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, EnSEMBL, NCBI) 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

# Results

# Discussion

# References

1. Neglected tropical diseases, <https://www.who.int/neglected_diseases/diseases/en/>

Klug DM, Gelb MH, Pollastri MP. Repurposing strategies for tropical disease drug discovery. *Bioorg Med Chem Lett*. 2016;26(11):2569–2576. doi:10.1016/j.bmcl.2016.03.103<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4853260/>

1. Gaulton A, Hersey A, Nowotka M, Bento AP, Chambers J, Mendez D, Mutowo P, Atkinson F, Bellis LJ, Cibrián-Uhalte E, Davies M, Dedman N, Karlsson A, Magariños MP, Overington JP, Papadatos G, Smit I, Leach AR. The ChEMBL database in 2017. Nucleic Acids Res. 2017 45(D1):D945-D954. DOI: 10.1093/nar/gkw1074
2. Stephen F. Altschul, Thomas L. Madden, Alejandro A.Schaffer, Jinghui Zhang, Zheng Zhang, Webb Miller, and David J.Lipman (1997), "Gapped BLAST and PSI-BLAST: a new generation of protein database search programs", Nucleic Acids Res. 25:3389-3402.

nhmmer: DNA homology search with profile HMMs

Travis J. Wheeler, Sean R. Eddy

Bioinformatics, Volume 29, Issue 19, 1 October 2013, Pages 2487-2489, <https://doi.org/10.1093/bioinformatics/btt403>

# 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

1. Neglected tropical diseases, <https://www.who.int/neglected_diseases/diseases/en/> [↑](#footnote-ref-1)
2. Gaulton A, Hersey A, Nowotka M, Bento AP, Chambers J, Mendez D, Mutowo P, Atkinson F, Bellis LJ, Cibrián-Uhalte E, Davies M, Dedman N, Karlsson A, Magariños MP, Overington JP, Papadatos G, Smit I, Leach AR. The ChEMBL database in 2017. Nucleic Acids Res. 2017 45(D1):D945-D954. DOI: 10.1093/nar/gkw1074 [↑](#footnote-ref-2)
3. Stephen F. Altschul, Thomas L. Madden, Alejandro A.Schaffer, Jinghui Zhang, Zheng Zhang, Webb Miller, and David J.Lipman (1997), "Gapped BLAST and PSI-BLAST: a new generation of protein database search programs", Nucleic Acids Res. 25:3389-3402. [↑](#footnote-ref-3)
4. nhmmer: DNA homology search with profile HMMs

   Travis J. Wheeler, Sean R. Eddy

   Bioinformatics, Volume 29, Issue 19, 1 October 2013, Pages 2487-2489, <https://doi.org/10.1093/bioinformatics/btt403> [↑](#footnote-ref-4)
5. World malaria report 2019

   WHO

   <https://www.who.int/news-room/feature-stories/detail/world-malaria-report-2019> [↑](#footnote-ref-5)
6. Emerging drug—resistance and guidelines for treatment of malaria

   [Khan MA](https://www.ncbi.nlm.nih.gov/pubmed/?term=Khan%20MA%5BAuthor%5D&cauthor=true&cauthor_uid=15225468)1, [Smego RA Jr](https://www.ncbi.nlm.nih.gov/pubmed/?term=Smego%20RA%20Jr%5BAuthor%5D&cauthor=true&cauthor_uid=15225468), [Razi ST](https://www.ncbi.nlm.nih.gov/pubmed/?term=Razi%20ST%5BAuthor%5D&cauthor=true&cauthor_uid=15225468), [Beg MA](https://www.ncbi.nlm.nih.gov/pubmed/?term=Beg%20MA%5BAuthor%5D&cauthor=true&cauthor_uid=15225468).

   [J Coll Physicians Surg Pak.](https://www.ncbi.nlm.nih.gov/pubmed/15225468) 2004 May;14(5):319-24. [↑](#footnote-ref-6)
7. Responding to antimalarial drug resistance

   <https://www.who.int/malaria/areas/drug_resistance/overview/en/> [↑](#footnote-ref-7)