Blowfly non-coding elements

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

Table of Contents

Programs used (and prerequisites):

Software	Version

BUSCO	5.4.4
QUAST	5.0.2
RepeatMasker	4.1.5
RMBlast	2.11.0
TRF	4.09.1
Cactus	2.6.4
HDF5	1.10.1
HAL	2.2
Clapack	3.2.1
PHAST	1.6
bedtools	2.30.0
BEDOPS	2.4.41
Mafft	7.310
PhyloACC	2.2.0
ASTRAL	5.7.1
IQTree	2.0.6
MEME Suite	5.5.4

0. New Genomes

Three new genomes:

Species	Company	Technology	HackMD link
Lucilia eximia	Dovetail	PacBio	link
Chrysomya putoria	Dovetail	PacBio	link
Chrysomya megacephala	Dovetail	PacBio	link

1. Previously available Genomes

Genomes of blowflies were dowloaded from the NCBI database. If a RefSeq version were available, it was preferred over the GenBank version. If not, I used the GenBank version.

Species	Accession code	Ref
Lucilia cuprina	GCF_022045245.1	Anstead et al. 2015
Lucilia sericata	GCF_015586225.1	Davis et al. 2021
Chrysomya rufifacies[1]	GCA_014858695.1	Andere et al. 2020
Phormia regina	GCA_001735545.1	Andere et al. 2016
Calliphora vicina	GCA_001017275.1	Vicoso & Bachtrog, 2015
Calliphora vomitoria	GCA_942486065.1	Darwin Tree of Life Project (https://portal.darwintreeoflife.org/data vomitoria)
Protocalliphora azurea	GCA_932274085.1	Darwin Tree of Life Project (https://portal.darwintreeoflife.org/data azurea)
Bellardia pandia	GCA_916048285.2	Darwin Tree of Life Project (https://portal.darwintreeoflife.org/data pandia)

Cochliomyia hominivorax[2]	https://datadryad.org/stash/dataset/doi:10.5061/dryad.d7wm37q4j	Tandonnet et al. 2023
Calliphora (Aldrichina) grahami[3]	http://dx.doi.org/10.5524/100673	Meng et al. 2020

[1]There are three assemblies (arrhenogenic female, thelygenic female, and male). This is the male assembly, I chose it for the Y chromosome.
[2]There is a GenBank assembly for this species (GCA_004302925.1), but I used the updated version of the genome (Sophie's paper).
[3] Not on NCBI, but another repository.

Darwin Tree of Life Project Genomes: https://portal.darwintreeoflife.org/tracking
https://projects.ensembl.org/darwin-tree-of-life/
https://portal.darwintreeoflife.org/tree

I edited sequence ids (left just the first identifier, erasing everything after the first space) to make it shorter and easier to read. These were used afterwards. All original genomes are stored in a compressed file.

```
# Original
/home/pedro/Non_Coding_Element_Evolution/0-Original_Genomes.tar.gz
# Edited
cd /home/pedro/Non_Coding_Element_Evolution/1-Genomes
for i in *; do sed -i 's/ .*//g' $i; done
```

Number of scaffolds/contigs in each assembly:
Agra.fasta - 1604
Bpan.fasta - 71
Chom.fasta - 522
Cruf.fasta - 109329
Cvic.fasta - 197510
Cvom.fasta - 129
Lcup.fasta - 8457
Lser.fasta - 4371
Pazu.fasta - 24
Preg.fasta - 192460

Edited genomes were also compressed after the RepeatMasker run finished (see section 3.2.)

2. Quality of genomes

```
mkdir /home/pedro/Non_Coding_Element_Evolution/2-Genome_quality

cd /home/pedro/Non_Coding_Element_Evolution/2-Genome_quality
```

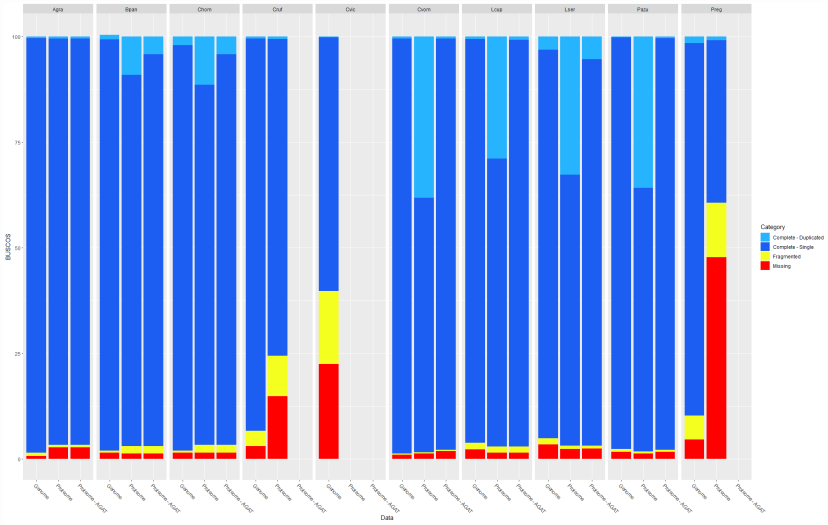
2.1. BUSCO

I ran BUSCO to check for the completeness of each genome.

```
# BUSCO v5.4.4
mkdir /home/pedro/Non_Coding_Element_Evolution/2-Genome_quality/1-BUSCO

cd 1-BUSCO/

# For some reason, it only works if the genomes are in the same folder, so I
cp /home/pedro/Non_Coding_Element_Evolution/1-Genomes/*.fasta .

sudo docker pull ezlabgva/busco:v5.4.4_cv1

for GENOME in *fasta; do sudo docker run -u $(id -u) -v $(pwd):/busco_wd ezl
```

Species	Complete (S + D)	Fragmented	Missing
Agra	98.6 (98.2 + 0.4)	0.7	0.7
Bpan	98.5 (97.4 + 1.1)	0.3	1.2

Chom	98.1 (96.0 + 2.1)	0.5	1.4
Cruf	93.4 (92.9 + 0.5)	3.6	3.0
Cvic	60.3 (60.1 + 0.2)	17.3	22.4
Cvom	98.7 (98.2 + 0.5)	0.3	1.0
Lcup	96.2 (95.6 + 0.6)	1.6	2.2
Lser	95.1 (92.0 + 3.1)	1.5	3.4
Pazu	97.7 (97.5 + 0.2)	0.2	1.6
Preg	89.8 (88.2 + 1.6)	5.6	4.6



OBS: This has Carol's results with the proteomes too, for comparison

2.2. QUAST

```
# QUAST v5.0.2
mkdir /home/pedro/Non_Coding_Element_Evolution/2-Genome_quality/2-QUAST

cd 2-QUAST/

# Running
for GENOME in /home/pedro/Non_Coding_Element_Evolution/2-Genome_quality/1-BUS



for i in *; do sudo quast.py $i -t 10 --eukaryote --large --memory-efficient
```

Species	#contigs	#contigs>50kb	Length	Length>50kb	N_{50}	N_{75}	L_{50}	L_{75}	Ns
Agra	1604	736	600090062	585702444 (97.6%)	1925180	796802	79	199	0
Bpan	71	47	617043932	616226721 (99.8%)	111823772	110188975	3	4	4.0
Chom	522	82	534400685	521892830 (97.6%)	101522387	93884324	3	4	74.
Cruf	109329	75	288472566	4853570 (1.7%)	7933	4737	5734	13038	108
Cvic	197510	1	459231066	53150 (0.01%)	4480	3596	13675	24759	187
Cvom	129	41	706678423	704415709 (99.7%)	131076759	125523795	3	4	5.4
Lcup	8457	6	409159265	392766389 (95.9%)	71024062	67696999	3	4	49

Lser	4371	2222	565305478	513238679 (90.7%)	296078	137019	500	1197	0
Pazu	24	8	874252225	873869904 (99.9%)	164199054	157909393	3	4	8.8
Preg	192460	283	549932840	18344189 (3.3%)	13034	6995	8518	18763	10

Decisions

The Cvic, Preg, and Cruf genomes are too fragmented and this mitght not bee got for the WGA and the analysis of the non-coding regions. So I won't even mask them. Preg and Cruf might be ok for the gene family analysis (not Cvic), but this is something to be analyzed later.

3. Repeats and soft masking

I used RepeatModeler and RepeatMasker to get softmasked versions of all the genomes. Also, it provides tables summarizing the repeats found in each genome.

I did it all in Darwin, using Vanessa's user, because everything was already installed in there.

```
# Genomes are here
/home/cunha/Pedro/1-Genomes
```

3.1. RepeatModeler

Using Docker. One container for each genome, because we've had some porblems with containers desappearing before.

```
#### General ####
# Get in
docker exec -it [name] /bin/bash
# Get out
Ctrl P+Q
# Check existing containers
docker ps

#### Agra ####
# New container
docker run -it --rm dfam/tetools:latest # elastic_lehmann

# Copying genome to Docker
docker cp /home/cunha/Pedro/1-Genomes/Agra.fasta elastic_lehmann:/home

# Accessing the container
docker exec -it elastic_lehmann /bin/bash

# Database
BuildDatabase -name Agra_database Agra.fasta

# Running
RepeatModeler -database Agra_database -threads 30 -LTRStruct  >Agra_log 2>Ag

# Copying the results
cd /home/cunha/Pedro/2-Masking/1-RepeatModeler

docker cp -a elastic_lehmann:/home/Agra_database-families.fa ./
docker cp -a elastic_lehmann:/home/Agra_database-families.stk  ./
docker cp -a elastic_lehmann:/home/Agra_database-rmod.log ./
docker cp -a elastic_lehmann:/home/Agra_RepeatModeler.tar.gz ./ # this has a

#### Bpan ####
# New container
docker run -it --rm dfam/tetools:latest # dreamy_hugle

# Copying genome to Docker
docker cp /home/cunha/Pedro/1-Genomes/Bpan.fasta dreamy_hugle:/home

# Accessing the container
docker exec -it dreamy_hugle /bin/bash

# Database
BuildDatabase -name Bpan_database Bpan.fasta

# Running
RepeatModeler -database Bpan_database -threads 30 -LTRStruct  >Bpan_log 2>Bp

# Copying the results
cd /home/cunha/Pedro/2-Masking/1-RepeatModeler

docker cp -a dreamy_hugle:/home/Bpan_database-families.fa ./
docker cp -a dreamy_hugle:/home/Bpan_database-families.stk  ./
docker cp -a dreamy_hugle:/home/Bpan_database-rmod.log ./
docker cp -a dreamy_hugle:/home/Bpan_RepeatModeler.tar.gz ./

#### Chom ####
# Container - elastic_lehmann (same as Agra)

# Copying genome to Docker
docker cp /home/cunha/Pedro/1-Genomes/Chom.fasta elastic_lehmann:/home

# Accessing the container
docker exec -it elastic_lehmann /bin/bash

# Database
BuildDatabase -name Chom_database Chom.fasta

# Running
RepeatModeler -database Chom_database -threads 30 -LTRStruct  >Chom_log 2>Ch

# Copying the results
cd /home/cunha/Pedro/2-Masking/1-RepeatModeler

docker cp -a elastic_lehmann:/home/Chom_database-families.fa ./
docker cp -a elastic_lehmann:/home/Chom_database-families.stk  ./
docker cp -a elastic_lehmann:/home/Chom_database-rmod.log ./
docker cp -a elastic_lehmann:/home/Chom_RepeatModeler.tar.gz ./

#### Cvom ####
# Container - dreamy_hugle (same as Bpan)

# Copying genome to Docker
docker cp /home/cunha/Pedro/1-Genomes/Cvom.fasta dreamy_hugle:/home

# Accessing the container
docker exec -it dreamy_hugle /bin/bash

```

```
# Database
BuildDatabase -name Cvom_database Cvom.fasta

# Running
RepeatModeler -database Cvom_database -threads 30 -LTRStruct  >Cvom_log 2>Cv

# Copying the results
cd /home/cunha/Pedro/2-Masking/1-RepeatModeler

docker cp -a dreamy_hugle:/home/Cvom_database-families.fa ./
docker cp -a dreamy_hugle:/home/Cvom_database-families.stk  ./
docker cp -a dreamy_hugle:/home/Cvom_database-rmod.log ./
docker cp -a dreamy_hugle:/home/Cvom_RepeatModeler.tar.gz ./

#### Lcup ####
# New container
docker run -it --rm dfam/tetools:latest # determined_ride

# Copying genome to Docker
docker cp /home/cunha/Pedro/1-Genomes/Lcup.fasta determined_ride:/home

# Accessing the container
docker exec -it determined_ride /bin/bash

# Database
BuildDatabase -name Lcup_database Lcup.fasta

# Running
RepeatModeler -database Lcup_database -threads 30 -LTRStruct  >Lcup_log 2>Lc

# Copying the results
cd /home/cunha/Pedro/2-Masking/1-RepeatModeler

docker cp -a determined_ride:/home/Lcup_database-families.fa ./
docker cp -a determined_ride:/home/Lcup_database-families.stk  ./
docker cp -a determined_ride:/home/Lcup_database-rmod.log ./
docker cp -a determined_ride:/home/Lcup_RepeatModeler.tar.gz ./

#### Lser ####
# New container
docker run -it --rm dfam/tetools:latest # happy_snyder

# Copying genome to Docker
docker cp /home/cunha/Pedro/1-Genomes/Lser.fasta happy_snyder:/home

# Accessing the container
docker exec -it happy_snyder /bin/bash

# Database
BuildDatabase -name Lser_database Lser.fasta

# Running
RepeatModeler -database Lser_database -threads 30 -LTRStruct  >Lser_log 2>Ls

# Copying the results
cd /home/cunha/Pedro/2-Masking/1-RepeatModeler

docker cp -a happy_snyder:/home/Lser_database-families.fa ./
docker cp -a happy_snyder:/home/Lser_database-families.stk  ./
docker cp -a happy_snyder:/home/Lser_database-rmod.log ./
docker cp -a happy_snyder:/home/Lser_RepeatModeler.tar.gz ./

# IMPORTANT NOT - THE LTR PIPELINE DID NOT WORK IN THIS AUTOMATED WAY

#### Pazu ####
# New container
docker run -it --rm dfam/tetools:latest # funny_payne

# Copying genome to Docker
docker cp /home/cunha/Pedro/1-Genomes/Pazu.fasta funny_payne:/home

# Accessing the container
docker exec -it funny_payne /bin/bash

# Database
BuildDatabase -name Pazu_database Pazu.fasta

# Running
RepeatModeler -database Pazu_database -threads 30 -LTRStruct  >Pazu_log 2>Pa

# Copying the results
cd /home/cunha/Pedro/2-Masking/1-RepeatModeler

docker cp -a funny_payne:/home/Pazu_database-families.fa ./
docker cp -a funny_payne:/home/Pazu_database-families.stk ./
docker cp -a funny_payne:/home/Pazu_database-rmod.log ./
docker cp -a funny_payne:/home/Pazu_RepeatModeler.tar.gz ./
```

```
# Removing the containers (since I already got all the outputs)
docker stop elastic_lehmann # Agra & Chom
docker stop dreamy_hugle # Bpan & Cvom
docker stop determined_ride # Lcup
docker stop happy_snyder # Lser NOT YET
docker stop funny_payne # Pazu
```

I did all of this on Darwin, but I'll use RepeatMasker on Rosalind, so I'll move all the files there.

```
cd /home/cunha/Pedro/2-Masking/1-RepeatModeler # in Darwin

scp -P 2205 * pedro@143.107.244.181:/home/pedro/Non_Coding_Element_Evolution/
```

3.2. RepeatMasker

3.2.1. Installation

Pre requisites

```
# h5py python library
pip install h5py

# RMBlast v2.11.0
scp -P 2205 Mestrado_genomas/rmblast-2.11.0+-x64-linux.tar.gz pedro@143.107.2

tar -xf rmblast-2.11.0+-x64-linux.tar.gz

# Tandem Repeat Finder v4.09.1
scp -P 2205 Mestrado_genomas/TRF-4.09.1.tar.gz pedro
@143.107.244.181:/home/pedro/Programs

tar -xf TRF-4.09.1.tar.gz

cd TRF-4.09.1/
mkdir build
cd build
../configure
make
sudo make install
```

RepeatMasker

```
# Distribution - RepeatMasker v4.1.2
scp -P 2205 Mestrado_genomas/RepeatMasker-4.1.5-p1.tar.gz pedro@143.107.244.1

tar -xf RepeatMasker-4.1.5-p1.tar.gz

cd home/pedro/Programs/RepeatMasker/

perl ./configure
```

About the additional libraries:

- The Dfam file (https://www.dfam.org/releases/Dfam_3.6/families/Dfam.h5.gz) is enormous and would take too long to download. Also, there already is a Dfam file in the RepeatMasker distribution, so it might work by itself (this is Dfam 3.3)
- The RepBase library requires a paid subscription to be downloaded, so we cannot use it.

3.2.2. Running

For this, I use the repeat libraries that I got using RepeatModeler, because they are species-specific.

```
# I'm doing this on Rosalind
cd /home/pedro/Non_Coding_Element_Evolution/3-Masking/2-RepeatMasker

# repeatmasker.bash
for GENOME in /home/pedro/Non_Coding_Element_Evolution/1-Genomes/*
do
    FASTA=${GENOME##/home/pedro/Non_Coding_Element_Evolution/1-Genomes/}
    DB=${FASTA%%.fasta}_database
    SPP=${FASTA%%.fasta}

    echo $SPP
    echo "RepeatMasker -lib /home/pedro/Non_Coding_Element_Evolution/3-Maski

    RepeatMasker -lib /home/pedro/Non_Coding_Element_Evolution/3-Masking/1-Re

done
```

With this command line, I had 6 outputs for each genome:

- Masked genome
- .align (alignments of all repeats to the genomes)
- .out (huge table with all information)
- .out.gff (outfile in gff format)
- .out.html (outfile in html format)
- .tbl (a smaller table that summarizes the .out table)

Organizing things:

```
cd /home/pedro/Non_Coding_Element_Evolution/3-Masking/2-RepeatMasker

# Moving some less important outputs to a subdirectory and compressing them
mkdir Other_outputs
mv *out* Other_outputs
mv *align Other_outputs
mv *gz Other_outputs
mv *log Other_outputs
tar -czvf Other_outputs.tar.gz Other_outputs
```

LTR - Lser

When running RepeatModeler, the LTR pipeline failed for the Lser genome. So I did a separate analysis to solve this using LTR Harvest and LTR Retriever.

```
docker run -it --rm dfam/tetools:latest # distracted_dewdney

docker cp /home/pedro/Non_Coding_Element_Evolution/1-Genomes/Lser.fasta dist

docker exec -it distracted_dewdney /bin/bash

# Got these command lines from the LTR retriever help guide
# This is the LTR.sh script
/opt/genometools/bin/gt  suffixerator -db Lser.fasta -indexname Lser_db.fa -t

/opt/genometools/bin/gt ltrharvest -index Lser_db.fa -minlenltr 100 -maxlenlt

/opt/LTR_retriever/LTR_retriever -genome Lser.fasta -inharvest Lser.harvest.s
```

This worked! I have a fasta file with the identified LTR sequences from the Lser genome (and a bunch of other outputs). I copied everything back from de container.

```
mkdir /home/pedro/Non_Coding_Element_Evolution/3-Masking/Lser_LTR
cd /home/pedro/Non_Coding_Element_Evolution/3-Masking/Lser_LTR

# Copy
docker cp -a distracted_dewdney:/home/Lser.fasta.LTR.gff3 ./
docker cp -a distracted_dewdney:/home/Lser.fasta.LTRlib.fa ./
docker cp -a distracted_dewdney:/home/Lser_LTR_ouputs.tar.gz ./
docker cp -a distracted_dewdney:/home/LTR.sh ./

# Terminate the container
docker stop distracted_dewdney
```

To check if it made any difference, I'll concatenate the Lser.fasta.LTRlib.fa to the output I had from the first RepeatModeler run, then use this new, improved, library, for RepeatMasker.

```
cd /home/pedro/Non_Coding_Element_Evolution/3-Masking/Lser_LTR

cp /home/pedro/Non_Coding_Element_Evolution/3-Masking/1-RepeatModeler/Lser_da

# New database
cat Lser_database-families.fa Lser.fasta.LTRlib.fa >Lser_COMPLETE_database-fa
```

Finally, I do a new RepeatMasker run, to check if there was any difference from the previous trial:

```
cd /home/pedro/Non_Coding_Element_Evolution/3-Masking/Lser_LTR

RepeatMasker -lib Lser_COMPLETE_database-families.fa -dir ./ -pa 20 -a -xsma
```

It was different (and better), so I'll use this new version of the masked genome.

```
# Renaming and compressing the first run
cd /home/pedro/Non_Coding_Element_Evolution/3-Masking/2-RepeatMasker/

mkdir Lser_NO_LTR

for i in Lser*; do mv $i Lser_NO_LTR${i##Lser}; done
mv Lser_NO_LTR.* Lser_NO_LTR/

cd Other_outputs/
for i in Lser*; do mv $i Lser_NO_LTR${i##Lser}; done
mv Lser_NO_LTR.* ../Lser_NO_LTR/

cd ..
tar -czvf Lser_NO_LTR.tar.gz Lser_NO_LTR

# Moving things from the second run
cd /home/pedro/Non_Coding_Element_Evolution/3-Masking/Lser_LTR

mv Lser_COMPLETE_database-families.fa ../1-RepeatModeler/

mv Lser.fasta.LTR.gff3 ../1-RepeatModeler/Lser_LTR/
mv Lser.fasta.LTRlib.fa ../1-RepeatModeler/Lser_LTR/
mv Lser_LTR_ouputs.tar.gz ../1-RepeatModeler/Lser_LTR/
mv LTR.sh ../1-RepeatModeler/Lser_LTR/

mv Lser.fasta.masked Lser_masked.fasta
mv Lser.fasta.tbl Lser_masked.tbl
mv Lser_masked* ../2-RepeatMasker/

mv * ../2-RepeatMasker/Other_outputs
```

3.3. Summarizing information

I organized the outputs using a python script I wrote and made some plots using an Rscript (I did this locally, but everything is on Rosalind now)

```
# Everything is in here
/home/pedro/Non_Coding_Element_Evolution/3-Masking/3-Organizing_information
```

3.4. Compressing unmasked genomes

Once it was finished. The unmasked genomes were compressed, and only the masked versions were kept for further analyses (this was made to take up less storage).

```
/home/pedro/Non_Coding_Element_Evolution/1-Genomes.tar.gz
```

Note on the new genomes

We new genomes (Cmeg, Cput and Lexi) using the same pipeline used for the genomes I downloaded. From this moment on, I'm using all of them. I copied these masked genomes to my working directory and renamed them (to keep all files with the same patterns). I also renamed my own outputs to make everything simpler for me.

```
cd /home/pedro/Non_Coding_Element_Evolution/3-Masking/2-RepeatMasker

# Renaming my outputs
for i in *masked; do mv $i ${i%%.fasta.masked}_masked.fasta; done

for i in *tbl; do mv $i ${i%%.fasta.tbl}_masked.tbl; done

# Renaming the outputs from the new genomes
mv cput_N_genome_final.fa.masked Cput_masked.fasta
mv cput_N_genome_final.fa.tbl Cput_masked.tbl

mv cmeg_N_genome_final.fa.masked Cmeg_masked.fasta
mv cmeg_N_genome_final.fa.tbl Cmeg_masked.tbl

mv lexi_N_genome.fa.masked Lexi_masked.fasta
mv lexi_N_genome.fa.tbl Lexi_masked.tbl
```

4. Whole genome alignment (WGA)

To find conserved non-coding elements, I had to create a WGA. To do so, I choose to use ProgessiveCactus. Briefly, it aligns the genomes based on a phylogeny. Pairwise alignments are made based on species relatedness and ancestral genomes are estimated on every node of the tree. Also, a final aligment of the genomes is provided.

General information on Cactus:
https://github.com/ComparativeGenomicsToolkit/cactus

4.1. Installation

Cactus

```
# Precompiled binaries - Cactus v2.6.4
wget https://github.com/ComparativeGenomicsToolkit/cactus/releases/download/v

tar -xf cactus-bin-v2.6.4.tar.gz

cd home/pedro/Programs/cactus-bin-v2.6.4/

# Set up Pyhton environment
python3 -m pip install virtualenv
virtualenv -p python3.8 cactus_env

echo "export PATH=$/home/pedro/Programs/cactus-bin-v2.6.4/bin:\$PATH" >> cact
echo "export PYTHONPATH=$/home/pedro/Programs/cactus-bin-v2.6.4/lib:\$PYTHONP

source cactus_env/bin/activate

python3 -m pip install -U setuptools pip==23.2.1
python3 -m pip install -U -r ./toil-requirement.txt
python3 -m pip install -U .

# Afterwards, that's how I can get in the environment:
source /home/pedro/Programs/cactus-bin-v2.6.4/cactus_env/bin/activate
# This is how to get out of it:
deactivate
```

HAL

I installed HAL tools so I could turn the Cactus HAL output (a reference free alignment) into other formats that can be used in other analyses.

```
# HDF5 1.10.1 with C++ API enabled
mkdir /home/pedro/Programs/hdf5
wget http://www.hdfgroup.org/ftp/HDF5/releases/hdf5-1.10/hdf5-1.10.1/src/hdf5
tar xzf hdf5-1.10.1.tar.gz
cd hdf5-1.10.1
./configure --enable-cxx --prefix /home/pedro/Programs/hdf5
make && make install

export PATH=/home/pedro/Programs/hdf5/bin:${PATH}
export h5prefix=--prefix=/home/pedro/Programs/hdf5

#sonLib
git clone https://github.com/ComparativeGenomicsToolkit/sonLib.git
pushd sonLib && make sonLibRootDir=/home/pedro/Programs/sonLib && popd

pip install sonLib

# Gitclone - HAL v2.2
cd /home/pedro/Programs
git clone https://github.com/ComparativeGenomicsToolkit/hal.git
cd hal
make
export PATH=/home/pedro/Programs/hal/bin:${PATH}
export PYTHONPATH=/home/pedro/Programs:${PYTHONPATH}
export PYTHONPATH=/home/pedro/Programs/hal:${PYTHONPATH}
```

cwltool

I had to install this one, because a message came up when I was performing the alignments (not a fatal error, but it's better to get things right)

```
# I need to this inside the environment
source /home/pedro/Programs/cactus-bin-v2.6.4/cactus_env/bin/activate

pip install cwltool
```

Normally, this would suffice. But I got a message suggesting that I used a different version to match other dependencies. I'm assuming this varies depending on the situation, but here's the information anyway:

```
pip install cwltool==3.1.20211107152837
```

4.2. Tree

Cactus requires a phylogenetic tree to perform the WGA. The tree Gisele and I used doesn't include all the species I'm working with for this analysis, nor does the Dimensions tree (which also has the problem of having more than 1 branch for some species). So I'll use a new tree using the BUSCO outputs. I'm doing this with Vanessa and Letícia. All the information is here.

After having the tree, I slightly edited it. I removed unwanted tips (the species that have no genomic information), and rewrote some names (some had lowercase letters)

```
# cactus_tree.R
library(ape)
library(phytools)

setwd("C:/Users/Pedro/OneDrive/Área de Trabalho/Cactus")

# Read tree
tree <- read.tree("Calliphoridae_tree_root.tre")
plotTree(tree) # check

# Remove branches that won't be used
cactus <- drop.tip(tree, c("clop","loch","cbez","calb"))
plotTree(cactus) # check

# Rename branches with lowercase letters
cactus$tip.label[1] <- "Cmeg"
cactus$tip.label[2] <- "Cput"
cactus$tip.label[7] <- "Lexi"
plotTree(cactus) # check

# Newick
write.tree(cactus, "Cactus_tree.tre")
```

4.3. Running Cactus

Cactus file

Cactus relies on a specific kind of file that simouteneously contains the tree and the pathway to each genome. Also, I can choose some genomes to be marked as having reference quality: I chose all the ones that had an L_{50}=3 (see QUAST results)

The file is here:

```
cd /home/pedro/Non_Coding_Element_Evolution/4-ProgressiveCactus_WGA/

# Cactus_file.txt
((((Cmeg:0.0184171866,Cput:0.0451563064):0.0294538378,Chom:0.059936427):0.00(

Agra /home/pedro/Non_Coding_Element_Evolution/3-Masking/2-RepeatMasker/Agra_r
*Bpan /home/pedro/Non_Coding_Element_Evolution/3-Masking/2-RepeatMasker/Bpan_
*Cvom /home/pedro/Non_Coding_Element_Evolution/3-Masking/2-RepeatMasker/Cvom_
Cmeg /home/pedro/Non_Coding_Element_Evolution/3-Masking/2-RepeatMasker/Cmeg_r
Cput /home/pedro/Non_Coding_Element_Evolution/3-Masking/2-RepeatMasker/Cput_r
*Chom /home/pedro/Non_Coding_Element_Evolution/3-Masking/2-RepeatMasker/Chom_
Lcup /home/pedro/Non_Coding_Element_Evolution/3-Masking/2-RepeatMasker/Lcup_r
Lser /home/pedro/Non_Coding_Element_Evolution/3-Masking/2-RepeatMasker/Lser_r
Lexi /home/pedro/Non_Coding_Element_Evolution/3-Masking/2-RepeatMasker/Lexi_r
*Pazu /home/pedro/Non_Coding_Element_Evolution/3-Masking/2-RepeatMasker/Pazu_
```

Alignment

```
cd /home/pedro/Non_Coding_Element_Evolution/4-ProgressiveCactus_WGA

source /home/pedro/Programs/cactus-bin-v2.6.4/cactus_env/bin/activate

# Running
cactus NEW_WGA Cactus_file.txt Calliphoridae_WGA.hal --maxCores 20 --logCrit

# After completion
deactivate

# Turn it to MAF format (one MAF file for each scaffold in the genome)
export PATH=/home/pedro/Programs/cactus-bin-v2.6.4/bin:${PATH}

hal2mafMP.py --numProc 5 --splitBySequence --refGenome Chom --noAncestors --

# Better names
for i in *maf; do mv $i ${i##Calliphoridae_WGA_}; done

# Remove second line of the files, because it has a tree with the ancestors
for i in *maf; do sed -i -e 2d $i; done

# Keep the files elsewhere
mkdir MAF_files_Chom_ref
mv *.maf MAF_files_Chom_ref/
```

5. PhastCons

I'll use the PHAST software (more specifically, PhastCons) to find conserved elements in the WGA of Calliphoridae species.

IMPORTANT REFERENCES:
http://compgen.cshl.edu/phast/resources.php
http://compgen.cshl.edu/phast/phastCons-HOWTO.html#paper
https://github.com/tsackton/ratite-genomics/tree/master/04_wga/02_ce_id

5.1. Installation

```
# Clapack v3.2.1 (required for PHAST)
cd /home/pedro/Programs

wget http://www.netlib.org/clapack/clapack.tgz
tar -xf clapack.tgz

cd CLAPACK-3.2.1/
cp make.inc.example make.inc && make f2clib && make blaslib && make lib

# PHAST v1.6
scp -P 2205 phast-1.6.tar.gz pedro@143.107.244.181:/home/pedro/Programs
tar -xf phast-1.6.tar.gz

cd phast-1.6/src
make CLAPACKPATH=/home/pedro/Programs/CLAPACK-3.2.1
```

5.2. Running

In the PHAST folder I have the WGA in MAF format (with a *Co. hominivorax* reference and without the ancestral genome estimates - see Cactus section), the *Co. hominivorax* genome gff file, and the tree I used for the WGA.

```
cd /home/pedro/Non_Coding_Element_Evolution/5-PHAST

# Now, to make the conserved and non-conserved models and predict the elemen

cd /home/pedro/Non_Coding_Element_Evolution/5-Phast/1-Chom_per_Scaffold

cp /home/pedro/Non_Coding_Element_Evolution/3-Masking/2-RepeatMasker/Chom_ma

# Making the sequence names equal to the MAF files
sed -i 's/Chom_/Chom.Chom_/g' Chom_masked.fasta

# Split fasta (found it here: https://gist.github.com/astatham/621901)
cat Chom_masked.fasta | awk '{if (substr($0, 1, 1)==">") {filename=(substr($

# I did this, because I changed the sequence headers earlier, but it's best
for i in *.fa; do mv $i ${i##Chom.};done

# Now, do the splitting - chunks.sh
cd /home/pedro/Non_Coding_Element_Evolution/5-Phast/2-MAF_chunks

for MAF in /home/pedro/Non_Coding_Element_Evolution/4-ProgressiveCactus_WGA/

do
    SCAFF=${MAF##/home/pedro/Non_Coding_Element_Evolution/4-ProgressiveCactu
    SCAFF=${SCAFF%%.maf}

    msa_split $MAF --in-format MAF --refseq /home/pedro/Non_Coding_Element_E
done

# GC content
# Get the average GC content of the WGA to use it in the model prediction
cd /home/pedro/Non_Coding_Element_Evolution/5-Phast/3-Models

mkdir basecomp
cd basecomp

export PATH=/home/pedro/Programs/hal/bin:${PATH}

for spp in $(halStats --genomes /home/pedro/Non_Coding_Element_Evolution/4-P

#halStats --basecomp -> the ouptut is fraction_of_As fraction_of_Gs fraction

# I don't need the information on the ancestral genomes, I'll put it away fo
mkdir Ancestral_genomes
mv Anc* Ancestral_genomes/

# Average CG content in my genomes
# I need columns 2 and 3, because they are G and C
cat *basecomp | awk '{SUM+=$2;SUM+=$3;print SUM/11}' | tail -n 1 > GC # Aver

# Model prediction
cd /home/pedro/Non_Coding_Element_Evolution/5-Phast/3-Models

mkdir Trees

# model_predict.sh
for MAF in /home/pedro/Non_Coding_Element_Evolution/5-Phast/2-MAF_chunks/*ss

do
    NAME=${MAF##/home/pedro/Non_Coding_Element_Evolution/5-Phast/2-MAF_chunk
    NAME=${NAME%%.maf}

    phyloFit --tree /home/pedro/Non_Coding_Element_Evolution/4-ProgressiveCa

    phastCons --gc 0.26 --estimate-trees Trees/$NAME --msa-format SS $MAF ir
done

# Organizing the initial models
cd /home/pedro/Non_Coding_Element_Evolution/5-Phast/3-Models
mkdir Initial_models
mv init.* Initial_models

# General organization
mv basecomp 1-Basecomp
mv Initial_models 2-Initial_models
mv Trees 3-Trees

# I need to filter the models, leaving only the ones with all species (becau
cd 3-Trees

mkdir not_all_spp

for MOD in *mod
do
    TREE=$(tail -n 1 $MOD)
    SPP='Cmeg'*'Cput'*'Chom'*'Pazu'*'Lser'*'Lcup'*'Lexi'*'Cvom'*'Agra'*'Bpar

    if [[ $TREE != *$SPP* ]]
```

```
    then
        mv $MOD not_all_spp/
    fi
donefor MOD in *mod
do
    TREE=$(tail -n 1 $MOD)
    SPP='Cmeg'*'Cput'*'Chom'*'Pazu'*'Lser'*'Lcup'*'Lexi'*'Cvom'*'Agra'*'Bpar

    if [[ $TREE != *$SPP* ]]
    then
        mv $MOD not_all_spp/
    fi
done

# Now I average the models to do a final run
cd /home/pedro/Non_Coding_Element_Evolution/5-Phast/3-Models
mkdir 4-Average_models
cd 4-Average_models

ls ../3-Trees/*.cons.mod > cons.txt
phyloBoot --read-mods '*cons.txt' --output-average ave_cons.mod

ls ../3-Trees/*.noncons.mod > noncons.txt
phyloBoot --read-mods '*noncons.txt' --output-average ave_noncons.mod

# Naming the ancestral nodes in the model (will need it afterwards)
tree_doctor --name-ancestors ave_cons.mod > ave_cons_named.mod
tree_doctor --name-ancestors ave_noncons.mod > ave_noncons_named.mod

# Final run with the averaged models
cd /home/pedro/Non_Coding_Element_Evolution/5-Phast/3-Models
mkdir 5-Final_predictions
cd 5-Final_predictions

mkdir Elements Scores

for MAF in /home/pedro/Non_Coding_Element_Evolution/5-Phast/2-MAF_chunks/*ss
do
    NAME=${MAF##/home/pedro/Non_Coding_Element_Evolution/5-Phast/2-MAF_chunk
    NAME=${NAME%%.ss}

    phastCons --gc 0.26 --most-conserved Elements/$NAME.bed --score --msa-fo
done

cd Elements

cat Chom*.bed | sort -k1,1 -k2,2n > most-conserved.bed

# 818,864 elements found
```

5.3. Element filtering

I used bedtools to filter my conserved elements in respect to the whole genome

5.3.1. Installing bedtools and bedops

```
# Now i need bedtools to get the elements that are not within genes
# bedtools v2.30.0
scp -P 2205 bedtools-2.30.0.tar.gz pedro@143.107.244.181:/home/pedro/Programs
tar -xvf bedtools-2.30.0.tar.gz

cd bedtools2
make

# BEDOPS v2.4.41
wget https://github.com/bedops/bedops/releases/download/v2.4.41/bedops_linux_

cd bedops_v2.4.41
```

5.3.2. Filtering

```
# Moving on
mkdir /home/pedro/Non_Coding_Element_Evolution/5-Phast/4-Conserved_elements
cd /home/pedro/Non_Coding_Element_Evolution/5-Phast/4-Conserved_elements

mv ../3-Models/5-Final_predictions/Elements/most-conserved.bed ./all_elements

# I also merged the element predictions that were 5bp or less away from each
bedtools merge -i all_elements.bed -d 5 -c 4,5,6 -o distinct > merged_element

# Filtering elements with at least 250bp
awk '($3 - $2) >= 250' merged_elements.bed > 250bp_elements.bed # 44,253 (25(

# Now, to keep only the elements that are non-exonic, I need to use the Co. h
mkdir Chom_ref
cd Chom_ref

 cp /home/blowflies/Gene_families/00-Genomic_data/05-GFF/Chom_GFF.gtf .

sed -i 's/Chr/Chom_Chr/g' Chom_GFF.gtf # just to make the gff equal to the pr

# Getting just the exon entries
grep "exon" Chom_GFF.gtf > Chom_exons.gtf

# Convert to bed
gtf2bed < Chom_exons.gtf > Chom_exons.bed

# When I tried to use bedops on the entire genome gff, I was warned that ther
sed -r 's/\;.*//' Chom_exons.bed  > Chom_exons_corrected.bed

# Get non-exonic elements
cd ..
bedtools intersect -a 250bp_elements.bed -b Chom_ref/Chom_exons_corrected.bed
```

All	Merged	250bp	Non-exonic
818,864	797,186	44,253	20,785

6. Element processing

6.1. Sequences in the other species

Now that I identified the CNEEs with the Co. hominivorax reference, I need to find them in the WGA for the other species. Afther that I can align them and make the following analyses.

```
# Finding the CNEEs in the other species
mkdir /home/pedro/Non_Coding_Element_Evolution/6-Element_processing
cd /home/pedro/Non_Coding_Element_Evolution/6-Element_processing

# hal2bed.sh
for SP in Cmeg Cput Pazu Lser Lcup Lexi Cvom Agra Bpan
do
    halLiftover /home/pedro/Non_Coding_Element_Evolution/4-ProgressiveCactus_
done

for i in *bed; do cat $i | sort -k1,1 -k2,2n > ${i%%.bed}_sorted.bed; done
```

Some Co. hominivorax CNEEs had more than one correlate in other species (a few bases apart), so merging using bedtools didn't work well. So I used a costum python script to do the merging (bed_file_organizer.py).

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

'''
PROGRAM: bed_file_organizer.py
'''
import glob
import sys
import getopt
#------------------------------------------------------------------------------
def main():
    pwd = arg()

    if pwd == None:
        usage()


    bed_files = glob.glob(f'{pwd}/*bed') # getting all files

    for bed in bed_files:

        start = None
        end = None
        ref = None
        merge={}

        file = open(bed,'r')
        out = open(f'{bed}_merged.bed', 'w')

        for l in file:
            info = l.split()

            chrom = info[0]
            start = int(info[1])
            end = int(info[2])
            ref = info[3]
            n = info[4]
            strand = info[5]

            name=chrom+ref

            if name not in merge:
                merge[name] = [chrom, [start], [end], ref, n, strand]
            else:
                merge[name][1]+=[start]
                merge[name][2]+=[end]

        for i in merge:
            s = min(merge[i][1])
            e = max(merge[i][2])

            out.write(f'{merge[i][0]}\t{s}\t{e}\t{merge[i][3]}\t{merge[i][4]

        file.close()
        out.close()

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

    arg_1 = None

    for opt, arg in opts:
        if opt in ['-p']:
            arg_1 = arg

    return arg_1
#------------------------------------------------------------------------------
def usage():
    sys.exit('Usage:\n'
           '\t-p: files directory pathway\n')
#------------------------------------------------------------------------------
if __name__ == '__main__':
    main()
```

```
cd /home/pedro/Non_Coding_Element_Evolution/6-Element_processing

python3 bed_file_organizer.py -p .

# Fixing the output names
for i in *merged*; do mv $i ${i%%.bed_merged.bed}_merged.bed; done

# Organizing
mkdir Unsorted Sorted Merged
mv *merged.bed Merged/
mv *sorted.bed Sorted/
mv *.bed Unsorted/

mkdir 1-Bed_other_spp
mv * 1-Bed_other_spp
```

Species	# CNEEs
Agra	16,562
Bpan	16,628
Cmeg	21,285
Cput	19,918
Cvom	16,599
Lcup	14,913
Lexi	21,710
Lser	15,748
Pazu	18,726

Now that I have the bed files for each species, I can generate fasta files:

```
mkdir /home/pedro/Non_Coding_Element_Evolution/6-Element_processing/2-Fasta_
cd /home/pedro/Non_Coding_Element_Evolution/6-Element_processing/2-Fasta_file

# Chom (because this is the reference, I'm using the final CNEEs I identifie
bedtools getfasta -fi /home/pedro/Non_Coding_Element_Evolution/3-Masking/2-Re

# Other species
for SP in Cmeg Cput Pazu Lser Lcup Lexi Cvom Agra Bpan; do bedtools getfasta

# I'll put each species name in the fasta headers to make it easier fo when
for SP in Agra Bpan Chom Cmeg Cput Cvom Lcup Lexi Lser Pazu; do sed -i "s/::
sed -i 's/,/_/g' *fasta # the comma would be a problem afterwards

# Now, I need one file for each CNEE
mkdir 1-Species_files 2-CNEE_files
mv *fasta 1-Species_files

cd 2-CNEE_files
cut -f4 /home/pedro/Non_Coding_Element_Evolution/5-Phast/4-Conserved_element

# Fixing things that might cause trouble with grep
sed -i 's/$/::/' CNEE_list.txt
sed -i 's/,/_/g' CNEE_list.txt
sed -i 's/^Chom/INIChom/' CNEE_list.txt

# Also fixing the species files
sed -i 's/>Chom/>INIChom/' ../1-Species_files/*fasta

# Now, the CNEE files
while read line; do seqkit grep -r -p "$line" ../1-Species_files/*fasta >> $

# Making the fasta files a bit better
sed -i 's/::/@/2' *fasta
sed -i 's/>.*::/>/' *fasta

# Also changing their name
for i in *fasta; do mv $i ${i##INI}; done
```

6.2. Filtering fasta files

Now everything worked! I have to filter these files so I can align them, and these are my criteria:

- No duplicates - just 1 sequence per species
- Sequences have to be <2x and >0.5x the *Co. hominivorax* reference
- All species have to be present

```
cd /home/pedro/Non_Coding_Element_Evolution/6-Element_processing/2-Fasta_file
mkdir Dupes No_dupes

# no_dupes.sh - selecting the ones with no duplicate species
for i in *.fasta
do
        a=$(grep -c ">" $i) # number of sequences
        b=$(grep ">" $i | sed 's/>//' | sed -r 's/@.*//' | uniq | wc -l) # nu

        if [ $a == $b ] # avoid duplicates (seq = spp)
        then
            mv $i No_dupes
        fi
done

mv *fasta Dupes # just what is left

# size_filter.py - filtering for size (remove sequences that are more than 2x
cd No_dupes
for i in *fasta; do python3 ../size_filter.py $i; done

cd ..
mkdir Size_filtered
mv No_dupes/*filt*fasta Size_filtered
```

Just for the record:

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

'''
PROGRAM: size_filter.py
'''

from Bio import SeqIO
import sys

def main(input_file):
    reference_length = None

    # Primeira iteração para encontrar o tamanho da sequência de referência
    for record in SeqIO.parse(input_file, "fasta"):
        if "Chom" in record.id:
            reference_length = len(record.seq)
            break

    if reference_length is None:
        print("Sequência de referência não encontrada em", input_file)
        return

    # Segunda iteração para escrever as sequências que atendem ao critério
    output_file = input_file.replace(".fasta", "_filtered.fasta")
    with open(output_file, "w") as output:
        for record in SeqIO.parse(input_file, "fasta"):
            if reference_length / 2 <= len(record.seq) <= reference_length *
                SeqIO.write(record, output, "fasta")

if __name__ == "__main__":
    if len(sys.argv) != 2:
        print("Uso: python3 size_filter.py arquivo.fasta")
        sys.exit(1)

    input_file = sys.argv[1]
    main(input_file)
```

Back to the filtering process:

```
cd /home/pedro/Non_Coding_Element_Evolution/6-Element_processing/2-Fasta_file
mkdir /home/pedro/Non_Coding_Element_Evolution/6-Element_processing/2-Fasta_

# n_seq_filter.sh - filtering for number of sequences
for i in *filt*.fasta
do
        a=$(grep -c ">" $i) # number of sequences

        if [ $a -eq 10 ] # all species
        then
            cp $i /home/pedro/Non_Coding_Element_Evolution/6-Element_process
        fi
done
# 3851
```

6.3. Alignments

I need to align each CNEE before I test for the acceleration.

```
mkdir /home/pedro/Non_Coding_Element_Evolution/6-Element_processing/3-Alignme
cd /home/pedro/Non_Coding_Element_Evolution/6-Element_processing/3-Alignment

# mafft_aln.sh
for FASTA in /home/pedro/Non_Coding_Element_Evolution/6-Element_processing/2-
do
    FILE=${FASTA##/home/pedro/Non_Coding_Element_Evolution/6-Element_process

    mafft-ginsi --thread 10  --maxiterate 1000 --adjustdirectionaccurately --
done

sed -i 's/_R_//' * # because of the --adjustdirectionaccurately option
```

7. PhyloACC

Now I'll PhyloACC to find elements that are accelarated in *Co. hominivorax* and/or *Pr. azurea*. For this I need the aligned CNEEs, a neutral model (I'll use the averaged non-conserved model for the WGA).

7.1. Installing

```
# PhyloACC v2.2.0
conda create -n phylo -c bioconda
conda activate phylo
conda install phyloacc

# astral v5.7.1
cd /home/pedro/Programs
wget https://github.com/smirarab/ASTRAL/archive/refs/tags/v5.7.1.tar.gz
tar -xzvf v5.7.1.tar.gz
cd ASTRAL-5.7.1
bash make.sh
```

7.2. Running

7.2.1. Coalescence tree

PhyloACC requires a tree in coalescent units, and I'll have to estimate that (and it has to have the same topology as the tree in the Phast model). I estimated it using the entire phylogeny (the I used for the WGA)

```
# Doing it on Darwin
cd /home/martins/Programs/ASTRAL-5.7.1

scp -P 2205 pedro@143.107.244.181:/home/pedro/Non_Coding_Element_Evolution/4

java -jar astral.5.7.1.jar -i Calliphoridae_tree.tre -o Calliphoridae_astral

# The root in the Astral tree is non-significative, so I rerooted in in R
astral <- read.tree(text="(cbez,(cmeg,((calb,cput)0.67:0.2876820724517809,((

a <- reroot(astral,21)

astral_new <- drop.tip(a, c("clop","loch","cbez","calb"))

write.tree(astral_new)

"((((Cmeg:NaN,Cput:NaN)0.67:0.2876820725,Chom:NaN)0.67:0.2876820725,Pazu:NaN

# I had to move the branches (respecting the topology), so the species were

"((((Cmeg:NaN,Cput:NaN)0.67:0.2876820725,Chom:NaN)0.67:0.2876820725,Pazu:NaN

# Back to Darwin
cd /home/martins/PhyloACC
nano Astral_tree.tre
((((Cmeg:NaN,Cput:NaN)0.67:0.2876820725,Chom:NaN)0.67:0.2876820725,Pazu:NaN)(

sed -i 's/:NaN//g' Astral_tree.tre

# Also, I removed the ancestor "names" (0.67), because this was also an issue
sed -i 's/0\.67:/:/g' Astral_tree.tre

((((Cmeg,(Calb,Cput)Calb-Cput:0.2876820725)Cmeg-Calb:0.2876820725,Chom)Cmeg-(
```

7.2.2. Other important files

```
cd /home/martins/PhyloACC

mkdir Aln

# Bringing alignments
scp -P 2205 pedro@143.107.244.181:/home/pedro/Non_Coding_Element_Evolution/6

sed -i 's/@.*//' *

# Getting the model file
scp -P 2205 pedro@143.107.244.181:/home/pedro/Non_Coding_Element_Evolution/5
```

7.2.3. Running

```
# PhyloACC
cd /home/martins/PhyloACC
conda activate phylo

# Preparing the data
phyloacc.py -d /home/martins/PhyloACC/Aln -m ave_noncons_named.mod -l Astral_

# Running - Option 1
# snakemake -p -s /home/martins/PhyloACC/PhyloBoth/phyloacc-job-files/snakema

# Running - Option 2
max_attempts=100 # Maximum number of attempts

for numero in {1..78}; do # Loop for batches 1-78
    echo "Tentando análise para número $numero:"

    # Loop to try many times (if needed)
    for ((i=1; i<=max_attempts; i++)); do
        # Actual command line
        PhyloAcc-GT PhyloBoth/phyloacc-job-files/cfgs/${numero}-gt.cfg > Phy

        # Did the command work? (0 = success)
        if [ $? -eq 0 ]; then
            echo "Análise bem-sucedida para número $numero!"
            break  # Break loop if it works
        else
            echo "Erro de segmentação. Tentando novamente para número $numero
            sleep 1  # Wait a second before trying again (optional)
        fi
    done
done

# Post-processing the results
phyloacc_post.py -i PhyloBoth
```

Note:

The *phyloacc.py* command creates batches of alignments to allow the next steps and the snakemake files to run each batch. But it didn't work as expected, as some batches resulted in segmentation faults, which not only haltered their execution, but the execution of all batches.

The authors suggested that I run each batch individually and then do the final step afterwards. Just as a reference, I left the original snakemake command line as a comment (Running - Option 1), but what was actually done is the the second one (Running - Option 2). Renan helped in the loop!

7.2.4. Back to Rosalind

```
scp -r -P 2205 /home/martins/PhyloACC pedro@143.107.244.181:/home/pedro/Non_

mv PhyloACC/ 7-PhyloAcc/

# Some organization
cd 7-PhyloAcc
mkdir Slurm_files
mv phylo*.* Slurm_files
```

I simplified the elem_lik.txt result file (`/home/pedro/Non_Coding_Element_Evolution/7-PhyloAcc/PhyloBoth/results/elem_lik.txt`). But I did this locally, using Excel. I uploaded the files this directory: `home/pedro/Non_Coding_Element_Evolution/7-PhyloAcc/Result_files` . Briefly, I filtered all elements for which M1 was the best fit. Then, I split this filtered file into smaller ones: accelerated in *Co. hominivorax*; accelerated in *Pr. azurea*; accelerated in both; accelerated in none.

8. Closest genes

Now that I identified the elements that have accelerated evolution, I wanted to find the genes whose start is closest to these elements.

8.1. Bed files

I filtered the original bed file with all the elements to leav only the interesting ones.

```
cd /home/pedro/Non_Coding_Element_Evolution
mkdir 8-Closest_genes
cd 8-Closest_genes
mkdir 1-Filtered_beds

# Bed file with all elements found
cp ~/Non_Coding_Element_Evolution/5-Phast/4-Conserved_elements/merged_element

# Lists of accelerated elements
cut -f2 ~/Non_Coding_Element_Evolution/7-PhyloAcc/Result_files/M1_both.txt >
cut -f2 ~/Non_Coding_Element_Evolution/7-PhyloAcc/Result_files/M1_all.txt > I
cut -f2 ~/Non_Coding_Element_Evolution/7-PhyloAcc/Result_files/M1_Chom.txt >
cut -f2 ~/Non_Coding_Element_Evolution/7-PhyloAcc/Result_files/M1_Pazu.txt >
cut -f2 ~/Non_Coding_Element_Evolution/7-PhyloAcc/Result_files/M1_none.txt >

# Some editions
sed -i -e 1d *txt
sed -i 's/_filtered_aln//' *txt

sed -i 's/,/_/g' merged_elements.bed

# New bed files
for i in *txt
do
    while read line
    do
        grep -w $line merged_elements.bed >> ${i%%.txt}.bed
    done < $i
done

for i in Elem*bed; do bedtools sort -i $i > ${i%%.bed}_sorted.bed; done
```

8.2. Transcription start sites

First, I need the entire *Co. hominivorax* gff, so I can extract the transcription start sites.

```
mkdir /home/pedro/Non_Coding_Element_Evolution/8-Closest_genes/2-Transcripti
cd /home/pedro/Non_Coding_Element_Evolution/8-Closest_genes/2-Transcription_

# I extracted the first CDS from each gene, so I knew what is the transcript

cp /home/pedro/Non_Coding_Element_Evolution/5-Phast/4-Conserved_elements/Chor

bedtools sort -i Chom_exons_corrected.bed > Chom_exons_sorted.bed

sed -i 's/\"//g' Chom_exons_sorted.bed
sed -i 's/transcript_id//' Chom_exons_sorted.bed

bedtools groupby -i Chom_exons_sorted.bed -g 4 -c 1,2,3,4,5,6 -o first > Chor

# Now I want jusrt the first site
awk '{print $2, $3, $3 + 1, $5, $6, $7}' Chom_first_exon.bed > Chom_transcri

# For some reason, the tabs were lost
sed -i "s/ /\t/g" Chom_transcription_start.bed
```

8.3. Closest genes

```
mkdir /home/pedro/Non_Coding_Element_Evolution/8-Closest_genes/3-Closest
cd /home/pedro/Non_Coding_Element_Evolution/8-Closest_genes/3-Closest

for i in ../1-Filtered_beds/Elem*sorted.bed; do bedtools closest -d -a $i -b

for i in *; do mv $i ${i%%_sorted.bed_closest.bed}_closest.bed; done

# Filtering the closest gene and the distance to the closest CNEE
for i in *bed
do
    echo -e "CNEE\tGene\tDistance" > ${i%%.bed}_list.txt
    cut -f4,10,13 $i >> ${i%%.bed}_list.txt
done
```

9. Transcription factors

I checked whether the accelerated CNEEs were enriched for transciption factor binding motifs. And then, I checked for GO terms associated to genes that are controlled by TFs binding to these motifs.

9.1. Getting the CNEEs

```
cd /home/pedro/Non_Coding_Element_Evolution/8-Closest_genes/1-Filtered_beds

bedtools getfasta -fi /home/pedro/Non_Coding_Element_Evolution/3-Masking/2-Re
```

9.2. TF binding motifs

MEME Suite - AME: https://meme-suite.org/meme/tools/ame

Image Not Showing

Possible Reasons

- The image was uploaded to a note which you don't have access to
- The note which the image was originally uploaded to has been deleted

Learn More →

9.3. GO enrichment for genes with the TF binding motifs

MEME Suite - GOMo: https://meme-suite.org/meme/tools/gomo

Image Not Showing

Possible Reasons

- The image was uploaded to a note which you don't have access to
- The note which the image was originally uploaded to has been deleted

Learn More →