Calliphoridae coding-sequence evolution

This notebook is for the description of the steps taken for the comparison of conding genes in the genomes of Calliphoridae species with different feeding habits.

Table of contents

Programs (and prerequisites)

Program	Version
AGAT	v1.2.1

seqkit	v2.2.0
Orthofinder	v2.5.4
Phylopypruner	v1.2.4
Mafft	v7.505
TranslatorX	v1.1
ETE3	v3.1.3
GOATOOLS	v1.4.12

Working directory

/home/blowflies/CDS_evolution

1. Genomic CDS

For this analysis, we are going to use the genomic coding sequences (CDS) of the Calliphoridae species for which genomic data is available (Table 1).

Species	Accession number / Repository	# CDS
Aldrichina grahami	http://dx.doi.org/10.5524/100673	12823
Bellardia pandia.	GCA_916048285.2	22397
Cochliomyia hominivorax		19678
Chrysomya megacephala	Data not published yet (Torres Lab)	27229
Chrysomya putoria	Data not published yet (Torres Lab)	20633
Chrysomya vomitoria	GCA_942486065.1	12917
Lucilia cuprina		13927
Lucilia eximia	Data not published yet (Torres Lab)	27610
Lucilia sericata		14704
Protocalliphora azurea		11936

2. Isoform removal

As orthologs of all species will be analysed, in order to reduce noise in the clusterization step of orthogroups, we will keep only the longest isoform per gene. This was carried out by Carolina when processing data for her Master's. Here is the link to her protocol: https://hackmd.io/@canettieri/r1lK_aBtT. The isoform removal is explained in the 2.2 topic.

Briefly, the removal of short isoforms was done using scripts from AGAT tool. It filters for longest isoforms based on information contained in annotation files as GFF/GTF. As it is explained in its user guide:

> "The script aims to filter isoforms when present. For a locus: - when all isoforms have CDS we keep the one with the longest CDS. - when some isoforms have CDS some others not, we keep the one with the longest CDS. - when none of the isoforms have CDS, we keep the one with the longest concatenated exons."

3. Extraction of orthogroups

3.1. Orthology assessment

```
mkdir /home/blowflies/CDS_evolution/1-Orthology/
cd /home/blowflies/CDS_evolution/1-Orthology/

mkdir WOI/
mkdir WOI/cds_WOI_data/

# Create a list with names of the species under analysis, i.e. Agra, Bpan, Cr
nano species-list

# Copy fasta files for cds with only the longest isoform
while read line; do rsync -c /home/blowflies/Gene_families/00-Genomic_data/0

# Copy Calliphoridae phylogenetic tree predicted
## md5 was checked
## Modify path to directory where the phylogeny was actually predicted (chec
scp -P 2205 /home/martins/CODEML/Cactus_tree.tre carol@143.107.244.181:/home/

# Edit fasta files' names to match to the species' names in the tree file, i

cd WOI/cds_WOI_data/
for file in *fasta; do mv $file ${file%%_cds_*fasta}.fasta; done
for file in *fasta
    do
        NAME=${file%%.fasta}
        sed -i "s/>/>${NAME}@/g" $file
    done


# Run Orthofinder to predict orthology relationships
## tmux a -t 3
/home/pedro/Programs/OrthoFinder/orthofinder -d -f WOI/cds_WOI_data/ -s Cactu
```

3.2 Paralog removal

3.2.1 Selection of orthogroups

We selected orthogroups containig at least 4 sequences, which is "the mimimum number required for tree inference with most tree inference programs" (https://github.com/davidemms/OrthoFinder#gene-trees-directory). Both sequences aligment and respective tree of sequendes are the input for PhyloPyPruner.

```
cd /home/blowflies/CDS_evolution/1-Orthology/WOI
mkdir Phylopypruner/
mkdir Phylopypruner/Input/
mkdir Phylopypruner/Sequence_alignment/

cd Phylopypruner/

## Copy gene trees and sequences of orthogroups
cp /home/blowflies/CDS_evolution/1-Orthology/WOI/Orthofinder/Results_Mar23/Ge
ls Input/ | wc -l # 14594

cd Input/
ls * | sed "s/_tree.txt//g" > ../orthogroups-names
cd ..
while read line; do cp /home/blowflies/CDS_evolution/1-Orthology/WOI/Orthofi
```

3.2.2 Alignment of sequences from orthogroups

```
# We did this step on Darwin
mkdir /home/martins/Mafft

# For WOI orthogroups
## Align sequences from orthogroup
mkdir /home/martins/Mafft/WOI
cd /home/martins/Mafft/WOI

rsync -c -e "ssh -p 2205" carol@143.107.244.181:/home/blowflies/CDS_evolutio

## We ran Mafft through slurm. Below are the options used and written in WOI
#
### ==================================
# !/usr/bin/bash

#SBATCH --job-name WOI ## nome que aparecerá na fila
#SBATCH --output=WOI.out ## nome do arquivo de saída; o %j é igual a jobID;
#SBATCH --ntasks=1 ## número de tarefas (análises) a serem executadas
#SBATCH --cpus-per-task=20 ## o número de threads alocados para cada tarefa
#SBATCH --partition=long ## as partições a serem executadas (separadas por v
#SBATCH --error=WOI.err

cd /home/martins/Mafft/WOI/

for file in *fa; do mafft --maxiterate 1000 --localpair --thread 20 $file > 
### ==============================
#
## Run code in slurm
sbatch WOI.slurm
```

3.2.3 Running PhyloPyPruner

```
# Transfer aligned sequences from Darwin server to Rosalind server's input d
## The next two command lines were run on Darwin
cd /home/martins/Mafft/WOI
rsync -c -e "ssh -p 2205" *ali carol@143.107.244.181:/home/blowflies/CDS_evo

# Rename tree and sequences files to match each other except by their extens
cd /home/blowflies/CDS_evolution/1-Orthology/WOI/Phylopypruner/Input/

for file in *ali; do mv $file ${file%%.fa.ali}.fasta; done
for file in *txt; do mv $file ${file%%_tree.txt}.tree; done

# Rename taxa in trees to match headers in fasta
while read line
do
  for file in *tree
  do
    sed -i "s/${line}_${line}@/${line}@/g" $file
  done
done < /home/blowflies/CDS_evolution/1-Orthology/species-list

# Run phylopypruner
cd /home/blowflies/CDS_evolution/1-Orthology/WOI/Phylopypruner

/home/vanessa/anaconda3/bin/phylopypruner --dir /home/blowflies/CDS_evolutio
```

3.2.4 Results

```
Alignment statistics:
  Description                          Input       Output
  No. of alignments                    14594        11322
  No. of sequences                    146330        98395
  No. of OTUs                             10           10
  Avg no. of sequences / alignment        10            8
  Avg no. of OTUs / alignment              7            8
  Avg sequence length (ungapped)        1664         1790
  Shortest sequence (ungapped)            54           54
  Longest sequence (ungapped)          72516        72516
  % missing data                       32.60        23.80
  Concatenated alignment length     30646102     24314468
```

- Number of 1:1 orthogroups from Orthofinder = ?
- Number of orthogroups with at least 4 species from Orthofinder (our input): 14594
- Number of orthogroups without paralogs from Phylopypruner: 11322
- Number of input sequences = 146,330
- Number of filtered sequences = 98,395
- Percentage of filtered sequences in relation to input sequences = 67,24%

4. Filtering and alignment

4.1. Filtering and fixing

We'll use all orthogroups that include Chom and/or Pazu (our target species)

```
cd /home/blowflies/CDS_evolution/
mkdir 2-TranslatorX
mkdir 2-TranslatorX/Filt_Orthogroups
cd 2-TranslatorX/Filt_Orthogroups

# Copy orthogroups with Chom and/or Pazu (path appended to orthogroups names)
grep "Chom" /home/blowflies/CDS_evolution/1-Orthology/WOI/Phylopypruner/Outpu
grep "Pazu" /home/blowflies/CDS_evolution/1-Orthology/WOI/Phylopypruner/Outpu

cat Chom_orthogroups Pazu_orthogroups | sort -u > All_orthogroups

wc -l *
# Chom_orthogroups = 9462
# Pazu_orthogroups = 8926
# All_orthogroups = 10199

while read line; do rsync -c $line .; done < All_orthogroups

# Remove the gaps to un-align the sequences
sed -i 's/-//g' *fasta

# Rename sequence headers to use in codeml analysis afterwards
sed -i 's/@.*//g' *fasta
```

4.2. TranslatorX

```
cd /home/blowflies/CDS_evolution/2-TranslatorX/Filt_Orthogroups
rsync -c /home/pedro/Calliphoridae_pedro_gi/2-Coding_Seq_Evol/1-Fasta_cluste

# Change usr path to MAFFT to Bruno's path to MAFFT
sed -i 's/\/usr\/bin\/mafft/\/home\/bruno\/anaconda3\/bin\/mafft/g' translat

# Make sure that all programs required by TranslatorX (e.g. Gblocks) are in

# Testing run of TranslatorX
perl translatorx_vLocal.pl -i OG0000013_pruned_1.fasta -o TESTE -p F -g -b5=a

# Run TranslatorX (tmux a -t 3)
for i in *.fasta; do perl translatorx_vLocal.pl -i ${i} -o ${i}_translatorX
```

Organizing the results

```
cd /home/blowflies/CDS_evolution/2-TranslatorX
mkdir Chom_align Pazu_align

sed -i 's/\/home\/blowflies\/CDS_evolution\/1-Orthology\/WOI\/Phylopypruner\/

sed -i 's/\/home\/blowflies\/CDS_evolution\/1-Orthology\/WOI\/Phylopypruner\/


sed -i 's/.fasta/.fasta_translatorX.nt_cleanali.fasta/' Filt_Orthogroups/Pazu

sed -i 's/.fasta/.fasta_translatorX.nt_cleanali.fasta/' Filt_Orthogroups/Chom


while read line; do rsync -c Filt_Orthogroups/$line Pazu_align; done < Filt_O

while read line; do rsync -c Filt_Orthogroups/$line Chom_align; done < Filt_O
```

5. Test of evolutionary models

5.1. Prepare data for CodeML

We are using CODEML implemented within ETE3. This step was carried out on Darwin

```
mkdir /home/martins/CodeML
mkdir /home/martins/CodeML/Chom /home/martins/CodeML/Pazu
cd /home/martins/CodeML

# Tree
## Modify path to directory where the phylogeny was actually predicted (chec
rsync -c CODEML/Cactus_tree.tre .

# Aligned sequences
rsync -c -e "ssh -p 2205" carol@143.107.244.181:/home/blowflies/CDS_evolutio
rsync -c -e "ssh -p 2205" carol@143.107.244.181:/home/blowflies/CDS_evolutio

# Scripts
nano CodeML/Chom/ete3_wrapper_Chom.py
nano CodeML/Pazu/ete3_wrapper_Pazu.py
```

See scripts in Topic 9.

5.2 Running CodeML

We ran CodeML using the following commands in slurm.

For Chom:

```
#!/usr/bin/bash
#SBATCH --job-name Chom_codeml ## nome que aparecerá na fila
#SBATCH --output Chom_codeml.out ## nome do arquivo de saída; o %j é igual a
#SBATCH --ntasks=1 ## número de tarefas (análises) a serem executadas
#SBATCH --cpus-per-task=30 ## o número de threads alocados para cada tarefa
#SBATCH --partition=long ## as partições a serem executadas (separadas por v
#SBATCH --error=Chom_codeml.err

cd /home/martins/CodeML/Chom
python3 ete3_wrapper_Chom.py -p /home/martins/CodeML/Chom/Aln -e .fasta -t /
```

For Pazu:

```
#!/usr/bin/bash
#SBATCH --job-name Pazu_codeml ## nome que aparecerá na fila
#SBATCH --output Pazu_codeml.out ## nome do arquivo de saída; o %j é igual a
#SBATCH --ntasks=1 ## número de tarefas (análises) a serem executadas
#SBATCH --cpus-per-task=30 ## o número de threads alocados para cada tarefa
#SBATCH --partition=long ## as partições a serem executadas (separadas por v
#SBATCH --error=Pazu_codeml.err

cd /home/martins/CodeML/Pazu
python3 ete3_wrapper_Pazu.py -p /home/martins/CodeML/Pazu/Aln -e .fasta -t /
```

sbatch Chom_codeml.slurm
sbatch Pazu_codeml.slurm

5.3 Filtering

We organized the results into tables with the important information using another script.

```
cd /home/martins/CodeML

rsync -c -e "ssh -p 2205" carol@143.107.244.181:/home/pedro/Calliphoridae_pe

python3 ete3_parser.py -p Chom -e .log -s Chom
python3 ete3_parser.py -p Pazu -e .log -s Pazu
```

See script in Topic 9.

After that, we corrected p-values using FDR in R and selected genes for which the bestfit model was MO, b_neut and b_free.

```
mv Chom/Chom_branch_raw_table.txt .
mv Pazu/Pazu_branch_raw_table.txt .

Rscript codeml_table_filter.R

rsync -c -e "ssh -p 2205" -a /home/martins/CodeML/ carol@143.107.244.181:/ho
```

See script in Topic 9.

We have 6 tables at the end (3 for each species):

Table	Content
[Chom/Pazu]_bestfit_M0.txt	Genes that did not reject the first null hypothesis (M0)
[Chom/Pazu]_bestfit_neut.txt	Genes that rejected the first null hypothesis (M0), but did not reject the second one (b_neut)
[Chom/Pazu]_bestfit_free.txt	Genes that rejected both null hypotheses and favoured the alternative one (free)

The following genes do not have significant omega because their aligned sequence (input for CodeML) were empty:

- Chom: "OG0000190_pruned_Chom", "OG0003185_pruned_1_Chom", "OG0003142_pruned_1_Chom"
- Pazu: "OG0000190_pruned_Pazu", "OG0001498_pruned_1_Pazu"

Thus, at the end, we have 9459 and 8924 genes with significant omegas for Chom and Pazu, respectively.

6. GO analysis

```
mkdir /home/blowflies/CDS_evolution/4-GO_Analysis
cd /home/blowflies/CDS_evolution/4-GO_Analysis
```

6.1 Annotation of GOs

We annotated only the genes from Chom and Pazu. We did this step on Darwin server using EnTAP software. Note that the headers are named as `$SPECIES@GENE_NAME`

```
mkdir /home/martins/CDS_EnTAP_Pedro
cd /home/martins/CDS_EnTAP_Pedro

mkdir Chom Pazu

## Copy files
rsync -c -e "ssh -p 2205" carol@143.107.244.181:/home/blowflies/CDS_evolutio
rsync -c -e "ssh -p 2205" carol@143.107.244.181:/home/blowflies/CDS_evolutio

# Run EnTAP with cds in queue using slurm

cd Chom # slurm file: Chom.slurm
EnTAP --runN -i /home/martins/CDS_EnTAP_Pedro/Transcripts/Chom.fasta -d /home

cd Pazu # slurm file: Pazu.slurm
EnTAP --runN -i /home/martins/CDS_EnTAP_Pedro/Transcripts/Pazu.fasta -d /home

# Transfer files to rosalind

# In Rosalind previously
## mkdir /home/blowflies/CDS_evolution/4-GO_Analysis/EnTAP
rsync -c -e "ssh -p 2205" /home/martins/CDS_EnTAP_Pedro/Chom/entap_outfiles/
rsync -c -e "ssh -p 2205" /home/martins/CDS_EnTAP_Pedro/Pazu/entap_outfiles/
```

6.2 Comparison between the omegas associated to GOs and the average omega of the genome

The analysis onwards were run on Rosalind server

```
mkdir /home/blowflies/CDS_evolution/4-GO_Analysis/2-Distribution_omega
cd /home/blowflies/CDS_evolution/4-GO_Analysis/2-Distribution_omega

# Write Chom and Pazu in a list
nano species-list.txt
```

Create a table with genes and their omegas estimated by their bestfit model

```
while read sp
do
    sed '1d' /home/blowflies/CDS_evolution/3-CodeML/${sp}_bestfit_M0.txt | cu
    sed '1d' /home/blowflies/CDS_evolution/3-CodeML/${sp}_bestfit_free.txt |
    sed '1d' /home/blowflies/CDS_evolution/3-CodeML/${sp}_bestfit_neut.txt |
    cat ${sp}*txt | sed '1i\Gene\tOmega\tBestfit_Model' > ${sp}_omegas_bestfi
    rm ${sp}_bestfit*txt
done < species-list.txt

# Check
wc -l Chom* # 9460 (header included)
wc -l Pazu* # 8925 (header included)
```

Filter genes with omega from EnTAP table

```
# Get ID of orthogroups that were tested in CodeML
while read sp
do
    sed '1d' ${sp}_omegas_bestfit.txt | cut -f1 | sed "s/_${sp}//g" > ${sp}_
done < species-list.txt

# Get IDs of genes that were tested in CodeML
cd /home/blowflies/CDS_evolution/4-GO_Analysis/1-EnTAP/All_orthogroups

while read sp
do
    while read id
    do
        cat ${id}.fasta | grep "${sp}" | sed "s/>${sp}@//g" >> ../../2-Distri
    done < /home/blowflies/CDS_evolution/4-GO_Analysis/2-Distribution_omega/$
done < /home/blowflies/CDS_evolution/4-GO_Analysis/2-Distribution_omega/speci

# Check
wc -l /home/blowflies/CDS_evolution/4-GO_Analysis/2-Distribution_omega/*genes

# Filter EnTAP table for those genes
cd /home/blowflies/CDS_evolution/4-GO_Analysis/EnTAP

while read sp
do
    grep -f /home/blowflies/CDS_evolution/4-GO_Analysis/2-Distribution_omega/
    cut -f1 ${sp}.tmp | grep -v -f /home/blowflies/CDS_evolution/4-GO_Analysi
    grep -v -f ${sp}.del ${sp}.tmp | sort > /home/blowflies/CDS_evolution/4-G
    rm ${sp}.tmp ${sp}.del
done < /home/blowflies/CDS_evolution/4-GO_Analysis/2-Distribution_omega/speci

# Check
wc -l /home/blowflies/CDS_evolution/4-GO_Analysis/2-Distribution_omega/*genes

# Create tables with infos
cd /home/blowflies/CDS_evolution/4-GO_Analysis/2-Distribution_omega

while read sp
do
    paste ${sp}_omegas_genes.id ${sp}_omegas_bestfit.txt | sort | paste - ${s
    cut -f1,2,3,4,17,38,39,40 ${sp}_all-infos.tsv > ${sp}_short-infos.tsv
done < species-list.txt
```

Create a table with GOs and the omega of the genes associated to them

```
## Create a list with GO categories (Write BP, CC and MF)
nano go-categories.txt

## Get the GO terms present in the dataset per category
while read sp
do
    while read go
    do
        if [[ $go == "BP" ]]
        then
            sed '1d' ${sp}_short-infos.tsv | cut -f6 | sed -e $'s/,/\\\n/g'
        elif [[ $go == "CC" ]]
        then
            sed '1d' ${sp}_short-infos.tsv | cut -f7 | sed -e $'s/,/\\\n/g'
        elif [[ $go == "MF" ]]
        then
            sed '1d' ${sp}_short-infos.tsv | cut -f8 | sed -e $'s/,/\\\n/g'
        fi
    done < go-categories.txt
done < species-list.txt

## Get the number of genes that each GO is associated to
N=10; i=0; while read go; do ((i=i%N)); ((i++==0)) && wait; (while read id; d

N=10; i=0; while read go; do ((i=i%N)); ((i++==0)) && wait; (while read id; d

## Create a table with GOs and the omega of the genes associated to them
N=10; i=0; while read go; do ((i=i%N)); ((i++==0)) && wait; (mkdir Chom_${go

N=10; i=0; while read go; do ((i=i%N)); ((i++==0)) && wait; (mkdir Pazu_${go

# Bind tables into one
N=5; i=0; while read sp; do ((i=i%N)); ((i++==0)) && wait; (while read go; do

# Check
for file in *amount; do echo $file; sort -k2n $file | tail -n1; done
# Chom_BP.amount
# GO:0008150      7506
# Chom_CC.amount
# GO:0005575      6254
# Chom_MF.amount
# GO:0003674      7344
# Pazu_BP.amount
# GO:0008150      7169
# Pazu_CC.amount
# GO:0005575      5973
# Pazu_MF.amount
# GO:0003674      6993
wc -l *go-omegas.tsv # headers included
# 7507 Chom_BP_go-omegas.tsv
# 6255 Chom_CC_go-omegas.tsv
# 7345 Chom_MF_go-omegas.tsv
# 7170 Pazu_BP_go-omegas.tsv
# 5974 Pazu_CC_go-omegas.tsv
# 6994 Pazu_MF_go-omegas.tsv
```

Perform Mann-Whitney U test for GOs that have at least X omegas of genes associated to them and, among GOs whose median is significantly different from genomes's median, it select the top 15 GOs that have the highest and the lowest medians compared to the genome's median (top 30 GOs at the end).

> The user can input multiple values of X to see how many siginificant GOs they get at the end
> A graph is created to visualize the amount of siginificant GOs versus the X values specified ($_{n}um_{s}ignificant_{p}lot.png)Theoutputtablehelpstheusertovisualizethetop30GOswhosemedianarethehighestandthelowe$ _top15_highest/lowest_headers.tsv)

```
N=5; i=0; while read sp; do ((i=i%N)); ((i++==0)) && wait; (while read go; do

# Just a command to verify how many GOs have at least a X value of genes ass
awk -F "\t" '{ if ($2 > 500) print $1}' Chom_BP.amount | wc -l
```

> When X = 30, a higher number of GOs were returned as the top 15 highest median. The results indicates that GOs associated to more genes, that is, the more generalist a GO are, the more close its median gets to genome's median. As we want to get more detailed information about the genome evolution of our species, we will proceed the analysis using X = 30.

Perform Mann-Whitney U test for GOs that have at least 30 omegas of genes associated to them, select the top 30 genes (see previously) and creates a table that compiles info necessary to plot a multiple boxplot graph (see Figure 5 of "Genomic analysis of a livestock pest (Scott et al., 2020)")

> Output written in ($_top30_GOs.tsv)

```
N=5; i=0; ((i=i%N)); ((i++==0)) && wait; while read sp; do Rscript MW_test_t
```

Before going to the following steps, we had to do some manual changes to the tables (if not specified otherwise, everything was done on Excel):

- GO term tables (*_top30_GOs.tsv):

1. The GO ids were written with ".", so we changed it o ":", because that's how it is recognized by GO-related tools (ex: "GO.0050907" to "GO:0050907");
2. We used a script made by ChatGPT to obtain the GO term names form the ids (see script 8.7), which returns a two-column table with the ids and the names;
3. We manually inserted the name column into the original table;
4. We removed the header.

- Whole genome table (*_short-infos.tsv):

1. We removed everything but the column with all omegas.

These were the inputs used in the R script we used to generate the final figures (script 8.8)

See scripts in Topic 8

We did everything locally (because of RStudio) and uploaded everything back to Rosalind afterwards: `/home/blowflies/CDS_evolution/4-GO_Analysis/3-GO_figure`

6.3 Enrichment analysis

Organization

We performed the enrichment analysis using GOATools. This software requires three inputs:

1. A file with a list of all genes in a given genome (population);
2. A file with all genes of interest (study - in our case, the genes whose best fit was the b_free model);
3. An association file, which relates all GO-annotated genes to their GOs.

```
cd /home/blowflies/CDS_evolution/4-GO_Analysis/4-Enrichment

# Population files
/home/blowflies/CDS_evolution/1-Orthology/WOI/cds_WOI_data

grep ">" /home/blowflies/CDS_evolution/1-Orthology/WOI/cds_WOI_data/Pazu.fast
grep ">" /home/blowflies/CDS_evolution/1-Orthology/WOI/cds_WOI_data/Chom.fast

# Study files
## These are a bit more challenging. Our codeml outputs contain the orthogrou

## 1. Get the names of the orthogroups we need, and their omega values
cut -f1,5 /home/blowflies/CDS_evolution/3-CodeML/Chom_bestfit_free.txt | sed

cut -f1,5 /home/blowflies/CDS_evolution/3-CodeML/Pazu_bestfit_free.txt | sed

## 2. Separate the orthogroups into the ones with above and below average (c

# Chom - mean = 0.073
while IFS=$'\t' read -r word value; do
  if (( $(echo "$value >= 0.073" | bc -l) )); then
    echo -e "$word" >> Chom_above.txt
  elif (( $(echo "$value < 0.073" | bc -l) )); then
    echo -e "$word" >> Chom_below.txt
  fi
done < Chom_orthogroups.txt

# Pazu - mean = 0.082
while IFS=$'\t' read -r word value; do
  if (( $(echo "$value >= 0.082" | bc -l) )); then
    echo -e "$word" >> Pazu_above.txt
  elif (( $(echo "$value < 0.082" | bc -l) )); then
    echo -e "$word" >> Pazu_below.txt
  fi
done < Pazu_orthogroups.txt

## 3. Extract the names of the genes of each species from the orthogroups (t
cp /home/pedro/Scripts/get_ids_from_cluster_list.py .

python3 get_ids_from_cluster_list.py -i Chom_above.txt -s Chom -c /home/blow
mv Chom_table_from_clusters.txt Chom_table_from_clusters_above.txt
cut -f2 Chom_table_from_clusters_above.txt | sed 's/>Chom@//' > Chom_above_s

python3 get_ids_from_cluster_list.py -i Chom_below.txt -s Chom -c /home/blow
mv Chom_table_from_clusters.txt Chom_table_from_clusters_below.txt
cut -f2 Chom_table_from_clusters_below.txt | sed 's/>Chom@//' > Chom_below_s

python3 get_ids_from_cluster_list.py -i Pazu_above.txt -s Pazu -c /home/blow
mv Pazu_table_from_clusters.txt Pazu_table_from_clusters_above.txt
cut -f2 Pazu_table_from_clusters_above.txt | sed 's/>Pazu@//' > Pazu_above_s

python3 get_ids_from_cluster_list.py -i Pazu_below.txt -s Pazu -c /home/blow
mv Pazu_table_from_clusters.txt Pazu_table_from_clusters_below.txt
cut -f2 Pazu_table_from_clusters_below.txt | sed 's/>Pazu@//' > Pazu_below_s

# Association files (preliminary - We need to do some more fixing on Excel!)
cut -f1,34,35,36 /home/blowflies/CDS_evolution/4-GO_Analysis/EnTAP/Chom_entap

cut -f1,34,35,36 /home/blowflies/CDS_evolution/4-GO_Analysis/EnTAP/Pazu_entap

# Changes in Excel:
# 1. Substitute "NA" for nothing
# 2. Concatenate the three GO columns in another one
# 3. Substitute ",GO" for ";GO"
# 4. Substitute "," for nothing (remove the final comma)
# 5. Reorder the table (Z -> A), so it's easier to remove the genes with no a
# 6. For some reason, when we open the files in a txt reader, quotes ("") app
# 7. Rename the files: Chom_id2GO.txt and Pazu_id2GO.txt (these are in the En
```

Just a little organization to make our lives easier afterwards

```
mkdir /home/blowflies/CDS_evolution/4-GO_Analysis/Enrichment/1-Initial_files
mv *txt 1-Initial_files

# Now, let's put the non-necessary files elsewhere, and leave only the neces
cd 1-Initial_files
mkdir TMP
mv mv Chom_above.txt Chom_below.txt *table* *orthogroups* *tmp* TMP/

# Done! Now we have one population file, two study files and one association
```

Analysis

We started by creating all necessary files on Rosalind, and then moved on to analysing them on Darwin. So we had to install the software and move the files.

```
# On Darwin
mkdir /home/martins/GO_enrichment
cd /home/martins/GO_enrichment

# Installation (July, 13 - 2024 -> Reference for the GO obo file)
wget http://current.geneontology.org/ontology/go-basic.obo
pip install goatools

# Bring the files
scp -P 2205 pedro@143.107.244.181:/home/blowflies/CDS_evolution/4-GO_Analysis

# Analysis
find_enrichment.py Chom_above_study.txt Chom_pop.txt Chom_id2GO.txt --obsolet
find_enrichment.py Chom_below_study.txt Chom_pop.txt Chom_id2GO.txt --obsolet
find_enrichment.py Pazu_above_study.txt Pazu_pop.txt Pazu_id2GO.txt --obsolet
find_enrichment.py Pazu_below_study.txt Pazu_pop.txt Pazu_id2GO.txt --obsolet

# Note: the go-basic.obo file is in the same directory as all the files, so
```

Bringing the raw results back to Rosalind

```
scp -P 2205 *enriched* pedro@143.107.244.181:/home/blowflies/CDS_evolution/4-
```

Filtering and figures

We downloaded the GOATOOLS outputs and hadled them using R (GO_enrichment_filter_plots.R - see 8.10). Afterwards, we brought everything back to Rosalind.

```
scp -P 2205 * pedro@143.107.244.181:/home/blowflies/CDS_evolution/4-GO_Analys
```

7. Extra analysis - Isoform removal by AGAT improves results

When we first ran this pipeline, we tested the analysis with two datasets: one containing only the longest isoform for each gene and another containing all isoforms, for all species. By doing this, we aimed to evaluate if AGAT's script to remove short isoforms would increase the number of orthogroups clustered by Phylopypruner in comparison to the number the latter can cluster with all-isoforms-dataset using its option '–mask longest'. The results of Phylopypruner for each dataset are shown below:

For dataset containig only the longest isoform (AGAT's script used)

```
Alignment statistics:
  Description                          Input       Output
  No. of alignments                    14630        11347
  No. of sequences                    145793        98661
  No. of OTUs                             10           10
  Avg no. of sequences / alignment         9            8
  Avg no. of OTUs / alignment              7            8
  Avg sequence length (ungapped)        1658         1794
  Shortest sequence (ungapped)            54           57
  Longest sequence (ungapped)          72588        72588
  % missing data                       32.80        23.90
  Concatenated alignment length     30767442     24385049
```

- Number of 1:1 orthogroups from Orthofinder = 2336
- Number of orthogroups with at least 4 species from Orthofinder (our input): 14630
- Number of orthogroups without paralogs from Phylopypruner: 11347
- Number of filtered sequences = 98,661
- Percentage of filtered sequences in relation to input sequences = 67.7%

For dataset containing all isoforms (Phylopypruner's masking option used)

```
Alignment statistics:
  Description                        Input       Output
  No. of alignments                  15515        11234
  No. of sequences                  182957        95704
  No. of OTUs                           10           10
  Avg no. of sequences / alignment      11            8
  Avg no. of OTUs / alignment            7            8
  Avg sequence length (ungapped)      1811         1745
  Shortest sequence (ungapped)          54           57
  Longest sequence (ungapped)        72588        70389
  % missing data                     34.80        25.50
  Concatenated alignment length   32314878     23800455
```

- Number of 1:1 orthogroups from Orthofinder = 752
- Number of orthogroups with at least 4 species from Orthofinder (our input): 15515
- Number of orthogroups without paralogs from Phylopypruner: 11234
- Number of filtered sequences = 95,704
- Percentage of filtered sequences in relation to input sequences = 52,3%

Looking at the results, we can say that using AGAT's script provided 110 more orthogroups and almost 3,000 more sequences to the analysis, which is advantageous. In this sense, we can recommend using this tool instead of just Phylopypruner's masking option.

Because Vanessa's detected a problem with the annotation of Cmeg, Lexi and Cput, we had to run all the pipeline again. As we had already run this comparison, it would not make sense to rerun it. We decided to directly use AGAT.
In order to not confound other users, we deleted the files regarding the results shown. However, we preserved their statistics in this section in order to convince new users to use AGAT.

8. Scripts

8.1 "ete3_wrapper_.py": runs CodeML (Topic 5.1)

The script below is for Pazu, but works the same for Chom if you substitute Pazu for Chom:

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
# ete3_wrapper.py

import sys
import os
import getopt
import glob
from ete3 import EvolTree
#-------------------------------------------------------------------------------
def main():
    pwd, end, t = arg()

    if pwd == None or end == None or t == None:
        usage ()

    clusters = glob.glob(f"{pwd}/*{end}")

    for cluster in clusters:
        # Get the name for the output directory
        name = cluster.split("/")[-1].split(".")[0]

        # What species are represented in the fasta file?
        species = []

        fasta = open(cluster, 'r', encoding='utf-8')

        for line in fasta:
            if ">" in line:
                sp = line.split(">")[1].strip()
                species += [sp]

        fasta.close()


        # Read the tree and prune it leaving just the species that have seque
        tree = EvolTree(t)
        tree.prune(species)

        tree_txt = tree.write()

        # New tree
        new_tree = open('temp_tree.nw','w')
        new_tree.write(tree_txt)
        new_tree.close()

        final_tree = "temp_tree.nw"


        # Run CODEML with ete3
        cmd = ["ete3 evol -t",final_tree, "--alg ", cluster, "--mark Pazu", "
                   "--models b_free b_neut M0 --tests b_free,b_neut b_free,M(
                   f"-o {name}_Pazu/ >{name}_Pazu.log 2>{name}_Pazu.err"]
        print (" ".join(cmd))
        os.system(" ".join(cmd))

#-------------------------------------------------------------------------------
def arg():
    argv = sys.argv[1:]
    try:
        opts, args = getopt.getopt(argv, 'p:e:t:')

    except:
        usage()

    arg_1, arg_2, arg_3 = None, None, None

    for opt, arg in opts:
        if opt in ['-p']:
            arg_1 = arg
        elif opt in ['-e']:
            arg_2 = arg
        elif opt in ['-t']:
            arg_3 = arg

    return arg_1, arg_2, arg_3
#-------------------------------------------------------------------------------
def usage():
    sys.exit('Usage:\n'
           '\t-p: fasta clusters directory pathway\n'
           '\t-e: fasta clusters name ending\n'
           '\t-t: tree file (newick format)')
#-------------------------------------------------------------------------------
if __name__ == '__main__':
    main()
```

8.2 "ete3_parser.py": organizes important results from CodeML into tables (Topic 5.3)

For reference of ete3_parser.py:

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

'''
PROGRAM: ete_codeml_parser.py

DESCRIPTION: Extracts CodeML results from many ete3 result files using regex
'''
import re
import glob
import sys
import getopt
#------------------------------------------------------------------------------
def main():
    pwd, end, spp = arg()
    if pwd == None or end == None or spp == None:
        usage()

    results = glob.glob(f'{pwd}/*{end}')

    species = spp.split(',')

    for s in species:
        # two tables per species
        table_branch = open(f'{pwd}/{s}_branch_raw_table.txt','w')
        table_branch.write('Gene\tAverage_Omega_M0\tOmega_foreground_neut\t'
                           'Omega_background_neut\tOmega_foreground_free\t'
                           'Omega_background_free\tpv_free_M0\tpv_free_neut\

        for r in results:
            if s in r: # just the results for that species
                file = open(r,'r')
                gene = r.split('/')[-1].split('.')[0]

                x = False # variable to check where we are in the file (beca
                          # there two lines that are equal to each other, bu
                          # need separate information -> this is for omega f
                          # and background values)

                pv_free_M0 = 'NA'
                pv_free_neut = 'NA'
                omega_fore_neut = 'NA'
                omega_back_neut = 'NA'
                omega_fore_free = 'NA'
                omega_back_free = 'NA'
                av_omega_M0 = 'NA'

                for line in file:
                    # pvalue for branch test free x M0
                    pv_free_M0_search = re.search(r'M0.*? b_free.*?\| (.*)',
                    if pv_free_M0_search:
                        pv_free_M0 = pv_free_M0_search.group(1)
                        pv_free_M0 = pv_free_M0.replace('*','')
                        continue

                    # pvalue for branch test free x neut
                    pv_free_neut_search = re.search(r'b_neut.*? b_free.*?\|
                    if pv_free_neut_search:
                        pv_free_neut = pv_free_neut_search.group(1)
                        pv_free_neut = pv_free_neut.replace('*','')
                        continue

                    # this line here separates the file
                    if '- Model b_neut' in line:
                        x = True
                        continue

                    # get omega values, but we differentiate the line by whe
                    # x is True or False
                    omega_fore_search = re.search(r'#1  => (.*)', line)
                    if omega_fore_search and not x:
                        omega_fore_free = omega_fore_search.group(1)
                        omega_fore_free = omega_fore_free.replace(' ','')
                        continue
                    elif omega_fore_search and x:
                        omega_fore_neut = omega_fore_search.group(1)
                        omega_fore_neut = omega_fore_neut.replace(' ','')
                        continue

                    omega_back_search = re.search(r'background  =>   (.*)',
                    if omega_back_search and not x:
                        omega_back_free = omega_back_search.group(1)
                        omega_back_free = omega_back_free.replace(' ','')
                        continue
                    elif omega_back_search and x:
                        omega_back_neut = omega_back_search.group(1)
                        omega_back_neut = omega_back_neut.replace(' ','')
                        continue
```

```
                    av_omega_search = re.search(r'Average omega for all tree
                    if av_omega_search:
                        av_omega_M0 = av_omega_search.group(1)
                        av_omega_M0 = av_omega_M0.replace(' ','')
                        continue

                table_branch.write(f'{gene}\t{av_omega_M0}\t{omega_fore_neut
                                   f'{omega_back_neut}\t{omega_fore_free}\t'
                                   f'{omega_back_free}\t{pv_free_M0}\t'
                                   f'{pv_free_neut}\n')

        table_branch.close()


#-------------------------------------------------------------------------
def arg():
    argv = sys.argv[1:]
    try:
        opts,args = getopt.getopt(argv, 'p:e:s:')
    except:
        usage()

    arg_1, arg_2, arg_3 = None, None, None

    for opt, arg in opts:
        if opt in ['-p']:
            arg_1 = arg
        elif opt in ['-e']:
            arg_2 = arg
        elif opt in ['-s']:
            arg_3 = arg

    return arg_1, arg_2, arg_3
#-------------------------------------------------------------------------
def usage():
    sys.exit('Usage:\n'
           '\t-p: result files directory pathway\n'
           '\t-e: result files ending\n'
           '\t-s: species names in the files, separated by commas')
#-------------------------------------------------------------------------
if __name__ == '__main__':
    main()
```

8.3 "codeml_table_filter.R": corrects p-values using FDR and compile genes for which the bestfit model was M0, bfree or bfree (Topic 5.3)

For reference of data manipulation in R for Chom (codeml_table_filter.R):

```r
setwd("/home/blowflies/CDS_evolution/3-CodeML")

#### Chom branch models ####

# Read table
chom_branch <- read.table("Chom_branch_raw_table.txt", header = T)

# Correct pvalue and get significant results (null = M0)
# Does the two omega model (b_free) explain the data better than the one ome
# model (M0)?
pv_M0 <- chom_branch$pv_free_M0
pv_M0_adj <- p.adjust(pv_M0, method = "fdr")

chom_branch_M0_corrected <- cbind(chom_branch[,1:7], pv_M0_adj, chom_branch[,
colnames(chom_branch_M0_corrected)[9] <- colnames(chom_branch)[8]

chom_branch_M0_significant <- subset(chom_branch_M0_corrected,
                                     chom_branch_M0_corrected$pv_M0_adj<0.05)

Chom_bestfit_M0 <- subset(chom_branch_M0_corrected,
                          chom_branch_M0_corrected$pv_M0_adj>0.05)

write.table(Chom_bestfit_M0, "Chom_bestfit_M0.txt",
            sep = "\t", row.names = F, quote = F)

# Correct pvalue and get significant results (null = b_neut)
# Does the free omega in foreground branch model (b_free) explain the data
# better than the foreground omega = 1 model (b_neut)?
# We only consider the genes that are significant in the first comparison
# (if M0 were the best fit, there would be no reason to run this second test)

pv_neut <- chom_branch_M0_significant$pv_free_neut
pv_neut_adj <- p.adjust(pv_neut, method = "fdr")

chom_branch_neut_corrected <- cbind(chom_branch_M0_significant, pv_neut_adj)

chom_branch_neut_significant <- subset(chom_branch_neut_corrected,
                                       chom_branch_neut_corrected$pv_neut_adj<0

write.table(chom_branch_neut_significant, "Chom_bestfit_free.txt",
            sep = "\t", row.names = F, quote = F)

Chom_bestfit_neut <- subset(chom_branch_neut_corrected,
                            chom_branch_neut_corrected$pv_neut_adj>0.05)

write.table(Chom_bestfit_neut, "Chom_bestfit_neut.txt",
            sep = "\t", row.names = F, quote = F)

# Check if any gene does not have a significant omega

initial_ids <- chom_branch$Gene
significant_ids_M0 <- chom_branch_M0_significant$Gene
bestfit_M0_ids <- Chom_bestfit_M0$Gene
significant_neut_ids <- chom_branch_neut_significant$Gene
bestfit_neut_ids <- Chom_bestfit_neut$Gene

length(initial_ids)
length(bestfit_M0_ids)
length(bestfit_neut_ids)
length(significant_neut_ids)

all_ids_combined <- unique(c(Chom_bestfit_M0$Gene, chom_branch_neut_signific

missing_ids <- setdiff(initial_ids, all_ids_combined)
print(missing_ids)
```

8.4 "bind-tables_go-omegas.R": binds tables of GOs and the omegas of their respective associated gene (Topic 6.2) (by ChatGPT)

```r
#!/usr/bin/env Rscript

# Usage: Rscript bind-tables_go-omegas.R {/path/to/working/directory} {species} {G

# Obtain the command line arguments
args <- commandArgs(trailingOnly = TRUE)
if (length(args) < 3) {
    stop("Usage: script.R <path> <species> <category>")
}
work_dir <- args[1]
species <- args[2]
category <- args[3]

# Define working directory
setwd(work_dir)
library(dplyr)
```

```
    # Define paths to the temporary directories based on species and category
    dir_path <- paste0(species, "_", category, "_tmp/")

    # Obtain a list with all files names that finish with .tmp
    file_paths <- list.files(path = dir_path, pattern = "\\.tmp$", full.names = TR

# Function to read the tables
read_tables <- function(file_paths) {
    lapply(file_paths, read.table, header = TRUE)
}

# Read tables
tables <- read_tables(file_paths)

# Find the maximum number of rows in each table
max_rows <- max(sapply(tables, nrow))

# Function to fill tables with NAs until reaching the maximum number of rows
fill_with_na <- function(df, max_rows) {
    if (nrow(df) < max_rows) df[(nrow(df) + 1):max_rows, ] <- NA
    return(df)
}

# Fill all tables with NA so they have the same number of rows
tables_filled <- lapply(tables, fill_with_na, max_rows)

# Bind tables side by side
merged_df <- bind_cols(tables_filled)

# Write te output file
output_file <- paste0(species, "_", category, "_go-omegas.tsv")
write.table(merged_df, file = output_file, sep = "\t", row.names = FALSE, quote =
```

8.5 "MW_test_multipleX.R": performs Mann-Whitney U test with GOs that have at least X genes associated to them (Topic 6.2) (by ChatGPT)

```r
#!/usr/bin/env Rscript

# Load necessary libraries
library(dplyr)
library(ggplot2)

# Define arguments
args <- commandArgs(trailingOnly = TRUE)
if (length(args) < 4) {
    stop("Usage: script.R <path> <species> <category> <X1,X2,...,XN>")
}
work_dir <- args[1]
species <- args[2]
category <- args[3]
X_values <- as.numeric(strsplit(args[4], ",")[[1]])

# Set working directory
setwd(work_dir)

# Define input files
input_file <- paste0(species, "_", category, "_go-omegas.tsv")
fixed_column_file <- paste0(species, "_short-infos.tsv")

# Read the main table
df <- read.table(input_file, header = TRUE, sep = "\t")

# Read the table containing the fixed column
fixed_df <- read.table(fixed_column_file, header = TRUE, sep = "\t")

# Assuming the fixed column is the third column
fixed_column <- fixed_df[, 3]

# Function to perform Mann-Whitney U test and select top 15 headers
perform_tests <- function(X) {
    # Filter columns with at least X non-NA values
    filtered_df <- df %>% select_if(~ sum(!is.na(.)) >= X)

    # Perform Mann-Whitney U test for each selected column and store p-value
    p_values <- sapply(filtered_df, function(column) {
        wilcox.test(column, fixed_column, na.rm = TRUE)$p.value
    })

    # Apply FDR correction to p-values
    p_values_adj <- p.adjust(p_values, method = "fdr")

    # Calculate medians of each column
    medians <- sapply(filtered_df, median, na.rm = TRUE)

    # Calculate the median of the fixed column
    fixed_median <- median(fixed_column, na.rm = TRUE)

    # Headers of columns with significantly higher medians
    higher_significant_headers <- colnames(filtered_df)[p_values_adj < 0.05
    higher_significant_headers <- higher_significant_headers[order(medians[h

    # Headers of columns with significantly lower medians
    lower_significant_headers <- colnames(filtered_df)[p_values_adj < 0.05 &
    lower_significant_headers <- lower_significant_headers[order(medians[low

    return(list(
        higher_significant_headers = higher_significant_headers,
        lower_significant_headers = lower_significant_headers
    ))
}

# Execute tests for each X value
results <- lapply(X_values, perform_tests)

# Extract results
higher_significant_headers_list <- lapply(results, function(result) result$h
lower_significant_headers_list <- lapply(results, function(result) result$lo

# Create tables for top 15 higher and lower headers
top15_higher_table <- data.frame(matrix(nrow = 15, ncol = length(X_values)))
top15_lower_table <- data.frame(matrix(nrow = 15, ncol = length(X_values)))
colnames(top15_higher_table) <- paste0("X", X_values)
colnames(top15_lower_table) <- paste0("X", X_values)

for (i in 1:length(X_values)) {
    top15_higher_table[, i] <- higher_significant_headers_list[[i]]
    top15_lower_table[, i] <- lower_significant_headers_list[[i]]
}

# Save the tables
write.table(top15_higher_table, file = paste0(species, "_", category, "_top1
write.table(top15_lower_table, file = paste0(species, "_", category, "_top15

# Function to perform Mann-Whitney U test and count the number of significan
```

```r
perform_test_counts <- function(X) {
    # Filter columns with at least X non-NA values
    filtered_df <- df %>% select_if(~ sum(!is.na(.)) >= X)

    # Perform Mann-Whitney U test for each selected column and store p-value
    p_values <- sapply(filtered_df, function(column) {
        wilcox.test(column, fixed_column, na.rm = TRUE)$p.value
    })

    # Apply FDR correction to p-values
    p_values_adj <- p.adjust(p_values, method = "fdr")

    # Count the number of significant tests (adjusted p-value < 0.05)
    num_significant <- sum(p_values_adj < 0.05)

    return(num_significant)
}

# Execute tests for each X value and count the significant tests
num_significant <- sapply(X_values, perform_test_counts)

# Create a data frame for the plot
df_plot <- data.frame(X = X_values, num_significant = num_significant)

# Generate the plot
plot_file <- paste0(species, "_", category, "_num_significant_plot.png")
ggplot(df_plot, aes(x = X, y = num_significant)) +
    geom_line() +
    geom_point() +
    labs(title = "Number of Significant Tests vs. X", x = "X", y = "Number o
    theme_minimal() +
    ggsave(plot_file)

print(paste("Graph saved to", plot_file))
```

8.6 "MW_test_table.R": creates a table that compiles info necessary to plot a multiple boxplot graph (by ChatGPT)

```r
#!/usr/bin/env Rscript

# Load necessary libraries
library(dplyr)

# Define arguments
args <- commandArgs(trailingOnly = TRUE)
if (length(args) < 4) {
    stop("Usage: script.R <path> <species> <X_value> <category1,category2,..
}
work_dir <- args[1]
species <- args[2]
X_value <- as.numeric(args[3])
categories <- strsplit(args[4], ",")[[1]]

# Set working directory
setwd(work_dir)

# Define the input file for the fixed column
fixed_column_file <- paste0(species, "_short-infos.tsv")

# Read the table containing the fixed column
fixed_df <- read.table(fixed_column_file, header = TRUE, sep = "\t")

# Assuming the fixed column is the third column
fixed_column <- fixed_df[, 3]

# Function to perform Mann-Whitney U test and select top 15 headers
perform_tests <- function(df, X) {
    # Filter columns with at least X non-NA values
    filtered_df <- df %>% select_if(~ sum(!is.na(.)) >= X)

    # Perform Mann-Whitney U test for each selected column and store p-value
    p_values <- sapply(filtered_df, function(column) {
        wilcox.test(column, fixed_column, na.rm = TRUE)$p.value
    })

    # Apply FDR correction to p-values
    p_values_adj <- p.adjust(p_values, method = "fdr")

    # Calculate medians of each column
    medians <- sapply(filtered_df, median, na.rm = TRUE)

    # Calculate the median of the fixed column
    fixed_median <- median(fixed_column, na.rm = TRUE)

    # Headers of columns with significantly higher medians
    higher_significant_headers <- colnames(filtered_df)[p_values_adj < 0.05
    higher_significant_headers <- higher_significant_headers[order(medians[h

    # Headers of columns with significantly lower medians
    lower_significant_headers <- colnames(filtered_df)[p_values_adj < 0.05 &
    lower_significant_headers <- lower_significant_headers[order(medians[low

    return(list(
        higher_significant_headers = higher_significant_headers,
        lower_significant_headers = lower_significant_headers,
        filtered_df = filtered_df
    ))
}

# Initialize lists to store results
category_list <- c()
header_list <- c()
values_list <- c()

# Process each category
for (category in categories) {
    # Define the input file for the category
    input_file <- paste0(species, "_", category, "_go-omegas.tsv")

    # Read the table for the category
    df <- read.table(input_file, header = TRUE, sep = "\t")

    # Perform tests to get significant headers
    results <- perform_tests(df, X_value)

    # Extract results
    higher_significant_headers <- results$higher_significant_headers
    lower_significant_headers <- results$lower_significant_headers
    filtered_df <- results$filtered_df

    # Collect data for higher significant headers
    for (header in higher_significant_headers) {
        if (!is.na(header)) {
            values <- paste(na.omit(filtered_df[[header]]), collapse = ", ")
            category_list <- c(category_list, category)
            header_list <- c(header_list, header)
```

```
            values_list <- c(values_list, values)
        }
    }

    # Collect data for lower significant headers
    for (header in lower_significant_headers) {
        if (!is.na(header)) {
            values <- paste(na.omit(filtered_df[[header]]), collapse = ", ")
            category_list <- c(category_list, category)
            header_list <- c(header_list, header)
            values_list <- c(values_list, values)
        }
    }
}

# Collect data for the fixed column
fixed_values <- paste(na.omit(fixed_column), collapse = ", ")
category_list <- c(category_list, rep("Genome", 1))
header_list <- c(header_list, rep("Genome", 1))
values_list <- c(values_list, fixed_values)

# Create the final data frame
final_df <- data.frame(Category = category_list, GO_ID = header_list, List_C

# Save the final table
output_file <- paste0(species, "_top30_GOs.tsv")
write.table(final_df, file = output_file, sep = "\t", row.names = FALSE, quo
```

8.7 "GO_terms.py": inputs a GO id list and outputs a table with ids and term names (by ChatGPT)

```
import requests
import pandas as pd

# Lista de GO IDs
go_ids = [
    "GO:0050912", "GO:0050909", "GO:0050907", "GO:0009593", "GO:0010951", "GO
]

# Função para obter os nomes completos dos termos GO usando a API do QuickGO
def get_go_names(go_ids):
    base_url = "https://www.ebi.ac.uk/QuickGO/services/ontology/go/terms/"
    headers = {"Accept": "application/json"}
    go_names = []

    for go_id in go_ids:
        response = requests.get(base_url + go_id, headers=headers)
        if response.status_code == 200:
            data = response.json()
            go_names.append({"GO ID": go_id, "Term Name": data['results'][0]
        else:
            go_names.append({"GO ID": go_id, "Term Name": "Not found"})

    return go_names

# Obter os nomes completos dos termos GO
go_names = get_go_names(go_ids)

# Converter a lista de dicionários em um DataFrame
df_go_names = pd.DataFrame(go_names)

# Salvar em um arquivo CSV (opcional)
df_go_names.to_csv("go_term_names.csv", index=False)

# Exibir a tabela
print(df_go_names)
```

8.8 "GO_boxplot.R": script to create figures like the one from Scott et al. (2020), for the rate of evolution of the genes associated to specific GOs

```r
library(tidyr)
library(dplyr)
library(ggplot2)
setwd("C:/Users/Pedro/OneDrive/Área de Trabalho/GO_figure")

#### Reading the files ####
Chom_genome <- read.table("Chom_genome_omegas.tsv", sep="\t", header=T)
Chom_top30 <- read.table("Chom_top30_GOs.tsv", sep="\t", header=F)

Pazu_genome <- read.table("Pazu_genome_omegas.tsv", sep="\t", header=T)
Pazu_top30 <- read.table("Pazu_top30_GOs.tsv", sep="\t", header=F)

#### Fixing the top30 tables ####
# Chom
colnames(Chom_top30)[1] <- "Category" # fixing the header
colnames(Chom_top30)[2] <- "GO_id"
colnames(Chom_top30)[3] <- "GO_term"

Chom_top30_long <- gather(Chom_top30, Number, Omega, V4:V580) # wide to long

Chom_top30_long <- na.omit(Chom_top30_long) # remove NA, because there diffe
                                            # quantity of genes with each GO

# Pazu
colnames(Pazu_top30)[1] <- "Category" # fixing the header
colnames(Pazu_top30)[2] <- "GO_id"
colnames(Pazu_top30)[3] <- "GO_term"

Pazu_top30_long <- gather(Pazu_top30, Number, Omega, V4:V537) # wide to long

Pazu_top30_long <- na.omit(Pazu_top30_long) # remove NA, because there diffe
                                            # quantity of genes with each GO

#### Fixing genome file and merging ####
# Chom
G <- rep("Genome", nrow(Chom_genome)) # make the genome table equal to the G
Chom_genome <- cbind(G,G,G,G,Chom_genome)
colnames(Chom_genome)[1] <- "Category"
colnames(Chom_genome)[2] <- "GO_id"
colnames(Chom_genome)[3] <- "GO_term"
colnames(Chom_genome)[4] <- "Number"

Chom_final <- rbind(Chom_top30_long,Chom_genome) # merge everything

Chom_final$Category <- factor(Chom_final$Category,
                              levels = c("BP", "MF", "CC", "Genome")) # fix
Chom_final$GO_term <- as.factor(Chom_final$GO_term)
Chom_final$Omega <- as.numeric(Chom_final$Omega)

Chom_final_medians <- Chom_final %>% # organize GO terms by omega mean
  group_by(GO_term) %>%
  summarize(median_Omega = median(Omega, na.rm = TRUE))

Chom_final <- Chom_final %>%
  mutate(GO_term = factor(GO_term, levels = Chom_final_medians$GO_term[order

# Pazu
G <- rep("Genome", nrow(Pazu_genome)) # make the genome table equal to the G
Pazu_genome <- cbind(G,G,G,G,Pazu_genome)
colnames(Pazu_genome)[1] <- "Category"
colnames(Pazu_genome)[2] <- "GO_id"
colnames(Pazu_genome)[3] <- "GO_term"
colnames(Pazu_genome)[4] <- "Number"

Pazu_final <- rbind(Pazu_top30_long,Pazu_genome) # merge everything

Pazu_final$Category <- factor(Pazu_final$Category,
                              levels = c("BP", "MF", "CC", "Genome")) # fix
Pazu_final$GO_term <- as.factor(Pazu_final$GO_term)
Pazu_final$Omega <- as.numeric(Pazu_final$Omega)

Pazu_final_medians <- Pazu_final %>% # organize GO terms by omega mean
  group_by(GO_term) %>%
  summarize(median_Omega = median(Omega, na.rm = TRUE))

Pazu_final <- Pazu_final %>%
  mutate(GO_term = factor(GO_term, levels = Pazu_final_medians$GO_term[order

#### Figure ####
# Chom
chom_genome_median = median(Chom_genome$Omega)

svg("Chom_GO_fig.svg", width=15, height=15)
ggplot(Chom_final, aes(x=Omega,y=GO_term,fill=Category)) +
  geom_boxplot () +
  facet_grid(Category ~ ., scales= "free") +
  labs(title = "Cochliomyia hominivorax", x="dN/dS ratio", y="GO terms") +
  theme(plot.title = element_text(face = "italic")) +
```

```
  geom_vline(xintercept=chom_genome_median, linetype="dashed", color="red")
dev.off()

png("Chom_GO_fig.png", width=1000, height=1000)
ggplot(Chom_final, aes(x=Omega,y=GO_term,fill=Category)) +
  geom_boxplot () +
  facet_grid(Category ~ ., scales= "free") +
  labs(title = "Cochliomyia hominivorax", x="dN/dS ratio", y="GO terms") +
  theme(plot.title = element_text(face = "italic")) +
  geom_vline(xintercept=chom_genome_median, linetype="dashed", color="red")
dev.off()

# Pazu
pazu_genome_median = median(Pazu_genome$Omega)

svg("Pazu_GO_fig.svg", width=15, height=15)
ggplot(Pazu_final, aes(x=Omega,y=GO_term,fill=Category)) +
  geom_boxplot () +
  facet_grid(Category ~ ., scales= "free") +
  theme(plot.title = element_text(face = "italic")) +
  labs(title = "Protocalliphora azurea", x="dN/dS ratio", y="GO terms") +
  geom_vline(xintercept=pazu_genome_median, linetype="dashed", color="red")
dev.off()

png("Pazu_GO_fig.png", width=1000, height=1000)
ggplot(Pazu_final, aes(x=Omega,y=GO_term,fill=Category)) +
  geom_boxplot () +
  facet_grid(Category ~ ., scales= "free") +
  labs(title = "Protocalliphora azurea", x="dN/dS ratio", y="GO terms") +
  theme(plot.title = element_text(face = "italic")) +
  geom_vline(xintercept=pazu_genome_median, linetype="dashed", color="red")
dev.off()
```

8.9 "get_ids_from_cluster_list.py"

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

'''
PROGRAM: get_sequences_from_id_list.py

DESCRIPTION: Reads a file with a list of ortholog clusters identifiers and
             recovers the sequences belonging to a given species from these
             clusters fasta files.

AUTHOR: Pedro Mariano Martins                    São Paulo, Brazil, 09.08.202
'''

import sys
import getopt
import glob
#-------------------------------------------------------------------------------
def main():
    ids, species, cluster_pwd, cluster_end = arg()

    if ids == None or species == None or cluster_pwd == None or cluster_end =
        usage ()

    clusters = glob.glob(f'{cluster_pwd}/*{cluster_end}')

    #making a list with the ids
    idents = open(ids, 'r', encoding='utf-8')
    id_list = []

    for line in idents:
        line = line.strip()
        id_list += [line]

    idents.close()

    #finding clusters and sequences, and saving them in an output file
    out = open(f'{species}_table_from_clusters.txt', 'w', encoding='utf-8')

    for cluster in clusters:
        NA = 0
        cluster_name = cluster.split(f'{cluster_pwd}/')[1]
        for i in id_list:
            if i in cluster:
                fasta = open(cluster, 'r', encoding='utf-8')
                for line in fasta:
                    if species in line:
                        line = line.strip()
                        out.write(f'{cluster_name}\t{line}\n')
                        NA = 1
                fasta.close()
                if NA == 0:
                    out.write(f'{cluster_name}\tNA\n')

    out.close()
#-------------------------------------------------------------------------------
def arg():
    argv = sys.argv[1:]
    try:
        opts, args = getopt.getopt(argv, 'i:s:c:e:')

    except:
        usage()

    arg_1, arg_2, arg_3, arg_4 = None, None, None, None

    for opt, arg in opts:
        if opt in ['-i']:
            arg_1 = arg
        elif opt in ['-s']:
            arg_2 = arg
        elif opt in ['-c']:
            arg_3 = arg
        elif opt in ['-e']:
            arg_4 = arg

    return arg_1, arg_2, arg_3, arg_4
#-------------------------------------------------------------------------------
def usage():
    sys.exit('Usage:\n'
           '\t-i: list of cluster names\n'
           '\t-s: species name\n'
           '\t-c: cluster files pathway\n'
           '\t-e: cluster files ending')
#-------------------------------------------------------------------------------
if __name__ == '__main__':
```