Finding Paralog Targets for Neglected Diseases

By

Jeremy Singer

1. 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[1, 2] [3].

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 [2]. Using this database and open source software, this paper identified *726* distinct approved drugs with their associated targets validating this approach.

6 other pathogens (*Plasmodium vivax, Cryptosporidium parvum*, *Trypanosoma brucei, Trypanosoma cruzi, Leishmania donovani, SARS-CoV2*) were also downloaded and run through the same pipeline, identifying potential targets and drugs.

2. 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 [4].

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]. 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** or **jackhmmer** to produce similarity reports, parse the results, and upload to supplementary tables in the PostgreSQL database [5, 6]. This analysis pipeline was first applied to the genome of *P. 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 tropical diseases. In 2018, there were over 228 million cases of malaria worldwide, causing over 408 thousand deaths [5]. Emerging drug resistance to existing drugs such as chloroquine and sulfadoxine-pyrimethamine increases demand for new drugs that are more effective or have fewer adverse effects [7, 8].

In addition to *P. falciparum*, we processed the following additional pathogens: Plasmodium vivax, *Cryptosporidium parvum, Trypanosoma brucei, Trypanosoma cruzi, Leishmania donovan*i. Using **jackhmmer** to measure similarity statistics to targets in the CHEMBL_25 database we loaded these statistics into supplementary tables in the CHEMBL_25 database that we created for that purpose.

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 based on similarity thresholds computed using kmeans clustering. We were able to find potential targets and drugs for each of these organisms and produce reports listing studies and approvals for drugs, where available.

3. Materials and Methods

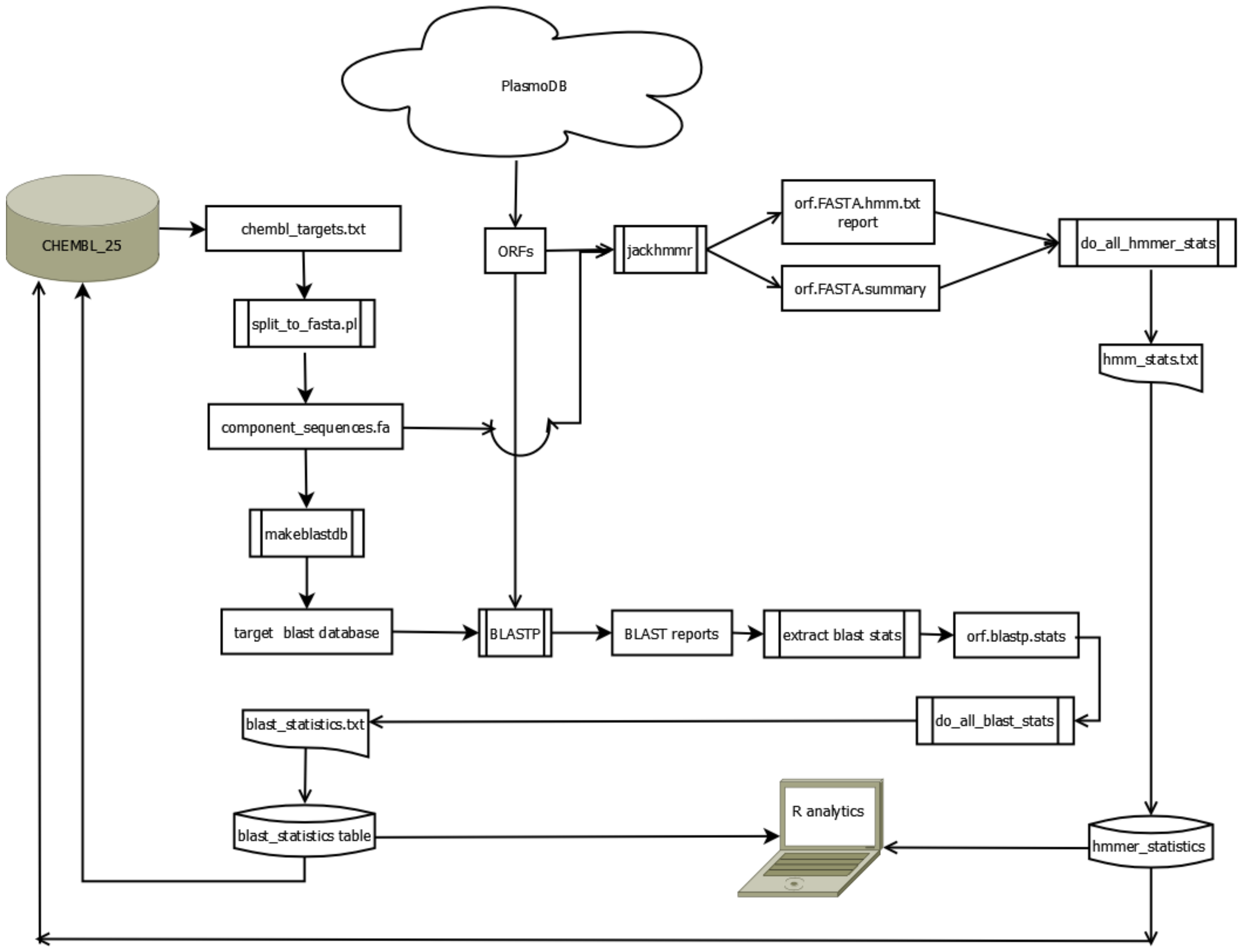


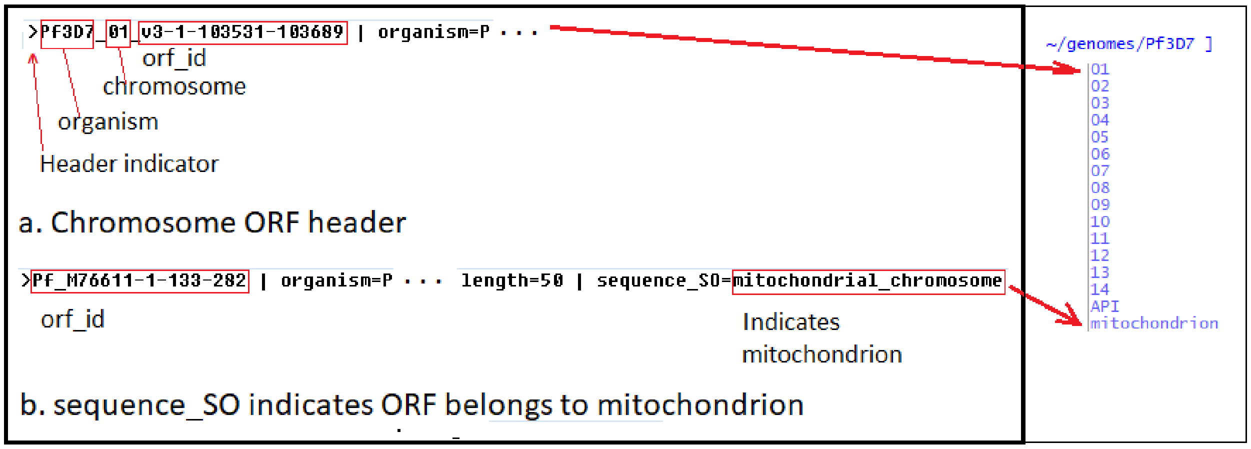
Figure 1: Processing overview

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` [3, 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 `).`



```
>Pf3D7_01_v3-1-103531-103689 | organism=P . . .
```

```
>Pf_M76611-1-133-282 | organism=P . . .  length=50 | sequence_SO=mitochondrial_chromosome
```

```
~/genomes/Pf3D7 ]
 01
 02
 03
 04
 05
 06
 07
 08
 09
 10
 11
 12
 13
 14
 API
 mitochondrion
```

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**.

3.1. Gathering BLASTP statistics

The **makeblastdb** utility converted the FASTA formatted file of targets to a **BLASTP** searchable database. The command was run in directory **`~/blast_targets`** :

```
makeblastdb -in component_sequences.fa -out chembl_25_targets
```

Custom bash scripts were used for further processing, including a script that applied 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). The script calls blast with these parameters:

```
blastp -db ~/blast_targets/chembl_25_targets -query $orf -num_alignments 10 -out
```

The $orf parameter is replaced in turn by each ORF in the genome to query the target database. The `$[6].blastp.txt` parameter specifies where the output of each query will go. num_alignments parameter specifies the maximum number of alignments that will be performed for each query. The rest of the parameters are defaulted for the program (Protein-Protein BLAST 2.10.0+).

Bash script (7.2.5. do_all_blast_stats.sh) applied Perl script (7.2.3. extract_header.pl) to parse each BLAST report into a *.stats* file for each ORF.

Bash script (7.2.4. make_blast_statistics.sh) consolidated all the <ORF>*.blastp.txt.stats* files into a tab delimited text file, **blast_statistics.txt**.

The **blast_statistics** table, which was created previously, (7.2.6. create_blast_statistics_tbl.sql), was populated by SQL script (7.2.7. import_p_falciparum.sql).

3.2. Gathering HMMER statistics

Script **do_all_jackhmmer.sh** applied the **jackhammer** utility to each of the ORFs of the *P. falciparum* genome, generating a report and a summary file for each (see **Error! Reference source not found.**).

A Bash script (7.3.3. do_all_hmmer_stats.sh) applied a Perl script (7.3.2. extract_hmm_summary.pl) that extracted the **jackhmmer** statistics from the reports and summaries and produced a consolidated tab delimited file for all ORFs called **hmm_stats.txt**.

Import SQL script (7.3.6. import_hmmer_statistics.sql) imported these statistics into previously created tables (see 7.3.5. `create_hmmer_stats_tbls.sql`).

An update statement set the tax_id and organism fields in the **hmmer_statistics** table to the appropriate values:

```
UPDATE hmmer_statistics set tax_id = 36329, organism = ‘Plasmodium falciparum 3D7’ where tax_id
is null;
```

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

```
\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 ~\Documents\RBIF120\consolidated_stats.txt CSV delimiter ' ‘
```

After the similarity statistics were uploaded through these ETL processes, they were analyzed using R graphical and statistical tools included here in the Results and Discussion section, and queries using criteria we will discuss, to produce spreadsheet reports of putative targets and drugs.

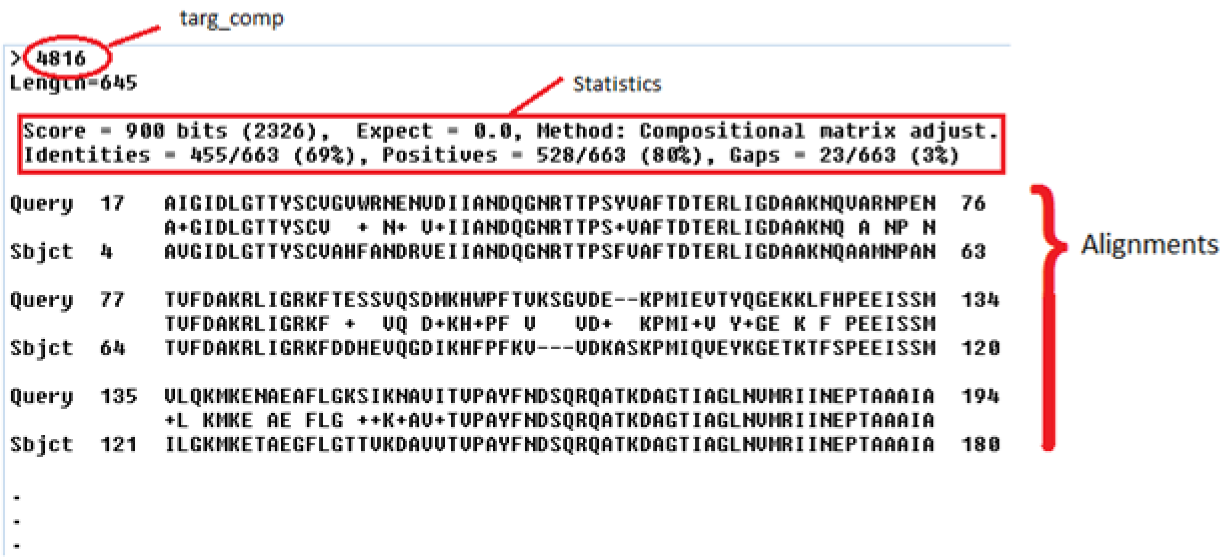
4. Discussion and Results

In the context of parasitic disease organisms, the “targetness” of a protein has to do with how indispensable its function is to the organism in question, since we are trying to kill the organism, or impair its success [10]. We are interested in protein targets that are highly conserved, because this indicates that the protein, in its conserved form, is necessary for the success of the disease organism [11]. We do not know specifically whether it is necessary for its infectious ability, its metabolic role, ability to transcribe DNA, translate proteins, or participate in the structure or outer integument of the organism. In addition, it will not be known whether the binding properties of the protein to any particular ligand has been preserved, even if the target is still useful as a target.

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 [11]. At the same time, we are searching exactly for those critically necessary proteins as targets for drugs that can impair them. Apicomplexan parasites such as the *Plasmodium* species preserve ribosomal targets in the apicoplast and mitochondrial genome that have conserved similarity due to their presumed origin in previous endosymbiotic events which give rise to their eukaryotic ancestors [12].

Uncertainty about whether existing drugs will effectively bind or interfere with the target proteins we identify is somewhat compensated for by the improvements of convenience and cost due to availability of the existing drugs, understanding of their dosage, and safety from existing studies [13, 14]. 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 choose threshold criteria for filtering the most promising candidates.

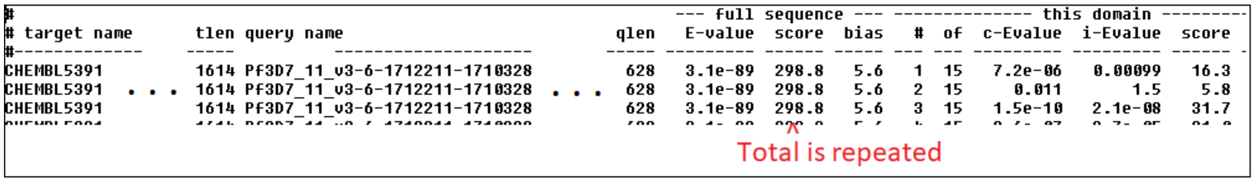
Both **BLASTP** and **jackhmmer** score similarity between amino acid sequences by aligning query and target sequences [5][6]. Their approach to scoring, however, differs. **BLASTP** detects similarities by scoring the likelihood of successive letters of the amino acid being the same in query and target sequences in comparison to chance [5]. By contrast, **jackhmmer** uses hidden Markoff models (HMM) that assess patterns by looking for larger domains [6]. We might therefore expect that this approach might be more discriminating, as it takes into account similarities at the level of protein domains. We examined the differences in results between **BLASTP** and **jackhmmer** to assess which approach would measure similarity in a way that would best identify the conservation we are looking for. We hypothesize that these similarity scores can be stratified into a group that is much more similar due to necessity to maintain conserved function.



```
>4816
Length=645

 Score = 900 bits (2326),  Expect = 0.0, Method: Compositional matrix adjust.
 Identities = 455/663 (69%), Positives = 528/663 (80%), Gaps = 23/663 (3%)

Query  17   AIGIDLGTTYSCVGVWRNENVDIIANDQGNRTTPSYVAFTDTERLIGDAAKNQVARNPEN  76
            A+GIDLGTTYSCV   + N+ V+IIANDQGNRTTPS+VAFTDTERLIGDAAKNQ A NP N
Sbjct  4    AVGIDLGTTYSCVAHFANDRVEIIANDQGNRTTPSFVAFTDTERLIGDAAKNQAAMNPAN  63

Query  77   TVFDAKRLIGRKFTESSVQSDMKHWPFTVKSGVDE--KPMIEVTYQGEKKLFHPEEISSM  134
            TVFDAKRLIGRKF +  VQ D+KH+PF V   VD+  KPMI+V Y+GE K F PEEISSM
Sbjct  64   TVFDAKRLIGRKFDDHEVQGDIKHFPFKV---VDKASKPMIQVEYKGETKTFSPEEISSM  120

Query  135  VLQKMKENAEAFLGKSIKNAVITVPAYFNDSQRQATKDAGTIAGLNVMRIINEPTAAAIA  194
            +L KMKE AE FLG ++K+AV+TVPAYFNDSQRQATKDAGTIAGLNVMRIINEPTAAAIA
Sbjct  121  ILGKMKETAEGFLGTTVKDAVVTVPAYFNDSQRQATKDAGTIAGLNVMRIINEPTAAAIA  180

.
.
.
```

Figure 3: Understanding BLAST statistics

BLASTP produces alignment reports containing the similarity statistics we are looking for (See 7.2.2. do_all_blast.sh). We extracted these in an ETL process and imported them into the **CHEMBL_25** database. (See 7.2.3. extract_header.pl, 7.2.4. make_blast_statistics.sh)



```
#                                                                --- full sequence --- -------------- this domain ---------
# target name        tlen query name                       qlen   E-value  score  bias   #  of  c-Evalue  i-Evalue  score
#-------------      -----      --------------------        ----- --------- ------ ----- --- --- --------- --------- ------
CHEMBL5391           1614 Pf3D7_11_v3-6-1712211-1710328     628   3.1e-89  298.8   5.6   1  15   7.2e-06   0.00099   16.3
CHEMBL5391   . . .   1614 Pf3D7_11_v3-6-1712211-1710328 . . . 628   3.1e-89  298.8   5.6   2  15     0.011       1.5    5.8
CHEMBL5391           1614 Pf3D7_11_v3-6-1712211-1710328     628   3.1e-89  298.8   5.6   3  15   1.5e-10   2.1e-08   31.7
```

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. For this study, we are only using similarity across the whole protein as a measure of conservation. 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.

4.1. 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.

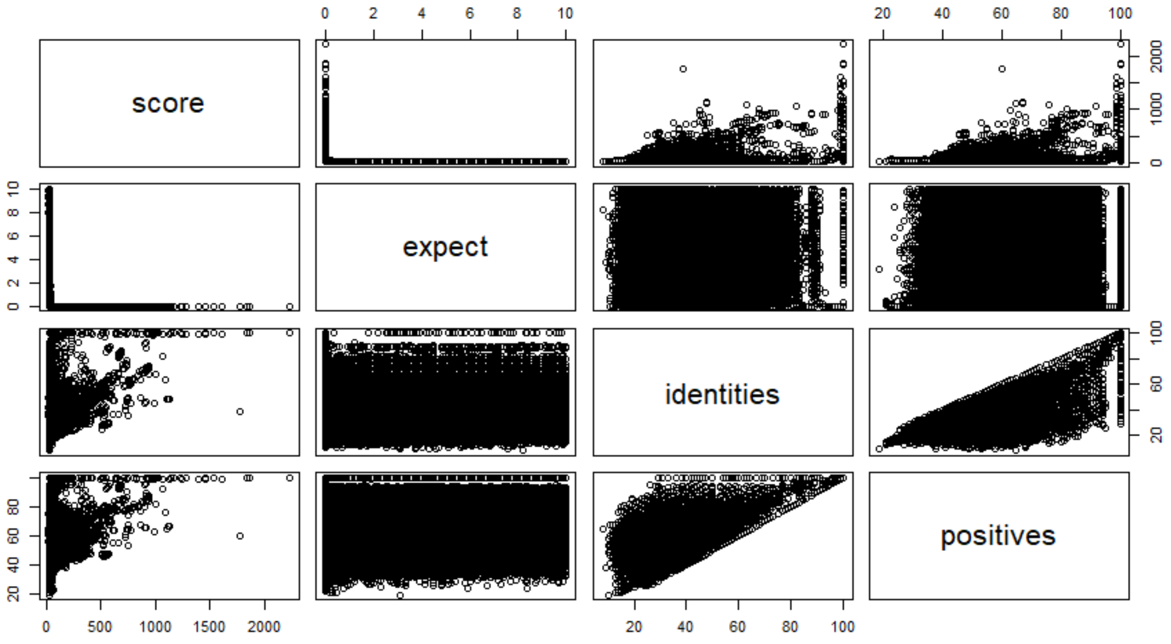


Figure 5: Pairwise comparison of BLAST metrics

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 [15, 16]. Although *expect* should follow (inversely) to score, the comparison here is uninformative because of loss of resolution at low values, which we see here causing overflow or underflow that shows no correlation at all (See figure above.). *Positives* trend like *identities* but show *greater than or equal* relationship to them. *Score* also trends with *positives,* showing greater detail.

Differences between the comparison of *scores* by *identities* and *expect* by *identities* varies substantially due to the imprecision caused by computations using *expect* [15, 16].

4.2. Comparison of BLASTP and jackhmmer scores

The default inclusion threshold for **jackhmmer** is much more stringent than the default threshold for **BLASTP**. BLASTP defaults to an expect value of 10.0, and initial word size match between query and target of 3. Up to 10 hits for each input are included in results [17, 18]. 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 7.4.1. consolidated_orf_target.sql).

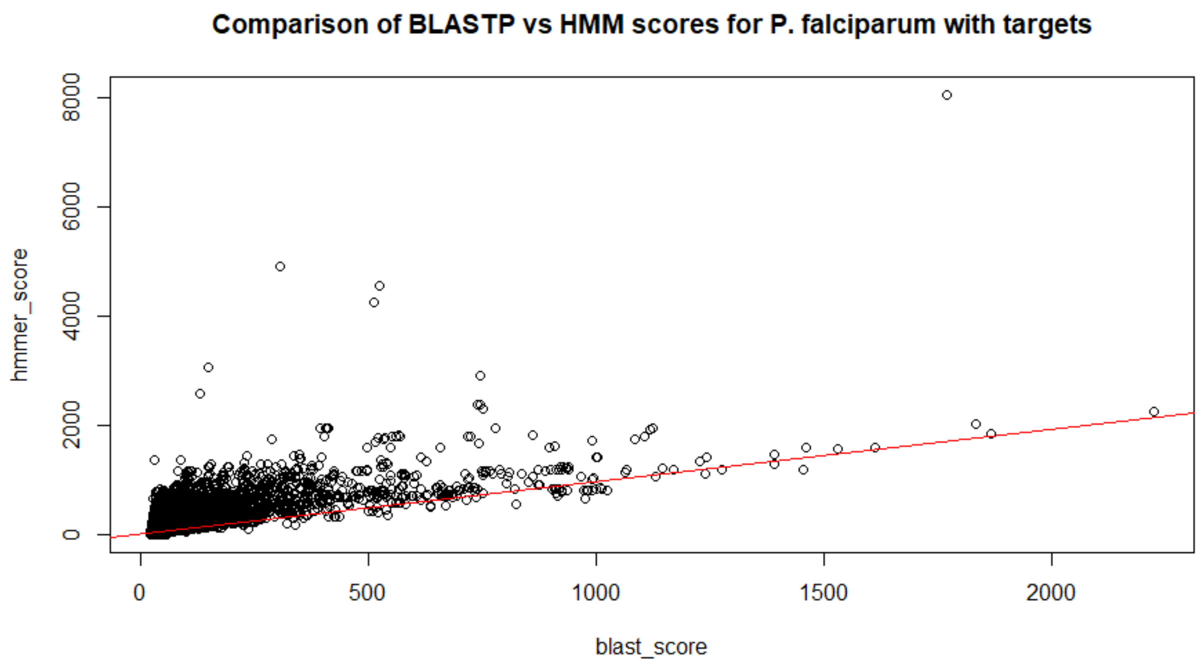


Figure 6: BLASTP vs HMM scoring comparison

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 7.4.2.

compare_scores.R). This may be attributed to **jackhmmers'**s increased sensitivity to structure over **BLASTP**.

The scale of *hmmer_score* to *blast_score* is **0.55**. We can use this later to estimate the appropriate selection threshold for *hmmer* statistics.

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*.

4.3. How departures from normality can reveal potential targets

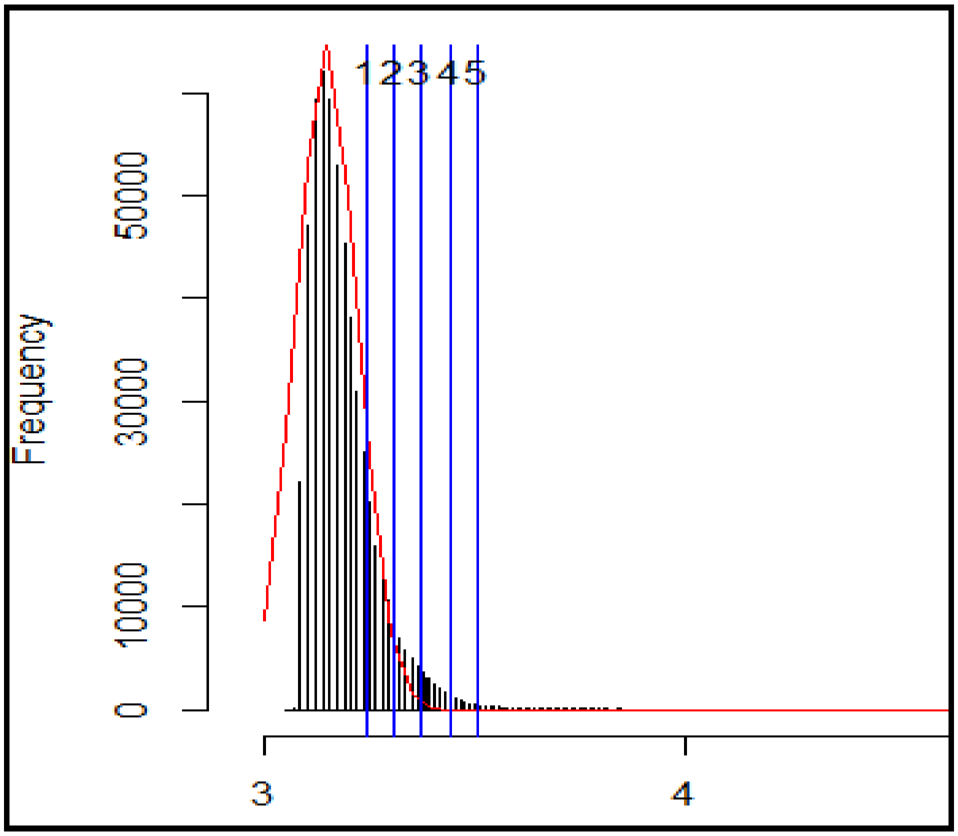


Figure 7: Histogram of log(score) for P. falciparum 3D7

Histogram of log BLASTP scores, showing superimposed normal plot in red.

Vertical blue lines show thresholds that are 1, 2, 3, 4, and 5 median absolute deviations from the median.

Normal curve was calculated using **R dnorm** function, which expects a standard deviation parameter, corrected here by a factor using **mad** function[19][20, 21]. Log scale was chosen to improve display of the curve.

Choosing a match metric and threshold is a conundrum. What determines whether a match is sufficiently close to suggest that it is likely to be a target?

While a very close match of the *positives* or *identities* (100% or 99%) would indicate that a match identifies a target by tautology, lower scores are hard to justify. A small sequence with a fairly high percentage may still not be as likely "hit" as a much larger string with a lower *positives* score and a much higher *score* statistic, as the cumulative score takes into account the greater difficulty of achieving a high score for the larger string.

The distribution of log(*score)* for all the matches in *P. falciparum* appears to be somewhat normal.

There are several reasons why we should expect these results.

1. The distribution is not symmetric around the mean because BLASTP discards matches where the score is too low, and only returns up to the ten highest scores per query.
2. Normally distributed matches are what we would expect due to random mutation for non-conserved sequences, and account for the normal appearance of the graph.
3. Due to conservation of essential peptides, there may be more high scoring matches than predicted by normally distributed processes. We can look for these likely target candidates in the area of the distribution which normally should be approaching zero asymptotically.
4. In addition, the target universe may be "lumpy" in the sense that there may be more than one centroid that has a family of similarities in the *P. falciparum* genome. This "lumpiness" is not important for purposes of finding unusual target similarity, as it is buried in a normal cohort.

It is interesting to note that the smallest *score* value for a comparison that has a 100% positives value is 21.6, which is close to the median of 23.9 (or log values of 3.07 and 3.17 respectively). Searching the upper tail means missing some targets that are closer to the median. Median and median absolute deviation were used as measures of centrality and dispersion rather than mean and standard deviation because these measures are more appropriate considering the asymmetry and contamination of the distribution (in the sense that the distribution contains more than a single normal distribution plus other non-normally distributed data.)

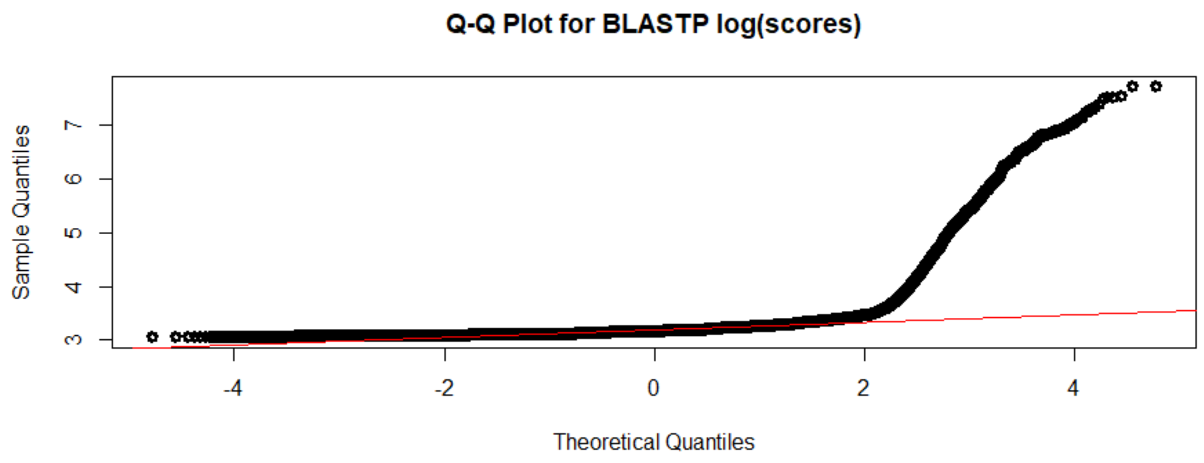


Figure 8:QQ Plot for all P. falciparum BLASTP log(scores)

The Q-Q plot reveals how closely the data distribution conforms to normality. A straight line indicates that the distribution is behaving normally. In the plot above, the red line shows expected normality. This plot shows departure from normality at the right (upper) portion of the graph.

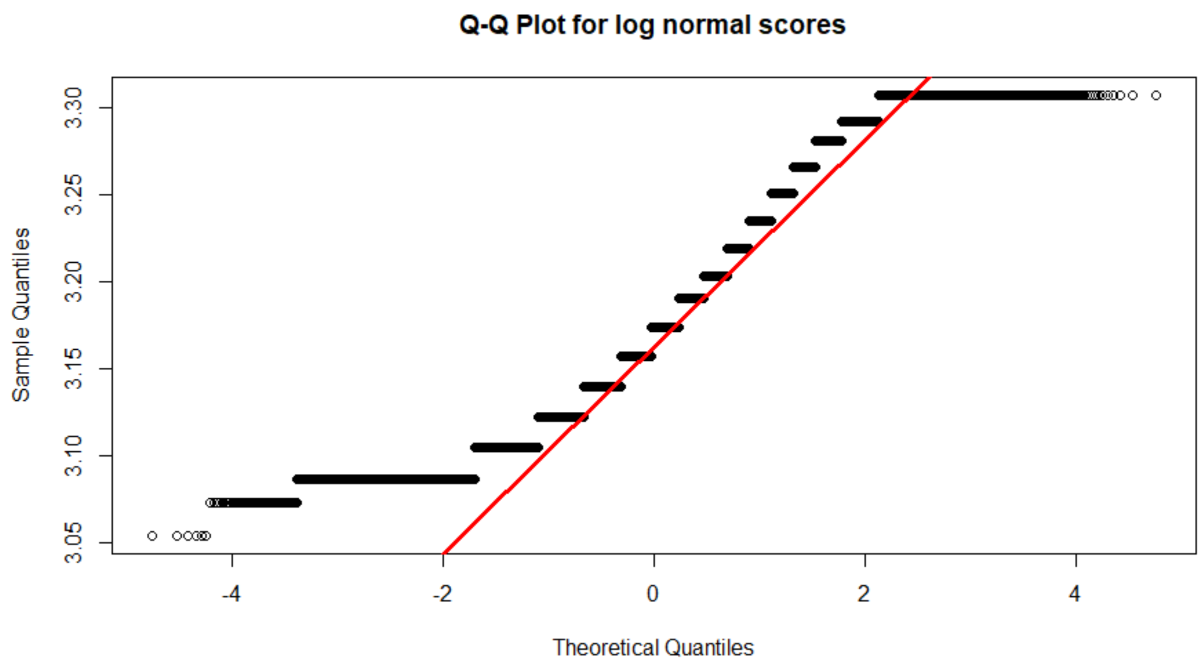


Figure 9: normal subset of log BLAST stats for P. falciparum

We chose a threshold of 2 *maximum absolute deviations* above the median as a cutoff below which the similarity results are *normal* and therefore *insignificant.* The normal score threshold was 27.16, and the log score threshold was 3.3073. We expect that scores above that will be more than randomly significant.

Applying the Q-Q plot to the *hmmer_score* values does not provide as neat an answer:

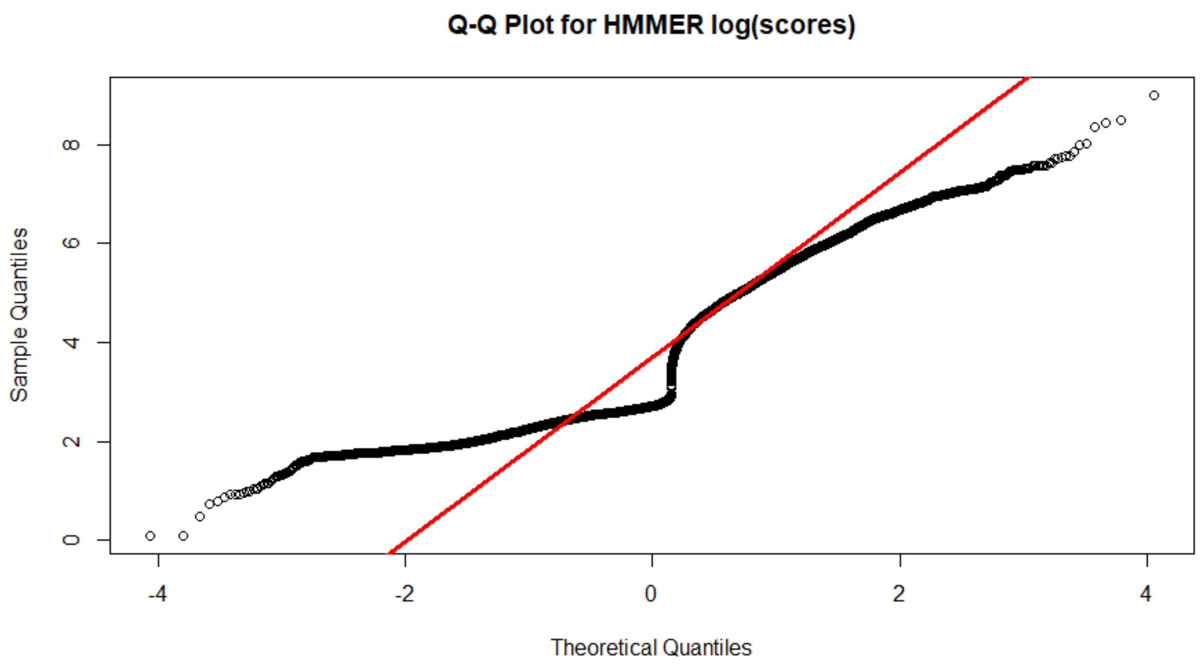


Figure 10: HMMER scores are less well behaved

However, the Q-Q plot shows a pronounced *jink* in the middle of the graph. This suggests that the HMMER scores may be easier to stratify into different similarity classes.

4.4. Using kmeans to find a significant cluster

Another approach to find a useful threshold is **kmeans** clustering [22, 23]. We are looking for a region that is less dense, with higher scores indicating unusually high similarity. **Kmeans** is a machine learning algorithm that classifies data by minimizing within-cluster sum of squares distance. We used it to classify our data into two clusters[22].

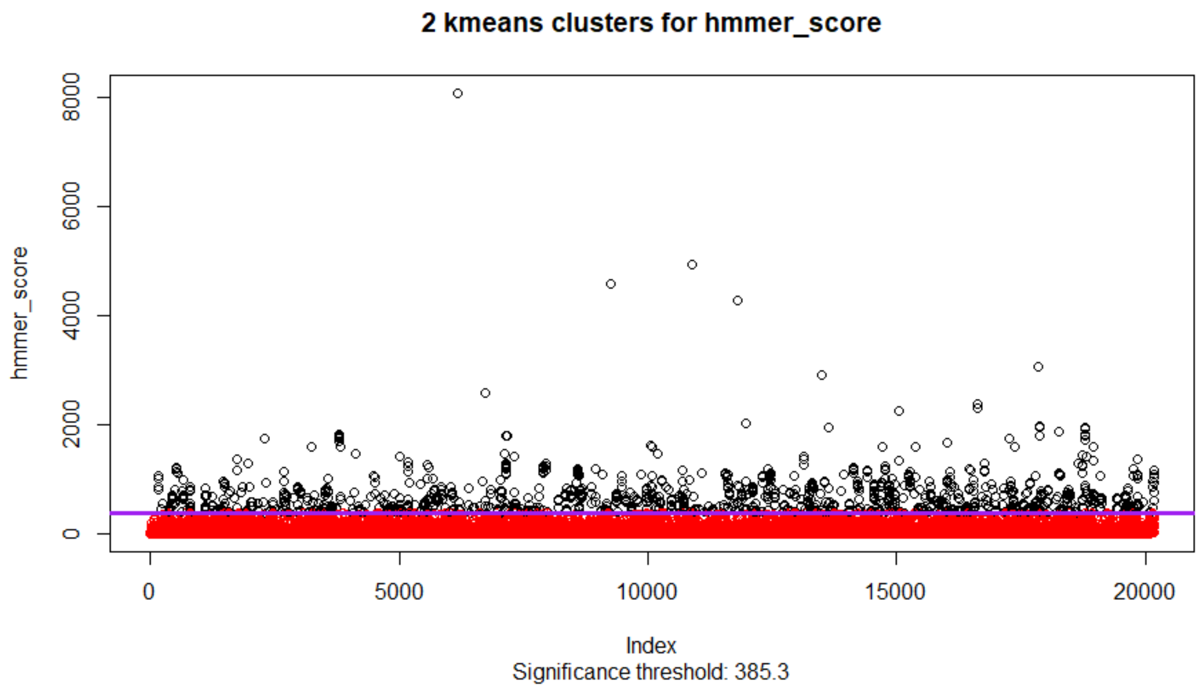


Figure 11: kmeans clustering of hmmer_score

The purple line shows the threshold separating less similar insignificant similarity (in red) from highly similar ORFs in black. The similarity threshold found by **kmeans** is more stringent than the one we found using *median* and *maximum absolute deviation*.

(See 7.4.3. Score normality and kmeans analysis)

1708 scores out of 20178 belong to the significant cluster.

Note that the significant cluster is much more dispersed than the insignificant cluster. The significance threshold should show a reasonable amount of separation within these clusters.

4.5. Downloading drugs and targets

Having determined a significance threshold, we are ready to select drugs and targets from the *CHEMBL_25* database. The query specifies tax_id to find targets for the associated organism. The score in this query was determined from kmeans, as described (See 7.4.3. Score normality and kmeans analysis).

```
select distinct h.target, score, td.tax_id as original_tax_id, td.organism as
orig_organism,td.pref_name, md.pref_name, md.chembl_id
from hmmer_statistics h
     join target_dictionary td
     on h.target = td.chembl_id
     join drug_mechanism dm
     ON dm.tid = td.tid
     join molecule_dictionary md
     ON dm.molregno = md.molregno
WHERE h.tax_id = 36329
  and score >= 385.3
order by score desc;
```

This query was reformatted as a **psql** command that downloads a tab delimited text file (See 7.7. Download *P. falciparum* drugs and targets).

The tab delimited file was imported into excel, providing a filterable document. This is useful, if we wish to exclude certain original organisms (9606 is human.) While the query downloads the data in descending score order, a user can sort this document by other columns, such as the drug *pref_name*.

P_falciparum_hmmer_drugs.xslx is included separately in the supplements.

At least one line for each target/drug with score from **jackhmmer** with document references, in pref_name / journal year order. (See 7.8. Download P. falciparum drugs and targets, with annotations) File **P_falciparum_hmmer_drugs_annotated.xslx** is included separately in the supplements.

Using R function “get_unique_drugs” (see 7.5. organism_hmmer_threshold.R

```
# organism_hmmer_threshold.R
# computes kmeans based threshold for selecting targets.

library(RPostgres)

db_name='chembl_25'
user_name = 'postgres'
host='192.168.1.180'
```

```
port=5432

conn = dbConnect(drv=RPostgres::Postgres(),
                 dbname=db_name,
                 user=user_name,
                 host=host,
                 port=port)

get_kmeans_threshold<-function(conn, tax_id, clusters=2){
  q_tax_org = paste0('SELECT distinct organism ',
                     'FROM target_dictionary ',
                     'where tax_id=',
                     tax_id)
  q_org_score = paste0(
    'select distinct score, orf, target
     from hmmer_statistics h
              join target_dictionary td
               on h.target = td.chembl_id
               join drug_mechanism dm
               ON dm.tid = td.tid
               join molecule_dictionary md
              ON dm.molregno = md.molregno
          WHERE h.tax_id ='
    , tax_id
  )

  org=dbGetQuery(conn,q_tax_org)
  org_score=dbGetQuery(conn, q_org_score)
  organism=org$organism[1]
  attach(org_score)
#  kmo=kmeans(score,2)
  kmo=kmeans(score,clusters)
  thresh=min(score[kmo$cluster==max(kmo$cluster)]) # minimum score of highest cluster

  plot(score,col=kmo$cluster, main=paste('kmeans for ',organism, ', threshold=',thresh))
  detach()
  return(thresh)
}

#dbDisconnect(conn)
```

7.6. get_unique_drugs.R) calculated the number of drugs by pref_name from the molecule_dictionary table.

726 unique approved drugs were found.

4.6. Validating drugs found

726 approved drugs were identified with **kmeans** threshold of 385.3.

Of the drugs found, 2 are already known in the *chembl_25* database for P. falciparum:

avg_score	original_tax_id	orig_organism	pref_name	chembl_id	mechanism_of_action	max_phase	first_approval
661.15	5833	Plasmodium falciparum	SULFACYTINE	CHEMBL1201056	Dihydropteroate synthetase inhibitor	4	1975
661.15	5833	Plasmodium falciparum	SULFADOXINE	CHEMBL1539	Dihydropteroate synthetase inhibitor	4	1981

These appear to be highly similar molecules differing by only one oxygen atom [24, 25].

vg_score	original_tax_id	orig_organism	pref_name	chembl_id	mechanism_of_action	max_phase	first_approval
1063.857143	2	Bacteria	AZITHROMYCIN	CHEMBL529	Bacterial 70S ribosome inhibitor	4	1991

Azithromycin was not known to be antimalarial in the **chembl_25** database, but was found independently in *ClinicalTrials.gov* as a treatment for *P. falciparum* malaria, demonstrating that paralog matching can find drugs that have been independently chosen for this use [26]. Azithromycin has completed clinical trials for treatment of *P. falciparum* malaria for uncomplicated malaria in combination with mefloquine, and in combination with other drugs for intermittent preventative use, validating our method [27, 28].

Clindamycin, identified in our screen, has had clinical trials in the 1970s and 1980s, with reviews in the 1990s [29]. Efficacy using Clindamycin alone with success varying from 89 to 100% has been shown in many trials in Africa, South America, and Southeast Asia [29].

avg_score	original tax_id	orig organism	pref_name	chembl_id	mechanism of action	max phase	first approval
459.78	2	Bacteria	CLINDAMYCIN HYDROCHLORIDE	CHEMBL1200588	Bacterial 70S ribosome inhibitor	4	1970
459.78	2	Bacteria	CLINDAMYCIN PALMITATE HYDROCHLORIDE	CHEMBL1200632	Bacterial 70S ribosome inhibitor	4	1986
459.78	2	Bacteria	CLINDAMYCIN PHOSPHATE	CHEMBL3184512	Bacterial 70S ribosome inhibitor	4	1972

Erythromycin, identified in our screen, is being studied in combination with Azithromycin or with quinine against multi-drug resistant *Plasmodium falciparum* in vitro, and alone [30],[31].

459.78	2	Bacteria	ERYTHROMYCIN ESTOLATE	CHEMBL2218877	Bacterial 70S ribosome inhibitor	4	1967
459.78	2	Bacteria	ERYTHROMYCIN ETHYLSUCCINATE	CHEMBL1200688	Bacterial 70S ribosome inhibitor	4	1965
459.78	2	Bacteria	ERYTHROMYCIN GLUCEPTATE	CHEMBL3545060	Bacterial 70S ribosome inhibitor	4	1982
459.78	2	Bacteria	ERYTHROMYCIN LACTOBIONATE	CHEMBL1200506	Bacterial 70S ribosome inhibitor	4	1964
459.78	2	Bacteria	ERYTHROMYCIN STEARATE	CHEMBL1200510	Bacterial 70S ribosome inhibitor	4	1964

Tetracyclines were identified in our screen, and have also been studied as antimalarials [32]. While adverse effects for Tetracyclines are well documented, Minocycline, which we also found in our results were approved as antimalarial in 1971 [33].

avg_score	original tax_id	orig organism	pref_name	chembl_id	mechanism of action	max phase	first approval
459.78	2	Bacteria	MINOCYCLINE HYDROCHLORIDE	CHEMBL1200881	Bacterial 70S ribosome inhibitor	4	1971

avg_scor e	original tax_id	orig organis m	pref_name	chembl_id	mechanis m of action	max phase	first approval
459.78	2	Bacteria	OXYTETRACYCLINE	CHEMBL1517	Bacterial 70S ribosome inhibitor	4	1964
459.78	2	Bacteria	OXYTETRACYCLINE CALCIUM	CHEMBL398956 8	Bacterial 70S ribosome inhibitor	4	1982
459.78	2	Bacteria	OXYTETRACYCLINE HYDROCHLORIDE	CHEMBL160748 0	Bacterial 70S ribosome inhibitor	4	1964

102 drugs are anti-bacterial. Of these, 41 are Bacterial 70S ribosome inhibitors; the rest are Bacterial penicillin-binding protein inhibitors.

4.7. Drugs for other organisms

Having demonstrated that we could find known drugs for *P. falciparum* malaria using our screening methods, we applied those methods to 5 other organisms identified by the WHO as neglected tropical diseases [34].

4.7.1. Plasmodium Vivax

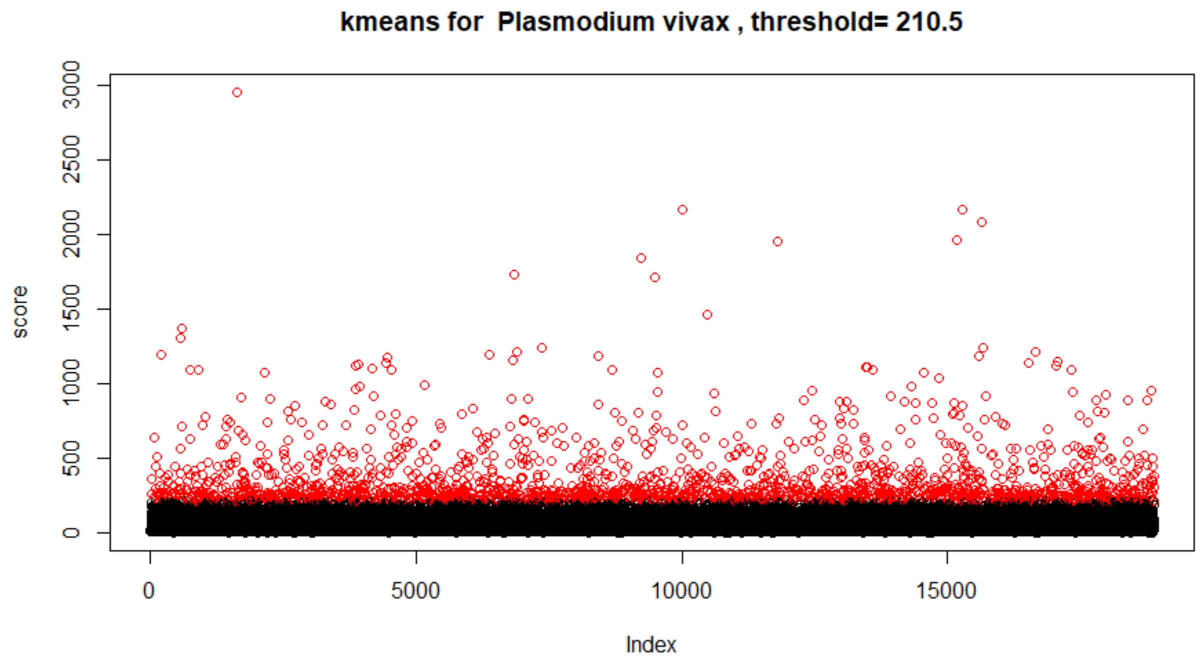
Taxonomy ID: 5855 (Non-strain specific)[2].

P. Vivax is responsible for 75% of the malaria burden in the Americas, and 53% of the malaria burden in the South-East Asia region [35].

Currently, intravenous artesunate treatment has shown rapid clinical response in P. vivax malaria, but there are no randomized clinical trials so far, but artesunate, artemether, and quinine are now recommended for severe *P. vivax malaria* [36].

Search was based on `PlasmoDB-46_PvivaxP01_ORFs_AA.fasta[37, 38].`

`Kmeans threshold computed using` (7.5. organism_hmmer_threshold.R)`.  tax_id = 5855.`



Drugs were downloaded by (7.9. P_vivax_jackhmmer_drugs.sql).

Spreadsheets of results are contained in the supplements as **P_vivax_hmmer_drugs.xlsx** and **P_vivax_hmmer_drugs_annotated.xlsx**.

Using an R function (see 7.5. organism_hmmer_threshold.R

```
# organism_hmmer_threshold.R
# computes kmeans based threshold for selecting targets.

library(RPostgres)

db_name='chembl_25'
user_name = 'postgres'
host='192.168.1.180'
port=5432

conn = dbConnect(drv=RPostgres::Postgres(),
                 dbname=db_name,
                 user=user_name,
                 host=host,
                 port=port)

get_kmeans_threshold<-function(conn, tax_id, clusters=2){
  q_tax_org = paste0('SELECT distinct organism ',
                     'FROM target_dictionary ',
                     'where tax_id=',
                     tax_id)
  q_org_score = paste0(
    'select distinct score, orf, target
     from hmmer_statistics h
              join target_dictionary td
               on h.target = td.chembl_id
               join drug_mechanism dm
               ON dm.tid = td.tid
               join molecule_dictionary md
              ON dm.molregno = md.molregno
          WHERE h.tax_id ='
    , tax_id
  )

  org=dbGetQuery(conn,q_tax_org)
  org_score=dbGetQuery(conn, q_org_score)
  organism=org$organism[1]
  attach(org_score)
#  kmo=kmeans(score,2)
  kmo=kmeans(score,clusters)
  thresh=min(score[kmo$cluster==max(kmo$cluster)]) # minimum score of highest cluster

  plot(score,col=kmo$cluster, main=paste('kmeans for ',organism, ', threshold=',thresh))
  detach()
  return(thresh)
}

#dbDisconnect(conn)
```

7.6. get_unique_drugs.R) we can retrieve unique drugs and do calculations in R Studio.

```
falciparum_drugs=get_unique_drugs(conn, 36329,221.2) # query returns drug names for tax_id,
threshold

vivax_drugs=get_unique_drugs(conn,5855,210.5) # query returns drug names for tax_id, threshold

dim(vivax_drugs)[1] # returns number of rows
```

721 unique approved drugs were found.

Differences were found for drugs that could be applied to *Plasmodium falciparum* vs. *Plasmodium vivax* malaria using R's setdiff function.

The following drugs were found for *P. falciparum* but not *P. vivax*:

ABEMACICLIB, CHOLINE FENOFIBRATE, CLOFIBRATE, EZETIMIBE, FENOFIBRATE, FENOFIBRIC ACID, GEMFIBROZIL, PENTOXIFYLLINE, RIBOCICLIB, RIBOCICLIB SUCCINATE.

The following drugs were found for *P. vivax* but not *P. falciparum*:

CLOXACILLIN SODIUM, METHICILLIN SODIUM, PERMETHRIN, SONIDEGIB PHOSPHATE, VISMODEGIB.

The R intersect function shows that there are 721 approved *P. falciparum drugs* and *P. vivax drugs* in common.

4.7.2. Cryptosporidium parvum

Taxonomy ID: 5807 (non-strain specific)[2]

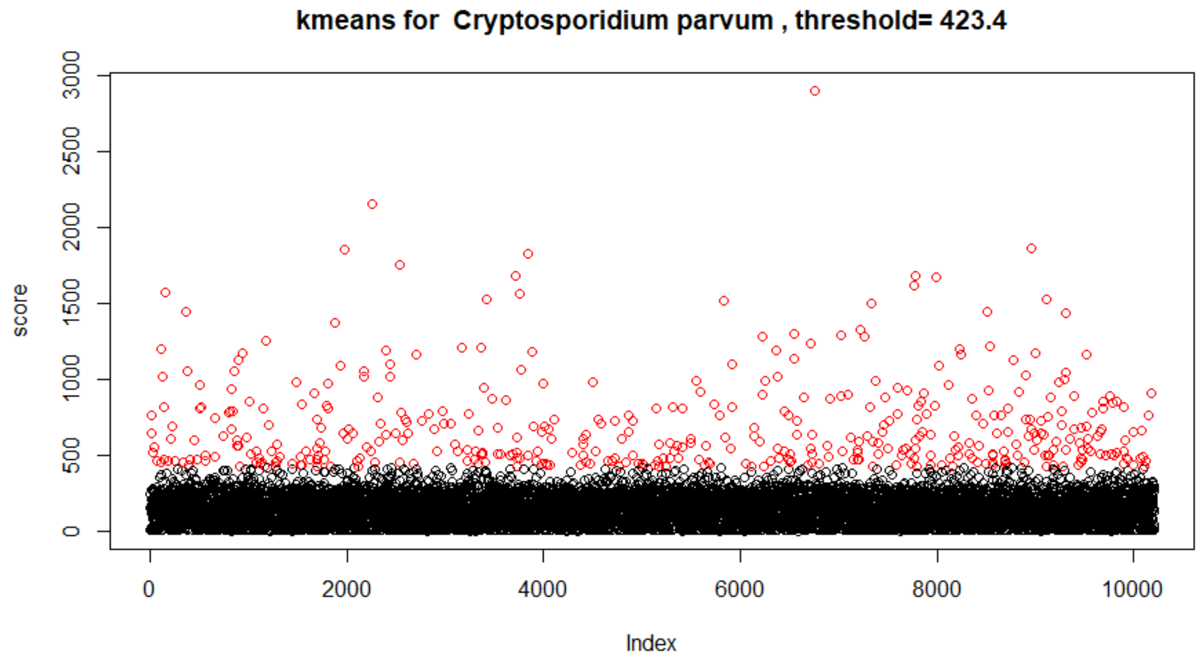
Cryptosporidiosis is an intestinal disease-causing diarrhea caused by the apicoplexan parasite *Cryptosporidium parvum* and *Cryptosporidium hominis*. Only *C.* parvum is considered here. While it is not include in WHO's analysis of neglected tropical diseases, it causes at least 8.7 million disability life years (DALYs), which may be un underestimate [34]. *C. parvum* has several genotypes that prefer different hosts. The genotype that is most commonly found in in immunocompromised humans is the one that prefers cattle, although genotypes that infect other mammals are occasionally found [39].

Current treatments include attacking the parasite with drugs such as Azithromycin, along with anti-motility agents, fluid replacement, and anti-retrovirals for those suffering from HIV/AIDS [40].

Search was based on **`CryptoDB-46_CparvumIOWA-ATCC_ORFs_AA.fasta[41, 42].`**

`Kmeans threshold computed using (`7.5. organism_hmmer_threshold.R`).  tax_id = 5807.`

`Threshold is 423.4.`



Spreadsheets of results are contained in the supplements as **C_parvum_hmmer_drugs.xlsx** an**d** **C_parvum_hmmer_drugs_annotated.xlsx**.

561 approved drugs were found for Cryptosporidium parvum.

4.7.3. Trypanosoma cruzi Brazil A4

Taxonomy ID: 5693 (Non-strain specific)[2].

Trypanosoma cruzi causes Chagas disease (also known as American Trypanosomiasis) is a blood disease transmitted by the bite of insect vectors in the Americas. While it is not directly fatal in the short run, it is a debilitating disease that can damage the esophagus, lymph nodes, colon, and can cause congestive heart failure [43].

Current treatment employs benznidazole or nifurtimox. Although both are very effective, effectiveness declines the longer the infection, and adverse reactions to the drugs increase with patient age [44].

Search was based on `TriTrypDB-46_TcruziBrazilA4_ORFs_AA.fasta` **[45, 46].**

This file has 1,707,427 consisting of 528,196 ORFs.

The **Contig** directory created by **fan_out_fasta_tryp.R** contains too many files to be handled by bash **ls** command:

```
[osboxes@osboxes ~/genomes/TcBrA4 ] ls -l Contig/*
-bash: /usr/bin/ls: Argument list too long
```

Using Perl, we find that the number of files in the **Contig** directory:

```
main::(-e:1):   4
  DB<1> @files=glob("*.*");

  DB<2> print(scalar(@files));
108582
```

Two bash scripts were rewritten as Perl scripts to overcome these limitations (See 7.3.1. do_all_jackhmmer.pl, 7.3.4. do_all_hmmer_stats.pl).

These were applied against organism directory TcBrA4:

```
perl do_all_jackhmmer.pl TcBrA4/
```

and

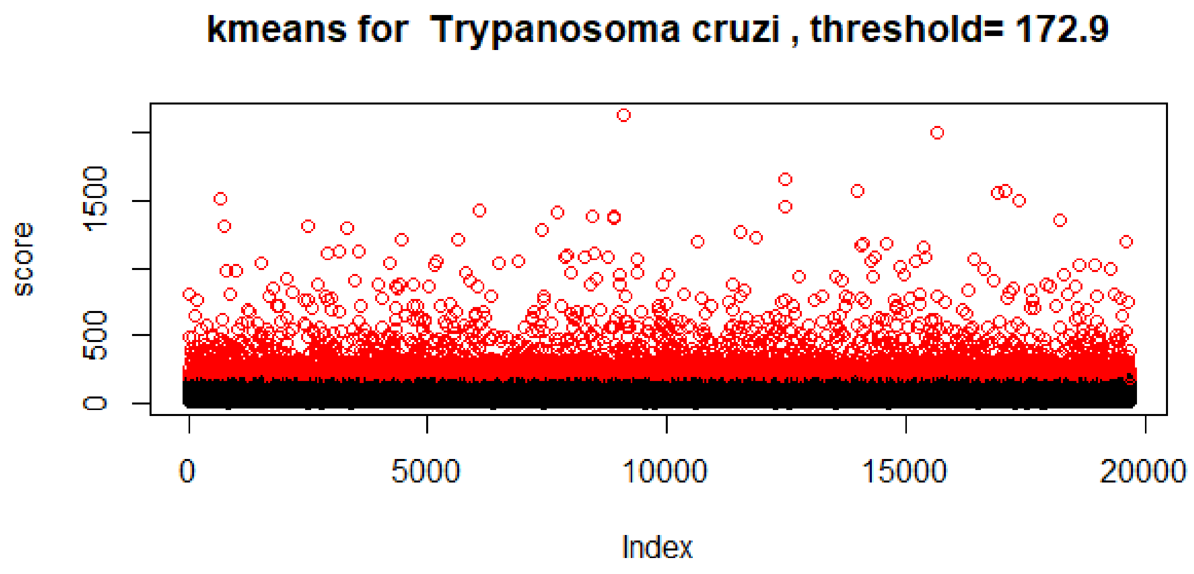
```
perl do_all_hmmer_statistics.pl TcBrA4/
```

After these scripts completed:

```
sudo su postgres

psql -U postgres -d chembl_25

\i import_hmmer_statistics.sql
```

```
TRUNCATE TABLE
INSERT 0 144447
chembl_25=#

update hmmer_statistics set tax_id = 5693, organism = 'Trypanosoma cruzi' where tax_id
is null;
```

Kmeans threshold computed using (7.5. organism_hmmer_threshold.R).



Spreadsheets of results are contained in the supplements as **T_cruzi_hmmer_drugs.xlsx** and **T_cruzi_hmmer_drugs_annotated.xlsx**.

809 approved drugs were found for Trypanosoma cruzi.

4.7.4. Trypanosoma brucei gambiense DAL972

Taxonomy ID: 679716[47].

Trypanosoma brucei gambiense causes West African Tryponosomiasis (also known as Sleeping Sickness.)

This blood borne parasite is transmitted by the bite of the Tsetse fly [48].

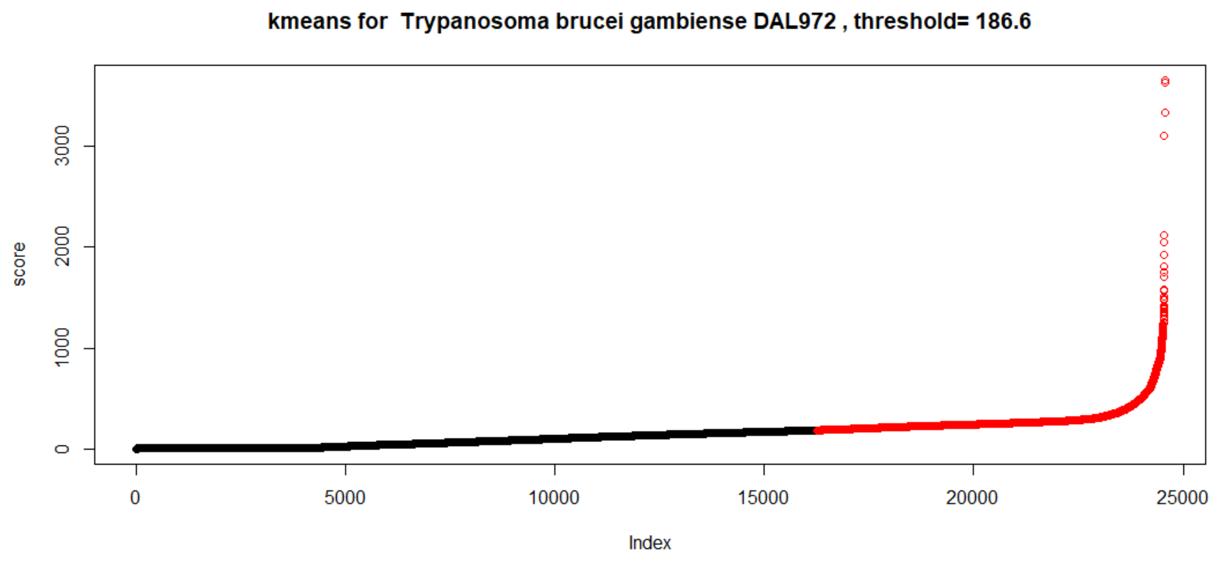
Current treatment relies on Pentamidine (the first treatment recommendation), suramin, melarsoprol, eflornithine, and nifortimax [49].

Search was based on **TriTrypDB-46_TbruceigambienseDAL972_ORFs_AA.fasta.**[23],[22]

```
perl do_all_jackhmmer.pl Tbg972/
```

```
perl do_all_hmmer_stats.pl Tbg972/
```

```
[osboxes@osboxes ~/genomes ] sudo su postgres
[sudo] password for osboxes:
[postgres@osboxes /home/osboxes/genomes] psql -U postgres -d chembl_25
psql (9.2.24)
Type "help" for help.

chembl_25=# \i import_hmmer_statistics.sql
TRUNCATE TABLE
INSERT 0 179183
chembl_25=# update hmmer_statistics set tax_id = 31285,organism = 'Trypanosoma brucei
gambiense' where tax_id is null;
UPDATE 179183
chembl_25=#
```

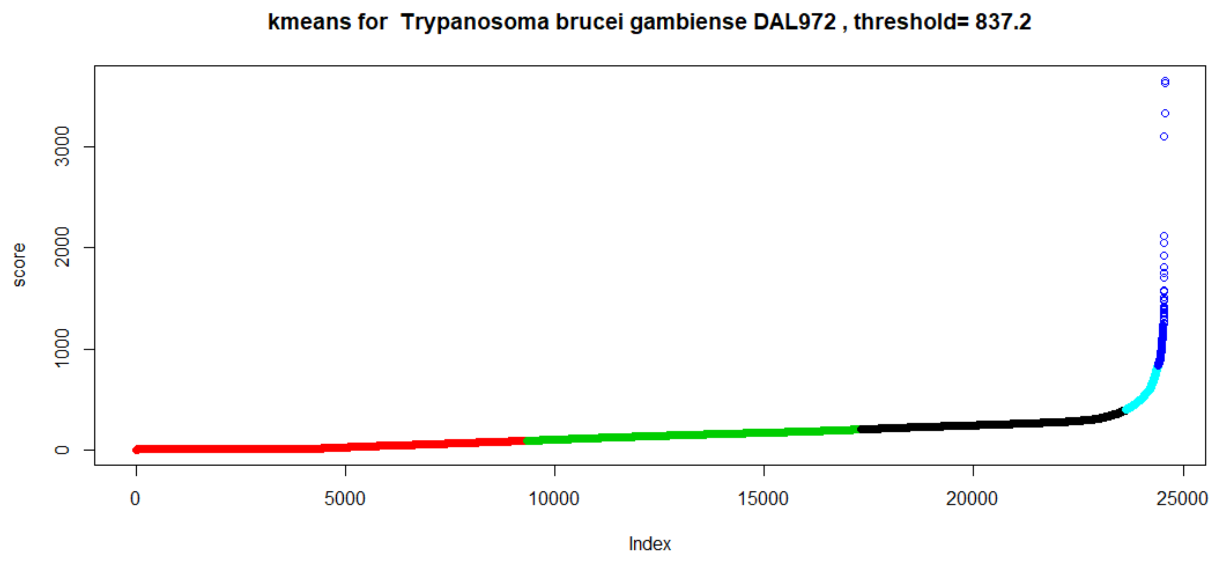


This kmeans threshold does not seem to be partitioning the data in a convincing way.

Adjusting the number of clusters identifies a cluster after the inflection point of the above graph:

`Kmeans threshold computed using` (7.5. organism_hmmer_threshold.R)

get_kmeans_threshold(conn, 31285, 5)



Cluster (in blue) chosen as significant cluster.

236 approved drugs were found for Trypanosoma brucei gambiense DAL972.

Spreadsheets of results are contained in the supplements: **T_brucei_hmmer_drugs.xlsx, T_brucei_hmmer_drugs_annotated.xlsx.**

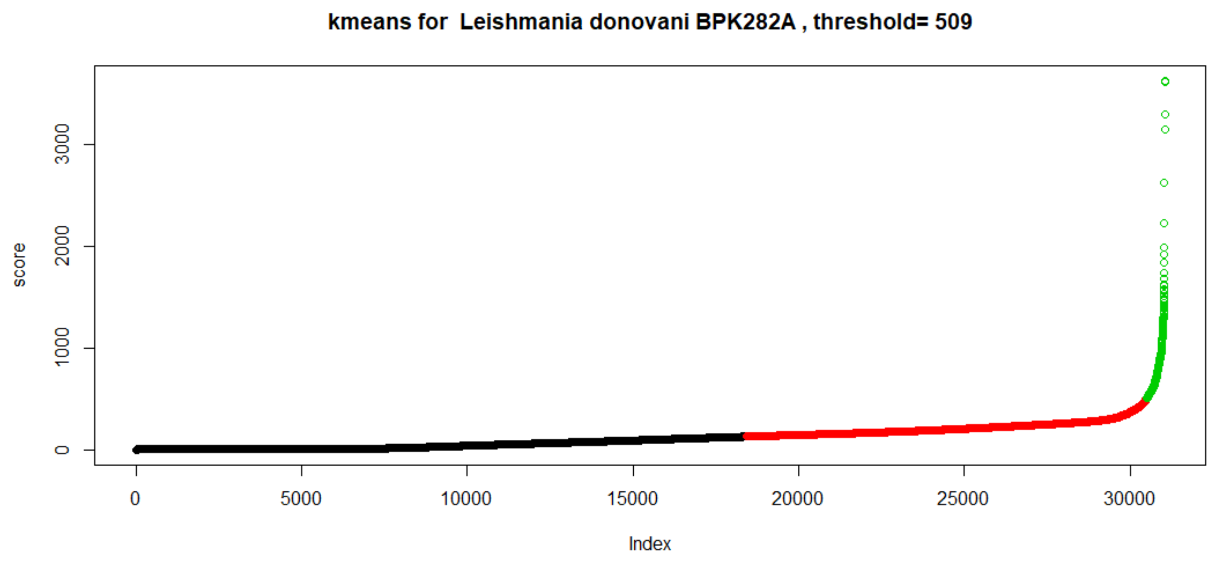
4.7.5. Leishmania donovani BPK282A

Taxonomy ID: 981087[50]

Leishmania donovani causes Leishmaniasis, a disease that affects 700,000 to 1,000,000 people annually [51]. This obligate parasite causes Leishmaniasis, which manifests most commonly in a cutaneous variant, (causing sores and ulcers), and as a visceral form [52]. The visceral form has a 95% fatality rate if left untreated [52]. It is one of the five diseases that has top priority from the World Health Organization (WHO)[34].

Search was based on `TriTrypDB-46_LdonovaniBPK282A1_ORFs_AA.fasta [46].`

Use these parameters in R routine: `get_kmeans_threshold(conn,981087,3)`



438 approved drugs were found for Leishmania donovani BPK282A.

Spreadsheets of results are included in the supplements: **Leishmania_hmmer_drugs.xlsx, Leishmania_hmmer_drugs_annotated.xlsx.**

4.7.6. SARS-CoV-2

Taxonomy ID: 2697049 [53].

This Coronavirus, also known as COVID-19, is a virus that causes flu like symptoms including respiratory distress, in many cases requiring respirators to maintain oxygenation in patients. It is highly contagious, and is currently causing pandemic infection, with a fatality rate estimated between 2% and 3% [54]. Persons over 60 have may have much higher fatality rates [55].

The disease is spread by droplets, either by close contact, by touching surfaces that have come in contact with an infected person, or by aerosolized droplets [56, 57].

Experimental trials for treatment with Remdesivir are in progress [58].

The nucleotide genome of the virus was downloaded as MN908947.3.FASTA [59].

ORFs were translated using EMBOSS tools [60].

```
[osboxes@osboxes ~/genomes/MN908947.3 ] getorf MN908947.3.FASTA
```

This creates file MN908947.3.orf, which contains all the ORFs found for the .FASTA file.

Commands run in R Studio quantify how many ORFs are contained:

```
> aa=read.table(file="mn908947.orf",header = FALSE, sep='~', stringsAsFactors = FALSE)
> aa=aa[!is.na(aa[,1]),] # filter out NA
> aa=data.frame(lines=aa, stringsAsFactors = FALSE)
> orf_headers=aa[substr(aa[,1],1,1)=='>' ,]
> length(orf_headers)
[1] 1572
```

1572 ORFs were found.

Jackhmmer was used to create reports and summaries of similarities with targets [6].

```
[osboxes@osboxes ~/genomes/MN908947.3 ] jackhmmer --domtblout orf.summary -o orf.hmm.txt mn908947.orf
~/hmmer_targets/component_sequences.fa
```

A perl script (See 7.3.2. extract_hmm_summary.pl .)

```
[osboxes@osboxes ~/genomes ] perl extract_hmm_summary.pl MN908947.3/orf.hmm.txt >> hmm_stats.txt
```

From psql, the data were imported into the **chembl_25** database:

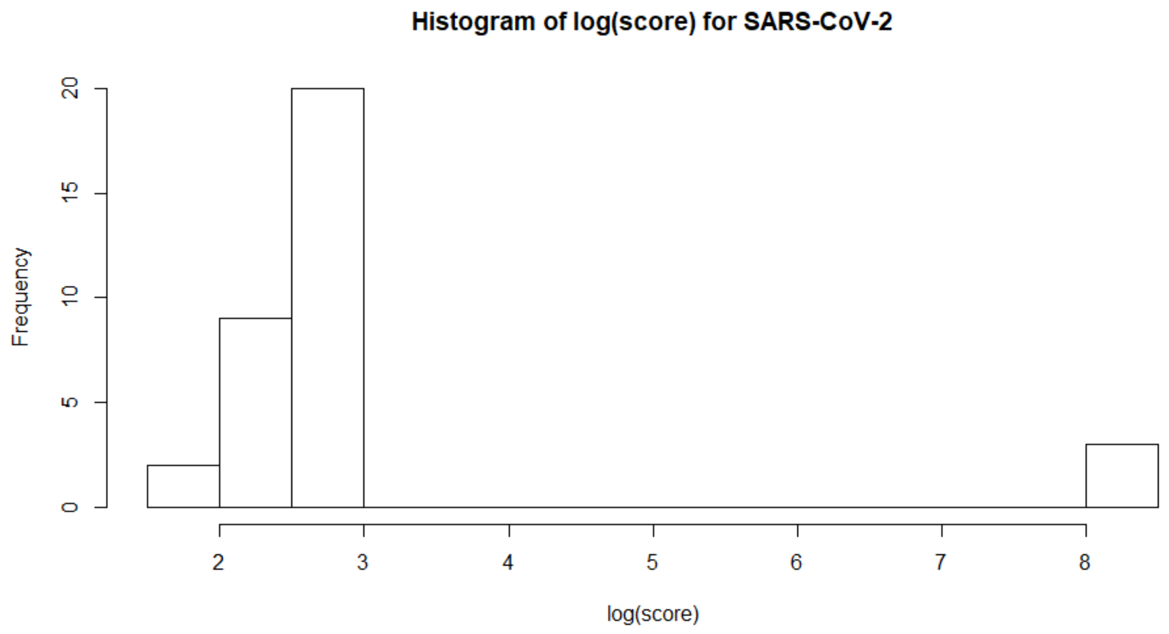
```
[postgres@osboxes /home/osboxes/genomes] psql -U postgres -d chembl_25
psql (9.2.24)
Type "help" for help.

chembl_25=# \i import_hmmer_statistics.sql
TRUNCATE TABLE
INSERT 0 49
chembl_25=# update hmmer_statistics set tax_id=2697049, organism='SARS-CoV-2
```

49 ORFs had enough similarity to targets to participate in our analysis.

Because of the small number of results, we examined the data more closely.

We performed the following histogram of results:

(See 7.16. hmmer_hist.R .)

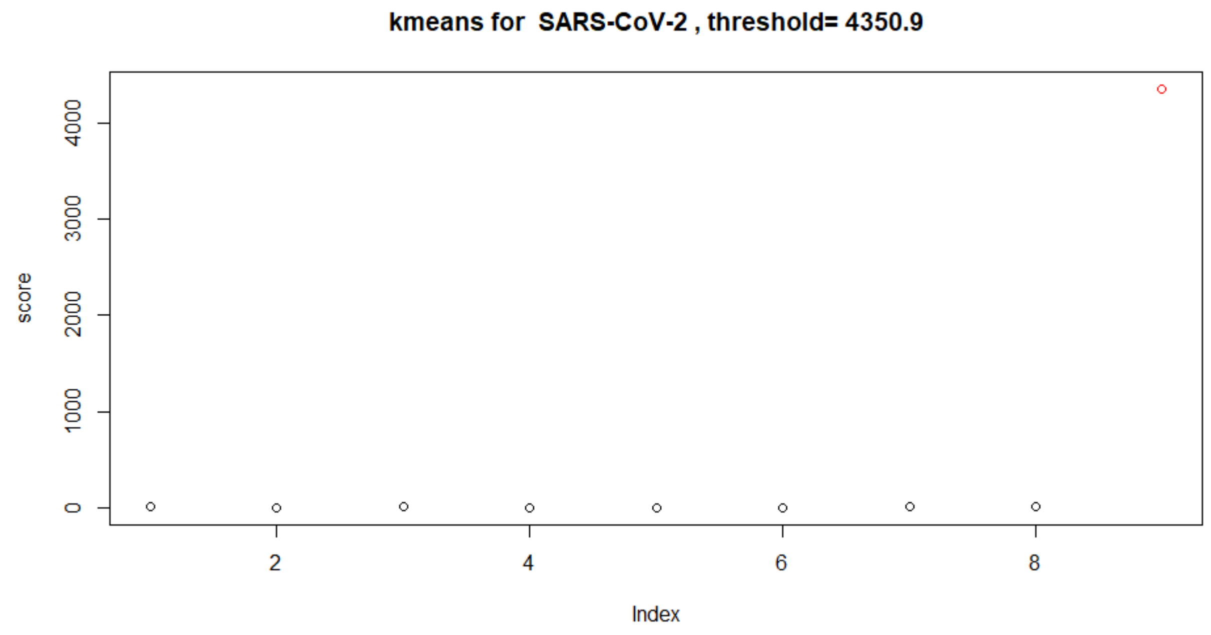


From R Studio, the *get_kmeans_threshold* routine (See 7.5. organism_hmmer_threshold.R .)

The data appear to show strong grouping of results.

```
get_kmeans_threshold(conn,2697049)
```

```
4350.9
```



Three drugs were found:

score	original tax id	orig organism	pref_name	chembl_id	mechanism_of_action	max phase	first approval
4350.9	1773	Mycobacterium tuberculosis	CAPREOMYCIN SULFATE	CHEMBL2218913	70S ribosome inhibitor	4	1971
4350.9	1773	Mycobacterium tuberculosis	PYRAZINAMIDE	CHEMBL614	70S ribosome inhibitor	4	1971
4350.9	1773	Mycobacterium tuberculosis	VIOMYCIN SULFATE	CHEMBL3989823	70S ribosome inhibitor	4	1982

Results saved as **Sars-CoV2_hmmer_drugs.xlsx**, **Sars-Cov2_hmmer_drugs_annotated.xlsx**.

5. Conclusions

Using paralog similarity, we validated a method of discovering cross-species targets by identifying 726 unique approved drugs for *P. falciparum* malaria. Of those, 5 were existing drugs that had been approved for use treating malaria in other trials. Drugs and targets were also identified for 6 other disease organisms: *Plasmodium vivax, Cryptosporidium parvum, Trypanosoma brucei, Trypanosoma cruzi, Leishmania donovan*i, *SARS-CoV-2. P. falciparum, P. vivax, and C. parvum* are all apicoplexans, while *T. brucei* and *T. cruzi* are kinetoplastids. We are not limited to any specific type of pathogenic organism, requiring only that the organism have enough genes to make a convincing case that there are distinguishable similarity clusters. In the case of SARS-CoV-2, there was enough clustering of highly similar results, even though the number of statistics were fairly low. Although most of the organisms we investigated had libraries of ORFs already translated, we showed that with a little more effort we can do the same kind of analysis using only nucleic acid sequences, as we did with SARS-CoV-2.

This platform provides a way to choose candidate drugs without knowing the identity of the pathogen if the pathogen's genome can be obtained. Because this method relies on intrinsic similarity with targets, it can also discriminate between different strains of otherwise similar species, providing strain specific recommendations. This method is fast, inexpensive, and provides access to rich annotations from the ChEMBL database [1, 2] providing information about dosage, safety, and previous experience with the recommended drugs. When time is of the essence and budget is lacking, this method can provide an inexpensive and rapid way to get started.

6. References

1. Gaulton A, Bellis LJ, Bento AP, Chambers J, Davies M, Hersey A, Light Y, McGlinchey S, Michalovich D, Al-Lazikani B *et al*: **ChEMBL: a large-scale bioactivity database for drug discovery**. *Nucleic Acids Res* 2012, **40**(Database issue):D1100-1107.
2. Gaulton A, Hersey A, Nowotka M, Bento AP, Chambers J, Mendez D, Mutowo P, Atkinson F, Bellis LJ, Cibrián-Uhalte E *et al*: **The ChEMBL database in 2017**. *Nucleic Acids Res* 2017, **45**(Database issue):D945-954.
3. Aurrecoechea C, Barreto A, Basenko EY, Brestelli J, Brunk BP, Cade S, Crouch K, Doherty R, Falke D, Fischer S *et al*: **EuPathDB: the eukaryotic pathogen genomics database resource**. *Nucleic Acids Res* 2017, **45**(Database issue):D581-591.
4. **World malaria report 2019** [https://www.who.int/news-room/feature-stories/detail/world-malaria-report-2019]
5. Altschul SF, Madden TL, Schäffer AA, Zhang J, Zhang Z, Miller W, Lipman DJ: **Gapped BLAST and PSI-BLAST: a new generation of protein database search programs**. In: *Nucleic Acids Res.* vol. 25; 1997: 3389-3402.
6. Wheeler TJ, HHMI Janelia Farm Research Campus A, VA 20147, USA, Eddy SR, HHMI Janelia Farm Research Campus A, VA 20147, USA: **nhmmer: DNA homology search with profile HMMs**. *Bioinformatics* 2013, **29**(19):2487-2489.
7. **WHO | Responding to antimalarial drug resistance**. *WHO* 2018.
8. Khan MA, Smego RA, Jr., Razi ST, Beg MA: **Emerging drug--resistance and guidelines for treatment of malaria**. *J Coll Physicians Surg Pak* 2004, **14**(5):319-324.
9. Aurrecoechea C, Brestelli J, Brunk BP, Dommer J, Fischer S, Gajria B, Gao X, Gingle A, Grant G, Harb OS *et al*: **PlasmoDB: a functional genomic database for malaria parasites**. In: *Nucleic Acids Res.* vol. 37; 2009: D539-543.
10. Lv W, Xu Y, Guo Y, Yu Z, Feng G, Liu P, Luan M, Zhu H, Liu G, Zhang M *et al*: **The drug target genes show higher evolutionary conservation than non-target genes**. In: *Oncotarget.* vol. 7; 2016: 4961-4971.
11. Alberts B: **Molecular Biology of the Cell**, 6 edn. Kindle Edition. : W. W. Norton & Company.
12. Gupta A, Shah P, Haider A, Gupta K, Siddiqi MI, Ralph SA, Habib S: **Reduced ribosomes of the apicoplast and mitochondrion of Plasmodium spp. and predicted interactions with antibiotics**. In: *Open Biol.* vol. 4; 2014.
13. Dimasi J: **Innovation in the pharmaceutical industry: New estimates of R&D costs**. *Journal of Health Economics* 2016, **47**(May 2016):3.
14. Janes J, Young ME, Chen E, Rogers NH, Burgstaller-Muehlbacher S, Hughes LD, Love MS, Hull MV, Kuhen KL, Woods AK *et al*: **The ReFRAME library as a comprehensive drug repurposing library and its application to the treatment of cryptosporidiosis**. 2018.
15. **On the Cost of Floating-Point ComputationWithout Extra-Precise Arithmetic** [https://people.eecs.berkeley.edu/~wkahan/Qdrtcs.pdf]
16. Campbell MP: **Losing My Precision: Tips For Handling Tricky Floating Point Arithmetic**. *Society of Actuaries* 2014(April 2014).
17. **BLAST Options and Defaults** [https://www.arabidopsis.org/Blast/BLASToptions.jsp]
18. Lamesch P, Department of Plant Biology CI, 260 Panama St., Stanford, CA 94305, USA, Berardini TZ, Department of Plant Biology CI, 260 Panama St., Stanford, CA 94305, USA, Li D, Department of Plant Biology CI, 260 Panama St., Stanford, CA 94305, USA, Swarbreck D, Department of Plant Biology CI, 260 Panama St., Stanford, CA 94305, USA, Wilks C, Department of Plant Biology CI,

260 Panama St., Stanford, CA 94305, USA *et al*: **The Arabidopsis Information Resource (TAIR): improved gene annotation and new tools**. *Nucleic Acids Research* 2020, **40**(D1).
19. **Lognormal function | R Documentation** [https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/Lognormal]
20. **MAd-package: Meta-Analysis with Mean Differences in MAd: Meta-Analysis with Mean Differences**. 2014.
21. Pham-Gia T, Hung TL: **PII: S0895-7177(01)00109-1 | Elsevier Enhanced Reader**. 2000.
22. **forgy: Initialization of cluster prototypes using Forgy's algorithm in inaparc: Initialization Algorithms for Partitioning Cluster Analysis**. 2020.
23. Hartigan JA, Wong MA: **Algorithm AS 136: A K-Means Clustering Algorithm**. *Journal of the Royal Statistical Society Series C (Applied Statistics)* 1979, **28**(1):100-108.
24. **Sulfadoxine | C12H14N4O4S | ChemSpider** [http://www.chemspider.com/Chemical-Structure.16218.html]
25. **Sulfacytine | C12H14N4O3S | ChemSpider** [http://www.chemspider.com/Chemical-Structure.5131.html]
26. **Search of: p falciparum malaria - List Results - ClinicalTrials.gov**. 2020.
27. **Azithromycin Plus Chloroquine Versus Mefloquine for the Treatment of Uncomplicated Malaria in Africa - Full Text View - ClinicalTrials.gov**. 2020.
28. Rosenthal PJ: **Azithromycin for Malaria?** *Am J Trop Med Hyg* 2016, **95**(1):2-4.
29. Lell B, Kremsner PG: **Clindamycin as an Antimalarial Drug: Review of Clinical Trials**. 2002.
30. Nakornchai S, Konthiang P: **Activity of azithromycin or erythromycin in combination with antimalarial drugs against multidrug-resistant Plasmodium falciparum in vitro**. *Acta Trop* 2006, **100**(3):185-191.
31. **In vitro evaluation of erythromycin in chloroquine resistant brazilian P. falciparum freshly isolates: modulating effect and antimalarial activity evidence**
32. Gaillard T, Madamet M, Pradines B: **Tetracyclines in malaria**. *Malaria Journal* 2015, **14**(1):1-10.
33. Willerson D: **Effects of Minocycline against Chloroquine-Resistant Falciparum Malaria* | The American Journal of Tropical Medicine and Hygiene**. *American Journal of Tropical Medicine and Hygiene* 1972, **21**(6):857-862.
34. Pisarski K: **The Global Burden of Disease of Zoonotic Parasitic Diseases: Top 5 Contenders for Priority Consideration**. In: *Trop Med Infect Dis.* vol. 4; 2019.
35. **World malaria report 2019**. 2019.
36. Baird JK, Valecha N, Duparc S, White NJ, Price RN: **Diagnosis and Treatment of Plasmodium vivax Malaria**. *Am J Trop Med Hyg* 2016, **95**(6 Suppl):35-51.
37. **PlasmoDB Download Files** [https://plasmodb.org/common/downloads/Current_Release/PvivaxP01/fasta/data/]
38. **Data Set GeneDB: The Sanger Institute Pathogen Genomics Database** [https://plasmodb.org/plasmo/app/record/dataset/DS_365c388131]
39. **WHO Guidelines for Drinking Water Quality** [https://www.who.int/water_sanitation_health/gdwqrevision/cryptodraft2.pdf]
40. **Cryptosporidium infection - Diagnosis and treatment - Mayo Clinic** [https://www.mayoclinic.org/diseases-conditions/cryptosporidium/diagnosis-treatment/drc-20351876]
41. **Data Set The Universal Protein Resource (UniProt)** [https://cryptodb.org/cryptodb/app/record/dataset/DS_81c1c359dc]
42. **CryptoDB Download Files** [https://cryptodb.org/common/downloads/Current_Release/CparvumIOWA-ATCC/fasta/data/]
43. Prevention C-CfDCa: **CDC - Chagas Disease**. 2019.

44. **Chagas disease**. 2020.
45. **Data Set Entrez Gene** [https://tritrypdb.org/tritrypdb/app/record/dataset/DS_975f67b61e]
46. **TriTrypDB Download Files** [https://tritrypdb.org/common/downloads/Current_Release/TcruziBrazilA4/fasta/data/]
47. taxonomy: **Taxonomy browser (Trypanosoma brucei gambiense DAL972)**. 2020.
48. **CDC - African Trypanosomiasis - Biology**. 2019.
49. **CDC - African Trypanosomiasis - Treatment**. 2019.
50. taxonomy: **Taxonomy browser (Leishmania donovani BPK282A1)**. 2020.
51. Magill AJ: **Leishmania donovani - an overview | ScienceDirect Topics**. *Medicinal Plant Research in Africa* 2013.
52. **Leishmaniasis** [https://www.who.int/news-room/fact-sheets/detail/leishmaniasis]
53. taxonomy: **Taxonomy browser (Severe acute respiratory syndrome coronavirus 2)**. 2020.
54. **Early Release - Case-Fatality Risk Estimates for COVID-19 Calculated by Using a Lag Time for Fatality - Volume 26, Number 6—June 2020 - Emerging Infectious Diseases journal - CDC**. 2020.
55. **Coronavirus Age, Sex, Demographics (COVID-19) - Worldometer**. 2020.
56. **Coronavirus**. 2020.
57. @CDCgov: **Healthcare Professionals: Frequently Asked Questions and Answers | CDC**. 2020.
58. **Severe 2019-nCoV Remdesivir RCT - Full Text View - ClinicalTrials.gov**. 2020.
59. **Severe acute respiratory syndrome coronavirus 2 isolate Wuhan-Hu-1, co - Nucleotide - NCBI** [https://www.ncbi.nlm.nih.gov/pubmed/]
60. **EMBOSS: The European Molecular Biology Open Software Suite (2000)** [http://emboss.sourceforge.net]

7. Appendix: Scripts

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

7.1. BLAST Targets

7.1.1. 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
```

7.1.2. 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);
```

7.2. Process FASTA

7.2.1. 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)[3, 9]))

# 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)
}
```

7.2.2. 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
$[17, 18].blastp.txt
        done
done
```

7.2.3. extract_header.pl

This Perl script extracts statistics from BLAST reports.

```
use Switch 'fallthough';

my @lines = <STDIN>;

my $phase = 0;
my @rec = [17];
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 = [17];
                        }
                }
        }
}

foreach my $record(keys %recs){
        print "$query\t$record\n";
}
```

7.2.4. 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
```

7.2.5. 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 < $[32]r > $[16].stats
       done
done
```

7.2.6. 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
);
```

7.2.7. 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;
```

7.3. HMM targets

7.3.1. do_all_jackhmmer.pl

(replaces do_all_jackhmmer.sh)

```
#!/usr/bin/perl
# do_all_jackhmmer.pl
# Applies jackhmmer to .FASTA files in chromosome directories under <organism directory>
#

if (scalar(@ARGV) < 1) { die ("Specify organism directory\n"); }
my $org_dir = pop(@ARGV);

if ( !( -e $org_dir and -d $org_dir ) ) {
	 die "$org_dir is not a directory\n";
}

my @chrom_dirs = glob("$org_dir*");
foreach my $chrom (@chrom_dirs) {
	my @fastas = glob("$chrom/*.FASTA");
	foreach my $fasta (@fastas) {
		if ( !( -e "$fasta.summary" ) ) {
			print "jackhmmer $fasta\n";
			system("jackhmmer --domtblout $fasta.summary -o $fasta.hmm.txt $fasta
~/hmmer_targets/component_sequences.fa");
		}
	}
}
```

7.3.2. 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;
                }
        }
}
```

7.3.3. do_all_hmmer_stats.sh (deprecated)

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
```

7.3.4. do_all_hmmer_stats.pl

(Replaces do_all_hmmer_stats.sh)

```
#!/usr/bin/perl
# do_all_hmmer_stats.pl
# Gathers jackhmmer stats in chromosome directories under <organism directory>
#

if (scalar(@ARGV) < 1) { die ("Specify organism directory\n"); }
my $org_dir = pop(@ARGV);

if ( !( -e $org_dir and -d $org_dir ) ) {
         die "$org_dir is not a directory\n";
}

my $hdr = "target\ttlen\torf\tqlen\tevalue\tscore\n";
open(OUT, '>','hmm_stats.txt');
print OUT $hdr;
close(OUT);

my @chrom_dirs = glob("$org_dir*");
foreach my $chrom (@chrom_dirs) {
        my @fastas = glob("$chrom/*.FASTA");
        foreach my $fasta (@fastas) {
                print( "Extract hmm stats for $fasta\n");
                system("perl ~/genomes/extract_hmm_summary.pl $fasta.summary >> hmm_stats.txt");
        }
}
```

7.3.5. `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
);
```

7.3.6. 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;
```

7.4. Consolidated statistics analysis

7.4.1. 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 ~\Documents\RBIF120\consolidated_stats.txt CSV delimiter ' `
```

7.4.2. 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()
```

7.4.3. Score normality and kmeans analysis

```
# score normality
stats=read.csv(file='../process_plasmodium/blast_statistics.txt', sep="\t", stringsAsFactors =
FALSE)
organism="P. falciparum 3D7"

attach(stats)
qqnorm(log(score),main='Q-Q Plot for BLASTP log(scores)')
qqline(log(score),col='red',lwd=3)
detach()

norm_log_thresh=median(log(stats$score))+2*mad(log(stats$score))
norm_thresh=median(stats$score)+2*mad(stats$score)
print(paste('norm_log_thresh:',norm_log_thresh))
print(paste('norm_thresh:', norm_thresh))

norm=stats[log(stats$score) < norm_log_thresh,]
qqnorm(log(norm$score), main='Q-Q Plot for log normal scores')
qqline(log(norm$score), lwd=3, col='red')

highly_similar=stats[stats$score > norm_thresh,]

consolidated_stats=read.csv(file = "consolidated_stats.txt", sep='\t', stringsAsFactors = FALSE)
attach(consolidated_stats)
qqnorm(log(hmmer_score),main='Q-Q Plot for HMMER log(scores)')
qqline(log(hmmer_score),lwd=3,col='red')

# kmeans analysis

kh=kmeans(hmmer_score,2,nstart=25)
k_threshold = min(hmmer_score[kh$cluster==1])
plot(hmmer_score,col=kh$cluster,main='2 kmeans clusters for hmmer_score',
     sub=paste('Significance threshold:',k_threshold))
print(paste('Kmeans threshold for significance:', k_threshold))
abline(h=k_threshold,lwd=3, col='purple')
detach()
```

7.5. organism_hmmer_threshold.R

```
# organism_hmmer_threshold.R
# computes kmeans based threshold for selecting targets.

library(RPostgres)

db_name='chembl_25'
user_name = 'postgres'
host='192.168.1.180'
port=5432

conn = dbConnect(drv=RPostgres::Postgres(),
                 dbname=db_name,
                 user=user_name,
                 host=host,
                 port=port)

get_kmeans_threshold<-function(conn, tax_id, clusters=2){
  q_tax_org = paste0('SELECT distinct organism ',
                     'FROM target_dictionary ',
                     'where tax_id=',
                     tax_id)
  q_org_score = paste0(
    'select distinct score, orf, target
     from hmmer_statistics h
              join target_dictionary td
               on h.target = td.chembl_id
               join drug_mechanism dm
               ON dm.tid = td.tid
               join molecule_dictionary md
              ON dm.molregno = md.molregno
          WHERE h.tax_id ='
    , tax_id
  )

  org=dbGetQuery(conn,q_tax_org)
  org_score=dbGetQuery(conn, q_org_score)
  organism=org$organism[1]
  attach(org_score)
#  kmo=kmeans(score,2)
  kmo=kmeans(score,clusters)
  thresh=min(score[kmo$cluster==max(kmo$cluster)]) # minimum score of highest cluster

  plot(score,col=kmo$cluster, main=paste('kmeans for ',organism, ', threshold=',thresh))
  detach()
  return(thresh)
}

#dbDisconnect(conn)
```

7.6. get_unique_drugs.R

```
#get_unique_drugs(conn, tax_id, threshold)

get_unique_drugs=function (conn, tax_id, threshold){
  where_clause = paste0('WHERE score >= ', threshold, ' and h.tax_id=',tax_id,
                        ' and md.first_approval is not null')

  q_unique_drugs=paste(  'SELECT max(h.score) as score, md.pref_name '
                        ,'from hmmer_statistics h'
                        , '    join target_dictionary td'
                        , '    ON h.target = td.chembl_id'
                        , '    JOIN drug_mechanism dm'
                        , '    ON td.tid = dm.tid'
                        , '    JOIN molecule_dictionary md'
                        , '    ON dm.molregno = md.molregno'
                        , where_clause
                        , 'group by md.chembl_id, md.pref_name'
```

```
                         , 'ORDER BY md.pref_name')

  drugs = dbGetQuery(conn,q_unique_drugs)
  return(drugs)
}
```

7.7. Download P. falciparum drugs and targets

/** this is the copy directive used by the **psql** command line to download **/

```
\copy (select avg(score) as avg_score, td.tax_id as original_tax_id, td.organism as
orig_organism, md.pref_name, md.chembl_id, md.max_phase, md.first_approval from hmmer_statistics
h join target_dictionary td  on h.target = td.chembl_id join drug_mechanism dm ON dm.tid = td.tid
join molecule_dictionary md ON dm.molregno = md.molregno WHERE h.tax_id = 36329 and score > 385.3
group by td.tax_id, td.organism, md.pref_name, md.chembl_id, md.max_phase, md.first_approval
order by pref_name ) to
~/Documents/RBIF120/paralog_targets/supplements/targets/P_falciparum_hmmer_drugs.txt' CSV HEADER
delimiter '    '
```

7.8. Download P. falciparum drugs and targets, with annotations

```
\copy (select avg(score) as avg_score, td.tax_id as original_tax_id, td.organism as
orig_organism, md.pref_name, md.chembl_id,journal, year, volume, issue, first_page,
last_page,pubmed_id, doi, title, authors from hmmer_statistics h  join target_dictionary td   on
h.target = td.chembl_id  join drug_mechanism dm  ON dm.tid = td.tid  join molecule_dictionary md
ON dm.molregno = md.molregno JOIN compound_records cr ON md.molregno = cr.molregno join docs on
cr.doc_id = docs.doc_id WHERE h.tax_id = 36329 and score > 385.3 group by td.tax_id, td.organism,
md.pref_name, md.chembl_id ,journal, year, volume, issue, first_page, last_page, pubmed_id, doi,
title, authors order by md.pref_name, year, volume, issue) to
~/Documents/RBIF120/paralog_targets/supplements/targets/p_falciparum_hmmer_drugs_annotated.txt'
CSV HEADER delimiter ' '
```

7.9. P_vivax_jackhmmer_drugs.sql

```
\copy (select avg(score) as avg_score, td.tax_id as original_tax_id, td.organism as
orig_organism, md.pref_name, md.chembl_id, dm.mechanism_of_action, md.max_phase,
md.first_approval from hmmer_statistics h join target_dictionary td  on h.target = td.chembl_id
join drug_mechanism dm ON dm.tid = td.tid join molecule_dictionary md ON dm.molregno =
md.molregno WHERE h.tax_id = 5855 and score > 210.5 group by td.tax_id, td.organism,
md.pref_name, md.chembl_id, dm.mechanism_of_action, md.max_phase, md.first_approval order by
pref_name ) to ~/Documents/RBIF120/paralog_targets/supplements/targets/P_vivax_hmmer_drugs.txt'
CSV HEADER delimiter ' '
```

7.10. C_parvum_jackhmmer_drugs.sql

```
\copy (select avg(score) as avg_score, td.tax_id as original_tax_id, td.organism as
orig_organism, md.pref_name, md.chembl_id, dm.mechanism_of_action, md.max_phase,
md.first_approval from hmmer_statistics h join target_dictionary td  on h.target = td.chembl_id
join drug_mechanism dm ON dm.tid = td.tid join molecule_dictionary md ON dm.molregno =
md.molregno WHERE h.tax_id = 5807 and score >= 423.4 group by td.tax_id, td.organism,
md.pref_name, md.chembl_id, dm.mechanism_of_action, md.max_phase, md.first_approval order by
pref_name ) to ~/Documents/RBIF120/paralog_targets/supplements/targets/C_parvum_hmmer_drugs.txt'
CSV HEADER delimiter ' '
```

7.11. C_parvum_jackhmmer_drugs_annotated.sql

```
\copy (select avg(score) as avg_score, td.tax_id as original_tax_id, td.organism as
orig_organism, md.pref_name, md.chembl_id,journal, year, volume, issue, first_page,
last_page,pubmed_id, doi, title, authors from hmmer_statistics h  join target_dictionary td   on
h.target = td.chembl_id  join drug_mechanism dm  ON dm.tid = td.tid  join molecule_dictionary md
ON dm.molregno = md.molregno JOIN compound_records cr ON md.molregno = cr.molregno join docs on
cr.doc_id = docs.doc_id WHERE h.tax_id = 5807 and score >= 423.4 group by td.tax_id, td.organism,
md.pref_name, md.chembl_id ,journal, year, volume, issue, first_page, last_page, pubmed_id, doi,
title, authors order by md.pref_name, year, volume, issue) to
~/Documents/RBIF120/paralog_targets/supplements/targets/C_parvum_hmmer_drugs_annotated.txt' CSV
HEADER delimiter '            '
```

7.12. T_cruzi_hmmer_drugs.sql

```
\copy (select avg(score) as avg_score, td.tax_id as original_tax_id, td.organism as
orig_organism, md.pref_name, md.chembl_id, dm.mechanism_of_action, md.max_phase,
md.first_approval from hmmer_statistics h join target_dictionary td  on h.target = td.chembl_id
join drug_mechanism dm ON dm.tid = td.tid join molecule_dictionary md ON dm.molregno =
md.molregno WHERE h.tax_id = 5693 and score >= 172.9 group by td.tax_id, td.organism,
md.pref_name, md.chembl_id, dm.mechanism_of_action, md.max_phase, md.first_approval order by
pref_name ) to `/Documents/RBIF120/paralog_targets/supplements/targets/T_cruzi_hmmer_drugs.txt'
CSV HEADER delimiter ' '
```

7.13. T_cruzi_jackhmmer_drugs_annotated.sql

```
\copy (select avg(score) as avg_score, td.tax_id as original_tax_id, td.organism as
orig_organism, md.pref_name, md.chembl_id,journal, year, volume, issue, first_page,
last_page,pubmed_id, doi, title, authors from hmmer_statistics h  join target_dictionary td   on
h.target = td.chembl_id  join drug_mechanism dm  ON dm.tid = td.tid  join molecule_dictionary md
ON dm.molregno = md.molregno JOIN compound_records cr ON md.molregno = cr.molregno join docs on
cr.doc_id = docs.doc_id WHERE h.tax_id = 5693 and score >= 172.9 group by td.tax_id, td.organism,
md.pref_name, md.chembl_id ,journal, year, volume, issue, first_page, last_page, pubmed_id, doi,
title, authors order by md.pref_name, year, volume, issue) to
~/Documents/RBIF120/paralog_targets/supplements/targets/T_cruzi_hmmer_drugs_annotated.txt' CSV
HEADER delimiter '             '
```

7.14. Leishmania_hmmer_drugs.sql

```
\copy (select avg(score) as avg_score, td.tax_id as original_tax_id, td.organism as
orig_organism, md.pref_name, md.chembl_id, dm.mechanism_of_action, md.max_phase,
md.first_approval from hmmer_statistics h join target_dictionary td  on h.target = td.chembl_id
join drug_mechanism dm ON dm.tid = td.tid join molecule_dictionary md ON dm.molregno =
md.molregno WHERE h.tax_id = 981087 and score >= 509 group by td.tax_id, td.organism,
md.pref_name, md.chembl_id, dm.mechanism_of_action, md.max_phase, md.first_approval order by
pref_name ) to 'C:/Users/Jeremy-
satellite/Documents/RBIF120/paralog_targets/supplements/targets/Leishmania_hmmer_drugs.txt' CSV
HEADER delimiter '     '
```

7.15. Sars-CoV2_jackhmmer_drugs.sql

```
\copy (select avg(score) as avg_score, td.tax_id as original_tax_id, td.organism as
orig_organism, md.pref_name, md.chembl_id, dm.mechanism_of_action, md.max_phase,
md.first_approval from hmmer_statistics h join target_dictionary td  on h.target = td.chembl_id
join drug_mechanism dm ON dm.tid = td.tid join molecule_dictionary md ON dm.molregno =
md.molregno WHERE h.tax_id = 2697049 and score >= 4350 group by td.tax_id, td.organism,
md.pref_name, md.chembl_id, dm.mechanism_of_action, md.max_phase, md.first_approval order by
pref_name ) to ~/Documents/RBIF120/paralog_targets/supplements/targets/Sars-CoV2_hmmer_drugs.txt'
CSV HEADER delimiter '        '
```

7.16. hmmer_hist.R

```
library(RPostgres)

db_name='chembl_25'
user_name = 'postgres'
host='192.168.1.180'
port=5432

conn = dbConnect(drv=RPostgres::Postgres(),
                 dbname=db_name,
                 user=user_name,
                 host=host,
                 port=port)
organism='SARS-CoV-2'
dmax=dnorm(0,mean=0,sd=1)
sig=dmax/2.06745

q_hmmer_statistics_SARS_COV_2="
select avg(score) as score
      , td.tax_id as original_tax_id
      , td.organism as orig_organism
      , md.pref_name
      , md.chembl_id
      , dm.mechanism_of_action
      , md.max_phase
      , md.first_approval
from hmmer_statistics h
    join target_dictionary td
    on h.target = td.chembl_id
    join drug_mechanism dm
    ON dm.tid = td.tid
    join molecule_dictionary md
    ON dm.molregno = md.molregno
    WHERE h.tax_id = 2697049
group by td.tax_id
  , td.organism
  , md.pref_name
  , md.chembl_id
  , dm.mechanism_of_action
  , md.max_phase
  , md.first_approval order by pref_name"

drugs=dbGetQuery(conn,q_hmmer_statistics_SARS_COV_2)
dbDisconnect(conn)

attach(drugs)
h= hist(log(score),breaks=20,main= paste("Histogram of log(score) for",organism))
xmean=match(max(h$counts),h$counts)

detach()
```