script-get-hmm-hits-per-gene-call
- anvi-script-get-primer-matches
- anvi-script-merge-collections
- anvi-script-pfam-accessions-to-hmms-directory
- anvi-script-predict-CPR-genomes
- anvi-script-process-genbank
- anvi-script-process-genbank-metadata
- anvi-script-reformat-fasta
- anvi-script-run-eggnog-mapper
- anvi-script-snvs-to-interactive
- anvi-script-tabulate

- anvi-script-transpose-matrix
- anvi-script-variability-to-vcf
- anvi-script-visualize-split-coverages
- anvi-search-functions
- anvi-self-test
- anvi-setup-interacdome
- anvi-setup-kegg-kofams
- anvi-setup-ncbi-cogs
- anvi-setup-pdb-database
- anvi-setup-pfams
- anvi-setup-scg-taxonomy
- anvi-setup-trna-taxonomy
- anvi-show-collections-and-bins
- anvi-show-misc-data
- anvi-split
- anvi-summarize
- anvi-trnaseq
- anvi-update-db-description
- anvi-update-structure-database
- anvi-upgrade

24.4 Module

You can load the modules by:

```
module load biocontainers
module load anvio
```

24.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Anvio on our our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
```

(continues on next page)

(continued from previous page)

```
#SBATCH -N 1
#SBATCH -n 8
#SBATCH --job-name=anvio
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers anvio

anvi-script-reformat-fasta assembly.fa -o contigs.fa -l 1000 --simplify-names  --seq-
↪type NT
anvi-gen-contigs-database -f contigs.fa -o contigs.db -n 'An example contigs database' --
↪num-threads 8
anvi-display-contigs-stats contigs.db
anvi-setup-ncbi-cogs --cog-data-dir $PWD --num-threads 8 --just-do-it --reset
anvi-run-ncbi-cogs -c contigs.db --cog-data-dir COG20 --num-threads 8
```

CHAPTER

TWENTYFIVE

ANY2FASTA

25.1 Introduction

Any2fasta can convert various sequence formats to FASTA.

For more information, please check:
BioContainers: https://biocontainers.pro/tools/any2fasta
Home page: https://github.com/tseemann/any2fasta

25.2 Versions

- 0.4.2

25.3 Commands

- any2fasta

25.4 Module

You can load the modules by:

```
module load biocontainers
module load any2fasta
```

25.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run any2fasta on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=any2fasta
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers any2fasta

any2fasta input.gff > out.fasta
```

CHAPTER

TWENTYSIX

ARCS

26.1 Introduction

ARCS is a tool for scaffolding genome sequence assemblies using linked or long read sequencing data.

For more information, please check:
Home page: https://github.com/bcgsc/arcs

26.2 Versions

- 1.2.4

26.3 Commands

- arcs
- arcs-make

26.4 Module

You can load the modules by:

```
module load biocontainers
module load arcs
```

26.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run arcs on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=arcs
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers arcs
```

CHAPTER

TWENTYSEVEN

ASGAL

27.1 Introduction

`ASGAL` (Alternative Splicing Graph ALigner) is a tool for detecting the alternative splicing events expressed in a RNA-Seq sample with respect to a gene annotation.

For more information, please check its | Docker hub: https://hub.docker.com/r/algolab/asgal and its home page on Github.

27.2 Versions

- 1.1.7

27.3 Commands

- asgal

27.4 Module

You can load the modules by:

```
module load biocontainers
module load asgal
```

27.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run ASGAL on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=asgal
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers asgal

asgal -g input/genome.fa \
    -a input/annotation.gtf \
    -s input/sample_1.fa -o outputFolder
```

CHAPTER

TWENTYEIGHT

ASSEMBLY-STATS

28.1 Introduction

`Assembly-stats` is a tool to get assembly statistics from FASTA and FASTQ files.

For more information, please check its website: https://biocontainers.pro/tools/assembly-stats and its home page on Github.

28.2 Versions

- 1.0.1

28.3 Commands

- assembly-stats

28.4 Module

You can load the modules by:

```
module load biocontainers
module load assembly-stats
```

28.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Assembly-stats on our our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 00:10:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=assembly-stats
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers assembly-stats

assembly-stats seq.fasta
```

CHAPTER

TWENTYNINE

ATAC-SEQ-PIPELINE

29.1 Introduction

The ENCODE ATAC-seq pipeline is used for quality control and statistical signal processing of short-read sequencing data, producing alignments and measures of enrichment. It was developed by Anshul Kundaje's lab at Stanford University.

For more information, please check:
Docker hub: https://hub.docker.com/r/encodedcc/atac-seq-pipeline
Home page: https://www.encodeproject.org/atac-seq/

29.2 Versions

- 2.1.3

29.3 Commands

- 10x_bam2fastq
- SAMstats
- SAMstatsParallel
- ace2sam
- aggregate_scores_in_intervals.py
- align_print_template.py
- alignmentSieve
- annotate.py
- annotateBed
- axt_extract_ranges.py
- axt_to_fasta.py
- axt_to_lav.py
- axt_to_maf.py

- bamCompare
- bamCoverage
- bamPEFragmentSize
- bamToBed
- bamToFastq
- bed12ToBed6
- bedToBam
- bedToIgv
- bed_bigwig_profile.py
- bed_build_windows.py
- bed_complement.py
- bed_count_by_interval.py
- bed_count_overlapping.py
- bed_coverage.py
- bed_coverage_by_interval.py
- bed_diff_basewise_summary.py
- bed_extend_to.py
- bed_intersect.py
- bed_intersect_basewise.py
- bed_merge_overlapping.py
- bed_rand_intersect.py
- bed_subtract_basewise.py
- bedpeToBam
- bedtools
- bigwigCompare
- blast2sam.pl
- bnMapper.py
- bowtie2sam.pl
- bwa
- chardetect
- closestBed
- clusterBed
- complementBed
- compress
- computeGCBias
- computeMatrix

- computeMatrixOperations
- correctGCBias
- coverageBed
- createDiff
- cutadapt
- cygdb
- cython
- cythonize
- deeptools
- div_snp_table_chr.py
- download_metaseq_example_data.py
- estimateReadFiltering
- estimateScaleFactor
- expandCols
- export2sam.pl
- faidx
- fastaFromBed
- find_in_sorted_file.py
- flankBed
- gene_fourfold_sites.py
- genomeCoverageBed
- getOverlap
- getSeq_genome_wN
- getSeq_genome_woN
- get_objgraph
- get_scores_in_intervals.py
- gffutils-cli
- groupBy
- gsl-config
- gsl-histogram
- gsl-randist
- idr
- int_seqs_to_char_strings.py
- interpolate_sam.pl
- intersectBed
- intersection_matrix.py

- interval_count_intersections.py
- interval_join.py
- intron_exon_reads.py
- jsondiff
- lav_to_axt.py
- lav_to_maf.py
- line_select.py
- linksBed
- lzop_build_offset_table.py
- mMK_bitset.py
- macs2
- maf_build_index.py
- maf_chop.py
- maf_chunk.py
- maf_col_counts.py
- maf_col_counts_all.py
- maf_count.py
- maf_covered_ranges.py
- maf_covered_regions.py
- maf_div_sites.py
- maf_drop_overlapping.py
- maf_extract_chrom_ranges.py
- maf_extract_ranges.py
- maf_extract_ranges_indexed.py
- maf_filter.py
- maf_filter_max_wc.py
- maf_gap_frequency.py
- maf_gc_content.py
- maf_interval_alignibility.py
- maf_limit_to_species.py
- maf_mapping_word_frequency.py
- maf_mask_cpg.py
- maf_mean_length_ungapped_piece.py
- maf_percent_columns_matching.py
- maf_percent_identity.py
- maf_print_chroms.py

- maf_print_scores.py
- maf_randomize.py
- maf_region_coverage_by_src.py
- maf_select.py
- maf_shuffle_columns.py
- maf_species_in_all_files.py
- maf_split_by_src.py
- maf_thread_for_species.py
- maf_tile.py
- maf_tile_2.py
- maf_tile_2bit.py
- maf_to_axt.py
- maf_to_concat_fasta.py
- maf_to_fasta.py
- maf_to_int_seqs.py
- maf_translate_chars.py
- maf_truncate.py
- maf_word_frequency.py
- makeBAM.sh
- makeDiff.sh
- makeFastq.sh
- make_unique
- makepBAM_genome.sh
- makepBAM_transcriptome.sh
- mapBed
- maq2sam-long
- maq2sam-short
- maskFastaFromBed
- mask_quality.py
- mergeBed
- metaseq-cli
- multiBamCov
- multiBamSummary
- multiBigwigSummary
- multiIntersectBed
- nib_chrom_intervals_to_fasta.py

- nib_intervals_to_fasta.py
- nib_length.py
- novo2sam.pl
- nucBed
- one_field_per_line.py
- out_to_chain.py
- pairToBed
- pairToPair
- pbam2bam
- pbam_mapped_transcriptome
- pbt_plotting_example.py
- peak_pie.py
- plot-bamstats
- plotCorrelation
- plotCoverage
- plotEnrichment
- plotFingerprint
- plotHeatmap
- plotPCA
- plotProfile
- prefix_lines.py
- pretty_table.py
- print_unique
- psl2sam.pl
- py.test
- pybabel
- pybedtools
- pygmentize
- pytest
- python-argcomplete-check-easy-install-script
- python-argcomplete-tcsh
- qv_to_bqv.py
- randomBed
- random_lines.py
- register-python-argcomplete
- sam2vcf.pl

- samtools
- samtools.pl
- seq_cache_populate.pl
- shiftBed
- shuffleBed
- slopBed
- soap2sam.pl
- sortBed
- speedtest.py
- subtractBed
- table_add_column.py
- table_filter.py
- tagBam
- tfloc_summary.py
- ucsc_gene_table_to_intervals.py
- undill
- unionBedGraphs
- varfilter.py
- venn_gchart.py
- venn_mpl.py
- wgsim
- wgsim_eval.pl
- wiggle_to_array_tree.py
- wiggle_to_binned_array.py
- wiggle_to_chr_binned_array.py
- wiggle_to_simple.py
- windowBed
- windowMaker
- zoom2sam.pl

29.4 Module

You can load the modules by:

```
module load biocontainers
module load atac-seq-pipeline
```

29.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run atac-seq-pipeline on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=atac-seq-pipeline
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers atac-seq-pipeline
```

CHAPTER

THIRTY

ATAQV

30.1 Introduction

`Ataqv` is a toolkit for measuring and comparing ATAC-seq results, made in the Parker lab at the University of Michigan.

For more information, please check its website: https://biocontainers.pro/tools/ataqv and its home page on Github.

30.2 Versions

- 1.3.0-py39

30.3 Commands

- ataqv

30.4 Module

You can load the modules by:

```
module load biocontainers
module load ataqv
```

30.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Ataqv on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=ataqv
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers ataqv

ataqv --peak-file sample_1_peaks.broadPeak \
    --name sample_1 --metrics-file sample_1.ataqv.json.gz \
    --excluded-region-file hg19.blacklist.bed.gz \
    --tss-file hg19.tss.refseq.bed.gz \
    --ignore-read-groups human sample_1.md.bam \
     > sample_1.ataqv.out

ataqv --peak-file sample_2_peaks.broadPeak \
    --name sample_2 --metrics-file sample_2.ataqv.json.gz \
    --excluded-region-file hg19.blacklist.bed.gz \
    --tss-file hg19.tss.refseq.bed.gz \
    --ignore-read-groups human sample_2.md.bam \
    > sample_2.ataqv.out

ataqv --peak-file sample_3_peaks.broadPeak \
    --name sample_3 --metrics-file sample_3.ataqv.json.gz \
    --excluded-region-file hg19.blacklist.bed.gz \
    --tss-file hg19.tss.refseq.bed.gz \
    --ignore-read-groups human sample_3.md.bam \
     > sample_3.ataqv.out

mkarv my_fantastic_experiment sample_1.ataqv.json.gz sample_2.ataqv.json.gz sample_3.
↪ataqv.json.gz
```

CHAPTER

THIRTYONE

ATRAM

31.1 Introduction

aTRAM (automated target restricted assembly method) is an iterative assembler that performs reference-guided local de novo assemblies using a variety of available methods.

Detailed usage can be found here: https://bioinformaticshome.com/tools/wga/descriptions/aTRAM.html

31.2 Versions

- 2.4.3

31.3 Commands

- atram.py
- atram_preprocessor.py
- atram_stitcher.py

31.4 Module

You can load the modules by:

```
module load biocontainers
module load atram/2.4.3
```

31.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run aTRAM on our our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 20:00:00
#SBATCH -N 1
#SBATCH -n 24
#SBATCH --job-name=atram
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers atram/2.4.3a

atram_preprocessor.py --blast-db=atram_db  \
                      --end-1=data/tutorial_end_1.fasta.gz \
                      --end-2=data/tutorial_end_2.fasta.gz \
                      --gzip
atram.py --query=tutorial-query.pep.fasta  \
         --blast-db=atram_db \
         --output=output \
         --assembler=velvet
```

CHAPTER

THIRTYTWO

ATROPOS

32.1 Introduction

`Atropos` is a tool for specific, sensitive, and speedy trimming of NGS reads.

For more information, please check its website: https://biocontainers.pro/tools/atropos and its home page on Github.

32.2 Versions

- 1.1.17
- 1.1.31

32.3 Commands

- atropos

32.4 Module

You can load the modules by:

```
module load biocontainers
module load atropos
```

32.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Atropos on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 4
#SBATCH --job-name=atropos
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers atropos

atropos --threads 4  \
    -a AGATCGGAAGAGCACACGTCTGAACTCCAGTCACGAGTTA \
    -o trimmed1.fq.gz -p trimmed2.fq.gz \
    -pe1 SRR13176582_1.fastq -pe2 SRR13176582_2.fastq
```

CHAPTER

THIRTYTHREE

AUGUR

33.1 Introduction

`Augur` is the bioinformatics toolkit we use to track evolution from sequence and serological data.

For more information, please check its website: https://biocontainers.pro/tools/augur and its home page on Github.

33.2 Versions

- 14.0.0
- 15.0.0

33.3 Commands

- augur

33.4 Module

You can load the modules by:

```
module load biocontainers
module load augur
```

33.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Augur on our our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=augur
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers augur

mkdir -p results
augur index --sequences zika-tutorial/data/sequences.fasta \
            --output results/sequence_index.tsv

augur filter --sequences zika-tutorial/data/sequences.fasta \
             --sequence-index results/sequence_index.tsv \
             --metadata  zika-tutorial/data/metadata.tsv \
             --exclude zika-tutorial/config/dropped_strains.txt \
             --output results/filtered.fasta \
             --group-by country year month \
             --sequences-per-group 20 \
             --min-date 2012

augur align --sequences results/filtered.fasta \
            --reference-sequence zika-tutorial/config/zika_outgroup.gb \
            --output results/aligned.fasta \
            --fill-gaps

augur tree --alignment results/aligned.fasta \
           --output results/tree_raw.nwk

augur refine --tree results/tree_raw.nwk \
             --alignment results/aligned.fasta \
             --metadata  zika-tutorial/data/metadata.tsv \
             --output-tree results/tree.nwk \
             --output-node-data results/branch_lengths.json \
             --timetree \
             --coalescent opt \
             --date-confidence \
             --date-inference marginal \
             --clock-filter-iqd 4
```

CHAPTER

THIRTYFOUR

AUGUSTUS

34.1 Introduction

AUGUSTUS is a program that predicts genes in eukaryotic genomic sequences.

For more information, please check its website: https://bioinf.uni-greifswald.de/augustus/.

34.2 Versions

- 3.4.0

34.3 Commands

- aln2wig
- augustus
- bam2wig
- bam2wig-dist
- consensusFinder
- curve2hints
- etraining
- fastBlockSearch
- filterBam
- getSeq
- getSeq-dist
- homGeneMapping
- joingenes
- prepareAlign

34.4 Module

You can load the modules by:

```
module load biocontainers
module load augustus/3.4.0
```

34.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run AUGUSTUS on our cluster:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 10:00:00
#SBATCH -N 1
#SBATCH -n 24
#SBATCH --job-name=AUGUSTUS
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers augustus/3.4.0

augustus --species=botrytis_cinerea genome.fasta > annotation.gff
```

CHAPTER

THIRTYFIVE

BACTOPIA

35.1 Introduction

Bactopia is a flexible pipeline for complete analysis of bacterial genomes. The goal of Bactopia is to process your data with a broad set of tools, so that you can get to the fun part of analyses quicker!

For more information, please check:
Docker hub: https://hub.docker.com/r/bactopia/bactopia
Home page: https://github.com/bactopia/bactopia

35.2 Versions

- 2.0.3

35.3 Commands

- bactopia

35.4 Module

You can load the modules by:

```
module load biocontainers
module load bactopia
```

35.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run bactopia on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 12
#SBATCH --job-name=bactopia
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers bactopia

bactopia datasets \
--ariba "vfdb_core,card" \
--species "Staphylococcus aureus" \
--include_genus \
--limit 100 \
--cpus 12

bactopia --accession SRX4563634 \
--datasets datasets/ \
--species "Staphylococcus aureus" \
--coverage 100 \
--genome_size median \
--outdir ena-single-sample \
--max_cpus 12
```

CHAPTER

THIRTYSIX

BALI-PHY

36.1 Introduction

Bali-phy is a tool for bayesian co-estimation of phylogenies and multiple alignments via MCMC.

For more information, please check:
BioContainers: https://biocontainers.pro/tools/bali-phy
Home page: https://github.com/bredelings/BAli-Phy

36.2 Versions

- 3.6.0

36.3 Commands

- bali-phy

36.4 Module

You can load the modules by:

```
module load biocontainers
module load bali-phy
```

36.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run bali-phy on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=bali-phy
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers bali-phy

bali-phy examples/sequences/ITS/ITS1.fasta 5.8S.fasta ITS2.fasta --test
bali-phy examples/sequences/5S-rRNA/5d-clustalw.fasta -S gtr+Rates.gamma[4]+inv -n 5d-
↪free
```

CHAPTER

THIRTYSEVEN

BAMGINEER

37.1 Introduction

`Bamgineer` is a tool that can be used to introduce user-defined haplotype-phased allele-specific copy number variations (CNV) into an existing Binary Alignment Mapping (BAM) file with demonstrated applicability to simulate somatic cancer CNVs in phased whole-genome sequencing datsets.

For more information, please check its | Docker hub: https://hub.docker.com/r/suluxan/bamgineer-v2 and its home page on Github.

37.2 Versions

- 1.1

37.3 Commands

- simulate.py

37.4 Module

You can load the modules by:

```
module load biocontainers
module load bamgineer
```

37.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Bamgineer on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=bamgineer
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers bamgineer

simulate.py -config inputs/config.cfg \
            -splitbamdir splitbams \
            -cnv_bed inputs/cnv.bed \
            -vcf inputs/normal_het.vcf \
            -exons inputs/exons.bed \
            -outbam tumour.bam \
            -results outputs \
            -cancertype LUAC1
```

CHAPTER

THIRTYEIGHT

BAMLIQUIDATOR

38.1 Introduction

`Bamliquidator` is a set of tools for analyzing the density of short DNA sequence read alignments in the BAM file format.

For more information, please check its | Docker hub: https://hub.docker.com/r/bioliquidator/bamliquidator/ and its home page on Github.

38.2 Versions

- 1.5.2

38.3 Commands

- bamliquidator
- bamliquidator_bins
- bamliquidator_regions
- bamliquidatorbatch

38.4 Module

You can load the modules by:

```
module load biocontainers
module load bamliquidator
```

38.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Bamliquidator on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=bamliquidator
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers bamliquidator
```

CHAPTER

THIRTYNINE

BAM-READCOUNT

39.1 Introduction

`Bam-readcount` is a utility that runs on a BAM or CRAM file and generates low-level information about sequencing data at specific nucleotide positions.

For more information, please check its | Docker hub: https://hub.docker.com/r/mgibio/bam-readcount and its home page on Github.

39.2 Versions

- 1.0.0

39.3 Commands

- bam-readcount

39.4 Module

You can load the modules by:

```
module load biocontainers
module load bam-readcount
```

39.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Bam-readcount on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=bam-readcount
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers bam-readcount

bam-readcount -f Homo_sapiens.GRCh38.dna.primary_assembly.fa Aligned.sortedByCoord.out.
↪bam
```

CHAPTER

FORTY

BAMSURGEON

40.1 Introduction

`Bamsurgeon` are tools for adding mutations to .bam files, used for testing mutation callers.

For more information, please check its | Docker hub: https://hub.docker.com/r/lethalfang/bamsurgeon and its home page on Github.

40.2 Versions

- 1.2

40.3 Commands

- addindel.py
- addsnv.py
- addsv.py

40.4 Module

You can load the modules by:

```
module load biocontainers
module load bamsurgeon
```

40.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Bamsurgeon on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=bamsurgeon
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers bamsurgeon

addsv.py -p 1 -v test_sv.txt -f testregion_realign.bam \
    -r reference.fasta -o testregion_sv_mut.bam \
    --aligner mem --keepsecondary --seed 1234 \
    --inslib test_inslib.fa
```

CHAPTER

FORTYONE

BAMTOOLS

41.1 Introduction

`BamTools` is a programmer API and an end-user toolkit for handling BAM files. This container provides a toolkit-only version (no API to build against).

For more information, please check its website: https://biocontainers.pro/tools/bamtools and its home page on Github.

41.2 Versions

- 2.5.1

41.3 Commands

- bamtools

41.4 Module

You can load the modules by:

```
module load biocontainers
module load bamtools
```

41.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run BamTools on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=bamtools
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH -ddd-error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers bamtools

bamtools convert -format fastq -in in.bam -out out.fastq
```

CHAPTER

FORTYTWO

BAMUTIL

42.1 Introduction

`Bamutil` is a collection of programs for working on SAM/BAM files.

For more information, please check its website: https://biocontainers.pro/tools/bamutil and its home page on Github.

42.2 Versions

- 1.0.15

42.3 Commands

- bam

42.4 Module

You can load the modules by:

```
module load biocontainers
module load bamutil
```

42.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Bamutil on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=bamutil
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers bamutil

bam validate --params --in test/testFiles/testInvalid.sam --refFile test/testFilesLibBam/
↪chr1_partial.fa --v --noph 2> results/validateInvalid.txt

bam convert --params --in test/testFiles/testFilter.bam --out results/convertBam.sam --
↪noph 2> results/convertBam.log

bam  splitChromosome --in test/testFile/sortedBam1.bam --out results/splitSortedBam --
↪noph 2> results/splitChromosome.txt

bam stats --basic --in test/testFiles/testFilter.sam --noph 2> results/basicStats.txt

bam gapInfo --in test/testFiles/testGapInfo.sam --out results/gapInfo.txt --noph 2>␣
↪results/gapInfo.log

bam findCigars --in test/testFiles/testRevert.sam --out results/cigarNonM.sam --nonM --
↪noph 2> results/cigarNonM.log
```

CHAPTER

FORTYTHREE

BARRNAP

43.1 Introduction

`Barrnap`: BAsic Rapid Ribosomal RNA Predictor.

For more information, please check its website: https://biocontainers.pro/tools/barrnap and its home page on Github.

43.2 Versions

- 0.9.4

43.3 Commands

- barrnap

43.4 Module

You can load the modules by:

```
module load biocontainers
module load barrnap
```

43.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Barrnap on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=barrnap
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers barrnap

barrnap --kingdom bac -o bac_16s.fasta < bac_genome.fasta > bac_16s.gff3
barrnap --kingdom euk -o euk_16s.fasta < euk_genome.fasta  > euk_16s.gff3
```

CHAPTER

FORTYFOUR

BASENJI

44.1 Introduction

`Basenji` is a tool for sequential regulatory activity predictions with deep convolutional neural networks.

For more information, please check its website: https://biocontainers.pro/tools/basenji and its home page on Github.

44.2 Versions

- 0.5.1

44.3 Commands

- akita_data.py
- akita_data_read.py
- akita_data_write.py
- akita_predict.py
- akita_sat_plot.py
- akita_sat_vcf.py
- akita_scd.py
- akita_scd_multi.py
- akita_test.py
- akita_train.py
- bam_cov.py
- basenji_annot_chr.py
- basenji_bench_classify.py
- basenji_bench_gtex.py
- basenji_bench_gtex_cmp.py
- basenji_bench_phylop.py

- basenji_bench_phylop_folds.py
- basenji_cmp.py
- basenji_data.py
- basenji_data2.py
- basenji_data_align.py
- basenji_data_gene.py
- basenji_data_hic_read.py
- basenji_data_hic_write.py
- basenji_data_read.py
- basenji_data_write.py
- basenji_fetch_app.py
- basenji_fetch_app1.py
- basenji_fetch_app2.py
- basenji_fetch_norm.py
- basenji_fetch_vcf.py
- basenji_gtex_folds.py
- basenji_hdf5_genes.py
- basenji_hidden.py
- basenji_map.py
- basenji_map_genes.py
- basenji_map_seqs.py
- basenji_motifs.py
- basenji_motifs_denovo.py
- basenji_norm_h5.py
- basenji_predict.py
- basenji_predict_bed.py
- basenji_predict_bed_multi.py
- basenji_sad.py
- basenji_sad_multi.py
- basenji_sad_norm.py
- basenji_sad_ref.py
- basenji_sad_ref_multi.py
- basenji_sad_table.py
- basenji_sat_bed.py
- basenji_sat_bed_multi.py
- basenji_sat_folds.py

- basenji_sat_plot.py
- basenji_sat_plot2.py
- basenji_sat_vcf.py
- basenji_sed.py
- basenji_sed_multi.py
- basenji_sedg.py
- basenji_test.py
- basenji_test_folds.py
- basenji_test_genes.py
- basenji_test_reps.py
- basenji_test_specificity.py
- basenji_train.py
- basenji_train1.py
- basenji_train2.py
- basenji_train_folds.py
- basenji_train_hic.py
- basenji_train_reps.py
- save_model.py
- sonnet_predict_bed.py
- sonnet_sad.py
- sonnet_sad_multi.py
- sonnet_sat_bed.py
- sonnet_sat_vcf.py
- tfr_bw.py
- tfr_hdf5.py
- tfr_qc.py
- upgrade_tf1.py

44.4 Module

You can load the modules by:

```
module load biocontainers
module load basenji
```

44.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Basenji on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=basenji
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers basenji
```

CHAPTER

FORTYFIVE

BBMAP

45.1 Introduction

`Bbmap` is a short read aligner, as well as various other bioinformatic tools.

For more information, please check its website: https://biocontainers.pro/tools/bbmap and its home page on Sourceforge.

45.2 Versions

- 38.93
- 38.96

45.3 Commands

- addadapters.sh
- a_sample_mt.sh
- bbcountunique.sh
- bbduk.sh
- bbest.sh
- bbfakereads.sh
- bbmap.sh
- bbmapskimmer.sh
- bbmask.sh
- bbmerge-auto.sh
- bbmergegapped.sh
- bbmerge.sh
- bbnorm.sh
- bbqc.sh

- bbrealign.sh
- bbrename.sh
- bbsketch.sh
- bbsplitpairs.sh
- bbsplit.sh
- bbstats.sh
- bbversion.sh
- bbwrap.sh
- calcmem.sh
- calctruequality.sh
- callpeaks.sh
- callvariants2.sh
- callvariants.sh
- clumpify.sh
- commonkmers.sh
- compare sketch.sh
- comparevcf.sh
- consect.sh
- countbarcodes.sh
- countgc.sh
- countsharedlines.sh
- crossblock.sh
- crosscontaminate.sh
- cutprimers.sh
- decontaminate.sh
- dedupe2.sh
- dedupebymapping.sh
- dedupe.sh
- demuxbyname.sh
- diskbench.sh
- estherfilter.sh
- explodetree.sh
- filterassemblysummary.sh
- filterbarcodes.sh
- filterbycoverage.sh
- filterbyname.sh

- filterbysequence.sh
- filterbytaxa.sh
- filterbytile.sh
- filterlines.sh
- filtersam.sh
- filtersubs.sh
- filtervcf.sh
- fungalrelease.sh
- fuse.sh
- getreads.sh
- gi2ancestors.sh
- gi2taxid.sh
- gitable.sh
- grademerge.sh
- gradesam.sh
- idmatrix.sh
- idtree.sh
- invertkey.sh
- kcompress.sh
- khist.sh
- kmercountexact.sh
- kmercountmulti.sh
- kmercoverage.sh
- loadreads.sh
- loglog.sh
- makechimeras.sh
- makecontaminatedgenomes.sh
- makepolymers.sh
- mapPacBio.sh
- matrixtocolumns.sh
- mergebarcodes.sh
- mergeOTUs.sh
- mergesam.sh
- msa.sh
- mutate.sh
- muxbyname.sh

- normandcorrectwrapper.sh
- partition.sh
- phylip2fasta.sh
- pileup.sh
- plotgc.sh
- postfilter.sh
- printtime.sh
- processfrag.sh
- processspeed.sh
- randomreads.sh
- readlength.sh
- reducesilva.sh
- reformat.sh
- removebadbarcodes.sh
- removecatdogmousehuman.sh
- removehuman2.sh
- removehuman.sh
- removemicrobes.sh
- removesmartbell.sh
- renameimg.sh
- rename.sh
- repair.sh
- replaceheaders.sh
- representative.sh
- rqcfilter.sh
- samtoroc.sh
- seal.sh
- sendsketch.sh
- shred.sh
- shrinkaccession.sh
- shuffle.sh
- sketchblacklist.sh
- sketch.sh
- sortbyname.sh
- splitbytaxa.sh
- splitnextera.sh

- splitsam4way.sh
- splitsam6way.sh
- splitsam.sh
- stats.sh
- statswrapper.sh
- streamsam.sh
- summarizecrossblock.sh
- summarizemerge.sh
- summarizequast.sh
- summarizescafstats.sh
- summarizeseal.sh
- summarizesketch.sh
- synthmda.sh
- tadpipe.sh
- tadpole.sh
- tadwrapper.sh
- taxonomy.sh
- taxserver.sh
- taxsize.sh
- taxtree.sh
- testfilesystem.sh
- testformat2.sh
- testformat.sh
- tetramerfreq.sh
- textfile.sh
- translate6frames.sh
- unicode2ascii.sh
- webcheck.sh

45.4 Module

You can load the modules by:

```
module load biocontainers
module load bbmap
```

45.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Bbmap on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=bbmap
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers bbmap

stats.sh in=SRR11234553_1.fastq > stats_out.txt
statswrapper.sh *.fastq > statswrapper_out.txt
pileup.sh in=map1.sam out=pileup_out.txt
readlength.sh in=SRR11234553_1.fastq in2=SRR11234553_2.fastq > readlength_out.txt
kmercountexact.sh in=SRR11234553_1.fastq in2=SRR11234553_2.fastq out=kmer_test.out␣
↪khist=kmer.khist peaks=kmer.peak
bbmask.sh in=SRR11234553_1.fastq out=test.mark sam=map1.sam
```

CHAPTER

FORTYSIX

BBTOOLS

46.1 Introduction

BBTools is a suite of fast, multithreaded bioinformatics tools designed for analysis of DNA and RNA sequence data.

For more information, please check:
Docker hub: https://hub.docker.com/r/staphb/bbtools
Home page: https://jgi.doe.gov/data-and-tools/software-tools/bbtools/

46.2 Versions

- 39.00

46.3 Commands

- Xcalcmem.sh
- a_sample_mt.sh
- addadapters.sh
- addssu.sh
- adjusthomopolymers.sh
- alltoall.sh
- analyzeaccession.sh
- analyzegenes.sh
- analyzesketchresults.sh
- applyvariants.sh
- bbcms.sh
- bbcountunique.sh
- bbduk.sh
- bbest.sh

- bbfakereads.sh
- bbmap.sh
- bbmapskimmer.sh
- bbmask.sh
- bbmerge-auto.sh
- bbmerge.sh
- bbnorm.sh
- bbrealign.sh
- bbrename.sh
- bbsketch.sh
- bbsplit.sh
- bbsplitpairs.sh
- bbstats.sh
- bbversion.sh
- bbwrap.sh
- bloomfilter.sh
- calcmem.sh
- calctruequality.sh
- callgenes.sh
- callpeaks.sh
- callvariants.sh
- callvariants2.sh
- clumpify.sh
- commonkmers.sh
- comparegff.sh
- compareSketch.sh
- comparessu.sh
- comparevcf.sh
- consect.sh
- consensus.sh
- countbarcodes.sh
- countgc.sh
- countsharedlines.sh
- crossblock.sh
- crosscontaminate.sh
- cutgff.sh

- cutprimers.sh
- decontaminate.sh
- dedupe.sh
- dedupe2.sh
- dedupebymapping.sh
- demuxbyname.sh
- diskbench.sh
- estherfilter.sh
- explodetree.sh
- fetchproks.sh
- filterassemblysummary.sh
- filterbarcodes.sh
- filterbycoverage.sh
- filterbyname.sh
- filterbysequence.sh
- filterbytaxa.sh
- filterbytile.sh
- filterlines.sh
- filterqc.sh
- filtersam.sh
- filtersilva.sh
- filtersubs.sh
- filtervcf.sh
- fixgaps.sh
- fungalrelease.sh
- fuse.sh
- gbff2gff.sh
- getreads.sh
- gi2ancestors.sh
- gi2taxid.sh
- gitable.sh
- grademerge.sh
- gradesam.sh
- icecreamfinder.sh
- icecreamgrader.sh
- icecreammaker.sh

- idmatrix.sh
- idtree.sh
- invertkey.sh
- kapastats.sh
- kcompress.sh
- keepbestcopy.sh
- khist.sh
- kmercountexact.sh
- kmercountmulti.sh
- kmercoverage.sh
- kmerfilterset.sh
- kmerlimit.sh
- kmerlimit2.sh
- kmerposition.sh
- kmutate.sh
- lilypad.sh
- loadreads.sh
- loglog.sh
- makechimeras.sh
- makecontaminatedgenomes.sh
- makepolymers.sh
- mapPacBio.sh
- matrixtocolumns.sh
- mergeOTUs.sh
- mergebarcodes.sh
- mergepgm.sh
- mergeribo.sh
- mergesam.sh
- mergesketch.sh
- mergesorted.sh
- msa.sh
- mutate.sh
- muxbyname.sh
- partition.sh
- phylip2fasta.sh
- pileup.sh

- plotflowcell.sh
- plotgc.sh
- postfilter.sh
- printtime.sh
- processfrag.sh
- processhi-c.sh
- processspeed.sh
- randomgenome.sh
- randomreads.sh
- readlength.sh
- readqc.sh
- reducesilva.sh
- reformat.sh
- reformatpb.sh
- removebadbarcodes.sh
- removecatdogmousehuman.sh
- removehuman.sh
- removehuman2.sh
- removemicrobes.sh
- removesmartbell.sh
- rename.sh
- renameimg.sh
- repair.sh
- replaceheaders.sh
- representative.sh
- rqcfilter.sh
- rqcfilter2.sh
- runhmm.sh
- samtoroc.sh
- seal.sh
- sendsketch.sh
- shred.sh
- shrinkaccession.sh
- shuffle.sh
- shuffle2.sh
- sketch.sh

- sketchblacklist.sh
- sketchblacklist2.sh
- sortbyname.sh
- splitbytaxa.sh
- splitnextera.sh
- splitribo.sh
- splitsam.sh
- splitsam4way.sh
- splitsam6way.sh
- stats.sh
- statswrapper.sh
- streamsam.sh
- subsketch.sh
- summarizecontam.sh
- summarizecoverage.sh
- summarizecrossblock.sh
- summarizemerge.sh
- summarizequast.sh
- summarizescafstats.sh
- summarizeseal.sh
- summarizesketch.sh
- synthmda.sh
- tadpipe.sh
- tadpole.sh
- tadwrapper.sh
- taxonomy.sh
- taxserver.sh
- taxsize.sh
- taxtree.sh
- testfilesystem.sh
- testformat.sh
- testformat2.sh
- tetramerfreq.sh
- textfile.sh
- translate6frames.sh
- unicode2ascii.sh

- unzip.sh
- vcf2gff.sh
- webcheck.sh

46.4 Module

You can load the modules by:

```
module load biocontainers
module load bbtools
```

46.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run bbtools on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=bbtools
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers bbtools
```

CHAPTER

FORTYSEVEN

BCFTOOLS

47.1 Introduction

`Bcftools` is a program for variant calling and manipulating files in the Variant Call Format (VCF) and its binary counterpart BCF.

For more information, please check its website: https://biocontainers.pro/tools/bcftools and its home page on Github.

47.2 Versions

- 1.13
- 1.14

47.3 Commands

- bcftools
- color-chrs.pl
- guess-ploidy.py
- plot-roh.py
- plot-vcfstats
- run-roh.pl
- vcfutils.pl

47.4 Module

You can load the modules by:

```
module load biocontainers
module load bcftools
```

47.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Bcftools on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=bcftools
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers bcftools

bcftools query -f '%CHROM %POS %REF %ALT\n' file.bcf
bcftools polysomy -v -o outdir/ file.vcf
```

CHAPTER

FORTYEIGHT

BCL2FASTQ

48.1 Introduction

bcl2fastq Conversion Software both demultiplexes data and converts BCL files generated by Illumina sequencing systems to standard FASTQ file formats for downstream analysis.

For more information, please check:
Docker hub: https://hub.docker.com/r/gcfntnu/bcl2fastq
Home page: https://support.illumina.com/sequencing/sequencing_software/bcl2fastq-conversion-software.html

48.2 Versions

- 2.20.0

48.3 Commands

- bcl2fastq

48.4 Module

You can load the modules by:

```
module load biocontainers
module load bcl2fastq
```

48.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run bcl2fastq on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=bcl2fastq
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers bcl2fastq
```

CHAPTER

FORTYNINE

BEAGLE

49.1 Introduction

`Beagle` is a software package for phasing genotypes and for imputing ungenotyped markers. Start it with: beagle [java options] [arguments] Note: Bref is not installed in this container.

For more information, please check its website: https://biocontainers.pro/tools/beagle and its home page: https://faculty.washington.edu/browning/beagle/beagle.html.

49.2 Versions

- 5.1_24Aug19.3e8

49.3 Commands

- beagle

49.4 Module

You can load the modules by:

```
module load biocontainers
module load beagle
```

49.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Beagle on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=beagle
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers beagle

beagle gt=test.vcf.gz out=test.out
```

CHAPTER

FIFTY

BEAST 2

50.1 Introduction

`BEAST 2` is a cross-platform program for Bayesian phylogenetic analysis of molecular sequences.

For more information, please check its website: https://biocontainers.pro/tools/beast2 and its home page: https://www.beast2.org.

50.2 Versions

- 2.6.3
- 2.6.4
- 2.6.6

50.3 Commands

- applauncher
- beast
- beauti
- densitree
- loganalyser
- logcombiner
- packagemanager
- treeannotator

50.4 Module

You can load the modules by:

```
module load biocontainers
module load beast2
```

50.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run BEAST 2 on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 4
#SBATCH --job-name=beast2
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers beast2

beast -threads 4 -prefix input input.xml
```

CHAPTER

FIFTYONE

BEDOPS

51.1 Introduction

`Bedops` is a software package for manipulating and analyzing genomic interval data.

For more information, please check its website: https://biocontainers.pro/tools/bedops and its home page: https://bedops.readthedocs.io/en/latest/.

51.2 Versions

- 2.4.39

51.3 Commands

- bam2bed
- bam2bed-float128
- bam2bed_gnuParallel
- bam2bed_gnuParallel-float128
- bam2bed_gnuParallel-megarow
- bam2bed_gnuParallel-typical
- bam2bed-megarow
- bam2bed_sge
- bam2bed_sge-float128
- bam2bed_sge-megarow
- bam2bed_sge-typical
- bam2bed_slurm
- bam2bed_slurm-float128
- bam2bed_slurm-megarow
- bam2bed_slurm-typical

- bam2bed-typical
- bam2starch
- bam2starch-float128
- bam2starch_gnuParallel
- bam2starch_gnuParallel-float128
- bam2starch_gnuParallel-megarow
- bam2starch_gnuParallel-typical
- bam2starch-megarow
- bam2starch_sge
- bam2starch_sge-float128
- bam2starch_sge-megarow
- bam2starch_sge-typical
- bam2starch_slurm
- bam2starch_slurm-float128
- bam2starch_slurm-megarow
- bam2starch_slurm-typical
- bam2starch-typical
- bedextract
- bedextract-float128
- bedextract-megarow
- bedextract-typical
- bedmap
- bedmap-float128
- bedmap-megarow
- bedmap-typical
- bedops
- bedops-float128
- bedops-megarow
- bedops-typical
- closest-features
- closest-features-float128
- closest-features-megarow
- closest-features-typical
- convert2bed
- convert2bed-float128
- convert2bed-megarow

- convert2bed-typical
- gff2bed
- gff2bed-float128
- gff2bed-megarow
- gff2bed-typical
- gff2starch
- gff2starch-float128
- gff2starch-megarow
- gff2starch-typical
- gtf2bed
- gtf2bed-float128
- gtf2bed-megarow
- gtf2bed-typical
- gtf2starch
- gtf2starch-float128
- gtf2starch-megarow
- gtf2starch-typical
- gvf2bed
- gvf2bed-float128
- gvf2bed-megarow
- gvf2bed-typical
- gvf2starch
- gvf2starch-float128
- gvf2starch-megarow
- gvf2starch-typical
- psl2bed
- psl2bed-float128
- psl2bed-megarow
- psl2bed-typical
- psl2starch
- psl2starch-float128
- psl2starch-megarow
- psl2starch-typical
- rmsk2bed
- rmsk2bed-float128
- rmsk2bed-megarow

- rmsk2bed-typical
- rmsk2starch
- rmsk2starch-float128
- rmsk2starch-megarow
- rmsk2starch-typical
- sam2bed
- sam2bed-float128
- sam2bed-megarow
- sam2bed-typical
- sam2starch
- sam2starch-float128
- sam2starch-megarow
- sam2starch-typical
- sort-bed
- sort-bed-float128
- sort-bed-megarow
- sort-bed-typical
- starch
- starchcat
- starchcat-float128
- starchcat-megarow
- starchcat-typical
- starchcluster_gnuParallel
- starchcluster_gnuParallel-float128
- starchcluster_gnuParallel-megarow
- starchcluster_gnuParallel-typical
- starchcluster_sge
- starchcluster_sge-float128
- starchcluster_sge-megarow
- starchcluster_sge-typical
- starchcluster_slurm
- starchcluster_slurm-float128
- starchcluster_slurm-megarow
- starchcluster_slurm-typical
- starch-diff
- starch-diff-float128

- starch-diff-megarow
- starch-diff-typical
- starch-float128
- starch-megarow
- starchstrip
- starchstrip-float128
- starchstrip-megarow
- starchstrip-typical
- starch-typical
- switch-BEDOPS-binary-type
- unstarch
- unstarch-float128
- unstarch-megarow
- unstarch-typical
- update-sort-bed-migrate-candidates
- update-sort-bed-migrate-candidates-float128
- update-sort-bed-migrate-candidates-megarow
- update-sort-bed-migrate-candidates-typical
- update-sort-bed-slurm
- update-sort-bed-slurm-float128
- update-sort-bed-slurm-megarow
- update-sort-bed-slurm-typical
- update-sort-bed-starch-slurm
- update-sort-bed-starch-slurm-float128
- update-sort-bed-starch-slurm-megarow
- update-sort-bed-starch-slurm-typical
- vcf2bed
- vcf2bed-float128
- vcf2bed-megarow
- vcf2bed-typical
- vcf2starch
- vcf2starch-float128
- vcf2starch-megarow
- vcf2starch-typical
- wig2bed
- wig2bed-float128

- wig2bed-megarow
- wig2bed-typical
- wig2starch
- wig2starch-float128
- wig2starch-megarow
- wig2starch-typical

51.4 Module

You can load the modules by:

```
module load biocontainers
module load bedops
```

51.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Bedops on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=bedops
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers bedops

bedops -m 001.merge.001.test > 001.merge.001.observed
bedops -c 001.merge.001.test > 001.complement.001.observed
bedops -i 001.intersection.001a.test 001.intersection.001b.test > 001.intersection.001.
↪observed
```

CHAPTER

FIFTYTWO

BEDTOOLS

52.1 Introduction

`Bedtools` is an extensive suite of utilities for genome arithmetic and comparing genomic features in BED format.

For more information, please check its website: https://biocontainers.pro/tools/bedtools and its home page on Github.

52.2 Versions

- 2.30.0

52.3 Commands

- annotateBed
- bamToBed
- bamToFastq
- bed12ToBed6
- bedpeToBam
- bedToBam
- bedToIgv
- bedtools
- closestBed
- clusterBed
- complementBed
- coverageBed
- expandCols
- fastaFromBed
- flankBed
- genomeCoverageBed

- getOverlap
- groupBy
- intersectBed
- linksBed
- mapBed
- maskFastaFromBed
- mergeBed
- multiBamCov
- multiIntersectBed
- nucBed
- pairToBed
- pairToPair
- randomBed
- shiftBed
- shuffleBed
- slopBed
- sortBed
- subtractBed
- tagBam
- unionBedGraphs
- windowBed
- windowMaker

52.4 Module

You can load the modules by:

```
module load biocontainers
module load bedtools
```

52.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Bedtools on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=bedtools
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers bedtools

bedtools intersect -a a.bed -b b.bed
bedtools annotate -i variants.bed -files genes.bed conserve.bed known_var.bed
```

CHAPTER

FIFTYTHREE

BIOAWK

53.1 Introduction

`Bioawk` is an extension to Brian Kernighan's awk, adding the support of several common biological data formats, including optionally gzip'ed BED, GFF, SAM, VCF, FASTA/Q and TAB-delimited formats with column names.

For more information, please check its website: https://biocontainers.pro/tools/bioawk and its home page on Github.

53.2 Versions

- 1.0

53.3 Commands

- bioawk

53.4 Module

You can load the modules by:

```
module load biocontainers
module load bioawk
```

53.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Bioawk on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=bioawk
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers bioawk

bioawk -c fastx '{print ">"$name;print revcomp($seq)}' seq.fa.gz
```

CHAPTER

FIFTYFOUR

BIOBAMBAM

54.1 Introduction

`Biobambam` is a collection of tools for early stage alignment file processing.

For more information, please check its website: https://biocontainers.pro/tools/biobambam and its home page on Gitlab.

54.2 Versions

- 2.0.183

54.3 Commands

- bam12auxmerge
- bam12split
- bam12strip
- bamadapterclip
- bamadapterfind
- bamalignfrac
- bamauxmerge
- bamauxmerge2
- bamauxsort
- bamcat
- bamchecksort
- bamclipXT
- bamclipreinsert
- bamcollate2
- bamdepth

- bamdepthintersect
- bamdifference
- bamdownsamplerandom
- bamexplode
- bamexploderef
- bamfastcat
- bamfastexploderef
- bamfastnumextract
- bamfastsplit
- bamfeaturecount
- bamfillquery
- bamfilteraux
- bamfiltereofblocks
- bamfilterflags
- bamfilterheader
- bamfilterheader2
- bamfilterk
- bamfilterlength
- bamfiltermc
- bamfilternames
- bamfilterrefid
- bamfilterrg
- bamfixmateinformation
- bamfixpairinfo
- bamflagsplit
- bamindex
- bamintervalcomment
- bamintervalcommenthist
- bammapdist
- bammarkduplicates
- bammarkduplicates2
- bammarkduplicatesopt
- bammaskflags
- bammdnm
- bammerge
- bamnumericalindex

- bamnumericalindexstats
- bamrank
- bamranksort
- bamrecalculatecigar
- bamrecompress
- bamrefextract
- bamrefinterval
- bamreheader
- bamreplacechecksums
- bamreset
- bamscrapcount
- bamseqchksum
- bamsormadup
- bamsort
- bamsplit
- bamsplitdiv
- bamstreamingmarkduplicates
- bamtofastq
- bamvalidate
- bamzztoname

54.4 Module

You can load the modules by:

```
module load biocontainers
module load biobambam
```

54.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Biobambam on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
```

(continues on next page)

(continued from previous page)

```
#SBATCH -N 1
#SBATCH -n 8
#SBATCH --job-name=biobambam
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers biobambam

bammarkduplicates I=Aligned.sortedByCoord.out.bam O=out.bam D=duplcate_out

bamsort I=Aligned.sortedByCoord.out.bam O=sorted.bam sortthreads=8

bamtofastq filename=Aligned.sortedByCoord.out.bam outputdir=fastq_out
```

CHAPTER

FIFTYFIVE

BIOCONVERT

55.1 Introduction

`Bioconvert` is a collaborative project to facilitate the interconversion of life science data from one format to another.

For more information, please check its website: https://biocontainers.pro/tools/bioconvert and its home page: https://bioconvert.readthedocs.io/en/master/.

55.2 Versions

- 0.4.3
- 0.5.2
- 0.6.1

55.3 Commands

- bioconvert

55.4 Module

You can load the modules by:

```
module load biocontainers
module load bioconvert
```

55.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Bioconvert on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=bioconvert
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers bioconvert

bioconvert fastq2fasta input.fastq output.fa
```

CHAPTER

FIFTYSIX

BIOPYTHON

56.1 Introduction

`Biopython` is a set of freely available tools for biological computation written in Python.

For more information, please check its website: https://biocontainers.pro/tools/biopython and its home page: https://biopython.org.

56.2 Versions

- 1.70-np112py27
- 1.70-np112py36
- 1.78

56.3 Commands

- easy_install
- f2py
- f2py3
- idle3
- pip
- pip3
- pydoc
- pydoc3
- python
- python3
- python3-config
- python3.9
- python3.9-config

- wheel

56.4 Module

You can load the modules by:

```
module load biocontainers
module load biopython
```

56.5 Interactive job

To run biopython interactively on our clusters:

```
(base) UserID@bell-fe00:~ $ sinteractive -N1 -n12 -t4:00:00 -A myallocation
salloc: Granted job allocation 12345869
salloc: Waiting for resource configuration
salloc: Nodes bell-a008 are ready for job
(base) UserID@bell-a008:~ $ module load biocontainers biopython
(base) UserID@bell-a008:~ $ python
Python 3.9.1 |  packaged by conda-forge |  (default, Jan 26 2021, 01:34:10)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from Bio import SeqIO
>>> with open("input.gb") as input_handle:
    for record in SeqIO.parse(input_handle, "genbank"):
          print(record)
```

56.6 Batch job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Biopython on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=biopython
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers biopython
```

(continues on next page)

(continued from previous page)

```
python script.py
```

CHAPTER

FIFTYSEVEN

BISMARK

57.1 Introduction

`Bismark` is a tool to map bisulfite treated sequencing reads to a genome of interest and perform methylation calls in a single step.

For more information, please check its website: https://biocontainers.pro/tools/bismark and its home page on Github.

57.2 Versions

- 0.23.0

57.3 Commands

- bismark
- bam2nuc
- bismark2bedGraph
- bismark2report
- bismark2summary
- bismark_genome_preparation
- bismark_methylation_extractor
- copy_bismark_files_for_release.pl
- coverage2cytosine
- deduplicate_bismark
- filter_non_conversion
- methylation_consistency

57.4 Dependencies

`Bowtie v2.4.2`, `Samtools v1.12`, `HISAT2 v2.2.1` were included in the container image. So users do not need to provide the dependency path in the bismark parameter.

57.5 Module

You can load the modules by:

```
module load biocontainers
module load bismark
```

57.6 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Bismark on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 12
#SBATCH --job-name=bismark
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers bismark

bismark_genome_preparation --bowtie2 data/ref_genome

bismark --multicore 12 --genome data/ref_genome seq.fastq
```

CHAPTER

FIFTYEIGHT

BLASR

58.1 Introduction

`Blasr` Blasr is a read mapping program that maps reads to positions in a genome by clustering short exact matches between the read and the genome, and scoring clusters using alignment.

For more information, please check its website: https://biocontainers.pro/tools/blasr and its home page on Github.

58.2 Versions

- 5.3.5

58.3 Commands

- blasr

58.4 Module

You can load the modules by:

```
module load biocontainers
module load blasr
```

58.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Blasr on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=blasr
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers blasr

blasr reads.bas.h5  ecoli_K12.fasta -sam
```

CHAPTER

FIFTYNINE

BLAST

59.1 Introduction

BLAST (Basic Local Alignment Search Tool) finds regions of similarity between biological sequences. The program compares nucleotide or protein sequences to sequence databases and calculates the statistical significance.

For more information, please check its website: https://biocontainers.pro/tools/blast and its home page: https://blast.ncbi.nlm.nih.gov/Blast.cgi?CMD=Web&PAGE_TYPE=BlastHome.

59.2 Versions

- 2.11.0
- 2.13.0

59.3 Commands

- blastn
- blastp
- blastx
- blast_formatter
- amino-acid-composition
- between-two-genes
- blastdbcheck
- blastdbcmd
- blastdb_aliastool
- cleanup-blastdb-volumes.py
- deltablast
- dustmasker
- eaddress

- eblast
- get_species_taxids.sh
- legacy_blast.pl
- makeblastdb
- makembindex
- makeprofiledb
- psiblast
- rpsblast
- rpstblastn
- run-ncbi-converter
- segmasker
- tblastn
- tblastx
- update_blastdb.pl
- windowmasker

59.4 Module

You can load the modules by:

```
module load biocontainers
module load blast
```

59.5 BLAST Databases

Local copies of the blast dabase can be found in the directory **/depot/itap/datasets/blast/latest/**. The environment varialbe `BLASTDB` was also set as `/depot/itap/datasets/blast/latest/`. If users want to use `cdd_delta`, `env_nr`, `env_nt`, `nr`, `nt`, `pataa`, `patnt`, `pdbnt`, `refseq_protein`, `refseq_rna`, `swissprot`, or `tsa_nt` databases, do not need to provide the database path. Instead, just use the format like this `-db nr`.

59.6 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run BLAST on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=blast
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers blast

blastp -query protein.fasta -db nr -out test_out -num_threads 4
```

CHAPTER

SIXTY

BLOBTOOLS

60.1 Introduction

`BlobTools` is a modular command-line solution for visualisation, quality control and taxonomic partitioning of genome datasets.

Detailed usage can be found here: https://github.com/DRL/blobtools

60.2 Versions

- 1.1.1

60.3 Commands

- blobtools

60.4 Module

You can load the modules by:

```
module load biocontainers
module load blobtools/1.1.1
```

60.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run blobtools on our our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 4
#SBATCH --job-name=blobtools
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers blobtools/1.1.1

blobtools create -i example/assembly.fna -b example/mapping_1.sorted.bam -t example/
↪blast.out -o test && \
blobtools view -i test.blobDB.json && \
blobtools plot -i test.blobDB.json
```

CHAPTER

SIXTYONE

BMGE

61.1 Introduction

`Bmge` is a program that selects regions in a multiple sequence alignment that are suited for phylogenetic inference.

For more information, please check its website: https://biocontainers.pro/tools/bmge and its home page: https://bioweb.pasteur.fr/packages/pack@BMGE@1.12.

61.2 Versions

- 1.12

61.3 Commands

- bmge

61.4 Module

You can load the modules by:

```
module load biocontainers
module load bmge
```

61.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Bmge on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=bmge
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers bmge

bmge -i seq.fa -t AA -o out.phy
```

CHAPTER

SIXTYTWO

BOWTIE

62.1 Introduction

`Bowtie` is an ultrafast, memory-efficient short read aligner. It aligns short DNA sequences (reads) to the human genome at a rate of over 25 million 35-bp reads per hour. Bowtie indexes the genome with a Burrows-Wheeler index to keep its memory footprint small: typically about 2.2 GB for the human genome (2.9 GB for paired-end).

For more information, please check its website: https://biocontainers.pro/tools/bowtie and its home page: http://bowtie-bio.sourceforge.net/.

62.2 Versions

- 1.3.1-py38

62.3 Commands

- bowtie
- bowtie-build
- bowtie-inspect

62.4 Module

You can load the modules by:

```
module load biocontainers
module load bowtie
```

62.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Bowtie on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=bowtie
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers bowtie

bowtie-build ref.fasta ref
bowtie -p 4 -x ref -1 input_1.fq -2 input_2.fq -S test.sam
```

CHAPTER

SIXTYTHREE

BOWTIE 2

63.1 Introduction

``Bowtie 2``is an ultrafast and memory-efficient tool for aligning sequencing reads to long reference sequences. It is particularly good at aligning reads of about 50 up to 100s or 1,000s of characters, and particularly good at aligning to relatively long (e.g. mammalian) genomes. Bowtie 2 indexes the genome with an FM Index to keep its memory footprint small: for the human genome, its memory footprint is typically around 3.2 GB. Bowtie 2 supports gapped, local, and paired-end alignment modes.

For more information, please check its website: https://biocontainers.pro/tools/bowtie2 and its home page on Github.

63.2 Versions

- 2.4.2-py38

63.3 Commands

- bowtie2
- bowtie2-build
- bowtie2-inspect

63.4 Module

You can load the modules by:

```
module load biocontainers
module load bowtie2
```

63.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Bowtie 2 on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=bowtie2
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers bowtie2

bowtie2-build ref.fasta ref
bowtie2 -p 4 -x ref -1 input_1.fq -2 input_2.fq -S test.sam
```

CHAPTER

SIXTYFOUR

BRACKEN

64.1 Introduction

`Bracken` (Bayesian Reestimation of Abundance with KrakEN) is a highly accurate statistical method that computes the abundance of species in DNA sequences from a metagenomics sample.

Detailed usage can be found here: https://github.com/jenniferlu717/Bracken

---

Note: Inside the `bracken` container image, `kraken2` was also installed. As a result, when you load `bracken/2.6.1-py37`, `kraken version 2.1.1` will be automatically loaded. Please do not load `kraken2` module together with `bracken` module to avaoid conflict.

64.2 Versions

- 2.6.1-py37
- 2.7-py39

64.3 Commands

- bracken
- bracken-build
- combine_bracken_outputs.py
- kraken2
- kraken2-build
- kraken2-inspect
- combine_bracken_outputs.py
- est_abundance.py
- generate_kmer_distribution.py

64.4 Module

You can load the modules by:

```
module load biocontainers
module load bracken/2.6.1-py37
```

64.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run bracken on our our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 10:00:00
#SBATCH -N 1
#SBATCH -n 24
#SBATCH --job-name=bracken
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers bracken/2.6.1-py37

DATABASE=minikraken2_v2_8GB_201904_UPDATE
kraken2 --threads 24  --report kranken2.report --db $DATABASE --paired --classified-out␣
↪cseqs#.fq SRR5043021_1.fastq SRR5043021_2.fastq
bracken -d  $DATABASE -i kranken2.report -o bracken_output -w bracken.report
```

CHAPTER

SIXTYFIVE

BRAKER

65.1 Introduction

BRAKER is a pipeline for fully automated prediction of protein coding gene structures with GeneMark-ES/ET and AUGUSTUS in novel eukaryotic genomes.

For more information. please check its github repository https://github.com/Gaius-Augustus/BRAKER.

65.2 Versions

- 2.1.6

65.3 Commands

braker.pl

65.4 Helper command

Note: Since BRAKER is a pipeline that trains AUGUSTUS, i.e. writes species specific parameter files, BRAKER needs writing access to the configuration directory of AUGUSTUS that contains such files. This installation comes with a stub of AUGUSTUS coniguration files, but you must copy them out from the container into a location where you have write permissions.

A helper command copy_augustus_config is provided to simplify the task. Follow the procedure below to put the config files in your scratch space:

```
$ mkdir -p $RCAC_SCRATCH/augustus
$ copy_augustus_config $RCAC_SCRATCH/augustus
$ export AUGUSTUS_CONFIG_PATH=$RCAC_SCRATCH/augustus/config
```

65.5 Module

You can load the modules by:

```
module load biocontainers
module load braker2/2.1.6
```

65.6 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run BRAKER on our cluster:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 10:00:00
#SBATCH -N 1
#SBATCH -n 24
#SBATCH --job-name=BRAKER2
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers braker2/2.1.6

# The augustus config step is only required for the first time to use BRAKER2
mkdir -p $RCAC_SCRATCH/augustus
copy_augustus_config $RCAC_SCRATCH/augustus
export AUGUSTUS_CONFIG_PATH=$RCAC_SCRATCH/augustus/config

braker.pl --genome genome.fa --bam RNAseq.bam --softmasking --cores 24
```

CHAPTER

SIXTYSIX

BRASS

66.1 Introduction

`Brass` is used to analyze one or more related BAM files of paired-end sequencing to determine potential rearrangement breakpoints.

For more information, please check its website: https://quay.io/repository/wtsicgp/brass and its home page on Github.

66.2 Versions

- 6.3.4

66.3 Commands

- brass-assemble
- brass_bedpe2vcf.pl
- brass_foldback_reads.pl
- brass-group
- brassI_filter.pl
- brassI_np_in.pl
- brassI_pre_filter.pl
- brassI_prep_bam.pl
- brass.pl

66.4 Module

You can load the modules by:

```
module load biocontainers
module load brass
```

66.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Brass on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 4
#SBATCH --job-name=brass
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers brass

brass.pl -c 4 -o myout -t tumour.bam -n normal.bam
```

CHAPTER

SIXTYSEVEN

BRESEQ

67.1 Introduction

`Breseq` is a computational pipeline for the analysis of short-read re-sequencing data.

For more information, please check its website: https://biocontainers.pro/tools/breseq and its home page on Github.

67.2 Versions

- 0.36.1

67.3 Commands

- breseq

67.4 Module

You can load the modules by:

```
module load biocontainers
module load breseq
```

67.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Breseq on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=breseq
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers breseq
```

CHAPTER

SIXTYEIGHT

BUSCO

68.1 Introduction

BUSCO (Benchmarking sets of Universal Single-Copy Orthologs) provides measures for quantitative assessment of genome assembly, gene set, and transcriptome completeness based on evolutionarily informed expectations of gene content from near-universal single-copy orthologs.

Detailed information can be found here: https://gitlab.com/ezlab/busco/

68.2 Versions

- 5.2.2
- 5.3.0
- 5.4.1

68.3 Commands

- busco
- generate_plot.py

68.4 Helper command

Note: Augustus is a gene prediction program for eukaryotes which is required by BUSCO. Augustus requires a writable configuration directory. This installation comes with a stub of AUGUSTUS coniguration files, but you `must` copy them out from the container into a location where you have write permissions.

A helper command `copy_augustus_config` is provided to simplify the task. Follow the procedure below to put the config files in your scratch space:

```
$ mkdir -p $RCAC_SCRATCH/augustus
$ copy_augustus_config $RCAC_SCRATCH/augustus
$ export AUGUSTUS_CONFIG_PATH=$RCAC_SCRATCH/augustus/config
```

68.5 Module

You can load the modules by:

```
module load biocontainers
module load busco
```

68.6 Example job for prokaryotic genomes

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run BUSCO on our cluster:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 12
#SBATCH --job-name=BUSCO
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers busco

## Print the full lineage datasets, and find the dataset fitting your organism.
busco --list-datasets

## run the evaluation
busco -f -c 12 -l actinobacteria_class_odb10  -i bacteria_genome.fasta -o busco_out -m␣
↪genome

## generate a simple summary plot
generate_plot.py -wd busco_out
```

68.7 Example job for eukaryotic genomes

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run BUSCO on our cluster:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 12
#SBATCH --job-name=BUSCO
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers busco

## The augustus config step is only required for the first time to use BUSCO
mkdir -p $RCAC_SCRATCH/augustus
copy_augustus_config $RCAC_SCRATCH/augustus

## This is required for eukaryotic genomes
export AUGUSTUS_CONFIG_PATH=$RCAC_SCRATCH/augustus/config

## Print the full lineage datasets, and find the dataset fitting your organism.
busco --list-datasets

## run the evaluation
busco -f -c 12 -l fungi_odb10 -i fungi_protein.fasta -o busco_out_protein  -m protein
busco -f -c 12 --augustus -l fungi_odb10 -i fungi_genome.fasta -o busco_out_genome  -m␣
↪genome

## generate a simple summary plot
generate_plot.py -wd busco_out_protein
generate_plot.py -wd busco_out_genome
```

CHAPTER

SIXTYNINE

BUSTOOLS

69.1 Introduction

`Bustools` is a program for manipulating BUS files for single cell RNA-Seq datasets.

For more information, please check its website: https://biocontainers.pro/tools/bustools and its home page on Github.

69.2 Versions

- 0.41.0

69.3 Commands

- bustools

69.4 Module

You can load the modules by:

```
module load biocontainers
module load bustools
```

69.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Bustools on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=bustools
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers bustools

bustools capture -s -o cDNA_capture.bus -c cDNA_transcripts.to_capture.txt -e matrix.ec -
↪t transcripts.txt output.correct.sort.bus
bustools count -o u -g cDNA_introns_t2g.txt -e matrix.ec -t transcripts.txt --genecounts␣
↪cDNA_capture.bus
```

CHAPTER

SEVENTY

BWA

70.1 Introduction

BWA (Burrows-Wheeler Aligner) is a fast, accurate, memory-efficient aligner for short and long sequencing reads.

For more information, please check its website: https://biocontainers.pro/tools/bwa and its home page: http://bio-bwa.sourceforge.net.

70.2 Versions

- 0.7.17

70.3 Commands

- bwa
- qualfa2fq.pl
- xa2multi.pl

70.4 Module

You can load the modules by:

```
module load biocontainers
module load bwa
```

70.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run BWA on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=bwa
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers bwa

bwa index ref.fasta
bwa mem ref.fasta input.fq > test.sam
```

CHAPTER

SEVENTYONE

BWAMETH

71.1 Introduction

Bwameth is a tool for fast and accurante alignment of BS-Seq reads.

For more information, please check:
BioContainers: https://biocontainers.pro/tools/bwameth
Home page: https://github.com/brentp/bwa-meth

71.2 Versions

- 0.2.5

71.3 Commands

- bwameth.py

71.4 Module

You can load the modules by:

```
module load biocontainers
module load bwameth
```

71.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run bwameth on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=bwameth
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers bwameth
```

CHAPTER

SEVENTYTWO

CACTUS

72.1 Introduction

`Cactus` is a reference-free whole-genome multiple alignment program.

For more information, please check its website: https://biocontainers.pro/tools/cactus and its home page on Github.

72.2 Versions

- 2.0.5
- 2.2.1

72.3 Commands

- cactus
- cactus-align
- cactus-align-batch
- cactus-blast
- cactus-graphmap
- cactus-graphmap-join
- cactus-graphmap-split
- cactus-minigraph
- cactus-prepare
- cactus-prepare-toil
- cactus-preprocess
- cactus-refmap
- cactus2hal-stitch.sh
- cactus2hal.py
- cactusAPITests

- cactus_analyseAssembly
- cactus_barTests
- cactus_batch_mergeChunks
- cactus_chain
- cactus_consolidated
- cactus_covered_intervals
- cactus_fasta_fragments.py
- cactus_fasta_softmask_intervals.py
- cactus_filterSmallFastaSequences.py
- cactus_halGeneratorTests
- cactus_local_alignment.py
- cactus_makeAlphaNumericHeaders.py
- cactus_softmask2hardmask

72.4 Module

You can load the modules by:

```
module load biocontainers
module load cactus
```

72.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Cactus on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=cactus
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers cactus

wget https://raw.githubusercontent.com/ComparativeGenomicsToolkit/cactus/master/examples/
```

(continues on next page)

(continued from previous page)

```
↪evolverMammals.txt
cactus jobStore evolverMammals.txt evolverMammals.hal
```

CHAPTER

SEVENTYTHREE

CAFE

73.1 Introduction

`Cafe` is a computational tool for the study of gene family evolution.

For more information, please check its website: https://biocontainers.pro/tools/cafe and its home page on Github.

73.2 Versions

- 4.2.1
- 5.0.0

73.3 Commands

- cafe

73.4 Module

You can load the modules by:

```
module load biocontainers
module load cafe
```

73.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Cafe on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=cafe
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers cafe

#To get a list of commands just call CAFE with the -h or --help arguments
cafe5 -h

#To estimate lambda with no among family rate variation issue the command
cafe5 -i mammal_gene_families.txt -t mammal_tree.txt
```

CHAPTER

SEVENTYFOUR

CANU

74.1 Introduction

Canu is a single molecule sequence assembler for genomes large and small.

Detailed usage can be found here: https://github.com/marbl/canu

74.2 Versions

- 2.1.1
- 2.2

74.3 Commands

- canu

74.4 Module

You can load the modules by:

```
module load biocontainers
module load canu/2.2
```

74.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run canu on our our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 20:00:00
#SBATCH -N 1
#SBATCH -n 24
#SBATCH --job-name=canu
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers canu/2.2

canu -p Cm -d clavibacter_pacbio genomeSize=3.4m  -pacbio *.fastq
```

CHAPTER

SEVENTYFIVE

CCS

75.1 Introduction

Pbccs is a tool to generate Highly Accurate Single-Molecule Consensus Reads (HiFi Reads).

For more information, please check:
BioContainers: https://biocontainers.pro/tools/pbccs
Home page: https://github.com/PacificBiosciences/ccs

75.2 Versions

- 6.4.0

75.3 Commands

- ccs

75.4 Module

You can load the modules by:

```
module load biocontainers
module load ccs
```

75.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run ccs on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=ccs
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers ccs

ccs --all subreads.bam ccs.bam
```

CHAPTER

SEVENTYSIX

CDBTOOLS

76.1 Introduction

`Cdbtools` is a collection of tools used for creating indices for quick retrieval of any particular sequences from large multi-FASTA files.

For more information, please check its website: https://biocontainers.pro/tools/cdbtools and its home page: http://compbio.dfci.harvard.edu/tgi.

76.2 Versions

- 0.99

76.3 Commands

- cdbfasta
- cdbyank

76.4 Module

You can load the modules by:

```
module load biocontainers
module load cdbtools
```

76.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Cdbtools on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=cdbtools
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers cdbtools

cdbfasta genome.fa
cdbyank -a 'seq_1' genome.fa.cidx
```

CHAPTER

SEVENTYSEVEN

CD-HIT

77.1 Introduction

`Cd-hit` is a very widely used program for clustering and comparing protein or nucleotide sequences.

For more information, please check its website: https://biocontainers.pro/tools/cd-hit and its home page on Github.

77.2 Versions

- 4.8.1

77.3 Commands

- FET.pl
- cd-hit
- cd-hit-2d
- cd-hit-2d-para.pl
- cd-hit-454
- cd-hit-clstr_2_blm8.pl
- cd-hit-div
- cd-hit-div.pl
- cd-hit-est
- cd-hit-est-2d
- cd-hit-para.pl
- clstr2tree.pl
- clstr2txt.pl
- clstr2xml.pl
- clstr_cut.pl
- clstr_list.pl

- clstr_list_sort.pl
- clstr_merge.pl
- clstr_merge_noorder.pl
- clstr_quality_eval.pl
- clstr_quality_eval_by_link.pl
- clstr_reduce.pl
- clstr_renumber.pl
- clstr_rep.pl
- clstr_reps_faa_rev.pl
- clstr_rev.pl
- clstr_select.pl
- clstr_select_rep.pl
- clstr_size_histogram.pl
- clstr_size_stat.pl
- clstr_sort_by.pl
- clstr_sort_prot_by.pl
- clstr_sql_tbl.pl
- clstr_sql_tbl_sort.pl
- make_multi_seq.pl
- plot_2d.pl
- plot_len1.pl

77.4 Module

You can load the modules by:

```
module load biocontainers
module load cd-hit
```

77.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Cd-hit on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=cd-hit
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers cd-hit

cd-hit -i Cm_pep.fasta  -o Cmdb90 -c 0.9 -n 5 -M 16000 -T 8

cd-hit-est -i Cm_dna.fasta  -o Cmdb90_nt -c 0.9 -n 5 -M 16000 -T 8
```

CHAPTER

SEVENTYEIGHT

CEGMA

78.1 Introduction

CEGMA (Core Eukaryotic Genes Mapping Approach) is a pipeline for building a set of high reliable set of gene annotations in virtually any eukaryotic genome.

For more information, please check:
Docker hub: https://hub.docker.com/r/chrishah/cegma
Home page: https://github.com/KorfLab/CEGMA_v2

78.2 Versions

- 2.5

78.3 Commands

- cegma

78.4 Module

You can load the modules by:

```
module load biocontainers
module load cegma
```

78.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run cegma on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=cegma
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers cegma

cegma --genome genome.fasta -o output
```

CHAPTER

SEVENTYNINE

CELLBENDER

79.1 Introduction

`Cellbender` is a software package for eliminating technical artifacts from high-throughput single-cell RNA sequencing (scRNA-seq) data.

For more information, please check its website: https://biocontainers.pro/tools/cellbender and its home page on Github.

79.2 Versions

- 0.2.0

79.3 Commands

- cellbender

79.4 Module

You can load the modules by:

```
module load biocontainers
module load cellbender
```

79.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Cellbender on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 10:00:00
#SBATCH -N 1
#SBATCH -n 24
#SBATCH --job-name=cellbender
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers cellbender

cellbender remove-background \
             --input cellranger/test_count/run_count_1kpbmcs/outs/raw_feature_bc_matrix.
↪h5 \
             --output output_cpu.h5 \
             --expected-cells 1000 \
             --total-droplets-included 20000 \
             --fpr 0.01 \
             --epochs 150
```

CHAPTER

EIGHTY

CELLPHONEDB

80.1 Introduction

CellPhoneDB is a publicly available repository of curated receptors, ligands and their interactions.

For more information, please check:
Docker hub: https://hub.docker.com/r/eagleshot/cellphonedb
Home page: https://github.com/Teichlab/cellphonedb

80.2 Versions

- 2.1.7

80.3 Commands

- cellphonedb

80.4 Module

You can load the modules by:

```
module load biocontainers
module load cellphonedb
```

80.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run cellphonedb on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=cellphonedb
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers cellphonedb
```

CHAPTER

EIGHTYONE

CELLRANGER

81.1 Introduction

`Cellranger` is a set of analysis pipelines that process Chromium single-cell data to align reads, generate feature-barcode matrices, perform clustering and other secondary analysis, and more. Detailed usage can be found here: https://support.10xgenomics.com/single-cell-gene-expression/software/pipelines/latest/what-is-cell-ranger.

81.2 Versions

- 6.0.1
- 6.1.1
- 6.1.2
- 7.0.0
- 7.0.1

81.3 Commands

- cellranger mkfastq
- cellranger count
- cellranger aggr
- cellranger reanalyze
- cellranger multi

81.4 Module

You can load the modules by:

```
module load biocontainers
module load cellranger
```

81.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run cellranger our our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 20:00:00
#SBATCH -N 1
#SBATCH -n 48
#SBATCH --job-name=cellranger
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers cellranger

cellranger count --id=run_count_1kpbmcs --fastqs=pbmc_1k_v3_fastqs --sample=pbmc_1k_v3 --
↪transcriptome=refdata-gex-GRCh38-2020-A
```

CHAPTER

EIGHTYTWO

CELLRANGER-ARC

82.1 Introduction

Cell Ranger ARC is a set of analysis pipelines that process Chromium Single Cell Multiome ATAC + Gene Expression sequencing data to generate a variety of analyses pertaining to gene expression (GEX), chromatin accessibility, and their linkage. Furthermore, since the ATAC and GEX measurements are on the very same cell, we are able to perform analyses that link chromatin accessibility and GEX.

For more information, please check:
Docker hub: https://hub.docker.com/r/cumulusprod/cellranger-arc
Home page:
https://support.10xgenomics.com/single-cell-multiome-atac-gex/software/pipelines/latest/what-is-cell-ranger-arc

82.2 Versions

- 2.0.2

82.3 Commands

- cellranger-arc

82.4 Module

You can load the modules by:

```
module load biocontainers
module load cellranger-arc
```

82.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run cellranger-arc on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=cellranger-arc
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers cellranger-arc
```

CHAPTER

EIGHTYTHREE

CELLRANGER-ATAC

83.1 Introduction

`Cellranger-atac` is a set of analysis pipelines that process Chromium Single Cell ATAC data.

For more information, please check its | Docker hub: https://hub.docker.com/r/cumulusprod/cellranger-atac and its home page: https://support.10xgenomics.com/single-cell-atac/software/pipelines/latest/algorithms/overview.

83.2 Versions

- 2.0.0
- 2.1.0

83.3 Commands

- cellranger-atac

83.4 Module

You can load the modules by:

```
module load biocontainers
module load cellranger-atac
```

83.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

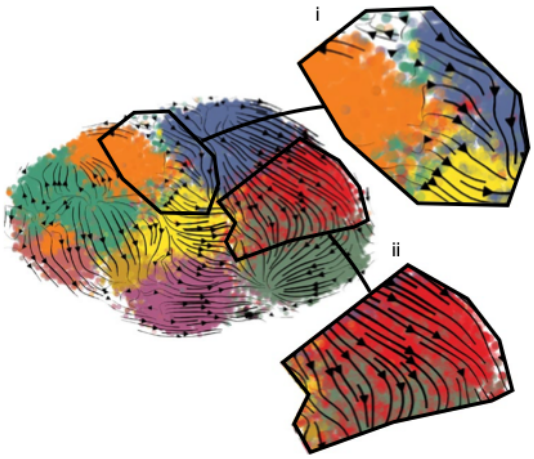
To run Cellranger-atac on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 8
#SBATCH --mem=64G
#SBATCH --job-name=cellranger-atac
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers cellranger-atac

cellranger-atac count --id=sample345 \
                      --reference=refdata-cellranger-arc-GRCh38-2020-A-2.0.0 \
                      --fastqs=runs/HAWT7ADXX/outs/fastq_path \
                      --sample=mysample \
                      --localcores=8 \
                      --localmem=64
```

CHAPTER

EIGHTYFOUR

CELLRANK



84.1 Introduction

`CellRank` a toolkit to uncover cellular dynamics based on Markov state modeling of single-cell data. Detailed information about CellRank can be found here: https://cellrank.readthedocs.io/en/stable/.

84.2 Versions

- 1.5.1

84.3 Commands

- python
- python3

84.4 Module

You can load the modules by:

```
module load biocontainers
module load cellrank/1.5.1
```

Note: The CellRank container also contained scVelo and scanpy. When you want to use CellRank, do not load scVelo or scanpy.

84.5 Interactive job

To run CellRank interactively on our clusters:

```
(base) UserID@bell-fe00:~ $ sinteractive -N1 -n12 -t4:00:00 -A myallocation
salloc: Granted job allocation 12345869
salloc: Waiting for resource configuration
salloc: Nodes bell-a008 are ready for job
(base) UserID@bell-a008:~ $ module load biocontainers cellrank/1.5.1
(base) UserID@bell-a008:~ $ python
Python 3.9.9 |  packaged by conda-forge |  (main, Dec 20 2021, 02:41:03)
[GCC 9.4.0] :: Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import scanpy as sc
>>> import scvelo as scv
>>> import cellrank as cr
>>> import numpy as np
>>> scv.settings.verbosity = 3
>>> scv.settings.set_figure_params("scvelo")
>>> cr.settings.verbosity = 2
```

84.6 Batch job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To submit a sbatch job on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 10:00:00
#SBATCH -N 1
#SBATCH -n 24
#SBATCH --job-name=cellrank
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
```

(continues on next page)

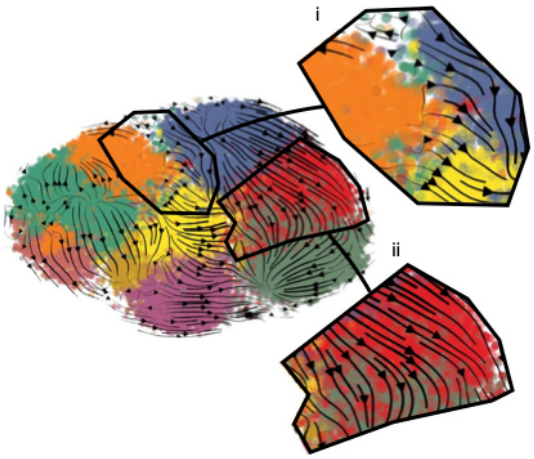
(continued from previous page)

```
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers cellrank/1.5.1

python script.py
```

CHAPTER

EIGHTYFIVE

CELLRANK-KRYLOV



85.1 Introduction

`CellRank` a toolkit to uncover cellular dynamics based on Markov state modeling of single-cell data. `CellRank-krylov` is `CellRank` installed with extra libraries, enabling it to have better performance for large datasets (>15k cells). Detailed information about CellRank can be found here: https://cellrank.readthedocs.io/en/stable/.

85.2 Versions

- 1.5.1

85.3 Commands

- python
- python3

85.4 Module

You can load the modules by:

```
module load biocontainers
module load cellrank-krylov/1.5.1
```

Note: The CellRank container also contained scVelo and scanpy. When you want to use CellRank, do not load scVelo or scanpy.

85.5 Interactive job

To run CellRank-krylov interactively on our clusters:

```
(base) UserID@bell-fe00:~ $ sinteractive -N1 -n12 -t4:00:00 -A myallocation
salloc: Granted job allocation 12345869
salloc: Waiting for resource configuration
salloc: Nodes bell-a008 are ready for job
(base) UserID@bell-a008:~ $ module load biocontainers cellrank-krylov/1.5.1
(base) UserID@bell-a008:~ $ python
Python 3.9.9 |  packaged by conda-forge |  (main, Dec 20 2021, 02:41:03)
[GCC 9.4.0] :: Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import scanpy as sc
>>> import scvelo as scv
>>> import cellrank as cr
>>> import numpy as np
>>> scv.settings.verbosity = 3
>>> scv.settings.set_figure_params("scvelo")
>>> cr.settings.verbosity = 2
```

85.6 Batch job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To submit a sbatch job on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 10:00:00
#SBATCH -N 1
#SBATCH -n 24
#SBATCH --job-name=cellrank-krylov
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
```

(continues on next page)

(continued from previous page)

```
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers cellrank-krylov/1.5.1

python script.py
```

CHAPTER

EIGHTYSIX

CELLSNP

86.1 Introduction

`cellSNP` aims to pileup the expressed alleles in single-cell or bulk RNA-seq data, which can be directly used for donor deconvolution in multiplexed single-cell RNA-seq data, particularly with vireo, which assigns cells to donors and detects doublets, even without genotyping reference.

For more information, please check its website: https://biocontainers.pro/tools/cellsnp-lite and its home page on Github.

86.2 Versions

- 1.2.2

86.3 Commands

- cellsnp-lite

86.4 Module

You can load the modules by:

```
module load biocontainers
module load cellsnp-lite
```

86.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run cellSNP on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 8
#SBATCH --job-name=cellsnp-lite
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers cellsnp-lite

cellsnp-lite -s sample.bam -b barcode.tsv -O cellsnp_out -p 8 --minMAF 0.1 --minCOUNT 100
```

CHAPTER

EIGHTYSEVEN

CELLTYPIST

87.1 Introduction

`Celltypist` is a tool for semi-automatic cell type annotation.

For more information, please check its website: https://biocontainers.pro/tools/celltypist and its home page on Github.

87.2 Versions

- 0.2.0
- 1.1.0

87.3 Commands

- celltypist
- python
- python3

87.4 Module

You can load the modules by:

```
module load biocontainers
module load celltypist
```

87.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Celltypist on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 8
#SBATCH --job-name=celltypist
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers celltypist

celltypist --indata demo_2000_cells.h5ad --model Immune_All_Low.pkl --outdir output
```

CHAPTER

EIGHTYEIGHT

CENTRIFUGE

88.1 Introduction

`Centrifuge` is a novel microbial classification engine that enables rapid, accurate, and sensitive labeling of reads and quantification of species on desktop computers.

For more information, please check its website: https://biocontainers.pro/tools/centrifuge and its home page: http://www.ccb.jhu.edu/software/centrifuge/.

88.2 Versions

- 1.0.4_beta

88.3 Commands

- centrifuge
- centrifuge-BuildSharedSequence.pl
- centrifuge-RemoveEmptySequence.pl
- centrifuge-RemoveN.pl
- centrifuge-build
- centrifuge-build-bin
- centrifuge-class
- centrifuge-compress.pl
- centrifuge-download
- centrifuge-inspect
- centrifuge-inspect-bin
- centrifuge-kreport
- centrifuge-sort-nt.pl
- centrifuge_evaluate.py

- centrifuge_simulate_reads.py

88.4 Module

You can load the modules by:

```
module load biocontainers
module load centrifuge
```

88.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Centrifuge on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=centrifuge
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers centrifuge

centrifuge-download -o taxonomy taxonomy
centrifuge-download -o library -m -d "archaea,bacteria,viral" refseq > seqid2taxid.map
cat library/*/*.fna > input-sequences.fna
centrifuge-build -p 8 --conversion-table seqid2taxid.map \
              --taxonomy-tree taxonomy/nodes.dmp --name-table taxonomy/names.dmp \
              input-sequences.fna abv
```

CHAPTER

EIGHTYNINE

CHECKM-GENOME

89.1 Introduction

CheckM provides a set of tools for assessing the quality of genomes recovered from isolates, single cells, or metagenomes.

For more information, please check:
BioContainers: https://biocontainers.pro/tools/checkm-genome
Home page: https://github.com/Ecogenomics/CheckM

89.2 Versions

- 1.2.0

89.3 Commands

- checkm-genome

89.4 Module

You can load the modules by:

```
module load biocontainers
module load checkm-genome
```

89.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run checkm-genome on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 8
#SBATCH --job-name=checkm-genome
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers checkm-genome

checkm lineage_wf -t 8 -x fa bins checkm
```

CHAPTER

NINETY

CHEWBBACA

90.1 Introduction

chewBBACA is a comprehensive pipeline including a set of functions for the creation and validation of whole genome and core genome MultiLocus Sequence Typing (wg/cgMLST) schemas, providing an allele calling algorithm based on Blast Score Ratio that can be run in multiprocessor settings and a set of functions to visualize and validate allele variation in the loci. chewBBACA performs the schema creation and allele calls on complete or draft genomes resulting from de novo assemblers.

For more information, please check:
BioContainers: https://biocontainers.pro/tools/chewbbaca
Home page: https://github.com/B-UMMI/chewBBACA

90.2 Versions

- 2.8.5

90.3 Commands

- chewBBACA.py

90.4 Module

You can load the modules by:

```
module load biocontainers
module load chewbbaca
```

90.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run chewbbaca on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 4
#SBATCH --job-name=chewbbaca
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers chewbbaca

chewBBACA.py CreateSchema -i complete_genomes/ -o tutorial_schema --ptf Streptococcus_
↪agalactiae.trn --cpu 4
chewBBACA.py AlleleCall -i complete_genomes/ -g tutorial_schema/schema_seed -o results32_
↪wgMLST --cpu 4
```

CHAPTER

NINETYONE

CHROMAP

91.1 Introduction

Chromap is an ultrafast method for aligning and preprocessing high throughput chromatin profiles.

For more information, please check:
BioContainers: https://biocontainers.pro/tools/chromap
Home page: https://github.com/haowenz/chromap

91.2 Versions

- 0.2.2

91.3 Commands

- chromap

91.4 Module

You can load the modules by:

```
module load biocontainers
module load chromap
```

91.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run chromap on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=chromap
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers chromap
```

CHAPTER

NINETYTWO

CICERO

92.1 Introduction

CICERO (Clipped-reads Extended for RNA Optimization) is an assembly-based algorithm to detect diverse classes of driver gene fusions from RNA-seq.

For more information, please check its home page on Github.

92.2 Versions

- 1.8.1

92.3 Commands

- Cicero.sh

92.4 Module

You can load the modules by:

```
module load biocontainers
module load cicero
```

92.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run CICERO on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=cicero
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers cicero
```

CHAPTER

NINETYTHREE

CIRCEXPLORER2

93.1 Introduction

CIRCexplorer2 is a comprehensive and integrative circular RNA analysis toolset. It is the successor of CIRCexplorer with plenty of new features to facilitate circular RNA identification and characterization.

For more information, please check:
BioContainers: https://biocontainers.pro/tools/circexplorer2
Home page: https://github.com/YangLab/CIRCexplorer2

93.2 Versions

- 2.3.8

93.3 Commands

- CIRCexplorer2
- fast_circ.py
- fetch_ucsc.py

93.4 Module

You can load the modules by:

```
module load biocontainers
module load circexplorer2
```

93.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run circexplorer2 on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=circexplorer2
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers circexplorer2
```

CHAPTER

NINETYFOUR

CIRCLATOR

94.1 Introduction

`Circlator` is a tool to circularize genome assemblies.

For more information, please check its | Docker hub: https://hub.docker.com/r/sangerpathogens/circlator and its home page on Github.

94.2 Versions

- 1.5.5

94.3 Commands

- circlator
- python3

94.4 Module

You can load the modules by:

```
module load biocontainers
module load circlator
```

94.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Circlator on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=circlator
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers circlator

circlator minimus2  minimus2_test_run_minimus2.in.fa  minimus2_test
```

CHAPTER

NINETYFIVE

CIRCOMPARA2

95.1 Introduction

CirComPara2 is a computational pipeline to detect, quantify, and correlate expression of linear and circular RNAs from RNA-seq data that combines multiple circRNA-detection methods.

For more information, please check:
Home page: https://github.com/egaffo/circompara2

95.2 Versions

- 0.1.2.1

95.3 Commands

- python
- Rscript
- circompara2
- CIRCexplorer2
- CIRCexplorer_compare.R
- CIRI.pl
- DCC
- DCC_patch_CombineCounts.py
- QRE_finder.py
- STAR
- bedtools
- bowtie
- bowtie-build
- bowtie-inspect

- bowtie2
- bowtie2-build
- bowtie2-inspect
- bwa
- ccp_circrna_expression.R
- cfinder_compare.R
- chimoutjunc_to_bed.py
- ciri_compare.R
- collect_read_stats.R
- convert_circrna_collect_tables.py
- cuffcompare
- cuffdiff
- cufflinks
- cuffmerge
- cuffnorm
- cuffquant
- dcc_compare.R
- dcc_fix_strand.R
- fasta_len.py
- fastq_rev_comp.py
- fastqc
- filterCirc.awk
- filterSpliceSiteCircles.pl
- filter_and_cast_circexp.R
- filter_fastq_reads.py
- filter_findcirc_res.R
- filter_segemehl.R
- find_circ.py
- findcirc_compare.R
- gene_annotation.R
- get_ce2_bwa_bks_reads.R
- get_ce2_bwa_circ_reads.py
- get_ce2_segemehl_bks_reads.R
- get_ce2_star_bks_reads.R
- get_ce2_th_bks_reads.R
- get_circompara_counts.R

- get_circrnaFinder_bks_reads.R
- get_ciri_bks_reads.R
- get_dcc_bks_reads.R
- get_findcirc_bks_reads.R
- get_gene_expression_files.R
- get_stringtie_rawcounts.R
- gffread
- gtfToGenePred
- gtf_collapse_features.py
- gtf_to_sam
- haarz.x
- hisat2
- hisat2-build
- htseq-count
- install_R_libs.R
- nrForwardSplicedReads.pl
- parallel
- pip
- postProcessStarAlignment.pl
- samtools
- samtools_v0
- scons
- segemehl.x
- split_start_end_gtf.py
- starCirclesToBed.pl
- stringtie
- testrealign_compare.R
- tophat2
- trim_read_header.py
- trimmomatic-0.39.jar
- unmapped2anchors.py
- cf_filterChimout.awk
- circompara
- get_unmapped_reads_from_bam.sh
- install_circompara
- make_circrna_html

- make_indexes

95.4 Module

You can load the modules by:

```
module load biocontainers
module load circompara2
```

95.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run circompara2 on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=circompara2
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers circompara2
```

CHAPTER

NINETYSIX

CIRCOS

96.1 Introduction

`Circos` is a software package for visualizing data and information.

For more information, please check its website: https://biocontainers.pro/tools/circos and its home page: http://circos.ca.

96.2 Versions

- 0.69.8

96.3 Commands

- circos

96.4 Module

You can load the modules by:

```
module load biocontainers
module load circos
```

96.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Circos on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=circos
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers circos

circos -conf circos.conf
```

CHAPTER

NINETYSEVEN

CIRIQUANT

97.1 Introduction

CIRIquant is a comprehensive analysis pipeline for circRNA detection and quantification in RNA-Seq data.

For more information, please check its | Docker hub: https://hub.docker.com/r/mortreux/ciriquant and its home page on Github.

97.2 Versions

- 1.1.2

97.3 Commands

- CIRIquant

97.4 Module

You can load the modules by:

```
module load biocontainers
module load ciriquant
```

97.5 config.yml

All required dependencies have been installed within the CIRIquant container image. But users still need toprovide the PATH of these exectuables in *config.yml*. Please use the below *config.yml* as example:

```
name: hg38
tools:
   bwa: /bin/bwa
   hisat2: /bin/hisat2
```

(continues on next page)

(continued from previous page)

```
   stringtie: /bin/stringtie
   samtools: /usr/local/bin/samtools
reference:
   fasta: reference/Homo_sapiens.GRCh38.dna.primary_assembly.fa
   gtf:  reference/Homo_sapiens.GRCh38.105.gtf
   bwa_index: reference/Homo_sapiens.GRCh38.dna.primary_assembly.fa
   hisat_index: reference/hg38_hisat2
```

97.6 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run CIRIquant on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 10:00:00
#SBATCH -N 1
#SBATCH -n 64
#SBATCH --job-name=ciriquant
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers ciriquant

CIRIquant -t 64 -1 SRR12095148_1.fastq -2 SRR12095148_2.fastq --config config.yml -o␣
↪Output -p test
```

CHAPTER

NINETYEIGHT

CLAIR3

98.1 Introduction

Clair3 is a germline small variant caller for long-reads. Clair3 makes the best of two major method categories: pileup calling handles most variant candidates with speed, and full-alignment tackles complicated candidates to maximize precision and recall. Clair3 runs fast and has superior performance, especially at lower coverage. Clair3 is simple and modular for easy deployment and integration.

For more information, please check:
Docker hub: https://hub.docker.com/r/hkubal/clair3
Home page: https://github.com/HKU-BAL/Clair3

98.2 Versions

- 0.1-r11
- 0.1-r12

98.3 Commands

- run_clair3.sh

98.4 Module

You can load the modules by:

```
module load biocontainers
module load clair3
```

98.5 Model_path

Note: `model_path` is in `/opt/models/`. The parameter will be like this `--model_path="/opt/models/MODEL_NAME"`

98.6 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run clair3 on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=clair3
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers clair3

run_clair3.sh \
      --bam_fn=input.bam \
      --ref_fn=ref.fasta \
      --threads=12 \
      --platform=ont \
      --model_path="/opt/models/ont" \
      --output=output
```

CHAPTER

NINETYNINE

CLAIRVOYANTE

99.1 Introduction

Clairvoyante is a deep neural network based variant caller.

For more information, please check:
Docker hub: https://hub.docker.com/r/lifebitai/clairvoyante
Home page: https://github.com/aquaskyline/Clairvoyante

99.2 Versions

- 1.02

99.3 Commands

- clairvoyante.py

99.4 Module

You can load the modules by:

```
module load biocontainers
module load clairvoyante
```

99.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run clairvoyante on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=clairvoyante
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers clairvoyante

cd training
clairvoyante.py callVarBam \
   --chkpnt_fn ../trainedModels/fullv3-illumina-novoalign-hg001+hg002-hg38/
↪learningRate1e-3.epoch500 \
   --bam_fn ../testingData/chr21/chr21.bam \
   --ref_fn ../testingData/chr21/chr21.fa \
   --bed_fn ../testingData/chr21/chr21.bed \
   --call_fn chr21_calls.vcf \
   --ctgName chr21
```

CHAPTER

CLEARCNV

100.1 Introduction

ClearCNV: CNV calling from NGS panel data in the presence of ambiguity and noise.

For more information, please check:
BioContainers: https://biocontainers.pro/tools/clearcnv
Home page: https://github.com/bihealth/clear-cnv

100.2 Versions

- 0.306

100.3 Commands

- clearCNV

100.4 Module

You can load the modules by:

```
module load biocontainers
module load clearcnv
```

100.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run clearcnv on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=clearcnv
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers clearcnv
```

CHAPTER

ONE

CLEVER-TOOLKIT

101.1 Introduction

Clever-toolkit is a collection of tools to discover and genotype structural variations in genomes from paired-end sequencing reads. The main software is written in C++ with some auxiliary scripts in Python.

For more information, please check:
BioContainers: https://biocontainers.pro/tools/clever-toolkit
Home page: https://bitbucket.org/tobiasmarschall/clever-toolkit/src/master/

101.2 Versions

- 2.4-py37

101.3 Commands

- clever
- laser
- bam-to-alignment-priors
- split-priors-by-chromosome
- clever-core
- postprocess-predictions
- evaluate-sv-predictions
- split-reads
- laser-core
- laser-recalibrate
- genotyper
- insert-length-histogram
- add-score-tags-to-bam
- bam2fastq

- remove-redundant-variations
- precompute-distributions
- extract-bad-reads
- filter-variations
- merge-to-vcf
- multiline-to-xa
- filter-bam
- read-group-stats

101.4 Module

You can load the modules by:

```
module load biocontainers
module load clever-toolkit
```

101.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run clever-toolkit on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=clever-toolkit
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers clever-toolkit

cat mapped.bam |  bam2fastq output_1.fq output_2.fq
```

CHAPTER

TWO

CLUSTALW

102.1 Introduction

`Clustalw` is a general purpose multiple alignment program for DNA or proteins.

For more information, please check its website: https://biocontainers.pro/tools/clustalw and its home page: http://www.clustal.org/clustal2/.

102.2 Versions

- 2.1

102.3 Commands

- clustalw

102.4 Module

You can load the modules by:

```
module load biocontainers
module load clustalw
```

102.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Clustalw on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=clustalw
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers clustalw

clustalw -tree -align -infile=seq.faa
```

CHAPTER

THREE

CNVKIT

103.1 Introduction

`CNVkit` is a command-line toolkit and Python library for detecting copy number variants and alterations genome-wide from high-throughput sequencing.

For more information, please check its website: https://biocontainers.pro/tools/cnvkit and its home page on Github.

103.2 Versions

- 0.9.9-py

103.3 Commands

- cnvkit.py
- cnv_annotate.py
- cnv_expression_correlate.py
- cnv_updater.py

103.4 Module

You can load the modules by:

```
module load biocontainers
module load cnvkit
```

103.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run CNVkit on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=cnvkit
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers cnvkit

cnvkit.py batch *Tumor.bam --normal *Normal.bam \
                --targets my_baits.bed --fasta hg19.fasta \
                --access data/access-5kb-mappable.hg19.bed \
                --output-reference my_reference.cnn
                --output-dir example/
```

CHAPTER

FOUR

CNVNATOR

104.1 Introduction

`Cnvnator` is a tool for discovery and characterization of copy number variation (CNV) in population genome sequencing data.

For more information, please check its website: https://biocontainers.pro/tools/cnvnator and its home page on Github.

104.2 Versions

- 0.4.1

104.3 Commands

- cnvnator
- cnvnator2VCF.pl
- plotbaf.py
- plotcircular.py
- plotrdbaf.py
- pytools.py

104.4 Module

You can load the modules by:

```
module load biocontainers
module load cnvnator
```

104.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Cnvnator on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=cnvnator
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers cnvnator

cnvnator -root file.root -tree file.bam -chrom $(seq 1 22) X Y

plotcircular.py file.root
```

CHAPTER

FIVE

COINFINDER

105.1 Introduction

Coinfinder is an algorithm and software tool that detects genes which associate and dissociate with other genes more often than expected by chance in pangenomes.

For more information, please check:
BioContainers: https://biocontainers.pro/tools/coinfinder
Home page: https://github.com/fwhelan/coinfinder

105.2 Versions

- 1.2.0

105.3 Commands

- coinfinder

105.4 Module

You can load the modules by:

```
module load biocontainers
module load coinfinder
```

105.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run coinfinder on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=coinfinder
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers coinfinder

coinfinder -i coinfinder-manuscript/gene_presence_absence.csv \
    -I -p coinfinder-manuscript/core-gps_fasttree.newick \
    -o output
```

CHAPTER

SIX

CONCOCT

106.1 Introduction

CONCOCT: Clustering cONtigs with COverage and ComposiTion.

Detailed usage can be found here: https://github.com/BinPro/CONCOCT

106.2 Versions

- 1.1.0-py38

106.3 Commands

- concoct
- concoct_refine
- concoct_coverage_table.py
- cut_up_fasta.py
- extract_fasta_bins.py
- merge_cutup_clustering.py

106.4 Module

You can load the modules by:

```
module load biocontainers
module load concoct/1.1.0-py38
```

106.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run concoct on our our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 20:00:00
#SBATCH -N 1
#SBATCH -n 24
#SBATCH --job-name=concoct
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers concoct/1.1.0-py38

cut_up_fasta.py final.contigs.fa -c 10000 -o 0 --merge_last -b contigs_10K.bed > contigs_
↪10K.fa
concoct_coverage_table.py contigs_10K.bed SRR1976948_sorted.bam > coverage_table.tsv
concoct --composition_file contigs_10K.fa --coverage_file coverage_table.tsv -b concoct_
↪output/
```

CHAPTER

SEVEN

CONTROL-FREEC

107.1 Introduction

`Control-freec` is a tool for detection of copy-number changes and allelic imbalances (including LOH) using deep-sequencing data.

For more information, please check its website: https://biocontainers.pro/tools/control-freec and its home page on Github.

107.2 Versions

- 11.6

107.3 Commands

- freec

107.4 Module

You can load the modules by:

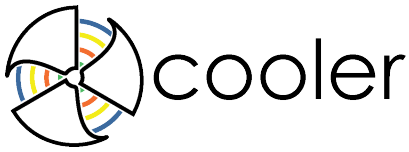
```
module load biocontainers
module load control-freec
```

107.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Control-freec on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=control-freec
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers control-freec

freec -conf config_chr19.txt
```



CHAPTER

EIGHT

COOLER

108.1 Introduction

`Cooler` is a support library for a sparse, compressed, binary persistent storage format, also called cooler, used to store genomic interaction data, such as Hi-C contact matrices.

For more information, please check its website: https://biocontainers.pro/tools/cooler and its home page on Github.

108.2 Versions

- 0.8.11

108.3 Commands

- cooler
- python
- python3

108.4 Module

You can load the modules by:

```
module load biocontainers
module load Cooler
```

108.5 Interactive job

To run Cooler interactively on our clusters:

```
(base) UserID@bell-fe00:~ $ sinteractive -N1 -n12 -t4:00:00 -A myallocation
salloc: Granted job allocation 12345869
salloc: Waiting for resource configuration
salloc: Nodes bell-a008 are ready for job
(base) UserID@bell-a008:~ $ module load biocontainers cooler
(base) UserID@bell-a008:~ $ python
Python 3.9.7 |  packaged by conda-forge |  (default, Sep 29 2021, 19:20:46)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import cooler
```

108.6 Batch job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Cooler batch jobs on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=cooler
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers cooler

cooler info data/Rao2014-GM12878-MboI-allreps-filtered.1000kb.cool
cooler info -f bin-size data/Rao2014-GM12878-MboI-allreps-filtered.1000kb.cool
cooler info -m data/Rao2014-GM12878-MboI-allreps-filtered.1000kb.cool
cooler tree data/Rao2014-GM12878-MboI-allreps-filtered.1000kb.cool
cooler attrs data/Rao2014-GM12878-MboI-allreps-filtered.1000kb.cool
```

CHAPTER

NINE

COVERM

109.1 Introduction

`Coverm` is a configurable, easy to use and fast DNA read coverage and relative abundance calculator focused on metagenomics applications.

For more information, please check its website: https://biocontainers.pro/tools/coverm and its home page on Github.

109.2 Versions

- 0.6.1

109.3 Commands

- coverm

109.4 Module

You can load the modules by:

```
module load biocontainers
module load coverm
```

109.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Coverm on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=coverm
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers coverm

coverm  genome  --genome-fasta-files xcc.fasta  --coupled SRR11234553_1.fastq␣
↪SRR11234553_2.fastq
```

CRISPRCASFINDER

110.1 Introduction

`CRISPRCasFinder` enables the easy detection of CRISPRs and cas genes in user-submitted sequence data. It is an updated, improved, and integrated version of CRISPRFinder and CasFinder.

Detailed usage can be found here: https://github.com/dcouvin/CRISPRCasFinder

110.2 Versions

- 4.2.20

110.3 Commands

- CRISPRCasFinder.pl

110.4 Module

You can load the modules by:

```
module load biocontainers
module load crisprcasfinder/4.2.20
```

110.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run CRISPRCasFinder on our our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 2:00:00
#SBATCH -N 1
```

(continues on next page)

(continued from previous page)

```
#SBATCH -n 12
#SBATCH --job-name=CRISPRCasFinder
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers crisprcasfinder/4.2.20

CRISPRCasFinder.pl -in install_test/sequence.fasta -cas -cf CasFinder-2.0.3 -def G -keep
```

CHAPTER

ONE

CRISPRESSO2

111.1 Introduction

CRISPResso2 is a software pipeline designed to enable rapid and intuitive interpretation of genome editing experiments.

For more information, please check:
Docker hub: https://hub.docker.com/r/pinellolab/crispresso2
Home page: https://github.com/pinellolab/CRISPResso2

111.2 Versions

- 2.2.10
- 2.2.8
- 2.2.9

111.3 Commands

- CRISPResso
- CRISPRessoAggregate
- CRISPRessoBatch
- CRISPRessoCompare
- CRISPRessoPooled
- CRISPRessoPooledWGSCompare
- CRISPRessoWGS

111.4 Module

You can load the modules by:

```
module load biocontainers
module load crispresso2
```

111.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run crispresso2 on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=crispresso2
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers crispresso2

CRISPResso --fastq_r1 nhej.r1.fastq.gz --fastq_r2 nhej.r2.fastq.gz -n nhej --amplicon_
↪seq \
   ␣
↪AATGTCCCCCAATGGGAAGTTCATCTGGCACTGCCCACAGGTGAGGAGGTCATGATCCCCTTCTGGAGCTCCCAACGGGCCGTGGTCTGGTTCATCATCTG
```

CHAPTER

TWO

CRISPRITZ

112.1 Introduction

`Crispritz` is a software package containing 5 different tools dedicated to perform predictive analysis and result assessement on CRISPR/Cas experiments.

For more information, please check its website: https://biocontainers.pro/tools/crispritz and its home page on Github.

112.2 Versions

- 2.6.5-py39

112.3 Commands

- crispritz.py

112.4 Module

You can load the modules by:

```
module load biocontainers
module load crispritz
```

112.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Crispritz on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=crispritz
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers crispritz

crispritz.py add-variants hg38_1000genomeproject_vcf/ hg38_ref/ &> output.redirect.out

crispritz.py index-genome hg38_ref hg38_ref/ 20bp-NGG-SpCas9.txt -bMax 2 &> output.
↪redirect.out

crispritz.py search hg38_ref/ 20bp-NGG-SpCas9.txt EMX1.sgRNA.txt emx1.hg38 -mm 4 -t -
↪scores hg38_ref/ &> output.redirect.out

crispritz.py search genome_library/NGG_2_hg38_ref/ 20bp-NGG-SpCas9.txt EMX1.sgRNA.txt␣
↪emx1.hg38.bulges -index -mm 4 -bDNA 1 -bRNA 1 -t &> output.redirect.out

crispritz.py annotate-results emx1.hg38.targets.txt hg38Annotation.bed emx1.hg38 &>␣
↪output.redirect.out
```

CHAPTER

THREE

CROSSMAP

113.1 Introduction

`Crossmap` is a program for genome coordinates conversion between different assemblies.

For more information, please check its website: https://biocontainers.pro/tools/crossmap and its home page: https://crossmap.readthedocs.io/en/latest/#convert-maf-format-files.

113.2 Versions

- 0.6.3

113.3 Commands

- CrossMap.py

113.4 Module

You can load the modules by:

```
module load biocontainers
module load crossmap
```

113.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Crossmap on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=crossmap
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers crossmap

CrossMap.py bed GRCh37_to_GRCh38.chain.gz test.bed
```

CHAPTER

FOUR

CROSS_MATCH

114.1 Introduction

`cross_match` is a general purpose utility for comparing any two DNA sequence sets using a 'banded' version of swat.

For more information, please check its home page: http://www.phrap.org/phredphrapconsed.html#block_phrap.

114.2 Versions

- 1.090518

114.3 Commands

- cross_match

114.4 Module

You can load the modules by:

```
module load biocontainers
module load cross_match
```

114.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run cross_match on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=cross_match
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers cross_match
```

CHAPTER

FIVE

CSVTK

115.1 Introduction

`Csvtk` is a cross-platform, efficient and practical CSV/TSV toolkit.

For more information, please check its website: https://biocontainers.pro/tools/csvtk and its home page on Github.

115.2 Versions

- 0.23.0
- 0.25.0

115.3 Commands

- csvtk

115.4 Module

You can load the modules by:

```
module load biocontainers
module load csvtk
```

115.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Csvtk on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=csvtk
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers csvtk

cat data.csv \
 |  csvtk summary --ignore-non-digits --fields f4:sum,f5:sum --groups f1,f2 \
 |  csvtk pretty
```

CHAPTER

SIX

CUFFLINKS

116.1 Introduction

`Cufflinks` assembles transcripts, estimates their abundances, and tests for differential expression and regulation in RNA-Seq samples. It accepts aligned RNA-Seq reads and assembles the alignments into a parsimonious set of transcripts. Cufflinks then estimates the relative abundances of these transcripts based on how many reads support each one, taking into account biases in library preparation protocols.

For more information, please check its website: https://biocontainers.pro/tools/cufflinks and its home page on Github.

116.2 Versions

- 2.2.1-py36

116.3 Commands

- cuffcompare
- cuffdiff
- cufflinks
- cuffmerge
- cuffnorm
- cuffquant
- gffread
- gtf_to_sam

116.4 Module

You can load the modules by:

```
module load biocontainers
module load cufflinks
```

116.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Cufflinks on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 8
#SBATCH --job-name=cufflinks
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers cufflinks

cufflinks -p 8 -G transcript.gtf --library-type fr-unstranded -o cufflinks_output tophat_
↪out/accepted_hits.bam
```

CHAPTER

SEVEN

CUTADAPT

117.1 Introduction

`Cutadapt` finds and removes adapter sequences, primers, poly-A tails and other types of unwanted sequence from your high-throughput sequencing reads.

For more information, please check its website: https://biocontainers.pro/tools/cutadapt and its home page: https://cutadapt.readthedocs.io/en/stable/.

117.2 Versions

- 3.4-py38
- 3.7-py37

117.3 Commands

- cutadapt

117.4 Module

You can load the modules by:

```
module load biocontainers
module load cutadapt
```

117.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Cutadapt on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=cutadapt
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers cutadapt


cutadapt -a AACCGGTT -o output.fastq input.fastq
```

CHAPTER

EIGHT

CYVCF2

118.1 Introduction

`Cyvcf2` is a cython wrapper around htslib built for fast parsing of Variant Call Format (VCF) files.

For more information, please check its website: https://biocontainers.pro/tools/cyvcf2 and its home page on Github.

118.2 Versions

- 0.30.14-py37

118.3 Commands

- cyvcf2
- python
- python3

118.4 Module

You can load the modules by:

```
module load biocontainers
module load cyvcf2
```

118.5 Interactive job

To run Cyvcf2 interactively on our clusters:

```
(base) UserID@bell-fe00:~ $ sinteractive -N1 -n1 -t1:00:00 -A myallocation
salloc: Granted job allocation 12345869
salloc: Waiting for resource configuration
salloc: Nodes bell-a008 are ready for job
(base) UserID@bell-a008:~ $ module load biocontainers scanpy/1.8.2
(base) UserID@bell-a008:~ $ python
Python 3.7.12 |  packaged by conda-forge |  (default, Oct 26 2021, 06:08:53)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from cyvcf2 import VCF
```

118.6 Batch job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Cyvcf2 on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=cyvcf2
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers cyvcf2

cyvcf2 --help
cyvcf2 [OPTIONS] <vcf_file>
```

CHAPTER

NINE

DBG2OLC

119.1 Introduction

`Dbg2olc` is used for efficient assembly of large genomes using long erroneous reads of the third generation sequencing technologies.

For more information, please check its website: https://biocontainers.pro/tools/dbg2olc and its home page on Github.

119.2 Versions

- 20180222
- 20200723

119.3 Commands

- AssemblyStatistics
- DBG2OLC
- RunSparcConsensus.txt
- SelectLongestReads
- SeqIO.py
- Sparc
- SparseAssembler
- split_and_run_sparc.sh
- split_and_run_sparc.sh.bak
- split_reads_by_backbone.py

119.4 Module

You can load the modules by:

```
module load biocontainers
module load dbg2olc
```

119.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Dbg2olc on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=dbg2olc
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers dbg2olc

SelectLongestReads sum 600000000 longest 0 o TEST.fq f SRR1976948.abundtrim.subset.pe.fq
```

CHAPTER

DEEPBGC

120.1 Introduction

`Deepbgc` is a tool for BGC detection and classification using deep learning.

For more information, please check its website: https://biocontainers.pro/tools/deepbgc and its home page on Github.

120.2 Versions

- 0.1.26
- 0.1.30

120.3 Commands

- deepbgc

120.4 Module

You can load the modules by:

```
module load biocontainers
module load deepbgc
```

120.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Deepbgc on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=deepbgc
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers deepbgc

export DEEPBGC_DOWNLOADS_DIR=$PWD
deepbgc download
deepbgc pipeline genome.fa  -o output
```

CHAPTER

ONE

DEEPCONSENSUS

121.1 Introduction

DeepConsensus uses gap-aware sequence transformers to correct errors in Pacific Biosciences (PacBio) Circular Consensus Sequencing (CCS) data.

For more information, please check:
Docker hub: https://hub.docker.com/r/google/deepconsensus
Home page: https://github.com/google/deepconsensus

121.2 Versions

- 0.2.0

121.3 Commands

- deepconsensus
- ccs
- actc

121.4 Module

You can load the modules by:

```
module load biocontainers
module load deepconsensus
```

121.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run deepconsensus on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=deepconsensus
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers deepconsensus

deepconsensus run \
    --subreads_to_ccs=subreads_to_ccs.bam  \
    --ccs_fasta=ccs.fasta \
    --checkpoint=checkpoint-50 \
    --output=output.fastq \
    --batch_zmws=100
```

CHAPTER

TWO

DEEPSIGNAL2

122.1 Introduction

`Deepsignal2` is a deep-learning method for detecting DNA methylation state from Oxford Nanopore sequencing reads.

For more information, please check its home page on Github.

122.2 Versions

- 0.1.2

122.3 Commands

- deepsignal2
- call_modification_frequency.py
- combine_call_mods_freq_files.py
- combine_two_strands_frequency.py
- concat_two_files.py
- evaluate_mods_call.py
- filter_samples_by_label.py
- filter_samples_by_positions.py
- gff_reader.py
- randsel_file_rows.py
- shuffle_a_big_file.py
- split_freq_file_by_5mC_motif.py
- txt_formater.py

122.4 Module

You can load the modules by:

```
module load biocontainers
module load deepsignal2
```

122.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Deepsignal2 on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=deepsignal2
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers deepsignal2
```

CHAPTER

THREE

DEEPTOOLS

123.1 Introduction

`DeepTools` is a collection of user-friendly tools for normalization and visualization of deep-sequencing data.

For more information, please check its website: https://biocontainers.pro/tools/deeptools and its home page on Github.

123.2 Versions

- 3.5.1-py

123.3 Commands

- alignmentSieve
- bamCompare
- bamCoverage
- bamPEFragmentSize
- bigwigCompare
- computeGCBias
- computeMatrix
- computeMatrixOperations
- correctGCBias
- deeptools
- estimateReadFiltering
- estimateScaleFactor
- multiBamSummary
- multiBigwigSummary
- plotCorrelation

- plotCoverage
- plotEnrichment
- plotFingerprint
- plotHeatmap
- plotPCA
- plotProfile

123.4 Module

You can load the modules by:

```
module load biocontainers
module load deeptools
```

123.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run DeepTools on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 4
#SBATCH --job-name=deeptools
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers deeptools

bamCoverage  --normalizeUsing CPM -p 32  \
     --effectiveGenomeSize  11000000  \
     -b WT_coord_sorted.bam  \
     -o WT_coord_sorted.bw
```

CHAPTER

FOUR

DEEPVARIANT

124.1 Introduction

DeepVariant is a deep learning-based variant caller that takes aligned reads (in BAM or CRAM format), produces pileup image tensors from them, classifies each tensor using a convolutional neural network, and finally reports the results in a standard VCF or gVCF file.

For more information, please check:
Home page: https://github.com/google/deepvariant

124.2 Versions

- 1.0.0
- 1.1.0

124.3 Commands

- call_variants
- get-pip.py
- make_examples
- model_eval
- model_train
- postprocess_variants
- run-prereq.sh
- run_deepvariant
- run_deepvariant.py
- settings.sh
- show_examples
- vcf_stats_report

124.4 Module

You can load the modules by:

```
module load biocontainers
module load deepvariant
```

124.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run deepvariant on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 4
#SBATCH --job-name=deepvariant
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers deepvariant

INPUT_DIR="${PWD}/quickstart-testdata"
DATA_HTTP_DIR="https://storage.googleapis.com/deepvariant/quickstart-testdata"
mkdir -p ${INPUT_DIR}
wget -P ${INPUT_DIR} "${DATA_HTTP_DIR}"/NA12878_S1.chr20.10_10p1mb.bam
wget -P ${INPUT_DIR} "${DATA_HTTP_DIR}"/NA12878_S1.chr20.10_10p1mb.bam.bai
wget -P ${INPUT_DIR} "${DATA_HTTP_DIR}"/test_nist.b37_chr20_100kbp_at_10mb.bed
wget -P ${INPUT_DIR} "${DATA_HTTP_DIR}"/test_nist.b37_chr20_100kbp_at_10mb.vcf.gz
wget -P ${INPUT_DIR} "${DATA_HTTP_DIR}"/test_nist.b37_chr20_100kbp_at_10mb.vcf.gz.tbi
wget -P ${INPUT_DIR} "${DATA_HTTP_DIR}"/ucsc.hg19.chr20.unittest.fasta
wget -P ${INPUT_DIR} "${DATA_HTTP_DIR}"/ucsc.hg19.chr20.unittest.fasta.fai
wget -P ${INPUT_DIR} "${DATA_HTTP_DIR}"/ucsc.hg19.chr20.unittest.fasta.gz
wget -P ${INPUT_DIR} "${DATA_HTTP_DIR}"/ucsc.hg19.chr20.unittest.fasta.gz.fai
wget -P ${INPUT_DIR} "${DATA_HTTP_DIR}"/ucsc.hg19.chr20.unittest.fasta.gz.gzi

run_deepvariant --model_type=WGS --ref="${INPUT_DIR}"/ucsc.hg19.chr20.unittest.fasta  --
↪reads="${INPUT_DIR}"/NA12878_S1.chr20.10_10p1mb.bam  --regions "chr20:10,000,000-10,
↪010,000"  --output_vcf="output/output.vcf.gz"  --output_gvcf="output/output.g.vcf.gz" -
↪-intermediate_results_dir "output/intermediate_results_dir"  --num_shards=4
```

CHAPTER

FIVE

DELLY

125.1 Introduction

`Delly` is an integrated structural variant (SV) prediction method that can discover, genotype and visualize deletions, tandem duplications, inversions and translocations at single-nucleotide resolution in short-read massively parallel sequencing data.

For more information, please check its website: https://biocontainers.pro/tools/delly and its home page on Github.

125.2 Versions

- 0.9.1
- 1.0.3
- 1.1.3
- 1.1.5

125.3 Commands

- delly

125.4 Module

You can load the modules by:

```
module load biocontainers
module load delly
```

125.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Delly on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=delly
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers delly

delly call -x hg19.excl -o delly.bcf -g hg19.fa input.bam
delly filter -f somatic -o t1.pre.bcf -s samples.tsv t1.bcf
```

CHAPTER

SIX

DIAMOND

126.1 Introduction

`Diamond` is a sequence aligner for protein and translated DNA searches, designed for high performance analysis of big sequence data. The key features are:

- Pairwise alignment of proteins and translated DNA at 100x-10,000x speed of BLAST.
- Frameshift alignments for long read analysis.
- Low resource requirements and suitable for running on standard desktops or laptops.
- Various output formats, including BLAST pairwise, tabular and XML, as well as taxonomic classification.

Detailed about its usage can be found here: https://github.com/bbuchfink/diamond

126.2 Versions

- 2.0.13
- 2.0.14
- 2.0.15

126.3 Commands

- diamond makedb
- diamond prepdb
- diamond blastp
- diamond blastx
- diamond view
- diamond version
- diamond dbinfo
- diamond help
- diamond test

126.4 Module

You can load the modules by:

```
module load biocontainers
module load diamond/2.0.14
```

126.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run diamond on our our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 24
#SBATCH --job-name=diamond
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers diamond/2.0.14

diamond makedb  --in uniprot_sprot.fasta -d uniprot_sprot
diamond blastp -p 24 -q test.faa -d uniprot_sprot  --very-sensitive -o blastp_output.txt
```

CHAPTER

SEVEN

DNAIO

127.1 Introduction

`Dnaio` is a Python 3.7+ library for very efficient parsing and writing of FASTQ and also FASTA files.

For more information, please check its website: https://biocontainers.pro/tools/dnaio and its home page on Github.

127.2 Versions

- 0.8.1-py37

127.3 Commands

- python
- python3

127.4 Module

You can load the modules by:

```
module load biocontainers
module load dnaio
```

127.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Dnaio on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=dnaio
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers dnaio

python dnaio_test.py
```

CHAPTER

EIGHT

DRAGONFLYE

128.1 Introduction

Dragonflye is a pipeline that aims to make assembling Oxford Nanopore reads quick and easy.

For more information, please check:
BioContainers: https://biocontainers.pro/tools/dragonflye
Home page: https://github.com/rpetit3/dragonflye

128.2 Versions

- 1.0.13

128.3 Commands

- dragonflye

128.4 Module

You can load the modules by:

```
module load biocontainers
module load dragonflye
```

128.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run dragonflye on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 8
#SBATCH --job-name=dragonflye
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers dragonflye

dragonflye --cpus 8 \
     --outdir output \
     --reads SRR18498195.fastq
```

CHAPTER

NINE

DREP

129.1 Introduction

`Drep` is a python program for rapidly comparing large numbers of genomes.

For more information, please check its website: https://biocontainers.pro/tools/drep and its home page on Github.

129.2 Versions

- 3.2.2

129.3 Commands

- dRep

129.4 Module

You can load the modules by:

```
module load biocontainers
module load drep
```

129.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Drep on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=drep
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers drep

dRep compare compare_out -g tests/genomes/*
dRep dereplicate dereplicate_out -g tests/genomes/*
```

CHAPTER

DROPEST

130.1 Introduction

`Dropest` is a pipeline for initial analysis of droplet-based single-cell RNA-seq data.

For more information, please check its website: https://biocontainers.pro/tools/dropest and its home page on Github.

130.2 Versions

- 0.8.6

130.3 Commands

- dropest
- droptag
- dropReport.Rsc
- R
- Rscript

130.4 Module

You can load the modules by:

```
module load biocontainers
module load dropest
```

130.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Dropest on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=dropest
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers dropest

dropest -f -c 10x.xml  -C 1200 neurons_900_possorted_genome_bam.bam
```

CHAPTER

ONE

DSUITE

131.1 Introduction

`Dsuite` is a fast C++ implementation, allowing genome scale calculations of the D and f4-ratio statistics across all combinations of tens or hundreds of populations or species directly from a variant call format (VCF) file.

For more information, please check its home page on Github.

131.2 Versions

- 0.4.r43
- 0.5.r44

131.3 Commands

- Dsuite
- dtools.py
- DtriosParallel

131.4 Module

You can load the modules by:

```
module load biocontainers
module load dsuite
```

131.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Dsuite on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 4
#SBATCH --job-name=dsuite
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers dsuite

Dsuite Dtrios -c -n no_geneflow -t simulated_tree_no_geneflow.nwk chr1_no_geneflow.vcf.
↪gz species_sets.txt
```

CHAPTER

TWO

EASYSFS

132.1 Introduction

`easySFS` is a tool for the effective selection of population size projection for construction of the site frequency spectrum.

For more information, please check its home page on Github.

132.2 Versions

- 1.0

132.3 Commands

- easySFS.py

132.4 Module

You can load the modules by:

```
module load biocontainers
module load easysfs
```

132.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run easySFS on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=easysfs
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers easysfs

easySFS.py -i example_files/wcs_1200.vcf -p example_files/wcs_pops.txt --preview -a
easySFS.py -i example_files/wcs_1200.vcf -p example_files/wcs_pops.txt -a --proj=7,7
```

CHAPTER

THREE

EDTA

133.1 Introduction

`Edta` is is developed for automated whole-genome de-novo TE annotation and benchmarking the annotation performance of TE libraries.

For more information, please check its website: https://biocontainers.pro/tools/edta and its home page on Github.

Note: Running EDTA, please use the command like this:
EDTA.pl [OPTIONS]

DO NOT call it 'perl EDTA.pl'

133.2 Versions

- 1.9.6
- 2.0.0

133.3 Commands

- EDTA.pl
- EDTA_processI.pl
- EDTA_raw.pl
- FET.pl
- bdf2gdfont.pl
- buildRMLibFromEMBL.pl
- buildSummary.pl
- calcDivergenceFromAlign.pl
- cd-hit-2d-para.pl
- cd-hit-clstr_2_blm8.pl
- cd-hit-div.pl

- cd-hit-para.pl
- check_result.pl
- clstr2tree.pl
- clstr2txt.pl
- clstr2xml.pl
- clstr_cut.pl
- clstr_list.pl
- clstr_list_sort.pl
- clstr_merge.pl
- clstr_merge_noorder.pl
- clstr_quality_eval.pl
- clstr_quality_eval_by_link.pl
- clstr_reduce.pl
- clstr_renumber.pl
- clstr_rep.pl
- clstr_reps_faa_rev.pl
- clstr_rev.pl
- clstr_select.pl
- clstr_select_rep.pl
- clstr_size_histogram.pl
- clstr_size_stat.pl
- clstr_sort_by.pl
- clstr_sort_prot_by.pl
- clstr_sql_tbl.pl
- clstr_sql_tbl_sort.pl
- convert_MGEScan3.0.pl
- convert_ltr_struc.pl
- convert_ltrdetector.pl
- createRepeatLandscape.pl
- down_tRNA.pl
- dupliconToSVG.pl
- filter_rt.pl
- genome_plot.pl
- genome_plot2.pl
- genome_plot_svg.pl
- getRepeatMaskerBatch.pl

- legacy_blast.pl
- lib-test.pl
- make_multi_seq.pl
- maskFile.pl
- plot_2d.pl
- plot_len1.pl
- rmOut2Fasta.pl
- rmOutToGFF3.pl
- rmToUCSCTables.pl
- update_blastdb.pl
- viewMSA.pl
- wublastToCrossmatch.pl

133.4 Module

You can load the modules by:

```
module load biocontainers
module load edta
```

133.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Edta on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 10
#SBATCH --job-name=edta
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers edta

EDTA.pl --genome genome.fa --cds genome.cds.fa --curatedlib EDTA/database/rice6.9.5.
↪liban --exclude genome.exclude.bed --overwrite 1 --sensitive 1 --anno 1 --evaluate 1 --
↪threads 10
```

CHAPTER

FOUR

EGGNOG-MAPPER

134.1 Introduction

`Eggnog-mapper` is a tool for fast functional annotation of novel sequences.

For more information, please check its website: https://biocontainers.pro/tools/eggnog-mapper and its home page on Github.

134.2 Versions

- 2.1.7

134.3 Commands

- create_dbs.py
- download_eggnog_data.py
- emapper.py
- hmm_mapper.py
- hmm_server.py
- hmm_worker.py
- vba_extract.py

134.4 Module

You can load the modules by:

```
module load biocontainers
module load eggnog-mapper
```

134.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Eggnog-mapper on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 24
#SBATCH --job-name=eggnog-mapper
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers eggnog-mapper

emapper.py -i proteins.faa --cpu 24 -o protein.out
emapper.py -m diamond --itype CDS -i cDNA.fasta -o cdna.out --cpu 24
```

CHAPTER

FIVE

EMBOSS

135.1 Introduction

`Emboss` is "The European Molecular Biology Open Software Suite".

For more information, please check its website: https://biocontainers.pro/tools/emboss and its home page: http://emboss.open-bio.org.

135.2 Versions

- 6.6.0

135.3 Commands

- aaindexextract
- abiview
- acdc
- acdgalaxy
- acdlog
- acdpretty
- acdtable
- acdtrace
- acdvalid
- aligncopy
- aligncopypair
- antigenic
- assemblyget
- backtranambig
- backtranseq

- banana
- biosed
- btwisted
- cachedas
- cachedbfetch
- cacheebeyesearch
- cacheensembl
- cai
- chaos
- charge
- checktrans
- chips
- cirdna
- codcmp
- codcopy
- coderet
- compseq
- cons
- consambig
- cpgplot
- cpgreport
- cusp
- cutgextract
- cutseq
- dan
- dbiblast
- dbifasta
- dbiflat
- dbigcg
- dbtell
- dbxcompress
- dbxedam
- dbxfasta
- dbxflat
- dbxgcg
- dbxobo

- dbxreport
- dbxresource
- dbxstat
- dbxtax
- dbxuncompress
- degapseq
- density
- descseq
- diffseq
- distmat
- dotmatcher
- dotpath
- dottup
- dreg
- drfinddata
- drfindformat
- drfindid
- drfindresource
- drget
- drtext
- edamdef
- edamhasinput
- edamhasoutput
- edamisformat
- edamisid
- edamname
- edialign
- einverted
- embossdata
- embossupdate
- embossversion
- emma
- emowse
- entret
- epestfind
- eprimer3

- eprimer32
- equicktandem
- est2genome
- etandem
- extractalign
- extractfeat
- extractseq
- featcopy
- featmerge
- featreport
- feattext
- findkm
- freak
- fuzznuc
- fuzzpro
- fuzztran
- garnier
- geecee
- getorf
- godef
- goname
- helixturnhelix
- hmoment
- iep
- infoalign
- infoassembly
- infobase
- inforesidue
- infoseq
- isochore
- jaspextract
- jaspscan
- jembossctl
- lindna
- listor
- makenucseq

- makeprotseq
- marscan
- maskambignuc
- maskambigprot
- maskfeat
- maskseq
- matcher
- megamerger
- merger
- msbar
- mwcontam
- mwfilter
- needle
- needleall
- newcpgreport
- newcpgseek
- newseq
- nohtml
- noreturn
- nospace
- notab
- notseq
- nthseq
- nthseqset
- octanol
- oddcomp
- ontocount
- ontoget
- ontogetcommon
- ontogetdown
- ontogetobsolete
- ontogetroot
- ontogetsibs
- ontogetup
- ontoisobsolete
- ontotext

- palindrome
- pasteseq
- patmatdb
- patmatmotifs
- pepcoil
- pepdigest
- pepinfo
- pepnet
- pepstats
- pepwheel
- pepwindow
- pepwindowall
- plotcon
- plotorf
- polydot
- preg
- prettyplot
- prettyseq
- primersearch
- printsextract
- profit
- prophecy
- prophet
- prosextract
- pscan
- psiphi
- rebaseextract
- recoder
- redata
- refseqget
- remap
- restover
- restrict
- revseq
- runJemboss.sh
- seealso

- seqcount
- seqmatchall
- seqret
- seqretsetall
- seqretsplit
- seqxref
- seqxrefget
- servertell
- showalign
- showdb
- showfeat
- showorf
- showpep
- showseq
- showserver
- shuffleseq
- sigcleave
- silent
- sirna
- sixpack
- sizeseq
- skipredundant
- skipseq
- splitsource
- splitter
- stretcher
- stssearch
- supermatcher
- syco
- taxget
- taxgetdown
- taxgetrank
- taxgetspecies
- taxgetup
- tcode
- textget

- textsearch
- tfextract
- tfm
- tfscan
- tmap
- tranalign
- transeq
- trimest
- trimseq
- trimspace
- twofeat
- union
- urlget
- variationget
- vectorstrip
- water
- whichdb
- wobble
- wordcount
- wordfinder
- wordmatch
- wossdata
- wossinput
- wossname
- wossoperation
- wossoutput
- wossparam
- wosstopic
- xmlget
- xmltext
- yank

135.4 Module

You can load the modules by:

```
module load biocontainers
module load emboss
```

135.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Emboss on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=emboss
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers emboss
```

CHAPTER

SIX

ENSEMBL-VEP

136.1 Introduction

Ensembl-vep(Ensembl Variant Effect Predictor) predicts the functional effects of genomic variants.

For more information, please check:
Docker hub: https://hub.docker.com/r/ensemblorg/ensembl-vep
Home page: https://github.com/Ensembl/ensembl-vep

136.2 Versions

- 106.1
- 107.0

136.3 Commands

- vep
- haplo
- variant_recoder

136.4 Module

You can load the modules by:

```
module load biocontainers
module load ensembl-vep
```

136.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run ensembl-vep on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=ensembl-vep
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers ensembl-vep

haplo -i bos_taurus_UMD3.1.vcf -o out.txt
```

CHAPTER

SEVEN

EPIC2

137.1 Introduction

`Epic2` is an ultraperformant Chip-Seq broad domain finder based on SICER.

For more information, please check its website: https://biocontainers.pro/tools/epic2 and its home page on Github.

137.2 Versions

- 0.0.51-py39

137.3 Commands

- epic2
- epic2-bw
- epic2-df

137.4 Module

You can load the modules by:

```
module load biocontainers
module load epic2
```

137.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Epic2 on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=epic2
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers epic2

epic2 -t /examples/test.bed.gz \
  -c /examples/control.bed.gz \
  > deleteme.txt
```

CHAPTER

EIGHT

EVIDENCEMODELER

138.1 Introduction

`Evidencemodeler` is a software combines ab intio gene predictions and protein and transcript alignments into weighted consensus gene structures.

For more information, please check its website: https://biocontainers.pro/tools/evidencemodeler and its home page on Github.

138.2 Versions

- 1.1.1

138.3 Commands

- evidence_modeler.pl
- BPbtab.pl
- EVMLite.pl
- EVM_to_GFF3.pl
- convert_EVM_outputs_to_GFF3.pl
- create_weights_file.pl
- execute_EVM_commands.pl
- extract_complete_proteins.pl
- gff3_file_to_proteins.pl
- gff3_gene_prediction_file_validator.pl
- gff_range_retriever.pl
- partition_EVM_inputs.pl
- recombine_EVM_partial_outputs.pl
- summarize_btab_tophits.pl

- write_EVM_commands.pl

138.4 Module

You can load the modules by:

```
module load biocontainers
module load evidencemodeler
```

138.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Evidencemodeler on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=evidencemodeler
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers evidencemodeler


evidence_modeler.pl --genome genome.fasta \
                    --weights weights.txt \
                    --gene_predictions gene_predictions.gff3 \
                    --protein_alignments protein_alignments.gff3 \
                    --transcript_alignments transcript_alignments.gff3 \
                  > evm.out
```

CHAPTER

NINE

EXONERATE

139.1 Introduction

`Exonerate` is a generic tool for pairwise sequence comparison/alignment.

For more information, please check its home page:
https://www.ebi.ac.uk/about/vertebrate-genomics/software/exonerate.

139.2 Versions

- 2.4.0

139.3 Commands

- exonerate

139.4 Module

You can load the modules by:

```
module load biocontainers
module load exonerate
```

139.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Exonerate on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=exonerate
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers exonerate

exonerate  -m genome2genome  cms.fasta cmm.fasta > cm_vs_cs.out
```

FASTA3

140.1 Introduction

`Fasta3` is a suite of programs for searching nucleotide or protein databases with a query sequence.

For more information, please check its website: https://biocontainers.pro/tools/fasta3 and its home page on Github.

140.2 Versions

- 36.3.8

140.3 Commands

- fasta36
- fastf36
- fastm36
- fasts36
- fastx36
- fasty36
- ggsearch36
- glsearch36
- lalign36
- ssearch36
- tfastf36
- tfastm36
- tfasts36
- tfastx36
- tfasty36

140.4 Module

You can load the modules by:

```
module load biocontainers
module load fasta3
```

140.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Fasta3 on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=fasta3
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers fasta3

fasta36 input.fasta genome.fasta
```

CHAPTER

ONE

FASTANI

141.1 Introduction

`FastANI` is developed for fast alignment-free computation of whole-genome Average Nucleotide Identity (ANI).

For more information, please check its website: https://biocontainers.pro/tools/fastani and its home page on Github.

141.2 Versions

- 1.32
- 1.33

141.3 Commands

- fastANI

141.4 Module

You can load the modules by:

```
module load biocontainers
module load fastani
```

141.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run FastANI on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=fastani
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers fastani

fastANI -q cmm.fasta -r cms.fasta -o cm_cs_out

fastANI -q cmm.fasta -r cms.fasta  --visualize -o cm_cs_visualize_out
```

CHAPTER

TWO

FASTP

142.1 Introduction

`Fastp` is an ultra-fast all-in-one FASTQ preprocessor (QC/adapters/trimming/filtering/splitting/merging, etc).

For more information, please check its website: https://biocontainers.pro/tools/fastp and its home page on Github.

142.2 Versions

- 0.20.1
- 0.23.2

142.3 Commands

- fastp

142.4 Module

You can load the modules by:

```
module load biocontainers
module load fastp
```

142.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Fastp on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=fastp
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers fastp

fastp -i input_1.fastq  -I input_2.fastq -o out.R1.fq.gz -O out.R2.fq.gz
```

CHAPTER

THREE

FASTQC

143.1 Introduction

`FastQC` aims to provide a simple way to do some quality control checks on raw sequence data coming from high throughput sequencing pipelines. It provides a modular set of analyses which you can use to give a quick impression of whether your data has any problems of which you should be aware before doing any further analysis.

For more information, please check its website: https://biocontainers.pro/tools/fastqc and its home page: https://www.bioinformatics.babraham.ac.uk/projects/fastqc/.

143.2 Versions

- 0.11.9

143.3 Commands

- fastqc

143.4 Module

You can load the modules by:

```
module load biocontainers
module load fastqc
```

143.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Fastqc on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 4
#SBATCH --job-name=fastqc
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers fastqc

fastqc -o fastqc_out -t 4 FASTQ1 FASTQ2
```

CHAPTER

FOUR

FASTQ_PAIR

144.1 Introduction

`Fastq_pair` is used to match up paired end fastq files quickly and efficiently.

For more information, please check its website: https://biocontainers.pro/tools/fastq_pair and its home page on Github.

144.2 Versions

- 1.0

144.3 Commands

- fastq_pair

144.4 Module

You can load the modules by:

```
module load biocontainers
module load fastq_pair
```

144.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Fastq_pair on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=fastq_pair
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers fastq_pair

fastq_pair seq_1.fastq  seq_2.fastq
```

CHAPTER

FIVE

FASTQ-SCAN

145.1 Introduction

Fastq-scan reads a FASTQ from STDIN and outputs summary statistics (read lengths, per-read qualities, per-base qualities) in JSON format.

For more information, please check:
Docker hub: https://hub.docker.com/r/staphb/fastq-scan
Home page: https://github.com/rpetit3/fastq-scan

145.2 Versions

- 1.0.0

145.3 Commands

- fastq-scan

145.4 Module

You can load the modules by:

```
module load biocontainers
module load fastq-scan
```

145.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run fastq-scan on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=fastq-scan
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers fastq-scan

cat example-q33.fq | fastq-scan -g 150000
```

CHAPTER

SIX

FASTSPAR

146.1 Introduction

`Fastspar` is a tool for rapid and scalable correlation estimation for compositional data.

For more information, please check its website: https://biocontainers.pro/tools/fastspar and its home page on Github.

146.2 Versions

- 1.0.0

146.3 Commands

- fastspar
- fastspar_bootstrap
- fastspar_pvalues
- fastspar_reduce

146.4 Module

You can load the modules by:

```
module load biocontainers
module load fastspar
```

146.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Fastspar on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=fastspar
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers fastspar
```

CHAPTER

SEVEN

FASTSTRUCTURE

147.1 Introduction

`fastStructure` is an algorithm for inferring population structure from large SNP genotype data. It is based on a variational Bayesian framework for posterior inference and is written in Python2.x.

Note: programs "structure.py", "chooseK.py" and "distruct.py" are standalone executable and should be called by name directly ("structure.py", etc). DO NOT invoke them as "python structure.py", or as "python /usr/local/bin/structure.py", this will not work!

Note: This containers lacks X11 libraries, so GUI plots with 'distruct.py' do not work. Instead, we need to tell the underlying Matplotlib to use a non-interactive plotting backend (to file). The easiest and most flexible way is to use the MPLBACKEND environment variable: env MPLBACKEND="svg" distruct.py –output myplot.svg .....

Available backends in this container:

Backend Filetypes Description agg png raster graphics – high quality PNG output ps ps eps vector graphics – Postscript output pdf pdf vector graphics – Portable Document Format svg svg vector graphics – Scalable Vector Graphics

Default MPLBACKEND="agg" (for PNG format output).

For more information, please check its website: https://biocontainers.pro/tools/faststructure and its home page on Github.

147.2 Versions

- 1.0-py27

147.3 Commands

- structure.py
- chooseK.py
- distruct.py

147.4 Module

You can load the modules by:

```
module load biocontainers
module load faststructure
```

147.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run fastStructure on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=faststructure
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers faststructure
```

CHAPTER

EIGHT

FASTTREE

148.1 Introduction

`FastTree` infers approximately-maximum-likelihood phylogenetic trees from alignments of nucleotide or protein sequences. FastTree can handle alignments with up to a million of sequences in a reasonable amount of time and memory.

Detailed usage can be found here: http://www.microbesonline.org/fasttree/

148.2 Versions

- 2.1.10
- 2.1.11

148.3 Commands

- fasttree
- FastTree
- FastTreeMP

Note: `fasttree` and `FastTree` are the same program, and they only support one CPU. If you want to use multiple CPUs, please use `FastTreeMP` and also set the `OMP_NUM_THREADS` to the number of cores you requested.

148.4 Module

You can load the modules by:

```
module load biocontainers
module load fasttree
```

148.5 Example job using single CPU

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run FastTree on our our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 20:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=fasttree
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers fasttree

FastTree alignmentfile > treefile
```

148.6 Example job using multiple CPUs

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run FastTree on our our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 20:00:00
#SBATCH -N 1
#SBATCH -n 24
#SBATCH --job-name=FastTreeMP
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers fasttree

export OMP_NUM_THREADS=24

FastTreeMP alignmentfile > treefile
```

CHAPTER

NINE

FASTX-TOOLKIT

149.1 Introduction

`FASTX-Toolkit` is a collection of command line tools for Short-Reads FASTA/FASTQ files preprocessing.

For more information, please check its website: https://biocontainers.pro/tools/fastx_toolkit and its home page on Github.

149.2 Versions

- 0.0.14

149.3 Commands

- fasta_clipping_histogram.pl
- fasta_formatter
- fasta_nucleotide_changer
- fastq_masker
- fastq_quality_boxplot_graph.sh
- fastq_quality_converter
- fastq_quality_filter
- fastq_quality_trimmer
- fastq_to_fasta
- fastx_artifacts_filter
- fastx_barcode_splitter.pl
- fastx_clipper
- fastx_collapser
- fastx_nucleotide_distribution_graph.sh
- fastx_nucleotide_distribution_line_graph.sh

- fastx_quality_stats
- fastx_renamer
- fastx_reverse_complement
- fastx_trimmer
- fastx_uncollapser

149.4 Module

You can load the modules by:

```
module load biocontainers
module load fastx_toolkit
```

149.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run FASTX-Toolkit on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=fastx_toolkit
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers fastx_toolkit
```

FILTLONG

150.1 Introduction

`Filtlong` is a tool for filtering long reads by quality. It can take a set of long reads and produce a smaller, better subset. It uses both read length (longer is better) and read identity (higher is better) when choosing which reads pass the filter.

For more information, please check its website: https://biocontainers.pro/tools/filtlong and its home page on Github.

150.2 Versions

- 0.2.1

150.3 Commands

- filtlong

150.4 Module

You can load the modules by:

```
module load biocontainers
module load filtlong
```

150.5 Example job

> **Warning:** Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Filtlong on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=filtlong
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers filtlong
```

CHAPTER

ONE

FLYE

151.1 Introduction

`Flye`: Fast and accurate de novo assembler for single molecule sequencing reads.

For more information, please check its website: https://biocontainers.pro/tools/flye and its home page on Github.

151.2 Versions

- 2.9.1
- 2.9
- 2.9-py38

151.3 Commands

- flye

151.4 Module

You can load the modules by:

```
module load biocontainers
module load flye
```

151.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Flye on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 12
#SBATCH --job-name=flye
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers flye

flye --pacbio-raw E.coli_PacBio_40x.fasta --out-dir out_pacbio --threads 12
flye --nano-raw Loman_E.coli_MAP006-1_2D_50x.fasta --out-dir out_nano --threads 12
```

CHAPTER

TWO

FRAGGENESCAN

152.1 Introduction

`Fraggenescan` is an application for finding (fragmented) genes in short reads. It can also be applied to predict prokaryotic genes in incomplete assemblies or complete genomes.

For more information, please check its website: https://biocontainers.pro/tools/fraggenescan and its home page on Github.

152.2 Versions

- 1.31

152.3 Commands

- FragGeneScan
- run_FragGeneScan.pl

152.4 Module

You can load the modules by:

```
module load biocontainers
module load fraggenescan
```

152.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Fraggenescan on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=fraggenescan
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers fraggenescan

FragGeneScanRs -t 454_10 < example/NC_000913-454.fna > example/NC_000913-454.faa
```

CHAPTER

THREE

FRAGGENESCANRS

153.1 Introduction

FragGeneScanRs is a better and faster Rust implementation of the FragGeneScan gene prediction model for short and error-prone reads. Its command line interface is backward compatible and adds extra features for more flexible usage. Compared to the original C implementation, shotgun metagenomic reads are processed up to 22 times faster using a single thread, with better scaling for multithreaded execution.

For more information, please check:
Home page: https://github.com/unipept/FragGeneScanRs

153.2 Versions

- 1.1.0

153.3 Commands

- FragGeneScanRs

153.4 Module

You can load the modules by:

```
module load biocontainers
module load fraggenescanrs
```

153.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run fraggenescanrs on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=fraggenescanrs
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers fraggenescanrs
```

CHAPTER

FOUR

FREEBAYES

154.1 Introduction

`Freebayes` is a Bayesian genetic variant detector designed to find small polymorphisms, specifically SNPs (single-nucleotide polymorphisms), indels (insertions and deletions), MNPs (multi-nucleotide polymorphisms), and complex events (composite insertion and substitution events) smaller than the length of a short-read sequencing alignment.

For more information, please check its website: https://biocontainers.pro/tools/freebayes and its home page on Github.

154.2 Versions

- 1.3.5-py38
- 1.3.6

154.3 Commands

- freebayes
- freebayes-parallel

154.4 Module

You can load the modules by:

```
module load biocontainers
module load freebayes
```

154.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Freebayes on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=freebayes
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers freebayes

freebayes -f ref.fa aln.cram >var.vcf
```

CHAPTER

FIVE

FSEQ

155.1 Introduction

`Fseq` is a feature density estimator for high-throughput sequence tags.

For more information, please check its home page: https://fureylab.web.unc.edu/software/fseq/.

155.2 Versions

- 2.0.3

155.3 Commands

- fseq2

155.4 Module

You can load the modules by:

```
module load biocontainers
module load fseq
```

155.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Fseq on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=fseq
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers fseq
```

CHAPTER

SIX

FUNANNOTATE

156.1 Introduction

`Funannotate` is a genome prediction, annotation, and comparison software package.

For more information, please check its | Docker hub: https://hub.docker.com/r/nextgenusfs/funannotate and its home page on Github.

156.2 Versions

- 1.8.10

156.3 Commands

- funannotate

156.4 Module

You can load the modules by:

```
module load biocontainers
module load funannotate
```

156.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Funannotate on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 12
#SBATCH --job-name=funannotate
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers funannotate

funannotate clean -i genome.fa -o genome_cleaned.fa
funannotate sort -i genome_cleaned.fa -o genome_cleaned_sorted.fa
funannotate predict -i genome_cleaned_sorted.fa -o predict_out --species "arabidopsis" --
↪rna_bam  RNAseq.bam --cpus 12
```

CHAPTER

SEVEN

FWDPY11

157.1 Introduction

Fwdpy11 is a Python package for forward-time population genetic simulation.

For more information, please check:
Docker hub: https://hub.docker.com/r/molpopgen/fwdpy11
Home page: https://github.com/molpopgen/fwdpy11

157.2 Versions

- 0.18.1

157.3 Commands

- python3
- python

157.4 Module

You can load the modules by:

```
module load biocontainers
module load fwdpy11
```

157.5 Interactive job

To run fwdpy11 interactively on our clusters:

```
(base) UserID@bell-fe00:~ $ sinteractive -N1 -n12 -t4:00:00 -A myallocation
salloc: Granted job allocation 12345869
salloc: Waiting for resource configuration
salloc: Nodes bell-a008 are ready for job
(base) UserID@bell-a008:~ $ module load biocontainers fwdpy11
(base) UserID@bell-a008:~ $ python
Python 3.8.10 (default, Mar 15 2022, 12:22:08)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import fwdpy11
>>> pop = fwdpy11.DiploidPopulation(100, 1000.0)
>>> print(f"N = {pop.N}, L = {pop.tables.genome_length}")
```

157.6 Batch job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run fwdpy11 on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=fwdpy11
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers fwdpy11

python script.py
```

CHAPTER

EIGHT

GADMA

158.1 Introduction

GADMA is a command-line tool. Basic pipeline presents a series of launches of the genetic algorithm folowed by local search optimization and infers demographic history from the Allele Frequency Spectrum of multiple populations (up to three).

For more information, please check:
BioContainers: https://biocontainers.pro/tools/gadma
Home page: https://github.com/ctlab/GADMA

158.2 Versions

- 2.0.0rc21

158.3 Commands

- gadma
- python
- python3

158.4 Module

You can load the modules by:

```
module load biocontainers
module load gadma
```

158.5 Interactive job

To run GADMA interactively on our clusters:

```
(base) UserID@bell-fe00:~ $ sinteractive -N1 -n12 -t4:00:00 -A myallocation
salloc: Granted job allocation 12345869
salloc: Waiting for resource configuration
salloc: Nodes bell-a008 are ready for job
(base) UserID@bell-a008:~ $ module load biocontainers gadma
(base) UserID@bell-a008:~ $ python
Python 3.8.13 |  packaged by conda-forge |  (default, Mar 25 2022, 06:04:10)
[GCC 10.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from gadma import *
```

158.6 Batch job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run gadma on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=gadma
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers gadma

gadma -p params_file
```

CHAPTER

NINE

GAMBIT

159.1 Introduction

GAMBIT (Genomic Approximation Method for Bacterial Identification and Tracking) is a tool for rapid taxonomic identification of microbial pathogens.

For more information, please check:
Docker hub: https://hub.docker.com/r/staphb/gambit
Home page: https://github.com/jlumpe/gambit

159.2 Versions

- 0.5.0

159.3 Commands

- gambit

159.4 Module

You can load the modules by:

```
module load biocontainers
module load gambit
```

159.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run gambit on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=gambit
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers gambit

gambit -d database query -o results.csv *.fasta
```

GAMMA

160.1 Introduction

GAMMA (Gene Allele Mutation Microbial Assessment) is a command line tool that finds gene matches in microbial genomic data using protein coding (rather than nucleotide) identity, and then translates and annotates the match by providing the type (i.e., mutant, truncation, etc.) and a translated description (i.e., Y190S mutant, truncation at residue 110, etc.). Because microbial gene families often have multiple alleles and existing databases are rarely exhaustive, GAMMA is helpful in both identifying and explaining how unique alleles differ from their closest known matches.

For more information, please check:
Docker hub: https://hub.docker.com/r/staphb/gamma
Home page: https://github.com/rastanton/GAMMA

160.2 Versions

- 1.4
- 2.2

160.3 Commands

- GAMMA-S.py
- GAMMA.py

160.4 Module

You can load the modules by:

```
module load biocontainers
module load gamma
```

160.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run gamma on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=gamma
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers gamma

GAMMA.py DHQP1701672_complete_genome.fasta ResFinderDB_Combined_05-06-20.fsa GAMMA_Test
```

CHAPTER

ONE

GATK

161.1 Introduction

GATK (Genome Analysis Toolkit) is a collection of command-line tools for analyzing high-throughput sequencing data with a primary focus on variant discovery.

For more information, please check its website: https://biocontainers.pro/tools/gatk and its home page: https://www.broadinstitute.org/gatk/.

161.2 Versions

- 3.8

161.3 Commands

- gatk3

161.4 Module

You can load the modules by:

```
module load biocontainers
module load gatk
```

161.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run GATK on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 24
#SBATCH --job-name=gatk
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers gatk

gatk3 -T HaplotypeCaller  \
    -nct 24  -R hg38.fa \
    -I 19P0126636WES.sorted.bam \
     -o 19P0126636WES.HC.vcf
```

CHAPTER

TWO

GATK4

162.1 Introduction

`GATK (Genome Analysis Toolkit)` is a collection of command-line tools for analyzing high-throughput sequencing data with a primary focus on variant discovery. Detailed usage can be found here: https://www.broadinstitute.org/gatk/.

162.2 Versions

- 4.2.0
- 4.2.5.0
- 4.2.6.1
- 4.3.0.0

162.3 Commands

gatk

162.4 Module

You can load the modules by:

```
module load biocontainers
module load gatk4/4.2.5.0
```

162.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run gatk4 our our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 20:00:00
#SBATCH -N 1
#SBATCH -n 24
#SBATCH --job-name=gatk4
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers gatk4/4.2.5.0

gatk  --java-options "-Xmx12G -XX:ParallelGCThreads=24" HaplotypeCaller -R hg38.fa -I␣
↪19P0126636WES.sorted.bam  -O 19P0126636WES.HC.vcf --sample-name 19P0126636
```

CHAPTER

THREE

GEMMA

163.1 Introduction

`Gemma` is a software toolkit for fast application of linear mixed models (LMMs) and related models to genome-wide association studies (GWAS) and other large-scale data sets.

For more information, please check its website: https://biocontainers.pro/tools/gemma and its home page on Github.

163.2 Versions

- 0.98.3

163.3 Commands

- gemma

163.4 Module

You can load the modules by:

```
module load biocontainers
module load gemma
```

163.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Gemma on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=gemma
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers gemma

gemma -g ./example/mouse_hs1940.geno.txt.gz -p ./example/mouse_hs1940.pheno.txt \
    -gk -o mouse_hs1940

gemma -g ./example/mouse_hs1940.geno.txt.gz \
    -p ./example/mouse_hs1940.pheno.txt -n 1 -a ./example/mouse_hs1940.anno.txt \
    -k ./output/mouse_hs1940.cXX.txt -lmm -o mouse_hs1940_CD8_lmm
```

CHAPTER

FOUR

GEMOMA

164.1 Introduction

Gene Model Mapper (GeMoMa) is a homology-based gene prediction program. GeMoMa uses the annotation of protein-coding genes in a reference genome to infer the annotation of protein-coding genes in a target genome. Thereby, GeMoMa utilizes amino acid sequence and intron position conservation. In addition, GeMoMa allows to incorporate RNA-seq evidence for splice site prediction.

For more information, please check:
BioContainers: https://biocontainers.pro/tools/gemoma
Home page: http://www.jstacs.de/index.php/GeMoMa

164.2 Versions

- 1.7.1

164.3 Commands

- GeMoMa

164.4 Module

You can load the modules by:

```
module load biocontainers
module load gemoma
```

164.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run gemoma on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=gemoma
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers gemoma
```

CHAPTER

FIVE

GENEMARK-ES/ET/EP

165.1 Introduction

`GeneMark-ES/ET/EP` contains GeneMark-ES, GeneMark-ET and GeneMark-EP+ algorithms.

165.2 Versions

- 4.68
- 4.69

165.3 Commands

- bed_to_gff.pl
- bp_seq_select.pl
- build_mod.pl
- calc_introns_from_gtf.pl
- change_path_in_perl_scripts.pl
- compare_intervals_exact.pl
- gc_distr.pl
- get_below_gc.pl
- get_sequence_from_GTF.pl
- gmes_petap.pl
- hc_exons2hints.pl
- histogram.pl
- make_nt_freq_mat.pl
- parse_ET.pl
- parse_by_introns.pl
- parse_gibbs.pl
- parse_set.pl

- predict_genes.pl
- reformat_gff.pl
- rescale_gff.pl
- rnaseq_introns_to_gff.pl
- run_es.pl
- run_hmm_pbs.pl
- scan_for_bp.pl
- star_to_gff.pl
- verify_evidence_gmhmm.pl

165.4 Academic license

To use GeneMark, users need to download license files by yourself.

Go to the GeneMark web site: http://exon.gatech.edu/GeneMark/license_download.cgi. Check the boxes for `GeneMark-ES/ET/EP ver 4.69_lic` and `LINUX 64` next to it, fill out the form, then click "I agree". In the next page, right click and copy the link addresses for `64 bit` licenss. Paste the link addresses in the commands below:

```
cd $HOME
wget "replace with license URL"
zcat gm_key_64.gz > .gm_key
```

165.5 Module

You can load the modules by:

```
module load biocontainers
module load genemark/4.68
```

165.6 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run GeneMark on our cluster:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 24
#SBATCH --job-name=genemark
#SBATCH --mail-type=FAIL,BEGIN,END
```

(continues on next page)

(continued from previous page)

```
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers genemark/4.68

gmes_petap.pl --ES  --cores 24 --sequence scaffolds.fasta
```

CHAPTER

SIX

GENEMARKS-2

166.1 Introduction

GeneMarkS-2 combines GeneMark.hmm (prokaryotic) and GeneMark (prokaryotic) with a self-training procedure that determines parameters of the models of both GeneMark.hmm and GeneMark.

For more information, please check:

The users need to download your own licence key from GeneMark website and copy key "gm_key" into users' home directory as: cp gm_key ~/.gm_key | Home page: http://opal.biology.gatech.edu/GeneMark/

166.2 Versions

- 1.14_1.25

166.3 Commands

- gms2.pl
- biogem
- compp
- gmhmmp2

166.4 Module

You can load the modules by:

```
module load biocontainers
module load genemarks-2
```

166.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run genemarks-2 on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=genemarks-2
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers genemarks-2
```

CHAPTER

SEVEN

GENMAP

167.1 Introduction

GenMap: Ultra-fast Computation of Genome Mappability.

For more information, please check:
BioContainers: https://biocontainers.pro/tools/genmap
Home page: https://github.com/cpockrandt/genmap

167.2 Versions

- 1.3.0

167.3 Commands

- genmap

167.4 Module

You can load the modules by:

```
module load biocontainers
module load genmap
```

167.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run genmap on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=genmap
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers genmap

export TMPDIR=$PWD/tmp
genmap index -F ~/.local/share/genomes/hg38/hg38.fa  -I hg38_index
genmap map -K 64 -E 2 -I hg38_index -O map_output_hg38 -t -w -bg
```

CHAPTER

EIGHT

GENOMEPY

168.1 Introduction

`Genomepy` is designed to provide a simple and straightforward way to download and use genomic data.

For more information, please check its website: https://biocontainers.pro/tools/genomepy and its home page on Github.

168.2 Versions

- 0.12.0

168.3 Commands

- genomepy

168.4 Module

You can load the modules by:

```
module load biocontainers
module load genomepy
```

168.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Genomepy on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=genomepy
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers genomepy
```

CHAPTER

NINE

GENOMESCOPE2

169.1 Introduction

`Genomescope2`: Reference-free profiling of polyploid genomes.

For more information, please check its website: https://biocontainers.pro/tools/genomescope2 and its home page on Github.

169.2 Versions

- 2.0

169.3 Commands

- genomescope2

169.4 Module

You can load the modules by:

```
module load biocontainers
module load genomescope2
```

169.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Genomescope2 on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=genomescope2
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers genomescope2

wget https://raw.githubusercontent.com/schatzlab/genomescope/master/analysis/real_data/
↪ara_F1_21.hist

genomescope2 -i ara_F1_21.hist -o output -k 21
```

GENOMICCONSENSUS

170.1 Introduction

`Genomicconsensus` is the current PacBio consensus and variant calling suite.

For more information, please check its website: https://biocontainers.pro/tools/genomicconsensus and its home page on Github.

170.2 Versions

- 2.3.3

170.3 Commands

- quiver
- arrow
- variantCaller

170.4 Module

You can load the modules by:

```
module load biocontainers
module load genomicconsensus
```

170.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Genomicconsensus on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=genomicconsensus
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers genomicconsensus

quiver -j12 out.aligned_subreads.bam \
    -r All4mer.V2.01_Insert-changed.fa  \
    -o consensus.fasta -o consensus.fastq
```

CHAPTER

ONE

GENRICH

171.1 Introduction

`Genrich` is a peak-caller for genomic enrichment assays (e.g. ChIP-seq, ATAC-seq). It analyzes alignment files generated following the assay and produces a file detailing peaks of significant enrichment.

For more information, please check its website: https://biocontainers.pro/tools/genrich and its home page on Github.

171.2 Versions

- 0.6.1

171.3 Commands

- Genrich

171.4 Module

You can load the modules by:

```
module load biocontainers
module load genrich
```

171.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Genrich on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=genrich
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers genrich

Genrich  -t sample.bam  -o sample.narrowPeak  -v
```

CHAPTER

TWO

GFASTATS

172.1 Introduction

gfastats is a single fast and exhaustive tool for summary statistics and simultaneous *fa* (fasta, fastq, gfa [.gz]) genome assembly file manipulation. gfastats also allows seamless fasta<>fastq<>gfa[.gz] conversion. It has been tested in genomes even >100Gbp.

For more information, please check:
BioContainers: https://biocontainers.pro/tools/gfastats
Home page: https://github.com/vgl-hub/gfastats

172.2 Versions

- 1.2.3

172.3 Commands

- gfastats

172.4 Module

You can load the modules by:

```
module load biocontainers
module load gfastats
```

172.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run gfastats on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=gfastats
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers gfastats

gfastats input.fasta -o gfa
```

CHAPTER

THREE

GFFCOMPARE

173.1 Introduction

`Gffcompare` is used to compare, merge, annotate and estimate accuracy of one or more GFF files.

For more information, please check its website: https://biocontainers.pro/tools/gffcompare and its home page: https://ccb.jhu.edu/software/stringtie/gffcompare.shtml.

173.2 Versions

- 0.11.2

173.3 Commands

- gffcompare

173.4 Module

You can load the modules by:

```
module load biocontainers
module load gffcompare
```

173.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Gffcompare on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=gffcompare
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers gffcompare

gffcompare -r annotation.gff transcripts.gtf
```

CHAPTER

FOUR

GFFREAD

174.1 Introduction

`Gffread` is used to validate, filter, convert and perform various other operations on GFF files.

For more information, please check its website: https://biocontainers.pro/tools/gffread and its home page: http://ccb.jhu.edu/software/stringtie/gff.shtml.

174.2 Versions

- 0.12.7

174.3 Commands

- gffread

174.4 Module

You can load the modules by:

```
module load biocontainers
module load gffread
```

174.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Gffread on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=gffread
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers gffread

gffread -E annotation.gff -o ann_simple.gff

gffread annotation.gff -T -o annotation.gtf

gffread -w transcripts.fa -g genome.fa annotation.gff
```

CHAPTER

FIVE

GIMMEMOTIFS

175.1 Introduction

GimmeMotifs is a suite of motif tools, including a motif prediction pipeline for ChIP-seq experiments.

For more information, please check:
BioContainers: https://biocontainers.pro/tools/gimmemotifs
Home page: https://github.com/vanheeringen-lab/gimmemotifs

175.2 Versions

- 0.17.1

175.3 Commands

- gimme
-

175.4 Module

You can load the modules by:

```
module load biocontainers
module load gimmemotifs
```

175.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run gimmemotifs on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=gimmemotifs
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers gimmemotifs

gimme motifs ENCFF407IVS.bed ENCFF407IVS_motifs \
    -g ~/.local/share/genomes/hg38/hg38.fa --denovo
```

CHAPTER

SIX

GLIMMER

176.1 Introduction

`Glimmer` is a system for finding genes in microbial DNA, especially the genomes of bacteria, archaea, and viruses.

For more information, please check its website: https://biocontainers.pro/tools/glimmer and its home page: http://ccb.jhu.edu/software/glimmer/index.shtml.

176.2 Versions

- 3.02

176.3 Commands

- anomaly
- build-fixed
- build-icm
- entropy-profile
- entropy-score
- extract
- g3-from-scratch.csh
- g3-from-training.csh
- g3-iterated.csh
- get-motif-counts.awk
- glim-diff.awk
- glimmer3
- long-orfs
- match-list-col.awk
- multi-extract

- not-acgt.awk
- score-fixed
- start-codon-distrib
- test
- uncovered
- upstream-coords.awk
- window-acgt

176.4 Module

You can load the modules by:

```
module load biocontainers
module load glimmer
```

176.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Glimmer on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=glimmer
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers glimmer

long-orfs -n -t 1.15 scaffolds.fasta run1.longorfs
extract -t scaffolds.fasta run1.longorfs > run1.train
build-icm -r run1.icm < run1.train
glimmer3 scaffolds.fasta run1.icm cm
```

CHAPTER

SEVEN

GLIMMERHMM

177.1 Introduction

`Glimmerhmm` is a new gene finder based on a Generalized Hidden Markov Model (GHMM).

For more information, please check its website: https://biocontainers.pro/tools/glimmerhmm and its home page: https://ccb.jhu.edu/software/glimmerhmm/.

177.2 Versions

- 3.0.4

177.3 Commands

- glimmerhmm
- glimmhmm.pl
- trainGlimmerHMM

177.4 Module

You can load the modules by:

```
module load biocontainers
module load glimmerhmm
```

177.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Glimmerhmm on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=glimmerhmm
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers glimmerhmm

trainGlimmerHMM Asperg.fasta Asperg.cds -d Asperg
glimmerhmm Asperg.fasta -d Asperg -o Asperg_glimmerhmm_out
```

CHAPTER

EIGHT

GLNEXUS

178.1 Introduction

Glnexus: Scalable gVCF merging and joint variant calling for population sequencing projects.

For more information, please check:
BioContainers: https://biocontainers.pro/tools/glnexus
Home page: https://github.com/dnanexus-rnd/GLnexus

178.2 Versions

- 1.4.1

178.3 Commands

- glnexus_cli

178.4 Module

You can load the modules by:

```
module load biocontainers
module load glnexus
```

178.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run glnexus on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=glnexus
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers glnexus

glnexus_cli --config DeepVariant \
    --bed ALDH2.bed \
    dv_1000G_ALDH2_gvcf/*.g.vcf.gz \
    > dv_1000G_ALDH2.bcf
```

CHAPTER

NINE

GMAP

179.1 Introduction

`Gmap` is a genomic mapping and alignment program for mRNA and EST sequences.

For more information, please check its website: https://biocontainers.pro/tools/gmap and its home page: http://research-pub.gene.com/gmap/.

179.2 Versions

- 2021.05.27
- 2021.08.25

179.3 Commands

- atoiindex
- cmetindex
- cpuid
- dbsnp_iit
- ensembl_genes
- fa_coords
- get-genome
- gff3_genes
- gff3_introns
- gff3_splicesites
- gmap
- gmap.avx2
- gmap_build
- gmap_cat

- gmapindex
- gmapl
- gmapl.avx2
- gmapl.nosimd
- gmap.nosimd
- gmap_process
- gsnap
- gsnap.avx2
- gsnapl
- gsnapl.avx2
- gsnapl.nosimd
- gsnap.nosimd
- gtf_genes
- gtf_introns
- gtf_splicesites
- gtf_transcript_splicesites
- gvf_iit
- iit_dump
- iit_get
- iit_store
- indexdb_cat
- md_coords
- psl_genes
- psl_introns
- psl_splicesites
- sam_sort
- snpindex
- trindex
- vcf_iit

179.4 Module

You can load the modules by:

```
module load biocontainers
module load gmap
```

179.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Gmap on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 4
#SBATCH --job-name=gmap
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers gmap

gmap_build -d Cmm -D Cmm genome.fasta
gmap -d Cmm -t 4 -D ./Cmm  cdna.fasta > gmap_out.txt

gmap_build -d GRCh38 -D GRCh38 Homo_sapiens.GRCh38.dna.primary_assembly.fa
gsnap -d GRCh38 -D ./GRCh38 --nthreads=4  SRR16956239_1.fastq SRR16956239_2.fastq >␣
↪gsnap_out.txt
```

CHAPTER

GOATOOLS

180.1 Introduction

`Goatools` is a python library for gene ontology analyses. Detailed information about its usage can be found here: https://github.com/tanghaibao/goatools

180.2 Versions

- 1.1.12
- 1.2.3

180.3 Commands

- python
- python3
- compare_gos.py
- fetch_associations.py
- find_enrichment.py
- go_plot.py
- map_to_slim.py
- ncbi_gene_results_to_python.py
- plot_go_term.py
- prt_terms.py
- runxlrd.py
- vba_extract.py
- wr_hier.py
- wr_sections.py

180.4 Module

You can load the modules by:

```
module load biocontainers
module load goatools/1.1.12
```

180.5 Interactive job

To run goatools interactively on our clusters:

```
(base) UserID@bell-fe00:~ $ sinteractive -N1 -n12 -t4:00:00 -A myallocation
salloc: Granted job allocation 12345869
salloc: Waiting for resource configuration
salloc: Nodes bell-a008 are ready for job
(base) UserID@bell-a008:~ $ module load biocontainers goatools/1.1.12
(base) UserID@bell-a008:~ $ python
Python 3.8.10 (default, Nov 26 2021, 20:14:08)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from goatools.base import download_go_basic_obo
>>> obo_fname = download_go_basic_obo()
```

180.6 Batch job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To submit a sbatch job on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 10:00:00
#SBATCH -N 1
#SBATCH -n 24
#SBATCH --job-name=goatools
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers goatools/1.1.12

python script.py

find_enrichment.py --pval=0.05 --indent data/study data/population data/association

go_plot.py --go_file=tests/data/go_plot/go_heartjogging6.txt -r -o heartjogging6_r1.png
```

CHAPTER

ONE

GRAPHLAN

181.1 Introduction

`Graphlan` is a software tool for producing high-quality circular representations of taxonomic and phylogenetic trees.

For more information, please check its website: https://biocontainers.pro/tools/graphlan and its home page: https://huttenhower.sph.harvard.edu/graphlan/.

181.2 Versions

- 1.1.3

181.3 Commands

- graphlan.py
- graphlan_annotate.py

181.4 Module

You can load the modules by:

```
module load biocontainers
module load graphlan
```

181.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Graphlan on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=graphlan
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers graphlan

graphlan_annotate.py hmptree.xml hmptree.annot.xml --annot annot.txt

graphlan.py hmptree.annot.xml hmptree.png --dpi 150 --size 14
```

CHAPTER

TWO

GRAPHMAP

182.1 Introduction

`Graphmap` is a novel mapper targeted at aligning long, error-prone third-generation sequencing data.

For more information, please check its website: https://biocontainers.pro/tools/graphmap and its home page on Github.

182.2 Versions

- 0.6.3

182.3 Commands

- graphmap2

182.4 Module

You can load the modules by:

```
module load biocontainers
module load graphmap
```

182.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Graphmap on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=graphmap
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers graphmap
```

CHAPTER

THREE

GRIDSS

183.1 Introduction

`Gridss` is a module software suite containing tools useful for the detection of genomic rearrangements.

For more information, please check its | Docker hub: https://hub.docker.com/r/gridss/gridss and its home page on Github.

183.2 Versions

- 2.13.2

183.3 Commands

- R
- Rscript
- gridss
- gridss_annotate_vcf_kraken2
- gridss_annotate_vcf_repeatmasker
- gridss_extract_overlapping_fragments
- gridss_somatic_filter
- gridsstools
- virusbreakend
- virusbreakend-build

183.4 Module

You can load the modules by:

```
module load biocontainers
module load gridss
```

183.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Gridss on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=gridss
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers gridss
```

CHAPTER

FOUR

GSEAPY

184.1 Introduction

`Gseapy` is a python wrapper for GESA and Enrichr.

For more information, please check its website: https://biocontainers.pro/tools/gseapy and its home page: https://gseapy.readthedocs.io/en/latest/introduction.html.

184.2 Versions

- 0.10.8

184.3 Commands

- gseapy
- python
- python3

184.4 Module

You can load the modules by:

```
module load biocontainers
module load gseapy
```

184.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Gseapy on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=gseapy
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers gseapy

gseapy ssgsea -d ./data/testSet_rand1200.gct \
            -g data/temp.gmt \
            -o test/ssgsea_report2  \
            -p 4 --no-plot --no-scale
gseapy replot -i data -o test/replot_test
```

CHAPTER

FIVE

GTDB-TK

185.1 Introduction

GTDB-Tk is a software toolkit for assigning objective taxonomic classifications to bacterial and archaeal genomes based on the Genome Database Taxonomy GTDB. It is designed to work with recent advances that allow hundreds or thousands of metagenome-assembled genomes (MAGs) to be obtained directly from environmental samples. It can also be applied to isolate and single-cell genomes.

GTDB-Tk reference data (R202) has been downloaded for users.

185.2 Versions

- 1.7.0
- 2.1.0

185.3 Commands

- gtdbtk

185.4 Module

module load biocontainers module load gtdbtk/1.7.0

185.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run GTDB-Tk our our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 20:00:00
#SBATCH -N 1
#SBATCH -n 24
#SBATCH --job-name=gtdbtk
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers gtdbtk/1.7.0

gtdbtk identify --genome_dir genomes --out_dir identify --extension gz --cpus 8
gtdbtk align --identify_dir identify --out_dir align --cpus 8
gtdbtk classify --genome_dir genomes --align_dir align --out_dir classify --extension gz␣
↪--cpus 8
```

CHAPTER

SIX

GUBBINS

186.1 Introduction

`Gubbins` is an algorithm that iteratively identifies loci containing elevated densities of base substitutions while concurrently constructing a phylogeny based on the putative point mutations outside of these regions.

For more information, please check its website: https://biocontainers.pro/tools/gubbins and its home page on Github.

186.2 Versions

- 3.2.0-py39

186.3 Commands

- extract_gubbins_clade.py
- generate_ska_alignment.py
- gubbins_alignment_checker.py
- mask_gubbins_aln.py
- run_gubbins.py
- sumlabels.py
- sumtrees.py

186.4 Module

You can load the modules by:

```
module load biocontainers
module load gubbins
```

186.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Gubbins on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=gubbins
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers gubbins

run_gubbins.py --prefix ST239 ST239.aln
```

CHAPTER

SEVEN

GUPPY

187.1 Introduction

Guppy is a data processing toolkit that contains the Oxford Nanopore Technologies' basecalling algorithms, and several bioinformatic post-processing features.

For more information, please check its | Docker hub: https://hub.docker.com/r/genomicpariscentre/guppy and its home page: https://community.nanoporetech.com.

187.2 Versions

- 6.0.1

187.3 Commands

- guppy_aligner
- guppy_barcoder
- guppy_basecall_server
- guppy_basecaller
- guppy_basecaller_duplex
- guppy_basecaller_supervisor
- guppy_basecall_client

187.4 Module

You can load the modules by:

```
module load biocontainers
module load guppy
```

187.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Guppy on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 12
#SBATCH --job-name=guppy
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers guppy

guppy_basecaller --compress_fastq -i data/fast5_tiny/ \
    -s basecall_tiny/ --cpu_threads_per_caller 12 \
    --num_callers 1 -c dna_r9.4.1_450bps_hac.cfg
```

CHAPTER

EIGHT

HAIL

188.1 Introduction

Hail is an open-source, general-purpose, Python-based data analysis tool with additional data types and methods for working with genomic data.

For more information, please check:
Docker hub: https://hub.docker.com/r/hailgenetics/hail
Home page: https://github.com/hail-is/hail

188.2 Versions

- 0.2.94
- 0.2.98

188.3 Commands

- python3

188.4 Module

You can load the modules by:

```
module load biocontainers
module load hail
```

188.5 Interactive job

To run Hail interactively on our clusters:

```
(base) UserID@bell-fe00:~ $ sinteractive -N1 -n12 -t4:00:00 -A myallocation
salloc: Granted job allocation 12345869
salloc: Waiting for resource configuration
salloc: Nodes bell-a008 are ready for job
(base) UserID@bell-a008:~ $ module load biocontainers hail
(base) UserID@bell-a008:~ $ python3
Python 3.7.13 (default, Apr 24 2022, 01:05:22)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import hail as hl
>>>  print(hl.citation())
Hail Team. Hail 0.2.94-f0b38d6c436f. https://github.com/hail-is/hail/commit/f0b38d6c436f.
```

188.6 Batch job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run hail on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=hail
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers hail
python3 script.py
```

CHAPTER

NINE

HAP.PY

189.1 Introduction

Hap.py is a tool to compare diploid genotypes at haplotype level.

For more information, please check:
Docker hub: https://hub.docker.com/r/pkrusche/hap.py
Home page: https://github.com/Illumina/hap.py

189.2 Versions

- 0.3.9

189.3 Commands

- bamstats.py
- cnx.py
- ftx.py
- guess-ploidy.py
- hap.py
- ovc.py
- plot-roh.py
- pre.py
- qfy.py
- som.py
- varfilter.py

189.4 Module

You can load the modules by:

```
module load biocontainers
module load hap.py
```

189.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run hap.py on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=hap.py
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers hap.py

hap.py  \
  example/happy/PG_NA12878_chr21.vcf.gz \
  example/happy/NA12878_chr21.vcf.gz \
  -f example/happy/PG_Conf_chr21.bed.gz \
  -r example/chr21.fa \
  -o test
```

CHAPTER

HELEN

190.1 Introduction

HELEN is a multi-task RNN polisher which operates on images produced by MarginPolish.

For more information, please check:
Docker hub: https://hub.docker.com/r/kishwars/helen
Home page: https://github.com/kishwarshafin/helen

190.2 Versions

- 1.0

190.3 Commands

- helen

190.4 Module

You can load the modules by:

```
module load biocontainers
module load helen
```

190.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run helen on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 32
#SBATCH --job-name=helen
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers helen

helen polish \
    --image_dir mp_output \
    --model_path "helen_modles/HELEN_r941_guppy344_microbial.pkl" \
    --threads 32 \
    --output_dir "helen_output/" \
    --output_prefix Staph_Aur_draft_helen
```

CHAPTER

ONE

HICEXPLORER

191.1 Introduction

`Hicexplorer` is a set of tools to process, normalize and visualize Hi-C data.

For more information, please check its website: https://biocontainers.pro/tools/hicexplorer and its home page: https://hicexplorer.readthedocs.io/en/latest/#.

191.2 Versions

- 3.7.2

191.3 Commands

- chicAggregateStatistic
- chicDifferentialTest
- chicExportData
- chicPlotViewpoint
- chicQualityControl
- chicSignificantInteractions
- chicViewpoint
- chicViewpointBackgroundModel
- hicAdjustMatrix
- hicAggregateContacts
- hicAverageRegions
- hicBuildMatrix
- hicCompareMatrices
- hicCompartmentalization
- hicConvertFormat

- hicCorrectMatrix
- hicCorrelate
- hicCreateThresholdFile
- hicDetectLoops
- hicDifferentialTAD
- hicexplorer
- hicFindEnrichedContacts
- hicFindRestSite
- hicFindTADs
- hicHyperoptDetectLoops
- hicHyperoptDetectLoopsHiCCUPS
- hicInfo
- hicInterIntraTAD
- hicMergeDomains
- hicMergeLoops
- hicMergeMatrixBins
- hicMergeTADbins
- hicNormalize
- hicPCA
- hicPlotAverageRegions
- hicPlotDistVsCounts
- hicPlotMatrix
- hicPlotSVL
- hicPlotTADs
- hicPlotViewpoint
- hicQC
- hicQuickQC
- hicSumMatrices
- hicTADClassifier
- hicTrainTADClassifier
- hicTransform
- hicValidateLocations

191.4 Module

You can load the modules by:

```
module load biocontainers
module load hicexplorer
```

191.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Hicexplorer on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=hicexplorer
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers hicexplorer
```

CHAPTER

TWO

HIFIASM

192.1 Introduction

`Hifiasm` is a fast haplotype-resolved de novo assembler for PacBio HiFi reads.

For more information, please check its website: https://biocontainers.pro/tools/hifiasm and its home page on Github.

192.2 Versions

- 0.16.0

192.3 Commands

- hifiasm

192.4 Module

You can load the modules by:

```
module load biocontainers
module load hifiasm
```

192.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Hifiasm on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=hifiasm
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers hifiasm
```

CHAPTER

THREE

HISAT2

193.1 Introduction

`HISAT2` is a fast and sensitive alignment program for mapping next-generation sequencing reads (both DNA and RNA) to a population of human genomes as well as to a single reference genome.

For more information, please check its website: https://biocontainers.pro/tools/hisat2 and its home page on Github.

193.2 Versions

- 2.2.1

193.3 Commands

- extract_exons.py
- extract_splice_sites.py
- hisat2
- hisat2-align-l
- hisat2-align-s
- hisat2-build
- hisat2-build-l
- hisat2-build-s
- hisat2-inspect
- hisat2-inspect-l
- hisat2-inspect-s
- hisat2_extract_exons.py
- hisat2_extract_snps_haplotypes_UCSC.py
- hisat2_extract_snps_haplotypes_VCF.py
- hisat2_extract_splice_sites.py

- hisat2_read_statistics.py
- hisat2_simulate_reads.py

193.4 Module

You can load the modules by:

```
module load biocontainers
module load hisat2
```

193.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run HISAT2 on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=hisat2
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers hisat2

hisat2-build genome.fa genome

# for single-end FASTA reads DNA alignment
hisat2 -f -x genome -U reads.fa -S output.sam --no-spliced-alignment

# for paired-end FASTQ reads alignment
hisat2 -x genome -1 reads_1.fq -2 read2_2.fq -S output.sam
```

CHAPTER

FOUR

HMMER

194.1 Introduction

`Hmmer` is used for searching sequence databases for sequence homologs, and for making sequence alignments.

For more information, please check its website: https://biocontainers.pro/tools/hmmer and its home page: http://hmmer.org.

194.2 Versions

- 3.3.2

194.3 Commands

- alimask
- easel
- esl-afetch
- esl-alimanip
- esl-alimap
- esl-alimask
- esl-alimerge
- esl-alipid
- esl-alirev
- esl-alistat
- esl-compalign
- esl-compstruct
- esl-construct
- esl-histplot
- esl-mask

- esl-mixdchlet
- esl-reformat
- esl-selectn
- esl-seqrange
- esl-seqstat
- esl-sfetch
- esl-shuffle
- esl-ssdraw
- esl-translate
- esl-weight
- hmmalign
- hmmbuild
- hmmconvert
- hmmemit
- hmmfetch
- hmmlogo
- hmmpgmd
- hmmpgmd_shard
- hmmpress
- hmmscan
- hmmsearch
- hmmsim
- hmmstat
- jackhmmer
- makehmmerdb
- nhmmer
- nhmmscan
- phmmer

194.4 Module

You can load the modules by:

```
module load biocontainers
module load hmmer
```

194.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Hmmer on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=hmmer
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers hmmer

hmmsearch Nramp.hmm protein.fa > out
```

CHAPTER

FIVE

HOMMER

195.1 Introduction

HOMMER (Hypergeometric Optimization of Motif EnRichment) is a suite of tools for Motif Discovery and next-gen sequencing analysis. Details about its usage can be found in HOMMER website.

195.2 Versions

- 4.11

195.3 Commands

- addDataHeader.pl
- addData.pl
- addGeneAnnotation.pl
- addInternalData.pl
- addOligos.pl
- adjustPeakFile.pl
- adjustRedunGroupFile.pl
- analyzeChIP-Seq.pl
- analyzeRepeats.pl
- analyzeRNA.pl
- annotateInteractions.pl
- annotatePeaks.pl
- annotateRelativePosition.pl
- annotateTranscripts.pl
- assignGeneWeights.pl
- assignTSStoGene.pl
- batchAnnotatePeaksHistogram.pl

- batchFindMotifsGenome.pl
- batchFindMotifs.pl
- batchMakeHiCMatrix.pl
- batchMakeMultiWigHub.pl
- batchMakeTagDirectory.pl
- batchParallel.pl
- bed2DtoUCSCbed.pl
- bed2pos.pl
- bed2tag.pl
- blat2gtf.pl
- bridgeResult2Cytoscape.pl
- changeNewLine.pl
- checkPeakFile.pl
- checkTagBias.pl
- chopify.pl
- chopUpBackground.pl
- chopUpPeakFile.pl
- cleanUpPeakFile.pl
- cleanUpSequences.pl
- cluster2bedgraph.pl
- cluster2bed.pl
- combineGO.pl
- combineHubs.pl
- compareMotifs.pl
- condenseBedGraph.pl
- cons2fasta.pl
- conservationAverage.pl
- conservationPerLocus.pl
- convertCoordinates.pl
- convertIDs.pl
- convertOrganismID.pl
- duplicateCol.pl
- eland2tags.pl
- fasta2tab.pl
- fastq2fasta.pl
- filterListBy.pl

- filterTADsAndCPs.pl
- filterTADsAndLoops.pl
- findcsRNATSS.pl
- findGO.pl
- findGOtxt.pl
- findHiCCompartments.pl
- findHiCDomains.pl
- findHiCInteractionsByChr.pl
- findKnownMotifs.pl
- findMotifsGenome.pl
- findMotifs.pl
- findRedundantBLAT.pl
- findTADsAndLoops.pl
- findTopMotifs.pl
- flipPC1toMatch.pl
- freq2group.pl
- genericConvertIDs.pl
- GenomeOntology.pl
- getChrLengths.pl
- getConservedRegions.pl
- getDifferentialBedGraph.pl
- getDifferentialPeaksReplicates.pl
- getDiffExpression.pl
- getDistalPeaks.pl
- getFocalPeaks.pl
- getGenesInCategory.pl
- getGWASoverlap.pl
- getHiCcorrDiff.pl
- getHomerQCstats.pl
- getLikelyAdapters.pl
- getMappingStats.pl
- getPartOfPromoter.pl
- getPos.pl
- getRandomReads.pl
- getSiteConservation.pl
- getTopPeaks.pl

- gff2pos.pl
- go2cytoscape.pl
- groupSequences.pl
- joinFiles.pl
- loadGenome.pl
- loadPromoters.pl
- makeBigBedMotifTrack.pl
- makeBigWig.pl
- makeBinaryFile.pl
- makeHiCWashUfile.pl
- makeMetaGeneProfile.pl
- makeMultiWigHub.pl
- map-fastq.pl
- merge2Dbed.pl
- mergeData.pl
- motif2Jaspar.pl
- motif2Logo.pl
- parseGTF.pl
- pos2bed.pl
- preparseGenome.pl
- prepForR.pl
- profile2seq.pl
- qseq2fastq.pl
- randomizeGroupFile.pl
- randomizeMotifs.pl
- randRemoveBackground.pl
- removeAccVersion.pl
- removeBadSeq.pl
- removeOutOfBoundsReads.pl
- removePoorSeq.pl
- removeRedundantPeaks.pl
- renamePeaks.pl
- resizePosFile.pl
- revoppMotif.pl
- rotateHiCmatrix.pl
- runHiCpca.pl

- sam2spliceJunc.pl
- scanMotifGenomeWide.pl
- scrambleFasta.pl
- selectRepeatBg.pl
- seq2profile.pl
- SIMA.pl
- subtractBedGraphsDirectory.pl
- subtractBedGraphs.pl
- tab2fasta.pl
- tag2bed.pl
- tag2pos.pl
- tagDir2bed.pl
- tagDir2hicFile.pl
- tagDir2HiCsummary.pl
- zipHomerResults.pl

195.4 Database

Selected database have been downloaded for users.

- `ORGANISMS`: yeast, worm, mouse, arabidopsis, zebrafish, rat, human and fly
- `PROMOTERS`: yeast, worm, mouse, arabidopsis, zebrafish, rat, human and fly
- `GENOMES`: hg19, hg38, mm10, ce11, dm6, rn6, danRer11, tair10, and sacCer3

195.5 Module

You can load the modules by:

```
module load biocontainers
module load hommer/4.11
```

195.6 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run HOMMER on our cluster:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 10:00:00
#SBATCH -N 1
#SBATCH -n 24
#SBATCH --job-name=hommer
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers hommer/4.11

configureHomer.pl -list   ## Check the installed database.
findMotifs.pl mouse_geneid.txt mouse motif_out_mouse
findMotifs.pl geneid.txt human motif_out
```

CHAPTER

SIX

HOW_ARE_WE_STRANDED_HERE

196.1 Introduction

`How_are_we_stranded_here` is a python package for testing strandedness of RNA-Seq fastq files.

For more information, please check its website: https://biocontainers.pro/tools/how_are_we_stranded_here and its home page on Github.

196.2 Versions

- 1.0.1

196.3 Commands

- check_strandedness

196.4 Module

You can load the modules by:

```
module load biocontainers
module load how_are_we_stranded_here
```

196.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run How_are_we_stranded_here on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=how_are_we_stranded_here
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers how_are_we_stranded_here

check_strandedness --gtf Homo_sapiens.GRCh38.105.gtf \
    --transcripts Homo_sapiens.GRCh38.cds.all.fa \
    --reads_1 seq_1.fastq  --reads_2 seq_2.fastq
```

CHAPTER

SEVEN

HTSEQ

197.1 Introduction

HTSeq is a Python library to facilitate processing and analysis of data from high-throughput sequencing (HTS) experiments.

For more information, please check its website: https://biocontainers.pro/tools/htseq and its home page on Github.

197.2 Versions

- 0.13.5-py36
- 0.13.5-py37
- 0.13.5-py38
- 1.99.2-py37
- 2.0.1-py37

197.3 Commands

- htseq-count
- htseq-count-barcodes
- htseq-qa
- python
- python3

197.4 Module

You can load the modules by:

```
module load biocontainers
module load htseq
```

197.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run HTSeq on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=htseq
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers htseq

python -m HTSeq.scripts.count \
       -f bam input.bam ref.gtf \
       > test.out
```

CHAPTER

EIGHT

HTSLIB

198.1 Introduction

`Htslib` is a C library for high-throughput sequencing data formats.

For more information, please check its website: https://biocontainers.pro/tools/htslib and its home page on Github.

198.2 Versions

- 1.14
- 1.15
- 1.16

198.3 Commands

- bgzip
- htsfile
- tabix

198.4 Module

You can load the modules by:

```
module load biocontainers
module load htslib
```

198.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Htslib on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=htslib
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers htslib

tabix sorted.gff.gz chr1:10,000,000-20,000,000
```

CHAPTER

NINE

HTSTREAM

199.1 Introduction

`Htstream` is a quality control and processing pipeline for High Throughput Sequencing data.

For more information, please check its website: https://biocontainers.pro/tools/htstream and its home page on Github.

199.2 Versions

- 1.3.3

199.3 Commands

- hts_AdapterTrimmer
- hts_CutTrim
- hts_LengthFilter
- hts_NTrimmer
- hts_Overlapper
- hts_PolyATTrim
- hts_Primers
- hts_QWindowTrim
- hts_SeqScreener
- hts_Stats
- hts_SuperDeduper

199.4 Module

You can load the modules by:

```
module load biocontainers
module load htstream
```

199.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Htstream on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=htstream
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers htstream
```

HUMANN 3

200.1 Introduction

`HUMAnN 3.0` is the next iteration of HUMAnN, the HMP Unified Metabolic Analysis Network. HUMAnN is a method for efficiently and accurately profiling the abundance of microbial metabolic pathways and other molecular functions from metagenomic or metatranscriptomic sequencing data.

For more information please check its website: https://huttenhower.sph.harvard.edu/humann/

200.2 Versions

- 3.0.0

200.3 Commands

- humann
- humann3
- humann3_databases
- humann_barplot
- humann_benchmark
- humann_build_custom_database
- humann_config
- humann_databases
- humann_genefamilies_genus_level
- humann_infer_taxonomy
- humann_join_tables
- humann_reduce_table
- humann_regroup_table
- humann_rename_table
- humann_renorm_table
- humann_split_stratified_table

- humann_split_table
- humann_test
- humann_unpack_pathways

200.4 Database

Full ChocoPhlAn, UniRef90, EC-filtered UniRef90, UniRef50, EC-filtered UniRef50, and utility_mapping databases have been downloaded for users.

200.5 Module

You can load the modules by:

```
module load biocontainers
module load humann/3.0.0
```

200.6 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run HUMAnN3 on our cluster:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 10:00:00
#SBATCH -N 1
#SBATCH -n 24
#SBATCH --job-name=humann
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers humann/3.0.0
# Check the database and config by:
humann_config --print

humann --threads 24 --input examples/demo.fastq --output demo_output --metaphlan-options
↪"--bowtie2db /depot/itap/datasets/metaphlan"
```

CHAPTER

ONE

HYPHY

201.1 Introduction

Hyphy is an open-source software package for the analysis of genetic sequences using techniques in phylogenetics, molecular evolution, and machine learning.

For more information, please check its website: https://biocontainers.pro/tools/hyphy and its home page on Github.

201.2 Versions

- 2.5.36

201.3 Commands

- hyphy

201.4 Module

You can load the modules by:

```
module load biocontainers
module load hyphy
```

201.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Hyphy on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=hyphy
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers hyphy
```

CHAPTER

TWO

IDBA

202.1 Introduction

`Idba` is a practical iterative De Bruijn Graph De Novo Assembler for sequence assembly in bioinfomatics.

For more information, please check its website: https://biocontainers.pro/tools/idba and its home page: https://i.cs.hku.hk/~alse/hkubrg/projects/idba/index.html.

202.2 Versions

- 1.1.3

202.3 Commands

- fa2fq
- filter_blat
- filter_contigs
- filterfa
- fq2fa
- idba
- idba_hybrid
- idba_tran
- idba_tran_test
- idba_ud
- parallel_blat
- parallel_rna_blat
- print_graph
- raw_n50
- run-unittest.py

- sample_reads
- scaffold
- scan.py
- shuffle_reads
- sim_reads
- sim_reads_tran
- sort_psl
- sort_reads
- split_fa
- split_fq
- split_scaffold
- test
- validate_blat
- validate_blat_parallel
- validate_component
- validate_contigs_blat
- validate_contigs_mummer
- validate_reads_blat
- validate_rna

202.4 Module

You can load the modules by:

```
module load biocontainers
module load idba
```

202.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Idba on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
```

(continues on next page)

(continued from previous page)

```
#SBATCH --job-name=idba
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers idba

fq2fa --paired --filter SRR1977249.abundtrim.subset.pe.fq SRR1977249.abundtrim.subset.pe.
↪fa
idba_ud  -r SRR1977249.abundtrim.subset.pe.fa -o output
```

CHAPTER

THREE

IGV

203.1 Introduction

`IGV` (Integrative Genomics Viewer) is a high-performance, easy-to-use, interactive tool for the visual exploration of genomic data.

For more information, please check its home page: http://www.broadinstitute.org/software/igv/home.

203.2 Versions

- 2.11.9
- 2.12.3

203.3 Commands

- igv_hidpi.sh
- igv.sh

203.4 Module

You can load the modules by:

```
module load biocontainers
module load igv
```

203.5 Interactive job

Since IGV requires GUI, it is recommended to run it within ThinLinc:

```
(base) UserID@bell-fe00:~ $ sinteractive -N1 -n12 -t4:00:00 -A myallocation
 salloc: Granted job allocation 12345869
 salloc: Waiting for resource configuration
 salloc: Nodes bell-a008 are ready for job
(base) UserID@bell-a008:~ $ module --force purge
(base) UserID@bell-a008:~ $ ml biocontainers igv
(base) UserID@bell-a008:~ $ igv.sh
```

CHAPTER

FOUR

IMPUTE2

204.1 Introduction

`Impute2` is a genotype imputation and haplotype phasing program.

For more information, please check its website: https://biocontainers.pro/tools/impute2 and its home page: https://mathgen.stats.ox.ac.uk/impute/impute_v2.html#home.

204.2 Versions

- 2.3.2

204.3 Commands

- impute2

204.4 Module

You can load the modules by:

```
module load biocontainers
module load impute2
```

204.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Impute2 on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=impute2
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers impute2

impute2 \
    -m Example/example.chr22.map \
    -h Example/example.chr22.1kG.haps \
    -l Example/example.chr22.1kG.legend \
    -g Example/example.chr22.study.gens \
    -strand_g Example/example.chr22.study.strand \
    -int 20.4e6 20.5e6 \
    -Ne 20000 \
    -o example.chr22.one.phased.impute2
```

CHAPTER

FIVE

INSTRAIN

205.1 Introduction

`Instrain` is a python program for analysis of co-occurring genome populations from metagenomes that allows highly accurate genome comparisons, analysis of coverage, microdiversity, and linkage, and sensitive SNP detection with gene localization and synonymous non-synonymous identification.

For more information, please check its website: https://biocontainers.pro/tools/instrain and its home page on Github.

205.2 Versions

- 1.5.7
- 1.6.3

205.3 Commands

- inStrain

205.4 Module

You can load the modules by:

```
module load biocontainers
module load instrain
```

205.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Instrain on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=instrain
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers instrain
```

CHAPTER

SIX

INTARNA

206.1 Introduction

`Intarna` is a general and fast approach to the prediction of RNA-RNA interactions incorporating both the accessibility of interacting sites as well as the existence of a user-definable seed interaction.

For more information, please check its website: https://biocontainers.pro/tools/intarna and its home page on Github.

206.2 Versions

- 3.3.1

206.3 Commands

- IntaRNA

206.4 Module

You can load the modules by:

```
module load biocontainers
module load intarna
```

206.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Intarna on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=intarna
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers intarna

IntaRNA -t CCCCCCCCGGGGGGGGGGGGG -q AAAACCCCCCCUUUU
```

CHAPTER

SEVEN

INTERPROSCAN

207.1 Introduction

`InterPro` is a database which integrates together predictive information about proteins' function from a number of partner resources, giving an overview of the families that a protein belongs to and the domains and sites it contains.

Users who have novel nucleotide or protein sequences that they wish to functionally characterise can use the software package `InterProScan` to run the scanning algorithms from the InterPro database in an integrated way. Sequences are submitted in FASTA format. Matches are then calculated against all of the required member database's signatures and the results are then output in a variety of formats.

207.2 Versions

- 5.54_87.0

207.3 Commands

interproscan.sh

207.4 Database

Latest version of database has been downloaded and setup in **/depot/itap/datasets/interproscan-5.54-87.0/data**.

207.5 Module

You can load the modules by:

```
module load biocontainers
module load interproscan/5.54_87.0
```

207.6 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run run_dbcan on our cluster:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 10:00:00
#SBATCH -N 1
#SBATCH -n 24
#SBATCH --job-name=interproscan
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers interproscan/5.54_87.0

interproscan.sh -cpu 24 -i test_proteins.fasta
interproscan.sh -cpu 24 -t n -i test_nt_seqs.fasta
```

CHAPTER

EIGHT

IQ-TREE

208.1 Introduction

IQ-TREE is an efficient phylogenomic software by maximum likelihood.

For more information, please check its website: https://biocontainers.pro/tools/iqtree and its home page: http://www.iqtree.org.

208.2 Versions

- 1.6.12
- 2.1.2
- 2.2.0_beta

208.3 Commands

- iqtree

208.4 Module

You can load the modules by:

```
module load biocontainers
module load iqtree
```

208.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run IQ-TREE on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=iqtree
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers iqtree

iqtree -s input.phy -m GTR+I+G > test.out
```

CHAPTER

NINE

ISOSEQ3

209.1 Introduction

`Isoseq3` - Scalable De Novo Isoform Discovery.

For more information, please check its website: https://biocontainers.pro/tools/isoseq3 and its home page on Github.

209.2 Versions

- 3.4.0
- 3.7.0

209.3 Commands

- isoseq3

209.4 Module

You can load the modules by:

```
module load biocontainers
module load isoseq3
```

209.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Isoseq3 on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=isoseq3
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers isoseq3

isoseq3 --version

isoseq3 refine --require-polya \
    alz.demult.5p--3p.bam \
    primers.fasta alz.flnc.bam

isoseq3 cluster alz.flnc.bam \
    alz.polished.bam --verbose --use-qvs
```

CHAPTER

IVAR

210.1 Introduction

Ivar is a computational package that contains functions broadly useful for viral amplicon-based sequencing.

For more information, please check:
Docker hub: https://hub.docker.com/r/andersenlabapps/ivar/
Home page: https://github.com/andersen-lab/ivar

210.2 Versions

- 1.3.1

210.3 Commands

- ivar

210.4 Module

You can load the modules by:

```
module load biocontainers
module load ivar
```

210.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run ivar on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=ivar
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers ivar
```

CHAPTER

ONE

JCVI

211.1 Introduction

Jcvi is a collection of Python libraries to parse bioinformatics files, or perform computation related to assembly, annotation, and comparative genomics.

For more information, please check:
Home page: https://github.com/tanghaibao/jcvi

211.2 Versions

- 1.2.7-py39

211.3 Commands

- python
- python3

211.4 Module

You can load the modules by:

```
module load biocontainers
module load jcvi
```

211.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run jcvi on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=jcvi
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers jcvi

python -m jcvi.formats.fasta format Vvinifera_145_Genoscope.12X.cds.fa.gz grape.cds
python -m jcvi.formats.fasta format Ppersica_298_v2.1.cds.fa.gz peach.cds
python -m jcvi.formats.gff bed --type=mRNA --key=Name --primary_only Vvinifera_145_
↪Genoscope.12X.gene.gff3.gz -o grape.bed
python -m jcvi.compara.catalog ortholog grape peach --no_strip_names
python -m jcvi.graphics.dotplot grape.peach.anchors
rm grape.peach.last.filtered
python -m jcvi.compara.catalog ortholog grape peach --cscore=.99 --no_strip_names
python -m jcvi.graphics.dotplot grape.peach.anchors
python -m jcvi.compara.synteny depth --histogram grape.peach.anchors
python -m jcvi.graphics.grabseeds seeds test-data/test.JPG
```

CHAPTER

TWO

KAIJU

212.1 Introduction

`Kaiju` is a tool for fast taxonomic classification of metagenomic sequencing reads using a protein reference database.

For more information, please check its website: https://biocontainers.pro/tools/kaiju and its home page on Github.

212.2 Versions

- 1.8.2

212.3 Commands

- kaiju
- kaiju-addTaxonNames
- kaiju-convertMAR.py
- kaiju-convertNR
- kaiju-excluded-accessions.txt
- kaiju-gbk2faa.pl
- kaiju-makedb
- kaiju-mergeOutputs
- kaiju-mkbwt
- kaiju-mkfmi
- kaiju-multi
- kaiju-taxonlistEuk.tsv
- kaiju2krona
- kaiju2table
- kaijup
- kaijux

212.4 Module

You can load the modules by:

```
module load biocontainers
module load kaiju
```

212.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Kaiju on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 24
#SBATCH --job-name=kaiju
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers kaiju

kaiju -t kaijudb/nodes.dmp \
     -f kaijudb/refseq/kaiju_db_refseq.fmi \
    -i input_1.fastq -j input_2.fastq
     -z 24
```

CHAPTER

THREE

KALLISTO

213.1 Introduction

`Kallisto` is a program for quantifying abundances of transcripts from RNA-Seq data, or more generally of target sequences using high-throughput sequencing reads. It is based on the novel idea of pseudoalignment for rapidly determining the compatibility of reads with targets, without the need for alignment.

Detailed usage can be found here: https://github.com/pachterlab/kallisto

213.2 Versions

- 0.46.2
- 0.48.0

213.3 Commands

- kallisto

213.4 Module

You can load the modules by:

```
module load biocontainers
module load kallisto/0.48.0
```

213.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run kallisto on our our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 10:00:00
#SBATCH -N 1
#SBATCH -n 24
#SBATCH --job-name=kallisto
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers kallisto/0.48.0

kallisto index -i transcripts.idx Homo_sapiens.GRCh38.cds.all.fa.gz
kallisto quant -t 24 -i transcripts.idx -o output -b 100  SRR11614709_1.fastq ␣
↪SRR11614709_2.fastq
```

CHAPTER

FOUR

KHMER

214.1 Introduction

`Khmer` is a tool for k-mer counting, filtering, and graph traversal FTW!

For more information, please check its website: https://biocontainers.pro/tools/khmer and its home page on Github.

214.2 Versions

- 3.0.0a3-py36

214.3 Commands

- abundance-dist.py
- abundance-dist-single.py
- annotate-partitions.py
- count-median.py
- cygdb
- cython
- cythonize
- do-partition.py
- extract-long-sequences.py
- extract-paired-reads.py
- extract-partitions.py
- fastq-to-fasta.py
- filter-abund.py
- filter-abund-single.py
- filter-stoptags.py
- find-knots.py

- interleave-reads.py
- load-graph.py
- load-into-counting.py
- make-initial-stoptags.py
- merge-partitions.py
- normalize-by-median.py
- partition-graph.py
- readstats.py
- sample-reads-randomly.py
- screed
- split-paired-reads.py
- trim-low-abund.py
- unique-kmers.py

214.4 Module

You can load the modules by:

```
module load biocontainers
module load khmer
```

214.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Khmer on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=khmer
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers khmer
```

CHAPTER

FIVE

KMA

215.1 Introduction

KMA is a mapping method designed to map raw reads directly against redundant databases, in an ultra-fast manner using seed and extend.

For more information, please check:
BioContainers: https://biocontainers.pro/tools/kma
Home page: https://bitbucket.org/genomicepidemiology/kma/src/master/

215.2 Versions

- 1.4.3

215.3 Commands

- kma
- kma_index
- kma_shm
- kma_update

215.4 Module

You can load the modules by:

```
module load biocontainers
module load kma
```

215.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run kma on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=kma
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers kma
```

CHAPTER

SIX

KMC

216.1 Introduction

Kmc is a tool for efficient k-mer counting and filtering of reads based on k-mer content.

For more information, please check its website: https://biocontainers.pro/tools/kmc and its home page on Github.

216.2 Versions

- 3.2.1

216.3 Commands

- kmc
- kmc_dump
- kmc_tools

216.4 Module

You can load the modules by:

```
module load biocontainers
module load kmc
```

216.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Kmc on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=kmc
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers kmc

kmc -k27 seq.fastq 27mers .
```

CHAPTER

SEVEN

JELLYFISH

217.1 Introduction

`Jellyfish` is a tool for fast, memory-efficient counting of k-mers in DNA. A k-mer is a substring of length k, and counting the occurrences of all such substrings is a central step in many analyses of DNA sequence.

For more information, please check its website: https://biocontainers.pro/tools/kmer-jellyfish and its home page: http://www.genome.umd.edu/jellyfish.html.

217.2 Versions

- 2.3.0

217.3 Commands

- jellyfish

217.4 Module

You can load the modules by:

```
module load biocontainers
module load kmer-jellyfish
```

217.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Jellyfish on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 12
#SBATCH --job-name=kmer-jellyfish
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers kmer-jellyfish

jellyfish count -m 16 -s 100M -t 12 \
     -o mer_counts -c 7  input.fastq
```

CHAPTER

EIGHT

KNEADDATA

218.1 Introduction

`KneadData` is a tool designed to perform quality control on metagenomic and metatranscriptomic sequencing data, especially data from microbiome experiments. In these experiments, samples are typically taken from a host in hopes of learning something about the microbial community on the host.

Detailed usage can be found here: https://huttenhower.sph.harvard.edu/kneaddata/

218.2 Versions

- 0.10.0

218.3 Commands

- kneaddata
- kneaddata_bowtie2_discordant_pairs
- kneaddata_build_database
- kneaddata_database
- kneaddata_read_count_table
- kneaddata_test
- kneaddata_trf_parallel

218.4 Module

You can load the modules by:

```
module load biocontainers
module load kneaddata
```

218.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run kneaddata on our our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 20:00:00
#SBATCH -N 1
#SBATCH -n 24
#SBATCH --job-name=kneaddata
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers kneaddata

kneaddata --input examples/demo.fastq --reference-db examples/demo_db --output kneaddata_
↪demo_outpu --threads 24 --processes 24
```

CHAPTER

NINE

KOVER

219.1 Introduction

Kover is an out-of-core implementation of rule-based machine learning algorithms that has been tailored for genomic biomarker discovery.

For more information, please check:
Docker hub: https://hub.docker.com/r/aldro61/kover
Home page: https://github.com/aldro61/kover

219.2 Versions

- 2.0.6

219.3 Commands

- kover

219.4 Module

You can load the modules by:

```
module load biocontainers
module load kover
```

219.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run kover on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=kover
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers kover
```

KRAKEN2

220.1 Introduction

Kraken2 is the newest version of Kraken, a taxonomic classification system using exact k-mer matches to achieve high accuracy and fast classification speeds. This classifier matches each k-mer within a query sequence to the lowest common ancestor (LCA) of all genomes containing the given k-mer.

Detailed usage can be found here: https://ccb.jhu.edu/software/kraken2/

220.2 Versions

- 2.1.2

220.3 Commands

- kraken2
- kraken2-build
- kraken2-inspect

220.4 Module

You can load the modules by:

```
module load biocontainers
module load kraken2/2.1.2
```

220.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run kraken2 on our our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 20:00:00
#SBATCH -N 1
#SBATCH -n 24
#SBATCH --job-name=kraken2
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers kraken2/2.1.2

kraken2 --threads 24  --report kranken2.report --db minikraken2_v2_8GB_201904_UPDATE --
↪paired --classified-out cseqs#.fq SRR5043021_1.fastq SRR5043021_2.fastq
```

CHAPTER

ONE

KRAKENTOOLS

221.1 Introduction

`KrakenTools` provides individual scripts to analyze Kraken/Kraken2/Bracken/KrakenUniq output files.

Detailed usage can be found here: https://github.com/jenniferlu717/KrakenTools

221.2 Versions

- 1.2

221.3 Commands

- alpha_diversity.py
- beta_diversity.py
- combine_kreports.py
- combine_mpa.py
- extract_kraken_reads.py
- filter_bracken.out.py
- fix_unmapped.py
- kreport2krona.py
- kreport2mpa.py
- make_kreport.py
- make_ktaxonomy.py

221.4 Module

You can load the modules by:

```
module load biocontainers
module load krakentools/1.2
```

221.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run krakentools on our our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 8
#SBATCH --job-name=krakentools
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers krakentools/1.2

extract_kraken_reads.py -k myfile.kraken -t 2 -s1 SRR5043021_1.fastq -s2 SRR5043021_2.
↪fastq -o extracted1.fq -o2 extracted2.fq
```

CHAPTER

TWO

LAMBDA

222.1 Introduction

Lambda is a local aligner optimized for many query sequences and searches in protein space.

For more information, please check its website: https://biocontainers.pro/tools/lambda and its home page: http://seqan.github.io/lambda/.

222.2 Versions

- 2.0.0

222.3 Commands

- lambda2

222.4 Module

You can load the modules by:

```
module load biocontainers
module load lambda
```

222.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Lambda on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=lambda
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers lambda

lambda2 mkindexp -d uniprot_sprot.fasta

lambda2 searchp \
    -q proteins.fasta \
    -i uniprot_sprot.fasta.lambda
```

CHAPTER

THREE

LAST

223.1 Introduction

`Last` is used to find & align related regions of sequences.

For more information, please check its website: https://biocontainers.pro/tools/last and its home page on Gitlab.

223.2 Versions

- 1268
- 1356
- 1411

223.3 Commands

- last-dotplot
- last-map-probs
- last-merge-batches
- last-pair-probs
- last-postmask
- last-split
- last-split5
- last-train
- lastal
- lastal5
- lastdb
- lastdb5

223.4 Module

You can load the modules by:

```
module load biocontainers
module load last
```

223.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Last on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=last
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers last

lastdb humdb humanMito.fa
lastal humdb fuguMito.fa > myalns.maf
```

CHAPTER

FOUR

LDSC

224.1 Introduction

ldsc is a command line tool for estimating heritability and genetic correlation from GWAS summary statistics.

For more information, please check:
BioContainers: https://biocontainers.pro/tools/ldsc
Home page: https://github.com/bulik/ldsc

224.2 Versions

- 1.0.1

224.3 Commands

- ldsc.py
- munge_sumstats.py

224.4 Module

You can load the modules by:

```
module load biocontainers
module load ldsc
```

224.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run ldsc on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=ldsc
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers ldsc
```

CHAPTER

FIVE

LIFTOFF

225.1 Introduction

`Liftoff` is an accurate GFF3/GTF lift over pipeline.

For more information, please check its website: https://biocontainers.pro/tools/liftoff and its home page on Github.

225.2 Versions

- 1.6.3

225.3 Commands

- liftoff
- python
- python3

225.4 Module

You can load the modules by:

```
module load biocontainers
module load liftoff
```

225.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Liftoff on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=liftoff
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers liftoff

liftoff -g reference.gff3 -o target.gff3 \
    -chroms chr_pairs.txt target.fasta reference.fa
```

CHAPTER

SIX

LIMA

226.1 Introduction

`Lima` is the standard tool to identify barcode and primer sequences in PacBio single-molecule sequencing data.

For more information, please check its website: https://biocontainers.pro/tools/lima and its home page: https://lima.how.

226.2 Versions

- 2.2.0

226.3 Commands

- lima

226.4 Module

You can load the modules by:

```
module load biocontainers
module load lima
```

226.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Lima on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 12
#SBATCH --job-name=lima
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers lima

lima --version
lima --isoseq --dump-clips \
    --peek-guess -j 12 \
    alz.ccs.bam primers.fasta \
    alz.demult.bam
```

CHAPTER

SEVEN

LINKS

227.1 Introduction

LINKS is a genomics application for scaffolding genome assemblies with long reads, such as those produced by Oxford Nanopore Technologies Ltd. It can be used to scaffold high-quality draft genome assemblies with any long sequences (eg. ONT reads, PacBio reads, other draft genomes, etc). It is also used to scaffold contig pairs linked by ARCS/ARKS.

For more information, please check:
BioContainers: https://biocontainers.pro/tools/links
Home page: https://github.com/bcgsc/LINKS

227.2 Versions

- 2.0.1

227.3 Commands

- LINKS

227.4 Module

You can load the modules by:

```
module load biocontainers
module load links
```

227.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run links on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=links
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers links
```

CHAPTER

EIGHT

LOFREQ

228.1 Introduction

`Lofreq` is a fast and sensitive variant-caller for inferring SNVs and indels from next-generation sequencing data.

For more information, please check its website: https://biocontainers.pro/tools/lofreq and its home page on Github.

228.2 Versions

- 2.1.5

228.3 Commands

- lofreq

228.4 Module

You can load the modules by:

```
module load biocontainers
module load lofreq
```

228.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Lofreq on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 8
#SBATCH --job-name=lofreq
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers lofreq

lofreq  call -f ref.fa -o vars.vcf out_sorted.bam

lofreq call-parallel --pp-threads 8 \
     -f ref.fa -o vars_pallel.vcf out_sorted.bam
```

CHAPTER

NINE

LONGQC

229.1 Introduction

LongQC is a tool for the data quality control of the PacBio and ONT long reads.

For more information, please check:
Docker hub: https://hub.docker.com/r/cymbopogon/longqc
Home page: https://github.com/yfukasawa/LongQC

229.2 Versions

- 1.2.0c

229.3 Commands

- longQC.py

229.4 Module

You can load the modules by:

```
module load biocontainers
module load longqc
```

229.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run longqc on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=longqc
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers longqc

longQC.py sampleqc -x pb-rs2 -o out_dir seq.fastq
```

CHAPTER

LRA

230.1 Introduction

Lra is a sequence alignment program that aligns long reads from single-molecule sequencing (SMS) instruments, or megabase-scale contigs from SMS assemblies.

For more information, please check its website: https://biocontainers.pro/tools/lra and its home page on Github.

230.2 Versions

- 1.3.2

230.3 Commands

- lra

230.4 Module

You can load the modules by:

```
module load biocontainers
module load lra
```

230.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Lra on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 12
#SBATCH --job-name=lra
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers lra

lra index genome.fasta

lra align genome.fasta input.fastq -t 12 -p s > output.sam
```

CHAPTER

ONE

LTR_FINDER

231.1 Introduction

LTR_Finder is an efficient program for finding full-length LTR retrotranspsons in genome sequences.

For more information, please check:
Home page: https://github.com/xzhub/LTR_Finder

231.2 Versions

- 1.07

231.3 Commands

- ltr_finder
- check_result.pl
- down_tRNA.pl
- filter_rt.pl
- genome_plot.pl
- genome_plot2.pl
- genome_plot_svg.pl

231.4 Module

You can load the modules by:

```
module load biocontainers
module load ltr_finder
```

231.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run ltr_finder on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=ltr_finder
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers ltr_finder

ltr_finder 3ds_72.fa -P 3ds_72 -w2  > test/3ds_72_result.txt \
    |   genome_plot.pl test/
```

CHAPTER

TWO

LTRPRED

232.1 Introduction

LTRpred(ict): de novo annotation of young and intact retrotransposons.

For more information, please check:
Docker hub: https://hub.docker.com/r/drostlab/ltrpred
Home page: https://github.com/HajkD/LTRpred

232.2 Versions

- 1.1.0

232.3 Commands

- R
- Rscript

232.4 Module

You can load the modules by:

```
module load biocontainers
module load ltrpred
```

232.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run ltrpred on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=ltrpred
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers ltrpred
```

CHAPTER

THREE

LUMPY-SV

233.1 Introduction

`Lumpy-sv` is a general probabilistic framework for structural variant discovery.

For more information, please check its website: https://biocontainers.pro/tools/lumpy-sv and its home page on Github.

233.2 Versions

- 0.3.1

233.3 Commands

- lumpy
- lumpyexpress

233.4 Module

You can load the modules by:

```
module load biocontainers
module load lumpy-sv
```

233.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Lumpy-sv on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 8
#SBATCH --job-name=lumpy-sv
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers lumpy-sv

lumpy -mw 4 -tt 0.0 -pe \
bam_file:AL87.discordant.sort.bam,histo_file:AL87.histo,mean:429,stdev:84,read_length:83,
↪min_non_overlap:83,discordant_z:4,back_distance:1,weight:1,id:1,min_mapping_
↪threshold:20 \
-sr bam_file:AL87.sr.sort.bam,back_distance:1,weight:1,id:2,min_mapping_threshold:20
```

CHAPTER

FOUR

LYVESET

234.1 Introduction

Lyveset is a method of using hqSNPs to create a phylogeny, especially for outbreak investigations.

For more information, please check:
Docker hub: https://hub.docker.com/r/staphb/lyveset
Home page: https://github.com/lskatz/lyve-SET

234.2 Versions

- 2.0.1

234.3 Commands

- applyFstToTree.pl
- cladeDistancesFromTree.pl
- clusterPairwise.pl
- convertAlignment.pl
- downloadDataset.pl
- errorProneRegions.pl
- filterMatrix.pl
- filterVcf.pl
- genomeDist.pl
- launch_bwa.pl
- launch_set.pl
- launch_smalt.pl
- launch_snap.pl
- launch_snpeff.pl

- launch_varscan.pl
- makeRegions.pl
- matrixToAlignment.pl
- pairwiseDistances.pl
- pairwiseTo2d.pl
- removeUninformativeSites.pl
- removeUninformativeSitesFromMatrix.pl
- run_assembly_isFastqPE.pl
- run_assembly_metrics.pl
- run_assembly_readMetrics.pl
- run_assembly_removeDuplicateReads.pl
- run_assembly_shuffleReads.pl
- run_assembly_trimClean.pl
- set_bayesHammer.pl
- set_diagnose.pl
- set_diagnose_msa.pl
- set_downloadTestData.pl
- set_findCliffs.pl
- set_findPhages.pl
- set_indexCase.pl
- set_manage.pl
- set_processPooledVcf.pl
- set_samtools_depth.pl
- set_test.pl
- shuffleSplitReads.pl
- snpDistribution.pl
- vcfToAlignment.pl
- vcfutils.pl

234.4 Module

You can load the modules by:

```
module load biocontainers
module load lyveset
```

234.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run lyveset on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=lyveset
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers lyveset

set_test.pl lambda
set_manage.pl --create setTest
```

CHAPTER

FIVE

MACS2

235.1 Introduction

`MACS2` is Model-based Analysis of ChIP-Seq for identifying transcript factor binding sites.

For more information, please check its website: https://biocontainers.pro/tools/macs2 and its home page on Github.

235.2 Versions

- 2.2.7.1-py39

235.3 Commands

- macs2

235.4 Module

You can load the modules by:

```
module load biocontainers
module load macs2
```

235.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run MACS2 on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=macs2
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers macs2

macs2 callpeak -t ChIP.bam -c Control.bam -f BAM -g hs -n test -B -q 0.01
```

CHAPTER

SIX

MACS3

236.1 Introduction

`MACS3` is Model-based Analysis of ChIP-Seq for identifying transcript factor.

For more information, please check its | Docker hub: https://hub.docker.com/r/lbmc/macs3/3.0.0a6 and its home page on Github.

236.2 Versions

- 3.0.0a6

236.3 Commands

- macs3

236.4 Module

You can load the modules by:

```
module load biocontainers
module load macs3
```

236.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Macs3 on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=macs3
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers macs3

macs3 callpeak -t ChIP.bam -c Control.bam -f BAM -g hs -n test -B -q 0.01
```

CHAPTER

SEVEN

MAFFT

237.1 Introduction

`MAFFT` is a multiple alignment program for amino acid or nucleotide sequences.

For more information, please check its website: https://biocontainers.pro/tools/mafft and its home page: https://mafft.cbrc.jp/alignment/software/.

237.2 Versions

- 7.475
- 7.490

237.3 Commands

- einsi
- fftns
- fftnsi
- ginsi
- linsi
- mafft
- mafft-distance
- mafft-einsi
- mafft-fftns
- mafft-fftnsi
- mafft-ginsi
- mafft-homologs.rb
- mafft-linsi
- mafft-nwns

- mafft-nwnsi
- mafft-profile
- mafft-qinsi
- mafft-sparsecore.rb
- mafft-xinsi
- nwns
- nwnsi

237.4 Module

You can load the modules by:

```
module load biocontainers
module load mafft
```

237.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run MAFFT on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=mafft
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers mafft
```

CHAPTER

EIGHT

MAGECK

238.1 Introduction

Model-based Analysis of Genome-wide CRISPR-Cas9 Knockout (MAGeCK) is a computational tool to identify important genes from the recent genome-scale CRISPR-Cas9 knockout screens (or GeCKO) technology.

For more information, please check:
Docker hub: https://hub.docker.com/r/davidliwei/mageck
Home page: https://bitbucket.org/liulab/mageck/src/master/

238.2 Versions

- 0.5.9.5

238.3 Commands

- mageck
- mageckGSEA
- RRA

238.4 Module

You can load the modules by:

```
module load biocontainers
module load mageck
```

238.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run mageck on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=mageck
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers mageck


mageck count -l library.txt -n demo \
     --sample-label L1,CTRL \
     --fastq test1.fastq test2.fastq

mageck test -k demo.count.txt \
     -t L1 -c CTRL -n demo
```

CHAPTER

NINE

MAKER

239.1 Introduction

`MAKER` is a popular genome annotation pipeline for both prokaryotic and eukaryotic genomes. This guide describes best practices for running MAKER on RCAC clusters. For detailed information about MAKER, see its offical website (http://weatherby.genetics.utah.edu/MAKER/wiki/index.php/MAKER_Tutorial_for_WGS_Assembly_and_Annotation_Winter_School_2018).

239.2 Versions

- 2.31.11
- 3.01.03

239.3 Commands

- cegma2zff
- chado2gff3
- compare
- cufflinks2gff3
- evaluator
- fasta_merge
- fasta_tool
- genemark_gtf2gff3
- gff3_merge
- iprscan2gff3
- iprscan_wrap
- ipr_update_gff
- maker
- maker2chado
- maker2eval_gtf

- maker2jbrowse
- maker2wap
- maker2zff
- maker_functional
- maker_functional_fasta
- maker_functional_gff
- maker_map_ids
- map2assembly
- map_data_ids
- map_fasta_ids
- map_gff_ids
- tophat2gff3

239.4 Module

You can load the modules by:

```
module load biocontainers
module load maker/2.31.11 # OR maker/3.01.03
```

Note: `Dfam release 3.5` (October 2021) downloaded from Dfam website (https://www.dfam.org/home) that required by `RepeatMasker` has been set up for users. The `RepeatMakser` library is stored here `/depot/itap/datasets/Maker/RepeatMasker/Libraries`.

239.5 Prerequisites

1. After loading MAKER modules, users can create MAKER control files by the folowing comand:

   ```
   maker -CTL
   ```

   This will generate three files:

- **maker_opts.ctl** (required to be modified)
- **maker_exe.ctl** (do not need to modify this file)
- **maker_bopts.ctl** (optionally modify this file)

2. maker_opts.ctl: - If not using RepeatMasker, modify `model_org=all` to `model_org=` - If not using RepeatMasker, modify `model_org=all` to an appropriate family/genus/species.

239.6 Example job non-mpi

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run MAKER on our cluster:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 10:00:00
#SBATCH -N 1
#SBATCH -n 24
#SBATCH --job-name=MAKER
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers maker/2.31.11  # or maker/3.01.03

maker -c 24
```

239.7 Example job mpi

To use MAKER in MPI mode, we cannot use the maker modules. Instead we have to use the singularity image files stored in `/apps/biocontainers/images`:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 5:00:00
#SBATCH -N 2
#SBATCH -n 24
#SBATCH -c 8
#SBATCH --job-name=MAKER_mpi
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --mail-user=UserID@purdue.edu
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

## MAKER2
mpirun -n 24 singularity exec /apps/biocontainers/images/maker_2.31.11.sif maker -c 8

## MAKER3
mpirun -n 24 singularity exec /apps/biocontainers/images/maker_3.01.03.sif maker -c 8
```

MANTA

240.1 Introduction

Manta calls structural variants (SVs) and indels from mapped paired-end sequencing reads.

For more information, please check:
BioContainers: https://biocontainers.pro/tools/manta
Home page: https://github.com/Illumina/manta

240.2 Versions

- 1.6.0

240.3 Commands

- configManta.py
- python

240.4 Module

You can load the modules by:

```
module load biocontainers
module load manta
```

240.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run manta on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=manta
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers manta

configManta.py --normalBam=HCC1954.NORMAL.30x.compare.COST16011_region.bam \
    --tumorBam=G15512.HCC1954.1.COST16011_region.bam \
    --referenceFasta=Homo_sapiens_assembly19.COST16011_region.fa \
    --region=8:107652000-107655000 \
    --region=11:94974000-94989000 \
    --exome --runDir="MantaDemoAnalysis"

 python MantaDemoAnalysis/runWorkflow.py
```

CHAPTER

ONE

MAPCALLER

241.1 Introduction

`Mapcaller` is an efficient and versatile approach for short-read mapping and variant identification using high-throughput sequenced data.

For more information, please check its website: https://biocontainers.pro/tools/mapcaller and its home page on Github.

241.2 Versions

- 0.9.9.41

241.3 Commands

- MapCaller

241.4 Module

You can load the modules by:

```
module load biocontainers
module load mapcaller
```

241.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Mapcaller on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 12
#SBATCH --job-name=mapcaller
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers mapcaller

MapCaller index ref.fasta ref

MapCaller -t 12 -i ref -f input_1.fastq  -f2 input_2.fastq  -vcf out.vcf
```

CHAPTER

TWO

MARGINPOLISH

242.1 Introduction

MarginPolish is a graph-based assembly polisher. It iteratively finds multiple probable alignment paths for run-length-encoded reads and uses these to generate a refined sequence. It takes as input a FASTA assembly and an indexed BAM (ONT reads aligned to the assembly), and it produces a polished FASTA assembly.

For more information, please check:
Docker hub: https://hub.docker.com/r/kishwars/margin_polish
Home page: https://github.com/UCSC-nanopore-cgl/MarginPolish

242.2 Versions

- 0.1.3

242.3 Commands

- marginpolish

242.4 Module

You can load the modules by:

```
module load biocontainers
module load marginpolish
```

242.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run marginpolish on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 32
#SBATCH --job-name=marginpolish
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers marginpolish

marginpolish \
    Reads_to_assembly_StaphAur.bam \
    Draft_assembly_StaphAur.fasta \
    helen_modles/MP_r941_guppy344_microbial.json \
    -t 32 \
    -o mp_output/mp_images \
    -f
```

CHAPTER

THREE

MASH

243.1 Introduction

`Mash` is a fast sequence distance estimator that uses MinHash.

For more information, please check its website: https://biocontainers.pro/tools/mash and its home page on Github.

243.2 Versions

- 2.3

243.3 Commands

- mash

243.4 Module

You can load the modules by:

```
module load biocontainers
module load mash
```

243.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Mash on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=mash
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers mash

mash dist genome1.fasta genome2.fasta
```

CHAPTER

FOUR

MASHMAP

244.1 Introduction

`Mashmap` is a fast approximate aligner for long DNA sequences.

For more information, please check its website: https://biocontainers.pro/tools/mashmap and its home page on Github.

244.2 Versions

- 2.0-pl5321

244.3 Commands

- mashmap

244.4 Module

You can load the modules by:

```
module load biocontainers
module load mashmap
```

244.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Mashmap on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 12
#SBATCH --job-name=mashmap
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers mashmap

mashmap -r ref.fasta -t 12 -q input.fasta
```

CHAPTER

FIVE

MASHTREE

245.1 Introduction

`Mashtree` is a tool to create a tree using Mash distances.

For more information, please check its website: https://biocontainers.pro/tools/mashtree and its home page on Github.

245.2 Versions

- 1.2.0

245.3 Commands

- mashtree
- mashtree_bootstrap.pl
- mashtree_cluster.pl
- mashtree_init.pl
- mashtree_jackknife.pl
- mashtree_wrapper_deprecated.pl

245.4 Module

You can load the modules by:

```
module load biocontainers
module load mashtree
```

245.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Mashtree on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=mashtree
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers mashtree
```

CHAPTER

SIX

MAUVE

246.1 Introduction

`Mauve` is a system for constructing multiple genome alignments in the presence of large-scale evolutionary events such as rearrangement and inversion.

For more information, please check its website: https://biocontainers.pro/tools/mauve and its home page: http://darlinglab.org/mauve/.

246.2 Versions

- 2.4.0

246.3 Commands

- mauveAligner
- progressiveMauve

246.4 Module

You can load the modules by:

```
module load biocontainers
module load mauve
```

246.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Mauve on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=mauve
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers mauve

mauveAligner seqs.fasta --output=mauveAligner_output

progressiveMauve --output=threeway.xmfa \
    --output-guide-tree=threeway.tree \
    --backbone-output=threeway.backbone genome1.gbk genome2.gbk genome3.gbk
```

CHAPTER

SEVEN

MAXBIN2

247.1 Introduction

Maxbin2 is a software for binning assembled metagenomic sequences based on an Expectation-Maximization algorithm.

For more information, please check:
Docker hub: https://hub.docker.com/r/nanozoo/maxbin2
Home page: https://sourceforge.net/projects/maxbin2/

247.2 Versions

- 2.2.7

247.3 Commands

- run_MaxBin.pl
- run_FragGeneScan.pl

247.4 Module

You can load the modules by:

```
module load biocontainers
module load maxbin2
```

247.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run maxbin2 on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=maxbin2
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers maxbin2

run_MaxBin.pl -contig subset_assembly.fa \
     -abund_list abundance.list -max_iteration 5 -out mbin
```

CHAPTER

EIGHT

MAXQUANT

248.1 Introduction

`Maxquant` is a quantitative proteomics software package designed for analyzing large mass-spectrometric data sets. It is specifically aimed at high-resolution MS data.

For more information, please check home page: https://www.maxquant.org.

248.2 Versions

- 2.1.0.0
- 2.1.3.0
- 2.1.4.0

248.3 Commands

- MaxQuantGui.exe
- MaxQuantCmd.exe

248.4 Module

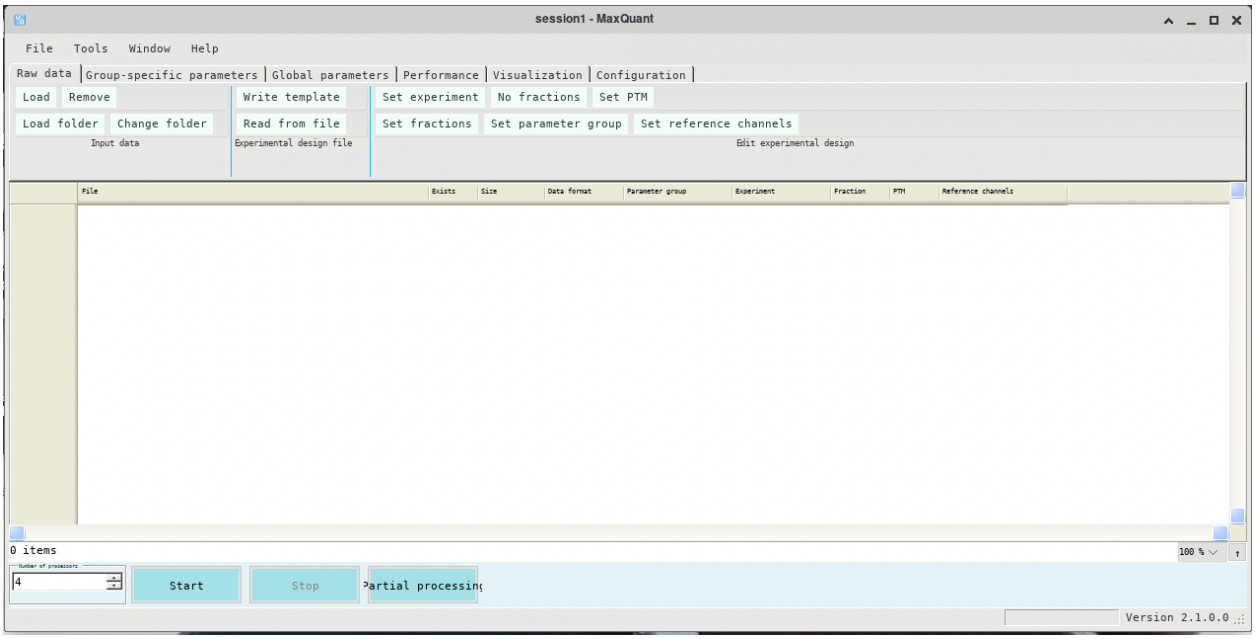
You can load the modules by:

```
module load biocontainers
module load maxquant
```

248.5 GUI

To run Maxquant with GUI, it is recommended to run within ThinLinc:

```
(base) UserID@bell-fe00:~ $ sinteractive -N1 -n12 -t4:00:00 -A myallocation
salloc: Granted job allocation 12345869
salloc: Waiting for resource configuration
salloc: Nodes bell-a008 are ready for job
(base) UserID@bell-a008:~ $ module load biocontainers maxquant
(base) UserID@bell-a008:~ $ MaxQuantGui.exe
```



248.6 CMD job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Maxquant without GUI on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=maxquant
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
```

(continues on next page)

(continued from previous page)

```
ml biocontainers maxquant

MaxQuantCmd.exe mqpar.xml
```

CHAPTER

NINE

MCL

249.1 Introduction

`Mcl` is short for the Markov Cluster Algorithm, a fast and scalable unsupervised cluster algorithm for graphs.

For more information, please check its website: https://biocontainers.pro/tools/mcl and its home page: http://micans.org/mcl/.

249.2 Versions

- 14.137-pl5262

249.3 Commands

- clm
- clmformat
- clxdo
- mcl
- mclblastline
- mclcm
- mclpipeline
- mcx
- mcxarray
- mcxassemble
- mcxdeblast
- mcxdump
- mcxi
- mcxload
- mcxmap

- mcxrand
- mcxsubs

249.4 Module

You can load the modules by:

```
module load biocontainers
module load mcl
```

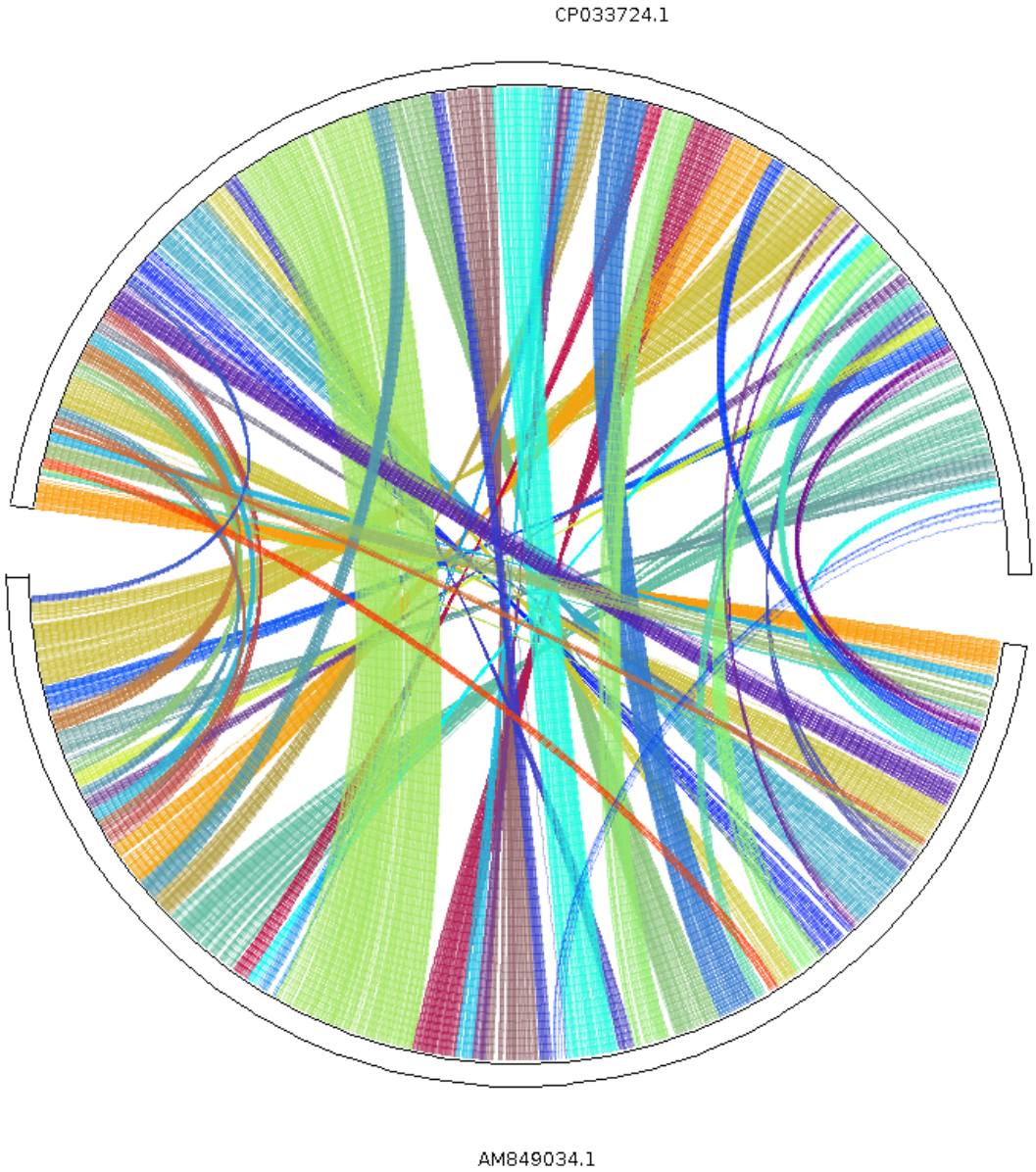
249.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Mcl on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=mcl
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers mcl
```

MCSCANX



250.1 Introduction

The MCScanX package has two major components: a modified version of MCscan algorithm allowing users to handle MCScan more conveniently and to view multiple alignment of syntenic blocks more clearly, and a variety of downstream analysis tools to conduct different biological analyses based on the synteny data generated by the modified MCScan algorithm.

For more information, please check:
Home page: https://github.com/wyp1125/MCScanX.

250.2 Versions

- default

250.3 Commands

- MCScanX
- MCScanX_h
- duplicate_gene_classifier
- add_ka_and_ks_to_collinearity
- add_kaks_to_synteny
- detect_collinearity_within_gene_families
- detect_synteny_within_gene_families
- group_collinear_genes
- group_syntenic_genes
- origin_enrichment_analysis

250.4 Module

You can load the modules by:

```
module load biocontainers
module load mcscanx
```

250.5 Helper command

Note: To conduct downstream analyses, users need to copy the folder `downstream_analyses` from container into the host system.

A helper command `copy_downstream_analyses` is provided to simplify the task. Follow the procedure below to copy downstream_analyses into target directory:

```
$ copy_downstream_analyses $PWD # this will copy the downstream_analyses into the␣
↪current directory.
```

250.6 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run mcscanx on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=mcscanx
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers mcscanx

## Run MCScanX
MCScanX Result/merge
## Copy downstream_analyses
copy_downstream_analyses $PWD
## Downstream analyses
java circle_plotter -g ../Result/merge.gff -s ../Result/merge.collinearity -c ../Result/
↪merge_circ.ctl -o ../Result/merge_circle.png
java dot_plotter -g ../Result/merge.gff -s ../Result/merge.collinearity -c ../Result/
↪merge_dot.ctl -o ../Result/merge_dot.png
java dual_synteny_plotter -g ../Result/merge.gff -s ../Result/merge.collinearity -c ../
↪Result/merge_dot.ctl -o ../Result/merge_dual_synteny.png
```

CHAPTER

ONE

MEDAKA

251.1 Introduction

`Medaka` is a tool to create consensus sequences and variant calls from nanopore sequencing data.

For more information, please check its | Docker hub: https://hub.docker.com/r/ontresearch/medaka and its home page on Github.

251.2 Versions

- 1.6.0

251.3 Commands

- medaka
- medaka_consensus
- medaka_counts
- medaka_data_path
- medaka_haploid_variant
- medaka_version_report

251.4 Module

You can load the modules by:

```
module load biocontainers
module load medaka
```

251.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Medaka on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=medaka
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers medaka
```

CHAPTER

TWO

MEGADEPTH

252.1 Introduction

`Megadepth` is an efficient tool for extracting coverage related information from RNA and DNA-seq BAM and BigWig files.

For more information, please check its website: https://biocontainers.pro/tools/megadepth and its home page on Github.

252.2 Versions

- 1.2.0

252.3 Commands

- megadepth

252.4 Module

You can load the modules by:

```
module load biocontainers
module load megadepth
```

252.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Megadepth on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=megadepth
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers megadepth

megadepth sorted.bam
```

CHAPTER

THREE

MEGAHIT

253.1 Introduction

`Megahit` is a ultra-fast single-node solution for large and complex metagenomics assembly via succinct de Bruijn graph.

For more information, please check its website: https://biocontainers.pro/tools/megahit and its home page on Github.

253.2 Versions

- 1.2.9

253.3 Commands

- megahit

253.4 Module

You can load the modules by:

```
module load biocontainers
module load megahit
```

253.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Megahit on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 12
#SBATCH --job-name=megahit
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers megahit

megahit --12 SRR1976948.abundtrim.subset.pe.fq.gz,SRR1977249.abundtrim.subset.pe.fq.gz -
↪o combined
```

CHAPTER

FOUR

MEGAN

254.1 Introduction

`Megan` is a computer program that allows optimized analysis of large metagenomic datasets. Metagenomics is the analysis of the genomic sequences from a usually uncultured environmental sample.

For more information, please check its website: https://biocontainers.pro/tools/megan and its home page: https://uni-tuebingen.de/fakultaeten/mathematisch-naturwissenschaftliche-fakultaet/fachbereiche/informatik/lehrstuehle/algorithms-in-bioinformatics/software/megan6/.

254.2 Versions

- 6.21.7

254.3 Commands

- MEGAN
- blast2lca
- blast2rma
- daa2info
- daa2rma
- daa-meganizer
- gc-assembler
- rma2info
- sam2rma
- references-annotator

254.4 Module

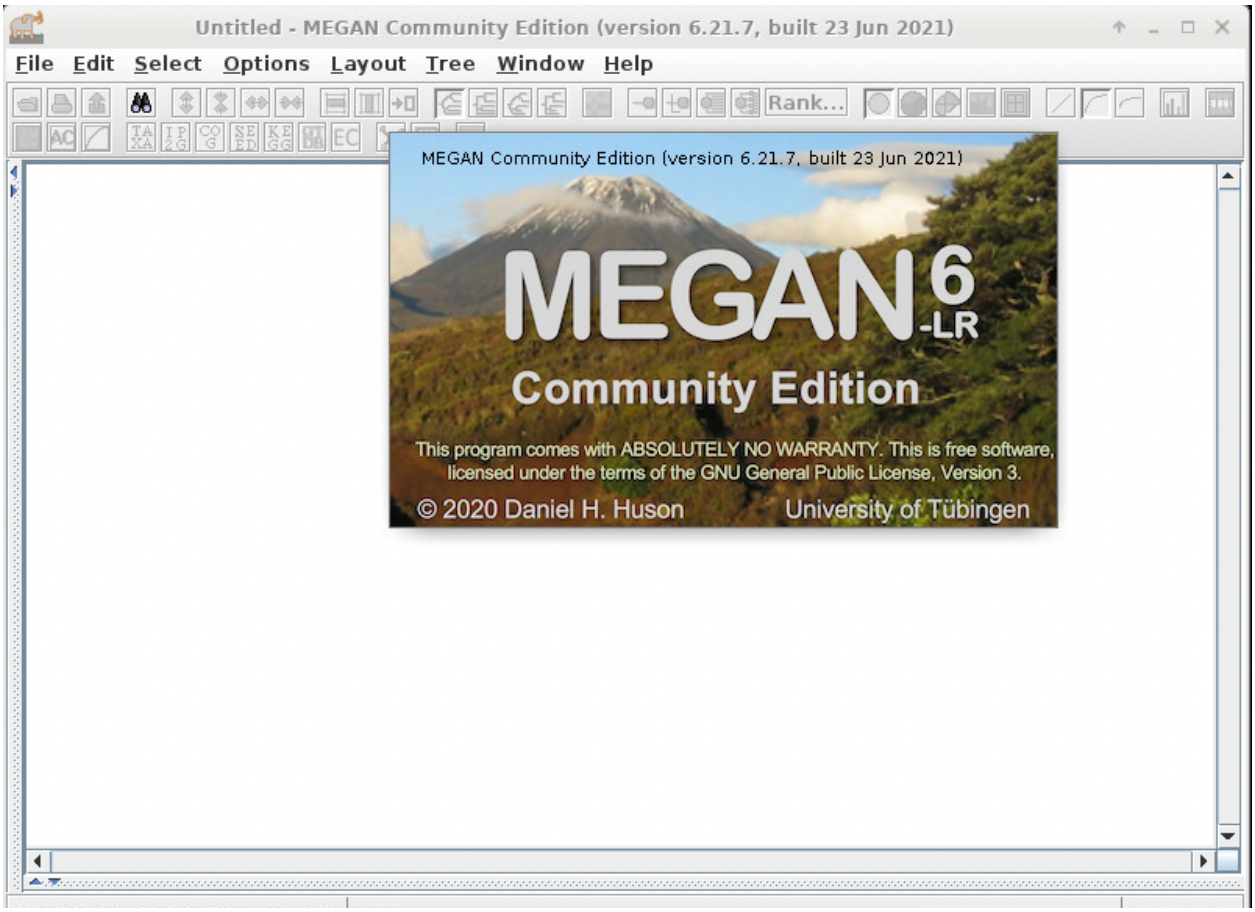
You can load the modules by:

```
module load biocontainers
module load megan
```

254.5 GUI

To run MEGAN with GUI, it is recommended to run within ThinLinc:

```
(base) UserID@bell-fe00:~ $ sinteractive -N1 -n12 -t4:00:00 -A myallocation
salloc: Granted job allocation 12345869
salloc: Waiting for resource configuration
salloc: Nodes bell-a008 are ready for job
(base) UserID@bell-a008:~ $ module load biocontainers megan
(base) UserID@bell-a008:~ $ MEGAN
```



254.6 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Megan on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=megan
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers megan
```

CHAPTER

FIVE

MEME

255.1 Introduction

`Meme` is a collection of tools for the discovery and analysis of sequence motifs.

For more information, please check its website: https://biocontainers.pro/tools/meme and its home page: https://meme-suite.org/meme/.

255.2 Versions

- 5.3.3
- 5.4.1

255.3 Commands

- ame
- centrimo
- dreme
- dust
- fimo
- glam2
- glam2scan
- gomo
- mast
- mcast
- meme
- meme-chip
- momo
- purge

- spamo
- tomtom

255.4 Module

You can load the modules by:

```
module load biocontainers
module load meme
```

255.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Meme on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=meme
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers meme

meme seq.fasta -dna -mod oops -pal

meme-chip Klf1.fna -o memechip_klf1_out
```

CHAPTER

SIX

MERQURY

256.1 Introduction

Merqury is a tool to evaluate genome assemblies with k-mers and more.

For more information, please check:
Docker hub: https://hub.docker.com/r/dovetailg/merqury
Home page: https://github.com/marbl/merqury

256.2 Versions

- 1.3

256.3 Commands

- merqury.sh

256.4 Module

You can load the modules by:

```
module load biocontainers
module load merqury
```

256.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run merqury on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=merqury
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers merqury

merqury.sh F1.k18.meryl col0.hapmer.meryl cvi0.hapmer.meryl \
    athal_COL.fasta athal_CVI.fasta test
```

CHAPTER

SEVEN

MERYL

257.1 Introduction

`Meryl` is a genomic k-mer counter (and sequence utility) with nice features.

For more information, please check its website: https://biocontainers.pro/tools/meryl and its home page on Github.

257.2 Versions

- 1.3

257.3 Commands

- meryl
- meryl-analyze
- meryl-import
- meryl-lookup
- meryl-simple

257.4 Module

You can load the modules by:

```
module load biocontainers
module load meryl
```

257.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Meryl on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=meryl
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers meryl

meryl count k=42 data/ec.fna.gz output ec.meryl
```

CHAPTER

EIGHT

METABAT

258.1 Introduction

`Metabat` is a robust statistical framework for reconstructing genomes from metagenomic data.

For more information, please check its | Docker hub: https://hub.docker.com/r/metabat/metabat and its home page: https://bitbucket.org/berkeleylab/metabat/src/master/

258.2 Versions

- 2.15-5

258.3 Commands

- aggregateBinDepths.pl
- aggregateContigOverlapsByBin.pl
- contigOverlaps
- jgi_summarize_bam_contig_depths
- merge_depths.pl
- metabat
- metabat1
- metabat2
- runMetaBat.sh

258.4 Module

You can load the modules by:

```
module load biocontainers
module load metabat
```

258.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Metabat on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=metabat
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers metabat
```

CHAPTER

NINE

METAPHLAN 3

259.1 Introduction

`MetaPhlAn` (Metagenomic Phylogenetic Analysis) is a computational tool for profiling the composition of microbial communities from metagenomic shotgun sequencing data. MetaPhlAn relies on unique clade-specific marker genes identified from ~17,000 reference genomes (~13,500 bacterial and archaeal, ~3,500 viral, and ~110 eukaryotic), allowing:

- up to 25,000 reads-per-second (on one CPU) analysis speed (orders of magnitude faster compared to existing methods);
- unambiguous taxonomic assignments as the MetaPhlAn markers are clade-specific;
- accurate estimation of organismal relative abundance (in terms of number of cells rather than fraction of reads);
- species-level resolution for bacteria, archaea, eukaryotes and viruses;
- extensive validation of the profiling accuracy on several synthetic datasets and on thousands of real metagenomes.

For more information, please check its user guide at: https://huttenhower.sph.harvard.edu/metaphlan/

259.2 Versions

- 3.0.14
- 3.0.9

259.3 Commands

metaphlan

259.4 Database

The lastest version of database(mpa_v30) has been downloaded and built in /depot/itap/datasets/metaphlan/.

259.5 Module

You can load the modules by:

```
module load biocontainers
module load metaphlan/3.0.14
```

259.6 Example job

Warning: Using #!/bin/sh -l as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use #!/bin/bash instead.

To run MetaPhlAn on our cluster:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 10:00:00
#SBATCH -N 1
#SBATCH -n 24
#SBATCH --job-name=MetaPhlAn
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers metaphlan/3.0.14

DATABASE=/depot/itap/datasets/metaphlan/
metaphlan SRR11234553_1.fastq,SRR11234553_2.fastq --input_type fastq --nproc 24 -o␣
↪profiled_metagenome.txt --bowtie2db $DATABASE  --bowtie2out metagenome.bowtie2.bz2
```

METHYLDACKEL

260.1 Introduction

MethylDackel (formerly named PileOMeth, which was a temporary name derived due to it using a PILEup to extract METHylation metrics) will process a coordinate-sorted and indexed BAM or CRAM file containing some form of BS-seq alignments and extract per-base methylation metrics from them. MethylDackel requires an indexed fasta file containing the reference genome as well.

For more information, please check:
BioContainers: https://biocontainers.pro/tools/methyldackel
Home page: https://github.com/dpryan79/MethylDackel

260.2 Versions

- 0.6.1

260.3 Commands

- MethylDackel

260.4 Module

You can load the modules by:

```
module load biocontainers
module load methyldackel
```

260.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run methyldackel on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=methyldackel
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers methyldackel

MethylDackel extract chgchh.fa chgchh_aln.bam
```

CHAPTER

ONE

METILENE

261.1 Introduction

Metilene is a versatile tool to study the effect of epigenetic modifications in differentiation/development, tumorigenesis, and systems biology on a global, genome-wide level.

For more information, please check:
BioContainers: https://biocontainers.pro/tools/metilene
Home page: https://www.bioinf.uni-leipzig.de/Software/metilene/

261.2 Versions

- 0.2.8

261.3 Commands

- metilene
- metilene_input.pl
- metilene_output.pl
- metilene_output.R

261.4 Module

You can load the modules by:

```
module load biocontainers
module load metilene
```

261.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run metilene on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=metilene
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers metilene

metilene -a g1 -b g2 methylation-file
```

CHAPTER

TWO

MHM2

262.1 Introduction

MetaHipMer is a de novo metagenome short-read assembler. Version 2 (MHM2) is written entirely in UPC++ and runs efficiently on both single servers and on multinode supercomputers, where it can scale up to coassemble terabase-sized metagenomes.

For more information, please check:
Home page: https://bitbucket.org/berkeleylab/mhm2/wiki/Home.md

262.2 Versions

- 2.0.0

262.3 Commands

- mhm2.py

262.4 Module

You can load the modules by:

```
module load biocontainers
module load mhm2
```

262.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run mhm2 on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=mhm2
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers mhm2

mhm2.py -r input_1.fastq,input_2.fastq
```

CHAPTER

THREE

MICROBEDMM

263.1 Introduction

`MicrobeDMM` is a suite of programs used for empirical Bayes fitting of DMM models.

For more information, please check its home page: https://code.google.com/archive/p/microbedmm.

263.2 Versions

- 1.0

263.3 Commands

- DirichletMixtureGHPFit

263.4 Module

You can load the modules by:

```
module load biocontainers
module load microbedmm
```

263.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run MicrobeDMM on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=microbedmm
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers microbedmm
```

CHAPTER

FOUR

MINIALIGN

264.1 Introduction

`Minialign` is a little bit fast and moderately accurate nucleotide sequence alignment tool designed for PacBio and Nanopore long reads.

For more information, please check its website: https://biocontainers.pro/tools/minialign and its home page on Github.

264.2 Versions

- 0.5.3

264.3 Commands

- minialign

264.4 Module

You can load the modules by:

```
module load biocontainers
module load minialign
```

264.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Minialign on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=minialign
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers minialign

minialign -d index.mai genome.fasta
minialign -l index.mai input.fastq > out.sam
```

CHAPTER

FIVE

MINIASM

265.1 Introduction

`Miniasm` is a very fast OLC-based de novo assembler for noisy long reads.

For more information, please check its website: https://biocontainers.pro/tools/miniasm and its home page on Github.

265.2 Versions

- 0.3_r179

265.3 Commands

- miniasm
- minidot

265.4 Module

You can load the modules by:

```
module load biocontainers
module load miniasm
```

265.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Miniasm on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 12
#SBATCH --job-name=miniasm
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers miniasm

miniasm -f Elysia_ont_test.fq  Elysia_reads.paf.gz \
     > Elysia_reads.gfa
```

CHAPTER

SIX

MINIMAP2

266.1 Introduction

`Minimap2` is a versatile pairwise aligner for genomic and spliced nucleotide sequences.

For more information, please check its website: https://biocontainers.pro/tools/minimap2 and its home page on Github.

266.2 Versions

- 2.22
- 2.24

266.3 Commands

- minimap2
- paftools.js
- k8

266.4 Module

You can load the modules by:

```
module load biocontainers
module load minimap2
```

266.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Minimap2 on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=minimap2
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers minimap2

minimap2 -ax sr Wuhan-Hu-1.fasta \
    seq_1.fastq seq_2.fastq \
    > aln.sam
```

CHAPTER

SEVEN

MINIPOLISH

267.1 Introduction

Minipolish is a tool for Racon polishing of miniasm assemblies.

For more information, please check:
Docker hub: https://hub.docker.com/r/staphb/minipolish
Home page: https://github.com/rrwick/Minipolish

267.2 Versions

- 0.1.3

267.3 Commands

- minipolish

267.4 Module

You can load the modules by:

```
module load biocontainers
module load minipolish
```

267.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run minipolish on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 8
#SBATCH --job-name=minipolish
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers minipolish

minipolish -t 8 long_reads.fastq.gz assembly.gfa > polished.gfa
```

CHAPTER

EIGHT

MINIPROT

268.1 Introduction

Miniprot aligns a protein sequence against a genome with affine gap penalty, splicing and frameshift. It is primarily intended for annotating protein-coding genes in a new species using known genes from other species. Miniprot is similar to GeneWise and Exonerate in functionality but it can map proteins to whole genomes and is much faster at the residue alignment step.

For more information, please check:
Home page: https://github.com/lh3/miniprot

268.2 Versions

- 0.3

268.3 Commands

- miniprot

268.4 Module

You can load the modules by:

```
module load biocontainers
module load miniprot
```

268.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run miniprot on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=miniprot
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers miniprot
```

CHAPTER

NINE

MIRDEEP2

269.1 Introduction

`miRDeep2` discovers active known or novel miRNAs from deep sequencing data (Solexa/Illumina, 454, ...).

For more information, please check its website: https://biocontainers.pro/tools/mirdeep2 and its home page on Github.

269.2 Versions

- 2.0.1.3

269.3 Commands

- bwa_sam_converter.pl
- clip_adapters.pl
- collapse_reads_md.pl
- convert_bowtie_output.pl
- excise_precursors_iterative_final.pl
- excise_precursors.pl
- extract_miRNAs.pl
- fastaparse.pl
- fastaselect.pl
- fastq2fasta.pl
- find_read_count.pl
- geo2fasta.pl
- get_mirdeep2_precursors.pl
- illumina_to_fasta.pl
- make_html2.pl

- make_html.pl
- mapper.pl
- mirdeep2bed.pl
- miRDeep2_core_algorithm.pl
- miRDeep2.pl
- parse_mappings.pl
- perform_controls.pl
- permute_structure.pl
- prepare_signature.pl
- quantifier.pl
- remove_white_space_in_id.pl
- rna2dna.pl
- samFLAGinfo.pl
- sam_reads_collapse.pl
- sanity_check_genome.pl
- sanity_check_mapping_file.pl
- sanity_check_mature_ref.pl
- sanity_check_reads_ready_file.pl
- select_for_randfold.pl
- survey.pl

269.4 Module

You can load the modules by:

```
module load biocontainers
module load mirdeep2
```

269.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run miRDeep2 on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
```

(continues on next page)

(continued from previous page)

```
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=mirdeep2
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers mirdeep2
```

CHAPTER

MIRTOP

270.1 Introduction

Mirtop is a ommand line tool to annotate with a standard naming miRNAs e isomiRs.

For more information, please check:
BioContainers: https://biocontainers.pro/tools/mirtop
Home page: https://github.com/miRTop/mirtop

270.2 Versions

- 0.4.25

270.3 Commands

- mirtop

270.4 Module

You can load the modules by:

```
module load biocontainers
module load mirtop
```

270.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run mirtop on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=mirtop
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers mirtop

mirtop gff --format prost --sps hsa
    --hairpin examples/annotate/hairpin.fa \
    --gtf examples/annotate/hsa.gff3 \
    -o test_out \
    examples/prost/prost.example.txt
```

CHAPTER

ONE

MITOFINDER

271.1 Introduction

`Mitofinder` is a pipeline to assemble mitochondrial genomes and annotate mitochondrial genes from trimmed read sequencing data.

For more information, please check its website: https://cloud.sylabs.io/library/remiallio/default/mitofinder and its home page on Github.

271.2 Versions

- 1.4.1

271.3 Commands

- mitofinder

271.4 Module

You can load the modules by:

```
module load biocontainers
module load mitofinder
```

271.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Mitofinder on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=mitofinder
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers mitofinder

mitofinder -j Aphaenogaster_megommata_SRR1303315 \
           -1 Aphaenogaster_megommata_SRR1303315_R1_cleaned.fastq.gz \
           -2 Aphaenogaster_megommata_SRR1303315_R2_cleaned.fastq.gz \
           -r reference.gb -o 5 -p 5 -m 10
```

CHAPTER

TWO

MLST

272.1 Introduction

Mlst is used to scan contig files against traditional PubMLST typing schemes.

For more information, please check:
Docker hub: https://hub.docker.com/r/staphb/mlst
Home page: https://github.com/tseemann/mlst

272.2 Versions

- 2.22.0

272.3 Commands

- mlst

272.4 Module

You can load the modules by:

```
module load biocontainers
module load mlst
```

272.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run mlst on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=mlst
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers mlst

mlst contigs.fa
mlst genome.gbk.gz
```

CHAPTER

THREE

MMSEQS2

273.1 Introduction

`Mmseqs2` is a software suite to search and cluster huge protein and nucleotide sequence sets.

For more information, please check its website: https://biocontainers.pro/tools/mmseqs2 and its home page on Github.

273.2 Versions

- 13.45111

273.3 Commands

- mmseqs

273.4 Module

You can load the modules by:

```
module load biocontainers
module load mmseqs2
```

273.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Mmseqs2 on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=mmseqs2
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers mmseqs2

mmseqs createdb examples/DB.fasta targetDB
mmseqs createtaxdb targetDB tmp
mmseqs createindex targetDB tmp
mmseqs easy-taxonomy examples/QUERY.fasta targetDB alnRes tmp
```

CHAPTER

FOUR

MOTHUR

274.1 Introduction

`Mothur` is an open source software package for bioinformatics data processing. The package is frequently used in the analysis of DNA from uncultured microbes.

Detailed information about Mothur can be found here: https://mothur.org

274.2 Versions

- 1.46.0
- 1.47.0
- 1.48.0

274.3 Commands

- mothur

274.4 Module

You can load the modules by:

```
module load biocontainers
module load mothur/1.47.0
```

274.5 Interactive job

To run mothur interactively on our clusters:

```
(base) UserID@bell-fe00:~ $ sinteractive -N1 -n12 -t4:00:00 -A myallocation
salloc: Granted job allocation 12345869
salloc: Waiting for resource configuration
salloc: Nodes bell-a008 are ready for job
(base) UserID@bell-a008:~ $ module load biocontainers mothur/1.47.0
(base) UserID@bell-a008:~ $ mothur
Linux version

Using ReadLine,Boost,HDF5,GSL
mothur v.1.47.0
Last updated: 1/21/22
by
Patrick D. Schloss

Department of Microbiology & Immunology

University of Michigan
http://www.mothur.org

When using, please cite:
Schloss, P.D., et al., Introducing mothur: Open-source, platform-independent, community-
↪supported software for describing and comparing microbial communities. Appl Environ␣
↪Microbiol, 2009. 75(23):7537-41.

Distributed under the GNU General Public License

Type 'help()' for information on the commands that are available

For questions and analysis support, please visit our forum at https://forum.mothur.org

Type 'quit()' to exit program

[NOTE]: Setting random seed to 19760620.

Interactive Mode

mothur > align.seqs(help)
mothur > quit()
```

274.6 Batch job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To submit a sbatch job on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 10:00:00
#SBATCH -N 1
#SBATCH -n 24
#SBATCH --job-name=mothur
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers mothur/1.47.0

mothur batch_file
```

CHAPTER

FIVE

MRBAYES

275.1 Introduction

`MrBayes` is a program for Bayesian inference and model choice across a wide range of phylogenetic and evolutionary models. MrBayes uses Markov chain Monte Carlo (MCMC) methods to estimate the posterior distribution of model parameters.

MrBayes is available both in a serial version ('mb') and in a parallel version ('mb-mpi') that uses MPI instructions to distribute computations across several processors or processor cores. The serial version does *not* support multi-threading, which means that you will not be able to utilize more than one core on a multi-core machine for a single MrBayes analysis. If you want to utilize all cores,you need to run the MPI version of MrBayes.

Note: 'mb-mpi' in this version of the container does not run across multiple nodes (only within a node). This is a bug in the container (upstream).

For more information, please check its website: https://biocontainers.pro/tools/mrbayes and its home page: http://mrbayes.net.

275.2 Versions

- 3.2.7

275.3 Commands

- mb
- mb-mpi
- mpirun
- mpiexec

275.4 Module

You can load the modules by:

```
module load biocontainers
module load mrbayes
```

275.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run MrBayes on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=mrbayes
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers mrbayes
```

CHAPTER

SIX

MULTIQC

276.1 Introduction

`Multiqc` is a reporting tool that parses summary statistics from results and log files generated by other bioinformatics tools.

For more information, please check its website: https://biocontainers.pro/tools/multiqc and its home page: https://multiqc.info.

276.2 Versions

- 1.11

276.3 Commands

- multiqc

276.4 Module

You can load the modules by:

```
module load biocontainers
module load multiqc
```

276.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Multiqc on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=multiqc
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers multiqc

multiqc fastqc_out -o multiqc_out
```

CHAPTER

SEVEN

MUMMER4

277.1 Introduction

`Mummer4` is a versatile alignment tool for DNA and protein sequences.

For more information, please check its website: https://biocontainers.pro/tools/mummer4 and its home page on Github.

277.2 Versions

- 4.0.0rc1-pl5262

277.3 Commands

- annotate
- combineMUMs
- delta-filter
- delta2vcf
- dnadiff
- exact-tandems
- mummer
- mummerplot
- nucmer
- promer
- repeat-match
- show-aligns
- show-coords
- show-diff
- show-snps

- show-tiling

277.4 Module

You can load the modules by:

```
module load biocontainers
module load mummer4
```

277.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Mummer4 on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=mummer4
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers mummer4

mummer -mum -b -c H_pylori26695_Eslice.fasta H_pyloriJ99_Eslice.fasta > mummer.mums
```

CHAPTER

EIGHT

MUSCLE

278.1 Introduction

`Muscle` is a modified progressive alignment algorithm which has comparable accuracy to MAFFT, but faster performance.

For more information, please check its website: https://biocontainers.pro/tools/muscle and its home page: http://www.drive5.com/muscle/muscle_userguide3.8.html.

278.2 Versions

- 3.8.1551
- 5.1

278.3 Versions

- 3.8.1551
- 5.1

278.4 Commands

- muscle

278.5 Module

You can load the modules by:

```
module load biocontainers
module load muscle
```

278.6 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Muscle on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=muscle
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers muscle

muscle -align seqs2.fasta  -output seqs.afa
```

CHAPTER

NINE

MUTMAP

279.1 Introduction

MutMap is a powerful and efficient method to identify agronomically important loci in crop plants.

For more information, please check:
BioContainers: https://biocontainers.pro/tools/mutmap
Home page: https://github.com/YuSugihara/MutMap#What-is-MutMap

279.2 Versions

- 2.3.3

279.3 Commands

- mutmap
- mutplot

279.4 Module

You can load the modules by:

```
module load biocontainers
module load mutmap
```

279.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run mutmap on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=mutmap
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers mutmap
```

CHAPTER

MYKROBE

280.1 Introduction

Mykrobe analyses the whole genome of a bacterial sample, all within a couple of minutes, and predicts which drugs the infection is resistant to.

For more information, please check:
Docker hub: https://hub.docker.com/r/staphb/mykrobe
Home page: https://github.com/Mykrobe-tools/mykrobe

280.2 Versions

- 0.11.0

280.3 Commands

- mykrobe

280.4 Module

You can load the modules by:

```
module load biocontainers
module load mykrobe
```

280.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run mykrobe on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=mykrobe
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers mykrobe
```

CHAPTER

ONE

NANOFILT

281.1 Introduction

`Nanofilt` is a tool for filtering and trimming of Oxford Nanopore Sequencing data.

For more information, please check its website: https://biocontainers.pro/tools/nanofilt and its home page on Github.

281.2 Versions

- 2.8.0

281.3 Commands

- NanoFilt

281.4 Module

You can load the modules by:

```
module load biocontainers
module load nanofilt
```

281.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Nanofilt on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=nanofilt
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers nanofilt

NanoFilt -q 12 --headcrop 75 reads.fastq |  gzip > trimmed-reads.fastq.gz
```

CHAPTER

TWO

NANOLYSE

282.1 Introduction

`Nanolyse` is a tool to remove reads mapping to the lambda phage genome from a fastq file.

For more information, please check its website: https://biocontainers.pro/tools/nanolyse and its home page on Github.

282.2 Versions

- 1.2.0

282.3 Commands

- NanoLyse

282.4 Module

You can load the modules by:

```
module load biocontainers
module load nanolyse
```

282.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Nanolyse on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=nanolyse
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers nanolyse

gunzip -c reads.fastq.gz |  NanoLyse |  gzip > reads_without_lambda.fastq.gz
```

CHAPTER

THREE

NANOPLOT

283.1 Introduction

`Nanoplot` is a plotting tool for long read sequencing data and alignments.

For more information, please check its website: https://biocontainers.pro/tools/nanoplot and its home page on Github.

283.2 Versions

- 1.39.0

283.3 Commands

- NanoPlot

283.4 Module

You can load the modules by:

```
module load biocontainers
module load nanoplot
```

283.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Nanoplot on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 12
#SBATCH --job-name=nanoplot
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers nanoplot

NanoPlot --summary sequencing_summary.txt --loglength -o summary-plots-log-transformed
NanoPlot -t 2 --fastq reads1.fastq.gz reads2.fastq.gz --maxlength 40000 --plots dot --
↪legacy hex
NanoPlot -t 12 --color yellow --bam alignment1.bam alignment2.bam alignment3.bam --
↪downsample 10000 -o bamplots_downsampled
```

CHAPTER

FOUR

NANOPOLISH

284.1 Introduction

`Nanopolish` is a software package for signal-level analysis of Oxford Nanopore sequencing data.

For more information, please check its website: https://biocontainers.pro/tools/nanopolish and its home page on Github.

284.2 Versions

- 0.13.2
- 0.14.0

284.3 Commands

- nanopolish

284.4 Module

You can load the modules by:

```
module load biocontainers
module load nanopolish
```

284.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Nanopolish on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=nanopolish
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers nanopolish

nanopolish index -d fast5_files/ reads.fasta

nanopolish variants --consensus \
    -o polished.vcf -w "tig00000001:200000-202000" \
     -r reads.fasta -b reads.sorted.bam  -g draft.fa
```

CHAPTER

FIVE

NCBI-AMRFINDERPLUS

285.1 Introduction

`Ncbi-amrfinderplus` and the accompanying database identify acquired antimicrobial resistance genes in bacterial protein and/or assembled nucleotide sequences as well as known resistance-associated point mutations for several taxa.

For more information, please check:
BioContainers: https://biocontainers.pro/tools/ncbi-amrfinderplus
Home page: https://github.com/ncbi/amr

285.2 Versions

- 3.10.30

285.3 Commands

- amrfinder

285.4 Module

You can load the modules by:

```
module load biocontainers
module load ncbi-amrfinderplus
```

Note: AMRFinderPlus database has been setup for users. Users can check the database version by `amrfinder -V`. RCAC will keep updating database for users. If you notice our database is out of date, you can contact us to update the database.

285.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run ncbi-amrfinderplus on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=ncbi-amrfinderplus
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers ncbi-amrfinderplus

# Protein AMRFinder with no genomic coordinates
amrfinder -p test_prot.fa

# Translated nucleotide AMRFinder (will not use HMMs)
amrfinder -n test_dna.fa

# Protein AMRFinder using GFF to get genomic coordinates and 'plus' genes
amrfinder -p test_prot.fa -g test_prot.gff --plus

# Protein AMRFinder with Escherichia protein point mutations
amrfinder -p test_prot.fa -O Escherichia

# Full AMRFinderPlus search combining results
amrfinder -p test_prot.fa -g test_prot.gff -n test_dna.fa -O Escherichia --plus
```

CHAPTER

SIX

NCBI-GENOME-DOWNLOAD

286.1 Introduction

`Ncbi-genome-download` is a script to download genomes from the NCBI FTP servers.

For more information, please check its website: https://biocontainers.pro/tools/ncbi-genome-download and its home page on Github.

286.2 Versions

- 0.3.1

286.3 Commands

- ncbi-genome-download

286.4 Module

You can load the modules by:

```
module load biocontainers
module load ncbi-genome-download
```

286.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Ncbi-genome-download on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 4
#SBATCH --job-name=ncbi-genome-download
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers ncbi-genome-download

ncbi-genome-download bacteria,viral --parallel 4
ncbi-genome-download --genera "Streptomyces coelicolor,Escherichia coli" bacteria
ncbi-genome-download --species-taxids 562 bacteria
```

CHAPTER

SEVEN

NEUSOMATIC

287.1 Introduction

NeuSomatic is based on deep convolutional neural networks for accurate somatic mutation detection. With properly trained models, it can robustly perform across sequencing platforms, strategies, and conditions. NeuSomatic summarizes and augments sequence alignments in a novel way and incorporates multi-dimensional features to capture variant signals effectively. It is not only a universal but also accurate somatic mutation detection method.

For more information, please check:
Docker hub: https://hub.docker.com/r/msahraeian/neusomatic/
Home page: https://github.com/bioinform/neusomatic

287.2 Versions

- 0.2.1

287.3 Commands

- call.py
- dataloader.py
- extract_postprocess_targets.py
- filter_candidates.py
- generate_dataset.py
- long_read_indelrealign.py
- merge_post_vcfs.py
- merge_tsvs.py
- network.py
- postprocess.py
- preprocess.py
- resolve_scores.py

- resolve_variants.py
- scan_alignments.py
- split_bed.py
- train.py
- utils.py

287.4 Module

You can load the modules by:

```
module load biocontainers
module load neusomatic
```

287.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run neusomatic on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=neusomatic
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers neusomatic
```

CHAPTER

EIGHT

NEXTALIGN

288.1 Introduction

`Nextalign` is a viral genome sequence alignment tool for command line.

For more information, please check its | Docker hub: https://hub.docker.com/r/nextstrain/nextalign and its home page: https://docs.nextstrain.org/projects/nextclade/en/stable/user/nextalign-cli.html.

288.2 Versions

- 1.10.3

288.3 Commands

- nextalign

288.4 Module

You can load the modules by:

```
module load biocontainers
module load nextalign
```

288.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Nextalign on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=nextalign
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers nextalign

nextalign \
     --sequences data/sars-cov-2/sequences.fasta \
     --reference data/sars-cov-2/reference.fasta \
     --genemap data/sars-cov-2/genemap.gff \
    --genes E,M,N,ORF1a,ORF1b,ORF3a,ORF6,ORF7a,ORF7b,ORF8,ORF9b,S \
    --output-dir output/ \
    --output-basename nextalign
```

CHAPTER

NINE

NEXTCLADE

289.1 Introduction

`Nextclade` is a tool that identifies differences between your sequences and a reference sequence, uses these differences to assign your sequences to clades, and reports potential sequence quality issues in your data.

For more information, please check its | Docker hub: https://hub.docker.com/r/nextstrain/nextclade and its home page: https://docs.nextstrain.org/projects/nextclade/en/stable/user/nextclade-cli.html.

289.2 Versions

- 1.10.3

289.3 Commands

- nextclade

289.4 Module

You can load the modules by:

```
module load biocontainers
module load nextclade
```

289.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Nextclade on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=nextclade
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers nextclade

mkdir -p data
nextclade dataset get --name 'sars-cov-2' --output-dir 'data/sars-cov-2'

nextclade \
    --in-order \
    --input-fasta data/sars-cov-2/sequences.fasta \
    --input-dataset data/sars-cov-2 \
    --output-tsv output/nextclade.tsv \
    --output-tree output/nextclade.auspice.json \
    --output-dir output/ \
    --output-basename nextclade
```

NEXTFLOW

290.1 Introduction

`Nextflow` is a bioinformatics workflow manager that enables the development of portable and reproducible workflows.

For more information, please check its website: https://biocontainers.pro/tools/nextflow and its home page on Github.

290.2 Versions

- 21.10.0

290.3 Commands

- nextflow

290.4 Module

You can load the modules by:

```
module load biocontainers
module load nextflow
```

290.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Nextflow on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
```

(continues on next page)

(continued from previous page)

```
#SBATCH -n 1
#SBATCH --job-name=nextflow
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers nextflow
```

CHAPTER

ONE

NGS-BITS

291.1 Introduction

`Ngs-bits` - Short-read sequencing tools.

For more information, please check its website: https://biocontainers.pro/tools/ngs-bits and its home page on Github.

291.2 Versions

- 2022_04

291.3 Commands

- SampleAncestry
- SampleDiff
- SampleGender
- SampleOverview
- SampleSimilarity
- SeqPurge
- CnvHunter
- RohHunter
- UpdHunter
- CfDnaQC
- MappingQC
- NGSDImportQC
- ReadQC
- SomaticQC
- VariantQC
- TrioMaternalContamination

- BamCleanHaloplex
- BamClipOverlap
- BamDownsample
- BamFilter
- BamToFastq
- BedAdd
- BedAnnotateFreq
- BedAnnotateFromBed
- BedAnnotateGC
- BedAnnotateGenes
- BedChunk
- BedCoverage
- BedExtend
- BedGeneOverlap
- BedHighCoverage
- BedInfo
- BedIntersect
- BedLiftOver
- BedLowCoverage
- BedMerge
- BedReadCount
- BedShrink
- BedSort
- BedSubtract
- BedToFasta
- BedpeAnnotateBreakpointDensity
- BedpeAnnotateCnvOverlap
- BedpeAnnotateCounts
- BedpeAnnotateFromBed
- BedpeFilter
- BedpeGeneAnnotation
- BedpeSort
- BedpeToBed
- FastqAddBarcode
- FastqConcat
- FastqConvert

- FastqDownsample
- FastqExtract
- FastqExtractBarcode
- FastqExtractUMI
- FastqFormat
- FastqList
- FastqMidParser
- FastqToFasta
- FastqTrim
- VcfAnnotateFromBed
- VcfAnnotateFromBigWig
- VcfAnnotateFromVcf
- VcfBreakMulti
- VcfCalculatePRS
- VcfCheck
- VcfExtractSamples
- VcfFilter
- VcfLeftNormalize
- VcfSort
- VcfStreamSort
- VcfToBedpe
- VcfToTsv
- SvFilterAnnotations
- NGSDExportGenes
- GenePrioritization
- GenesToApproved
- GenesToBed
- GraphStringDb
- PhenotypeSubtree
- PhenotypesToGenes
- PERsim
- FastaInfo

291.4 Module

You can load the modules by:

```
module load biocontainers
module load ngs-bits
```

291.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Ngs-bits on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=ngs-bits
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers ngs-bits

SeqPurge -in1 input1_1.fastq input2_1.fastq \
     -in2 input2_2.fastq input2_2.fastq \
     -out1 R1.fastq.gz -out2 R2.fastq.gz
```

CHAPTER

TWO

NGSUTILS

292.1 Introduction

`Ngsutils` is a suite of software tools for working with next-generation sequencing datasets.

For more information, please check its website: https://biocontainers.pro/tools/ngsutils and its home page: http://ngsutils.org.

292.2 Versions

- 0.5.9

292.3 Commands

- ngsutils
- bamutils
- bedutils
- fastqutils
- gtfutils

292.4 Module

You can load the modules by:

```
module load biocontainers
module load ngsutils
```

292.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Ngsutils on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=ngsutils
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers ngsutils

bamutils filter \
    input.bam \
    MQ10filtered.bam  \
    -mapped \
    -noqcfail \
    -gte MAPQ 10

bamutils stats \
   -gtf genome.gtf MQ10filtered.bam \
   > MQ10filtered_bamstats
```

CHAPTER

THREE

ORTHOFINDER

293.1 Introduction

`OrthoFinder`: phylogenetic orthology inference for comparative genomics

Detailed usage can be found here: https://github.com/davidemms/OrthoFinder

293.2 Versions

- 2.5.2
- 2.5.4

293.3 Commands

- orthofinder

293.4 Module

You can load the modules by:

```
module load biocontainers
module load orthofinder/2.5.4
```

293.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run orthofinder on our our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 20:00:00
#SBATCH -N 1
#SBATCH -n 24
#SBATCH --job-name=orthofinder
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers orthofinder/2.5.4

orthofinder -t 24 -f InputData -o output
```

CHAPTER

FOUR

PAML

294.1 Introduction

`Paml` is a package of programs for phylogenetic analyses of DNA or protein sequences using maximum likelihood.

For more information, please check its website: https://biocontainers.pro/tools/paml and its home page: http://abacus.gene.ucl.ac.uk/software/paml.html.

294.2 Versions

- 4.9

294.3 Commands

- baseml
- basemlg
- chi2
- codeml
- evolver
- infinitesites
- mcmctree
- pamp
- yn00

294.4 Module

You can load the modules by:

```
module load biocontainers
module load paml
```

294.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Paml on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=paml
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers paml
```

CHAPTER

FIVE

PANACOTA

295.1 Introduction

`Panacota` is a software providing tools for large scale bacterial comparative genomics.

For more information, please check its website: https://biocontainers.pro/tools/panacota and its home page on Github.

295.2 Versions

- 1.3.1

295.3 Commands

- PanACoTA

295.4 Module

You can load the modules by:

```
module load biocontainers
module load panacota
```

295.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Panacota on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=panacota
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers panacota

PanACoTA annotate \
    -d Examples/genomes_init \
    -l Examples/input_files/list_genomes.lst \
    -r Examples/2-res-QC -Q
```

CHAPTER

SIX

PANAROO

296.1 Introduction

Panaroo is an updated pipeline for pangenome investigation.

For more information, please check:
BioContainers: https://biocontainers.pro/tools/panaroo
Home page: https://github.com/gtonkinhill/panaroo

296.2 Versions

- 1.2.10

296.3 Commands

- panaroo
- panaroo-extract-gene
- panaroo-filter-pa
- panaroo-fmg
- panaroo-gene-neighbourhood
- panaroo-img
- panaroo-integrate
- panaroo-merge
- panaroo-msa
- panaroo-plot-abundance
- panaroo-qc
- panaroo-spydrpick

296.4 Module

You can load the modules by:

```
module load biocontainers
module load panaroo
```

296.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run panaroo on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=panaroo
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers panaroo

panaroo -i gff/*.gff -o results --clean-mode strict
```

CHAPTER

SEVEN

PANDASEQ

297.1 Introduction

`Pandaseq` is a program to align Illumina reads, optionally with PCR primers embedded in the sequence, and reconstruct an overlapping sequence.

For more information, please check its | Docker hub: https://hub.docker.com/r/pipecraft/pandaseq and its home page on Github.

297.2 Versions

- 2.11

297.3 Commands

- pandaseq

297.4 Module

You can load the modules by:

```
module load biocontainers
module load pandaseq
```

297.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Pandaseq on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=pandaseq
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers pandaseq

pandaseq -f SRR069027_1.fastq -r SRR069027_2.fastq
```

CHAPTER

EIGHT

PANDORA

298.1 Introduction

Pandora is a tool for bacterial genome analysis using a pangenome reference graph (PanRG). It allows gene presence/absence detection and genotyping of SNPs, indels and longer variants in one or a number of samples.

For more information, please check:
BioContainers: https://biocontainers.pro/tools/pandora
Home page: https://github.com/rmcolq/pandora

298.2 Versions

- 0.9.1

298.3 Commands

- pandora

298.4 Module

You can load the modules by:

```
module load biocontainers
module load pandora
```

298.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run pandora on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 4
#SBATCH --job-name=pandora
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers pandora

pandora index -t 4 GC00006032.fa
```

CHAPTER

NINE

PANGOLIN

299.1 Introduction

`Pangolin` is a software package for assigning SARS-CoV-2 genome sequences to global lineages.

For more information, please check its website: https://biocontainers.pro/tools/pangolin and its home page on Github.

299.2 Versions

- 3.1.20
- 4.0.6
- 4.1.2

299.3 Commands

- pangolin

299.4 Module

You can load the modules by:

```
module load biocontainers
module load pangolin
```

299.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Pangolin on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=pangolin
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers pangolin
```

PANPHLAN

300.1 Introduction

`PanPhlAn` (Pangenome-based Phylogenomic Analysis) is a strain-level metagenomic profiling tool for identifying the gene composition and in-vivo transcriptional activity of individual strains in metagenomic samples.

For more information, please check its home page: http://segatalab.cibio.unitn.it/tools/panphlan/.

300.2 Versions

- 3.1

300.3 Commands

- panphlan_download_pangenome.py
- panphlan_map.py
- panphlan_profiling.py

300.4 Module

You can load the modules by:

```
module load biocontainers
module load panphlan
```

300.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run PanPhlAn on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=panphlan
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers panphlan
```

CHAPTER

ONE

CLARA PARABRICKS

301.1 Introduction

NVIDIA's Clara Parabricks brings next generation sequencing to GPUs, accelerating an array of gold-standard tooling such as BWA-MEM, GATK4, Google's DeepVariant, and many more. Users can achieve a 30-60x acceleration and 99.99% accuracy for variant calling when comparing against CPU-only BWA-GATK4 pipelines, meaning a single server can process up to 60 whole genomes per day. These tools can be easily integrated into current pipelines with drop-in replacement commands to quickly bring speed and data-center scale to a range of applications including germline, somatic and RNA workflows.

For more information, please check:
NGC Container: https://catalog.ngc.nvidia.com/orgs/nvidia/teams/clara/containers/clara-parabricks
Home page: https://docs.nvidia.com/clara/

301.2 Versions

- 4.0.0-1

301.3 Commands

- pbrun

301.4 Module

You can load the modules by:

```
module load biocontainers
module load parabricks
```

301.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

Note: As Clara Parabricks depends on Nvidia GPU, it is only deployed in Scholar, Gilbreth, and ACCESS Anvil.

To run Clara Parabricks on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --gpus=1
#SBATCH --job-name=parabricks
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers parabricks

pbrun haplotypecaller \
  --ref  FVZG01.1.fsa_nt \
  --in-bam output.bam \
  --out-variants variants.vcf
```

CHAPTER

TWO

PARALLEL-FASTQ-DUMP

302.1 Introduction

`Parallel-fastq-dump` is the parallel fastq-dump wrapper.

For more information, please check its website: https://biocontainers.pro/tools/parallel-fastq-dump and its home page on Github.

302.2 Versions

- 0.6.7

302.3 Commands

- parallel-fastq-dump

302.4 Module

You can load the modules by:

```
module load biocontainers
module load parallel-fastq-dump
```

302.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Parallel-fastq-dump on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 4
#SBATCH --job-name=parallel-fastq-dump
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers parallel-fastq-dump

parallel-fastq-dump -s SRR11941281/SRR11941281.sra \
    --split-files --threads 4 --gzip
```

CHAPTER

THREE

PARLIAMENT2

303.1 Introduction

Parliament2 identifies structural variants in a given sample relative to a reference genome. These structural variants cover large deletion events that are called as Deletions of a region, Insertions of a sequence into a region, Duplications of a region, Inversions of a region, or Translocations between two regions in the genome.

For more information, please check:
Docker hub: https://hub.docker.com/r/dnanexus/parliament2
Home page: https://github.com/fritzsedlazeck/parliament2

303.2 Versions

- 0.1.11

303.3 Commands

- parliament2.py

303.4 Module

You can load the modules by:

```
module load biocontainers
module load parliament2
```

303.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run parliament2 on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=parliament2
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers parliament2
```

CHAPTER

FOUR

PARSNP

304.1 Introduction

`Parsnp` is used to align the core genome of hundreds to thousands of bacterial genomes within a few minutes to few hours.

For more information, please check its website: https://biocontainers.pro/tools/parsnp and its home page on Github.

304.2 Versions

- 1.6.2

304.3 Commands

- parsnp

304.4 Module

You can load the modules by:

```
module load biocontainers
module load parsnp
```

304.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Parsnp on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 8
#SBATCH --job-name=parsnp
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers parsnp

parsnp -g examples/mers_virus/ref/England1.gbk \
     -d examples/mers_virus/genomes/*.fna -c -p 8
```

CHAPTER

FIVE

PBMM2

305.1 Introduction

Pbmm2 is a minimap2 frontend for PacBio native data formats.

For more information, please check its website: https://biocontainers.pro/tools/pbmm2 and its home page on Github.

305.2 Versions

- 1.7.0

305.3 Commands

- pbmm2

305.4 Module

You can load the modules by:

```
module load biocontainers
module load pbmm2
```

305.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Pbmm2 on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 12
#SBATCH --job-name=pbmm2
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers pbmm2

pbmm2 --version

pbmm2 align hg38.fa \
    alz.polished.hq.bam alz.aligned.bam \
     -j 12 --preset ISOSEQ --sort \
     --log-level INFO
```

CHAPTER

SIX

PBPTYPER

306.1 Introduction

pbptyper is a tool to identify the Penicillin Binding Protein (PBP) of Streptococcus pneumoniae assemblies.

For more information, please check:
Docker hub: https://hub.docker.com/r/staphb/pbptyper
Home page: https://github.com/rpetit3/pbptyper

306.2 Versions

- 1.0.4

306.3 Commands

- pbptyper

306.4 Module

You can load the modules by:

```
module load biocontainers
module load pbptyper
```

306.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run pbptyper on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=pbptyper
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers pbptyper

pbptyper --assembly test/SRR2912551.fna.gz --outdir output
```

CHAPTER

SEVEN

PCANGSD

307.1 Introduction

`PCAngsd` is a program that estimates the covariance matrix and individual allele frequencies for low-depth next-generation sequencing (NGS) data in structured/heterogeneous populations using principal component analysis (PCA) to perform multiple population genetic analyses using genotype likelihoods.

For more information, please check its home page on Github.

307.2 Versions

- 1.10

307.3 Commands

- pcangsd

307.4 Module

You can load the modules by:

```
module load biocontainers
module load pcangsd
```

307.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run PCAngsd on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 12
#SBATCH --job-name=pcangsd
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers pcangsd

pcangsd -b pupfish.beagle.gz --inbreedSites \
     --selection -o pup_pca2 --threads 12
```

CHAPTER

EIGHT

PEAKRANGER

308.1 Introduction

`Peakranger` is a multi-purpose software suite for analyzing next-generation sequencing (NGS) data.

For more information, please check its website: https://biocontainers.pro/tools/peakranger and its home page: http://ranger.sourceforge.net.

308.2 Versions

- 1.18

308.3 Commands

- peakranger

308.4 Module

You can load the modules by:

```
module load biocontainers
module load peakranger
```

308.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Peakranger on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=peakranger
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers peakranger

peakranger ccat --format bam  27-1_sorted_MDRD_MQ30filtered.bam 27-4_sorted_MDRD_
↪MQ30filtered.bam \
     ccat_result_with_HTML_report_5kb_region --report \
     --gene_annot_file refGene.txt --plot_region 10000
```

CHAPTER

NINE

PEPPER_DEEPVARIANT

309.1 Introduction

PEPPER is a genome inference module based on recurrent neural networks that enables long-read variant calling and nanopore assembly polishing in the PEPPER-Margin-DeepVariant pipeline. This pipeline enables nanopore-based variant calling with DeepVariant.

For more information, please check:
Docker hub: https://hub.docker.com/r/kishwars/pepper_deepvariant
Home page: https://github.com/kishwarshafin/pepper

309.2 Versions

- r0.4.1

309.3 Commands

- run_pepper_margin_deepvariant

309.4 Module

You can load the modules by:

```
module load biocontainers
module load pepper_deepvariant
```

309.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run pepper_deepvariant on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 32
#SBATCH --job-name=pepper_deepvariant
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers pepper_deepvariant

BASE=$PWD

# Set up input data
INPUT_DIR="${BASE}/input/data"
REF="GRCh38_no_alt.chr20.fa"
BAM="HG002_ONT_2_GRCh38.chr20.quickstart.bam"

# Set the number of CPUs to use
THREADS=32

# Set up output directory
OUTPUT_DIR="${BASE}/output"
OUTPUT_PREFIX="HG002_ONT_2_GRCh38_PEPPER_Margin_DeepVariant.chr20"
OUTPUT_VCF="HG002_ONT_2_GRCh38_PEPPER_Margin_DeepVariant.chr20.vcf.gz"
TRUTH_VCF="HG002_GRCh38_1_22_v4.2.1_benchmark.quickstart.vcf.gz"
TRUTH_BED="HG002_GRCh38_1_22_v4.2.1_benchmark_noinconsistent.quickstart.bed"

# Create local directory structure
mkdir -p "${OUTPUT_DIR}"
mkdir -p "${INPUT_DIR}"

# Download the data to input directory
wget -P ${INPUT_DIR} https://storage.googleapis.com/pepper-deepvariant-public/quickstart_
↪data/HG002_ONT_2_GRCh38.chr20.quickstart.bam
wget -P ${INPUT_DIR} https://storage.googleapis.com/pepper-deepvariant-public/quickstart_
↪data/HG002_ONT_2_GRCh38.chr20.quickstart.bam.bai
wget -P ${INPUT_DIR} https://storage.googleapis.com/pepper-deepvariant-public/quickstart_
↪data/GRCh38_no_alt.chr20.fa
wget -P ${INPUT_DIR} https://storage.googleapis.com/pepper-deepvariant-public/quickstart_
↪data/GRCh38_no_alt.chr20.fa.fai
wget -P ${INPUT_DIR} https://storage.googleapis.com/pepper-deepvariant-public/quickstart_
```

(continues on next page)

(continued from previous page)

```
↪data/HG002_GRCh38_1_22_v4.2.1_benchmark.quickstart.vcf.gz
wget -P ${INPUT_DIR} https://storage.googleapis.com/pepper-deepvariant-public/quickstart_
↪data/HG002_GRCh38_1_22_v4.2.1_benchmark_noinconsistent.quickstart.bed

run_pepper_margin_deepvariant call_variant \
    -b input/data/HG002_ONT_2_GRCh38.chr20.quickstart.bam \
    -f input/data/GRCh38_no_alt.chr20.fa -o output \
    -p HG002_ONT_2_GRCh38_PEPPER_Margin_DeepVariant.chr20 \
    -t 32 -r chr20:1000000-1020000 \
    --ont_r9_guppy5_sup --ont
```

BIOPERL

310.1 Introduction

`BioPerl` is a collection of Perl modules that facilitate the development of Perl scripts for bioinformatics applications. It provides software modules for many of the typical tasks of bioinformatics programming.

For more information, please check its website: https://biocontainers.pro/tools/perl-bioperl.

310.2 Versions

- 1.7.2-pl526

310.3 Commands

- SOAPsh.pl
- ace.pl
- bam2bedgraph
- bamToGBrowse.pl
- bdf2gdfont.pl
- bdftogd
- binhex.pl
- bp_aacomp.pl
- bp_biofetch_genbank_proxy.pl
- bp_bioflat_index.pl
- bp_biogetseq.pl
- bp_blast2tree.pl
- bp_bulk_load_gff.pl
- bp_chaos_plot.pl
- bp_classify_hits_kingdom.pl
- bp_composite_LD.pl

- bp_das_server.pl
- bp_dbsplit.pl
- bp_download_query_genbank.pl
- bp_extract_feature_seq.pl
- bp_fast_load_gff.pl
- bp_fastam9_to_table.pl
- bp_fetch.pl
- bp_filter_search.pl
- bp_find-blast-matches.pl
- bp_flanks.pl
- bp_gccalc.pl
- bp_genbank2gff.pl
- bp_genbank2gff3.pl
- bp_generate_histogram.pl
- bp_heterogeneity_test.pl
- bp_hivq.pl
- bp_hmmer_to_table.pl
- bp_index.pl
- bp_load_gff.pl
- bp_local_taxonomydb_query.pl
- bp_make_mrna_protein.pl
- bp_mask_by_search.pl
- bp_meta_gff.pl
- bp_mrtrans.pl
- bp_mutate.pl
- bp_netinstall.pl
- bp_nexus2nh.pl
- bp_nrdb.pl
- bp_oligo_count.pl
- bp_pairwise_kaks
- bp_parse_hmmsearch.pl
- bp_process_gadfly.pl
- bp_process_sgd.pl
- bp_process_wormbase.pl
- bp_query_entrez_taxa.pl
- bp_remote_blast.pl

- bp_revtrans-motif.pl
- bp_search2alnblocks.pl
- bp_search2gff.pl
- bp_search2table.pl
- bp_search2tribe.pl
- bp_seq_length.pl
- bp_seqconvert.pl
- bp_seqcut.pl
- bp_seqfeature_delete.pl
- bp_seqfeature_gff3.pl
- bp_seqfeature_load.pl
- bp_seqpart.pl
- bp_seqret.pl
- bp_seqretsplit.pl
- bp_split_seq.pl
- bp_sreformat.pl
- bp_taxid4species.pl
- bp_taxonomy2tree.pl
- bp_translate_seq.pl
- bp_tree2pag.pl
- bp_unflatten_seq.pl
- ccconfig
- chartex
- chi2
- chrom_sizes.pl
- circo
- clustalw
- clustalw2
- corelist
- cpan
- cpanm
- dbilogstrip
- dbiprof
- dbiproxy
- debinhex.pl
- enc2xs

- encguess
- genomeCoverageBed.pl
- h2ph
- h2xs
- htmltree
- instmodsh
- json_pp
- json_xs
- lwp-download
- lwp-dump
- lwp-mirror
- lwp-request
- perl
- perl5.26.2
- perlbug
- perldoc
- perlivp
- perlthanks
- piconv
- pl2pm
- pod2html
- pod2man
- pod2text
- pod2usage
- podchecker
- podselect
- prove
- ptar
- ptardiff
- ptargrep
- shasum
- splain
- stag-autoschema.pl
- stag-db.pl
- stag-diff.pl
- stag-drawtree.pl

- stag-filter.pl
- stag-findsubtree.pl
- stag-flatten.pl
- stag-grep.pl
- stag-handle.pl
- stag-itext2simple.pl
- stag-itext2sxpr.pl
- stag-itext2xml.pl
- stag-join.pl
- stag-merge.pl
- stag-mogrify.pl
- stag-parse.pl
- stag-query.pl
- stag-splitter.pl
- stag-view.pl
- stag-xml2itext.pl
- stubmaker.pl
- t_coffee
- tpage
- ttree
- unflatten
- webtidy
- xml_grep
- xml_merge
- xml_pp
- xml_spellcheck
- xml_split
- xpath
- xsubpp
- zipdetails

310.4 Module

You can load the modules by:

```
module load biocontainers
module load perl-bioperl
```

310.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run BioPerl on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=perl-bioperl
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers perl-bioperl
```

CHAPTER

ONE

PHD2FASTA

311.1 Introduction

`Phd2fasta` is a tool to convert Phred 'phd' format files to 'fasta' format.

For more information, please check its home page: http://www.phrap.org/phredphrapconsed.html.

311.2 Versions

- 0.990622

311.3 Commands

- phd2fasta

311.4 Module

You can load the modules by:

```
module load biocontainers
module load phd2fasta
```

311.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Phd2fasta on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=phd2fasta
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers phd2fasta
```

CHAPTER

TWO

PHG

312.1 Introduction

Practical Haplotype Graph (PHG) is a general, graph-based, computational framework that can be used with a variety of skim sequencing methods to infer high-density genotypes directly from low-coverage sequence.

For more information, please check:
Docker hub: https://hub.docker.com/r/maizegenetics/phg
Home page: https://www.maizegenetics.net/phg

312.2 Versions

- 1.0

312.3 Commands

- CreateConsensi.sh
- CreateHaplotypes.sh
- CreateReferenceIntervals.sh
- CreateSmallDataSet.sh
- CreateValidIntervalsFile.sh
- IndexPangenome.sh
- LoadAssemblyAnchors.sh
- LoadGenomeIntervals.sh
- ParallelAssemblyAnchorsLoad.sh
- RunLiquibaseUpdates.sh
- CreateHaplotypesFromBAM.groovy
- CreateHaplotypesFromFastq.groovy
- CreateHaplotypesFromGVCF.groovy

312.4 Module

You can load the modules by:

```
module load biocontainers
module load phg
```

312.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run phg on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=phg
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers phg
```

CHAPTER

THREE

PHRAP

313.1 Introduction

`phrap` is a program for assembling shotgun DNA sequence data.

For more information, please check its home page: http://www.phrap.org/phredphrapconsed.html#block_phrap.

313.2 Versions

- 1.090518

313.3 Commands

- phrap

313.4 Module

You can load the modules by:

```
module load biocontainers
module load phrap
```

313.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run phrap on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=phrap
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers phrap
```

CHAPTER

FOUR

PHRED

314.1 Introduction

`phred` software reads DNA sequencing trace files, calls bases, and assigns a quality value to each called base.

For more information, please check its home page: http://www.phrap.org/phredphrapconsed.html#block_phred.

314.2 Versions

- 0.071220.c

314.3 Commands

- phred

314.4 Module

You can load the modules by:

```
module load biocontainers
module load phred
```

314.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run phred on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=phred
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers phred
```

CHAPTER

FIVE

PICARD TOOLS

315.1 Introduction

`Picard` is a set of command line tools for manipulating high-throughput sequencing (HTS) data and formats such as SAM/BAM/CRAM and VCF. Detailed usage can be found here: https://broadinstitute.github.io/picard/

315.2 Versions

- 2.25.1
- 2.26.10

315.3 Commands

picard

315.4 Module

You can load the modules by:

```
module load biocontainers
module load picard/2.26.10
```

315.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run picard our our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 20:00:00
#SBATCH -N 1
#SBATCH -n 24
#SBATCH --job-name=picard
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers picard/2.26.10

picard MarkDuplicates -Xmx64g I=19P0126636WES_sorted.bam O=19P0126636WES_sorted_md.bam␣
↪M=19P0126636WES.sorted.markdup.txt REMOVE_DUPLICATES=true
picard BuildBamIndex -Xmx64g I=19P0126636WES_sorted_md.bam
picard CreateSequenceDictionary -R hg38.fa -O hg38.dict
```

CHAPTER

SIX

PICRUST2

316.1 Introduction

`Picrust2` is a software for predicting functional abundances based only on marker gene sequences.

For more information, please check its website: https://biocontainers.pro/tools/picrust2 and its home page on Github.

316.2 Versions

- 2.4.2
- 2.5.0

316.3 Commands

- add_descriptions.py
- convert_table.py
- hsp.py
- metagenome_pipeline.py
- pathway_pipeline.py
- picrust2_pipeline.py
- place_seqs.py
- print_picrust2_config.py
- run_abundance.py
- run_sepp.py
- run_tipp.py
- run_tipp_tool.py
- run_upp.py
- shuffle_predictions.py
- split_sequences.py

- sumlabels.py
- sumtrees.py

316.4 Module

You can load the modules by:

```
module load biocontainers
module load picrust2
```

316.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Picrust2 on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 10
#SBATCH --job-name=picrust2
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers picrust2


place_seqs.py -s ../seqs.fna -o out.tre -p 10 \
          --intermediate intermediate/place_seqs

hsp.py -i 16S -t out.tre -o marker_predicted_and_nsti.tsv.gz -p 10 -n

hsp.py -i EC -t out.tre -o EC_predicted.tsv.gz -p 10

metagenome_pipeline.py -i ../table.biom -m marker_predicted_and_nsti.tsv.gz -f EC_
↪predicted.tsv.gz -o EC_metagenome_out --strat_out

convert_table.py EC_metagenome_out/pred_metagenome_contrib.tsv.gz \
             -c contrib_to_legacy \
             -o EC_metagenome_out/pred_metagenome_contrib.legacy.tsv.gz

pathway_pipeline.py -i EC_metagenome_out/pred_metagenome_contrib.tsv.gz \
                -o pathways_out -p 10
```

(continues on next page)

(continued from previous page)

```
add_descriptions.py -i EC_metagenome_out/pred_metagenome_unstrat.tsv.gz -m EC \
                -o EC_metagenome_out/pred_metagenome_unstrat_descrip.tsv.gz


add_descriptions.py -i pathways_out/path_abun_unstrat.tsv.gz -m METACYC \
                -o pathways_out/path_abun_unstrat_descrip.tsv.gz

picrust2_pipeline.py -s chemerin_16S/seqs.fna -i chemerin_16S/table.biom \
    -o picrust2_out_pipeline -p 10
```

CHAPTER

SEVEN

PILON

317.1 Introduction

`Pilon` is an automated genome assembly improvement and variant detection tool.

For more information, please check its website: https://biocontainers.pro/tools/pilon and its home page on Github.

317.2 Versions

- 1.24

317.3 Commands

- pilon.jar

317.4 Module

You can load the modules by:

```
module load biocontainers
module load pilon
```

317.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Pilon on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 12
#SBATCH --job-name=pilon
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers pilon

pilon.jar --nostrays \
     --genome scaffolds.fasta \
     --frags out_sorted.bam \
     --vcf --verbose --threads 12 \
     --output pilon_corrected \
     --outdir pilon_outdir
```

CHAPTER

EIGHT

PINDEL

318.1 Introduction

`Pindel` is used to detect breakpoints of large deletions, medium sized insertions, inversions, tandem duplications and other structural variants at single-based resolution from next-gen sequence data.

For more information, please check its website: https://biocontainers.pro/tools/pindel and its home page: http://gmt.genome.wustl.edu/packages/pindel/index.html.

318.2 Versions

- 0.2.5b9

318.3 Commands

- pindel
- pindel2cvf

318.4 Module

You can load the modules by:

```
module load biocontainers
module load pindel
```

318.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Pindel on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=pindel
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers pindel

pindel -i simulated_config.txt -f simulated_reference.fa -o bamtest -c ALL

pindel -p COLO-829_20-p_ok.txt -f hs_ref_chr20.fa -o colontumor -c 20

pindel2vcf -r hs_ref_chr20.fa -R HUMAN_G1K_V2 -d 20100101 -p colontumor_D -e 5
```

CHAPTER

NINE

PIRATE

319.1 Introduction

`Pirate` is a pangenome analysis and threshold evaluation toolbox.

For more information, please check its website: https://biocontainers.pro/tools/pirate and its home page on Github.

319.2 Versions

- 1.0.4

319.3 Commands

- PIRATE
- FET.pl
- PIRATE_to_Rtab.pl
- PIRATE_to_roary.pl
- SOAPsh.pl
- ace.pl
- analyse_blast_outputs.pl
- analyse_loci_list.pl
- annotate_treeWAS_output.pl
- bamToGBrowse.pl
- bdf2gdfont.pl
- binhex.pl
- bp_aacomp.pl
- bp_biofetch_genbank_proxy.pl
- bp_bioflat_index.pl
- bp_biogetseq.pl

- bp_blast2tree.pl
- bp_bulk_load_gff.pl
- bp_chaos_plot.pl
- bp_classify_hits_kingdom.pl
- bp_composite_LD.pl
- bp_das_server.pl
- bp_dbsplit.pl
- bp_download_query_genbank.pl
- bp_extract_feature_seq.pl
- bp_fast_load_gff.pl
- bp_fastam9_to_table.pl
- bp_fetch.pl
- bp_filter_search.pl
- bp_find-blast-matches.pl
- bp_flanks.pl
- bp_gccalc.pl
- bp_genbank2gff.pl
- bp_genbank2gff3.pl
- bp_generate_histogram.pl
- bp_heterogeneity_test.pl
- bp_hivq.pl
- bp_hmmer_to_table.pl
- bp_index.pl
- bp_load_gff.pl
- bp_local_taxonomydb_query.pl
- bp_make_mrna_protein.pl
- bp_mask_by_search.pl
- bp_meta_gff.pl
- bp_mrtrans.pl
- bp_mutate.pl
- bp_netinstall.pl
- bp_nexus2nh.pl
- bp_nrdb.pl
- bp_oligo_count.pl
- bp_parse_hmmsearch.pl
- bp_process_gadfly.pl

- bp_process_sgd.pl
- bp_process_wormbase.pl
- bp_query_entrez_taxa.pl
- bp_remote_blast.pl
- bp_revtrans-motif.pl
- bp_search2alnblocks.pl
- bp_search2gff.pl
- bp_search2table.pl
- bp_search2tribe.pl
- bp_seq_length.pl
- bp_seqconvert.pl
- bp_seqcut.pl
- bp_seqfeature_delete.pl
- bp_seqfeature_gff3.pl
- bp_seqfeature_load.pl
- bp_seqpart.pl
- bp_seqret.pl
- bp_seqretsplit.pl
- bp_split_seq.pl
- bp_sreformat.pl
- bp_taxid4species.pl
- bp_taxonomy2tree.pl
- bp_translate_seq.pl
- bp_tree2pag.pl
- bp_unflatten_seq.pl
- cd-hit-2d-para.pl
- cd-hit-clstr_2_blm8.pl
- cd-hit-div.pl
- cd-hit-para.pl
- chrom_sizes.pl
- clstr2tree.pl
- clstr2txt.pl
- clstr2xml.pl
- clstr_cut.pl
- clstr_list.pl
- clstr_list_sort.pl

- clstr_merge.pl
- clstr_merge_noorder.pl
- clstr_quality_eval.pl
- clstr_quality_eval_by_link.pl
- clstr_reduce.pl
- clstr_renumber.pl
- clstr_rep.pl
- clstr_reps_faa_rev.pl
- clstr_rev.pl
- clstr_select.pl
- clstr_select_rep.pl
- clstr_size_histogram.pl
- clstr_size_stat.pl
- clstr_sort_by.pl
- clstr_sort_prot_by.pl
- clstr_sql_tbl.pl
- clstr_sql_tbl_sort.pl
- convert_to_distmat.pl
- convert_to_treeWAS.pl
- debinhex.pl
- genomeCoverageBed.pl
- legacy_blast.pl
- make_multi_seq.pl
- pangenome_variants_to_treeWAS.pl
- paralogs_to_Rtab.pl
- plot_2d.pl
- plot_len1.pl
- stag-autoschema.pl
- stag-db.pl
- stag-diff.pl
- stag-drawtree.pl
- stag-filter.pl
- stag-findsubtree.pl
- stag-flatten.pl
- stag-grep.pl
- stag-handle.pl

- stag-itext2simple.pl
- stag-itext2sxpr.pl
- stag-itext2xml.pl
- stag-join.pl
- stag-merge.pl
- stag-mogrify.pl
- stag-parse.pl
- stag-query.pl
- stag-splitter.pl
- stag-view.pl
- stag-xml2itext.pl
- stubmaker.pl
- subsample_outputs.pl
- subset_alignments.pl
- unique_sequences.pl
- update_blastdb.pl

319.4 Module

You can load the modules by:

```
module load biocontainers
module load pirate
```

319.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Pirate on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=pirate
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out
```

(continues on next page)

(continued from previous page)

```
module --force purge
ml biocontainers pirate
```

CHAPTER

PIXY

320.1 Introduction

pixy is a command-line tool for painlessly estimating average nucleotide diversity within () and between (dxy) populations from a VCF.

For more information, please check:
Home page: https://github.com/ksamuk/pixy

320.2 Versions

- 1.2.7

320.3 Commands

- pixy

320.4 Module

You can load the modules by:

```
module load biocontainers
module load pixy
```

320.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run pixy on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=pixy
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers pixy
```

CHAPTER

ONE

PLASMIDFINDER

321.1 Introduction

PlasmidFinder identifies plasmids in total or partial sequenced isolates of bacteria.

For more information, please check:
Docker hub: https://hub.docker.com/r/staphb/plasmidfinder
Home page: https://bitbucket.org/genomicepidemiology/plasmidfinder/src/master/

321.2 Versions

- 2.1.6

321.3 Commands

- plasmidfinder.py

321.4 Module

You can load the modules by:

```
module load biocontainers
module load plasmidfinder
```

321.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run plasmidfinder on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=plasmidfinder
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers plasmidfinder

plasmidfinder.py -p test/database \
    -i test/test.fsa -o output -mp blastn -x -q
```

CHAPTER

TWO

PLATYPUS

322.1 Introduction

`Platypus` is a tool designed for efficient and accurate variant-detection in high-throughput sequencing data.

For more information, please check its website: https://biocontainers.pro/tools/platypus and its home page: https://www.well.ox.ac.uk/research/research-groups/lunter-group/lunter-group/platypus-a-haplotype-based-variant-caller-for-next-generation-sequence-data.

322.2 Versions

- 0.8.1

322.3 Commands

- platypus

322.4 Module

You can load the modules by:

```
module load biocontainers
module load platypus
```

322.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Platypus on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=platypus
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers platypus
```

CHAPTER

THREE

PLINK

323.1 Introduction

`Plink` is a free, open-source whole genome association analysis toolset, designed to perform a range of basic, large-scale analyses in a computationally efficient manner.

For more information, please check its website: https://biocontainers.pro/tools/plink and its home page: https://zzz.bwh.harvard.edu/plink/.

323.2 Versions

- 1.90b6.21

323.3 Commands

- plink
- prettify

323.4 Module

You can load the modules by:

```
module load biocontainers
module load plink
```

323.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Plink on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=plink
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers plink

plink --file toy --freq --out toy_analysis
```

CHAPTER

FOUR

PLINK2

324.1 Introduction

`Plink2` is a whole genome association analysis toolset.

For more information, please check its website: https://biocontainers.pro/tools/plink2 and its home page on Github.

324.2 Versions

- 2.00a2.3

324.3 Commands

- plink2

324.4 Module

You can load the modules by:

```
module load biocontainers
module load plink2
```

324.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Plink2 on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=plink2
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers plink2

plink2 --bfile HapMap_3_r3_1 --freq --out HapMap_3_r3_1_out
```

CHAPTER

FIVE

PLOTSR

325.1 Introduction

Plotsr generates high-quality visualisation of synteny and structural rearrangements between multiple genomes. For this, it uses the genomic structural annotations between multiple chromosome-level assemblies.

For more information, please check:
Home page: https://github.com/schneebergerlab/plotsr

325.2 Versions

- 0.5.4

325.3 Commands

- plotsr

325.4 Module

You can load the modules by:

```
module load biocontainers
module load plotsr
```

325.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run plotsr on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=plotsr
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers plotsr

plotsr syri.out refgenome qrygenome -H 8 -W 5
```

CHAPTER

SIX

POMOXIS

326.1 Introduction

Pomoxis comprises a set of basic bioinformatic tools tailored to nanopore sequencing. Notably tools are included for generating and analysing draft assemblies. Many of these tools are used by the research data analysis group at Oxford Nanopore Technologies.

For more information, please check:
Docker hub: https://hub.docker.com/r/zeunas/pomoxis
Home page: https://github.com/nanoporetech/pomoxis

326.2 Versions

- 0.3.9

326.3 Commands

- assess_assembly
- catalogue_errors
- common_errors_from_bam
- coverage_from_bam
- coverage_from_fastx
- fast_convert
- find_indels
- intersect_assembly_errors
- long_fastx
- mini_align
- mini_assemble
- pomoxis_path
- qscores_from_summary

- ref_seqs_from_bam
- reverse_bed
- split_fastx
- stats_from_bam
- subsample_bam
- summary_from_stats
- tag_bam
- trim_alignments

326.4 Module

You can load the modules by:

```
module load biocontainers
module load pomoxis
```

326.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run pomoxis on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 4
#SBATCH --job-name=pomoxis
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers pomoxis

assess_assembly \
    -i helen_output/Staph_Aur_draft_helen.fa \
    -r truth_assembly_staph_aur.fasta \
    -p polished_assembly_quality \
    -l 50 \
    -t 4 \
    -e \
    -T
```

CHAPTER

SEVEN

POPSCLE

327.1 Introduction

`Popscle` is a suite of population scale analysis tools for single-cell genomics data.

For more information, please check its | Docker hub: https://hub.docker.com/r/cumulusprod/popscle and its home page on Github.

327.2 Versions

- 0.1b

327.3 Commands

- popscle

327.4 Module

You can load the modules by:

```
module load biocontainers
module load popscle
```

327.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Popscle on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=popscle
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers popscle

popscle dsc-pileup --sam data/$bam --vcf data/$ref_vcf --out data/$pileup
```

CHAPTER

EIGHT

PRINSEQ

328.1 Introduction

`Prinseq` is a tool that generates summary statistics of sequence and quality data and that is used to filter, reformat and trim next-generation sequence data.

For more information, please check its website: https://biocontainers.pro/tools/prinseq and its home page: http://prinseq.sourceforge.net.

328.2 Versions

- 0.20.4

328.3 Commands

- prinseq-graphs-noPCA.pl
- prinseq-graphs.pl
- prinseq-lite.pl

328.4 Module

You can load the modules by:

```
module load biocontainers
module load prinseq
```

328.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Prinseq on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=prinseq
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers prinseq

prinseq-lite.pl -verbose -fastq  SRR5043021_1.fastq -fastq2 SRR5043021_2.fastq -graph_
↪data test.gd -out_good null -out_bad null
prinseq-graphs.pl -i test.gd -png_all -o test
prinseq-graphs-noPCA.pl -i test.gd -png_all -o test_noPCA
```

CHAPTER

NINE

PRODIGAL

329.1 Introduction

`Prodigal` is a tool for fast, reliable protein-coding gene prediction for prokaryotic genome.

For more information, please check its website: https://biocontainers.pro/tools/prodigal and its home page on Github.

329.2 Versions

- 2.6.3

329.3 Commands

- prodigal

329.4 Module

You can load the modules by:

```
module load biocontainers
module load prodigal
```

329.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Prodigal on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=prodigal
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers prodigal

prodigal -i genome.fasta -o output.genes -a proteins.faa
```

PROKKA

330.1 Introduction

`Prokka` is a pipeline for rapidly annotating prokaryotic genomes. It produces GFF3, GBK and SQN files that are ready for editing in Sequin and ultimately submitted to Genbank/DDJB/ENA.

Detailed usage can be found here: https://github.com/tseemann/prokka

330.2 Versions

- 1.14.6

330.3 Commands

- prokka
- prokka-abricate_to_fasta_db
- prokka-biocyc_to_fasta_db
- prokka-build_kingdom_dbs
- prokka-cdd_to_hmm
- prokka-clusters_to_hmm
- prokka-genbank_to_fasta_db
- prokka-genpept_to_fasta_db
- prokka-hamap_to_hmm
- prokka-tigrfams_to_hmm
- prokka-uniprot_to_fasta_db

330.4 Module

You can load the modules by:

```
module load biocontainers
module load prokka
```

330.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run prokka on our our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 20:00:00
#SBATCH -N 1
#SBATCH -n 24
#SBATCH --job-name=prokka
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers prokka

prokka --compliant --centre UoN --outdir PRJEB12345 --locustag EHEC --prefix EHEC-Chr1␣
↪contigs.fa  --cpus 24
prokka-genbank_to_fasta_db Coccus1.gbk Coccus2.gbk Coccus3.gbk Coccus4.gbk > Coccus.faa
```

CHAPTER

ONE

PROTEINORTHO

331.1 Introduction

`Proteinortho` is a tool to detect orthologous genes within different species.

For more information, please check its website: https://biocontainers.pro/tools/proteinortho and its home page on Gitlab.

331.2 Versions

- 6.0.33

331.3 Commands

- proteinortho
- proteinortho2html.pl
- proteinortho2tree.pl
- proteinortho2xml.pl
- proteinortho6.pl
- proteinortho_cleanupblastgraph
- proteinortho_clustering
- proteinortho_compareProteinorthoGraphs.pl
- proteinortho_do_mcl.pl
- proteinortho_extract_from_graph.pl
- proteinortho_ffadj_mcs.py
- proteinortho_formatUsearch.pl
- proteinortho_grab_proteins.pl
- proteinortho_graphMinusRemovegraph
- proteinortho_history.pl

- proteinortho_singletons.pl
- proteinortho_summary.pl
- proteinortho_treeBuilderCore

331.4 Module

You can load the modules by:

```
module load biocontainers
module load proteinortho
```

331.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Proteinortho on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=proteinortho
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers proteinortho

proteinortho6.pl test/C.faa test/E.faa test/L.faa test/M.faa
```

CHAPTER

TWO

PROTHINT

332.1 Introduction

`ProtHint` is a pipeline for predicting and scoring hints (in the form of introns, start and stop codons) in the genome of interest by mapping and spliced aligning predicted genes to a database of reference protein sequences.

332.2 Versions

- 2.6.0

332.3 Commands

- cds_with_upstream_support.py
- combine_gff_records.pl
- count_cds_overlaps.py
- flag_top_proteins.py
- gff_from_region_to_contig.pl
- make_chains.py
- nucseq_for_selected_genes.pl
- print_high_confidence.py
- print_longest_isoform.py
- proteins_from_gtf.pl
- prothint.py
- prothint2augustus.py
- run_spliced_alignment.pl
- run_spliced_alignment_pbs.pl
- select_best_proteins.py
- select_for_next_iteration.py
- spalnBatch.sh

- spaln_to_gff.py

332.4 Academic license

ProtHint depends on GenMark. To use GeneMark, users need to download license files by yourself.

Go to the GeneMark web site: http://exon.gatech.edu/GeneMark/license_download.cgi. Check the boxes for `GeneMark-ES/ET/EP ver 4.69_lic` and `LINUX 64` next to it, fill out the form, then click "I agree". In the next page, right click and copy the link addresses for `64 bit` licenss. Paste the link addresses in the commands below:

```
cd $HOME
wget "replace with license URL"
zcat gm_key_64.gz > .gm_key
```

332.5 Module

You can load the modules by:

```
module load biocontainers
module load prothint
```

332.6 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run ProtHint on our cluster:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 4
#SBATCH --job-name=prothint
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers prothint

prothint.py --threads 4 input/genome.fasta input/proteins.fasta --geneSeeds input/
↪genemark.gtf --workdir test
```

CHAPTER

THREE

PULLSEQ

333.1 Introduction

Pullseq is an utility program for extracting sequences from a fasta/fastq file.

For more information, please check:
BioContainers: https://biocontainers.pro/tools/pullseq
Home page: https://github.com/bcthomas/pullseq

333.2 Versions

- 1.0.2

333.3 Commands

- pcre-config
- pcregrep
- pcretest
- pullseq
- seqdiff

333.4 Module

You can load the modules by:

```
module load biocontainers
module load pullseq
```

333.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run pullseq on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=pullseq
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers pullseq
```

CHAPTER

FOUR

PVACTOOLS

334.1 Introduction

pVACtools is a cancer immunotherapy tools suite consisting of pVACseq, pVACbind, pVACfuse, pVACvector, and pVACview.

For more information, please check:
Docker hub: https://hub.docker.com/r/griffithlab/pvactools/
Home page: https://pvactools.readthedocs.io/en/latest/

334.2 Versions

- 3.0.1

334.3 Commands

- pvacbind
- pvacfuse
- pvacseq
- pvactools
- pvacvector
- pvacview

334.4 Module

You can load the modules by:

```
module load biocontainers
module load pvactools
```

334.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run pvactools on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=pvactools
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers pvactools

pvacseq download_example_data .

pvacseq run \
  pvacseq_example_data/input.vcf \
  Test \
  HLA-A*02:01,HLA-B*35:01,DRB1*11:01 \
  MHCflurry MHCnuggetsI MHCnuggetsII NNalign NetMHC PickPocket SMM SMMPMBEC SMMalign \
  pvacseq_output_data \
  -e1 8,9,10 \
  -e2 15 \
  --iedb-install-directory /opt/iedb
```

CHAPTER

FIVE

PYANI

335.1 Introduction

`Pyani` is an application and Python module for whole-genome classification of microbes using Average Nucleotide Identity.

For more information, please check its website: https://biocontainers.pro/tools/pyani and its home page on Github.

335.2 Versions

- 0.2.11

335.3 Commands

- average_nucleotide_identity.py
- genbank_get_genomes_by_taxon.py
- delta_filter_wrapper.py

335.4 Module

You can load the modules by:

```
module load biocontainers
module load pyani
```

335.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Pyani on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=pyani
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers pyani

average_nucleotide_identity.py -i tests/ -o tests/test_ANIm_output -m ANIm -g
average_nucleotide_identity.py -i tests/  -o tests/test_ANIb_output -m ANIb -g
average_nucleotide_identity.py -i tests/ -o tests/test_ANIblastall_output -m ANIblastall␣
↪-g
average_nucleotide_identity.py -i tests/  -o tests/test_TETRA_output -m TETRA -g
```

CHAPTER

SIX

PYBEDTOOLS

336.1 Introduction

`Pybedtools` wraps and extends BEDTools and offers feature-level manipulations from within Python.

For more information, please check its website: https://biocontainers.pro/tools/pybedtools and its home page on Github.

336.2 Versions

- 0.9.0-py37

336.3 Commands

- python
- python3

336.4 Module

You can load the modules by:

```
module load biocontainers
module load pybedtools
```

336.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Pybedtools on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=pybedtools
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers pybedtools
```

CHAPTER

SEVEN

PYBIGWIG

337.1 Introduction

`Pybigwig` is a python extension, written in C, for quick access to bigBed files and access to and creation of bigWig files.

For more information, please check its website: https://biocontainers.pro/tools/pybigwig and its home page on Github.

337.2 Versions

- 0.3.18-py36

337.3 Commands

- python
- python3

337.4 Module

You can load the modules by:

```
module load biocontainers
module load pybigwig
```

337.5 Interactive job

To run pybigwig interactively on our clusters:

```
(base) UserID@bell-fe00:~ $ sinteractive -N1 -n12 -t4:00:00 -A myallocation
salloc: Granted job allocation 12345869
salloc: Waiting for resource configuration
salloc: Nodes bell-a008 are ready for job
(base) UserID@bell-a008:~ $ module load biocontainers pybigwig
(base) UserID@bell-a008:~ $ python
Python 3.6.15 |  packaged by conda-forge |  (default, Dec  3 2021, 18:49:41)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import pyBigWig
>>> bw = pyBigWig.open("test/test.bw")
```

337.6 Batch job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run batch jobs on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=pybigwig
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers pybigwig

python script.py
```

CHAPTER

EIGHT

PYCHOPPER

338.1 Introduction

Pychopper is a tool to identify, orient and trim full-length Nanopore cDNA reads. The tool is also able to rescue fused reads.

For more information, please check:
BioContainers: https://biocontainers.pro/tools/pychopper
Home page: https://github.com/nanoporetech/pychopper

338.2 Versions

- 2.5.0

338.3 Commands

- cdna_classifier.py

338.4 Module

You can load the modules by:

```
module load biocontainers
module load pychopper
```

338.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run pychopper on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=pychopper
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers pychopper
```

CHAPTER

NINE

PYCOQC

339.1 Introduction

Pycoqc is a tool that computes metrics and generates interactive QC plots for Oxford Nanopore technologies sequencing data.

For more information, please check its website: https://biocontainers.pro/tools/pycoqc and its home page on Github.

339.2 Versions

- 2.5.2

339.3 Commands

- pycoQC
- python
- python3

339.4 Module

You can load the modules by:

```
module load biocontainers
module load pycoqc
```

339.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Pycoqc on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=pycoqc
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers pycoqc

pycoQC \
    -f Albacore-1.2.1_basecall-1D-DNA_sequencing_summary.txt\
     -o Albacore-1.2.1_basecall-1D-DNA.html \
    --quiet
```

PYENSEMBL

340.1 Introduction

`Pyensembl` is a Python interface to Ensembl reference genome metadata such as exons and transcripts.

For more information, please check its website: https://biocontainers.pro/tools/pyensembl and its home page on Github.

340.2 Versions

- 1.9.4

340.3 Commands

- pyensembl
- python
- python3

340.4 Module

You can load the modules by:

```
module load biocontainers
module load pyensembl
```

340.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Pyensembl on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=pyensembl
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers pyensembl
```

CHAPTER

ONE

PYFAIDX

341.1 Introduction

`Pyfaidx` is a Python package for random access and indexing of fasta files.

For more information, please check its website: https://biocontainers.pro/tools/pyfaidx and its home page on Github.

341.2 Versions

- 0.6.4

341.3 Commands

- python
- python3

341.4 Module

You can load the modules by:

```
module load biocontainers
module load pyfaidx
```

341.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Pyfaidx on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=pyfaidx
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers pyfaidx
```

CHAPTER

TWO

PYGENOMETRACKS

342.1 Introduction

pyGenomeTracks aims to produce high-quality genome browser tracks that are highly customizable.

For more information, please check:
BioContainers: https://biocontainers.pro/tools/pygenometracks
Home page: https://github.com/deeptools/pyGenomeTracks

342.2 Versions

- 3.7

342.3 Commands

- make_tracks_file
- pyGenomeTracks

342.4 Module

You can load the modules by:

```
module load biocontainers
module load pygenometracks
```

342.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run pygenometracks on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=pygenometracks
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers pygenometracks

make_tracks_file --trackFiles domains.bed bigwig.bw -o tracks.ini

pyGenomeTracks --tracks tracks.ini \
   --region chr2:10,000,000-11,000,000 --outFileName nice_image.pdf
```

CHAPTER

THREE

PYGENOMEVIZ

343.1 Introduction

pyGenomeViz is a genome visualization python package for comparative genomics implemented based on matplotlib.

For more information, please check:
Docker hub: https://hub.docker.com/r/staphb/pygenomeviz
Home page: https://github.com/moshi4/pyGenomeViz#cli-examples

343.2 Versions

- 0.2.2

343.3 Commands

- pgv-download-dataset
- pgv-mmseqs
- pgv-mummer
- pgv-pmauve
- python
- python3

343.4 Module

You can load the modules by:

```
module load biocontainers
module load pygenomeviz
```

343.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run pygenomeviz on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=pygenomeviz
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers pygenomeviz
```

CHAPTER

FOUR

PYRANGES

344.1 Introduction

`Pyranges` are collections of intervals that support comparison operations (like overlap and intersect) and other methods that are useful for genomic analyses.

For more information, please check its website: https://biocontainers.pro/tools/pyranges and its home page on Github.

344.2 Versions

- 0.0.115

344.3 Commands

- python
- python3

344.4 Module

You can load the modules by:

```
module load biocontainers
module load pyranges
```

344.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Pyranges on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=pyranges
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers pyranges
```

CHAPTER

FIVE

PYSAM

345.1 Introduction

`Pysam` is a python module that makes it easy to read and manipulate mapped short read sequence data stored in SAM/BAM files.

For more information, please check its website: https://biocontainers.pro/tools/pysam and its home page on Github.

345.2 Versions

- 0.18.0-py37

345.3 Commands

- python
- python3

345.4 Module

You can load the modules by:

```
module load biocontainers
module load pysam
```

345.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Pysam on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=pysam
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers pysam
```

CHAPTER

SIX

QIIME 2

346.1 Introduction

`QIIME 2` is a is a powerful, extensible, and decentralized microbiome analysis package with a focus on data and analysis transparency. QIIME 2 enables researchers to start an analysis with raw DNA sequence data and finish with publication-quality figures and statistical results.

For more information, please check its website: https://quay.io/repository/qiime2/core and its home page: https://qiime2.org/.

346.2 Versions

- 2021.2
- 2022.2
- 2022.8

346.3 Commands

- qiime
- python
- python3

346.4 Module

You can load the modules by:

```
module load biocontainers
module load qiime2
```

346.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run QIIME 2 on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=qiime2
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers qiime2

qiime metadata tabulate \
    --m-input-file rep-seqs.qza \
    --m-input-file taxonomy.qza \
    --o-visualization tabulated-feature-metadata.qzv
```

CHAPTER

SEVEN

QUALIMAP

347.1 Introduction

`Qualimap` is a platform-independent application written in Java and R that provides both a Graphical User Inteface (GUI) and a command-line interface to facilitate the quality control of alignment sequencing data and its derivatives like feature counts.

For more information, please check its website: https://biocontainers.pro/tools/qualimap and its home page: http://qualimap.conesalab.org.

347.2 Versions

- 2.2.1

347.3 Commands

- qualimap

347.4 Module

You can load the modules by:

```
module load biocontainers
module load qualimap
```

347.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Qualimap on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=qualimap
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers qualimap
```

CHAPTER

EIGHT

QUAST

348.1 Introduction

`Quast` is Quality Assessment Tool for Genome Assemblies.

Note: Running QUAST, please use the command: quast.py| metaquast.py fastafile [OTHER OPTIONS] DO NOT call it 'python quast.py| metaquast.py'

For more information, please check its website: https://biocontainers.pro/tools/quast and its home page on Github.

348.2 Versions

- 5.0.2-py37

348.3 Commands

- quast.py
- metaquast.py

348.4 Module

You can load the modules by:

```
module load biocontainers
module load quast
```

348.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Quast on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 8
#SBATCH --job-name=quast
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers quast

metaquast.py  --gene-finding --threads 8  \
    meta_contigs_1.fasta meta_contigs_2.fasta \
    -r meta_ref_1.fasta,meta_ref_2.fasta,meta_ref_3.fasta \
    -o quast_out_genefinding
```

CHAPTER

NINE

QUICKMIRSEQ

349.1 Introduction

`QuickMIRSeq` is an integrated pipeline for quick and accurate quantification of known miRNAs and isomiRs by jointly processing multiple samples.

For more information, please check its | Docker hub: https://hub.docker.com/r/gcfntnu/quickmirseq and its home page on Github.

349.2 Versions

- 1.0

349.3 Commands

- perl
- QuickMIRSeq-report.sh

349.4 Module

You can load the modules by:

```
module load biocontainers
module load quickmirseq
```

Note: This module defines program installation directory (note: inside the container!) as environment variable `$QuickMIRSeq`. Once again, this is not a host path, this path is only available from inside the container.

With the way this module is organized, you should be able to use the variable freely for both the `perl $QuickMIRSeq/QuickMIRSeq.pl allIDs.txt run.config` and the `$QuickMIRSeq/QuickMIRSeq-report.sh` steps as directed by the user guide.

A simple `QuickMIRSeq.pl` and `QuickMIRSeq-report.sh` will also work (and can be a backup if the variable expansion somehow does not work for you).

You will also need a run configuration file. You can copy from an existing one, or take from the user guide, or as a last resort, use Singularity to copy the template (in `$QuickMIRSeq/run.config.template`) from inside the container image. `singularity shell` may be an easiest way for the latter.

349.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run QuickMIRSeq on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=quickmirseq
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers quickmirseq

quickmerge -d out.rq.delta -q q.fasta -r scab8722.fasta  -hco 5.0 -c 1.5 -l n -ml m -p␣
↪prefix
```

R

350.1 Introduction

R is a system for statistical computation and graphics.

This is a plain R-base installation (see https://github.com/rocker-org/rocker/) repackaged by RCAC with an addition of a handful prerequisite libraries (libcurl, libopenssl, libxml2, libcairo2 and libXt) and their header files.

For more information, please check its | Docker hub: https://hub.docker.com/_/r-base and its home page: https://www.r-project.org/.

350.2 Versions

- 4.1.1

350.3 Commands

- R
- Rscript

350.4 Module

You can load the modules by:

```
module load biocontainers
module load r
```

350.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run R on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=r
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers r
```

CHAPTER

ONE

RACON

351.1 Introduction

Racon is a consensus module for raw de novo DNA assembly of long uncorrected reads.

For more information, please check its website: https://biocontainers.pro/tools/racon and its home page on Github.

351.2 Versions

- 1.4.20
- 1.5.0

351.3 Commands

- racon

351.4 Module

You can load the modules by:

```
module load biocontainers
module load racon
```

351.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Racon on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=racon
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers racon
```

CHAPTER

TWO

RAGOUT

352.1 Introduction

`Ragout` is a tool for chromosome-level scaffolding using multiple references.

For more information, please check its website: https://biocontainers.pro/tools/ragout and its home page on Github.

352.2 Versions

- 2.3-py37

352.3 Commands

- ragout

352.4 Module

You can load the modules by:

```
module load biocontainers
module load ragout
```

352.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Ragout on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=ragout
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers ragout
```

CHAPTER

THREE

RAGTAG

353.1 Introduction

`Ragtag` is a tool for fast reference-guided genome assembly scaffolding.

For more information, please check its website: https://biocontainers.pro/tools/ragtag and its home page on Github.

353.2 Versions

- 2.1.0

353.3 Commands

- ragtag.py

353.4 Module

You can load the modules by:

```
module load biocontainers
module load ragtag
```

353.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Ragtag on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=ragtag
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers ragtag

ragtag.py correct ref.fasta query.fasta
ragtag.py patch target.fa query.fa
```

CHAPTER

FOUR

RAPMAP

354.1 Introduction

RapMap is a testing ground for ideas in quasi-mapping and selective alignment.

For more information, please check:
BioContainers: https://biocontainers.pro/tools/rapmap
Home page: https://github.com/COMBINE-lab/RapMap

354.2 Versions

- 0.6.0

354.3 Commands

- rapmap

354.4 Module

You can load the modules by:

```
module load biocontainers
module load rapmap
```

354.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run rapmap on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=rapmap
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers rapmap
```

CHAPTER

FIVE

RASUSA

355.1 Introduction

Rasusa: Randomly subsample sequencing reads to a specified coverage.

For more information, please check:
Docker hub: https://hub.docker.com/r/staphb/rasusa
Home page: https://github.com/mbhall88/rasusa

355.2 Versions

- 0.6.0
- 0.7.0

355.3 Commands

- rasusa

355.4 Module

You can load the modules by:

```
module load biocontainers
module load rasusa
```

355.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run rasusa on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=rasusa
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers rasusa

rasusa -i seq_1.fq -i seq_2.fq  \
    --coverage 100 --genome-size 35mb  \
    -o out.r1.fq -o out.r2.fq
```

CHAPTER

SIX

RAVEN-ASSEMBLER

356.1 Introduction

`Raven-assembler` is a de novo genome assembler for long uncorrected reads.

For more information, please check its website: https://biocontainers.pro/tools/raven-assembler and its home page on Github.

356.2 Versions

- 1.8.1

356.3 Commands

- raven

356.4 Module

You can load the modules by:

```
module load biocontainers
module load raven-assembler
```

356.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Raven-assembler on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 12
#SBATCH --job-name=raven-assembler
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers raven-assembler

raven -t 12 input.fastq
```

CHAPTER

SEVEN

RAXML

357.1 Introduction

`Raxml` (Randomized Axelerated Maximum Likelihood) is a program for the Maximum Likelihood-based inference of large phylogenetic trees.

For more information, please check its website: https://biocontainers.pro/tools/raxml and its home page: https://cme.h-its.org/exelixis/web/software/raxml/.

357.2 Versions

- 8.2.12

357.3 Commands

- raxmlHPC
- raxmlHPC-AVX2
- raxmlHPC-PTHREADS
- raxmlHPC-PTHREADS-AVX2
- raxmlHPC-PTHREADS-SSE3
- raxmlHPC-SSE3

357.4 Module

You can load the modules by:

```
module load biocontainers
module load raxml
```

357.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Raxml on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 36
#SBATCH --job-name=raxml
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers raxml

raxmlHPC-SSE3 -m GTRGAMMA  -p 12345 -s input.fasta -n HPC-SSE3_out -# 20 -T 36
raxmlHPC -m GTRGAMMA  -p 12345 -s input.fasta -n HPC_out -# 20 -T 36
raxmlHPC-AVX2  -m GTRGAMMA  -p 12345 -s input.fasta -n HPC-AVX2_out -# 20 -T 36
raxmlHPC-PTHREADS  -m GTRGAMMA  -p 12345 -s input.fasta -n HPC-PTHREADS_out -# 20 -T 36
raxmlHPC-PTHREADS-AVX2  -m GTRGAMMA  -p 12345 -s input.fasta -n HPC-PTHREADS-AVX2_out -#␣
↪20 -T 36
raxmlHPC-PTHREADS-SSE3  -m GTRGAMMA  -p 12345 -s input.fasta -n HPC-PTHREADS-SSE3_out -#␣
↪20 -T 36
```

CHAPTER

EIGHT

RAXML-NG

358.1 Introduction

`Raxml-ng` is a phylogenetic tree inference tool which uses maximum-likelihood (ML) optimality criterion.

For more information, please check its website: https://biocontainers.pro/tools/raxml-ng and its home page on Github.

358.2 Versions

- 1.1.0

358.3 Commands

- raxml-ng
- raxml-ng-mpi
- mpirun
- mpiexec

358.4 Module

You can load the modules by:

```
module load biocontainers
module load raxml-ng
```

358.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Raxml-ng on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=raxml-ng
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers raxml-ng

raxml-ng --bootstrap --msa alignment.phy \
     --model GTR+G --threads 12 --bs-trees 1000
```

CHAPTER

NINE

REBALER

359.1 Introduction

`Rebaler` is a program for conducting reference-based assemblies using long reads.

For more information, please check its website: https://biocontainers.pro/tools/rebaler and its home page on Github.

359.2 Versions

- 0.2.0

359.3 Commands

- rebaler

359.4 Module

You can load the modules by:

```
module load biocontainers
module load rebaler
```

359.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Rebaler on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=rebaler
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers rebaler
```

RECIPROCAL SMALLEST DISTANCE

360.1 Introduction

The `reciprocal smallest distance` (RSD) algorithm accurately infers orthologs between pairs of genomes by considering global sequence alignment and maximum likelihood evolutionary distance between sequences.

For more information, please check its home page on Github.

360.2 Versions

- 1.1.7

360.3 Commands

- rsd_search
- rsd_blast
- rsd_format

360.4 Module

You can load the modules by:

```
module load biocontainers
module load reciprocal_smallest_distance
```

360.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Reciprocal Smallest Distance on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=reciprocal_smallest_distance
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers reciprocal_smallest_distance

rsd_search
    -q Mycoplasma_genitalium.aa \
    --subject-genome=Mycobacterium_leprae.aa \
    -o Mycoplasma_genitalium.aa_Mycobacterium_leprae.aa_0.8_1e-5.orthologs.txt

rsd_format -g Mycoplasma_genitalium.aa

rsd_blast -v -q Mycoplasma_genitalium.aa \
    --subject-genome=Mycobacterium_leprae.aa \
    --forward-hits q_s.hits --reverse-hits s_q.hits \
    --no-format --evalue 0.1
```

CHAPTER

ONE

RECYCLER

361.1 Introduction

`Recycler` is a tool designed for extracting circular sequences from de novo assembly graphs.

For more information, please check its website: https://biocontainers.pro/tools/recycler and its home page on Github.

361.2 Versions

- 0.7

361.3 Commands

- make_fasta_from_fastg.py
- get_simple_cycs.py
- recycle.py

361.4 Module

You can load the modules by:

```
module load biocontainers
module load recycler
```

361.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Recycler on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=recycler
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers recycler

recycle.py -g test/assembly_graph.fastg \
    -k 55 -b test/test.sort.bam -i True
```

CHAPTER

TWO

REPEATMASKER

362.1 Introduction

`RepeatMakser` is a program that screens DNA sequences for interspersed repeats and low complexity DNA sequences. Detailed usage can be found here: http://www.repeatmasker.org.

362.2 Versions

- 4.1.2

362.3 Commands

- RepeatMasker

362.4 Database

Note: As of May 20, 2019 GIRI has rescinded the working agreement allowing the www.repeatmasker.org website to offer a repeatmasking service utilizing the RepBase RepeatMasker Edition library. As a result, repeatmasker can only offer masking using the open database Dfam, which starting in 3.0 includes consensus sequences in addition to profile hidden Markov models for many transposable element families. Users requiring RepBase will need to purchase a commercial or academic license from GIRI and run RepeatMasker localy.

In our cluster, we set up the Dfam relaese 3.5 (October 2021) that include 285,580 repetitive DNA families.

362.5 Species name

Note: Since v4.1.1, RepeatMakser has switched to the FamDB format for the Dfam database. Due to this change, RepeatMasker becomes more strict with regards to what is acceptable for the `-species` flag. The commonly used names such as "mammal" and "mouse" will not be accepted. To check for valid names, you can query the database using the python script `famdb.py` (https://github.com/Dfam-consortium/FamDB).

See `famdb.py --help` for usage information and below for an example the check the valid name for "mammal" using our copy of the Dfam database:

```
/depot/itap/datasets/Maker/RepeatMasker/Libraries/famdb.py -i /depot/itap/datasets/Maker/
↪RepeatMasker/Libraries/Dfam.h5 names mammal
```

362.6 Module

You can load the modules by:

```
module load biocontainers
module load repeatmasker/4.1.2
```

362.7 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run RepeatMasker on our cluster:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 2:00:00
#SBATCH -N 1
#SBATCH -n 24
#SBATCH --job-name=repeatmsker
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers repeatmasker/4.1.2

RepeatMasker -pa 24 -species mammals genome.fasta
```

CHAPTER

THREE

REPEATMODELER

363.1 Introduction

`RepeatModeler` is a de novo transposable element (TE) family identification and modeling package.

For more information, please check its website: https://biocontainers.pro/tools/repeatmodeler and its home page: http://www.repeatmasker.org/RepeatModeler/.

363.2 Versions

- 2.0.2
- 2.0.3

363.3 Commands

- RepeatModeler
- BuildDatabase
- RepeatClassifier

363.4 Module

You can load the modules by:

```
module load biocontainers
module load repeatmodeler
```

363.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run RepeatModeler on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=repeatmodeler
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers repeatmodeler
```

CHAPTER

FOUR

REPEATSCOUT

364.1 Introduction

`RepeatScout` is a tool to discover repetitive substrings in DNA.

For more information, please check its website: https://biocontainers.pro/tools/repeatscout and its home page on Github.

364.2 Versions

- 1.0.6

364.3 Commands

- RepeatScout
- build_lmer_table
- compare-out-to-gff.prl
- filter-stage-1.prl
- filter-stage-2.prl
- merge-lmer-tables.prl

364.4 Module

You can load the modules by:

```
module load biocontainers
module load repeatscout
```

364.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run RepeatScout on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=repeatscout
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers repeatscout

build_lmer_table -l 14 -sequence genome.fasta -freq Final_assembly.freq

RepeatScout -sequence genome.fasta -output Final_assembly_repeats.fasta -freq Final_
↪assembly.freq -l 14
```

CHAPTER

FIVE

RESFINDER

365.1 Introduction

ResFinder identifies acquired antimicrobial resistance genes in total or partial sequenced isolates of bacteria.

For more information, please check:
Home page: https://github.com/cadms/resfinder

365.2 Versions

- 4.1.5

365.3 Commands

- run_resfinder.py
- run_batch_resfinder.py

365.4 Module

You can load the modules by:

```
module load biocontainers
module load resfinder
```

365.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run resfinder on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=resfinder
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers resfinder

run_resfinder.py -o output -db_res db_resfinder/ \
     -db_res_kma db_resfinder/kma_indexing -db_point db_pointfinder/ \
     -s "Escherichia coli" --acquired --point -ifq data/test_isolate_01_*
```

CHAPTER

SIX

REVBAYES

366.1 Introduction

RevBayes – Bayesian phylogenetic inference using probabilistic graphical models and an interactive language.

For more information, please check:
Home page: https://github.com/revbayes/revbayes

366.2 Versions

- 1.1.1

366.3 Commands

- rb
- rb-mpi

366.4 Module

You can load the modules by:

```
module load biocontainers
module load revbayes
```

366.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run revbayes on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=revbayes
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers revbayes
```

CHAPTER

SEVEN

RMATS

367.1 Introduction

MATS is a computational tool to detect differential alternative splicing events from RNA-Seq data. The statistical model of MATS calculates the P-value and false discovery rate that the difference in the isoform ratio of a gene between two conditions exceeds a given user-defined threshold. From the RNA-Seq data, MATS can automatically detect and analyze alternative splicing events corresponding to all major types of alternative splicing patterns. MATS handles replicate RNA-Seq data from both paired and unpaired study design.

Detailed usage can be found here: http://rnaseq-mats.sourceforge.net

367.2 Versions

- 4.1.1-py37

367.3 Commands

- rmats.py

367.4 Module

You can load the modules by:

```
module load biocontainers
module load rmats
```

367.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run rmats on our our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 10:00:00
#SBATCH -N 1
#SBATCH -n 24
#SBATCH --job-name=rmats
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers rmats

rmats.py --b1 SR_b1.txt --b2 SR_b2.txt --gtf Homo_sapiens.GRCh38.105.gtf --od rmats_out_
↪homo --tmp rmats_tmp  -t paired --nthread 10 --readLength 150
```

CHAPTER

EIGHT

RMATS2SASHIMIPLOT

368.1 Introduction

`rmats2sashimiplot` produces a sashimiplot visualization of rMATS output. rmats2sashimiplot can also produce plots using an annotation file and genomic coordinates. The plotting backend is MISO.

Detailed usage can be found here: https://github.com/Xinglab/rmats2sashimiplot

368.2 Versions

- 2.0.4-py37

368.3 Commands

- rmats2sashimiplot

368.4 Module

You can load the modules by:

```
module load biocontainers
module load rmats2sashimiplot
```

368.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run rmats on our our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 8
#SBATCH --job-name=rmats2sashimiplot
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers rmats2sashimiplot

rmats2sashimiplot --s1 sample_1_replicate_1.sam,sample_1_replicate_2.sam,sample_1_
↪replicate_3.sam \
                  --s2 sample_2_replicate_1.sam,sample_2_replicate_2.sam,sample_2_
↪replicate_3.sam \
                  -t SE -e SE.MATS.JC.txt --l1 SampleOne --l2 SampleTwo --exon_s 1 --
↪intron_s 5 \
                  -o test_events_output
```

CHAPTER

NINE

RNAINDEL

369.1 Introduction

`RNAIndel` calls coding indels from tumor RNA-Seq data and classifies them as somatic, germline, and artifactual. RNAIndel supports GRCh38 and 37.

For more information, please check its Github package: https://github.com/stjude/RNAIndel/pkgs/container/rnaindel and its home page on Github.

369.2 Versions

- 3.0.9

369.3 Commands

- rnaindel

369.4 Module

You can load the modules by:

```
module load biocontainers
module load rnaindel
```

369.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run RNAIndel on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=rnaindel
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers rnaindel
```

RNAPEG

370.1 Introduction

`RNApeg` is an RNA junction calling, correction, and quality-control package. RNAIndel supports GRCh38 and 37.

For more information, please check its Github package: https://github.com/stjude/RNApeg/pkgs/container/rnapeg and its home page on Github.

370.2 Versions

- 2.7.1

370.3 Commands

- RNApeg.sh

370.4 Module

You can load the modules by:

```
module load biocontainers
module load rnapeg
```

370.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run RNApeg on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=rnapeg
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers rnapeg
```

CHAPTER

ONE

RNAQUAST

371.1 Introduction

`Rnaquast` is a quality assessment tool for de novo transcriptome assemblies.

For more information, please check its website: https://biocontainers.pro/tools/rnaquast and its home page: http://cab.spbu.ru/software/rnaquast/.

371.2 Versions

- 2.2.1

371.3 Commands

- rnaQUAST.py

371.4 Module

You can load the modules by:

```
module load biocontainers
module load rnaquast
```

371.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Rnaquast on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 12
#SBATCH --job-name=rnaquast
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers rnaquast

rnaQUAST.py -t 12 -o output \
     --transcripts test_data/Trinity.fasta test_data/idba.fasta \
     --reference test_data/Saccharomyces_cerevisiae.R64-1-1.75.dna.toplevel.fa \
     --gtf test_data/Saccharomyces_cerevisiae.R64-1-1.75.gtf
```

CHAPTER

TWO

ROARY

372.1 Introduction

Roary is a high speed stand alone pan genome pipeline, which takes annotated assemblies in GFF3 format (produced by Prokka) and calculates the pan genome.

For more information, please check:
Docker hub: https://hub.docker.com/r/staphb/roary
Home page: https://github.com/sanger-pathogens/Roary

372.2 Versions

- 3.13.0

372.3 Commands

- roary

372.4 Module

You can load the modules by:

```
module load biocontainers
module load roary
```

372.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run roary on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=roary
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers roary

roary -f demo -e -n -v gff/*.gff
```

CHAPTER

THREE

R-RNASEQ

373.1 Introduction

`r-rnaseq` is a customerized R module based on `R/4.1.1` used for RNAseq analysis.

In the module, we have some packages installed:

- BiocManager 1.30.16
- ComplexHeatmap 2.9.4
- DESeq2 1.34.0
- edgeR 3.36.0
- pheatmap 1.0.12
- limma 3.48.3
- tibble 3.1.5
- tidyr 1.1.4
- readr 2.0.2
- readxl 1.3.1
- purrr 0.3.4
- dplyr 1.0.7
- stringr 1.4.0
- forcats 0.5.1
- ggplot2 3.3.5
- openxlsx 4.2.5

373.2 Versions

- 4.1.1-1
- 4.1.1-1-rstudio

373.3 Commands

- R
- Rscript
- rstudio (only for the rstudio version)

373.4 Module

You can load the modules by:

```
module load biocontainers
module load r-rnaseq/4.1.1-1
# If you want to use Rstudio, load the rstudio version
module load r-rnaseq/4.1.1-1-rstudio
```

373.5 Install packages

Note: Users can also install packages they need. The installed location depends on the setting in your `~/.Rprofile`. Detailed guide about installing R packages can be found here: https://www.rcac.purdue.edu/knowledge/bell/run/examples/apps/r/package.

373.6 Interactive job

To run interactively on our clusters:

```
(base) UserID@bell-fe00:~ $ sinteractive -N1 -n12 -t4:00:00 -A myallocation
salloc: Granted job allocation 12345869
salloc: Waiting for resource configuration
salloc: Nodes bell-a008 are ready for job
(base) UserID@bell-a008:~ $ module load biocontainers r-rnaseq/4.1.1-1 # or r-rnaseq/4.1.
↪1-1-rstudio
(base) UserID@bell-a008:~ $ R

R version 4.1.1 (2021-08-10) -- "Kick Things"
Copyright (C) 2021 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)
```

(continues on next page)

(continued from previous page)

```
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.


> library(edgeR)
> library(pheatmap)
```

373.7 Batch job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To submit a sbatch job on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 10:00:00
#SBATCH -N 1
#SBATCH -n 24
#SBATCH --job-name=r_RNAseq
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers r-rnaseq

Rscript RNAseq.R
```

CHAPTER

FOUR

RSTUDIO

374.1 Introduction

`RStudio` is an integrated development environment (IDE) for the R statistical computation and graphics system.

This is an RStudio IDE together with a plain R-base installation (see https://github.com/rocker-org/rocker/), repackaged by RCAC with an addition of a handful prerequisite libraries (libcurl, libopenssl, libxml2, libcairo2 and libXt) and their header files. It is intentionally separate from the biocontainers' 'r' module for reasons of image size (700MB vs 360MB).

For more information, please check its | Docker hub: https://hub.docker.com/_/r-base and its home page: https://www.rstudio.com/products/rstudio/ and https://www.r-project.org/.

374.2 Versions

- 4.1.1

374.3 Commands

- R
- Rscript
- rstudio

374.4 Module

You can load the modules by:

```
module load biocontainers
module load r-studio
```

374.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run RStudio on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=r-studio
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers r-studio
```

CHAPTER

FIVE

R-SCRNASEQ

375.1 Introduction

`r-scrnaseq` is a customerized R module based on `R/4.1.1` or `R/4.2.0` used for scRNAseq analysis.

In the module, we have some packages installed:

- BiocManager 1.30.16
- Seurat 4.1.0
- SeuratObject 4.0.4
- SeuratWrappers 0.3.0
- monocle3 1.0.0
- SnapATAC 1.0.0
- SingleCellExperiment 1.14.1, 1.16.0
- scDblFinder 1.8.0
- SingleR 1.8.1
- scCATCH 3.0
- scMappR 1.0.7
- rliger 1.0.0
- schex 1.8.0
- CoGAPS 3.14.0
- celldex 1.4.0
- dittoSeq 1.6.0
- DropletUtils 1.14.2
- miQC 1.2.0
- Nebulosa 1.4.0
- tricycle 1.2.0
- pheatmap 1.0.12
- limma 3.48.3, 3.50.0
- tibble 3.1.5
- tidyr 1.1.4

- readr 2.0.2
- readxl 1.3.1
- purrr 0.3.4
- dplyr 1.0.7
- stringr 1.4.0
- forcats 0.5.1
- ggplot2 3.3.5
- openxlsx 4.2.5

375.2 Versions

- 4.1.1-1
- 4.1.1-1-rstudio
- 4.2.0
- 4.2.0-rstudio

375.3 Commands

- R
- Rscript
- rstudio (only for the rstudio version)

375.4 Module

You can load the modules by:

```
module load biocontainers
module load r-scrnaseq
# or module load r-scrnaseq/4.2.0
# If you want to use Rstudio, load the rstudio version
module load r-scrnaseq/4.1.1-1-rstudio
# or module load r-scrnaseq/4.2.0-rstudio
```

375.5 Install packages

Note: Users can also install packages they need. The installed location depends on the setting in your `~/.Rprofile`. Detailed guide about installing R packages can be found here: https://www.rcac.purdue.edu/knowledge/bell/run/examples/apps/r/package.

375.6 Interactive job

To run interactively on our clusters:

```
(base) UserID@bell-fe00:~ $ sinteractive -N1 -n12 -t4:00:00 -A myallocation
salloc: Granted job allocation 12345869
salloc: Waiting for resource configuration
salloc: Nodes bell-a008 are ready for job
(base) UserID@bell-a008:~ $ module load biocontainers r-scrnaseq/4.2.0 # or r-scrnaseq/4.
↪2.0-rstudio
(base) UserID@bell-a008:~ $ R

R version 4.2.0 (2022-04-22) -- "Vigorous Calisthenics"
Copyright (C) 2022 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.


> library(Seurat)
> library(monocle3)
```

375.7 Batch job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To submit a sbatch job on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 10:00:00
#SBATCH -N 1
#SBATCH -n 24
#SBATCH --job-name=r_scRNAseq
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers r-scrnaseq

Rscript scRNAseq.R
```

CHAPTER

SIX

RSEM

376.1 Introduction

RSEM is a software package for estimating gene and isoform expression levels from RNA-Seq data. Further information can be found here: https://deweylab.github.io/RSEM/.

376.2 Versions

- 1.3.3

376.3 Commands

- rsem-bam2readdepth
- rsem-bam2wig
- rsem-build-read-index
- rsem-calculate-credibility-intervals
- rsem-calculate-expression
- rsem-control-fdr
- rsem-extract-reference-transcripts
- rsem-generate-data-matrix
- rsem-generate-ngvector
- rsem-gen-transcript-plots
- rsem-get-unique
- rsem-gff3-to-gtf
- rsem-parse-alignments
- rsem-plot-model
- rsem-plot-transcript-wiggles
- rsem-prepare-reference
- rsem-preref

- rsem-refseq-extract-primary-assembly
- rsem-run-ebseq
- rsem-run-em
- rsem-run-gibbs
- rsem-run-prsem-testing-procedure
- rsem-sam-validator
- rsem-scan-for-paired-end-reads
- rsem-simulate-reads
- rsem-synthesis-reference-transcripts
- rsem-tbam2gbam

376.4 Dependencies

`STAR v2.7.9a`, `Bowtie v1.2.3`, `Bowtie2 v2.3.5.1`, `HISAT2 v2.2.1` were included in the container image. So users do not need to provide the dependency path in the RSEM parameter.

376.5 Module

You can load the modules by:

```
module load biocontainers
module load rsem/1.3.3
```

376.6 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run RSEM on our cluster:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 10:00:00
#SBATCH -N 1
#SBATCH -n 24
#SBATCH --job-name=rsem
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers rsem/1.3.3
```

(continues on next page)

(continued from previous page)

```
rsem-prepare-reference --gtf Homo_sapiens.GRCh38.105.gtf --bowtie Homo_sapiens.GRCh38.
↪dna.primary_assembly.fa Gh38_bowtie  -p 24
rsem-prepare-reference --gtf Homo_sapiens.GRCh38.105.gtf --bowtie2 Homo_sapiens.GRCh38.
↪dna.primary_assembly.fa Gh38_bowtie2  -p 24
rsem-prepare-reference --gtf Homo_sapiens.GRCh38.105.gtf --hisat2-hca  Homo_sapiens.
↪GRCh38.dna.primary_assembly.fa Gh38_hisat2  -p 24
rsem-prepare-reference --gtf Homo_sapiens.GRCh38.105.gtf --star Homo_sapiens.GRCh38.dna.
↪primary_assembly.fa Gh38_star  -p 24
rsem-calculate-expression --paired-end --star -p 24 SRR12095148_1.fastq SRR12095148_2.
↪fastq  Gh38_star SRR12095148_rsem_expression
```

CHAPTER

SEVEN

RSEQC

377.1 Introduction

`Rseqc` is a package provides a number of useful modules that can comprehensively evaluate high throughput sequence data especially RNA-seq data.

For more information, please check its website: https://biocontainers.pro/tools/rseqc and its home page: http://rseqc.sourceforge.net.

377.2 Versions

- 4.0.0-py37

377.3 Commands

- FPKM-UQ.py
- FPKM_count.py
- RNA_fragment_size.py
- RPKM_saturation.py
- aggregate_scores_in_intervals.py
- align_print_template.py
- axt_extract_ranges.py
- axt_to_fasta.py
- axt_to_lav.py
- axt_to_maf.py
- bam2fq.py
- bam2wig.py
- bam_stat.py
- bed_bigwig_profile.py

- bed_build_windows.py
- bed_complement.py
- bed_count_by_interval.py
- bed_count_overlapping.py
- bed_coverage.py
- bed_coverage_by_interval.py
- bed_diff_basewise_summary.py
- bed_extend_to.py
- bed_intersect.py
- bed_intersect_basewise.py
- bed_merge_overlapping.py
- bed_rand_intersect.py
- bed_subtract_basewise.py
- bnMapper.py
- clipping_profile.py
- deletion_profile.py
- div_snp_table_chr.py
- divide_bam.py
- find_in_sorted_file.py
- geneBody_coverage.py
- geneBody_coverage2.py
- gene_fourfold_sites.py
- get_scores_in_intervals.py
- infer_experiment.py
- inner_distance.py
- insertion_profile.py
- int_seqs_to_char_strings.py
- interval_count_intersections.py
- interval_join.py
- junction_annotation.py
- junction_saturation.py
- lav_to_axt.py
- lav_to_maf.py
- line_select.py
- lzop_build_offset_table.py
- mMK_bitset.py

- maf_build_index.py
- maf_chop.py
- maf_chunk.py
- maf_col_counts.py
- maf_col_counts_all.py
- maf_count.py
- maf_covered_ranges.py
- maf_covered_regions.py
- maf_div_sites.py
- maf_drop_overlapping.py
- maf_extract_chrom_ranges.py
- maf_extract_ranges.py
- maf_extract_ranges_indexed.py
- maf_filter.py
- maf_filter_max_wc.py
- maf_gap_frequency.py
- maf_gc_content.py
- maf_interval_alignibility.py
- maf_limit_to_species.py
- maf_mapping_word_frequency.py
- maf_mask_cpg.py
- maf_mean_length_ungapped_piece.py
- maf_percent_columns_matching.py
- maf_percent_identity.py
- maf_print_chroms.py
- maf_print_scores.py
- maf_randomize.py
- maf_region_coverage_by_src.py
- maf_select.py
- maf_shuffle_columns.py
- maf_species_in_all_files.py
- maf_split_by_src.py
- maf_thread_for_species.py
- maf_tile.py
- maf_tile_2.py
- maf_tile_2bit.py

- maf_to_axt.py
- maf_to_concat_fasta.py
- maf_to_fasta.py
- maf_to_int_seqs.py
- maf_translate_chars.py
- maf_truncate.py
- maf_word_frequency.py
- mask_quality.py
- mismatch_profile.py
- nib_chrom_intervals_to_fasta.py
- nib_intervals_to_fasta.py
- nib_length.py
- normalize_bigwig.py
- one_field_per_line.py
- out_to_chain.py
- overlay_bigwig.py
- prefix_lines.py
- pretty_table.py
- qv_to_bqv.py
- random_lines.py
- read_GC.py
- read_NVC.py
- read_distribution.py
- read_duplication.py
- read_hexamer.py
- read_quality.py
- split_bam.py
- split_paired_bam.py
- table_add_column.py
- table_filter.py
- tfloc_summary.py
- tin.py
- ucsc_gene_table_to_intervals.py
- wiggle_to_array_tree.py
- wiggle_to_binned_array.py
- wiggle_to_chr_binned_array.py

- wiggle_to_simple.py

377.4 Module

You can load the modules by:

```
module load biocontainers
module load rseqc
```

377.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Rseqc on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=rseqc
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers rseqc

bam_stat.py -i *.bam -q 30
```

CHAPTER

EIGHT

RUN-DBCAN

378.1 Introduction

`run_dbCAN` using genomes/metagenomes/proteomes of any assembled organisms (prokaryotes, fungi, plants, animals, viruses) to search for CAZymes. This is a standalone tool of http://bcb.unl.edu/dbCAN2/. Details aobut its uage can be found in its Github repository.

378.2 Versions

- 3.0.2

378.3 Commands

run_dbcan

378.4 Database

Latest version of database has been downloaded and setup, including CAZyDB.09242021.fa, dbCAN-HMMdb-V10.txt, tcdb.fa, tf-1.hmm, tf-2.hmm, and stp.hmm.

378.5 Module

You can load the modules by:

```
module load biocontainers
module load run_dbcan/3.0.2
```

378.6 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run run_dbcan on our cluster:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 10:00:00
#SBATCH -N 1
#SBATCH -n 24
#SBATCH --job-name=run_dbcan
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers run_dbcan/3.0.2

run_dbcan protein.faa protein --out_dir test1_dbcan
run_dbcan genome.fasta prok --out_dir test2_dbcan
```

CHAPTER

NINE

RUSH

379.1 Introduction

`rush` is a tool similar to GNU parallel and gargs. rush borrows some idea from them and has some unique features, e.g., supporting custom defined variables, resuming multi-line commands, more advanced embeded replacement strings.

For more information, please check its home page on Github.

379.2 Versions

- 0.4.2

379.3 Commands

- rush

379.4 Module

You can load the modules by:

```
module load biocontainers
module load rush
```

379.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run rush on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=rush
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers rush
```

SALMON

380.1 Introduction

`Salmon` is a wicked-fast program to produce a highly-accurate, transcript-level quantification estimates from RNA-seq data.

Detailed usage can be found here: https://github.com/COMBINE-lab/salmon

380.2 Versions

- 1.5.2
- 1.6.0
- 1.7.0
- 1.8.0
- 1.9.0

380.3 Commands

- salmon index
- salmon quant
- salmon alevin
- salmon swim
- salmon quantmerge

380.4 Module

You can load the modules by:

```
module load biocontainers
module load salmon
```

380.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Salmon on our our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 10:00:00
#SBATCH -N 1
#SBATCH -n 24
#SBATCH --job-name=salmon
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers salmon

salmon index -t  Homo_sapiens.GRCh38.cds.all.fa -i salmon_index
salmon quant -i salmon_index -l A -p 24 -1 SRR16956239_1.fastq -2 SRR16956239_2.fastq --
↪validateMappings -o transcripts_quan
```

CHAPTER

ONE

SAMBAMBA

381.1 Introduction

Sambamba is a high performance highly parallel robust and fast tool (and library), written in the D programming language, for working with SAM and BAM files.

For more information, please check its website: https://biocontainers.pro/tools/sambamba and its home page on Github.

381.2 Versions

- 0.8.2

381.3 Commands

- sambamba

381.4 Module

You can load the modules by:

```
module load biocontainers
module load sambamba
```

381.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Sambamba on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=sambamba
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers sambamba

sambamba view --reference-info input.bam
sambamba view -c -F "mapping_quality >= 40" input.bam
```

CHAPTER

TWO

SAMBLASTER

382.1 Introduction

`Samblaster` is a tool to mark duplicates and extract discordant and split reads from sam files.

For more information, please check its website: https://biocontainers.pro/tools/samblaster and its home page on Github.

382.2 Versions

- 0.1.26

382.3 Commands

- samblaster

382.4 Module

You can load the modules by:

```
module load biocontainers
module load samblaster
```

382.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Samblaster on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=samblaster
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers samblaster
```

CHAPTER

THREE

SAMCLIP

383.1 Introduction

Samclip is a tool to filter SAM file for soft and hard clipped alignments.

For more information, please check:
BioContainers: https://biocontainers.pro/tools/samclip
Home page: https://github.com/tseemann/samclip

383.2 Versions

- 0.4.0

383.3 Commands

- samclip

383.4 Module

You can load the modules by:

```
module load biocontainers
module load samclip
```

383.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run samclip on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=samclip
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers samclip

samclip --ref test.fna < test.sam > out.sam
```

CHAPTER

FOUR

SAMPLOT

384.1 Introduction

`Samplot` is a command line tool for rapid, multi-sample structural variant visualization.

For more information, please check its website: https://biocontainers.pro/tools/samplot and its home page on Github.

384.2 Versions

- 1.3.0

384.3 Commands

- samplot

384.4 Module

You can load the modules by:

```
module load biocontainers
module load samplot
```

384.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Samplot on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=samplot
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers samplot

samplot plot \
-n NA12878 NA12889 NA12890 \
-b samplot/test/data/NA12878_restricted.bam \
  samplot/test/data/NA12889_restricted.bam \
  samplot/test/data/NA12890_restricted.bam \
-o 4_115928726_115931880.png \
-c chr4 \
-s 115928726 \
-e 115931880 \
-t DEL
```

CHAPTER

FIVE

SAMTOOLS

385.1 Introduction

`Samtools` is a set of utilities for the Sequence Alignment/Map (SAM) format.

For more information, please check its website: https://biocontainers.pro/tools/samtools and its home page on Github.

385.2 Versions

- 1.15
- 1.9

385.3 Commands

- samtools
- ace2sam
- htsfile
- maq2sam-long
- maq2sam-short
- tabix
- wgsim

385.4 Module

You can load the modules by:

```
module load biocontainers
module load samtools
```

385.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Samtools on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=samtools
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers samtools
```

CHAPTER

SIX

SCANPY

386.1 Introduction

Scanpy is scalable toolkit for analyzing single-cell gene expression data. It includes preprocessing, visualization, clustering, pseudotime and trajectory inference and differential expression testing. The Python-based implementation efficiently deals with datasets of more than one million cells. Details about its usage can be found here (https://scanpy.readthedocs.io/en/stable/)

386.2 Versions

- 1.8.2
- 1.9.1

386.3 Commands

- python
- python3

386.4 Module

You can load the modules by:

```
module load biocontainers
module load scanpy/1.8.2
```

386.5 Interactive job

To run scanpy interactively on our clusters:

```
(base) UserID@bell-fe00:~ $ sinteractive -N1 -n12 -t4:00:00 -A myallocation
salloc: Granted job allocation 12345869
salloc: Waiting for resource configuration
salloc: Nodes bell-a008 are ready for job
(base) UserID@bell-a008:~ $ module load biocontainers scanpy/1.8.2
(base) UserID@bell-a008:~ $ python
Python 3.9.5 (default, Jun  4 2021, 12:28:51)
[GCC 7.5.0] :: Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import scanpy as sc
>>> sc.tl.umap(adata, **tool_params)
```

386.6 Batch job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To submit a sbatch job on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 10:00:00
#SBATCH -N 1
#SBATCH -n 24
#SBATCH --job-name=scanpy
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers scanpy/1.8.2

python script.py
```

CHAPTER

SEVEN

SCARCHES

387.1 Introduction

scArches is a package to integrate newly produced single-cell datasets into integrated reference atlases.

For more information, please check:
Home page: https://github.com/theislab/scarches

387.2 Versions

- 0.5.3

387.3 Commands

- python
- python3

387.4 Module

You can load the modules by:

```
module load biocontainers
module load scarches
```

387.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run scarches on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=scarches
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers scarches
```

CHAPTER

EIGHT

SCGEN

388.1 Introduction

scGen is a generative model to predict single-cell perturbation response across cell types, studies and species.

For more information, please check:
Home page: https://github.com/theislab/scgen

388.2 Versions

- 2.1.0

388.3 Commands

- python
- python3

388.4 Module

You can load the modules by:

```
module load biocontainers
module load scgen
```

388.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run scgen on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=scgen
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers scgen
```

CHAPTER

NINE

SCIRPY

389.1 Introduction

Scirpy is a scalable python-toolkit to analyse T cell receptor (TCR) or B cell receptor (BCR) repertoires from single-cell RNA sequencing (scRNA-seq) data. It seamlessly integrates with the popular scanpy library and provides various modules for data import, analysis and visualization.

For more information, please check:
Home page: https://github.com/scverse/scirpy

389.2 Versions

- 0.10.1

389.3 Commands

- python
- python3

389.4 Module

You can load the modules by:

```
module load biocontainers
module load scirpy
```

389.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run scirpy on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=scirpy
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers scirpy
```

SCVELO

390.1 Introduction

scVelo is a scalable toolkit for RNA velocity analysis in single cells, based on https://doi.org/10.1038/s41587-020-0591-3. Its detailed usage can be found here: https://scvelo.readthedocs.io.

390.2 Versions

- 0.2.4

390.3 Commands

- python
- python3

390.4 Module

You can load the modules by:

```
module load biocontainers
module load scvelo/0.2.4
```

390.5 Interactive job

To run scVelo interactively on our clusters:

```
(base) UserID@bell-fe00:~ $ sinteractive -N1 -n12 -t4:00:00 -A myallocation
salloc: Granted job allocation 12345869
salloc: Waiting for resource configuration
salloc: Nodes bell-a008 are ready for job
(base) UserID@bell-a008:~ $ module load biocontainers scvelo/0.2.4
(base) UserID@bell-a008:~ $ python
Python 3.9.5 (default, Jun  4 2021, 12:28:51)
[GCC 7.5.0] :: Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license" for more information.
```

(continues on next page)

(continued from previous page)

```
>>> import scvelo as scv
>>> scv.set_figure_params()
```

390.6 Batch job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To submit a sbatch job on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 10:00:00
#SBATCH -N 1
#SBATCH -n 24
#SBATCH --job-name=scvelo
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers scvelo/0.2.4

python script.py
```

CHAPTER

ONE

SCVI-TOOLS

391.1 Introduction

scvi-tools (single-cell variational inference tools) is a package for end-to-end analysis of single-cell omics data primarily developed and maintained by the Yosef Lab at UC Berkeley.

For more information, please check:
Home page: https://scvi-tools.org

391.2 Versions

- 0.16.2

391.3 Commands

- python
- python3
- R
- Rscript

391.4 Module

You can load the modules by:

```
module load biocontainers
module load scvi-tools
```

391.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run scvi-tools on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=scvi-tools
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers scvi-tools
```

CHAPTER

TWO

SEIDR

392.1 Introduction

`Seidr` is a community gene network inference and exploration toolkit.

For more information, please check its website: https://biocontainers.pro/tools/seidr and its home page on Github.

392.2 Versions

- 0.14.2

392.3 Commands

- correlation
- seidr
- mi
- pcor
- narromi
- plsnet
- llr-ensemble
- svm-ensemble
- genie3
- tigress
- el-ensemble
- makeconv
- genrb
- gencfu
- gencnval
- gendict

- tomsimilarity

392.4 Module

You can load the modules by:

```
module load biocontainers
module load seidr
```

392.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Seidr on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=seidr
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers seidr
```

CHAPTER

THREE

SEPP

393.1 Introduction

Sepp stands for SATé-Enabled Phylogenetic Placement and addresses the problem of phylogenetic placement for metagenomic short reads.

For more information, please check its website: https://biocontainers.pro/tools/sepp and its home page on Github.

393.2 Versions

- 4.5.1-py37

393.3 Commands

- run_sepp.py
- run_upp.py
- split_sequences.py
- sumlabels.py
- sumtrees.py

393.4 Module

You can load the modules by:

```
module load biocontainers
module load sepp
```

393.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Sepp on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=sepp
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers sepp

run_sepp.py -t mock/rpsS/sate.tre \
    -r mock/rpsS/sate.tre.RAxML_info \
    -a mock/rpsS/sate.fasta \
    -f mock/rpsS/rpsS.even.fas \
    -o rpsS.out.default
```

CHAPTER

FOUR

SEQKIT

394.1 Introduction

`Seqkit` is a rapid tool for manipulating fasta and fastq files.

For more information, please check its website: https://biocontainers.pro/tools/seqkit and its home page on Github.

394.2 Versions

- 2.0.0
- 2.1.0

394.3 Commands

- seqkit

394.4 Module

You can load the modules by:

```
module load biocontainers
module load seqkit
```

394.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Seqkit on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=seqkit
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers seqkit

seqkit stats configs.fasta > contigs_statistics.txt
```

CHAPTER

FIVE

SEQYCLEAN

395.1 Introduction

Seqyclean is used to pre-process NGS data in order to prepare for downstream analysis.

For more information, please check:
Docker hub: https://hub.docker.com/r/staphb/seqyclean
Home page: https://github.com/ibest/seqyclean

395.2 Versions

- 1.10.09

395.3 Commands

- seqyclean

395.4 Module

You can load the modules by:

```
module load biocontainers
module load seqyclean
```

395.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run seqyclean on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=seqyclean
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers seqyclean
```

CHAPTER

SIX

SHASTA

396.1 Introduction

Shasta is a software for de novo assembly from Oxford Nanopore reads.

For more information, please check:
Home page: https://github.com/chanzuckerberg/shasta

396.2 Versions

- 0.10.0

396.3 Commands

- shasta

396.4 Module

You can load the modules by:

```
module load biocontainers
module load shasta
```

396.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run shasta on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=shasta
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers shasta

shasta --input r94_ec_rad2.181119.60x-10kb.fasta \
    --config Nanopore-May2022
```

CHAPTER

SEVEN

SHORAH

397.1 Introduction

Shorah is an open source project for the analysis of next generation sequencing data.

For more information, please check its website: https://biocontainers.pro/tools/shorah and its home page on Github.

397.2 Versions

- 1.99.2-py37

397.3 Commands

- shorah
- b2w
- diri_sampler
- fil

397.4 Module

You can load the modules by:

```
module load biocontainers
module load shorah
```

397.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Shorah on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=shorah
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers shorah

shorah amplicon -b ampli_sorted.bam -f reference.fasta
shorah shotgun -b test_aln.cram -f test_ref.fasta
shorah shotgun -a 0.1 -w 42 -x 100000 -p 0.9 -c 0 -r REF:42-272 -R 42 -b test_aln.cram -
↪f ref.fasta
```

CHAPTER

EIGHT

SHORTSTACK

398.1 Introduction

`Shortstack` is a tool for comprehensive annotation and quantification of small RNA genes.

For more information, please check its website: https://biocontainers.pro/tools/shortstack and its home page on Github.

398.2 Versions

- 3.8.5

398.3 Commands

- ShortStack

398.4 Module

You can load the modules by:

```
module load biocontainers
module load shortstack
```

398.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Shortstack on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=shortstack
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers shortstack
```

CHAPTER

NINE

SHOVILL

399.1 Introduction

Shovill is a tool to assemble bacterial isolate genomes from Illumina paired-end reads.

For more information, please check:
Docker hub: https://hub.docker.com/r/staphb/shovill
Home page: https://github.com/tseemann/shovill

399.2 Versions

- 1.1.0

399.3 Commands

- shovill

399.4 Module

You can load the modules by:

```
module load biocontainers
module load shovill
```

399.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run shovill on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=shovill
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers shovill

shovill --outdir out \
    --R1 test/R1.fq.gz \
    --R2 test/R2.fq.gz
```

SICER

400.1 Introduction

`Sicer` is a clustering approach for identification of enriched domains from histone modification ChIP-Seq data.

For more information, please check its website: https://biocontainers.pro/tools/sicer and its home page: http://home.gwu.edu/~wpeng/Software.htm.

400.2 Versions

- 1.1

400.3 Commands

- SICER-df-rb.sh
- SICER-df.sh
- SICER-rb.sh
- SICER.sh

400.4 Module

You can load the modules by:

```
module load biocontainers
module load sicer
```

400.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Sicer on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=sicer
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers sicer

SICER.sh ./ test.bed control.bed . hg18 1 200 150 0.74 600 .01

SICER-rb.sh ./ test.bed . hg18 1 200 150 0.74 400 100
```

CHAPTER

ONE

SICER2

401.1 Introduction

`Sicer2` is the redesigned and improved ChIP-seq broad peak calling tool SICER.

For more information, please check its website: https://biocontainers.pro/tools/sicer2 and its home page on Github.

401.2 Versions

- 1.0.3
- 1.2.0

401.3 Commands

- sicer
- sicer_df
- recognicer
- recognicer_df

401.4 Module

You can load the modules by:

```
module load biocontainers
module load sicer2
```

401.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Sicer2 on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=sicer2
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers sicer2

sicer_df -t ./test/treatment_1.bed ./test/treatment_2.bed \
    -c ./test/control_1.bed ./test/control_2.bed \
    -s hg38 --significant_reads

recognicer_df -t ./test/treatment_1.bed ./test/treatment_2.bed \
    -c ./test/control_1.bed ./test/control_2.bed \
    -s hg38 --significant_reads
```

CHAPTER

TWO

SIGNALP

402.1 Introduction

`SignalP` predicts the presence and location of signal peptide cleavage sites in amino acid sequences from different organisms: Gram-positive prokaryotes, Gram-negative prokaryotes, and eukaryotes.

For more information, please check its home page: https://services.healthtech.dtu.dk/service.php?SignalP-4.1.

402.2 Versions

- 4.1

402.3 Commands

- signalp

402.4 Module

You can load the modules by:

```
module load biocontainers
module load signalp
```

402.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run SignalP on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=signalp
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers signalp

signalp -t gram+ -f all proka.fasta > proka_out
signalp -t euk -f all euk.fasta > euk.out
```

CHAPTER

THREE

SIGNALP6

403.1 Introduction

SignalP predicts the presence and location of signal peptide cleavage sites in amino acid sequences from different organisms: Gram-positive prokaryotes, Gram-negative prokaryotes, and eukaryotes.

For more information, please check:
Home page: https://services.healthtech.dtu.dk/service.php?SignalP

403.2 Versions

- 6.0-fast
- 6.0-slow

403.3 Commands

- signalp6

403.4 Module

You can load the modules by:

```
module load biocontainers
module load signalp6
```

403.5 Example job for fast mode

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run signalp6 on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 2:00:00
#SBATCH -N 1
#SBATCH -n 24
#SBATCH --job-name=signalp6-fast
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers signalp6/6.0-fast

signalp6 --write_procs 24 --fastafile proteins_clean.fasta  \
    --organism euk --output_dir output_fast  \
    --format txt --mode fast
```

403.6 Example job for slow mode

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run signalp6 on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 12:00:00
#SBATCH -N 1
#SBATCH -n 24
#SBATCH --job-name=signalp6-slow
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers signalp6/6.0-slow

signalp6 --write_procs 24 --fastafile proteins_clean.fasta  \
    --organism euk --output_dir output_slow  \
    --format txt --mode slow
```

(continues on next page)

(continued from previous page)

```
signalp6 --write_procs 24 --fastafile proteins_clean.fasta  \
    --organism euk --output_dir output_slow-sequential  \
    --format txt --mode slow-sequential
```

CHAPTER

FOUR

SIMUG

404.1 Introduction

`Simug` is a general-purpose genome simulator.

For more information, please check its website: https://biocontainers.pro/tools/simug and its home page on Github.

404.2 Versions

- 1.0.0

404.3 Commands

- simuG
- vcf2model

404.4 Module

You can load the modules by:

```
module load biocontainers
module load simug
```

404.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Simug on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=simug
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers simug
```

CHAPTER

FIVE

SKEWER

405.1 Introduction

`Skewer` is a fast and accurate adapter trimmer for paired-end reads.

For more information, please check its website: https://biocontainers.pro/tools/skewer and its home page on Github.

405.2 Versions

- 0.2.2

405.3 Commands

- skewer

405.4 Module

You can load the modules by:

```
module load biocontainers
module load skewer
```

405.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Skewer on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=skewer
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers skewer

skewer -l 50 -m pe -o skewerQ30 --mean-quality 30 \
     --end-quality 30 -t 10 -x TruSeq3-PE.fa \
     input_1.fastq input_2.fastq
```

CHAPTER

SIX

SLAMDUNK

406.1 Introduction

Slamdunk is a novel, fully automated software tool for automated, robust, scalable and reproducible SLAMseq data analysis.

For more information, please check:
Docker hub: https://hub.docker.com/r/tobneu/slamdunk
Home page: http://t-neumann.github.io/slamdunk/

406.2 Versions

- 0.4.3

406.3 Commands

- slamdunk
- alleyoop

406.4 Module

You can load the modules by:

```
module load biocontainers
module load slamdunk
```

406.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run slamdunk on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=slamdunk
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers slamdunk
```

CHAPTER

SEVEN

SMOOVE

407.1 Introduction

`Smoove` simplifies and speeds calling and genotyping SVs for short reads.

For more information, please check its website: https://biocontainers.pro/tools/smoove and its home page on Github.

407.2 Versions

- 0.2.7

407.3 Commands

- smoove

407.4 Module

You can load the modules by:

```
module load biocontainers
module load smoove
```

407.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Smoove on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 24
#SBATCH --job-name=smoove
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers smoove

smoove call \
    -x --name my-cohort \
    --exclude hg38_blacklist.bed \
    --fasta  Homo_sapiens.GRCh38.dna.primary_assembly.fa \
     -p 24 \
    --genotype input_bams/*.bam
```

CHAPTER

EIGHT

SNAKEMAKE

408.1 Introduction

`Snakemake` is a workflow engine that provides a readable Python-based workflow definition language and a powerful execution environment that scales from single-core workstations to compute clusters without modifying the workflow.

For more information, please check its website: https://biocontainers.pro/tools/snakemake and its home page: https://snakemake.readthedocs.io/en/stable/.

408.2 Versions

- 6.8.0

408.3 Commands

- snakemake

408.4 Module

You can load the modules by:

```
module load biocontainers
module load snakemake
```

408.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Snakemake on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=snakemake
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers snakemake
```

CHAPTER

NINE

SNAP

409.1 Introduction

Snap is a semi-HMM-based Nucleic Acid Parser – gene prediction tool.

For more information, please check its website: https://biocontainers.pro/tools/snap and its home page: http://korflab.ucdavis.edu/software.html.

409.2 Versions

- 2013_11_29

409.3 Commands

- snap

409.4 Module

You can load the modules by:

```
module load biocontainers
module load snap
```

409.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Snap on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=snap
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers snap
```

SNAP-ALIGNER

410.1 Introduction

`Snap-aligner` (Scalable Nucleotide Alignment Program) is a fast and accurate read aligner for high-throughput sequencing data.

For more information, please check its website: https://biocontainers.pro/tools/snap-aligner and its home page: http://snap.cs.berkeley.edu/.

410.2 Versions

- 2.0.0

410.3 Commands

- snap-aligner

410.4 Module

You can load the modules by:

```
module load biocontainers
module load snap-aligner
```

410.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Snap-aligner on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=snap-aligner
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers snap-aligner
```

CHAPTER

ONE

SNAPTOOLS

411.1 Introduction

`Snaptools` is a python module for pre-processing and working with snap file.

For more information, please check its website: https://biocontainers.pro/tools/snaptools and its home page on Github.

411.2 Versions

- 1.4.8

411.3 Commands

- snaptools

411.4 Module

You can load the modules by:

```
module load biocontainers
module load snaptools
```

411.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Snaptools on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=snaptools
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers snaptools
```

CHAPTER

TWO

SNIPPY

412.1 Introduction

`Snippy` is a tool for rapid haploid variant calling and core genome alignment.

For more information, please check its | Docker hub: https://hub.docker.com/r/staphb/snippy and its home page on Github.

412.2 Versions

- 4.6.0

412.3 Commands

- snippy
- snippy-clean_full_aln
- snippy-core
- snippy-multi
- snippy-vcf_extract_subs
- snippy-vcf_report
- snippy-vcf_to_tab

412.4 Module

You can load the modules by:

```
module load biocontainers
module load snippy
```

412.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Snippy on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=snippy
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers snippy
```

CHAPTER

THREE

SNP-DISTS

413.1 Introduction

Snp-dists is a tool to convert a FASTA alignment to SNP distance matrix.

For more information, please check:
Docker hub: https://hub.docker.com/r/staphb/snp-dists
Home page: https://github.com/tseemann/snp-dists

413.2 Versions

- 0.8.2

413.3 Commands

- snp-dists

413.4 Module

You can load the modules by:

```
module load biocontainers
module load snp-dists
```

413.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run snp-dists on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=snp-dists
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers snp-dists

snp-dists test/good.aln > distances.tab
```

CHAPTER

FOUR

SNPEFF

414.1 Introduction

`Snpeff` is an open source tool that annotates variants and predicts their effects on genes by using an interval forest approach.

For more information, please check its website: https://biocontainers.pro/tools/snpeff and its home page on Github.

414.2 Versions

- 5.1

414.3 Commands

- snpEff

414.4 Module

You can load the modules by:

```
module load biocontainers
module load snpeff
```

414.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Snpeff on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=snpeff
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers snpeff

snpEff GRCh37.75 examples/test.chr22.vcf > test.chr22.ann.vcf
```

CHAPTER

FIVE

SNPGENIE

415.1 Introduction

`Snpgenie` is a collection of Perl scripts for estimating N/S, dN/dS, and gene diversity from next-generation sequencing (NGS) single-nucleotide polymorphism (SNP) variant data.

For more information, please check its website: https://biocontainers.pro/tools/snpgenie and its home page on Github.

415.2 Versions

- 1.0

415.3 Commands

- fasta2revcom.pl
- gtf2revcom.pl
- snpgenie.pl
- snpgenie_between_group.pl
- snpgenie_between_group_processor.pl
- snpgenie_within_group.pl
- snpgenie_within_group_processor.pl
- vcf2revcom.pl

415.4 Module

You can load the modules by:

```
module load biocontainers
module load snpgenie
```

415.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Snpgenie on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=snpgenie
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers snpgenie

snpgenie.pl --minfreq=0.01 --snpreport=CLC_SNP_EXAMPLE.txt \
    --fastafile=REFERENCE_EXAMPLE.fasta --gtffile=CDS_EXAMPLE.gtf
```

CHAPTER

SIX

SNPHYLO

416.1 Introduction

Snphylo is a pipeline to generate a phylogenetic tree from huge SNP data.

For more information, please check:
Docker hub: https://hub.docker.com/r/finchnsnps/snphylo
Home page: https://github.com/thlee/SNPhylo

416.2 Versions

- 20180901

416.3 Commands

- Rscript
- snphylo.sh
- convert_fasta_to_phylip.py
- convert_simple_to_hapmap.py
- determine_bs_tree.R
- draw_unrooted_tree.R
- generate_snp_sequence.R
- remove_low_depth_genotype_data.py
- remove_no_genotype_data.py

416.4 Module

You can load the modules by:

```
module load biocontainers
module load snphylo
```

416.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run snphylo on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=snphylo
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers snphylo
```

CHAPTER

SEVEN

SNPSIFT

417.1 Introduction

`Snpsift` is a tool used to annotate genomic variants using databases, filters, and manipulates genomic annotated variants.

For more information, please check its website: https://biocontainers.pro/tools/snpsift and its home page on Github.

417.2 Versions

- 4.3.1t

417.3 Commands

- SnpSift

417.4 Module

You can load the modules by:

```
module load biocontainers
module load snpsift
```

417.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Snpsift on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=snpsift
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers snpsift

SnpSift annotate -id dbSnp132.vcf \
    variants.vcf > variants_annotated.vcf
```

CHAPTER

EIGHT

SNP-SITES

418.1 Introduction

SNP-sites is a tool that apidly extracts SNPs from a multi-FASTA alignment.

For more information, please check:
Docker hub: https://hub.docker.com/r/staphb/snp-sites
Home page: https://github.com/sanger-pathogens/snp-sites

418.2 Versions

- 2.5.1

418.3 Commands

- snp-sites

418.4 Module

You can load the modules by:

```
module load biocontainers
module load snp-sites
```

418.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run snp-sites on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=snp-sites
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers snp-sites

snp-sites salmonella_serovars_core_genes.aln
```

CHAPTER

NINE

SOAPDENOVO2

419.1 Introduction

`Soapdenovo2` is a short-read assembly method to build de novo draft assembly.

For more information, please check its website: https://biocontainers.pro/tools/soapdenovo2 and its home page: http://soap.genomics.org.cn/soapdenovo.html.

419.2 Versions

- 2.40

419.3 Commands

- SOAPdenovo-127mer
- SOAPdenovo-63mer

419.4 Module

You can load the modules by:

```
module load biocontainers
module load soapdenovo2
```

419.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Soapdenovo2 on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=soapdenovo2
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers soapdenovo2

SOAPdenovo-127mer all -s config_file -K 63 -R -o graph_prefix 1>ass.log 2>ass.err
```

CHAPTER

SORTMERNA

420.1 Introduction

`SortMeRNA` is a local sequence alignment tool for filtering, mapping and clustering.

For more information, please check its website: https://biocontainers.pro/tools/sortmerna and its home page on Github.

420.2 Versions

- 2.1b
- 4.3.4

420.3 Commands

- sortmerna

420.4 Module

You can load the modules by:

```
module load biocontainers
module load sortmerna
```

420.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run SortMeRNA on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=sortmerna
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers sortmerna

sortmerna --ref silva-bac-16s-id90.fasta,silva-bac-16s-db \
    --reads set2_environmental_study_550_amplicon.fasta \
    --fastx --aligned Test
```

CHAPTER

ONE

SOUPORCELL

421.1 Introduction

souporcell is a method for clustering mixed-genotype scRNAseq experiments by individual.

For more information, please check:
Home page: https://github.com/wheaton5/souporcell

421.2 Versions

- 2.0

421.3 Commands

- check_modules.py
- compile_stan_model.py
- consensus.py
- renamer.py
- retag.py
- shared_samples.py
- souporcell.py
- souporcell_pipeline.py

421.4 Module

You can load the modules by:

```
module load biocontainers
module load souporcell
```

421.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run souporcell on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 8
#SBATCH --job-name=souporcell
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers souporcell

souporcell_pipeline.py -i A.merged.bam \
    -b GSM2560245_barcodes.tsv \
    -f refdata-cellranger-GRCh38-3.0.0/fasta/genome.fa \
    -t 8 -o demux_data_test -k 4
```

CHAPTER

TWO

SOURMASH

422.1 Introduction

`Sourmash` is a tool for quickly search, compare, and analyze genomic and metagenomic data sets.

For more information, please check its website: https://biocontainers.pro/tools/sourmash and its home page on Github.

422.2 Versions

- 4.3.0
- 4.5.0

422.3 Commands

- sourmash

422.4 Module

You can load the modules by:

```
module load biocontainers
module load sourmash
```

422.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Sourmash on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=sourmash
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers sourmash

sourmash sketch dna -p k=31 *.fna.gz
sourmash compare *.sig -o cmp.dist
sourmash plot cmp.dist --labels
```

CHAPTER

THREE

SPACERANGER

423.1 Introduction

`Spaceranger` is a set of analysis pipelines that process Visium Spatial Gene Expression data with brightfield and fluorescence microscope images.

For more information, please check its | Docker hub: https://hub.docker.com/r/cumulusprod/spaceranger/tags and its home page:
https://support.10xgenomics.com/spatial-gene-expression/software/pipelines/latest/what-is-space-ranger.

423.2 Versions

- 1.3.0
- 1.3.1
- 2.0.0

423.3 Commands

- spaceranger

423.4 Module

You can load the modules by:

```
module load biocontainers
module load spaceranger
```

423.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Spaceranger on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=spaceranger
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers spaceranger

spaceranger count --id=sample345 \ #Output directory
               --transcriptome=/opt/refdata/GRCh38-2020-A \ #Path to Reference
               --fastqs=/home/jdoe/runs/HAWT7ADXX/outs/fastq_path \ #Path to FASTQs
               --sample=mysample \ #Sample name from FASTQ filename
               --image=/home/jdoe/runs/images/sample345.tiff \ #Path to brightfield image
               --slide=V19J01-123 \ #Slide ID
               --area=A1 \ #Capture area
               --localcores=8 \ #Allowed cores in localmode
               --localmem=64 #Allowed memory (GB) in localmode
```

CHAPTER

FOUR

SPADES

424.1 Introduction

`SPAdes`- St. Petersburg genome assembler - is an assembly toolkit containing various assembly pipelines.

Detailed usage can be found here: https://github.com/ablab/spades

424.2 Versions

- 3.15.3
- 3.15.4
- 3.15.5

424.3 Commands

- coronaspades.py
- metaplasmidspades.py
- metaspades.py
- metaviralspades.py
- plasmidspades.py
- rnaspades.py
- rnaviralspades.py
- spades.py
- spades_init.py
- truspades.py
- spades-bwa
- spades-convert-bin-to-fasta
- spades-core
- spades-corrector-core
- spades-gbuilder

- spades-gmapper
- spades-gsimplifier
- spades-hammer
- spades-ionhammer
- spades-kmer-estimating
- spades-kmercount
- spades-read-filter
- spades-truseq-scfcorrection

424.4 Module

You can load the modules by:

```
module load biocontainers
module load spades
```

424.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run spades on our our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 20:00:00
#SBATCH -N 1
#SBATCH -n 24
#SBATCH --job-name=spades
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers spades

spades.py --pe1-1 SRR11234553_1.fastq --pe1-2 SRR11234553_2.fastq -o spades_out -t 24
```

CHAPTER

FIVE

SPROD

425.1 Introduction

Sprod: De-noising Spatially Resolved Transcriptomics Data Based on Position and Image Information.

For more information, please check:
Home page: https://github.com/yunguan-wang/SPROD

425.2 Versions

- 1.0

425.3 Commands

- python
- python3
- sprod.py

425.4 Module

You can load the modules by:

```
module load biocontainers
module load sprod
```

425.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run sprod on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=sprod
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers sprod

python3 test_examples.py
```

CHAPTER

SIX

SQUEEZEMETA

426.1 Introduction

SqueezeMeta is a fully automated metagenomics pipeline, from reads to bins.

For more information, please check:
Home page: https://github.com/jtamames/SqueezeMeta

426.2 Versions

- 1.5.1

426.3 Commands

- 01.merge_assemblies.pl
- 01.merge_sequential.pl
- 01.remap.pl
- 01.run_assembly.pl
- 01.run_assembly_merged.pl
- 02.rnas.pl
- 03.run_prodigal.pl
- 04.rundiamond.pl
- 05.run_hmmer.pl
- 06.lca.pl
- 07.fun3assign.pl
- 08.blastx.pl
- 09.summarycontigs3.pl
- 10.mapsamples.pl
- 11.mcount.pl

- 12.funcover.pl
- 13.mergeannot2.pl
- 14.runbinning.pl
- 15.dastool.pl
- 16.addtax2.pl
- 17.checkM_batch.pl
- 18.getbins.pl
- 19.getcontigs.pl
- 20.minpath.pl
- 21.stats.pl
- SqueezeMeta.pl
- SqueezeMeta_conf.pl
- SqueezeMeta_conf_original.pl
- parameters.pl
- restart.pl
- add_database.pl
- cover.pl
- sqm2ipath.pl
- sqm2itol.pl
- sqm2keggplots.pl
- sqm2pavian.pl
- sqm_annot.pl
- sqm_hmm_reads.pl
- sqm_longreads.pl
- sqm_mapper.pl
- sqm_reads.pl
- versionchange.pl
- find_missing_markers.pl
- remove_duplicate_markers.pl
- anvi-filter-sqm.py
- anvi-load-sqm.py
- sqm2anvio.pl
- configure_nodb.pl
- configure_nodb_alt.pl
- download_databases.pl
- make_databases.pl

- make_databases_alt.pl
- test_install.pl

426.4 Module

You can load the modules by:

```
module load biocontainers
module load squeezemeta
```

426.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run squeezemeta on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=squeezemeta
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers squeezemeta

SqueezeMeta.pl -m coassembly -p Hadza -s test.samples -f raw
```

CHAPTER

SEVEN

SRA-TOOLKIT

427.1 Introduction

`SRA-Toolkit` is a collection of tools and libraries for using data in the INSDC Sequence Read Archives. Its detailed documentation can be found in https://trace.ncbi.nlm.nih.gov/Traces/sra/sra.cgi?view=toolkit_doc.

427.2 Versions

- 2.11.0-pl5262

427.3 Commands

- abi-dump
- align-cache
- align-info
- bam-load
- cache-mgr
- cg-load
- fasterq-dump
- fasterq-dump-orig
- fastq-dump
- fastq-dump-orig
- illumina-dump
- kar
- kdbmeta
- kget
- latf-load
- md5cp
- prefetch

- prefetch-orig
- rcexplain
- read-filter-redact
- sam-dump
- sam-dump-orig
- sff-dump
- sra-pileup
- sra-pileup-orig
- sra-sort
- sra-sort-cg
- sra-stat
- srapath
- srapath-orig
- sratools
- test-sra
- vdb-config
- vdb-copy
- vdb-diff
- vdb-dump
- vdb-encrypt
- vdb-lock
- vdb-passwd
- vdb-unlock
- vdb-validate

427.4 Module

You can load the modules by:

```
module load biocontainers
module load sra-tools/2.11.0-pl5262
```

427.5 Configuring SRA-Toolkit

Users can config SRA-Toolkit by the command `vdb-config`. For example, the below command set up the current working directory for downloading:

```
vdb-config --prefetch-to-cwd
```

427.6 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run SRA-Toolkit on our cluster:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 8
#SBATCH --job-name=SRA-Toolkit
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers sra-tools/2.11.0-pl5262

vdb-config --prefetch-to-cwd # The data will be downloaded to the current working␣
↪directory.
prefetch SRR11941281
fastq-dump --split-3 SRR11941281/SRR11941281.sra --threads 8
```

CHAPTER

EIGHT

SRST2

428.1 Introduction

Srst2 is designed to take Illumina sequence data, a MLST database and/or a database of gene sequences (e.g. resistance genes, virulence genes, etc) and report the presence of STs and/or reference genes.

For more information, please check:
Docker hub: https://hub.docker.com/r/staphb/srst2
Home page: https://github.com/katholt/srst2

428.2 Versions

- 0.2.0

428.3 Commands

- getmlst.py
- srst2
- slurm_srst2.py

428.4 Module

You can load the modules by:

```
module load biocontainers
module load srst2
```

428.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run srst2 on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=srst2
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers srst2
```

CHAPTER

NINE

STACKS

429.1 Introduction

`Stacks` is a software pipeline for building loci from RAD-seq.

For more information, please check its website: https://biocontainers.pro/tools/stacks and its home page: https://catchenlab.life.illinois.edu/stacks/.

429.2 Versions

- 2.60

429.3 Commands

- clone_filter
- count_fixed_catalog_snps.py
- cstacks
- denovo_map.pl
- gstacks
- integrate_alignments.py
- kmer_filter
- phasedstacks
- populations
- process_radtags
- process_shortreads
- ref_map.pl
- sstacks
- stacks-dist-extract
- stacks-gdb

- stacks-integrate-alignments
- tsv2bam
- ustacks

429.4 Module

You can load the modules by:

```
module load biocontainers
module load stacks
```

429.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Stacks on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 8
#SBATCH --job-name=stacks
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers stacks

denovo_map.pl -T 8 -M 4 -o ./stacks/  \
    --samples ./samples --popmap ./popmaps/popmap
```

CHAPTER

STAR

430.1 Introduction

STAR: ultrafast universal RNA-seq aligner.

Detailed usage can be found here: https://github.com/alexdobin/STAR

430.2 Versions

- 2.7.10a
- 2.7.9a

430.3 Commands

- STAR
- STARlong

430.4 Module

You can load the modules by:

```
module load biocontainers
module load star/2.7.10a
```

430.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run STAR on our our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 20:00:00
#SBATCH -N 1
#SBATCH -n 24
#SBATCH --job-name=star
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers star/2.7.10a


STAR  --runThreadN 24  --runMode genomeGenerate  --genomeDir ref_genome  --
↪genomeFastaFiles ref_genome.fasta

STAR --runThreadN 24 --genomeDir ref_genome --readFilesIn seq_1.fastq seq_2.fastq  --
↪outSAMtype BAM SortedByCoordinate --outWigType wiggle read2
```

CHAPTER

ONE

STARAMR

431.1 Introduction

staramr scans bacterial genome contigs against the ResFinder, PointFinder, and PlasmidFinder databases (used by the ResFinder webservice and other webservices offered by the Center for Genomic Epidemiology) and compiles a summary report of detected antimicrobial resistance genes.

For more information, please check:
Docker hub: https://hub.docker.com/r/staphb/staramr
Home page: https://github.com/phac-nml/staramr

431.2 Versions

- 0.7.1

431.3 Commands

- staramr

431.4 Module

You can load the modules by:

```
module load biocontainers
module load staramr
```

431.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run staramr on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=staramr
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers staramr

staramr db info
staramr search \
    --pointfinder-organism salmonella \
    -o out *.fasta
```

CHAPTER

TWO

STAR-FUSION

432.1 Introduction

`STAR-Fusion` is a component of the Trinity Cancer Transcriptome Analysis Toolkit (CTAT).

For more information, please check its | Docker hub: https://hub.docker.com/r/trinityctat/starfusion and its home page on Github.

432.2 Versions

- 1.11b

432.3 Commands

- STAR-Fusion

432.4 Module

You can load the modules by:

```
module load biocontainers
module load starfusion
```

432.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run STAR-Fusion on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 24
#SBATCH --job-name=starfusion
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers starfusion

STAR-Fusion --CPU 24 --left_fq ../star/SRR12095148_1.fastq --right_fq  ../star/
↪SRR12095148_2.fastq\
     --genome_lib_dir  GRCh38_gencode_v33_CTAT_lib_Apr062020.plug-n-play/ctat_genome_lib_
↪build_dir \
     --FusionInspector validate \
     --denovo_reconstruct \
     --examine_coding_effect \
     --output_dir STAR-Fusion-output
```

CHAPTER

THREE

STREAM

433.1 Introduction

STREAM (Single-cell Trajectories Reconstruction, Exploration And Mapping) is an interactive pipeline capable of disentangling and visualizing complex branching trajectories from both single-cell transcriptomic and epigenomic data.

For more information, please check its | Docker hub: https://hub.docker.com/r/pinellolab/stream and its home page on Github.

433.2 Versions

- 1.0

433.3 Commands

- python
- python3

433.4 Module

You can load the modules by:

```
module load biocontainers
module load stream
```

433.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run STREAM on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=stream
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers stream
```

CHAPTER

FOUR

STRINGTIE

434.1 Introduction

`StringTie`: efficient transcript assembly and quantitation of RNA-Seq data.

Stringtie employs efficient algorithms for transcript structure recovery and abundance estimation from bulk RNA-Seq reads aligned to a reference genome. It takes as input spliced alignments in coordinate-sorted SAM/BAM/CRAM format and produces a GTF output which consists of assembled transcript structures and their estimated expression levels (FPKM/TPM and base coverage values).

Detailed usage can be found here: https://github.com/gpertea/stringtie

434.2 Versions

- 2.1.7
- 2.2.1

434.3 Commands

- stringtie

434.4 Module

You can load the modules by:

```
module load biocontainers
module load stringtie
```

434.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run stringtie on our our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 20:00:00
#SBATCH -N 1
#SBATCH -n 24
#SBATCH --job-name=stringtie
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers stringtie

stringtie -o SRR11614710.gtf -G Homo_sapiens.GRCh38.105.gtf SRR11614710Aligned.
↪sortedByCoord.out.bam
```

CHAPTER

FIVE

STRIQUE

435.1 Introduction

STRique is a python package to analyze repeat expansion and methylation states of short tandem repeats (STR) in Oxford Nanopore Technology (ONT) long read sequencing data.

For more information, please check:
Docker hub: https://hub.docker.com/r/giesselmann/strique
Home page: https://github.com/giesselmann/STRique

435.2 Versions

- 0.4.2

435.3 Commands

- STRique.py
- STRique_test.py
- fast5Masker.py

435.4 Module

You can load the modules by:

```
module load biocontainers
module load strique
```

435.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run strique on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=strique
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers strique

STRique_test.py
STRique.py index data/ > data/reads.fofn
cat data/c9orf72.sam |  STRique.py count ./data/reads.fofn ./models/r9_4_450bps.model ./
↪configs/repeat_config.tsv --config ./configs/STRique.json
```

CHAPTER

SIX

STRUCTURE

436.1 Introduction

Structure is a software package for using multi-locus genotype data to investigate population structure.

For more information, please check:
Home page: https://web.stanford.edu/group/pritchardlab/structure.html

436.2 Versions

- 2.3.4

436.3 Commands

- structure

436.4 Module

You can load the modules by:

```
module load biocontainers
module load structure
```

436.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run structure on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=structure
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers structure
```

CHAPTER

SEVEN

SUBREAD

437.1 Introduction

`Subread` carries out high-performance read alignment, quantification and mutation discovery. It is a general-purpose read aligner which can be used to map both genomic DNA-seq reads and DNA-seq reads. It uses a new mapping paradigm called seed-and-vote to achieve fast, accurate and scalable read mapping. Subread automatically determines if a read should be globally or locally aligned, therefore particularly powerful in mapping RNA-seq reads. It supports INDEL detection and can map reads with both fixed and variable lengths.

For more information, please check its website: https://biocontainers.pro/tools/subread and its home page: http://subread.sourceforge.net.

437.2 Versions

- 1.6.4
- 2.0.1

437.3 Commands

- detectionCall
- exactSNP
- featureCounts
- flattenGTF
- genRandomReads
- propmapped
- qualityScores
- removeDup
- repair
- subindel
- subjunc

- sublong
- subread-align
- subread-buildindex
- subread-fullscan
- txUnique

437.4 Module

You can load the modules by:

```
module load biocontainers
module load subread
```

437.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Subread on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 4
#SBATCH --job-name=subread
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers subread

featureCounts -s 2 -p -Q 10 -T 4 -a genome.gtf -o featurecounts.txt mapped.bam
```

CHAPTER

EIGHT

SURVIVOR

438.1 Introduction

SURVIVOR is a tool set for simulating/evaluating SVs, merging and comparing SVs within and among samples, and includes various methods to reformat or summarize SVs.

For more information, please check:
BioContainers: https://biocontainers.pro/tools/survivor
Home page: https://github.com/fritzsedlazeck/SURVIVOR

438.2 Versions

- 1.0.7

438.3 Commands

- SURVIVOR

438.4 Module

You can load the modules by:

```
module load biocontainers
module load survivor
```

438.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run survivor on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=survivor
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers survivor

SURVIVOR simSV parameter_file
SURVIVOR simSV ref.fa parameter_file 0.1 0 simulated
SURVIVOR eval caller.vcf simulated.bed 10 eval_res
```

~

CHAPTER

NINE

SVABA

439.1 Introduction

SvABA is a method for detecting structural variants in sequencing data using genome-wide local assembly.

For more information, please check:
BioContainers: https://biocontainers.pro/tools/svaba
Home page: https://github.com/walaj/svaba

439.2 Versions

- 1.1.0

439.3 Commands

- svaba

439.4 Module

You can load the modules by:

```
module load biocontainers
module load svaba
```

439.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run svaba on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 8
#SBATCH --job-name=svaba
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers svaba

DBSNP=dbsnp_indel.vcf
TUM_BAM=G15512.HCC1954.1.COST16011_region.bam
NORM_BAM=HCC1954.NORMAL.30x.compare.COST16011_region.bam
CORES=8 ## set any number of cores
REF=Homo_sapiens_assembly19.COST16011_region.fa
svaba run -t $TUM_BAM -n $NORM_BAM \
    -p $CORES -D $DBSNP \
    -a somatic_run -G $REF
```

SVTOOLS

440.1 Introduction

Svtools is a suite of utilities designed to help bioinformaticians construct and explore cohort-level structural variation calls.

For more information, please check:
Docker hub: https://hub.docker.com/r/halllab/svtools
Home page: https://github.com/hall-lab/svtools

440.2 Versions

- 0.5.1

440.3 Commands

- svtools

440.4 Module

You can load the modules by:

```
module load biocontainers
module load svtools
```

440.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run svtools on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=svtools
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers svtools
```

CHAPTER

ONE

SVTYPER

441.1 Introduction

SVTyper performs breakpoint genotyping of structural variants (SVs) using whole genome sequencing data. svtyper is the original implementation of the genotyping algorithm, and works with multiple samples. svtyper-sso is an alternative implementation of svtyper that is optimized for genotyping a single sample. svtyper-sso is a parallelized implementation of svtyper that takes advantage of multiple CPU cores via the multiprocessing module. svtyper-sso can offer a 2x or more speedup (depending on how many CPU cores used) in genotyping a single sample. NOTE: svtyper-sso is not yet stable. There are minor logging differences between the two and svtyper-sso may exit with an error prematurely when processing CRAM files.

For more information, please check:
BioContainers: https://biocontainers.pro/tools/svtyper
Home page: https://github.com/hall-lab/svtyper

441.2 Versions

- 0.7.1

441.3 Commands

- svtyper
- svtyper-sso
- python
- python2

441.4 Module

You can load the modules by:

```
module load biocontainers
module load svtyper
```

441.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run svtyper on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=svtyper
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers svtyper

svtyper \
    -i data/example.vcf \
    -B data/NA12878.target_loci.sorted.bam \
    -l data/NA12878.bam.json \
    > out.vcf
```

CHAPTER

TWO

SWAT

442.1 Introduction

`swat` is a program for searching one or more DNA or protein query sequences, or a query profile, against a sequence database, using an efficient implementation of the Smith-Waterman or Needleman-Wunsch algorithms with linear (affine) gap penalties.

For more information, please check its home page: http://www.phrap.org/phredphrapconsed.html#block_phrap.

442.2 Versions

- 1.090518

442.3 Commands

- swat

442.4 Module

You can load the modules by:

```
module load biocontainers
module load swat
```

442.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run swat on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=swat
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers swat
```

CHAPTER

THREE

SYRI

443.1 Introduction

Syri compares alignments between two chromosome-level assemblies and identifies synteny and structural rearrangements.

For more information, please check:
Home page: https://github.com/schneebergerlab/syri

443.2 Versions

- 1.6

443.3 Commands

- syri

443.4 Module

You can load the modules by:

```
module load biocontainers
module load syri
```

443.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run syri on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=syri
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers syri

syri -c out.sam -r refgenome -q qrygenome -k -F S
```

CHAPTER

FOUR

TALON

444.1 Introduction

`Talon` is a Python package for identifying and quantifying known and novel genes/isoforms in long-read transcriptome data sets.

For more information, please check its website: https://biocontainers.pro/tools/talon and its home page on Github.

444.2 Versions

- 5.0

444.3 Commands

- talon
- talon_abundance
- talon_create_GTF
- talon_fetch_reads
- talon_filter_transcripts
- talon_generate_report
- talon_get_sjs
- talon_initialize_database
- talon_label_reads
- talon_reformat_gtf
- talon_summarize

444.4 Module

You can load the modules by:

```
module load biocontainers
module load talon
```

444.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Talon on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=talon
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers talon
```

CHAPTER

FIVE

TARGETP

445.1 Introduction

TargetP-2.0 tool predicts the presence of N-terminal presequences: signal peptide (SP), mitochondrial transit peptide (mTP), chloroplast transit peptide (cTP) or thylakoid luminal transit peptide (luTP). For the sequences predicted to contain an N-terminal presequence a potential cleavage site is also predicted.

For more information, please check:
Home page: https://services.healthtech.dtu.dk/service.php?TargetP-2.0

445.2 Versions

- 2.0

445.3 Commands

- targetp

445.4 Module

You can load the modules by:

```
module load biocontainers
module load targetp
```

445.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run targetp on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=targetp
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers targetp
```

CHAPTER

SIX

TASSEL

446.1 Introduction

TASSEL is a software package used to evaluate traits associations, evolutionary patterns, and linkage disequilibrium.

For more information, please check:
Home page: https://www.maizegenetics.net/tassel

446.2 Versions

- 5.0

446.3 Commands

- run_pipeline.pl
- start_tassel.pl
- Tassel5

446.4 Module

You can load the modules by:

```
module load biocontainers
module load tassel
```

446.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run tassel on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=tassel
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers tassel
```

CHAPTER

SEVEN

TAXONKIT

447.1 Introduction

`Taxonkit` is a practical and efficient NCBI taxonomy toolkit.

For more information, please check its website: https://biocontainers.pro/tools/taxonkit and its home page on Github.

447.2 Versions

- 0.9.0

447.3 Commands

- taxonkit

447.4 Module

You can load the modules by:

```
module load biocontainers
module load taxonkit
```

447.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Taxonkit on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=taxonkit
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers taxonkit

taxonkit list --show-rank --show-name --indent "    " --ids 9605,239934
```

CHAPTER

EIGHT

T-COFFEE

448.1 Introduction

`T-coffee` is a multiple sequence alignment software using a progressive approach.

For more information, please check its website: https://biocontainers.pro/tools/t-coffee and its home page on Github.

448.2 Versions

- 13.45.0.4846264

448.3 Commands

- t_coffee

448.4 Module

You can load the modules by:

```
module load biocontainers
module load t-coffee
```

448.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run T-coffee on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=t-coffee
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers t-coffee

t_coffee  OG0002077.fa -mode  expresso
```

CHAPTER

NINE

TETRANSCRIPTS

449.1 Introduction

`Tetranscripts` is a package for including transposable elements in differential enrichment analysis of sequencing datasets.

For more information, please check its website: https://biocontainers.pro/tools/tetranscripts and its home page on Github.

449.2 Versions

- 2.2.1

449.3 Commands

- TEtranscripts
- TEcount

449.4 Module

You can load the modules by:

```
module load biocontainers
module load tetranscripts
```

449.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Tetranscripts on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=tetranscripts
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers tetranscripts

TEtranscripts --format BAM --mode multi \
    -t treatment_sample1.bam treatment_sample2.bam treatment_sample3.bam \
    -c control_sample1.bam control_sample2.bam control_sample3.bam \
    --GTF genic-GTF-file \
    --GTF genic-GTF-file \
    --project sample_nosort_test
```

TIARA

450.1 Introduction

`Tiara` is a deep-learning-based approach for identification of eukaryotic sequences in the metagenomic data powered by PyTorch.

For more information, please check its | Docker hub: https://hub.docker.com/r/zhan4429/tiara and its home page on Github.

450.2 Versions

- 1.0.2

450.3 Commands

- tiara

450.4 Module

You can load the modules by:

```
module load biocontainers
module load tiara
```

450.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Tiara on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 24
#SBATCH --job-name=tiara
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers tiara

tiara -t 24 -i archaea_fr.fasta -o archaea_out.txt
tiara -t 24 -i bacteria_fr.fasta -o bacteria_out.txt
tiara -t 24 -i eukarya_fr.fasta -o eukarya_out.txt
tiara -t 24 -i mitochondria_fr.fasta -o mitochondria_out.txt
tiara -t 24  -i plast_fr.fasta -o plast_out.txt
tiara -t 24  -i total.fasta -o mix_out.txt  --tf all  -p 0.65 0.60 --probabilities
```

CHAPTER

ONE

TIGMINT

451.1 Introduction

Tigmint identifies and corrects misassemblies using linked (e.g. MGI's stLFR, 10x Genomics Chromium) or long (e.g. Oxford Nanopore Technologies long reads) DNA sequencing reads. The reads are first aligned to the assembly, and the extents of the large DNA molecules are inferred from the alignments of the reads. The physical coverage of the large molecules is more consistent and less prone to coverage dropouts than that of the short read sequencing data. The sequences are cut at positions that have insufficient spanning molecules. Tigmint outputs a BED file of these cut points, and a FASTA file of the cut sequences.

For more information, please check:
Home page: https://github.com/bcgsc/tigmint

451.2 Versions

- 1.2.6

451.3 Commands

- tigmint
- tigmint-arcs-tsv
- tigmint-cut
- tigmint-make
- tigmint_estimate_dist.py
- tigmint_molecule.py
- tigmint_molecule_paf.py

451.4 Module

You can load the modules by:

```
module load biocontainers
module load tigmint
```

451.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run tigmint on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=tigmint
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers tigmint
```

CHAPTER

TWO

TOBIAS

452.1 Introduction

`Tobias` is a collection of command-line bioinformatics tools for performing footprinting analysis on ATAC-seq data.

For more information, please check its website: https://biocontainers.pro/tools/tobias and its home page on Github.

452.2 Versions

- 0.13.3-py37

452.3 Commands

- TOBIAS

452.4 Module

You can load the modules by:

```
module load biocontainers
module load tobias
```

452.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Tobias on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 8
#SBATCH --job-name=tobias
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers tobias

TOBIAS DownloadData --bucket data-tobias-2020
mv data-tobias-2020/ test_data/

TOBIAS PlotAggregate --TFBS test_data/BATF_all.bed \
     --signals test_data/Bcell_corrected.bw test_data/Tcell_corrected.bw \
     --output BATFJUN_footprint_comparison_all.pdf \
     --share_y both --plot_boundaries --signal-on-x

TOBIAS BINDetect --motifs test_data/motifs.jaspar \
     --signals test_data/Bcell_footprints.bw test_data/Tcell_footprints.bw \
     --genome test_data/genome.fa.gz \
     --peaks test_data/merged_peaks_annotated.bed \
     --peak_header test_data/merged_peaks_annotated_header.txt \
     --outdir BINDetect_output --cond_names Bcell Tcell --cores 8

TOBIAS ATACorrect --bam test_data/Bcell.bam \
    --genome test_data/genome.fa.gz \
    --peaks test_data/merged_peaks.bed \
    --blacklist test_data/blacklist.bed \
    --outdir ATACorrect_test --cores 8

TOBIAS FootprintScores --signal test_data/Bcell_corrected.bw \
    --regions test_data/merged_peaks.bed \
    --output Bcell_footprints.bw --cores 8
```

CHAPTER

THREE

TOMBO

453.1 Introduction

Tombo is a suite of tools primarily for the identification of modified nucleotides from nanopore sequencing data. Tombo also provides tools for the analysis and visualization of raw nanopore signal.

For more information, please check its website: https://biocontainers.pro/tools/ont-tombo and its home page on Github.

453.2 Versions

- 1.5.1

453.3 Commands

- tombo

453.4 Module

You can load the modules by:

```
module load biocontainers
module load tombo
```

453.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Tombo on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 4
#SBATCH --job-name=tombo
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers tombo

tombo resquiggle path/to/fast5s/ genome.fasta --processes 4 --num-most-common-errors 5
tombo detect_modifications alternative_model --fast5-basedirs path/to/fast5s/ \
    --statistics-file-basename native.e_coli_sample \
    --alternate-bases dam dcm --processes 4

# plot raw signal at most significant dcm locations
tombo plot most_significant --fast5-basedirs path/to/fast5s/ \
    --statistics-filename native.e_coli_sample.dcm.tombo.stats \
    --plot-standard-model --plot-alternate-model dcm \
    --pdf-filename sample.most_significant_dcm_sites.pdf

# produces wig file with estimated fraction of modified reads at each valid reference␣
↪site
tombo text_output browser_files --statistics-filename native.e_coli_sample.dam.tombo.
↪stats \
     --file-types dampened_fraction --browser-file-basename native.e_coli_sample.dam
# also produce successfully processed reads coverage file for reference
tombo text_output browser_files --fast5-basedirs path/to/fast5s/ \
    --file-types coverage --browser-file-basename native.e_coli_sample
```

CHAPTER

FOUR

TOPHAT

454.1 Introduction

`TopHat` is a fast splice junction mapper for RNA-Seq reads. It aligns RNA-Seq reads to mammalian-sized genomes using the ultra high-throughput short read aligner Bowtie, and then analyzes the mapping results to identify splice junctions between exons.

For more information, please check its website: https://biocontainers.pro/tools/tophat and its home page: https://ccb.jhu.edu/software/tophat/index.shtml.

454.2 Versions

- 2.1.1-py27

454.3 Commands

- tophat
- tophat2

454.4 Module

You can load the modules by:

```
module load biocontainers
module load tophat
```

454.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run TopHat on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=tophat
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers tophat

tophat -r 20 test_ref reads_1.fq reads_2.fq
```

CHAPTER

FIVE

TPMCALCULATOR

455.1 Introduction

`TPMCalculator` quantifies mRNA abundance directly from the alignments by parsing BAM files.

Detailed usage can be found here: https://github.com/ncbi/TPMCalculator

455.2 Versions

- 0.0.3
- 0.0.4

455.3 Commands

- TPMCalculator

455.4 Module

You can load the modules by:

```
module load biocontainers
module load tpmcalculator
```

455.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run tpmcalculator on our our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 10:00:00
#SBATCH -N 1
#SBATCH -n 12
#SBATCH --job-name=tpmcalculator
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers transdecoder

TPMCalculator -g Homo_sapiens.GRCh38.105.chr.gtf -b SRR12095148Aligned.sortedByCoord.out.
↪bam
```

CHAPTER

SIX

TRANSABYSS

456.1 Introduction

`Transabyss` is a tool for De novo assembly of RNAseq data using ABySS.

For more information, please check its website: https://bioconda.github.io/recipes/transabyss and its home page on Github.

456.2 Versions

- 2.0.1

456.3 Commands

- transabyss
- transabyss-merge

456.4 Module

You can load the modules by:

```
module load biocontainers
module load transabyss
```

456.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Transabyss on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 12
#SBATCH --job-name=transabyss
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers transabyss

transabyss --name  SRR12095148 \
    --pe SRR12095148_1.fastq SRR12095148_2.fastq \
    --outdir  SRR12095148_assembly  --threads 12
```

CHAPTER

SEVEN

TRANSDECODER

457.1 Introduction

`TransDecoder` identifies candidate coding regions within transcript sequences, such as those generated by de novo RNA-Seq transcript assembly using Trinity, or constructed based on RNA-Seq alignments to the genome using Tophat and Cufflinks.

- TransDecoder identifies likely coding sequences based on the following criteria:
- a minimum length open reading frame (ORF) is found in a transcript sequence
- a log-likelihood score similar to what is computed by the GeneID software is > 0.
- the above coding score is greatest when the ORF is scored in the 1st reading frame as compared to scores in the other 2 forward reading frames.
- if a candidate ORF is found fully encapsulated by the coordinates of another candidate ORF, the longer one is reported. However, a single transcript can report multiple ORFs (allowing for operons, chimeras, etc).
- a PSSM is built/trained/used to refine the start codon prediction.
- **optional** the putative peptide has a match to a Pfam domain above the noise cutoff score.

Detailed usage can be found here: https://github.com/TransDecoder/TransDecoder/wiki#running-transdecoder

457.2 Versions

- 5.5.0

457.3 Commands

- TransDecoder.LongOrfs
- TransDecoder.Predict
- cdna_alignment_orf_to_genome_orf.pl
- compute_base_probs.pl
- exclude_similar_proteins.pl
- fasta_prot_checker.pl
- ffindex_resume.pl

- gene_list_to_gff.pl
- get_FL_accs.pl
- get_longest_ORF_per_transcript.pl
- get_top_longest_fasta_entries.pl
- gff3_file_to_bed.pl
- gff3_file_to_proteins.pl
- gff3_gene_to_gtf_format.pl
- gtf_genome_to_cdna_fasta.pl
- gtf_to_alignment_gff3.pl
- gtf_to_bed.pl
- nr_ORFs_gff3.pl
- pfam_runner.pl
- refine_gff3_group_iso_strip_utrs.pl
- refine_hexamer_scores.pl
- remove_eclipsed_ORFs.pl
- score_CDS_likelihood_all_6_frames.pl
- select_best_ORFs_per_transcript.pl
- seq_n_baseprobs_to_loglikelihood_vals.pl
- start_codon_refinement.pl
- train_start_PWM.pl
- uri_unescape.pl

457.4 Module

You can load the modules by:

```
module load biocontainers
module load transdecoder
```

457.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run transdecoder on our our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 20:00:00
#SBATCH -N 1
#SBATCH -n 24
#SBATCH --job-name=transdecoder
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers transdecoder

gtf_genome_to_cdna_fasta.pl transcripts.gtf test.genome.fasta > transcripts.fasta
gtf_to_alignment_gff3.pl transcripts.gtf > transcripts.gff3
TransDecoder.LongOrfs -t transcripts.fasta
TransDecoder.Predict -t transcripts.fasta
```

CHAPTER

EIGHT

TRANSRATE

458.1 Introduction

Transrate is software for de-novo transcriptome assembly quality analysis.

For more information, please check:
Docker hub: https://hub.docker.com/r/arnaudmeng/transrate
Home page: http://hibberdlab.com/transrate/

458.2 Versions

- 1.0.3

458.3 Commands

- transrate

458.4 Module

You can load the modules by:

```
module load biocontainers
module load transrate
```

458.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run transrate on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 12
#SBATCH --job-name=transrate
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers transrate

transrate --assembly mm10/Mus_musculus.GRCm38.cds.all.fa \
    --left seq_1.fq.gz \
    --right seq_2.fq.gz \
    --threads 12
```

CHAPTER

NINE

TRANSVAR

459.1 Introduction

`Transvar` is a multi-way annotator for genetic elements and genetic variations.

For more information, please check its | Docker hub: https://hub.docker.com/r/zhouwanding/transvar and its home page: https://bioinformatics.mdanderson.org/public-software/transvar/.

459.2 Versions

- 2.5.9

459.3 Commands

- transvar

459.4 Module

You can load the modules by:

```
module load biocontainers
module load transvar
```

459.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Transvar on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=transvar
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers transvar

# set up databases
transvar config --download_anno --refversion hg19

# in case you don't have a reference
transvar config --download_ref --refversion hg19

transvar panno -i 'PIK3CA:p.E545K' --ucsc --ccds
```

TRAX

460.1 Introduction

`tRAX` (tRNA Analysis of eXpression) is a software package built for in-depth analyses of tRNA-derived small RNAs (tDRs), mature tRNAs, and inference of RNA modifications from high-throughput small RNA sequencing data.

For more information, please check its | Docker hub: https://hub.docker.com/r/ucsclowelab/trax and its home page on Github.

460.2 Versions

- 1.0.0

460.3 Commands

- TestRun.bash
- quickdb.bash
- maketrnadb.py
- trimadapters.py
- processamples.py

460.4 Module

You can load the modules by:

```
module load biocontainers
module load trax
```

460.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run tRAX on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=trax
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers trax
```

CHAPTER

ONE

TREETIME

461.1 Introduction

`Treetime` is a tool for maximum likelihood dating and ancestral sequence inference.

For more information, please check its website: https://biocontainers.pro/tools/treetime and its home page on Github.

461.2 Versions

- 0.8.6
- 0.9.4

461.3 Commands

- treetime

461.4 Module

You can load the modules by:

```
module load biocontainers
module load treetime
```

461.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Treetime on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=treetime
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers treetime

treetime ancestral --aln input.fasta --tree input.nwk
```

CHAPTER

TWO

TRIMAL

462.1 Introduction

`Trimal` is a tool for the automated removal of spurious sequences or poorly aligned regions from a multiple sequence alignment.

For more information, please check its website: https://biocontainers.pro/tools/trimal and its home page: http://trimal.cgenomics.org.

462.2 Versions

- 1.4.1

462.3 Commands

- trimal
- readal
- statal

462.4 Module

You can load the modules by:

```
module load biocontainers
module load trimal
```

462.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Trimal on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=trimal
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers trimal

trimal -in input.fasta -out output1 -htmlout output1.html -gt 1
```

CHAPTER

THREE

TRIM-GALORE

463.1 Introduction

`Trim-galore` is a wrapper tool that automates quality and adapter trimming to FastQ files.

For more information, please check its website: https://biocontainers.pro/tools/trim-galore and its home page: https://www.bioinformatics.babraham.ac.uk/projects/trim_galore/.

463.2 Versions

- 0.6.7

463.3 Commands

- trim_galore

463.4 Module

You can load the modules by:

```
module load biocontainers
module load trim-galore
```

463.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Trim-galore on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 4
#SBATCH --job-name=trim-galore
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers trim-galore

trim_galore  --paired --fastqc --length 20 -o sample1_trimmed Sample1_1.fq Sample1_2.fq
```

CHAPTER

FOUR

TRIMMOMATIC

464.1 Introduction

`Trimmomatic` is a flexible read trimming tool for Illumina NGS data.

For more information, please check its website: https://biocontainers.pro/tools/trimmomatic and its home page: http://www.usadellab.org/cms/index.php?page=trimmomatic.

464.2 Versions

- 0.39

464.3 Commands

- trimmomatic

464.4 Module

You can load the modules by:

```
module load biocontainers
module load trimmomatic
```

464.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Trimmomatic on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 8
#SBATCH --job-name=trimmomatic
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers trimmomatic

trimmomatic PE -threads 8 \
    input_forward.fq.gz input_reverse.fq.gz \
    output_forward_paired.fq.gz output_forward_unpaired.fq.gz \
    output_reverse_paired.fq.gz output_reverse_unpaired.fq.gz \
    ILLUMINACLIP:TruSeq3-PE.fa:2:30:10:2:True LEADING:3 TRAILING:3 MINLEN:36
```

CHAPTER

FIVE

TRINITY

465.1 Introduction

`Trinity` assembles transcript sequences from Illumina RNA-Seq data.

For more information, please check its website: https://biocontainers.pro/tools/trinity and its home page on Github.

465.2 Versions

- 2.12.0
- 2.13.2
- 2.14.0

465.3 Commands

- Trinity
- TrinityStats.pl
- Trinity_gene_splice_modeler.py
- ace2sam
- align_and_estimate_abundance.pl
- analyze_blastPlus_topHit_coverage.pl
- analyze_diff_expr.pl
- blast2sam.pl
- bowtie
- bowtie2
- bowtie2-build
- bowtie2-inspect
- bowtie2sam.pl
- contig_ExN50_statistic.pl

- define_clusters_by_cutting_tree.pl
- export2sam.pl
- extract_supertranscript_from_reference.py
- filter_low_expr_transcripts.pl
- get_Trinity_gene_to_trans_map.pl
- insilico_read_normalization.pl
- interpolate_sam.pl
- jellyfish
- novo2sam.pl
- retrieve_sequences_from_fasta.pl
- run_DE_analysis.pl
- sam2vcf.pl
- samtools
- samtools.pl
- seq_cache_populate.pl
- seqtk-trinity
- sift_bam_max_cov.pl
- soap2sam.pl
- tabix
- trimmomatic
- wgsim
- wgsim_eval.pl
- zoom2sam.pl

465.4 Module

You can load the modules by:

```
module load biocontainers
module load trinity
```

465.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Trinity on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 6
#SBATCH --job-name=trinity
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers trinity

Trinity --seqType fq --left reads_1.fq --right reads_2.fq \
    --CPU 6 --max_memory 20G
```

CHAPTER

SIX

TRINOTATE

466.1 Introduction

`Trinotate` is a comprehensive annotation suite designed for automatic functional annotation of transcriptomes, particularly de novo assembled transcriptomes, from model or non-model organisms.

For more information, please check its website: https://biocontainers.pro/tools/trinotate and its home page on Github.

466.2 Versions

- 3.2.2

466.3 Commands

- Trinotate
- Build_Trinotate_Boilerplate_SQLite_db.pl
- EMBL_dat_to_Trinotate_sqlite_resourceDB.pl
- EMBL_swissprot_parser.pl
- PFAM_dat_parser.pl
- PFAMtoGoParser.pl
- RnammerTranscriptome.pl
- TrinotateSeqLoader.pl
- Trinotate_BLAST_loader.pl
- Trinotate_GO_to_SLIM.pl
- Trinotate_GTF_loader.pl
- Trinotate_GTF_or_GFF3_annot_prep.pl
- Trinotate_PFAM_loader.pl
- Trinotate_RNAMMER_loader.pl
- Trinotate_SIGNALP_loader.pl

- Trinotate_TMHMM_loader.pl
- Trinotate_get_feature_name_encoding_attributes.pl
- Trinotate_report_writer.pl
- assign_eggnog_funccats.pl
- autoTrinotate.pl
- build_DE_cache_tables.pl
- cleanMe.pl
- cleanme.pl
- count_table_fields.pl
- create_clusters_tables.pl
- extract_GO_assignments_from_Trinotate_xls.pl
- extract_GO_for_BiNGO.pl
- extract_specific_genes_from_all_matrices.pl
- import_DE_results.pl
- import_Trinotate_xls_as_annot.pl
- import_expression_and_DE_results.pl
- import_expression_matrix.pl
- import_samples_n_expression_matrix.pl
- import_samples_only.pl
- import_transcript_annotations.pl
- import_transcript_clusters.pl
- import_transcript_names.pl
- init_Trinotate_sqlite_db.pl
- legacy_blast.pl
- make_cXp_html.pl
- obo_tab_to_sqlite_db.pl
- obo_to_tab.pl
- prep_nuc_prot_set_for_trinotate_loading.pl
- print.pl
- rnammer_supperscaffold_gff_to_indiv_transcripts.pl
- runMe.pl
- run_TrinotateWebserver.pl
- run_cluster_functional_enrichment_analysis.pl
- shrink_db.pl
- sqlite.pl
- superScaffoldGenerator.pl

- test_Barplot.pl
- test_GO_DAG.pl
- test_GenomeBrowser.pl
- test_Heatmap.pl
- test_Lineplot.pl
- test_Piechart.pl
- test_Scatter2D.pl
- test_Sunburst.pl
- trinotate_report_summary.pl
- update_blastdb.pl
- update_seq_n_annotation_fields.pl

466.4 Module

You can load the modules by:

```
module load biocontainers
module load trinotate
```

466.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Trinotate on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=trinotate
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers trinotate

sqlite_db="myTrinotate.sqlite"

Trinotate ${sqlite_db} init \
    --gene_trans_map data/Trinity.fasta.gene_to_trans_map \
```

(continues on next page)

(continued from previous page)

```
    --transcript_fasta data/Trinity.fasta \
     --transdecoder_pep \
    data/Trinity.fasta.transdecoder.pep

Trinotate ${sqlite_db} LOAD_swissprot_blastp data/swissprot.blastp.outfmt6

Trinotate ${sqlite_db} LOAD_pfam data/TrinotatePFAM.out
```

CHAPTER

SEVEN

TRNASCAN-SE

467.1 Introduction

`Trnascan-se` is a convenient, ready-for-use means to identify tRNA genes in one or more query sequences.

For more information, please check its website: https://biocontainers.pro/tools/trnascan-se and its home page: http://lowelab.ucsc.edu/tRNAscan-SE/.

467.2 Versions

- 2.0.9

467.3 Commands

- tRNAscan-SE

467.4 Module

You can load the modules by:

```
module load biocontainers
module load trnascan-se
```

467.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Trnascan-se on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 12
#SBATCH --job-name=trnascan-se
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers trnascan-se

tRNAscan-SE --thread 12 -o tRNA.out \
    -f rRNA.ss -m tRNA.stats genome.fasta
```

CHAPTER

EIGHT

TRUST4

468.1 Introduction

Tcr Receptor Utilities for Solid Tissue (TRUST) is a computational tool to analyze TCR and BCR sequences using unselected RNA sequencing data, profiled from solid tissues, including tumors.

For more information, please check:
BioContainers: https://biocontainers.pro/tools/trust4
Home page: https://github.com/liulab-dfci/TRUST4

468.2 Versions

- 1.0.7

468.3 Commands

- run-trust4
- BuildDatabaseFa.pl
- BuildImgtAnnot.pl
- trust-airr.pl
- trust-barcoderep.pl
- trust-simplerep.pl
- trust-smartseq.pl

468.4 Module

You can load the modules by:

```
module load biocontainers
module load trust4
```

468.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run trust4 on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=trust4
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers trust4

run-trust4 -b mapped.bam -f hg38_bcrtcr.fa --ref human_IMGT+C.fa
```

CHAPTER

NINE

TRYCYCLER

469.1 Introduction

Trycycler is a tool for generating consensus long-read assemblies for bacterial genomes. I.e. if you have multiple long-read assemblies for the same isolate, Trycycler can combine them into a single assembly that is better than any of your inputs.

For more information, please check:
Docker hub: https://hub.docker.com/r/staphb/trycycler
Home page: https://github.com/rrwick/Trycycler

469.2 Versions

- 0.5.0
- 0.5.3

469.3 Commands

- trycycler

469.4 Module

You can load the modules by:

```
module load biocontainers
module load trycycler
```

469.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run trycycler on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=trycycler
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers trycycler

trycycler cluster --assemblies \
    test/test_cluster/assembly_*.fasta \
    --read test/test_cluster/reads.fastq \
    --out_dir trycycler_out
```

UCSC EXECUTABLES

470.1 Introduction

`UCSC Executables` is a variety of executables that perform functions ranging from sequence analysis and format conversion, to basic number crunching and statistics, to complex database generation and manipulation.

These executables have been downloaded from http://hgdownload.soe.ucsc.edu/admin/exe/linux.x86_64.v369/ and made available on RCAC clusters.

470.2 Versions

- 369

470.3 Commands

- addCols
- ameme
- autoDtd
- autoSql
- autoXml
- ave
- aveCols
- axtChain
- axtSort
- axtSwap
- axtToMaf
- axtToPsl
- bamToPsl
- barChartMaxLimit
- bedClip
- bedCommonRegions
- bedCoverage

- bedExtendRanges
- bedGeneParts
- bedGraphPack
- bedGraphToBigWig
- bedIntersect
- bedItemOverlapCount
- bedJoinTabOffset
- bedJoinTabOffset.py
- bedMergeAdjacent
- bedPartition
- bedPileUps
- bedRemoveOverlap
- bedRestrictToPositions
- bedSingleCover.pl
- bedSort
- bedToBigBed
- bedToExons
- bedToGenePred
- bedToPsl
- bedWeedOverlapping
- bigBedInfo
- bigBedNamedItems
- bigBedSummary
- bigBedToBed
- bigGenePredToGenePred
- bigHeat
- bigMafToMaf
- bigPslToPsl
- bigWigAverageOverBed
- bigWigCat
- bigWigCluster
- bigWigCorrelate
- bigWigInfo
- bigWigMerge
- bigWigSummary
- bigWigToBedGraph

- bigWigToWig
- binFromRange
- blastToPsl
- blastXmlToPsl
- blat
- calc
- catDir
- catUncomment
- chainAntiRepeat
- chainBridge
- chainCleaner
- chainFilter
- chainMergeSort
- chainNet
- chainPreNet
- chainScore
- chainSort
- chainSplit
- chainStitchId
- chainSwap
- chainToAxt
- chainToPsl
- chainToPslBasic
- checkAgpAndFa
- checkCoverageGaps
- checkHgFindSpec
- checkTableCoords
- chopFaLines
- chromGraphFromBin
- chromGraphToBin
- chromToUcsc
- clusterGenes
- clusterMatrixToBarChartBed
- colTransform
- countChars
- cpg_lh

- crTreeIndexBed
- crTreeSearchBed
- dbSnoop
- dbTrash
- endsInLf
- estOrient
- expMatrixToBarchartBed
- faAlign
- faCmp
- faCount
- faFilter
- faFilterN
- faFrag
- faNoise
- faOneRecord
- faPolyASizes
- faRandomize
- faRc
- faSize
- faSomeRecords
- faSplit
- faToFastq
- faToTab
- faToTwoBit
- faToVcf
- faTrans
- fastqStatsAndSubsample
- fastqToFa
- featureBits
- fetchChromSizes
- findMotif
- fixStepToBedGraph.pl
- gapToLift
- genePredCheck
- genePredFilter
- genePredHisto

- genePredSingleCover
- genePredToBed
- genePredToBigGenePred
- genePredToFakePsl
- genePredToGtf
- genePredToMafFrames
- genePredToProt
- gensub2
- getRna
- getRnaPred
- gff3ToGenePred
- gff3ToPsl
- gmtime
- gtfToGenePred
- headRest
- hgBbiDbLink
- hgFakeAgp
- hgFindSpec
- hgGcPercent
- hgGoldGapGl
- hgLoadBed
- hgLoadChain
- hgLoadGap
- hgLoadMaf
- hgLoadMafSummary
- hgLoadNet
- hgLoadOut
- hgLoadOutJoined
- hgLoadSqlTab
- hgLoadWiggle
- hgSpeciesRna
- hgTrackDb
- hgWiggle
- hgsql
- hgsqldump
- hgvsToVcf

- hicInfo
- htmlCheck
- hubCheck
- hubClone
- hubPublicCheck
- ixIxx
- lastz-1.04.00
- lastz_D-1.04.00
- lavToAxt
- lavToPsl
- ldHgGene
- liftOver
- liftOverMerge
- liftUp
- linesToRa
- localtime
- mafAddIRows
- mafAddQRows
- mafCoverage
- mafFetch
- mafFilter
- mafFrag
- mafFrags
- mafGene
- mafMeFirst
- mafNoAlign
- mafOrder
- mafRanges
- mafSpeciesList
- mafSpeciesSubset
- mafSplit
- mafSplitPos
- mafToAxt
- mafToBigMaf
- mafToPsl
- mafToSnpBed

- mafsInRegion
- makeTableList
- maskOutFa
- matrixClusterColumns
- matrixMarketToTsv
- matrixNormalize
- mktime
- mrnaToGene
- netChainSubset
- netClass
- netFilter
- netSplit
- netSyntenic
- netToAxt
- netToBed
- newProg
- newPythonProg
- nibFrag
- nibSize
- oligoMatch
- overlapSelect
- para
- paraFetch
- paraHub
- paraHubStop
- paraNode
- paraNodeStart
- paraNodeStatus
- paraNodeStop
- paraSync
- paraTestJob
- parasol
- positionalTblCheck
- pslCDnaFilter
- pslCat
- pslCheck

- pslDropOverlap
- pslFilter
- pslHisto
- pslLiftSubrangeBlat
- pslMap
- pslMapPostChain
- pslMrnaCover
- pslPairs
- pslPartition
- pslPosTarget
- pslPretty
- pslRc
- pslRecalcMatch
- pslRemoveFrameShifts
- pslReps
- pslScore
- pslSelect
- pslSomeRecords
- pslSort
- pslSortAcc
- pslStats
- pslSwap
- pslToBed
- pslToBigPsl
- pslToChain
- pslToPslx
- pslxToFa
- qaToQac
- qacAgpLift
- qacToQa
- qacToWig
- raSqlQuery
- raToLines
- raToTab
- randomLines
- rmFaDups

- rowsToCols
- sizeof
- spacedToTab
- splitFile
- splitFileByColumn
- sqlToXml
- strexCalc
- stringify
- subChar
- subColumn
- tabQuery
- tailLines
- tdbQuery
- tdbRename
- tdbSort
- textHistogram
- tickToDate
- toLower
- toUpper
- trackDbIndexBb
- transMapPslToGenePred
- trfBig
- twoBitDup
- twoBitInfo
- twoBitMask
- twoBitToFa
- ucscApiClient
- udr
- vai.pl
- validateFiles
- validateManifest
- varStepToBedGraph.pl
- webSync
- wigCorrelate
- wigEncode
- wigToBigWig

- wordLine
- xmlCat
- xmlToSql

470.4 Module

You can load the modules by:

```
module load biocontainers
module load ucsc_genome_toolkit/369
```

470.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run UCSC executables on our our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 12
#SBATCH --job-name=UCSC
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers ucsc_genome_toolkit/369

blat genome.fasta input.fasta blat.out
fastqToFa input.fastq  output.fasta
```

CHAPTER

ONE

UNICYCLER

471.1 Introduction

`Unicycler` is an assembly pipeline for bacterial genomes.

For more information, please check its website: https://biocontainers.pro/tools/unicycler and its home page on Github.

471.2 Versions

- 0.5.0

471.3 Commands

- unicycler

471.4 Module

You can load the modules by:

```
module load biocontainers
module load unicycler
```

471.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Unicycler on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 12
#SBATCH --job-name=unicycler
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers unicycler

unicycler -t 12 -1 SRR11234553_1.fastq  -2 SRR11234553_2.fastq -o shortout

unicycler -t 12  -l SRR3982487.fastq  -o longout
```

CHAPTER

TWO

VADR

472.1 Introduction

VADR is a suite of tools for classifying and analyzing sequences homologous to a set of reference models of viral genomes or gene families. It has been mainly tested for analysis of Norovirus, Dengue, and SARS-CoV-2 virus sequences in preparation for submission to the GenBank database.

For more information, please check:
Docker hub: https://hub.docker.com/r/staphb/vadr
Home page: https://github.com/ncbi/vadr

472.2 Versions

- 1.4.1
- 1.4.2
- 1.5

472.3 Commands

- parse_blast.pl
- v-annotate.pl
- v-build.pl
- v-test.pl

472.4 Module

You can load the modules by:

```
module load biocontainers
module load vadr
```

472.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run vadr on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=vadr
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers vadr

v-annotate.pl noro.9.fa va-noro.9
```

CHAPTER

THREE

VARDICT-JAVA

473.1 Introduction

VarDictJava is a variant discovery program written in Java and Perl. It is a Java port of VarDict variant caller.

For more information, please check:
Docker hub: https://hub.docker.com/r/hydragenetics/vardict
Home page: https://github.com/AstraZeneca-NGS/VarDictJava

473.2 Versions

- 1.8.3

473.3 Commands

- vardict-java
- var2vcf_paired.pl
- var2vcf_valid.pl
- testsomatic.R
- teststrandbias.R

473.4 Module

You can load the modules by:

```
module load biocontainers
module load vardict-java
```

473.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run vardict-java on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=vardict-java
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers vardict-java

AF_THR="0.01" # minimum allele frequency
vardict-java -G genome.fasta \
    -f $AF_THR -N genome \
    -b input.bam \
    -c 1 -S 2 -E 3 -g 4 output.bed \
     |  teststrandbias.R \
     |  var2vcf_valid.pl \
     -N genome -E -f $AF_THR \
     > vars.vcf
```

CHAPTER

FOUR

VARLOCIRAPTOR

474.1 Introduction

`Varlociraptor` implements a novel, unified fully uncertainty-aware approach to genomic variant calling in arbitrary scenarios.

For more information, please check its website: https://biocontainers.pro/tools/varlociraptor and its home page on Github.

474.2 Versions

- 4.11.4

474.3 Commands

- varlociraptor

474.4 Module

You can load the modules by:

```
module load biocontainers
module load varlociraptor
```

474.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Varlociraptor on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=varlociraptor
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers varlociraptor

varlociraptor call variants tumor-normal --purity 0.75 --tumor
```

CHAPTER

FIVE

VARSCAN

475.1 Introduction

`Varscan` is a tool used for variant detection in massively parallel sequencing data.

For more information, please check its home page: http://varscan.sourceforge.net/index.html.

475.2 Versions

- 2.4.2
- 2.4.4

475.3 Commands

- VarScan.v2.4.4.jar

475.4 Module

You can load the modules by:

```
module load biocontainers
module load varscan
```

475.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Varscan on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=varscan
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers varscan
```

CHAPTER

SIX

VARTRIX

476.1 Introduction

`Vartrix` is a software tool for extracting single cell variant information from 10x Genomics single cell data.

For more information, please check its website: https://biocontainers.pro/tools/vartrix and its home page on Github.

476.2 Versions

- 1.1.22

476.3 Commands

- vartrix

476.4 Module

You can load the modules by:

```
module load biocontainers
module load vartrix
```

476.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Vartrix on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=vartrix
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers vartrix

vartrix -v test/test.vcf -b test/test.bam \
    -f test/test.fa -c test/barcodes.tsv \
    -o output.matrix
```

CHAPTER

SEVEN

VATOOLS

477.1 Introduction

VAtools is a python package that includes several tools to annotate VCF files with data from other tools.

For more information, please check:
Docker hub: https://hub.docker.com/r/griffithlab/vatools
Home page: https://vatools.readthedocs.io/en/latest/

477.2 Versions

- 5.0.1

477.3 Commands

- ref-transcript-mismatch-reporter
- transform-split-values
- vcf-expression-annotator
- vcf-genotype-annotator
- vcf-info-annotator
- vcf-readcount-annotator
- vep-annotation-reporter

477.4 Module

You can load the modules by:

```
module load biocontainers
module load vatools
```

477.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run vatools on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=vatools
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers vatools

vcf-readcount-annotator <input_vcf> <snv_bam_readcount_file> <DNA| RNA> \
            -s <sample_name> -t snv -o <snv_annotated_vcf>
```

CHAPTER

EIGHT

VCF2MAF

478.1 Introduction

To convert a VCF into a MAF, each variant must be mapped to only one of all possible gene transcripts/isoforms that it might affect. This selection of a single effect per variant, is often subjective. So this project is an attempt to make the selection criteria smarter, reproducible, and more configurable. And the default criteria must lean towards best practices.

For more information, please check:
Home page: https://github.com/mskcc/vcf2maf

478.2 Versions

- 1.6.21

478.3 Commands

- maf2maf.pl
- maf2vcf.pl
- vcf2maf.pl
- vcf2vcf.pl

478.4 Module

You can load the modules by:

```
module load biocontainers
module load vcf2maf
```

478.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

Note: If users need to use `vep`, please add `--vep-path /opt/conda/bin`.

To run vcf2maf on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=vcf2maf
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers vcf2maf

vcf2maf.pl --vep-path /opt/conda/bin \
    --ref-fasta Homo_sapiens.GRCh37.dna.toplevel.fa.gz \
    --input-vcf tests/test.vcf --output-maf test.vep.maf
```

CHAPTER

NINE

VCF2PHYLIP

479.1 Introduction

vcf2phylip is a tool to convert SNPs in VCF format to PHYLIP, NEXUS, binary NEXUS, or FASTA alignments for phylogenetic analysis.

For more information, please check:
Home page: https://github.com/edgardomortiz/vcf2phylip

479.2 Versions

- 2.8

479.3 Commands

- vcf2phylip.py

479.4 Module

You can load the modules by:

```
module load biocontainers
module load vcf2phylip
```

479.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run vcf2phylip on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=vcf2phylip
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers vcf2phylip

vcf2phylip --input myfile.vcf
```

VCF-KIT

480.1 Introduction

VCF-kit is a command-line based collection of utilities for performing analysis on Variant Call Format (VCF) files.

For more information, please check:
BioContainers: https://biocontainers.pro/tools/vcf-kit
Home page: https://github.com/AndersenLab/VCF-kit

480.2 Versions

- 0.2.6
- 0.2.9

480.3 Commands

- vk

480.4 Module

You can load the modules by:

```
module load biocontainers
module load vcf-kit
```

480.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run vcf-kit on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=vcf-kit
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers vcf-kit
```

CHAPTER

ONE

VCFTOOLS

481.1 Introduction

`VCFtools` is a program package designed for working with VCF files, such as those generated by the 1000 Genomes Project. The aim of VCFtools is to provide easily accessible methods for working with complex genetic variation data in the form of VCF files.

For more information, please check its website: https://biocontainers.pro/tools/vcftools and its home page on Github.

481.2 Versions

- 0.1.16

481.3 Commands

- vcftools

481.4 Module

You can load the modules by:

```
module load biocontainers
module load vartrix
```

481.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run VCFtools on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=vcftools
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers vcftools

vcftools --vcf input_data.vcf --chr 1 \
    --from-bp 1000000 --to-bp 2000000
```

CHAPTER

TWO

VELOCYTO.PY

482.1 Introduction

`Velocyto.py` a library for the analysis of RNA velocity.

Detailed information about velocyto.py can be found here: https://github.com/velocyto-team/velocyto.py.

482.2 Versions

- 0.17.17-py39

482.3 Commands

- python
- python3
- velocyto

482.4 Module

You can load the modules by:

```
module load biocontainers
module load velocyto.py/0.17.17-py39
```

482.5 Interactive job

To run `Velocyto.py` interactively on our clusters:

```
(base) UserID@bell-fe00:~ $ sinteractive -N1 -n12 -t4:00:00 -A myallocation
salloc: Granted job allocation 12345869
salloc: Waiting for resource configuration
salloc: Nodes bell-a008 are ready for job
(base) UserID@bell-a008:~ $ module load biocontainers cellrank/1.5.1
```

(continues on next page)

(continued from previous page)

```
(base) UserID@bell-a008:~ $ python
Python 3.9.10 |  packaged by conda-forge |  (main, Feb  1 2022, 21:24:11)
[GCC 9.4.0] :: Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import velocyto as vcy
>>> vlm = vcy.VelocytoLoom("YourData.loom")
>>> vlm.normalize("S", size=True, log=True)
>>> vlm.S_norm  # contains log normalized
```

482.6 Batch job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To submit a sbatch job on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 10:00:00
#SBATCH -N 1
#SBATCH -n 24
#SBATCH --job-name=Velocyto
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers velocyto.py/0.17.17-py39

velocyto run10x cellranger_count_1kpbmcs_out refdata-gex-GRCh38-2020-A/genes/genes.gtf
```

CHAPTER

THREE

VELVET

483.1 Introduction

`Velvet` is a sequence assembler for very short reads.

For more information, please check its website: https://biocontainers.pro/tools/velvet and its home page: https://www.ebi.ac.uk/~zerbino/velvet/.

483.2 Versions

- 1.2.10

483.3 Commands

- velveth
- velvetg

483.4 Module

You can load the modules by:

```
module load biocontainers
module load trimmomatic
```

483.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Velvet on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=velvet
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers velvet

velveth output_directory 21 -fasta -short solexa1.fa solexa2.fa solexa3.fa -long␣
↪capillary.fa
velvetg output_directory -cov_cutoff 4
```

CHAPTER

FOUR

VG

484.1 Introduction

Variation graphs (vg) provides tools for working with genome variation graphs.

For more information, please check:

Quay.io: https://quay.io/repository/vgteam/vg?tabinfo | Home page: https://github.com/vgteam/vg

484.2 Versions

- 1.40.0

484.3 Commands

- vg

484.4 Module

You can load the modules by:

```
module load biocontainers
module load vg
```

484.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run vg on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=vg
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers vg

vg construct -r test/small/x.fa -v test/small/x.vcf.gz >x.vg

# GFA output
vg view x.vg >x.gfa

# dot output suitable for graphviz
vg view -d x.vg >x.dot

# And if you have a GAM file
cp small/x-s1337-n1.gam x.gam

# json version of binary alignments
vg view -a x.gam >x.json

vg align -s CTACTGACAGCAGAAGTTTGCTGTGAAGATTAAATTAGGTGATGCTTG x.vg
```

CHAPTER

FIVE

VIENNARNA

485.1 Introduction

`Viennarna` is a set of standalone programs and libraries used for prediction and analysis of RNA secondary structures.

For more information, please check its website: https://biocontainers.pro/tools/viennarna and its home page: https://www.tbi.univie.ac.at/RNA/.

485.2 Versions

- 2.5.0-py37

485.3 Commands

- RNA2Dfold
- RNALalifold
- RNALfold
- RNAPKplex
- RNAaliduplex
- RNAalifold
- RNAcofold
- RNAdistance
- RNAdos
- RNAduplex
- RNAeval
- RNAfold
- RNAforester
- RNAheat
- RNAinverse

- RNAlocmin
- RNAmultifold
- RNApaln
- RNAparconv
- RNApdist
- RNAplex
- RNAplfold
- RNAplot
- RNApvmin
- RNAsnoop
- RNAsubopt
- RNAup
- Kinfold
- b2ct
- popt

485.4 Module

You can load the modules by:

```
module load biocontainers
module load viennarna
```

485.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Viennarna on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=viennarna
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
```

(continues on next page)

(continued from previous page)

```
ml biocontainers viennarna

RNAfold < test.seq
RNAfold -p --MEA < test.seq
```

CHAPTER

SIX

VSEARCH

486.1 Introduction

`Vsearch` is a versatile open source tool for metagenomics.

For more information, please check its website: https://biocontainers.pro/tools/vsearch and its home page on Github.

486.2 Versions

- 2.19.0
- 2.21.1

486.3 Commands

- vsearch

486.4 Module

You can load the modules by:

```
module load biocontainers
module load vsearch
```

486.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Vsearch on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=vsearch
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers vsearch

vsearch -sintax SRR8723605_merged.fasta -db rdp_16s_v16_sp.fa \
    -tabbedout SRR8723605_out.txt -strand both -sintax_cutoff 0.5
```

CHAPTER

SEVEN

WEBLOGO

487.1 Introduction

`Weblogo` is a web based application designed to make the generation of sequence logos as easy and painless as possible.

For more information, please check its website: https://biocontainers.pro/tools/weblogo and its home page: http://weblogo.threeplusone.com.

487.2 Versions

- 3.7.8

487.3 Commands

- weblogo

487.4 Module

You can load the modules by:

```
module load biocontainers
module load weblogo
```

487.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Weblogo on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=weblogo
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers weblogo

weblogo --resolution 600 --format PNG \
     <seq.fasta >logo.png
```

CHAPTER

EIGHT

WHATSHAP

488.1 Introduction

Whatshap is a software for phasing genomic variants using DNA sequencing reads, also called read-based phasing or haplotype assembly. It is especially suitable for long reads, but works also well with short reads.

For more information, please check:
BioContainers: https://biocontainers.pro/tools/whatshap
Home page: https://github.com/whatshap/whatshap

488.2 Versions

- 1.4-py39

488.3 Commands

- whatshap

488.4 Module

You can load the modules by:

```
module load biocontainers
module load whatshap
```

488.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run whatshap on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=whatshap
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers whatshap

whatshap phase --indels \
    --reference=reference.fasta \
    variants.vcf pacbio.bam
```

CHAPTER

NINE

WIGGLETOOLS

489.1 Introduction

The WiggleTools package allows genomewide data files to be manipulated as numerical functions, equipped with all the standard functional analysis operators (sum, product, product by a scalar, comparators), and derived statistics (mean, median, variance, stddev, t-test, Wilcoxon's rank sum test, etc).

For more information, please check:
Docker hub: https://hub.docker.com/r/ensemblorg/wiggletools
Home page: https://github.com/Ensembl/WiggleTools

489.2 Versions

- 1.2.11

489.3 Commands

- wiggletools

489.4 Module

You can load the modules by:

```
module load biocontainers
module load wiggletools
```

489.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run wiggletools on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=wiggletools
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers wiggletools

wiggletools test/fixedStep.wig
wiggletools test/fixedStep.bw
wiggletools test/bedfile.bg
wiggletools test/overlapping.bed
wiggletools test/bam.bam
wiggletools test/cram.cram
wiggletools test/vcf.vcf
wiggletools test/bcf.bcf
```

WINNOWMAP

490.1 Introduction

Winnowmap is a long-read mapping algorithm optimized for mapping ONT and PacBio reads to repetitive reference sequences.

For more information, please check:
BioContainers: https://biocontainers.pro/tools/winnowmap
Home page: https://github.com/marbl/Winnowmap

490.2 Versions

- 2.03

490.3 Commands

- winnowmap

490.4 Module

You can load the modules by:

```
module load biocontainers
module load winnowmap
```

490.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run winnowmap on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 1
#SBATCH --job-name=winnowmap
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers winnowmap

winnowmap -W repetitive_k15.txt \
    -ax map-pb Cm.contigs.fasta \
    SRR3982487.fastq > output.sam
```

CHAPTER

ONE

WTDBG2

491.1 Introduction

`Wtdbg2` is a de novo sequence assembler for long noisy reads produced by PacBio or Oxford Nanopore Technologies (ONT).

For more information, please check its website: https://biocontainers.pro/tools/wtdbg and its home page on Github.

491.2 Versions

- 2.5

491.3 Commands

- wtdbg-cns
- wtdbg2
- wtpoa-cns

491.4 Module

You can load the modules by:

```
module load biocontainers
module load wtdbg
```

491.5 Example job

Warning: Using `#!/bin/sh -l` as shebang in the slurm job script will cause the failure of some biocontainer modules. Please use `#!/bin/bash` instead.

To run Wtdbg2 on our clusters:

```
#!/bin/bash
#SBATCH -A myallocation     # Allocation name
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH -n 24
#SBATCH --job-name=wtdbg
#SBATCH --mail-type=FAIL,BEGIN,END
#SBATCH --error=%x-%J-%u.err
#SBATCH --output=%x-%J-%u.out

module --force purge
ml biocontainers wtdbg

wtpoa-cns -t 24 -i dbg.ctg.lay.gz -fo dbg.ctg.fa
```