analysis_workflow

January 4, 2024

1 Concatenate all fastas

[ ]:
```
TOTAL=$(ls ../structures_sequences | wc -l)
COUNT=0

for FILE in ../structures_sequences/*fasta ; do

if [[ $(basename $FILE) != PART* ]] ; then
    cat $FILE >> all_full.fasta
fi

COUNT=$(($COUNT+1))
echo "$COUNT / $TOTAL"

done
```

2 MMseqs clustering

[ ]:
```
mkdir -p seq_cluster
mmseqs easy-cluster \
all_full.fasta \
seq_cluster/seq_clusters \
seq_cluster/tmp \
--max-seqs 50000 \
-c 0.7 \
--cov-mode 0 \
--min-seq-id 0.2 \
--cluster-mode 0 \
--threads 5
```

3 Foldseek clustering

[ ]:
```
# Collect representative structures
mkdir -p seq_cluster/rep_structures
COUNT=0
```

```
cut -f1 seq_cluster/seq_clusters_cluster.tsv  | sort -u | while read LINE ; do
    BASE=$(basename $LINE)

    if [[ ! -f seq_cluster/rep_structures/${BASE}.pdb ]] ; then
        cp /wynton/group/gladstone/users/jnomburg/projects/viral_structure/
  ↪structure_symlinks/${BASE}.pdb seq_cluster/rep_structures
    fi

    COUNT=$(($COUNT+1))
    echo $COUNT

done

# Run foldseek
$CODE/vpSAT/bin/foldseek.sh \
-i seq_cluster/rep_structures \
-o foldseek/foldseek_clusters.m8 \
-C foldseek/ignoreme.tsv \
-t 5 \
-v 0.7 \
-c

# Filter on TMscore
sat.py aln_filter \
-a foldseek/foldseek_clusters.m8 \
-o foldseek/foldseek_clusters_mode0cov0.7_TMscore0.4.filt.m8 \
-f␣
  ↪"query,target,fident,alnlen,qlen,tlen,mismatch,gapopen,qstart,qend,tstart,tend,evalue,bits,
  ↪\
-m 0.4 \
-M 1 \
-x alntmscore

# Generate a cluster file
ls seq_cluster/rep_structures > foldseek/all_inputs.txt

sat.py aln_cluster \
-a foldseek/foldseek_clusters_mode0cov0.7_TMscore0.4.filt.m8 \
-o foldseek/foldseek_clusters.tsv \
-A foldseek/all_inputs.txt
```

4 Merge structure and sequence cluster files

[ ]:
```
mkdir -p merged_clusters

sat.py aln_expand_clusters \
-c foldseek/foldseek_clusters.tsv \
-s seq_cluster/seq_clusters_cluster.tsv \
-o merged_clusters/merged_clusters.tsv \
-F "cluster_rep,cluster_member" \
-f "cluster_rep,cluster_member"

# Generate counts file. This wasn't really used.
sat.py aln_taxa_counts \
-c merged_clusters/merged_clusters.tsv \
-o merged_clusters/merged_clusters.counts.tsv \
-F "cluster_ID,cluster_rep,subcluster_rep,cluster_member,cluster_count"

# Add taxonomy
# This is adapting aln_add_taxonomy, which is deisnged for alignments rather␣
 ↪than
# cluster files.
sat.py aln_add_taxonomy \
-a merged_clusters/merged_clusters.tsv \
-o merged_clusters/merged_clusters.tax.tsv.TEMP \
-f "cluster_ID,cluster_rep,query,target,cluster_count"

# Reformat the taxonomy columns to general the file clusters file
awk 'BEGIN {FS=OFS="\t"}
NR==1 {
    for (i=1; i<=NF; i++) {
        if ($i == "query") {
            $i = "subcluster_rep";
            col[i]=1;
        } else if ($i == "target") {
            $i = "cluster_member";
            col[i]=1;
        } else if ($i ~ /^target_/) {
            $i = substr($i, 8);
            col[i]=1;
        } else if ($i ~ /^query_/) {
            col[i]=0;
        } else {
            col[i]=1;
        }
    }
}
{
    for (i=1; i<=NF; i++) {
        if (col[i]) printf "%s%s", $i, (i<NF ? OFS : "\n")
    }
}' merged_clusters/merged_clusters.tax.tsv.TEMP >  merged_clusters/
 ↪merged_clusters.tax.tsv
```

5 Create connection map

[ ]:
```
# This is just for making the family-family network
sat.py aln_connection_map \
-c merged_clusters/merged_clusters.tax.tsv \
-o merged_clusters/connection_map.tsv
```

6 Run DALI to compare reps from all 5.7K-ish protein clusters that have more than 1 member

[ ]:
```
# First collect the structures
mkdir -p dali_euk_vs_euk/strucs
COUNT=0
awk '$5 > 1'  merged_clusters/merged_clusters.tsv | cut -f2 | sort -u | while␣
 ↪read LINE ; do
    cp seq_cluster/rep_structures/${LINE}.pdb dali_euk_vs_euk/strucs
    COUNT=$(($COUNT+1))
    echo "$COUNT"
done

# Import to DALI
$CODE/vpSAT/bin/dali_format_inputs.sh \
-d dali_euk_vs_euk/strucs \
-o dali_euk_vs_euk/euk_dali_db \
-s dali_euk_vs_euk/euk_dali_key.tsv \
-b ~/phage_dali/phage_structure_key.txt \
-L dali_euk_vs_euk/euk_dali_symlinks

# Prepare an SGE array
$CODE/vpSAT/bin/prepare_job_array_sge.sh \
-d dali_euk_vs_euk/euk_dali_db \
-J dali_euk_vs_euk/dali_lists \
-N 1
```

[ ]:
```
# Running the array in an SGE submission
LIST=$(sed "${SGE_TASK_ID}q;d" dali_euk_vs_euk/dali_lists_lists/sublist_list.
 ↪txt)
```

```
TEMP=${SGE_TASK_ID}__$RANDOM

echo "Copying over queries..."
cat $LIST | while read LINE ; do
    FILE=dali_euk_vs_euk/euk_dali_db/$LINE
    mkdir -p $TEMP
    mkdir $TEMP/query
    cp $FILE $TEMP/query
done

cd $TEMP

# Make a copy of the full db here
echo "Copying over the target directory"
cp -r path/to/db target

# Copy the query(s) to the target db so I can get qlen
# NOTE - this isn't necessary for this particular search, bc it's already␣
 ↪all-by-all
echo "Copying the query to the target dir too"
cp query/* target

echo "running the search"
$CODE/vpSAT/bin/dali.sh \
-q query \
-t target \
-o path_to/euk_dali_result \
-n 5

cd ..

rm -r $TEMP
```

[ ]:
```
# Parsing the DALI results
IN_DIR=path_to/euk_dali_result
OUT_DIR=path_to/euk_dali_parsed

for FILE in $IN_DIR/* ; do

sat.py aln_parse_dali \
-a $FILE \
-o ${OUT_DIR}/$(basename ${FILE%.txt}).m8 \
-s dali_euk_vs_euk/euk_dali_key.tsv

done
```

```
[ ]: # Filter: Remove self alignments, filter for Z >= alnlen/10 -4, alnlen > 120
awk -F '\t' 'NR==1 || ($11 >= ($5/10) - 4)'  dali_euk_vs_euk.m8 | awk '$1 !=␣
  ↪$2' | awk '$5 >= 120' > dali_euk_vs_euk.filt.m8
```

7 Running InterProScan on all sequences

```
[ ]: for FILE in pasth/to/structures_sequences/*fasta ; do

    cat $FILE >> all.fasta

done

FASTA=all.fasta

interproscan.sh \
-i $FASTA \
-f tsv \
-appl TIGRFAM,Pfam,CDD \
-o interproscan_PFAM_TIGRFAM_CDD.tsv
```

8 Running DALI to determine cluster purity

Copying and organizing the structures

```
[ ]: STRUCS=path/to/my/structures

COUNT=0

awk '$5 >= 100' merged_clusters.tax.tsv | awk '$1 != "cluster_ID"' | while read␣
  ↪LINE ; do

CLUSTER_ID=$(echo $LINE | awk '{print $1}')
CLUSTER_REP=$(echo $LINE | awk '{print $2}')
CLUSTER_MEMBER=$(echo $LINE | awk '{print $4}')

mkdir -p clusters/cluster_${CLUSTER_ID}/{structures,rep_structure}

# Copy the rep if necessary
if [[ ! -f clusters/cluster_${CLUSTER_ID}/rep_structure/${CLUSTER_REP}.pdb ]] ;␣
  ↪then
    cp $STRUCS/${CLUSTER_REP}.pdb clusters/cluster_${CLUSTER_ID}/rep_structure/
fi

if [[ $CLUSTER_REP == $CLUSTER_MEMBER ]] ; then
    continue
```

```
fi

# Copy the members
if [[ ! -f clusters/cluster_${CLUSTER_ID}/structures/${CLUSTER_MEMBER}.pdb ]] ;␣
↪then
    cp $STRUCS/${CLUSTER_MEMBER}.pdb clusters/cluster_${CLUSTER_ID}/structures
fi

COUNT=$(($COUNT+1))
echo $COUNT

done
```

SGE Array to run the searches

[ ]:
```
#!/bin/bash
#$ -S /bin/bash
#$ -o ./
#$ -e ./
#$ -cwd
#$ -r y
#$ -j y
#$ -l mem_free=10G
#$ -l scratch=10G
#$ -l h_rt=2:00:00
#$ -t 1-57

CLUSTER_DIR=clusters/cluster_${SGE_TASK_ID}

cd $CLUSTER_DIR

# Make the databases
echo "Making the structures database"
$CODE/vpSAT/bin/dali_format_inputs.sh \
-d structures \
-o structure_db \
-s structures_key_NOREP.txt \
-L structures_symlink

echo "Making the rep database"
$CODE/vpSAT/bin/dali_format_inputs.sh \
-d rep_structure \
-o rep_structure_db \
-s structures_key_REP_ONLY.txt \
-L rep_structure_db_symlink \
-b structures_key_NOREP.txt
```

```
# The final key
echo "Merging the key"
cat structures_key*txt > structures_key.txt

# Run DALI
echo "Running DALI"
$CODE/vpSAT/bin/dali.sh \
-q rep_structure_db \
-t structure_db \
-o dali_result

# Parse DALI
echo "Parsing the dali result"
conda activate SAT
for FILE in dali_result/*txt ; do
sat.py aln_parse_dali \
-a $FILE \
-s structures_key.txt \
-o cluster_${SGE_TASK_ID}_result.m8
done
```

[ ]:
```
# Collect results
cat clusters/*/*m8 > dali_cluster_purity.m8
```

9 Aligning Virus protein cluster representatives against the 2.3M AFDB cluster representatives

9.1 Downloading the reps

Python script for the download

[ ]:
```
import csv
import json
import os
import sys
from urllib.request import urlopen, HTTPError
from concurrent.futures import ThreadPoolExecutor

def download_af_pdb(accession, outdir):
    url = f"https://alphafold.ebi.ac.uk/api/prediction/{accession}"
    try:
        with urlopen(url) as res:
            payload = res.read().decode("utf-8")
            obj = json.loads(payload)
            pdb_url = obj[0]["pdbUrl"]
    except HTTPError as e:
        if e.code == 404:
            sys.stderr.write(f"Accession {accession} not found, skipping...\n")
            return
        else:
            raise  # Re-raise the exception for other HTTP errors

    filename = os.path.basename(pdb_url)
    filepath = os.path.join(outdir, filename)

    # Only download if the file does not exist
    if not os.path.exists(filepath):
        with open(filepath, "wb") as fh, urlopen(pdb_url) as res:
            for chunk in res:
                fh.write(chunk)

def main():
    if len(sys.argv) != 3:
        sys.stderr.write("Usage: python download_af_pdb_from_tsv.py input.tsv␣
 ↪outdir\n")
        sys.exit(1)

    input_tsv = sys.argv[1]
    outdir = sys.argv[2]

    # Ensure output directory exists
    if not os.path.exists(outdir):
        os.makedirs(outdir)

    # Read existing filenames in the output directory
    existing_files = set(os.listdir(outdir))

    with ThreadPoolExecutor() as executor, open(input_tsv, 'r') as tsv_file:
        reader = csv.reader(tsv_file, delimiter='\t')
        next(reader)  # Skip the header row if present

        # Filter out accessions that have already been downloaded
        tasks = []
        for row in reader:
            accession = row[0]
            filename = f"{accession}.pdb"  # Assuming the files are saved as␣
 ↪'<accession>.pdb'
            if filename not in existing_files:
                task = executor.submit(download_af_pdb, accession, outdir)
                tasks.append(task)

        # Wait for all futures to complete
        for future in tasks:
            future.result()
```

```
if __name__ == "__main__":
    main()
```

Download using 2-repId_isDark_nMem_repLen_avgLen_repPlddt_avgPlddt_LCAtaxId.tsv.gz from https://afdb-cluster.steineggerlab.workers.dev/

```
[ ]: python3 download_reps.py␣
      ↪2-repId_isDark_nMem_repLen_avgLen_repPlddt_avgPlddt_LCAtaxId.tsv reps
```

9.2 Running the search

Make foldseek database

```
[ ]: #!/bin/bash
     #$ -S /bin/bash
     #$ -o ./
     #$ -e ./
     #$ -cwd
     #$ -r y
     #$ -j y
     #$ -l mem_free=7G
     #$ -l scratch=10G
     #$ -l h_rt=40:00:00
     #$ -pe smp 8

     conda activate vpSAT

     STRUCS=reps
     DB_DIR=db

     foldseek createdb $STRUCS ${DB_DIR}/AF2db_reps_db --threads 8
```

Doing the alignment

```
[ ]: #!/bin/bash
     #$ -S /bin/bash
     #$ -o ./
     #$ -e ./
     #$ -cwd
     #$ -r y
     #$ -j y
     #$ -l mem_free=7G
     #$ -l scratch=10G
     #$ -l h_rt=40:00:00
     #$ -pe smp 8

     conda activate vpSAT
```

```
STRUCS=/path/to/virus/cluster_reps/strucs
DB=path/to/db/AF2db_reps_db

$CODE/vpSAT/bin/foldseek.sh \
-i $STRUCS \
-o vir_protein_reps_vs_AF2_reps.m8 \
-d $DB \
-t 8
```

Filter on TMscore

```
[ ]: # TMscore filtering
sat.py aln_filter \
-a vir_protein_reps_vs_AF2_reps.m8 \
-o vir_protein_reps_vs_AF2_reps.TMscorefilt.m8 \
-f␣
  ↪'query,target,fident,alnlen,qlen,tlen,mismatch,gapopen,qstart,qend,tstart,tend,evalue,bits,
```