Finding Paralog Targets for Neglected Diseases

By

Jeremy Singer

# Abstract

This paper describes a method that can be used to discover and repurpose existing drugs and drug targets by discovering cross species genomic sequence similarities. It uses public domain databases (ChEMBL, EupathDB) and open source software to find measures of sequence similarity with existing targets.

This method can be applied to pathogens with at least a medium sized genome (several thousand genes.) *Neglected tropical diseases* caused by pathogenic protists are good subjects for this approach because many have genomes of sufficient size and because many have genomic features in common with organisms for which there are known targets.

The genome of the apicomplexan parasite *Plasmodium falciparum*, which is responsible for the most virulent form of malaria, was chosen to validate a method that identifies paralogs to existing disease targets because it has known cross-species targets.

ChEMBL provides a PostgreSQL database that contains a list of thousands of targets and target protein sequences as well as ligands for those targets. Using this database and open source software, this paper identified *[<number>]* distinct drugs and *[<number>]* targets validating this approach.

Seven other pathogens (*Plasmodium vivax, Toxoplasma gondii, Trypanosoma brucei, Trypanosoma cruzi, Leishmania major, and Entamoeba )* were also downloaded and run through the same pipeline, identifying potential targets and drugs.

# Introduction

*Neglected Tropical Diseases* are those diseases that affect tropical areas underserved for health care due to the poverty of those areas. These diseases affect over a billion people in over 149 countries, and damage the economies of these areas at a cost of many billions of dollars[[1]](#footnote-1).

Repurposing drugs and generating leads for finding new drugs by repurposing targets could be a cost -effective way for combating these diseases. Finding new targets can be difficult, as it requires understanding many specific details for each pathogen. A systematic method of discovering new targets that does not require this specific understanding can reduce the cost and effort of finding these targets.

This paper describes a method for *Drug Repurposing* and *Target Repurposing* based on discovering similarities between existing targets and pathogen genomes.

ChEMBL provides a downloadable database that includes drug targets and drug information for those targets, as well as amino acid sequences of the protein targets[[2]](#footnote-2). Drug targets tend to be proteins that are important enough to the organism to which they belong that they tend to be conserved. If we can find a protein sequence in a disease organism that is sufficiently similar to a known target, the protein may be a promising target in that organism, and drugs used against that target may be successfully used in that organism.

The analysis pipeline uses **BLASTP** [[3]](#footnote-3) or **jackhmmer** [[4]](#footnote-4) to produce similarity reports, parse the results, and upload to supplementary tables in the PostgreSQL database.

This analysis pipeline was first applied to the genome of *Plasmodium falciparum* using both BLASTP and HMMER to generate similarity statistics, and custom scripts included in the Appendix. The scores returned from these two different programs were compared to evaluate which could provide better discrimination criteria of useful targets and drugs.

*Plasmodium falciparum* was chosen for this evaluation because it is the most significant of these neglected diseases. In 2018, there were over 228 million cases of malaria worldwide, causing over 408 thousand deaths.[[5]](#footnote-5) Emerging drug resistance to existing drugs such as choloroquin and sulfadoxine-pyrimethamine, as well as quinine increase demand for new drugs that are more effective.[[6]](#footnote-6),[[7]](#footnote-7)

Database queries identify promising targets and drugs according to criteria developed and implemented in R.

In addition to *p. falciparum*, we processed the following additional pathogens using [*preferred method]*:

[*pathogen list, see abstract].* The statistics were loaded into supplementary tables in the PostgreSQL database.

Queries using the existing ChEMBL\_25 database, in combination with these similarity statistics were used to identify candidate targets and drugs for each of these pathogens.

# Materials and Methods



Figure 1: Processing overview

PlasmoDB and Chembl provide the data that are analyzed in this process flow.

Amino acid sequences of putative *Open Reading Frames* (ORFs) for *Plasmodium falciparum 3D7* were downloaded from *PlasmoDB.org* as file**PlasmoDB-46\_Pfalciparum3D7\_ORFs\_AA.fasta**.[[8]](#footnote-8),[[9]](#footnote-9)

The FASTA formatted dataset consists of all ORFs in a single file. Each ORF consists of a header line followed by a number of lines containing multiple characters of single letter codes representing an amino acid.

Header lines are formatted according to two different patterns. The first pattern encodes the ORF id that is comprised of the organism code, chromosome, and identifier. The second pattern contains a type identifier that identifies the record as belonging to the mitochondrion, and contains a unique identifier for the ORF id. A script fans out the ORF records into individual files in a directory structure having a separate subdirectory structure for each chromosome. (See script 7.2.1. fan\_out\_fasta.R ).



Figure 2: ORF header structure determines fan out destination

The set of target sequences comes from the ChEMBL\_25 PostgreSQL database and downloaded by a *psql* script (See 7.1.1. chembl\_25\_targets.sql) as file **chembl\_targets.txt**.

These targets are converted by a Perl script (See 7.1.2. split\_to\_fasta.pl) to FASTA formatted sequences (See 7.1.2. split\_to\_fasta.pl) creating file **component\_sequences.fa**.

## Gathering BLASTP statistics

The **makeblastdb** utility converts the FASTA formatted file of targets to a **BLASTP** searchable database in the **~/genomes/blast\_targets** directory.

A script applies the **BLASTP** utility to each of the ORFs for the *P. falciparum* organism against the BLAST database (see 7.2.2. do\_all\_blast.sh).

A *bash* script ( 7.2.5. do\_all\_blast\_stats.sh) applies a Perl script (7.2.3. extract\_header.pl) that parses each BLAST report into a *.stats* file for each ORF.

A bash script ( 7.2.4. make\_blast\_statistics.sh ) consolidates all the <ORF>.*blastp.txt.stats* files into a tab delimited text file, **blast\_statistics.txt**.

The **blast\_statistics** table, which has been created previously (7.2.6. create\_blast\_statistics\_tbl.sql), is populated by SQL script (7.2.7. import\_p\_falciparum.sql).

## Gathering HMMER statistics

A script applies the **jackhammer** utility to each of the ORFs of the *p. falciparum* organism, generating a report and a summary file for each (see 7.3.1. do\_all\_jackhmmer.sh).

A *bash* script ( 7.3.3. do\_all\_hmmer\_stats.sh) applies a Perl script ( 7.3.2. extract\_hmm\_summary.pl) that extracts the **jackhmmer** statistics from the reports and summaries and produces a consolidated tab delimited file for all ORFs called **hmm\_stats.txt**.

An import SQL script (7.3.5. import\_hmmer\_statistics.sql) imports these statistics into previously created tables (see 7.3.4. create\_hmmer\_stats\_tbls.sql).

Consolidated statistics records having the same ORF/target were downloaded using this join:

# Discussion and Results

The “targetness” of a protein has to do with how indispensable its function is to the disease organism. This description of the nature of protein targets suggests that paralogous proteins in our organism of interest could also be targets, if they are sufficiently similar to existing targets. The closeness of the match will suggest that the function of the protein has been conserved between the previously identified target organism and our organism of interest. Those sequences in the pathogen organism which are most necessary for its survival are also least likely to change, as mutation would tend to impair functions necessary for survival. At the same time, we are searching exactly for those critically necessary proteins as targets for drugs that can impair them.

To find likely targets in the genome, we need to measure similarity between ORFs from its genome and our target database. When we have computed these similarities, we need to chose threshold criteria for filtering the most promising candidates.

Both **BLASTP** and **jackhmmer** score similarity between amino acid sequences by aligning query and target sequences. Their approach to scoring, however, differs. **BLASTP** detects similarities by scoring the likelihood of successive letters of the amino acid being the same in query and target sequences in comparison to chance. It uses transition matrices based on conserved sequences in different genomes to assign the likelihood that that a particular amino acid will follow another. By contrast, **jackhmmer** uses hidden Markoff models (HMM) that assess patterns by looking for larger domains. We might therefore expect that this approach might be more discriminating, as it takes into account similarities at the level of protein domains. In effect, the **BLASTP** approach uses the limited case of a hidden Markoff model of length two, and then sums the scores, adding for more likely matches, zero for matches for no match, and negative numbers for mismatches. Gaps also result in penalties.



Figure 3: Understanding BLAST statistics

Each match and alignment contains various scores which BLASTP computed for it.

The “Expect” statistic measures the probability that the matches could occur by chance. For example: if we assume equal probability for the occurrence of a particular amino acid in a sequence (.05), this is the expected value for a random match of this length sequence. This is much too high an expected value to accept this as a non-random match. Results with an expected value >= .001 were filtered out.

The ratio of exact matches to the length of the *subject* gives the *identities* statistics; The ratio of conserved matches (including identities and functionally similar amino acids) to the total number of characters in the *subject* gives the *positives* statistics. In the same way, gaps are tabulated and compared with the total subject length. The score statistic gives BLASTP’s estimate of the query and target sequences similarity by accumulating the positive matches weighted by the match length minus gap penalties.

We use score in this paper to rank similarity of proteins. It is computed by adding successive matching scores using the BLOSUM62 scoring matrix to rate the “goodness” of a particular match given log odds of the frequencies of the given amino acids for the particular “blocks” database built with sequences having no more than 62% in common. Experiments have shown that this matrix is one of the best for detecting even weak protein similarities. The matrix values are calculated from these data by calculating the log base 2 odds of the observed frequencies vs expected, rounded to the nearest unit.  A score of zero indicates that the frequency with which a given two amino acids were found aligned in the database was as expected by chance, while a positive score indicates that the alignment was found more often than by chance, and negative score indicates that the alignment was found less often than by chance. The score is cumulative (computed for each residue in the query string.)

Figure 4: ORF.FASTA.summary from jackhmmer

The summary for a query may hit multiple targets. Each target record is repeated for each domain that **jackhmmer** matches. We are only interested in the total score for the full sequence. The Perl script (7.3.2. extract\_hmm\_summary.pl) creates a single record per ORF/target by de-duping these values. Score is used in a similar way to BLAST, but is computed differently.

## Choosing appropriate metrics for selecting target candidates

**BLASTP** and **jackhmmer** also compute *expected* values, which can be very small numbers. In contrast, the *scores* values are always integers that are easily comparable.

Other statistics computed by BLASTP are *identities*, which are the percentage of exact matches, and *positives*, which are inexact matches that conserve function because the amino acids involved in the comparison function in a compatible manner to each other in the protein.



Figure 6: Pairwise comparison of BLAST metrics

An advantage of the *score* statistic is that it is additive. While both statistics reflect the cumulative values for matches, products of probability scores that accumulate in the *expect* statistic can lose resolution during computation. Although *expect* should follow (inversely) to score, the comparison here is uninformative because of loss of resolution at low values. *Positives* trend like *identities* but show *greater than or equal*  relationship to them. *Score* also trends with *positives,*  showing greater detail.

## Comparison of BLASTP and jackhmmer scores

The default inclusion threshold for **jackhmmer** is much more stringent than the default threshold for **BLASTP**. The **blast\_statistics** table imported 562,039 records, where the **hmmer\_statistics** table contains only 127,306 records.

A query shows that there are 20,178 **blast\_statistics** records that join with the **hmmer\_statistics** records with the same OFR/target. A tab delimited file named **consolidated\_stats.txt** was downloaded with these records (see 6.4.1. consolidated\_orf\_target.sql).



Figure 5: BLASTP vs HMM scoring comparison

The figure shows that the two kinds of scores generally trend the same, but there are many excursions for *hmm\_score* that are much greater than the *blast score* would lead one to expect. (See 6.4.2. compare\_scores.R) Thismay be attributed to **jackhmmers’**s increased sensitivity to structure over **BLASTP**.

While the **jackhmmer** statistics may provide us with a more accurate scoring of the similarity of parasite ORFs with our target universe, we can use the broader **BLASTP** statistics to provide statistical guidelines for selecting a significance threshold. By pairing these two sets of statistics, *blast statistics* can help us inform useful methods for selection from *jackhmmer statistics*.

# References

Altschul, S. F., T. L. Madden, A. A. Schäffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman. “Gapped BLAST and PSI-BLAST: A New Generation of Protein Database Search Programs.” *Nucleic Acids Research* 25, no. 17 (September 1, 1997): 3389–3402. https://doi.org/10.1093/nar/25.17.3389.

Aurrecoechea, Cristina, John Brestelli, Brian P. Brunk, Jennifer Dommer, Steve Fischer, Bindu Gajria, Xin Gao, et al. “PlasmoDB: A Functional Genomic Database for Malaria Parasites.” *Nucleic Acids Research* 37, no. Database issue (January 2009): D539-543. https://doi.org/10.1093/nar/gkn814.

Gaulton, Anna, Anne Hersey, Michał Nowotka, A. Patrícia Bento, Jon Chambers, David Mendez, Prudence Mutowo, et al. “The ChEMBL Database in 2017.” *Nucleic Acids Research* 45, no. D1 (January 4, 2017): D945–54. https://doi.org/10.1093/nar/gkw1074.

Khan, M. Aslam, Raymond A. Smego, Syed Tabish Razi, and M. Asim Beg. “Emerging Drug--Resistance and Guidelines for Treatment of Malaria.” *Journal of the College of Physicians and Surgeons--Pakistan: JCPSP* 14, no. 5 (May 2004): 319–24. https://doi.org/05.2004/JCPSP.319324.

“PlasmoDB Download Files.” Accessed February 12, 2020. https://plasmodb.org/common/downloads/Current\_Release/Pfalciparum3D7/fasta/data/.

Wheeler, Travis J., and Sean R. Eddy. “Nhmmer: DNA Homology Search with Profile HMMs.” *Bioinformatics* 29, no. 19 (October 1, 2013): 2487–89. https://doi.org/10.1093/bioinformatics/btt403.

WHO. “WHO | Responding to Antimalarial Drug Resistance.” Accessed February 12, 2020. http://www.who.int/malaria/areas/drug\_resistance/overview/en/.

WHO. “WHO | World Health Organization.” Accessed February 12, 2020. http://www.who.int/neglected\_diseases/diseases/en/.

“World Malaria Report 2019.” Accessed February 12, 2020. https://www.who.int/news-room/feature-stories/detail/world-malaria-report-2019.

# Appendix: Scripts

These scripts can be found in the **supplements** directory for this **git** repository.

## BLAST Targets

### chembl\_25\_targets.sql

Run this script at the command line of psql attached as chembl\_25 in the blast\_targets directory.

This script creates **chembl\_targets.txt** file.

\copy (select td.chembl\_id, cs.sequence from target\_dictionary td join target\_components tc on td.tid = tc.tid join component\_sequences cs on tc.targcomp\_id=cs.component\_id) to chembl\_targets.txt

### split\_to\_fasta.pl

Run this script from the *bash* command line in the **blast\_targets** directory: **perl split\_to\_fasta.pl**

This script creates the **component\_sequences.fa**file which can be found in the **supplements**/**blast\_targets** directory.

#######################################

# split\_to\_fasta.pl

# input recs: <key><delim><sequence>

# output : rec1 = ><key>

# rec2 = <sequence>

#######################################

my $infile = 'chembl\_targets.txt';

my $outfile = 'component\_sequences.fa';

my $delim = '\t';

open(IN, $infile) or die("Unable to open $infile\n");

my @lines = <IN>;

close(IN);

open(OUT,">",$outfile) or die ("Unable to open $outfile\n");

foreach my $line(@lines)

{

my @rec = split($delim,$line);

if (scalar(@rec) > 1)

{

print OUT ">$rec[0]\n";

print OUT "$rec[1]\n";

}

}

close(OUT);

exit(0);

## Process FASTA

### fan\_out\_fasta.R

This script is run within RStudio to fan out the single FASTA file from *Plasmodb* into separate directories by organism and chromosome. Each ORF is separated out for ease of obtaining BLAST and HMMER reports for each.

# Fan out AA\_fasta file from plasmodbc

# based on the structure of Plasmodium AA\_orf files.

# FASTA headers come in two varieties:

# 1. >Pf3D7\_01\_v3-1-60871-61059 | organism=Plasmodium\_falciparum\_3D7 | location=Pf3D7\_01\_v3:60871-61059(+) | length=63 | sequence\_SO=chromosome

# ^unique ORF identifier---^ <other stuff> <sequence\_SO=<ORF type> i.e. chromosome, apicoplast, mitochondrial

# ^head indicator

# ^organism

# ^chromosome id

# ^orf\_name

# 2. >Pf\_M76611-5-344-75 | organism=Plasmodium\_falciparum\_3D7 | location=Pf\_M76611:75-344(-) | length=90 | sequence\_SO=mitochondrial\_chromosome

# ^orfname---------^ <other stuff> sequence\_SO=mitochondrial\_chromosome

# parsing strategy is: for non-mitochondrial, parse out chromosome\_name, orf\_name.

# For mitochondrial, orfname is one piece.

library(stringr)

setwd('~/genomes')

aa\_file=file.choose()

aa=read.table(file=aa\_file,header = FALSE, sep='~', stringsAsFactors = FALSE)

aa=aa[!is.na(aa[,1]),] # filter out NA

firstrec=aa[1] # scalar

aa=data.frame(lines=aa, stringsAsFactors = FALSE)

parsed=strsplit(firstrec,'\_')

organism\_pref=substring(parsed[[1]][1],2)

# make a directory for this organism

system(paste('mkdir',organism\_pref))

orf\_headers=aa[substr(aa[,1],1,1)=='>' ,]

mi\_headers=orf\_headers[grep('sequence\_SO=mitochondrial\_chromosome',orf\_headers)]

chrom\_headers = setdiff(orf\_headers, mi\_headers)

parsed=strsplit(chrom\_headers,'\_')

chromosomes=unique(sapply(parsed,function(p){p[2]}))

# make a directory for each chromosome

for (chromosome in chromosomes){

dirname=paste(organism\_pref,chromosome,sep='/')

system(paste('mkdir',dirname))

}

dirname=paste(organism\_pref,'mitochondrion', sep='/')

system(paste('mkdir',dirname))

orf.df = data.frame(line='')

orf\_name=''

orf.df=data.frame(line='')

for(orf\_line in aa[,1]){

print(paste('orf\_line: ',orf\_line))

if (substr(orf\_line,1,1)=='>'){

print('FASTA header line')

if ( is.na(orf\_name) || nchar(orf\_name) > 0){

orf\_name=paste0(orf\_name,'.FASTA')

print("write statement")

write\_dir\_name = paste(organism\_pref,chromosome, orf\_name,sep='/')

write.table(orf.df, file=write\_dir\_name,row.names = FALSE,col.names = FALSE, quote=FALSE)

}

orf.df = data.frame(line=orf\_line)

if ( length(grep('mitochondrial',orf\_line)) > 0){

chromosome='mitochondrion'

print(paste("Chromosome:", chromosome))

parsed=unlist(strsplit(orf\_line,' '))

orf\_name=substr(parsed[1],2,nchar(parsed[1]) -1)

} else {

parsed=unlist(strsplit(orf\_line,' '))

parsed=unlist(strsplit(parsed[1],'\_'))

chromosome=parsed[2]

orf\_name=parsed[3]

}

print(paste('chromosome:',chromosome,', orf\_name:', orf\_name))

} else {

print('rbind FASTA sequence')

orf\_line.df=data.frame(line=orf\_line)

orf.df = rbind(orf.df, orf\_line.df);

}

}

if (is.na(orf\_name) || nchar(orf\_name) > 0){

print("write statement")

write\_dir\_name = paste(organism\_pref,chromosome, orf\_name,sep='/')

write.table(orf.df, file=write\_dir\_name,row.names = FALSE,col.names = FALSE,quote=FALSE)

}

### do\_all\_blast.sh

Run this script in the genome directory.

Specify the *Organism\_dir* on the command line.

#!/bin/bash

if [ -z $1 ]

then

while [ -z $org\_dir ]

do

read -p "Organism directory: " -a org\_dir

done

else

org\_dir=$1

fi

echo $org\_dir

for chrom\_dir in $( ls -d $org\_dir\*/ );do

for orf in $( ls $chrom\_dir\*.FASTA);do

echo "BLASTP " $orf

blastp -db ~/blast\_targets/chembl\_25\_targets -query $orf -num\_alignments 10 -out ${orf}.blastp.txt

done

done

### extract\_header.pl

This Perl script extracts statistics from BLAST reports.

use Switch 'fallthough';

my @lines = <STDIN>;

my $phase = 0;

my @rec = {};

my $rec\_string;

my %recs = ();

my $query;

foreach my $line(@lines)

{

switch($phase){

case 0 {

if ( $line =~ m/Query=\s\*(\S+)/)

{

$query = $1;

}

if ( $line =~ m/\>\s\*(\S+)/) # orf id

{

$phase = 1;

$rec[0] = $1;

}

}

case 1 {

if ( $line =~ m/Length\=(\S+)/ ){

$rec[scalar(@rec)] = $1;

$phase = 2;

}

}

case 2 {

if ( $line =~ m/Score\s\=\s(\S+)/){

$rec[scalar(@rec)] = $1;

$line =~ m/Expect\s\=\s(\S+),/;

$rec[scalar(@rec)] = $1;

$phase = 3;

}

}

case 3 {

if ( $line =~ m/Identities\s\=\s\S+\s\((\S+)\%/){

$rec[scalar(@rec)] = $1;

$line =~ m/Positives\s\=\s\S+\s\((\S+)\%/;

$rec[scalar(@rec)] = $1;

$line =~ m/Gaps\s\=\s\S+\s\((\S+)\%\)/;

$rec[scalar(@rec)] = $1;

$rec\_string = join("\t",@rec);

$recs{$rec\_string} = 1;

$phase = 0;

@rec = {};

}

}

}

}

foreach my $record(keys %recs){

print "$query\t$record\n";

}

### make\_blast\_statistics.sh

Create the blast\_statistics file by concatenating all the \*.blast.stat files.

#!/bin/bash

if [ -z $1 ]

then

while [ -z $org\_dir ]

do

read -p "Organism directory: " -a org\_dir

done

else

org\_dir=$1

fi

echo $org\_dir

echo "orf\_id target query\_length score expect identities positives gaps" > blast\_statistics.txt

for chrom\_dir in $( ls -d $org\_dir\*/ );do

cat $( ls $chrom\_dir\*.blastp.txt.stats) >> blast\_statistics.txt

done

### do\_all\_blast\_stats.sh

Apply the Perl script (**extract\_header.pl**) that extracts statistics to all the BLAST reports.

#!/bin/bash

if [ -z $1 ]

then

while [ -z $org\_dir ]

do

read -p "Organism directory: " -a org\_dir

done

else

org\_dir=$1

fi

echo $org\_dir

for chrom\_dir in $( ls -d $org\_dir\*/ );do

for orf in $( ls $chrom\_dir\*.blastp.txt);do

echo "BLAST stats " $orf

perl ~/genomes/extract\_header.pl < ${orf} > ${orf}.stats

done

done

### create\_blast\_statistics\_tbl.sql

Enter this at the psql command line:

CREATE TABLE blast\_statistics

(

sk\_blast\_statistics SERIAL -- synthetic primary key

, tax\_id bigint -- NCBI taxonomy id of target

, organism character varying(100) -- convenience name of organism

, chromosome character varying(50)

, orf\_id character varying(50)

, target character varying(50) -- typically, chembl\_id

, query\_length int

, score numeric

, expect numeric

, identities numeric

, positives numeric

, gaps numeric

, import\_date timestamp not null default clock\_timestamp()

);

CREATE TABLE blast\_statistics\_import

(

orf\_id character varying(50)

, target character varying(50)

, query\_length int

, score numeric

, expect numeric

, identities numeric

, positives numeric

, gaps numeric

);

### import\_p\_falciparum.sql

(run this at psql prompt logged in as chembl\_25:

truncate table blast\_statistics\_import;

\copy blast\_statistics\_import from 'blast\_statistics.txt' delimiter E'\t' CSV HEADER

insert into blast\_statistics

( tax\_id, organism, orf\_id, target, query\_length, score, expect, identities, positives, gaps)

SELECT 36329 -- tax\_id

, 'Plasmodium falciparum 3D7'

, orf\_id

, target

, query\_length

, score

, expect

, identities

, positives

, gaps

FROM blast\_statistics\_import;

## HMM targets

### do\_all\_jackhmmer.sh

#!/bin/bash

if [ -z $1 ]

then

while [ -z $org\_dir ]

do

read -p "Organism directory: " -a org\_dir

done

else

org\_dir=$1

fi

echo $org\_dir

for chrom\_dir in $( ls -d $org\_dir\*/ );do

for orf in $( ls $chrom\_dir\*.FASTA);do

echo "jackhmmer " $orf

jackhmmer --domtblout $orf.summary -o $orf.hmm.txt $orf ~/hmmer\_targets/component\_sequences.fa

done

done

### extract\_hmm\_summary.pl

(in the ~/genomes directory.)

#!/bin/perl

use Switch;

if (scalar(@ARGV) < 1) {die "No filename passed.\n";}

my $text\_fn = $ARGV[0];

my $summary\_fn;

# print $text\_fn,"\n";

$summary\_fn = $text\_fn;

$summary\_fn =~ s/.hmm.txt/.summary/;

# print $summary\_fn,"\n";

my @lines;

open($IN, "<", $summary\_fn ) or die "Can't open $summary\_fn\n";

@lines = <$IN>;

close($IN);

# print "Lines: ",scalar(@lines), "\n";

my %target;

foreach my $line(@lines){

if ( $line =~ m/^(CHEMBL\S+)\s+(\S+)\s+(\S+)\s+(\S+)\s+(\S+)\s+(\S+)\s+(\S+)\s+(\S+)\s+/ ) {

if ( ! exists $target{$1} ) { # prevent duplicate line for a target match

print $1,"\t", $3, "\t", $4,"\t",$6, "\t", $7, "\t", $8, "\n";

$target{$1} = 1;

}

}

}

### do\_all\_hmmer\_stats.sh

Run this script in the **~/genomes** directory to create **hmm\_stats.txt** file which gathers all the generated stats.

#!/bin/bash

if [ -z $1 ]

then

while [ -z $org\_dir ]

do

read -p "Organism directory: " -a org\_dir

done

else

org\_dir=$1

fi

echo $org\_dir

echo "target tlen orf qlen evalue score" > hmm\_stats.txt

for chrom\_dir in $( ls -d $org\_dir\*/ );do

for orf in $( grep -L "\[No hits" $chrom\_dir\*hmm.txt ); do

perl ~/genomes/extract\_hmm\_summary.pl $orf >> hmm\_stats.txt

done

done

### create\_hmmer\_stats\_tbls.sql

Import this from the psql command line as chembl\_25 user.

CREATE TABLE hmmer\_statistics

(

hmmer\_statistics\_id SERIAL

, tax\_id numeric

, organism character varying(100)

, chromosome character varying(50)

, target character varying(50)

, tlen int

, orf character varying(50)

, qlen int

, evalue numeric

, score numeric

, import\_date timestamp not null default clock\_timestamp()

);

CREATE TABLE hmmer\_statistics\_import

(

target character varying(30)

, tlen int

, orf character varying(30)

, qlen int

, evalue numeric

, score numeric

);

### import\_hmmer\_statistics.sql

Import this script from the psql command line as user chembl\_25.

This script is in **~/genomes**.

truncate table hmmer\_statistics\_import;

\copy hmmer\_statistics\_import from 'hmm\_stats.txt' delimiter E'\t' CSV HEADER

insert into hmmer\_statistics

( target, tlen, orf, qlen, evalue, score)

select target, tlen, orf, qlen, evalue, score

from hmmer\_statistics\_import;

## Consolidated statistics analysis

### consolidated\_orf\_target.sql

\copy ( select h.orf

, b.target,b.score as blast\_score

, h.score as hmmer\_score

, b.expect as blast\_expect, h.evalue

FROM blast\_statistics b

join hmmer\_statistics h

on b.orf\_id = h.orf and b.target = h.target)

to C:\Users\Jeremy-satellite\Documents\RBIF120\consolidated\_stats.txt CSV delimiter ' ‘

### compare\_scores.R

# Scores comparison

consolidated\_stats=read.csv(file = "consolidated\_stats.txt", sep='\t', stringsAsFactors = FALSE)

attach(consolidated\_stats)

plot(blast\_score,hmmer\_score,main='Comparison of BLASTP vs HMM scores for P. falciparum with targets')

abline(a=0,b=median(hmmer\_score/blast\_score)+mad(hmmer\_score/blast\_score),col='red')

detach()

1. “WHO | World Health Organization,” WHO, 1, accessed February 12, 2020, http://www.who.int/neglected\_diseases/diseases/en/. [↑](#footnote-ref-1)
2. Anna Gaulton et al., “The ChEMBL Database in 2017,” *Nucleic Acids Research* 45, no. D1 (January 4, 2017): D945–54, https://doi.org/10.1093/nar/gkw1074. [↑](#footnote-ref-2)
3. S. F. Altschul et al., “Gapped BLAST and PSI-BLAST: A New Generation of Protein Database Search Programs,” *Nucleic Acids Research* 25, no. 17 (September 1, 1997): 3389–3402, https://doi.org/10.1093/nar/25.17.3389. [↑](#footnote-ref-3)
4. Travis J. Wheeler and Sean R. Eddy, “Nhmmer: DNA Homology Search with Profile HMMs,” *Bioinformatics* 29, no. 19 (October 1, 2013): 2487–89, https://doi.org/10.1093/bioinformatics/btt403. [↑](#footnote-ref-4)
5. “World Malaria Report 2019,” accessed February 12, 2020, https://www.who.int/news-room/feature-stories/detail/world-malaria-report-2019. [↑](#footnote-ref-5)
6. M. Aslam Khan et al., “Emerging Drug--Resistance and Guidelines for Treatment of Malaria,” *Journal of the College of Physicians and Surgeons--Pakistan: JCPSP* 14, no. 5 (May 2004): 319–24, https://doi.org/05.2004/JCPSP.319324. [↑](#footnote-ref-6)
7. “WHO | Responding to Antimalarial Drug Resistance,” WHO, accessed February 12, 2020, http://www.who.int/malaria/areas/drug\_resistance/overview/en/. [↑](#footnote-ref-7)
8. Cristina Aurrecoechea et al., “PlasmoDB: A Functional Genomic Database for Malaria Parasites,” *Nucleic Acids Research* 37, no. Database issue (January 2009): D539-543, https://doi.org/10.1093/nar/gkn814. [↑](#footnote-ref-8)
9. “PlasmoDB Download Files,” accessed February 12, 2020, https://plasmodb.org/common/downloads/Current\_Release/Pfalciparum3D7/fasta/data/. [↑](#footnote-ref-9)