Discovering new targets and drugs for neglected diseases by paralog matching.

Author: Jeremy B. Singer

Affiliations:

Jeremy B. Singer is a private scholar studying for a masters in bioinformatics at Brandeis University. He works as a Senior I.T. developer at Clarivate Analytics (formerly Thomson Reuters Life Sciences), and can be reached at [jbsing@brandeis.edu](mailto:jbsing@brandeis.edu)

# Abstract

This paper describes a method that can be used to discover and repurpose existing drugs and drug targets by discovering cross species similarities. It uses public domain databases (MyCheEMBL , EnSEMBL, NCBI) and open source software to find measures of sequence conservation 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 they have genomes of sufficient size and because many have genomic features in common with organisms for which there are known targets.

Repurposing existing drugs can save much time and cost, as previously identified drugs have well understood safety and dosage profiles. Many of these drugs can assume new roles after losing effectiveness against their original 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.

MyChEMBL is a down loadable self-installing platform that contains a PostgreSQL collection of thousands of protein targets, and includes the sequences of their associated proteins. Using relational database and open source software, the methods described in this paper identified hundreds of promising cross species protein targets which were close paralogs to those found in *p. falciparum*. We identified fifty ligands, many of which are known anti-malarial compounds, validating this approach.

# Introduction

The *World Health Organization* has said that “In 2015, there were an estimated 214 million new malaria cases worldwide, causing 438,000 deaths.” *Plasmodium falciparum*, an obligate apicomplexan parasite, causes the most acute form of this disease. Mutations decreasing the effectiveness of existing medicines increase the need to discover new ones. The author has chosen *p. falciparum* as a model to validate a method to repurpose existing targets and drugs.

Protein targets are those proteins that provide vital functions to the organisms to which they belong. Because they provide vital function, we expect that their amino acid sequences would be conserved. This description of the nature of protein targets suggests that paralogous proteins in our organism of interest could also be targets.

MyCheMBL, a downloadable virtualized platform contains a database of targets and target sequences in a PostgreSQL database that also contains molecule and drug information. With a suitable similarity metric and discriminant criteria, we can distinguish new targets for our organism of interest, as well as drugs suitable to treat disease related to those targets. By itself, MyCheMBL does not contain everything we need to discriminate these targets.

*BLASTP,* a protein sequence search and alignment tool, finds matches for query sequences while scoring matches for conservation . Querying a *BLASTP* database with all the protein sequences from an organism produces reports from which we can extract similarity scores. Creating a new table, *blast\_statistics* , enables us to analyze the target similarity data using *score* as a measure of conservation.

# Materials and Methods

## The computation platform

All work was performed on a consumer grade laptop PC, with 12 GB of ram and 1 terrabyte of disk storage, with an Intel® Core™ i7-5500U CPU running at 2.4 ghz clock. This CPU has 4 cores, which can improve performance when multiple virtual machines are running. The ram configuration leaves plenty of RAM for multiple VMs (Virtual machines) to run simultaneously. This laptop runs Windows 10. (See Figure 1:Hardware Configuration.)

Most bioinformatics software runs under Linux. Using Oracle’s *VirtualBox* software, multiple Linux virtual machines can be supported on a PC.

## Setting up a Linux Virtual Machine using Oracle’s VirtualBox software

Download the latest VirtualBox software from Oracle (available at <https://www.virtualbox.org/> .)

Windows update can cause this software to stop working, which requires downloading newer software.

A CentOS 6.5 x86\_64 vdi image was downloaded from <https://virtualboximages.com> . This virtual device image is compatible with *Guest Additions* that makes mouse integration between Windows and a running Linux virtual machine easier.

A VM was created with 2gb of ram, using the downloaded VDI. This VM has its own IP address, which can be reached by *ssh* software installed on the PC. (See Figure 2: VM configuration parameters.)

A user account was created with *sudoer* privileges. This account was used for all other setup activities. The *sudo* command can be used to assume *root* privileges temporarily to manage resources on the VM for a command’s duration, and then returning to lower user privilege for safety. Typically, applications run with lower privilege using only resources that are allowed for them. (See Linux user setup.)

## Installations

The following tools were used for the analytical pipeline:

### Install tools on work Linux VM.

### Glimmer3.02 – identify genes.

### EMBOSS Tools (version 6.5.7) – to translate genes to peptide ORFs.

### BLAST Tools - ncbi-blast-2.3.0 – to score similarity of peptides to targets.

### PostgreSQL server and client – create a blast\_statistics table on the work server first.

### Install MyChEMBL 20 vagrant VM. (See Installing MyChEMBL VM.)

### Install ssh client on PC to communicate with work VM.

### Install PostgreSQL client on PC to communicate with MyChEMBL PostgreSQL database.

## Analytical Workflow

(See Figure 4: Genome target analysis workflow)

The target blast database was created on the work VM using target sequences extracted from the *chembl\_20* PostgreSQL database provided by the *myChEMBL* VM.

The genome of *plasmodium falciparum* was downloaded to the work VM.

The 14 chromosomes of *p. falciparum* were downloaded from this URL: <ftp://ftp.ensemblgenomes.org/pub/protists/release-32/fasta/plasmodium_falciparum/dna/>

Mitochondrial dna sequence was downloaded from here: GenBank: AJ276844.1 <https://www.ncbi.nlm.nih.gov/nuccore/8346980?report=fasta>

Apicoplast sequence was downloaded from here: Plasmodium falciparum genome assembly, organelle: plastid:apicoplast

NCBI Reference Sequence: NC\_030754.1

<https://www.ncbi.nlm.nih.gov/nuccore/1052489052?report=fasta>

*Glimmer3.02* identified putative genes. Some installation tweaks were required to make the build work. Training sequences from p. falciparum were provided to glimmer, and the full genome was processed into files named according to the scheme *<chromosome>.gene*, with each *.gene* file containing predicted genes locations. Mitocondrial genes were named *mt.gene* and apicoplast genes were named *apicoplast.gene*.

ORF files (Open Reading Frame) were created using *transeq (*from EMBOSS-6.5.7), creating 12,480 files under directories named by chromosome or organelle. These files are peptide *.FASTA* formatted files.

### BLASTP searches, aligns, and scores each ORF against the BLASTP database.

A bash script runs the **BLASTP** command for each ORF in all the chromosome/organelle directories, creating an alignment report for each ORF named *<ORF name>.blastp.txt*. (See do\_blastp.sh script to produce alignment and scores for ORFs in chromosome/organelle directories.) This query produces the top 10 target sequence alignments for each ORF.

### An Extract Translate and Load (ETL) process populates the BLAST\_STATISTICS table in a PostgreSQL database on the work VM.

For each ORF, a file named like *<ORF>.blastp.stats* is createdby ­­blast**\_stats**.sh, which runs Perl scriptextract\_header.pl that extracts the blast statistics from the BLASTP reports.

The create\_populate\_blast\_statistics.sh bash script assembles the statistics into the necessary plsql command file **populate\_blast\_statistics.**sql that will be used to load the **blast\_statistics** table, once the table is created.

Using the *psql* client, the **blast\_statistics.sql** table creates the *blast\_statistics* and *tmp\_blast\_statistics* tables. The *tmp\_blast\_statistics* table is a temporary data table that is used like a bucket to do inserts of one ORF at a time into the *blast\_statistics* table. After each insert , the tmp\_blast\_statistics table is truncated.

Finally the **populate\_blast\_statistics.sql**file is run, populating the *blast\_statistics* table.

Once the *blast\_statistics* table is populated in the database on the work VM, it can be exported and re-imported into a similarly created table on the myChEMBL VM.

After *blast\_statistics* are populated in the *chembl\_20* databases, we can analyze the data more easily using relational database tools, and R.

# Results and Discussion

## Preparing peptide sequences.

To avoid missing any relevant genes, we searched for peptide sequences ourselves to find as many putative peptides as we could by using **glimmer3.02**, which wrote out separate files by chromosome or organelle (mitochondrion or apicoplast) named according to the scheme *chromosome/organelle>.genes*. Each line in these files corresponds to one peptide; we identified 12,480 peptides in this way (See Gene count.)

Using EMBL tools, each gene was extracted and translated into an ORF file (Open Reading Frame) file to a directory corresponding to the *genes* file.

These processes took about an hour.

## BLASTP scoring

Applying BLASTP to each ORF created a BLAST report file containing up to 10 alignments to targets and including their scores. The reports were named *<ORF>.blastp.txt*. This process took about an hour. (See Figure 5:Understanding BLAST statistics.)

Each blast report was parsed into a file named *<ORF>.blastp.stats*. Parsing took less than a minute.

The ETL process took about 15 minutes, and created 55,848 *blast\_statistics* records in the PostgreSQL database on the work VM. These records were exported and re-imported into the *chembl\_20* database for combination with target and drug information for analysis.

## Testing the hypothesis: Sequences with more than normally expected similarity to targets are likely to be targets themselves.

The malaria organism, *p. falciparum,* has two kinds of organelles that have genes more closely related to bacteria than to other protists (see Figure 6: Plasmodium structural diagram showing organelles.) The proteins translated from these genes are known malaria targets:

“Several antibiotics, including clindamycin, chloramphenicol and the macrolides erythromycin and azithromycin, bind in the vicinity of the ribosome LSU peptidyl transferase centre or the peptide exit tunnel and inhibit parasite growth. This group also includes thiostrepton that contacts ribosomal protein L11 and the GTPase region of 23S rRNA. Translation inhibitory antibiotics have two putative target organelles, the apicoplast and mitochondrion, of the parasite. Some antibiotics (e. g. clindamycin, azithromycin, chloramphenicol and tetracycline) have been demonstrated to have a delayed-death effect, a phenotype associated with apicoplast-specific action... Thiostrepton causes immediate parasite killing and is proposed to have additional targets in P. falciparum.” – Ankit Gupta, et al.

We attempted to separate the population of cross-species targets from the general population of proteins in *p. falciparum* using the signature of normal distribution of data to separate these two groups.

The following concepts support this approach:

1. Protein targets have a functional role that is vital to the pathogenic organism. Drugs interfere with these targets.
2. Sequences that are functionally important are conserved, as changes to these sequences interfere with their function, preventing their propagation.
3. Over time, gene sequences mutate randomly. Sequences that are highly similar are either closely related, or functionally similar.

These concepts lead us to expect that there should be two separate populations of proteins having different distributions of scores: a lower scoring set whose distribution would be explained by random mutation, and a higher scoring set consistent with functional conservation.

A distribution showing normality would support the null hypothesis of similarity scores resulting from the random process of mutation.

### Data analysis using R.

Using R, the *blast\_statistics* table was imported from the *chembl\_20* database where we had created it (Loading *blast\_statistics* dataframe.) using the ETL procedure described (Figure 4: Genome target analysis workflow.) Histogram shows that the *score* data are log distributed (Figure 7:Histogram showing distribution of scores.)

R’s **qqnorm** plot describes visually how well a distribution approaches normality. The graph produced by the **qqnorm** plot combines with a red line drawn by **qqline** delineating expected normality for comparison.

The complete dataset of all malaria similarity scores departs markedly from normality in the upper range of scores (Figure 8: "All malaria scores" qqnorm plot.)

We can construct an upper bound on scores that discriminates the normally distributed scores from those that are not.

BLASTP is configured to omit low scores in the reports, which means that *mean* and *standard deviation* are not the best measures of centrality and dispersion. *Median* and *Maximum Absolute Deviation (MAD)* provide better metrics of the behavior of our dataset. We chose a *discriminant* *threshold* as *median(score) + 2 \* mad(score)*. Below this threshold score, we expect the data to be distributed normally. The value of this threshold score for *p. falciparum*, rounded to the nearest unit, is 37.

Plotting the dataset obtained be excluding values greater than this threshold displays normal behavior, as expected (Figure 9: p.falciparum normal scores.)

The remainder of statistics greater than the significance threshold belong to sequences that are more than normally similar to targets sequences.

### Selecting promising sequences as targets using PostgreSQL in the augmented *chembl\_20* database.

Besides the similarity criteria, we also need exclusion criteria for targets so that we do not choose targets that are too similar to human or model organisms that we wish to protect. We also wish to exclude targets that are already identified as malaria targets so that we will show only novel results.

We created the *exclude\_organisms* table to designate those organisms whose targets we wish to avoid (Table 1: Exclude\_organisms.)

To demonstrate the usefulness of the selection criteria, mapped targets meeting our criteria to known drugs (Figure 10: From statistics to drugs.) Results show 54 hits ( Table 2: Drugs and targets showing organism and mechanism.) Some drugs were duplicated; for example, some drugs were reported for more than one organism, and some drugs were similar but slightly different formulations.

The drugs clindamycin, chloramphenicol and tetracycline mentioned by Ankit Gupta, et al were identified. Neomycin, which was identified, is the same as thiostrepton, was also identified. Other antibiotics not mentioned in their paper were also identified.

The Query for new targets returns 592 records, which are available for further study. Many of these targets match to more than one genome location.

# Conclusions

* This paper validates a method for discovering cross species targets by identifying 29 distinct drugs for malaria (53, counting different formulations).
* We describe a method to compute the **target selection threshold criterion** for a given genome. For *p. falciparum*, this threshold score is 37.
* Further development could create a platform to find paralogous targets for any target organism that has a genome of sufficient size.
* A database containing paralogs could be automatically created for many genomes, including those for neglected diseases.
* Private target sequences could be used to enrich such a database.

# Acknowledgements

Many thanks to Nikolaus Obholzer and Sylvain Meylan for their encouragement and advice.

Thanks to my editor, Eric Fischbach.

And to my wife, Glenda, for her patience and support.

# References

1. Fact Sheet: World Malaria Report 2015, p. 1.

World Health Organization

9 December 2015

<http://www.who.int/malaria/media/world-malaria-report-2015/en/>

1. myChEMBL: A.P. Bento, A. Gaulton, A. Hersey, L.J. Bellis, J. Chambers, M. Davies, F.A. Krüger, Y. Light, L. Mak, S. McGlinchey, M. Nowotka, G. Papadatos, R. Santos & J.P. Overington (2014) 'The ChEMBL bioactivity database: an update' Nucl. Acids Res. Database Issue. 42 D1083-D1090 DOI:10.1093/nar/gkt1031 PMID:242149652.
2. M. Davies, M. Nowotka, G. Papadatos, F. Atkinson, G.J.P. van Westen, N Dedman, R. Ochoa and J.P. Overington (2014) 'myChEMBL: A Virtual Platform for Distributing Cheminformatics Tools and Open Data' Challenges 5 (334-337) [pdf](http://doi.org/10.1093/bioinformatics/btt666)
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.
4. Virtualbox Wiki Downloads: <https://www.virtualbox.org/wiki/Downloads>

The 14 chromosomes of *p. falciparum* were downloaded from this URL: <ftp://ftp.ensemblgenomes.org/pub/protists/release-32/fasta/plasmodium_falciparum/dna/>

Mitochondrial dna sequence was downloaded from here: GenBank: AJ276844.1 <https://www.ncbi.nlm.nih.gov/nuccore/8346980?report=fasta>

Apicoplast sequence was downloaded from here: Plasmodium falciparum genome assembly, organelle: plastid:apicoplast

NCBI Reference Sequence: NC\_030754.1

<https://www.ncbi.nlm.nih.gov/nuccore/1052489052?report=fasta>

1. Microbial gene identification using interpolated Markov models

S. Salzberg, A. Delcher, S. Kasif, and O. White

Nucl. Acids Res., 26(2):544–548, 1998

1. Ankit Gupta, et al. “Reduced ribosomes of the apicoplast and mitochondrion of Plasmodium spp. and predicted interactions with antibiotics”, Figure 1 and quotation.

<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4042851/figure/RSOB140045F1/>

© 2014 The Authors. Published by the Royal Society under the terms of the Creative Commons Attribution License<http://creativecommons.org/licenses/by/3.0/>, which permits unrestricted use, provided the original author and source are credited.

1. Plasmodium structure. <https://upload.wikimedia.org/wikipedia/commons/f/fa/Plasmodium.png>

# Supporting Information



Figure 1:Hardware Configuration.



Figure 2: VM configuration parameters.

## Installing MyChEMBL VM.

MyChEMBL is a virtual machine platform that is downloadable to a personal computer. It runs in a separate virtual machine running CENTOS (a variety of Redhat Linux) under Oracle’s Virtualbox software. Each MyChEMBL virtual machine contains its own PostgreSQL database and Tomcat webserver. We recommend using a PC with at least 8 gb of memory; we used 12 gb.

To install, we can download one of the releases:

[ftp://ftp.ebi.ac.uk/pub/databases/chembl/VM/myChEMBL/releases](ftp://ftp.ebi.ac.uk/pub/databases/chembl/VM/myChEMBL/releases/)

We are using release 20.

Install Oracle’s Virtualbox:

(See Setting up a Linux Virtual Machine using Oracle’s VirtualBox software.)

Install Vagrant:

<https://www.vagrantup.com/downloads.html>

From a command prompt, run :

**vagrant init chembl/myChEMBL && vagrant up**

wait a bit…

In a browser, go to this url: <http://127.0.0.1:8000/>

Which takes you to the MyChEMBL landing page hosted on your own P.C.

(see Figure 3:MyCHeMBL landing page. )



Figure 3:MyCHeMBL landing page.



Figure 4: Genome target analysis workflow

The analysis workflow uses genome information downloaded from NCBI.gov and from EMBL to a *work* virtual machine running on a PC. Target and molecule information resides in a PostgreSQL database named *chembl\_20* that resides in a different MyChEMBL virtual machine running on the same PC, which acts as the integration platform.

The *p.falciparum* genome is decompressed as it is downloaded to the *work* VM. It consists of files with filetype *.fa* containing FASTA formatted sequences of nucleotides for each chromosome. The mitochondrial and apicoplast genome files were downloaded from <https://www.ncbi.nlm.nih.gov/nuccore/8346980?report=fasta> for mitochondrial genome, <https://www.ncbi.nlm.nih.gov/nuccore/1052489052?report=fasta>

for apicoplast genome, and chromosomal genome files were downloaded from <ftp://ftp.ensemblgenomes.org/pub/protists/release-32/fasta/plasmodium_falciparum/dna/>*.*

The workflow continues as glimmer 3.02 identifies the coordinates of putative genes. Extract stitches together these together into gene files for each chromosome. A Perl script (See chrom\_genes\_to\_proteins.pl Perl Script) parses gene files, creating Open Reading Frame files for each peptide by invoking **transeq**. These ORF files are amino acid .FASTA files that encode polypeptides.

The split\_to\_fasta.pl Perl script converts the exported target sequences from the **chembl\_20** PostgreSQL database to a concatenated .FASTA file, which is converted to a blast database by the **makeblastdb** command.

BLASTP queries the *target blast database* using each ORF (Open Reading Frame) file creating a blast report file for each. The **extract\_blast\_stats.pl**Perl script extracts a blast stats file for each ORF file. The **create\_populate\_blast\_statistics.pl** Perl script creates the **populate\_blast\_statistics.sql** file which loads the blast stats files into the *blast\_statistics*table, for analytical use.



Figure 5: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.)

## Linux user setup.

1. Start VM
2. Login as **adminuser**, password **adminuser** # default for this distro
3. From *Applications*, run *System Tools -> Terminal*
4. **adduser <username>**
5. **passwd <username>**
6. **sudo visudo** Edits the visudo file in VI. Uncomment the line that mentions %%wheel.
7. **usermod –G wheel <username>** # this makes user <username> a sudoer.

## Gene count

These commands were performed on the work VM.

[~/genome/genes] wc -l \*.genes

30 apicoplast.genes

402 chromosome.1.genes

691 chromosome.2.genes

675 chromosome.3.genes

728 chromosome.4.genes

252 chromosome.5.genes

2010 chromosome.6.genes

786 chromosome.7.genes

258 chromosome.8.genes

1031 chromosome.9.genes

329 chromosome.10.genes

1349 chromosome.11.genes

1418 chromosome.12.genes

575 chromosome.13.genes

1940 chromosome.14.genes

6 mt.genes

12480 total

## chrom\_genes\_to\_proteins.pl Perl Script

#!/usr/bin/perl

# chrom\_genes\_to\_proteins.pl

# invoke: chrom\_genes\_to\_proteins.pl <input chrom nucleic acid fasta>

# Input filename looks like <chromosomename.genes>

# Program creates a directory that looks like <chromosomename> and

# creates a separate file named <orfname>.fasta in that directory.

if (@ARGV < 1) {die "Specify gene filename.\n"}

my $gene\_filename = shift @ARGV;

$gene\_filename =~ m/(\S+)\.genes/;

my $dest = $1;

mkdir $dest;

open(CHROMFILE,"$gene\_filename") or die ("Unable to open $gene\_filename.\n");

my @orfs=<CHROMFILE>;

close(CHROMFILE);

foreach my $orf(@orfs)

{

$orf =~ m/(\S+)\s+(\S+)/;

my $orfname = $1;

my $nucleotides = $2;

open(TEMPGENE,">tempgene.fa") or die ("Unable to create tempgene");

print TEMPGENE $nucleotides;

close(TEMPGENE);

system "transeq -sequence tempgene.fa -outseq $dest/$orfname.fasta";

}

## split\_to\_fasta.pl Perl script

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

# split\_to\_fasta.pl

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

# output : rec1 = ><key>

# rec2 = <sequence>

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

my $infile = 'component\_sequences.txt';

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

my $delim = '\|';

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);

## blast\_statistics.sql

create table blast\_statistics

(

tax\_id bigint,

chromosome character varying(50),

orf character varying(50),

targ\_comp int,

query\_length smallint,

score numeric,

expect numeric,

identities numeric,

positives numeric,

gaps numeric

);

create table tmp\_blast\_statistics

(

tax\_id bigint,

chromosome character varying(50),

orf character varying(50),

targ\_comp int,

query\_length smallint,

score numeric,

expect numeric,

identities numeric,

positives numeric,

gaps numeric

);



Figure 6: Plasmodium structural diagram showing organelles.

## Loading *blast\_statistics* dataframe.

library(RPostgreSQL)

drv=dbDriver('PostgreSQL')

con=dbConnect(drv,dbname='chembl\_20',port=5432,host='localhost',user='mychembl')

blast\_statistics=dbReadTable(con,'blast\_statistics')



Figure 7:Histogram showing distribution of scores.

These diagrams were generated by the following statements:

old.par=par(mfrow=c(2,2))

hist(score)

hist(log(score))

par(old.par)



Figure 8: "All malaria scores" qqnorm plot.

The red line shows expected normality. This plot shows departure from normality at the right (upper) portion of the graph.

These statements produce the graph:

qqnorm(malaria\_statistics$score,main='All Malaria Scores')

qqline(malaria\_statistics$score,col='red',lwd=3)



Figure 9: p.falciparum normal scores.

Plot was produced by these R commands:

significance\_threshold<-function(score){

median(score)+2\*mad(score)

}

malaria\_threshold=significance\_threshold(malaria\_statistics$score)

norm\_falciparum=malaria\_statistics[malaria\_statistics$score < malaria\_threshold,]

qqnorm(norm\_falciparum$score,main='p. falciparum normal scores')

qqline(norm\_falciparum$score,col='red',lwd=3)

|  |  |
| --- | --- |
| **tax\_id** | **organism** |
| 9483 | Callithrix jacchus |
| 9541 | Macaca fascicularis |
| 9541 | Macaaca fascicularis |
| 9544 | Macaca mulatta |
| 9606 | Homo sapiens |
| 9615 | Canis lupus famiaris |
| 9615 | Canis lupus familiaris |
| 9615 | Canis familiaris |
| 9823 | Sus scrofa |
| 9913 | Bos taurus |
| 9940 | Ovis aries |
| 9986 | Oryctolagus cuniculus |
| 10090 | Mus musculus |
| 10116 | Rattus norvegicus |

Table 1: Exclude\_organisms.

Entries for tax\_id 9541 and 9615 show multiple entries due to variations in the spelling of the organism name in the database. These organisms were excluded because of close relationship to humans or close relationship to an experimental model organism.



Figure 10: From statistics to drugs.

This query produces a list of targets that have drugs:

select distinct td.tax\_id, td.organism, md.pref\_name, md.chembl\_id, mechanism\_of\_action

FROM blast\_statistics bs

, target\_components tc

, target\_dictionary td

, drug\_mechanism dm

, molecule\_dictionary md

, binding\_sites bind

WHERE bs.score > 37

and bs.expect < .001

and bs.targ\_comp = tc.component\_id

and bs.tax\_id = 5833

and tc.tid = td.tid

and bind.tid = tc.tid

and dm.site\_id = bind.site\_id

and md.molregno = dm.molregno

and not exists (select 1

from exclude\_organisms eo

where eo.tax\_id = td.tax\_id

)

and upper(td.organism) not like '%PLASMODIUM%'

order by md.pref\_name;

## Query for new targets

select distinct td.chembl\_id,td.tax\_id,organism

FROM blast\_statistics bs

, target\_components tc

, target\_dictionary td

WHERE bs.score > 37

and bs.expect < .001

and bs.targ\_comp = tc.component\_id

and tc.tid = td.tid

and not exists (select 1

from exclude\_organisms eo

where eo.tax\_id = td.tax\_id

)

and upper(td.organism) not like '%PLASMODIUM%'

group by td.chembl\_id, td.tax\_id, organism

order by chembl\_id;



Table 2: Drugs and targets showing organism and mechanism.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **tax\_id** | **organism** | **pref\_name** | **chembl\_id** | **mechanism\_of\_action** |
| 6231 | Nematoda | ALBENDAZOLE | CHEMBL1483 | Tubulin inhibitor |
| 2 | Bacteria | AMIKACIN SULFATE | CHEMBL1164318 | Bacterial 70S ribosome inhibitor |
| 1773 | Mycobacterium tuberculosis | CAPREOMYCIN SULFATE | CHEMBL2218913 | 70S ribosome inhibitor |
| 2 | Bacteria | CHLORAMPHENICOL | CHEMBL130 | Bacterial 70S ribosome inhibitor |
| 2 | Bacteria | CHLORAMPHENICOL PALMITATE | CHEMBL1506 | Bacterial 70S ribosome inhibitor |
| 2 | Bacteria | CHLORAMPHENICOL SODIUM SUCCINATE | CHEMBL1200729 | Bacterial 70S ribosome inhibitor |
| 2 | Bacteria | CHLORTETRACYCLINE HYDROCHLORIDE | CHEMBL2146063 | Bacterial 70S ribosome inhibitor |
| 2 | Bacteria | CLINDAMYCIN HYDROCHLORIDE | CHEMBL1200588 | Bacterial 70S ribosome inhibitor |
| 2 | Bacteria | CLINDAMYCIN PALMITATE HYDROCHLORIDE | CHEMBL1237086 | Bacterial 70S ribosome inhibitor |
| 2 | Bacteria | CLINDAMYCIN PHOSPHATE | CHEMBL3184512 | Bacterial 70S ribosome inhibitor |
| 2 | Bacteria | DALFOPRISTIN | CHEMBL1200937 | Bacterial 70S ribosome inhibitor |
| 2 | Bacteria | DEMECLOCYCLINE HYDROCHLORIDE | CHEMBL1200474 | Bacterial 70S ribosome inhibitor |
| 2 | Bacteria | DIRITHROMYCIN | CHEMBL3039471 | Bacterial 70S ribosome inhibitor |
| 2 | Bacteria | ERYTHROMYCIN | CHEMBL532 | Bacterial 70S ribosome inhibitor |
| 2 | Bacteria | ERYTHROMYCIN ESTOLATE | CHEMBL2218877 | Bacterial 70S ribosome inhibitor |
| 2 | Bacteria | ERYTHROMYCIN ETHYLSUCCINATE | CHEMBL1200688 | Bacterial 70S ribosome inhibitor |
| 2 | Bacteria | ERYTHROMYCIN GLUCEPTATE | CHEMBL1200657 | Bacterial 70S ribosome inhibitor |
| 2 | Bacteria | ERYTHROMYCIN LACTOBIONATE | CHEMBL1200506 | Bacterial 70S ribosome inhibitor |
| 2 | Bacteria | ERYTHROMYCIN STEARATE | CHEMBL1200510 | Bacterial 70S ribosome inhibitor |
| 2 | Bacteria | GENTAMICIN SULFATE | CHEMBL3039594 | Bacterial 70S ribosome inhibitor |
| 5550 | Trichophyton | GRISEOFULVIN | CHEMBL562 | Tubulin inhibitor |
| 2 | Bacteria | KANAMYCIN SULFATE | CHEMBL1446 | Bacterial 70S ribosome inhibitor |
| 1313 | Streptococcus pneumoniae | LINCOMYCIN HYDROCHLORIDE | CHEMBL1201097 | 70S ribosome inhibitor |
| 1313 | Streptococcus pneumoniae | LINEZOLID | CHEMBL126 | 70S ribosome inhibitor |
| 6231 | Nematoda | MEBENDAZOLE | CHEMBL685 | Tubulin inhibitor |
| 2 | Bacteria | MECLOCYCLINE SULFOSALICYLATE | CHEMBL261772 | Bacterial 70S ribosome inhibitor |
| 2 | Bacteria | METHACYCLINE HYDROCHLORIDE | CHEMBL2146123 | Bacterial 70S ribosome inhibitor |
| 2 | Bacteria | MINOCYCLINE HYDROCHLORIDE | CHEMBL1200881 | Bacterial 70S ribosome inhibitor |
| 2 | Bacteria | NEOMYCIN SULFATE | CHEMBL1275977 | Bacterial 70S ribosome inhibitor |
| 2 | Bacteria | NETILMICIN SULFATE | CHEMBL1200872 | Bacterial 70S ribosome inhibitor |
| 2 | Bacteria | OXYTETRACYCLINE | CHEMBL1517 | Bacterial 70S ribosome inhibitor |
| 2 | Bacteria | OXYTETRACYCLINE CALCIUM | CHEMBL2068727 | Bacterial 70S ribosome inhibitor |
| 2 | Bacteria | OXYTETRACYCLINE HYDROCHLORIDE | CHEMBL1607480 | Bacterial 70S ribosome inhibitor |
| 2 | Bacteria | PAROMOMYCIN SULFATE | CHEMBL2206196 | Bacterial 70S ribosome inhibitor |
| 1773 | Mycobacterium tuberculosis | PYRAZINAMIDE | CHEMBL614 | 70S ribosome inhibitor |
| 2 | Bacteria | QUINUPRISTIN | CHEMBL1200649 | Bacterial 70S ribosome inhibitor |
| 1280 | Staphylococcus aureus | RETAPAMULIN | CHEMBL409542 | 70S ribosome inhibitor |
| 1314 | Streptococcus pyogenes | RETAPAMULIN | CHEMBL409542 | 70S ribosome inhibitor |
| 2 | Bacteria | RIFABUTIN | CHEMBL444633 | Bacterial DNA-directed RNA polymerase inhibitor |
| 2 | Bacteria | RIFAMPICIN | CHEMBL374478 | Bacterial DNA-directed RNA polymerase inhibitor |
| 2 | Bacteria | RIFAPENTINE | CHEMBL1660 | Bacterial DNA-directed RNA polymerase inhibitor |
| 2 | Bacteria | RIFAXIMIN | CHEMBL1617 | Bacterial DNA-directed RNA polymerase inhibitor |
| 2 | Bacteria | SPECTINOMYCIN HYDROCHLORIDE | CHEMBL1200407 | Bacterial 70S ribosome inhibitor |
| 2 | Bacteria | STREPTOMYCIN SULFATE | CHEMBL453087 | Bacterial 70S ribosome inhibitor |
| 2 | Bacteria | TELITHROMYCIN | CHEMBL1136 | Bacterial 70S ribosome inhibitor |
| 2 | Bacteria | TETRACYCLINE | CHEMBL1440 | Bacterial 70S ribosome inhibitor |
| 2 | Bacteria | TETRACYCLINE HYDROCHLORIDE | CHEMBL454950 | Bacterial 70S ribosome inhibitor |
| 2 | Bacteria | TETRACYCLINE PHOSPHATE COMPLEX | CHEMBL1201071 | Bacterial 70S ribosome inhibitor |
| 6231 | Nematoda | THIABENDAZOLE | CHEMBL625 | Tubulin inhibitor |
| 2 | Bacteria | TIGECYCLINE | CHEMBL376140 | Bacterial 70S ribosome inhibitor |
| 287 | Pseudomonas aeruginosa | TOBRAMYCIN | CHEMBL1747 | 70S ribosome inhibitor |
| 287 | Pseudomonas aeruginosa | TOBRAMYCIN SULFATE | CHEMBL1200780 | 70S ribosome inhibitor |
| 1313 | Streptococcus pneumoniae | TROLEANDOMYCIN | CHEMBL564085 | 70S ribosome inhibitor |
| 1773 | Mycobacterium tuberculosis | VIOMYCIN SULFATE | CHEMBL1200526 | 70S ribosome inhibitor |