**the ultimate TEUlt checklist**

Annotations

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Transposon Ultimate** |  | **Manual** |
|  | Helitron Scanner |  | MUST |
|  | Mitefind |  | SINE Finder |
|  | Mitetracker |  | LTR Harvest |
|  | Repeatmodeler |  | LTR Finder |
|  | Repeatmasker |  | LTR Retriever |
|  | Sinescan |  |  |
|  | Tirvish |  |  |
|  | TransposonPSI |  |  |
|  | NCBICDD1000 |  |  |

Changes to TEUlt source code

* add parseLtrRetriever function to AnnotationParser.py

def parseLtrRetriever(pathResDir, fastaFile, targetGFFFile, targetFastaFile):

print("Parse ltrRetriever outputs...")

transposonAnnotations = {}

*#folder = os.listdir(pathResDir)[0]*

files = os.listdir(pathResDir)

selFile = ""

for f in files:

if(f.endswith(".gff3")):

selFile = os.path.join(pathResDir,f)

f = open(selFile,"r")

line = f.readline()

counter = 0

while line!="":

if line.startswith("#") == False:

transposons = line.split("\t")

chrom = transposons[0]

if(not chrom in transposonAnnotations):

transposonAnnotations[chrom] = list()

print(counter)

annoSoftware = "ltrRetriever"

features = "transposon"

strand = transposons[6]

start = int(transposons[3])

end = int(transposons[4])

start,end = correctPositions(start, end)

score = "."

phase = "."

transpNr = str(counter)

comments = transposons[8].split(";")

*# print(comments)*

component\_id = comments[0].split('=')[1]

if(component\_id[0:13] == 'repeat\_region'):

features = "repeat\_region"

attributes = "Repeat region of transposon "+str(counter)

transposonAnnotations[chrom].append(chrom+"\t"+annoSoftware+"\t"+features+"\t"+str(start)+"\t"+str(end)+"\t"+str(score)+"\t"+strand+"\t"+phase+"\t"+transpNr+"\t"+attributes)

elif(component\_id[0:5] == 'LTRRT'):

features = "transposon"

attributes = str(comments)

counter += 1

transposonAnnotations[chrom].append(chrom+"\t"+annoSoftware+"\t"+features+"\t"+str(start)+"\t"+str(end)+"\t"+str(score)+"\t"+strand+"\t"+phase+"\t"+transpNr+"\t"+attributes)

elif(component\_id[0:4] == 'lLTR'):

features = "ltr"

attributes = "Left LTR of transposon "+str(counter)

transposonAnnotations[chrom].append(chrom+"\t"+annoSoftware+"\t"+features+"\t"+str(start)+"\t"+str(end)+"\t"+str(score)+"\t"+strand+"\t"+phase+"\t"+transpNr+"\t"+attributes)

elif(component\_id[0:4] == 'rLTR'):

features = "ltr"

attributes = "Right LTR of transposon "+str(counter)

transposonAnnotations[chrom].append(chrom+"\t"+annoSoftware+"\t"+features+"\t"+str(start)+"\t"+str(end)+"\t"+str(score)+"\t"+strand+"\t"+phase+"\t"+transpNr+"\t"+attributes)

elif( component\_id[0:4] == 'lTSD'):

features = "tsd"

attributes = "Left TSD of transposon "+str(counter)

transposonAnnotations[chrom].append(chrom+"\t"+annoSoftware+"\t"+features+"\t"+str(start)+"\t"+str(end)+"\t"+str(score)+"\t"+strand+"\t"+phase+"\t"+transpNr+"\t"+attributes)

elif(component\_id[0:4] == 'rTSD'):

features = "tsd"

attributes = "Right TSD of transposon "+str(counter)

transposonAnnotations[chrom].append(chrom+"\t"+annoSoftware+"\t"+features+"\t"+str(start)+"\t"+str(end)+"\t"+str(score)+"\t"+strand+"\t"+phase+"\t"+transpNr+"\t"+attributes)

line = f.readline()

f.close()

*# Print results to Annotation file*

f = open('targetGFFFile',"w+")

f.write("# ltrRetriever Annotation")

*# f.write('\n')*

writeGFFhead(f)

keys = list(transposonAnnotations.keys())

keys.sort()

for key in keys:

for i in range(0,len(transposonAnnotations[key])):

f.write(transposonAnnotations[key][i])

f.write("\n")

f.close()

*# Get Transposon Sequences*

exportFastaFile(targetFastaFile, fastaFile, keys, transposonAnnotations)

* add checkLtrRetriever to line 11 list of imported functions in AnnotationParser.py
* add if(checkLtrRetriever) to line ~1400 of AnnotationParser.py

if(checkLtrRetriever(projectFolderPath)):

pathResDir = os.path.join(projectFolderPath,"ltrRetriever")

fastaFile = os.path.join(projectFolderPath,"sequence.fasta")

targetGFFFile = os.path.join(projectFolderPath,"parsedAnnotations","ltrRetriever.gff3")

targetFastaFile = os.path.join(projectFolderPath,"parsedAnnotations","ltrRetriever.fasta")

parseLtrRetriever(pathResDir, fastaFile, targetGFFFile, targetFastaFile)