

```

1  import os.path
2
3  from glob import glob
4  import atexit
5  from pprint import pprint
6
7  from tools import *
8  from techs import *
9
10
11  #####
12  ## SETUP
13  #####
14  dataDir = setup.dataDir
15  outDir = setup.outDir
16  print "OUT DIR",outDir
17  print "DATA DIR",dataDir
18
19
20  #####
21  ## CHECKING
22  #####
23  env = Environment()
24
25  if not os.path.exists(outDir):
26      print "CRETING OUTPUT DIR"
27      Mkdir(outDir)
28  print "OUT DIR",outDir
29  Decider('timestamp-newer')
30
31
32  #####
33  ## FUNCTIONS
34  #####
35  ##fastq2jf_obj = Builder( action = 'jellyfish $setup.JELLYPARAMS --output $TARGET $SOURCE',
36  ##fastq2jf_obj = Builder( action = runners.fastq2jf,
37  ##                          suffix = '.jf',
38  ##                          src_suffix = '.fastq')
39  ##
40  ##
41  ##env['BUILDERS']['fastq2jf_B'] = fastq2jf_obj
42
43  ##F5_Illumina_GOG20_matepair_2000_110401_SN365_A_s_4_2_seq_GOG-20.RD30.NotEmpty.NotLink.fastq.histo
44  ##F5_Illumina_GOG20_matepair_2000_110401_SN365_A_s_4_2_seq_GOG-20.RD30.NotEmpty.NotLink.fastq.histo.desc
45  ##F5_Illumina_GOG20_matepair_2000_110401_SN365_A_s_4_2_seq_GOG-20.RD30.NotEmpty.NotLink.fastq.histo.png
46  ##F5_Illumina_GOG20_matepair_2000_110401_SN365_A_s_4_2_seq_GOG-20.RD30.NotEmpty.NotLink.fastq.jf
47  ##F5_Illumina_GOG20_matepair_2000_110401_SN365_A_s_4_2_seq_GOG-20.RD30.NotEmpty.NotLink.fastq.nfo
48  ##F5_Illumina_GOG20_matepair_2000_110401_SN365_A_s_4_2_seq_GOG-20.RD30.NotEmpty.NotLink.fastq.stats
49  ##F5_Illumina_GOG20_matepair_2000.histo
50  ##F5_Illumina_GOG20_matepair_2000.histo.desc
51  ##F5_Illumina_GOG20_matepair_2000.histo.png
52  ##F5_Illumina_GOG20_matepair_2000.jf
53  ##F5_Illumina_GOG20_matepair_2000.stats
54  ##F5_Illumina.histo
55  ##F5_Illumina.histo.desc
56  ##F5_Illumina.histo.png
57  ##F5_Illumina.jf
58  ##F5_Illumina.stats
59
60
61
62  def checkFq(env, infiles, outfiles=[], baseDir=outDir, dataset=None, library=None, pair=None, object=None):
63      for file in infiles:
64          bn = os.path.basename(str(file))
65
66          if library is not None:
67              bn = library + '_' + bn
68
69          if dataset is not None:
70              bn = dataset + '_' + bn
71
72          bn = os.path.join(baseDir, bn)
73
74          nfo = File(bn + ".nfo")
75          nfoOut = env.Command( [nfo], [file], runners.makeNfo )
76          Precious( nfoOut )
77          NoClean( nfoOut )
78
79          outJf = bn + ".jf"
80          outStat = bn + ".stats"
81          outHisto = bn + ".histo"
82          outPng = bn + ".histo.png"
83          outDesc = bn + ".histo.desc"
84          okFile = bn + ".ok"
85
86          ##fastqc = bn +
87          ##LISTCMDQCF[COQCF]
88  ="/histoProjectDoQcDoFastqc \"${PROJFOLDER}\" \"${TECHFOLDER}\" \"${LIBFOLDER}\" \"${LOGFILE}\" \"${fastq
89  \"\"
90          ##CMD=\"$QCP -o '$OUTQCDS' $fastq\";
91          ##LISTCMDQCS[COQCS]
92  ="/histoProjectDoQcDoSolexaqa \"${PROJFOLDER}\" \"${TECHFOLDER}\" \"${LIBFOLDER}\" \"${LOGFILE}\" \"${fas
93  tq\" \"

```

```

91     ##CMD="$SOLEXAQA -s $SOLEXAQASAMPLESIZE $JELLYIN -o $OUTQCDS";
92
93     if object is not None:
94         if 'addOutput' in dir(object) and callable(getattr(object, 'addOutput')):
95             object.addOutput('jellyDb'      , outJf)
96             object.addOutput('jellyStats'    , outStat)
97             object.addOutput('jellyHisto'    , outHisto)
98             object.addOutput('jellyPng'     , outPng)
99             object.addOutput('jellyOk'      , okFile)
100            object.addOutput('desc'         , outDesc)
101            object.addOutput('nfo'          , str(nfo))
102
103
104            JFout = env.Command([outJf, outStat, outHisto ], [file], runners.fastq2jf)
105            ImgOut = env.Command([outPng, outDesc], [outStat, outHisto], runners.histo2png)
106            Depends( ImgOut, nfoOut )
107            Depends( JFout, nfoOut )
108            Depends( ImgOut, JFout )
109
110            all = []
111            all.append( file )
112            all.extend( nfoOut )
113            all.extend( JFout )
114            all.extend( ImgOut )
115
116            ok = env.Command([ okFile ], all, runners.checkOk)
117            outfiles = [ outJf, okFile ]
118
119            print "COUNT KMERS ON FASTQ FILES\n\t", "\n\t".join(map(lambda x: str(x), infiles)), '\nto\n\t', "\n\t".join
120            map(lambda x:
121                str(x), outfiles)), '\non\n\t', baseDir
122            print "OUT FQ\n\t", "\n\t".join(map(lambda x: str(x), outfiles)), "\n\n"
123            return outfiles
124            AddMethod(Environment, checkFq)
125
126            def joinjf(env, infiles, outfiles=
127            [], baseDir=outDir, dataset=None, library=None, pair=None, object=None, name=""):
128                file = ""
129                print "MERGING JF FILES [" , name, "]\n\t", "\n\t".join(map(lambda x: str(x), infiles)), '\nto\n\t', "\n\t".join
130                map(lambda x
131                    : str(x), outfiles)), '\non\n\t', baseDir
132
133                if len(outfiles) == 0:
134                    ##infilesShort = [ os.path.basename(str(x)) for x in infiles ]
135                    ##print "IN FILES SHORT", str(infilesShort)
136                    ##file = os.path.commonprefix(infilesShort)
137                    ##print "COMMON", file
138                    ##file = file.rstrip("_.-")
139                    ##if file == "": file = "mergedRuns"
140                    exit(1)
141                else:
142                    file = outfiles[0]
143                    if file[-3:] == ".jf":
144                        file = file[:-3]
145                    else:
146                        pass
147
148                    ##infiles = [file]
149                    ##print "INFILES", map(lambda x: str(x), infiles), "FILE", file
150
151                    if os.path.dirname(os.path.abspath(file)) != baseDir:
152                        bn = os.path.basename(file)
153                        if library is not None:
154                            bn = library + '_' + bn
155                        if dataset is not None:
156                            bn = dataset + '_' + bn
157                        file = os.path.join(baseDir, bn)
158
159                    outJf = file + ".jf"
160                    outStat = file + ".stats"
161                    outHisto = file + ".histo"
162                    outPng = file + ".histo.png"
163                    outDesc = file + ".histo.desc"
164                    okFile = file + ".ok"
165
166                    if object is not None:
167                        if 'addOutput' in dir(object) and callable(getattr(object, 'addOutput')):
168                            object.addOutput('jellyDb'      , outJf)
169                            object.addOutput('jellyStats'    , outStat)
170                            object.addOutput('jellyHisto'    , outHisto)
171                            object.addOutput('jellyHistoDesc', outDesc)
172                            object.addOutput('jellyPng'     , outPng)
173                            object.addOutput('jellyOk'      , okFile)
174
175                    JFout = env.Command([outJf, outStat, outHisto], [x for x in infiles if str(x).endswith
176                    ('.jf')], runners.j
177                    oinjf)
178                    ImgOut = env.Command([outPng, outDesc], [outStat, outHisto], runners.histo2png)
179                    Depends(JFout, infiles)
180                    ##Depends( ImgOut, JFout )
181
182                    all = []

```

```

179     all.append( infiles )
180     all.extend( JFout )
181     all.extend( ImgOut )
182     ok = env.Command([ okFile ], all, runners.checkOk)
183     ##print "OK",okFile
184     outfiles = [ outJf , okFile ]
185
186     print "OUT JF\n\t", "\n\t".join(map(lambda x: str(x), outfiles)), "\n\n"
187     return outfiles
188 ##env.AddMethod(fastq2jf_def, "fastq2jf")
189 AddMethod(Environment, joinjf)
190
191
192
193 def fastqc(env, infiles, outfiles=[], baseDir=outDir, dataset=None, library=None, pair=None, object=None):
194     print "RUNNING FASTQC\n\t", "\n\t".join(map(lambda x: str(x), infiles)), '\nto\n\t', "\n\t".join(map
(lambda x: str(x), out
files)), '\non\n\t', baseDir
195
196     if len(outfiles) == 0:
197         exit(1)
198
199     file = outfiles[0]
200     if os.path.dirname(os.path.abspath(file)) != baseDir:
201         bn = os.path.basename(file)
202         if library is not None:
203             bn = library + '_' + bn
204         if dataset is not None:
205             bn = dataset + '_' + bn
206         file = os.path.join(baseDir, bn)
207
208     outZip = file + ".zip"
209     outPath = file + "_fastqc"
210     outData = os.path.join(outPath, 'fastqc_data.txt')
211     outReport = os.path.join(outPath, 'fastqc_report.html')
212     outSummary = os.path.join(outPath, 'summary.txt')
213
214     okFile = file + ".fqc.ok"
215
216     if object is not None:
217         if 'addOutput' in dir(object) and callable(getattr(object, 'addOutput')):
218             ##TODO: ADD INDIVIDUAL FILES INSIDE DIR ?
219             object.addOutput('fastQCZip', outZip)
220             object.addOutput('fastQCDir', outPath)
221             object.addOutput('fastQCData', outData)
222             object.addOutput('fastQCReport', outReport)
223             object.addOutput('fastQCSummary', outSummary)
224             object.addOutput('fastQCOK', okFile)
225
226     #print "\tFASTQC\n\t\t", outZip, "\n\t\t", outDir
227     FQCout = env.Command([outZip, outData, outReport, outSummary], infiles, runners.fastqc)
228     #print "\n\tOUT\n\t\t", FQCout
229     Clean(FQCout, outPath)
230
231     all = []
232     all.append( infiles )
233     all.extend( FQCout )
234     ok = env.Command([ okFile ], all, runners.checkOk)
235     ##print "OK",okFile
236     outfiles = ok
237
238     print "OUT FQC\n\t", "\n\t".join(map(lambda x: str(x), outfiles)), "\n\n"
239     return outfiles
240 ##env.AddMethod(fastq2jf_def, "fastq2jf")
241 AddMethod(Environment, fastqc)
242
243
244
245 def solexaqa(env, infiles, outfiles=[], baseDir=outDir, dataset=None, library=None, pair=None, object=None):
246     print "RUNNING SOLEXAQA\n\t", "\n\t".join(map(lambda x: str(x), infiles)), '\nto\n\t', "\n\t".join(map
(lambda x: str(x), o
utfiles)), '\non\n\t', baseDir
247
248     if len(outfiles) == 0:
249         exit(1)
250
251     file = outfiles[0]
252     bn = os.path.basename(file)
253     if os.path.dirname(os.path.abspath(file)) != baseDir:
254         bn = os.path.basename(file)
255         if library is not None:
256             bn = library + '_' + bn
257         if dataset is not None:
258             bn = dataset + '_' + bn
259         file = os.path.join(baseDir, bn)
260
261     #QUALP="$OUTQCDS/$fastqName$fastqExt.quality.pdf"
262     #QUALI="$OUTQCDS/$fastqName$fastqExt.quality.pdf.png"
263     #HISP="$OUTQCDS/$fastqName$fastqExt.segments.hist.pdf"
264     #HISI="$OUTQCDS/$fastqName$fastqExt.segments.hist.pdf.png"
265
266     outPath = file + '_solexaqa'
267     outMtx = os.path.join(outPath, bn) + ".matrix"
268     outQualPdf = os.path.join(outPath, bn) + ".quality.pdf"
269     outHistPdf = os.path.join(outPath, bn) + ".segments.hist.pdf"

```

```

270     outMtxPng = outMtx + ".png"
271     outQualPng = outQualPdf + '.png'
272     outHistPng = outHistPdf + '.png'
273     okFile = file + ".sqa.ok"
274
275     if object is not None:
276         if 'addOutput' in dir(object) and callable(getattr(object, 'addOutput')):
277             #TODO: ADD INDIVIDUAL FILES INSIDE DIR ?
278             object.addOutput('solexaqaMatrix', outMtx)
279             object.addOutput('solexaqaMatrixPng', outMtxPng)
280             object.addOutput('solexaqaQualPdf', outQualPdf)
281             object.addOutput('solexaqaQualPng', outQualPng)
282             object.addOutput('solexaqaHistPdf', outHistPdf)
283             object.addOutput('solexaqaHistPng', outHistPng)
284             object.addOutput('solexaqaDir', outPath)
285             object.addOutput('solexaqaOk', okFile)
286
287     #print "\tFASTQC\n\t\t",outZip,"\n\t\t",outDir
288     SQAout = env.Command
289     ([outMtx, outMtxPng, outQualPdf, outQualPng, outHistPdf, outHistPng], infiles, runners.solexaqa)
290     #print "\n\tOUT\n\t\t",FQCout
291     Clean(SQAout, outPath)
292
293     all = []
294     all.append( infiles )
295     all.extend( SQAout )
296     ok = env.Command([ okFile ], all, runners.checkOk)
297     #print "OK",okFile
298     outfiles = ok
299
300     print "OUT FQC\n\t", "\n\t".join(map(lambda x: str(x), outfiles)), "\n\n"
301     return outfiles
302     ##env.AddMethod(fastq2jff_def, "fastq2jff")
303     AddMethod(Environment, solexaqa)
304
305     #####
306     ## GLOBAL TARGETS
307     #####
308     def getTargets():
309         for dataset in setup.allSpecies:
310             datasetName = dataset.getName()
311
312             ##path = os.path.abspath(os.path.join(dataDir,datasetName))
313             ##bn = os.path.basename(path)
314             ##if not os.path.exists(path) or not os.path.isdir(path):
315             ##    print "DATASET",datasetName,"DOES NOT EXISTS ON",dataDir
316             ##    continue
317             sppok = os.path.join(os.path.join(outDir, datasetName), datasetName+'.ok')
318             ##print "tgt",datasetName,">",sppok
319             yield [datasetName, sppok]
320
321     def checkTargets(spp):
322         if spp not in valid_targets:
323             print " Invalid species '",spp,"'"
324             print " Valid species are:"
325             print "\t" + "\n\t".join(valid_targets)
326             exit(1)
327
328     ##yield [None, 'all', os.path.join(outDir,'all')+'.ok']
329
330     allName = setup.allSpecies.getName()
331     allAlias = []
332     for spp in getTargets():
333         print 'ADDING ALIAS',spp[0],'TO FILE',spp[1]
334         env.Alias(spp[0], spp[1])
335         allAlias.append(env.Alias(spp[0]))
336         ##print "FINISHED " + str( finished )
337     ##
338     ##
339     Default(env.Alias(allName))
340     env.Alias( allName, [ os.path.join(outDir,allName+'.ok') ] )
341
342     valid_targets = map(lambda x: x[0], getTargets())
343
344     ##env['DISTDIR'] = setup.outDir
345
346
347
348
349
350     #####
351     ## CALLERS
352     #####
353     def doSpecies(datasetName, dataset):
354         print " RUNNING DATASET '" + datasetName + "'"
355         baseDir = os.path.join(outDir,datasetName)
356         ##str(dataset)
357
358         print " DATASET",dataset.getName()," ",dataset
359
360         datasetOut = []
361         for key in dataset.getIlluminaNames():
362             print " ILLUMINA DATASET NAME",key
363             illDataset = dataset.getIllumina(key)

```

```

364         #print illDataset
365
366     libs = []
367     for lib in illDataset:
368         libName = lib.getName()
369         print 'LIB ' + libName
370
371     pairs = []
372     for pair in lib:
373         pairName = pair.getName()
374         print 'PAIR ' + pairName + ' TYPE ' + pair.getType()
375
376     runFiles = []
377     pairFiles = []
378     for run in pair:
379         #print 'RUN ' + run.getShortName() + ' FN ' + run.getFileName()
380         pairFiles.append(run.getFileName())
381         runFiles.extend(env.checkFq([run.getFileName()
382         ], [], baseDir=baseDir, dataset=datasetName, library=libName,
383         me, pair=pairName, object=run))
384         ##ifinished = env.Command(env.Alias(spp), ends, runners.checkOk)
385
386     if ( setup.runFastqc ): datasetOut.extend(env.fastqc
387     ( pairFiles, [ pairName ], baseDir=baseDir, dataset=
388     datasetName, library=libName, pair=pairName, object=pair))
389     if ( setup.runSolexaqa ): datasetOut.extend(env.solexaqa
390     (pairFiles, [ pairName ], baseDir=baseDir, dataset=
391     datasetName, library=libName, pair=pairName, object=pair))
392     if ( setup.runJellyfish ): pairs.extend( env.joinjf
393     ( runFiles, [ pairName ], baseDir=baseDir, dataset=
394     datasetName, library=libName, pair=pairName, object=pair, name="PAIR_" + datasetName + "_" + libName + "_" + pairName))
395
396     ##print "PAIRS",map(lambda x: str(x), pairs)
397     if ( setup.runJellyfish ): libs.extend(env.joinjf
398     (pairs, [ libName ], baseDir=baseDir, dataset=datasetName, object=lib, name="LIB_" + datasetName + "_" + libName))
399
400     ##print "LIBS",map(lambda x: str(x), libs)
401     if ( setup.runJellyfish ): datasetOut.extend(env.joinjf
402     (libs, [ datasetName ], baseDir=baseDir, object=illDataset, name="DATASET_" + datasetName))
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443

```

```

444     else:
445         for spp in [str(x).strip() for x in BUILD_TARGETS ]:
446             if spp == allName: continue
447
448             checkTargets(spp)
449             print "  Valid species",spp
450             dataset = setup.allSpecies.get(spp)
451             ends = doSpecies(spp, dataset)
452             ##finished = env.Command(env.Alias(spp), ends, runners.checkOk)
453
454             print "ENDS " + str(map(lambda x: str(x), ends))
455             ##env.Depends(env.Alias(spp), ends)
456
457
458
459     setup.exporter()
460     ##exit(0)
461
462
463
464
465
466
467
468
469
470
471
472     # Make the build fail if we pass fail=1 on the command line
473     if ARGUMENTS.get('fail', 0):
474         Command('target', 'source', ['/bin/false'])
475
476     def bf_to_str(bf):
477         """Convert an element of GetBuildFailures() to a string
478         in a useful way."""
479         import SCons.Errors
480         if bf is None: # unknown targets product None in list
481             return '(unknown tgt)'
482         elif isinstance(bf, SCons.Errors.StopError):
483             return str(bf)
484         elif bf.node:
485             return str(bf.node) + ': ' + bf.errstr + "\nEXECUTOR: " + str
486         (bf.executor) + "\nACTION: " + str(bf.action)
487         elif bf.filename:
488             return bf.filename + ': ' + bf.errstr + "\nCOMMAND: " + bf.command + "\nEXECUTOR: " + str
489         (bf.executor) + "\nACTION: " + str(bf.action)
490
491         return 'unknown failure: ' + bf.errstr
492
493     def build_status():
494         """Convert the build status to a 2-tuple, (status, msg)."""
495         from SCons.Script import GetBuildFailures
496         bf = GetBuildFailures()
497         if bf:
498             # bf is normally a list of build failures; if an element is None,
499             # it's because of a target that scons doesn't know anything about.
500             status = 'failed'
501             failures_message = "\n".join(["Failed building %s" % bf_to_str(x)
502                                         for x in bf if x is not None])
503         else:
504             # if bf is None, the build completed successfully.
505             status = 'ok'
506             failures_message = ''
507         return (status, failures_message)
508
509     def display_build_status():
510         """Display the build status. Called by atexit.
511         Here you could do all kinds of complicated things."""
512         status, failures_message = build_status()
513         if status == 'failed':
514             print "FAILED!!!" # could display alert, ring bell, etc.
515         elif status == 'ok':
516             print "Build succeeded."
517             print failures_message
518
519     atexit.register(display_build_status)
520
521
522
523
524
525     Help("""
526     Type: 'run' to build the genome
527     """)

```