Concatenate all fastas

In [ ]:
```
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
```

MMseqs clustering

In [ ]:
```
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
```

Foldseek clustering

In [ ]:
```
# 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/stru
    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,t
-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
```

Merge structure and sequence cluster files

In [ ]:

```
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 t
# 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_clust
```

Create connection map

In [ ]:
```
# 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
```

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

In [ ]:
```
# First collect the structures
mkdir -p dali_euk_vs_euk/strucs
COUNT=0
awk '$5 > 1'  merged_clusters/merged_clusters.tsv | cut -f2 | sort -u | while
    cp seq_cluster/rep_structures/${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
```

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

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-
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
```

In [ ]:
```
# 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
```

In [ ]:
```
# 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 != $
```

Running InterProScan on all sequences

In [ ]:
```
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
```

Running DALI to determine cluster purity

Copying and organizing the structures

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

COUNT=0

awk '$5 >= 100' merged_clusters.tax.tsv | awk '$1 != "cluster_ID"' | while rea

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 ]]
    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 ]]
    cp $STRUCS/${CLUSTER_MEMBER}.pdb clusters/cluster_${CLUSTER_ID}/structures
fi

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

done
```

SGE Array to run the searches

In [ ]:
```
#!/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
```

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

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

Downloading the reps

Python script for the download

In [ ]:
```
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 (
        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 '
            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/

In [ ]:
```
python3 download_reps.py 2-repId_isDark_nMem_repLen_avgLen_repPlddt_avgPlddt_L
```

Running the search

Make foldseek database

In [ ]:
```
#!/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

In [ ]:
```
#!/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

In [ ]:
```
# 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,t
```

Benchmarking structure vs sequence searches

Benchmarking virus-virus comparisons

Prepare proteins

Steps:

1. In R, find protein clusters that have more than one sequence cluster
2. Use seqtk subseq to generate one fasta per protein cluster. -- Here, first use a python script to write a single text file with all member names for each cluster --- then, use each of those files as input to seqtk subseq

In [ ]:
```
# This is make_cluster_lists.py

import pandas as pd
import os
import sys

# Check for correct usage
if len(sys.argv) < 2:
    print("Usage: python script.py <input_file>")
    sys.exit(1)

# The first command line argument is the script name, so the second one is the
input_file_path = sys.argv[1]

# Ensure the directory for the cluster files exists
output_dir = "cluster_lists"
os.makedirs(output_dir, exist_ok=True)

# Load the dataset, assuming tab separation
df = pd.read_csv(input_file_path, sep='\t')

# Get the unique cluster IDs
unique_cluster_ids = df['cluster_ID'].unique()

# Iterate over each unique cluster ID and create a separate file
for cluster_id in unique_cluster_ids:
    cluster_df = df[df['cluster_ID'] == cluster_id]
    output_file_path = os.path.join(output_dir, f"{cluster_id}.tsv")

    # Write to file, including the header
    cluster_df.to_csv(output_file_path, sep='\t', index=False, header=False)
```

```
print(f"Finished creating individual files for each cluster ID in {output_dir}
```

In [ ]:
```
# R code
# Finding protein clusters with more than 1 sequence representative

multi_seq_rep_list <- clusters %>%
  select(cluster_ID, seq_rep) %>%
  distinct() %>%
  group_by(cluster_ID) %>%
  summarize(n = n_distinct(seq_rep)) %>%
  filter(n > 1) %>%
  pull(cluster_ID)

clusters %>%
  filter(cluster_ID %in% multi_seq_rep_list) %>%
  write_tsv("outputs/clusters_multi_seq_reps_only.tsv")
```

In [ ]:
```
# Generate input lists for seqtk subseq
python make_cluster_lists.py  clusters_multi_seq_reps_only.tsv

for FILE in cluster_lists/*tsv ; do

cut -f 4 $FILE > ${FILE}.list

done
```

In [ ]:
```
FASTA=all_full.fasta

mkdir -p cluster_fastas

for FILE in cluster_lists/*list ; do

seqtk subseq $FASTA $FILE > cluster_fastas/$(basename ${FILE%.tsv.list}).fasta

done
```

DIAMOND

In [ ]:
```
# Prepare an SGE job array
$CODE/vpSAT/bin/prepare_job_array_sge.sh \
-d cluster_fastas \
-N 10 \
-J arrays
```

In [ ]:
```
# Run DIAMOND


#!/bin/bash
#$ -S /bin/bash
#$ -o ./
#$ -e ./
#$ -cwd
#$ -r y
#$ -j y
#$ -l mem_free=10G
#$ -l scratch=10G
```

```
#$ -l h_rt=1:00:00
#$ -t 1-86

LIST=$(sed "${SGE_TASK_ID}q;d" arrays_lists/sublist_list.txt)

mkdir -p diamond/alignment_files
mkdir -p diamond/cluster_files

cat $LIST | while read LINE ; do

FILE=cluster_fastas/$LINE

DB=${FILE%.fasta}
BASE=$(basename ${FILE%.fasta})

conda activate vpSAT

diamond makedb --in $FILE -d $DB

diamond blastp \
-d $DB \
-q $FILE \
-o diamond/alignment_files/${BASE}.tsv \
--outfmt 6 \
--threads 5 \
--id 0.2 \
--query-cover 0.7 \
--subject-cover 0.7 \
--more-sensitive

conda deactivate

conda activate SAT
sat.py aln_cluster \
-a diamond/alignment_files/${BASE}.tsv \
-o diamond/cluster_files/${BASE}.cluster.tsv \
-f "query,target,pident,length,mismatch,gapopen,qstart,qend,sstart,send,evalue
-A cluster_lists/${BASE}.tsv.list

# also count total number of unique clusters
CLUSTER_COUNT=$(cut -f1 diamond/cluster_files/${BASE}.cluster.tsv | sort -u | \
N_IN_LARGEST_CLUSTER=$(cut -f1 diamond/cluster_files/${BASE}.cluster.tsv | uni
TOTAL=$(wc -l diamond/cluster_files/${BASE}.cluster.tsv | awk '{print $1}')
echo -e "${BASE}\t${CLUSTER_COUNT}\t${N_IN_LARGEST_CLUSTER}\t${TOTAL}" >> diam

done
```

MMseqs2

In [ ]:

```
#!/bin/bash
#$ -S /bin/bash
#$ -o ./
#$ -e ./
#$ -cwd
#$ -r y
#$ -j y
#$ -l mem_free=10G
#$ -l scratch=10G
```

```
#$ -l h_rt=1:00:00
#$ -t 1-86

LIST=$(sed "${SGE_TASK_ID}q;d" arrays_lists/sublist_list.txt)

mkdir -p mmseqs2/alignment_files
mkdir -p mmseqs2/cluster_files

cat $LIST | while read LINE ; do

FILE=cluster_fastas/$LINE

conda activate vpSAT

RAND=${RANDOM}_${RANDOM}
BASE=$(basename ${FILE%.fasta})

mmseqs easy-search \
$FILE \
$FILE \
mmseqs2/alignment_files/${BASE}.m8 \
$RAND \
--max-seqs 50000 \
-c 0.7 \
--cov-mode 0 \
--min-seq-id 0.2 \
--threads 5

rm -r $RAND

conda activate SAT
sat.py aln_cluster \
-a mmseqs2/alignment_files/${BASE}.m8 \
-o mmseqs2/cluster_files/${BASE}.cluster.tsv \
-f "query,target,fident,alnlen,mismatch,gapopen,qstart,qend,tstart,tend,evalue
-A cluster_lists/${BASE}.tsvncd ...list

# also count total number of unique clusters
CLUSTER_COUNT=$(cut -f1 mmseqs2/cluster_files/${BASE}.cluster.tsv | sort -u | 
N_IN_LARGEST_CLUSTER=$(cut -f1 mmseqs2/cluster_files/${BASE}.cluster.tsv | uni
TOTAL=$(wc -l mmseqs2/cluster_files/${BASE}.cluster.tsv | awk '{print $1}')
echo -e "${BASE}\t${CLUSTER_COUNT}\t${N_IN_LARGEST_CLUSTER}\t${TOTAL}" >> mmse

done
```

jackhmmer

In [ ]:

```
#!/bin/bash
#$ -S /bin/bash
#$ -o ./
#$ -e ./
#$ -cwd
#$ -r y
#$ -j y
#$ -l mem_free=10G
#$ -l scratch=10G
#$ -l h_rt=4:00:00
#$ -t 1-86
```

```
LIST=$(sed "${SGE_TASK_ID}q;d" arrays_lists/sublist_list.txt)

mkdir -p jackhmmer/alignment_files
mkdir -p jackhmmer/cluster_files

cat $LIST | while read LINE ; do

FILE=cluster_fastas/$LINE

DB=${FILE%.fasta}
BASE=$(basename ${FILE%.fasta})

conda activate hmmer

jackhmmer \
--tblout jackhmmer/alignment_files/${BASE}.jackhmmer.txt.tmp \
-E 0.01 \
--cpu 5 \
-N 3 \
$FILE \
$FILE

# Parse the jackhmmer output into a better file...
# colnames are target,query,evalue,bits
awk 'NR > 3 {print $1, $3, $5, $6}' FS=" " OFS="\t" jackhmmer/alignment_files/$

# There are a few lines starting with #, need to remove them
grep -v "^#" jackhmmer/alignment_files/${BASE}.jackhmmer.txt.tmp2 > jackhmmer/a

# Make cluster file
conda deactivate
conda activate SAT

sat.py aln_cluster \
-a jackhmmer/alignment_files/${BASE}.jackhmmer.txt \
-o jackhmmer/cluster_files/${BASE}.jackhmmer.cluster.tsv \
-f "target,query,evalue,bits" \
-A cluster_lists/${BASE}.tsv.list

# also count total number of unique clusters
CLUSTER_COUNT=$(cut -f1 jackhmmer/cluster_files/${BASE}.jackhmmer.cluster.tsv
N_IN_LARGEST_CLUSTER=$(cut -f1 jackhmmer/cluster_files/${BASE}.jackhmmer.cluste
TOTAL=$(wc -l jackhmmer/cluster_files/${BASE}.jackhmmer.cluster.tsv | awk '{pri
echo -e "${BASE}\t${CLUSTER_COUNT}\t${N_IN_LARGEST_CLUSTER}\t${TOTAL}" >> jackl

conda deactivate

done
```

Searches for Extended data fig 8D

In [ ]:

```
# MMseqs2
QUERY=queries.fasta
TARGET=/wynton/group/gladstone/users/jnomburg/projects/viral_structure/2023-09-

mkdir -p mmseqs2_tmp
```

```
mmseqs easy-search \
$QUERY \
$TARGET \
mmseqs2.m8 \
mmseqs2_tmp \
--max-seqs 50000 \
--threads 5

# DIAMOND
QUERY=queries.fasta
TARGET=/wynton/group/gladstone/users/jnomburg/projects/viral_structure/2023-09-


diamond makedb --in $TARGET -d all_full

diamond blastp \
-d all_full.dmnd \
-q $QUERY \
-o diamond.m8 \
--outfmt 6 \
--threads 5 \
--more-sensitive

# Jackhmmer
QUERY=queries.fasta
TARGET=/wynton/group/gladstone/users/jnomburg/projects/viral_structure/2023-09-

jackhmmer \
--tblout jackhmmer.m8.tmp1 \
--cpu 5 \
-N 3 \
$QUERY \
$TARGET

# Parse the jackhmmer output into a better file...
# colnames are query,target,evalue,bits
awk 'NR > 3 {print $3, $1, $5, $6}' FS=" " OFS="\t" jackhmmer.m8.tmp1 > jackhm

# There are a few lines starting with #, need to remove them
grep -v "#" jackhmmer.m8.tmp2 > jackhmmer.m8 && rm jackhmmer.m8.tmp2
```

HHpred vs DALI benchmark

Establishing benchmark set

In [ ]:
```
# see fig2.qmd, it's towards the end. This code requires some existing objects
```

Running HHpred (aka HHblits --> HHsearch)

In [ ]:
```
# Prepare a job array

$CODE/vpSAT/bin/prepare_job_array_sge.sh \
-d /wynton/group/gladstone/users/jnomburg/projects/viral_structure_newer_direc
-N 150 \
-j array
```

In [ ]:

```
#!/bin/bash
#$ -S /bin/bash
#$ -o ./
#$ -e ./
#$ -cwd
#$ -r y
#$ -j y
#$ -l mem_free=20G
#$ -l scratch=10G
#$ -l h_rt=12:00:00
#$ -l ssd_scratch=1
#$ -l scratch=200G
#$ -pe smp 10
#$ -t 1-9

conda activate hhsuite

LIST=$(sed "${SGE_TASK_ID}q;d" _lists/sublist_list.txt)

DATE=$(date)
echo "STARTING at $DATE"

UNIREF=/path/UniRef30_2023_02
PDB=/path/pdb70

mkdir -p hhblits_uniref_results
mkdir -p hhsearch_pdb_results

TEMP=${RANDOM}_${RANDOM}
echo "TEMP is $TEMP"
mkdir -p $TEMP/query_fastas

# Copy the fastas over
cat $LIST | while read LINE ; do
FILE=/wynton/group/gladstone/users/jnomburg/projects/viral_structure_newer_dir
cp $FILE $TEMP/query_fastas
done

echo "Copying UNIREF to TMPDIR"
time cp ${UNIREF}* $TMPDIR

echo "Running hhblits"
for FILE in $TEMP/query_fastas/*fasta ; do

BASE=$(basename ${FILE%.fasta})

hhblits \
-i $FILE \
-d $TMPDIR/UniRef30_2023_02 \
-blasttab hhblits_uniref_results/${BASE}.uniref.m8 \
-oa3m hhblits_uniref_results/${BASE}.uniref.a3m \
-n 1 \
-cpu 10 \
-cov 20
done

echo "Removing uniref from TMPDIR"
rm $TMPDIR/UniRef30_2023_02*
```

```
echo "Copying PDB to TMPDIR"
time cp ${PDB}* $TMPDIR

echo "Running hhsearch"
for FILE in hhblits_uniref_results/*a3m ; do

BASE=$(basename ${FILE%.a3m})

time hhsearch \
-i $FILE \
-d $TMPDIR/pdb70 \
-blasttab hhsearch_pdb_results/${BASE}.pdb.m8 \
-cov 20 \
-cpu 10
done

rm -r $TEMP

DATE=$(date)
echo "ENDED at $DATE"
```

In [ ]:
```
# Filter for E < 0.001
COUNT=0

for FILE in hhsearch_pdb_results/* ; do

awk '$11 < 0.001' $FILE  >> 2024-05-02_benchmark_hhsearch_E0.001.m8

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

done
```

Running DALI

In [ ]:
```
$CODE/vpSAT/bin/prepare_job_array_slurm.sh \
-d benchmark_viral_dali_db \
-N 10 \
-J array10
```

In [ ]:
```
#!/bin/bash
#SBATCH --account=ac_ribosome
#SBATCH --partition=lr6
#SBATCH --qos=lr_normal
#SBATCH --ntasks=20
#SBATCH --cpus-per-task=1
#SBATCH --time=20:00:00
#SBATCH --array=1-132

module load gcc openmpi

PDB=/path/pdb25_db

LIST=$(sed "${SLURM_ARRAY_TASK_ID}q;d" array10_lists/sublist_list.txt)

echo "Starting. SLURM_ARRAY_TASK_ID is $SLURM_ARRAY_TASK_ID"

mkdir -p dali_results
```

```
# Set up temp dir
TEMP=${RANDOM}_${RANDOM}
mkdir -p ${TEMP}/query_db
echo "TEMP: $TEMP"

# Copy over query
cat $LIST | while read LINE ; do
FILE=benchmark_viral_dali_db/$LINE
cp $FILE ${TEMP}/query_db
done

# Copy over target
cp -r $PDB ${TEMP}

# Run DALI
$CODE/vpSAT/bin/dali.sh \
-q ${TEMP}/query_db \
-t ${TEMP}/pdb25_db \
-o dali_results \
-n 20

rm -r $TEMP
```

In [ ]:
```
$CODE/vpSAT/bin/prepare_job_array_slurm.sh \
-d dali_results \
-N 10 \
-J parse_dali
```

In [ ]:
```
#!/bin/bash
#SBATCH --account=ac_ribosome
#SBATCH --partition=lr6
#SBATCH --qos=lr_normal
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=1
#SBATCH --time=0:50:00
#SBATCH --array=1-132

conda activate SAT

mkdir -p dali_results_parsed

LIST=$(sed "${SLURM_ARRAY_TASK_ID}q;d" parse_dali_lists/sublist_list.txt)

cat $LIST | while read LINE ; do

FILE=dali_results/$LINE
BASE=$(basename ${FILE%.txt})

sat.py aln_parse_dali \
-a $FILE \
-s benchmark_viral_key.txt \
-o dali_results_parsed/${BASE}.m8

done
```

In [ ]:
```
# Concatenate everything
head -n1 dali_results_parsed/AEctA.m8 > 2024-05-02_benchmark_dali_results.m8
```

```
for FILE in dali_results_parsed/* ; do
tail -n +2 $FILE >> 2024-05-02_benchmark_dali_results.m8
done
```

Searching the transporter classification database

Downloading TCDB classifications

List of PDB structures per class: https://www.tcdb.org/cgi-bin/projectv/public/pdb.py

In [ ]:
```
# R code
tcdb <- read_tsv("inputs/tcdb_analysis/pdb.txt", col_names = c("PDB", "TCDB_ID"


tcdb %>%
  distinct() %>%

  # Weird, wonky, truncated
  filter(PDB != "1HXI") %>%

  group_by(Classification) %>%
  sample_n(size=5, replace=T) %>%
  write_tsv("outputs/tcdb_5_per_type.tsv")
```

Downloading PDBs

In [ ]:
```
COUNT=0
tail -n +2 ../tcdb_5_per_type.tsv  | cut -f1 | while read LINE; do

if [[ ! -f ${LINE}.pdb ]] ; then
    wget https://files.rcsb.org/download/${LINE}.pdb
fi

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

done
```

Making DALI databases

In [ ]:
```
mkdir -p tcdb_db
for FILE in tcdb_pdbs/*pdb ; do

PDB=$(basename ${FILE%.pdb})

if [[ ! -f tcdb_db/${PDB}*.dat ]] ; then

import.pl \
--pdbfile $FILE \
--pdbid $PDB \
--dat tcdb_db \
```

```
--clean

fi

done


# Take just the first chain for multi-chain entries.
mkdir tcdb_db_final

for FILE in tcdb_pdbs/*pdb ; do

PDB=$(basename ${FILE%.pdb})

LIST=$(ls tcdb_db/${PDB}*)
FIRST_FILE=$(echo $LIST | cut -d ' ' -f1)

cp $FIRST_FILE tcdb_db_final

done
```

Spike in AFDB ENT1-4

note that the AFDB ENT1 and ENT2 were removed during downstream visualization/analysis in R.

cd AFDB_ENT_PDBs

wget https://alphafold.ebi.ac.uk/files/AF-Q99808-F1-model_v4.pdb && mv AF-Q99808-F1-model_v4.pdb ENT1.pdb wget https://alphafold.ebi.ac.uk/files/AF-Q14542-F1-model_v4.pdb && mv AF-Q14542-F1-model_v4.pdb ENT2.pdb wget https://alphafold.ebi.ac.uk/files/AF-Q9BZD2-F1-model_v4.pdb && mv AF-Q9BZD2-F1-model_v4.pdb ENT3.pdb wget https://alphafold.ebi.ac.uk/files/AF-Q7RTT9-F1-model_v4.pdb && mv AF-Q7RTT9-F1-model_v4.pdb ENT4.pdb

for FILE in AFDB_ENT_PDBs/* ; do

BASE=(*basename*{FILE%.pdb})

import.pl \ --pdbfile $FILE -- -pdbid${BASE} \ --dat tcdb_db_final \ --clean || echo "${BASE} FAILED"

done

Search

In [ ]:
```
$CODE/vpSAT/bin/dali.sh \
-q query_db \
```

```
-t tcdb_db_tmp \
-o dali_result \
-d /wynton/group/gladstone/users/jnomburg/software/DaliLite.v5.newinstall/DaliL


conda activate SAT

for FILE in dali_result/* ; do

BASE=$(basename ${FILE%A.txt})

sat.py aln_parse_dali \
-a $FILE \
-s fake_key.txt \
-o dali_result_parsed/${BASE}.m8

done
```

In [ ]: