All scripts have been specifically designed to work with the GAGA dataset and its name/file system. I have renamed all sequences in the original dataset as such: “AameOR190\_A\_GAGA-0001\_Acromyrmex-ameliae”.

There’s still a need to enter some scripts and modify some values (list of species when preparing the data, thresholds for statistics, path to scripts/programs if needed, partitions to run on, etc…), but these corrections are minimal.

Start by creating the main directory where you want to work, its name will be used in some scripts as ‘clade’ to name some files and sub-directories. Then all scripts should be run from within that directory unless told otherwise.

# Preparing the data

[prepare.py](https://drive.google.com/file/d/1O2O2T1bKI_3lUB_sm9DNgPiwgiF6OAYA/view?usp=drive_link), 2 variables can be set by modifying the script:

GAGAdir => directory in which we have the fasta files of all ORs subfamilies. Emphasize on the fact that the files used as input are the subfamilies fasta (e.g. ‘Subfamily\_\*.cds.fasta’), and the script only takes from each the sequences belonging to the species we define in species. Also, because the 9-Exon has been divided (in 2 and then 6), make sure you’re only taking from one version of the division by renaming the files you don’t want to use (just add something after ‘.fasta’ to make the script skip the file). By default now, it takes the 9-Exon into 6 groups. Technically no need to change the path unless working with different subfamilies, default set as:

/data/home/privman/ypellen/ORs/GAGA\_Final\_OR\_annots/GAGA\_final\_subfamilies/

species => list of species to study, identified with their GAGA id

There’s a few checks done to know which sequences are taken or not:

* Must not have any internal stop codons
* Its length must be a multiple of 3, as we want to build codon alignments it would be a problem for Guidance2 if it’s not divisible by 3
* Removes final stop codon if it’s still present

Some subfamilies might have less than 4 sequences, which is the minimum for Guidance2 to work. In such cases, you can merge the small subfamily with the closest one based on this phylogeny [GAGA\_final\_subfamilies.MAFFTeinsi.faa.tree.png](https://drive.google.com/file/d/1IcURdRGkNVcwKgGUlru75AcL4ihj9-AC/view?usp=sharing).

You will end up with one directory for each subfamily, with a fasta file inside containing all sequences for the species you gave that are part of this subfamily.

To check the number of sequences: egrep -c ">" sf\_\*/\*.cds.trim.fna

# Guidance2

[guidance.sh](https://drive.google.com/file/d/18sHSjhcBwQQTBtmwAlUaTXD0tSzd5puF/view?usp=drive_link) and [guidance\_correction.py](https://drive.google.com/file/d/11_OjStaJxa0O6qrRU2trP8t4bId1y_2g/view?usp=drive_link)

The script is called as: bash PATH/TO/guidance\_multi.sh <function>

Masking and trimming rely on the python script, the path has to be updated in the bash script if you move it.

Note that you can run any function without risk of deleting anything because it always checks before if something is still running or if it has the necessary flag file.

The different functions are:

1. align : actual run of *Guidance2*, building a codon alignment using *PRANK*. *Guidance2* creates the file ENDS\_OK in case of a success, used as a flag to run further functions.
2. mask : masking the codon alignment based on the residues score from <subfamily>/guidance2/MSA.PRANK.Guidance2\_res\_pair\_res.scr, with a loop on the threshold value, starting at 0.8, so that we don’t mask more than 20% of the alignment. If an alignment can’t meet the requirement of 20% masked residues maximum even with a threshold at 0.1, the orthogroup will be discarded from the analysis. With the python script, checks if some sequences become identical after masking, creates a file called G\_checkup.$sf.txt with the % of masked residues and gaps in the alignment. Creates G\_CORREC if it works, used as a flag to run further functions.
3. trim : removing the bad sequences. Add the names of the sequences that have been removed to G\_checkup.$sf.txt. Creates G\_TRIM if it works, used as a flag to run further functions.
4. clean : removing all intermediate files that are no longer useful to make some space, basically keeping the alignments and scoring files from *Guidance2*.

# RAxML

[raxml.sh](https://drive.google.com/file/d/1TH1X4pjmMsN0ayeofi2i6t40HcVBwMRa/view?usp=drive_link), nothing to change here.

*RAxML* is runned with the masked amino acid alignment created with *Guidance2*, using the PROTCATLG model and 123456 as seed for the initial tree, and 100 bootstraps.

It runs only if the flag file G\_TRIM has been created.

If it runs successfully, the tree file RAxML\_bestTree.$sf is created and used as a flag to run the following analysis.

It also removes all intermediate files to make some space.

# Godon

[godon.sh](https://drive.google.com/file/d/1hiS_hTntYPkf5Gmcql0MErV_wfUUXQYJ/view?usp=drive_link), nothing to change here. Run ‘tail -n 1 sf\_\*/godon/godonBS.\*.out’ to check if it ended OK, should finish with a “runtime” line.

~~When Godon is done, run~~ [~~godon\_stats.Hive.sh~~](https://drive.google.com/file/d/1E5pmg9dbetO_KoR_xx50jMtoWaUCKZAJ/view?usp=drive_link) ~~to do the FDR correction and get a phylogenetic tree with branches under positive selection marked. It relies on~~ [~~branchSite.lrt.fdr.r~~](https://drive.google.com/file/d/1ApRXNuDiUQ6gtssea3fXPMbGEPmtK50b/view?usp=drive_link) ~~for the FDR correction and statistical test (CSV file clade\*.godonBS.all.fdr.csv in your main working directory).~~

ALL GODON STATS ONLY RUNS LOCALLY

~~Run~~ [~~godon\_stats.sh~~](https://drive.google.com/file/d/12bKtsGpoMPxciFqlIoC-4I9EwoGiTSVh/view?usp=drive_link)~~, which uses~~ [~~godon\_correction.py~~](https://drive.google.com/file/d/1arrHDZ_FBapFjUJes5if86Fx1wVw8bPf/view?usp=drive_link) ~~to build the gene trees with internal branch names numbered according to the~~ *~~Godon~~* ~~output.~~

# Generax

If not already done, start by numbering the internal branches of the species tree, in\_tree is your input species tree and out\_tree is the name of the output tree:

python PATH/TO/generax.py -f treelabel -it <in\_tree> -ot <out\_tree>

The branches will be named ‘spbranchXX’ and can be viewed using FigTree 'Branch Labels > Display > label'.

The original GAGA species tree is [GAGA\_phylogeny\_dated-1.png](https://drive.google.com/file/d/11sYoa1wFbnEzng0udVaxsW01xkWbTnnI/view?usp=drive_link), then you have this version to see the numbers for the internal branches [GAGA\_sp\_tree.branch\_numbers.jpg](https://drive.google.com/file/d/1x5jAKPi4XBBBD229gizbRRl-DV16vk2Y/view?usp=drive_link).

🚨 The python module *ete3* is sometimes causing trouble and doesn’t run on Hive. Move the data locally for the next part.

Run [generax\_local.sh](https://drive.google.com/file/d/184JrP2fQ125Em9r0x1rf3T3Wnv216Yjo/view?usp=drive_link), which relies on [generax.py](https://drive.google.com/file/d/10ci7StJA5fphkPlPI0IHx3Spu9K7WKGp/view?usp=drive_link), to prepare the files used as input by *Generax*.

When the *Generax* files are created, move everything back to Hive and run [generax\_Hive.sh](https://drive.google.com/file/d/1Fwadup5ytJCUIAKO_vO-NNIMzq0sGz2c/view?usp=drive_link). This just runs *Generax*, do it from within the main working directory.

When *Generax* is done, run [mapping\_table.py](https://drive.google.com/file/d/1hIflWCOyiK55vhcPSwbjHY_dhqjWBMWj/view?usp=drive_link) to build a table showing all statistics for each gene branch, and the species branch it’s mapped on.

# Hypergeometric test

You can run it for as many branches as you want, then all the tests in one go with [hypergeometric\_test.py](https://drive.google.com/file/d/19-7KLNPohDGIarMpkD46hrnF-X36IfCs/view?usp=drive_link). Within it, you can change the threshold for the pvalue from Godon.