# Requires Biopython, pandas

# Requires prodigal sequences and the corresponding metachip sequences

# These must be in the same order–

# ensure by converting GTG start codons from V-M

# in the metachip prodigal files and sorting by sequence

# See script by\_sample\_link\_metachip\_to\_prodigal.sh

from Bio import SeqIO

#import SeqFeature,SeqLocation

from glob import glob

import pandas as pd

import os

fasta\_folder="/monfs00/scratch/lwoo0007/WoodsL/Mackay\_glacier/MAGs"

ext=".fa"

# Use seqkit common and seqkit sort for the metchip faa and prodigal faa

# We lose information from the metachip prodial run, so

# we have to compare back to our prodigal run to get the

# proper information back out (feature location mostly)

foi="HGT-recipient"

#metachip\_folder="/monfs00/scratch/lwoo0007/WoodsL/MS6-5\_hybrid\_seq\_sample/HGT\_discovery/metachip/MS6-5\_noblast\_MetaCHIP\_wd/MS6-5\_noblast\_pcofg\_detected\_HGTs\_donor\_genes\_by\_sample\_metachip\_ids"

metachip\_folder="Mackay\_MAGs\_noblast\_pcofg\_detected\_HGTs\_recipient\_genes\_by\_sample\_metachip\_ids/"

#prodigal\_folder="/monfs00/scratch/lwoo0007/WoodsL/MS6-5\_hybrid\_seq\_sample/HGT\_discovery/metachip/MS6-5\_noblast\_MetaCHIP\_wd/MS6-5\_noblast\_pcofg\_detected\_HGTs\_recipient\_genes\_by\_sample\_prodigal\_ids"

prodigal\_folder="Mackay\_MAGs\_noblast\_pcofg\_detected\_HGTs\_recipient\_genes\_by\_sample\_prodigal\_ids"

metachip\_ext=".faa"

prodigal\_ext=".faa"

hgt\_file="Mackay\_MAGs\_noblast\_MetaCHIP\_wd/Mackay\_MAGs\_noblast\_combined\_pcofg\_HGTs\_ip90\_al200bp\_c75\_ei80\_f10kbp/Mackay\_MAGs\_noblast\_pcofg\_detected\_HGTs.txt"

#hgt\_file="/monfs00/scratch/lwoo0007/WoodsL/MS6-5\_hybrid\_seq\_sample/HGT\_discovery/metachip/MS6-5\_noblast\_MetaCHIP\_wd/MS6-5\_noblast\_combined\_pcofg\_HGTs\_ip90\_al200bp\_c75\_ei80\_f10kbp/MS6-5\_noblast\_pcofg\_detected\_HGTs.txt"

#op\_folder="/monfs00/scratch/lwoo0007/WoodsL/MS6-5\_hybrid\_seq\_sample/HGT\_discovery/metachip/MS6-5\_noblast\_MetaCHIP\_wd/donor\_gffs"

op\_folder="Mackay\_MAGs\_metachip\_recipient\_gffs"

if "recipient" in str(prodigal\_folder):

partner\_prodigal\_folder=prodigal\_folder.replace("recipient","donor")

partner\_metachip\_folder=metachip\_folder.replace("recipient","donor")

donor\_bool=False

elif "donor" in str(prodigal\_folder):

partner\_prodigal\_folder=prodigal\_folder.replace("donor","recipient")

partner\_metachip\_folder=metachip\_folder.replace("donor","recipient")

donor\_bool=True

if os.path.exists(op\_folder)==False:

os.mkdir(op\_folder)

all\_hgt\_df=pd.read\_csv(hgt\_file, sep='\t')

#print(hgt\_df)

for file in glob("%s/\*%s\*" % (fasta\_folder, ext)):

sample\_name=file.split("/")[-1].split(ext)[0]

if donor\_bool==True:

arrow="%s-->" % (sample\_name)

elif donor\_bool==False:

arrow="-->%s" % (sample\_name)

hgt\_df=all\_hgt\_df[all\_hgt\_df['direction'].str.contains(arrow)==True]

print("1. %s" % sample\_name)

print(hgt\_df)

prodigal\_file="%s/%s%s" % (prodigal\_folder, sample\_name, prodigal\_ext)

metachip\_file="%s/%s%s" % (metachip\_folder, sample\_name, metachip\_ext)

print("2. %s" % prodigal\_file)

print("3. %s" % metachip\_file)

prodigal\_data=[rec for rec in SeqIO.parse(prodigal\_file, "fasta")]

metachip\_data=[rec for rec in SeqIO.parse(metachip\_file, "fasta")]

sample\_op\_file="%s/%s.gff" % (op\_folder,sample\_name)

op\_handle=open(sample\_op\_file,'w')

for rec\_p, rec\_m in zip(prodigal\_data,metachip\_data):

print("New metachip entry: %s" % (rec\_m.id))

print("Working on: %s" % sample\_name)

arrow=""

orf=rec\_p.id.split('\_')[-1]

seqname=rec\_p.id.replace("\_%s" % (orf),"")

source="metachip"

feature="%s" % (foi)

desc\_elements=rec\_p.description.replace(" ","").split("#")

print(desc\_elements)

start=desc\_elements[1].strip()

#print("Start: %s" % (start))

end=desc\_elements[2].strip()

#print("End: %s" % (end))

strand=desc\_elements[3].strip()

#print("Strand: %s" % (strand))

frame="."

m\_id=rec\_m.id

partner\_gene\_col="Gene\_2"

partner\_gene\_col\_loc=1

sample\_df=hgt\_df[hgt\_df["Gene\_1"]==m\_id]

if len(sample\_df)==0:

sample\_df=hgt\_df[hgt\_df["Gene\_2"]==m\_id]

partner\_gene\_col="Gene\_1"

partner\_gene\_col\_loc=0

print(sample\_df)

# Handle two recipients/two donors

att1=desc\_elements[4].split(";")[0]

for row in sample\_df.itertuples(index=False):

print(row)

score=row[2]

att2=row[partner\_gene\_col\_loc]

att3=row[6]

att4="."

#score=str(sample\_df['Identity'].iloc[0])

#att1=desc\_elements[4].split(";")[0]

#att2=('').join(sample\_df[partner\_gene\_col].to\_list())

#att3=('').join(sample\_df['direction'].to\_list())

#att4=(".")

partner\_sample='\_'.join(att2.split("\_")[0:-1])

print(partner\_sample)

partner\_prodigal\_file="%s/%s%s" % (partner\_prodigal\_folder, partner\_sample, prodigal\_ext)

partner\_metachip\_file="%s/%s%s" % (partner\_metachip\_folder, partner\_sample, metachip\_ext)

partner\_prodigal\_data=[rec for rec in SeqIO.parse(partner\_prodigal\_file, "fasta")]

partner\_metachip\_data=[rec for rec in SeqIO.parse(partner\_metachip\_file, "fasta")]

for partner\_rec\_p,partner\_rec\_m in zip(partner\_prodigal\_data,partner\_metachip\_data):

#print(partner\_rec\_m.id)

if partner\_rec\_m.id==att2:

print(partner\_rec\_m.id)

att4=partner\_rec\_p.id

print(att4)

atts="%s;paired\_gene\_metachip\_id=%s;paired\_gene\_prodigal\_id=%s;direction=%s\n" % (str(att1),str(att2),str(att4),str(att3))

#print(score)

#print(end)

new\_line="\t".join([str(prop) for prop in [seqname,source,feature,start,end,score,strand,frame,atts]])

op\_handle.write(new\_line)

op\_handle.close()