

PosiGene User Guide

Table of Contents

Technical prerequisites	2
Installation.....	3
Abbreviations	3
Overview.....	3
Input	4
Anchor species.....	4
Target species.....	4
Input files.....	5
HomoloGene based ortholog assignment.....	5
Reference based ortholog assignment.....	5
Symbol based ortholog assignment	5
File formats.....	6
Data sources	7
Examples.....	10
Example I: Only HomoloGene species	11
Example II: HomoloGene and non-HomoloGene species	11
Example III: Reference based ortholog assignment	12
Example IV: Symbol based ortholog assignment	13
Output	15
Result tables	15
Visualizations	16
Jalview visualizations.....	16
PNG images	17
Trees	19
Context species.....	21
Modularization – How to accelerate (re-)analysis	22
Examples.....	23
Different evolutionary branches	23
Different anchor species	24
Adding non-HomoloGene species	24
PosiGene arguments	24

Obligatory arguments.....	25
CDS input files.....	25
Optional arguments.....	25
Frequently used arguments	25
Fine-tuning arguments	26
Appendix.....	28
Appendix I: Used Perl modules (part of the package).....	28
Appendix II: Used Software (part of the package)	29
Appendix III: HomoloGene species.....	29

Technical prerequisites

Absolutely needed to run PosiGene:

- A Linux 64-Bit system
- A Perl installation
 - We recommend having at least version 5.10.0 installed, because we tested PosiGene on different Perl installations between version 5.10.0 and 5.22.0. However, it will probably also run on lower versions – but without guarantee.
 - You can download Perl, e.g., here: https://www.perl.org/get.html#unix_like

Optional:

- GD-Library
 - PosiGene can create alignment visualizations for every gene that was tested for positive selection (with selected sites and functional domains highlighted). If the GD-Library is installed images in PNG format are automatically created and referenced in the result file.
 - You can download the library here: <https://github.com/libgd/libgd/releases>
- Java
 - PosiGene also offers alignment visualization via Jalview, a graphical, java-based alignment editor that is part of the package you downloaded. The files that store the information necessary for the visualization are created and referenced in the result files whether Java is installed or not. If you want to use that kind of visualization you can install Java later, too.
- Bioperl
 - All Perl modules that are needed by PosiGene are part of the package you downloaded and will be used if you do not have the respective module installed. Therefore at the moment (February 2016) and in the near future any standard Perl installation should be sufficient to run PosiGene. However, we cannot exclude the possibility of version conflicts somewhere later in the future. If you want to install the latest versions of all needed modules on your computer you can either:

- Install the modules listed in Appendix I: Used Perl modules (part of the package) yourself
- or call PosiGene with the `–install_modules` parameter:

```
perl PosiGene.pl –install_modules
```

Installation

No further installation or configuration needed.

Abbreviations

PSG – Positively Selected Gene

CDS – Coding Sequence

LCA – Last Common Ancestor

NCBI – National Center for Biotechnology Information

BBB – Best Bidirectional Blast

Overview

PosiGene is computer program to scan for positively selected genes (PSGs) on a genome-wide scale based on interspecies sequence comparison and codon based methods. What you need to run PosiGene is:

- An idea in which evolutionary branch you want to test for PSGs.
 - This means a today living species, e.g. “human” or “Tibetan Antelope”. It could also be an ancestral lineage, e.g. “the last common ancestor (LCA) of human and chimp” or “the LCA of all mole-rats”.
- Coding sequences (CDSs) from that or those species that are scanned for PSGs.
 - In case of our example “LCA of human and chimp” that would be of course human and chimp.
- CDSs from a set of comparison species that you must compile according to the evolutionary branch you want to test.
 - In the “LCA of human and chimp” example that would be at least their closest relative, the gorilla, and one outgroup – probably another old world monkey.
- One of the included species that you select to serve as “anchor” for the analysis
 - In case of the example “LCA of human and chimp” human would be a good choice, because it has the largest and most extensively annotated set of CDS.

You can put all this in one command line call, in case of “LCA of human and chimp” e.g.:

```
perl PosiGene.pl - target_species=Homo_sapiens, Pan_troglodytes
- homologue_species=data/Homo_sapiens.gbk, data/Pan_troglodytes.gbk, data/Macaca_mula
tta.gbk - non_homologue_species=Gorilla_gorilla:/usr/local/my_seq_data/Gorilla.fa
- anchor_species=Homo_sapiens
```

(You would need Gorilla.fa file from an external source)

What PosiGene will produce for you from that information is:

- A list of genes ranked by their probability to be positively selected including positively selected sites, d_N/d_S ratios, references to alignment visualizations etc.

This User Guide explains how the input should look like, how to dictate your idea about what should be tested to the program, where you can get data for comparison species from and how the output is structured.

Input

The arguments (and their short form) that are always needed to perform a full PosiGene run are:

- `-anchor_species (-as)`
- and `-target_species (-ts)`

which both expect exactly one `(-as)` respectively at least one `(-ts)` **species name**. Furthermore **one** of these combinations of arguments is mandatory:

- `-homologous_species (-hs)` and optional `-non_homologous_species (-nhs)`
- or `-reference_species (-rs)` and `-non_homologous_species_by_reference (-nhsbr)`
- or `-non_homologous_species_by_symbol (-nhsbs)`

which all expect comma separated lists of **file paths** to CDSs containing files (one separate file for each species). The chosen combination of arguments determines how ortholog groups are built from the CDSs (see Input files).

Anchor species

The anchor species is determined by the argument `-anchor_species (-as)`. It expects a **single species name** referencing to a species that you added to the analysis by a CDS file. **The meaning of the anchor species is that the program will try to find for every isoform of each gene of the anchor species the best matching isoform of each other species.** From that way found combinations of isoforms the alignments are built that are the basis of the later positive selection analysis. So genes and isoforms that are lacked by the anchor species will not be a part of the later analysis. Additionally the domains and functional sites that are listed anchor species CDS file will be transferred and shown in the alignment visualizations. So, **we recommend to use that species as anchor that has the according to your estimation the best annotated set of CDSs in terms of completeness and quality.** The anchor species can be freely chosen among all included species. It does not have to be one of the target species.

Target species

The target species are determined by the argument `-target_species (-ts)`. One or multiple **species names** are expected. Multiple target species are separated by comma or by multiple use of the `-target_species` argument. **The argument determines which evolutionary branch will be analyzed for positive selection.** During a PosiGene run a phylogenetic tree is calculated. **The LCA of all target species will be tested for positive selection.** See Examples.

Input files

The first thing PosiGene does with the CDSs of your chosen species set is to build ortholog groups from them. The arguments you use to pass the CDS files will determine the strategy that is used for this task. PosiGene offers three different ways: HomoloGene based, reference based and symbol based ortholog assignment. While it is theoretically possible to mix those strategies we would advise against it. If applicable we recommend using HomoloGene based ortholog assignment.

HomoloGene based ortholog assignment

HomoloGene is a database from NCBI that contains ortholog relationships of genes from 21 species (February 2016) covering a wide evolutionary range of eukaryotes, see Appendix III: HomoloGene species. The PosiGene package includes a copy of this database and can build ortholog groups for the genes of the 21 HomoloGene species based on this information. The argument `-homologene_species (-hs)` expects at least one **file path** to a CDS containing file in genbank (.gbk) format. It is also possible to pass multiple CDS to the program files by giving a comma separated list of file paths to the `-homologene_species (-hs)` argument or by using the argument multiple times. **The package includes already the CDS files for the 21 HomoloGene species in the data/ subdirectory.** You only have to give the file path to the files that represent your chosen species to the `-homologene_species (-hs)` argument. However, you using the CDS files in the /data subdirectory is not mandatory. You can download, e.g., sequences from newer genome releases from NCBI, if such are available, and use them instead as well.

You can furthermore add as many non-HomoloGene species as you want by using – non_homologene_species (-nhs) argument. Here **file paths to GenBank (.gb,.gbk) or fasta (.fasta,.fa) files** are allowed. They will be added to your analysis by a best bidirectional blast (BBB) system against your chosen set of HomoloGene species. **The HomoloGene based ortholog assignment requires you to add at least one HomoloGene species with -homologene_species (-hs) argument.**

We recommend to use the HomoloGene based ortholog assignment if applicable and to add all HomoloGene species to an analysis that make sense in it.

Reference based ortholog assignment

This strategy is for situations in which the HomoloGene based approach is not applicable, because all HomoloGene species are evolutionary too distantly related from the branch that shall scanned for PSGs. **Instead here gene symbols are assigned by BBB against a user-chosen reference species. The -reference_species (-rs) argument expects exactly one file path** to a CDS containing file in GenBank (.gb, .gbk) or fasta (.fasta,.fa) format. The gene symbol of the best matching gene of the reference species will be assigned to a respective gene from a species you want to add to the run. **The CDS files of those species you want to add reference based are passed with the -non_homologene_species_by_reference (-nhsbr) argument.**

Attention: The reference species will itself not automatically be added to the PosiGene run. Enlist it with the `-non_homologene_species_by_reference (-nhsbr)` argument if you want it to be part of further analysis (it then will be technically added in the symbol based manner).

Symbol based ortholog assignment

This strategy is for situations where you want PosiGene to use an already existing ortholog assignment. **Here, simply the gene symbols that are listed in your chosen CDS files will be used.**

Sequences that are listed with the same gene symbol across the different species and their respective CDS files will be in one ortholog group. **The CDS files of those species you want to add symbol based are passed with the -non_homologene_species_by_symbol (-nhsbs) argument.** Files in GenBank (.gb, .gbk) or fasta (.fasta, .fa) format are allowed.

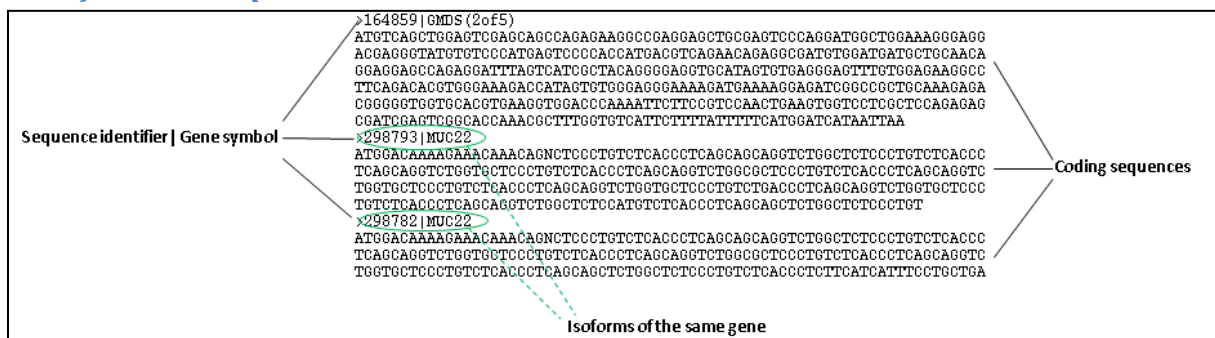
File formats

- For the arguments -non_homologene_species (-nhs), -reference_species (-rs), -non_homologene_species_by_reference (-nhsbr) and -non_homologene_species_by_symbol (-nhsbs) files in GenBank (.gb, .gbk) or fasta (.fasta, .fa) format are allowed
- For the argument -homologene_species (nhs) only files in GenBank format (.gb, .gbk) are allowed

The program assumes as default passed CDS files to be in GenBank format. Files in fasta format are indicated this way: **Species_name:path_to_fasta_file**. The occurrence of a double point character (":") in an element of the comma separated file list, indicates a fasta file. However, the part after the double point has to be the actual path to the fasta file. For the part before the double point you have to enter the species name.

PosiGene expects for both formats, GenBank and fasta, unique sequence identifiers. **If you have multiple isoforms of your genes, we recommend giving them all to PosiGene** (instead of using only the longest one or something like that). The program will assign the best matching isoform of species to each anchor species isoform. However, you have to ensure that PosiGene recognizes different sequences of a species to be isoforms of the same gene by listing the same gene symbol for those sequences in the respective CDS file. In regard to the fasta format PosiGene divides everything after the greater than character (">") in fields separated by the pipe character ("|"). The first field serves as sequence identifier, the second field as gene symbol. If only one field is present it serves as both.

Fasta format example



GenBank format example

Sequence identifier — LOCUS NM_001293312 3113 bp mRNA linear PRI 10-JUN-2014

DEFINITION Homo sapiens aspartyl-tRNA synthetase (DARS), transcript variant 2, mRNA.

ACCESSION NM_001293312

VERSION NM_001293312.1 GI:648216371

KEYWORDS RefSeq.

Species name — SOURCE Homo sapiens (human)

ORGANISM Homo sapiens

Eukaryota; Metazoa; Chordata; Craniata; Vertebrata; Euteleostomi; Mammalia; Eutheria; Euarchontoglires; Primates; Haplorrhini; Catarrhini; Hominidae; Homo.

gene 1..3113 — Gene symbol

/gene="DARS"

/gene_synonym="aspRS; HBSL"

/note="aspartyl-tRNA synthetase"

/db_xref="GeneID:1615"

/db_xref="HGNC:2678"

/db_xref="MIM:603084"

CDS 459..1664 — CDS coordinates within gene or transcript

/gene="DARS"

ORIGIN

1 gagaccctgc gcgcggccgc cagcgctccg ggatctcgag atagccgcag ctctcgcgat

61 ctttctggag ccgcacctcc acgcggagtc cgagcgcgtg tgcctgagac ccagggtcgg

121 gaagcgccgag actgggaggg agggagaagc ccctttggcc tgcctgagac ccagctcgca

181 gggagggttg tgctccactgc ccagttccgt gtcccgatgc ccagcgccag ccgaccgccg

241 aagagctcag agaagccgcg ggagatcatg gacgcggcgg aaatcgagtt ttggttcggg

301 tttagagact gacaatacaa aaagctgatg aagttgtttg ggtacgtgca agagttcata

361 caagcagagc taaagggaac cagtgctctt tagtctctac tcagcagcag tttaatgtcc

421 aggcctcttg ggcggtggga gaccatgcaa gcaagcagat ggtaaatttt gctgccaaca

481 tcaacaagaa gagcattgtg gatgtagaag gtgtgttgag aaaagtgaat cagaaaaattg

541 gaagctgtac acagcaagac gttgagttac atgttcagaa gatttatgtg atcagtttgg

2821 tgtttgaag ttaaaagggg tgtttaccat ctatattgta cagctcttga aagtttagat

2861 tcagttctta tttttttctt tcatatctca ccagaatacc ttaaatatag aagcaccacc

2941 aaaagtattt cataatttaa tccacagcaa gaataaaata tccattctct atagttgtga

3001 attgctagac cctaaaatag ttggccctag agctgttctt aatttcattg

3061 aattatagct tgtttcaaaa taaagtgtgt aaagctctaa aaaaaaaaaa aaa

Start of next sequence — //

LOCUS NM_001349 3171 bp mRNA linear PRI 10-JUN-2014

Data sources

There are a lot of databases which allow downloading CDSs files for selected species in the required format. You can find a list containing such databases [here](#). However, the databases [NCBI](#) and [Ensembl](#) already provide access to CDSs of most eukaryotic species that were sequenced so far. It follows a short explanation how one can get species specific CDS files from those two databases at the example of *Sus scrofa*, the wild boar.

NCBI

Go to the nucleotide section of the NCBI website: <http://www.ncbi.nlm.nih.gov/nucleotide>

The screenshot shows the NCBI Nucleotide search results page for 'Sus scrofa'. The search criteria are: Species: Sus scrofa[organism], Molecule types: mRNA, Source databases: RefSeq. The results show 47,445 items. A 'Send to' dialog box is open, showing the 'Choose Destination' options: File, Clipboard, and Collections. The 'Format' dropdown is set to 'GenBank (full)', and the 'Create File' button is highlighted.

1. Type in the search field the name of your chosen species. You can use the scientific latin name as well as the common English one. In this example: “Sus scrofa” or “wild boar”.
2. Click on “Search”.
3. Check “mRNA” to restrict your search to protein-coding RNA.
4. Check also “RefSeq”, if the number of sequences is not decreased to a too low level by that, to ensure a high quality standard. If more then 10.000 mRNAs for a species are available after restricting to RefSeq, you normally should take the restricted set.
5. Click on “Send to”.
6. Choose output file format to “GenBank(full)”.
7. Click on “Create File” and download it to your computer.

Alternatively you can download GenBank files from NCBI via ftp:

[ftp://ftp.ncbi.nlm.nih.gov/genomes/\[species_name\]/RNA/rna.gbk.gz](ftp://ftp.ncbi.nlm.nih.gov/genomes/[species_name]/RNA/rna.gbk.gz)

E.g.:

ftp://ftp.ncbi.nlm.nih.gov/genomes/Sus_scrofa/RNA/rna.gbk.gz

Ensembl Biomart

Go to the Ensembl Biomart website: <http://www.ensembl.org/biomart/martview/>

e!Ensembl BLAST/BLAT | BioMart | Tools | Downloads | Help & Documentation | Blog | Mirrors Login/Register

Search all species...

New **Count** **Results** **URL** **XML** **Perl** **Help**

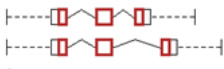
Dataset Sus scrofa genes (Sscrofa10.2)	<div style="border: 1px solid #ccc; padding: 2px; margin-bottom: 5px;"> Ensembl Genes 83 1. </div> <div style="border: 1px solid #ccc; padding: 2px; margin-bottom: 5px;"> Sus scrofa genes (Sscrofa10.2) 2. </div> <div> Filters [None selected] </div> <div> Attributes 3. Ensembl Gene ID Ensembl Transcript ID </div>
--	---

Datasets -> Filters (filtering and inputs) -> Attributes (desired output) -> Results
[BioMart tutorial](#) | [YouTube](#) | [YouKu](#)

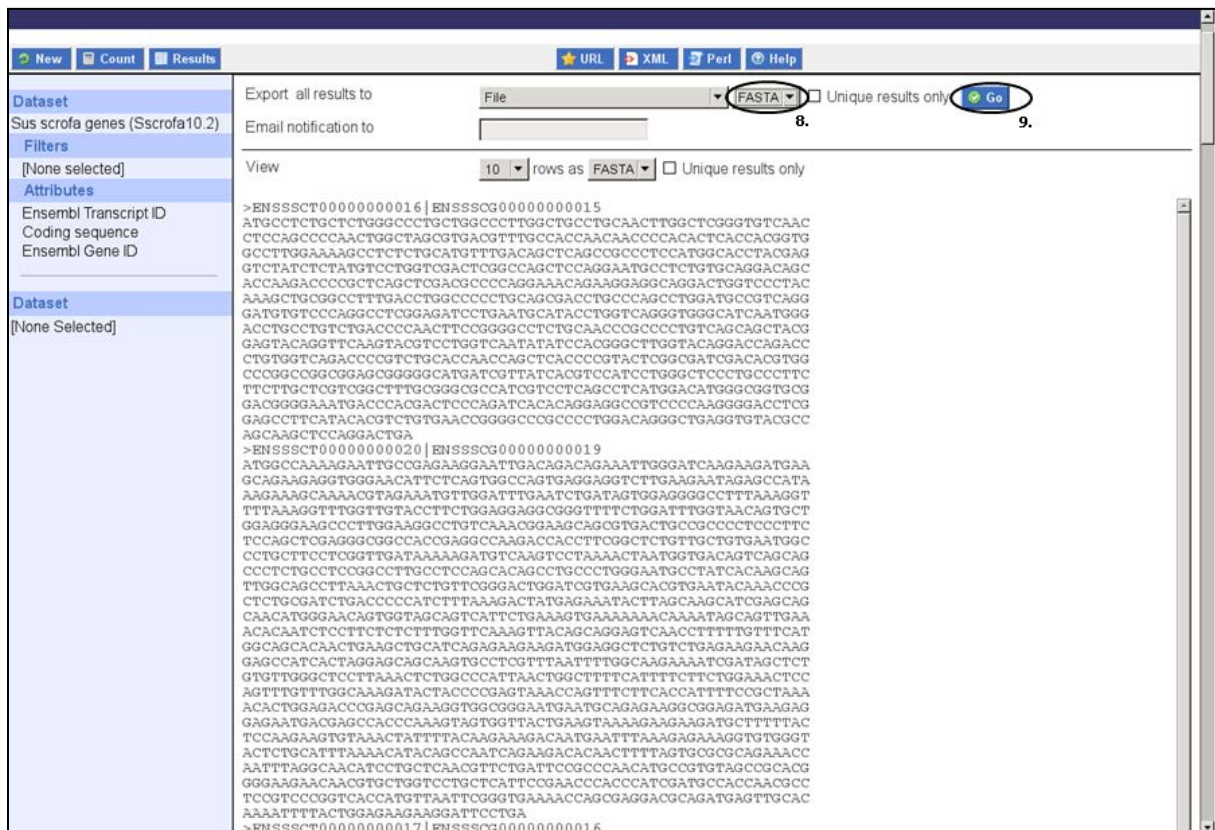
1. Select „Ensembl genes“ from the dropdown menu.
2. Select your species of choice from the dropdown menu. In this example: “Sus scrofa”.
3. Click on “Attributes”.

New **Count** **Results** **URL** **XML** **Perl** **Help**

Please select columns to be included in the output and hit 'Results' when ready

Dataset Sus scrofa genes (Sscrofa10.2)	<div> <input type="radio"/> Features <input type="radio"/> Variant (Germline) <input type="radio"/> Structures <input type="radio"/> Variant (Somatic) <input type="radio"/> Homologs <input checked="" type="radio"/> Sequences 4. </div> <div> SEQUENCES Sequences (max 1)  </div> <div> <input type="radio"/> Unspliced (Transcript) <input type="radio"/> Unspliced (Gene) <input type="radio"/> Flank (Transcript) <input type="radio"/> Flank (Gene) <input type="radio"/> Flank-coding region (Transcript) <input type="radio"/> Flank-coding region (Gene) <div style="float: right;"> <input type="radio"/> 5' UTR <input type="radio"/> 3' UTR <input type="radio"/> Exon sequences <input type="radio"/> cDNA sequences <input checked="" type="radio"/> Coding sequence 5. <input type="radio"/> Protein </div> </div> <div> Upstream flank <input type="checkbox"/> Upstream flank: <input type="text"/> </div> <div> Downstream flank <input type="checkbox"/> Downstream flank: <input type="text"/> </div> <div> Header Information Gene information <input checked="" type="checkbox"/> Ensembl Gene ID 6. <input type="checkbox"/> Description <input type="checkbox"/> Associated Gene Name <input type="checkbox"/> Associated Gene Source <input type="checkbox"/> Chromosome Name <div style="float: right;"> <input type="checkbox"/> Gene Start (bp) <input type="checkbox"/> Gene End (bp) <input type="checkbox"/> Gene type <input type="checkbox"/> Ensembl Protein Family ID(s) </div> </div> <div> Transcript Information <input type="checkbox"/> CDS start (within cDNA) <input type="checkbox"/> CDS end (within cDNA) <input type="checkbox"/> 5' UTR Start <input type="checkbox"/> 5' UTR End <input type="checkbox"/> 3' UTR Start <input type="checkbox"/> 3' UTR End <input checked="" type="checkbox"/> Ensembl Transcript ID <div style="float: right;"> <input type="checkbox"/> Ensembl Protein ID <input type="checkbox"/> Transcript type <input type="checkbox"/> Strand <input type="checkbox"/> Transcript Start (bp) <input type="checkbox"/> Transcript End (bp) <input type="checkbox"/> Transcription Start Site (TSS) <input type="checkbox"/> Transcript length (including UTRs and CDS) </div> </div>
--	--

4. Select “Sequences”.
5. Select “Coding Sequence”.
6. Check off “Ensemble Gene ID” and check it on again. This will cause gene id and transcript id switching places in the fasta sequence header to the order required by PosiGene: transcript_ID|gene_ID.
7. Click on “Results”.



8. Ensure the output file format is fasta.
9. Click on “Go” and download the file to your computer.

Examples

The following examples shall illustrate how PosiGene can be used to identify PSGs given different situations. All examples are “real” and were conducted in a very similar way in our lab.

In the examples some assumptions were made regarding file paths:

Since PosiGene needs file paths to CDSs containing files also the following examples must include them. We assume the Posigene main directory to be the current directory of the command shell. However, this assumption is only made to shorten the example commands a little bit and you do not have to be in that directory to run PosiGene in general. Because the output path, controlled by the argument - output (-o), in these examples is a relative path to the current directory, each result would be written to a respective subdirectory of the PosiGene main directory. Furthermore, paths to CDSs files of species that are not part of the PosiGene package are used sometimes. We assume them in these examples to lie in a directory /usr/local/my_seq_data/. How one can acquire publicly available CDSs for a species of his choice is described in Data sources.

Example I: Only HomoloGene species

Assume we want to search for PSGs in the human lineage using the smallest possible number of species. We would need the human sequences as well as those from at least three other species. In this example sequences from its closest relative the chimp (*Pan troglodytes*) and from another monkey, the macaque, as well mouse (*Mus musculus*) as outgroup are used:

```
perl PosiGene.pl -hs=data/Homo_sapiens.gbk,data/Pan_troglodytes.gbk,data/Macaca_mulatta.gbk,data/Mus_musculus.gbk
-as=Homo_sapiens -ts=Homo_sapiens -o=human_four_species/ -tn=48
```

- Only HomoloGene species are used. So `-homologene_species` (`-hs`) is the only argument that gets input files here and all used files are available already within the PosiGene package without need of further downloads.
- Human will be used as anchor species (`-as`) meaning the program will try to test every human isoform against the most similar isoforms of the other species. Also for the visualization the available domain information in `data/Homo_sapiens.gbk` will be applied to the alignment.
- Among other things the following phylogenetic tree will be reconstructed during the run and one of the terminal branches, that one labeled with “Homo_sapiens”, be tested for positive selection due to the argument `-target_species` (`-ts`):



- The results would be written to `human_four_species/`.
- Due to the `-thread_number` (`-tn`) argument the number of processors that shall be used is set to 48. This, of course, only makes only sense if one has at least that many cores.

Example II: HomoloGene and non-HomoloGene species

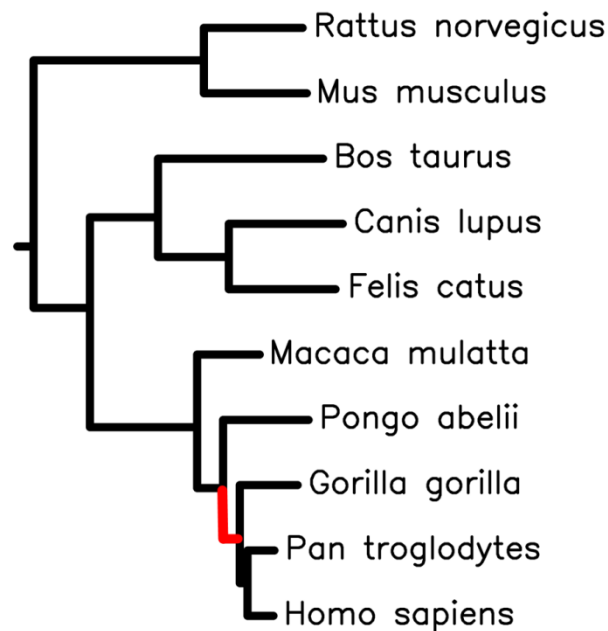
Assume we want to identify the PSGs in the last common ancestor of the hominines. The CDSs of all three hominines - human, chimp and gorilla – and of their closest living relative – the orangutan (*Pongo abelii*) - are within the PosiGene package or publicly available. In this example we would use a larger outgroup set of additional six species:

```
perl ~/workspace2/Test/positive_selection_pipeline.pl -o=hominines -as=Mus_musculus
-ts=Homo_sapiens,Gorilla_gorilla -tn=64
-hs=data/Homo_sapiens.gbk,data/Mus_musculus.gbk,data/Rattus_norvegicus.gbk,data/Macaca_mulatta.gbk,data/Pan_troglodytes.gbk,data/Bos_taurus.gbk,data/Canis_lupus.gbk
-nhs=Gorilla_gorilla:/usr/local/my_seq_data/Gorilla_gorilla.fa,Pongo_abelii:/usr/local/my_seq_data/Pongo_abelii.fasta,/usr/local/my_seq_data/Felis_catus.gbk
```

- In total we have here seven HomoloGene (`-hs`) and three non-HomoloGene (`-nhs`) species
- CDS files in fasta as well as GenBank format can be passed with `-nhs` argument, while the `-hs` argument expects only GenBank format
- The anchor species, here mouse (*Mus musculus*), neither have to be a HomoloGene or one of the target species. However, it should be well annotated and not too distantly related to the

targets. So, especially in the light of the second condition we may should change the argument `-as=Mus_musculus` to `-as=Homo_sapiens` again.

- During the PosiGene run the following phylogenetic tree will be reconstructed and due to `-ts=Homo_sapiens, Gorilla_gorilla` the red marked branch tested. The same branch would have been tested with `-ts=Pan_troglodytes, Gorilla_gorilla` or `-ts=Homo_sapiens, Pan_troglodytes, Gorilla_gorilla` but not with `-ts=Homo_sapiens, Pan_troglodytes`:



- The results would be written to hominines/ by `-o=hominines`
- 64 processors would be used due to `-tn=64`. It advisable to use as many cores as available for data sets as large as this or larger. In real this data set ran several days on a server of our lab with that number of cores. If you do not know how many cpu cores your system has, the following command will tell you: `cat /proc/cpuinfo | grep ^processor -c`

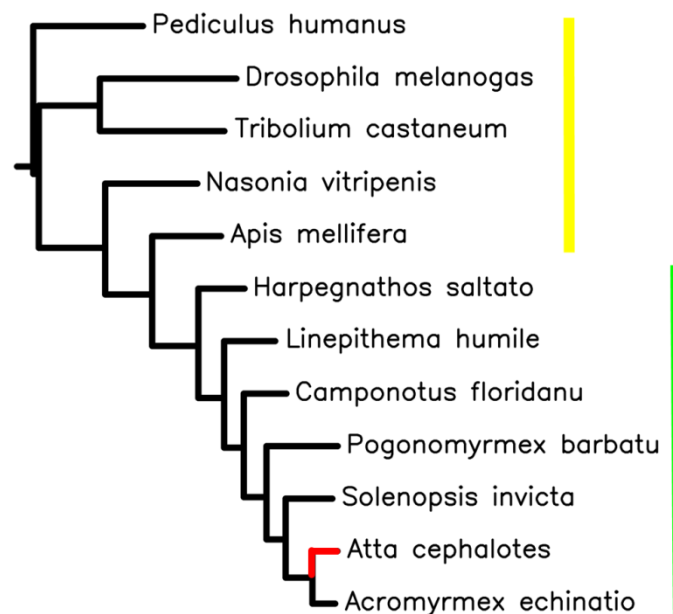
Example III: Reference based ortholog assignment

Assume we want to identify PSGs in a **seven** terminal **ant species** branches. Additionally we want to use **five** other insect **outgroups**. These outgroups include the closest possible HomoloGene relative of the ants *Drosophila melanogaster*, the fruit fly. However, it is still a long evolutionary distance between today living flies and ants. Thus, ortholog assignment between the genes of our seven ant species, we are actually interested in, to that one very distantly related outgroup is probably not the best choice. Instead we use **one of the ant species (*Camponotus floridanus*) itself as reference species** for all other ants and the outgroups.

```
perl PosiGene.pl -o=ants_12_species -as=Acromyrmex_echinator -tn=64 -rs=
Camponotus_floridanus:/usr/local/my_seq_data/Cflo.fa -ts=Atta_cephalotes
-nhsbr=Acromyrmex_echinator:/usr/local/my_seq_data/Aech.fa,Atta_cephalotes:/usr/lo
cal/my_seq_data/Acep.fa,Camponotus_floridanus:/usr/local/my_seq_data/Cflo.fa,Harpeg
nathos_salinator:/usr/local/my_seq_data/Hsal.fa,Linapi thema_humile:/usr/local/my_seq
_data/Lhum.fa,Pbar:/usr/local/my_seq_data/Pogonomymex_barbatus.fa,Solenopsis_invic
ta:/usr/local/my_seq_data/Sinv.fa,data/Drosophila_melanogaster.gbk,Apis_mellifera:
/usr/local/my_seq_data/Amel.fa,Nasonia_vitripennis:/usr/local/my_seq_data/Nvit.fa,
```

Tribolium castaneum: /usr/local/my_seq_data/Tcas.fa: *Pediculus humanus*: Phum.fa
/usr/local/my_seq_data/

- Since the `-reference_species (-rs)` argument is used, all input files have to be passed with the `-non_homologous_species_by_reference (-nhsbr)` argument.
- The reference species *Camponotus floridanus* is here also used as anchor species (`-as`). Each other species could have been chosen as well to serve as anchor. However, because of the long evolutionary distances and our interest in the ants in this scenario it makes sense to also choose one of the ants as anchor.
- The reference species *Camponotus floridanus* is not mentioned only in the `-reference_species (-rs)` argument but also listed in the `(-nhsbr)` argument. Otherwise it would only be used as reference for ortholog assignment and not be itself part of the positive selection analysis.
- This PosiGene run would search for PSGs in the *Atta cephalotes* branch due to the `-target_species (-ts)` argument. The following tree would be created during the run:



- Other terminal and ancestral ant could be scanned for PSGs with the same command as above. Only the `-target_species (-ts)` argument changed accordingly. However, this would unnecessarily repeat again the same ortholog assignment, alignment calculations and phylogenetic tree reconstruction. How this can be avoided and by that time saved is explained in the Modularization section.
- The results would be written to `ants_12_species/`
- The number of threads is set to 64 by `-tn=64`

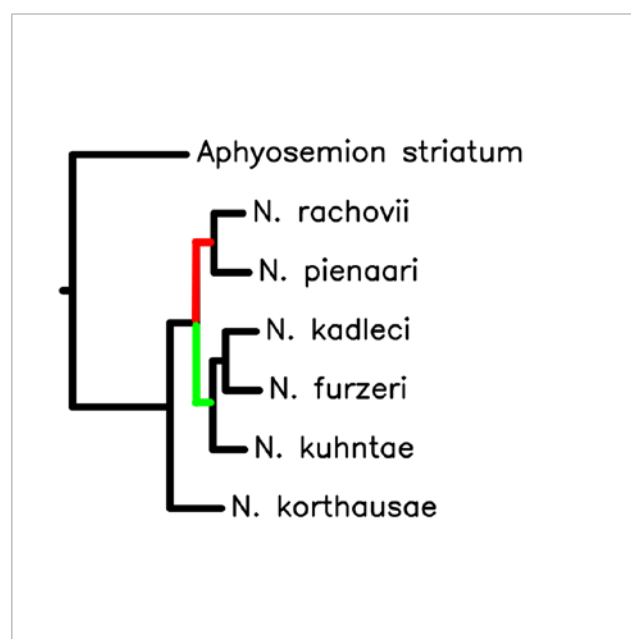
Example IV: Symbol based ortholog assignment

Assume your lab sequenced and annotated the genome or transcriptome of several killifishes from the *Nothobranchius* clade plus a closely related outgroup. The group leader wants us to identify PSGs in two ancestral branches. There is a quite long evolutionary distance between those killifishes and

their closest HomoloGene relative *Danio rerio*, the zebrafish. However, your colleagues already assigned gene symbols to the *Nothobranchius* CDSs by best bidirectional blast against the zebrafish and potentially some other fishes. There would be no advantage in letting PosiGene apply basically the same best bidirectional blast strategy against the zebrafish again. So we decide to save time and to simply use the gene symbols assigned by our colleagues to build ortholog groups from them. See Fasta format example to see how the input files should be formatted.

```
perl PosiGene.pl -o=Nothobranchius -as=Nothobranchius_furzeri
-ts=Nothobranchius_rachovii, Nothobranchius_pienaari -tn=64
-nhsbs=Nothobranchius_furzeri:
/usr/local/my_seq_data/NFU.fa, Nothobranchius_kadleci:
/usr/local/my_seq_data/NKA.fa, Nothobranchius_korthausae:
/usr/local/my_seq_data/NKO.fa, Nothobranchius_kuhntae:
/usr/local/my_seq_data/NKU.fa, Nothobranchius_pienaarii:
/usr/local/my_seq_data/NPI.fa, Nothobranchius_rachovii,
/usr/local/my_seq_data/Aphyosemion_striatum: APS.fa
```

- Since symbol based ortholog assignment is used all input CDS files are passed with the `-non_homologous_species_by_symbol (-nhsbs)` argument
- The *Nothobranchius furzeri* is picked as anchor species due to `-as=Nothobranchius_furzeri`
- During the PosiGene run the following phylogenetic tree would be created and the red marked branch tested for positive selection due to the `-ts=Nothobranchius_rachovii, Nothobranchius_pienaari` argument



- The green marked branch could be tested with the same command changing only the `-target_species (-ts)` argument to `-ts=Nothobranchius_kadleci, Nothobranchius_furzeri, Nothobranchius_kuhntae` or `-ts=Nothobranchius_kadleci, Nothobranchius_kuhntae` or `-ts=Nothobranchius_furzeri, Nothobranchius_kuhntae`
- The results would be written to Nothobranchius/ due to `-o=Nothobranchius`
- The number of threads is set to 64 by `-tn=64`

Output

The directory, to which all file output of a run will be written to, is passed with the argument `-output (-o)`. On default this is the current directory of the command shell. After a full run the pipeline will have created five subdirectories: `individual_results/`, `logs/`, `ortholog_assignment/`, `result_tables/` and `trees/`.

Result tables

Four result tables are written to the `result_tables/` subdirectory at the end of a full PosiGene run. They are all named as the used anchor species, followed by the value of the `-branch_name (-bn)` argument (if it was used) and, in three of four cases, a further description of the table. E.g. in our [Example I](#) – Only *Ho molo* Gene species we would find the following content of the `result_tables/` subdirectory:

- `Homo_sapiens_results.tsv`
- `Homo_sapiens_results_best_iso_per_gene.tsv`
- `Homo_sapiens_results_worst_iso_per_gene.tsv`
- **`Homo_sapiens_results_short.tsv`**

During the PosiGene run not more than one isoform of each other species – the most similar one – was aligned to those of the anchor species and the test for positive selection conducted based on these alignments. So each line of all result tables basically represents one **isoform** of the anchor species.

The first table contains the following information per line:

- The name of the respective anchor species isoform
- p-values: “raw” p-values, that are calculated from the CodeML output, adjusted p-Values by the PosiGene pipeline, multiple test adjusted ones
- Path to the folder within `individual_results/` that contains all file output regarding this isoform that was produced during the PosiGene run. The path also includes the gene name.
- Number and names of species represented in the respective alignment
- Paths to alignment visualizations (stored within `individual_results/`).
- Omega (d_N/d_S ratio) and kappa (transition/transversion ratio) values for alternative- and null-model
- Positively selected sites: Positions in sequences and alignments, respective codons and amino acids

However, normally not all isoforms of the anchor species are represented in the result tables due to the various filter mechanisms of the pipeline. E.g. if in an alignment too few sequences are left that meet certain percentage identity requirements in respect to the anchor sequence and among each other, the alignment of that anchor species isoform will not be processed further in the run. If all isoforms of an anchor species gene (or the respective alignments) do not pass all filters, also such a gene will not appear in the result lists.

The last three tables are in principle different excerpts from the first one. While the first table can include multiple lines per anchor species gene (due to multiple isoforms of a gene), the tables with the descriptions “`best_iso_per_gene`” and “`worst_iso_per_gene`” will include at most one line per anchor species gene: that line with the lowest and highest p-value of the gene for

“best_iso_per_gene” respectively “worst_iso_per_gene” . Apart from that only the multiple test corrected p-values are different to those from the first table due to different list length.

The last table, “short”, contains the same number of lines (isoforms) as the first table but fewer columns. It is handier than the other tables.

Visualizations

The visualizations show the positively selected in the context of a multiple sequence alignment that was used to predict them. If such information is available also domains and functional sites are marked.

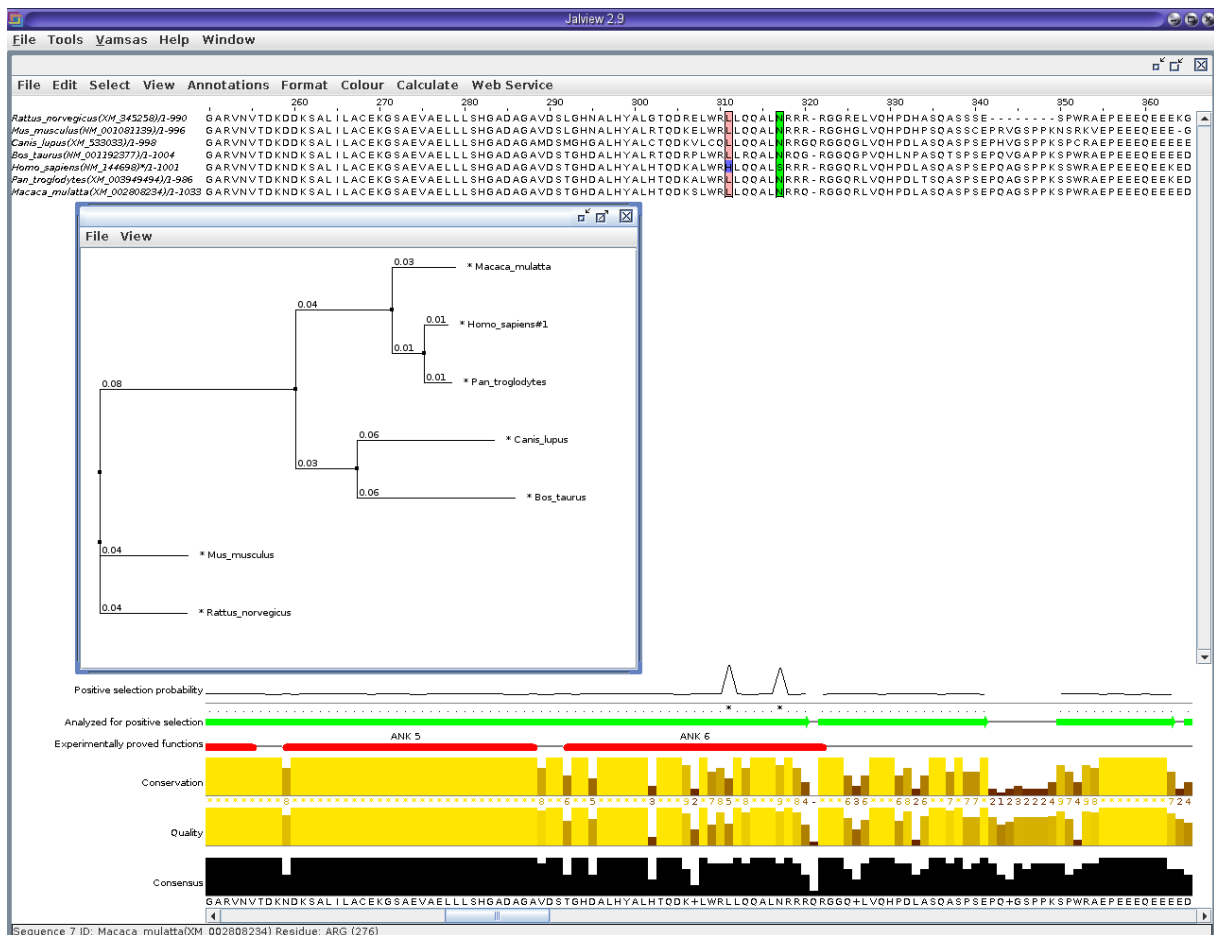
Technically PosiGene produces two kinds of alignment visualizations: png-images and Jalview alignments. The first kind can and will only be produced if the GD-library is installed on the system. The files of the latter kind will be produced anyway but need Java to be installed on the system to show them.

Jalview visualizations

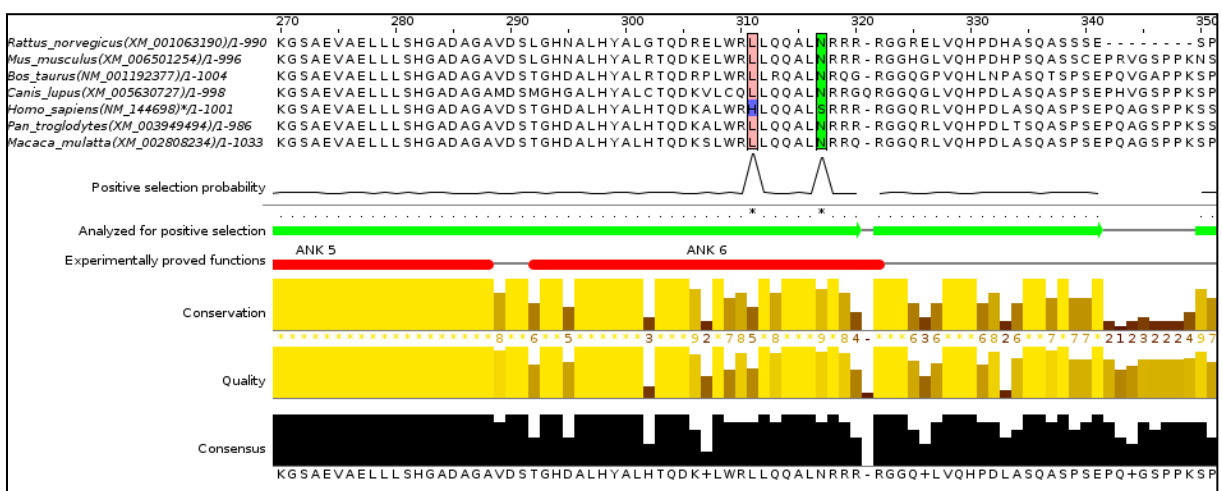
Jalview is a java-based alignment-editor that is integrated in the PosiGene package. Visualization can be started with a command line call of PosiGene and use of the argument `-view` that gets the path to a respective *.view file. The *.view files are referenced in the result tables. In the complete result table you will find two *.view files per result line – one for the protein-level and one for the nucleotide level (the “short” table only references to protein-level visualization). E.g.: Imagine a PosiGene run on the human lineage that were written with the `-o` argument to destination “some_human_PosiGene_run/”. The result tables will contain a column “Protein alignment (Jalview)”. For the isoform NM_144698 of *ANKRD35* gene the entry of this column would be a file path to a *.view file. If this file path is given with the `-view` argument to PosiGene...

```
perl PosiGene.pl
-view=some_human_PosiGene_run/individual_results/ANKRD35/Homo_sapiens/NM_144698_codeml/prank/p
rank.best.fas.translation.view
```

...Jalview will be started and two subwindows will pop up within the Jalview window – one with the used tree and one with the alignment of *ANKRD35*. In the alignment columns of positively selected sites are highlighted according to the chemical properties of the amino acids (in the case of the protein-level visualization):



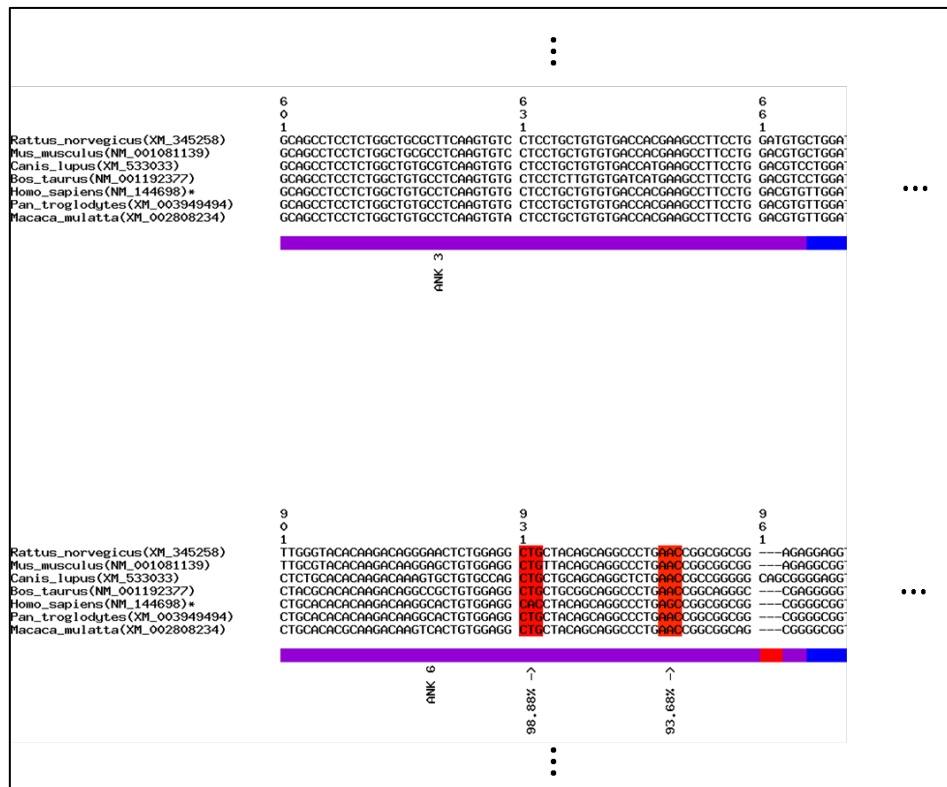
The exact probability of highlighted sites to be under positive selection is shown with tooltips that occur if the cursor remains a short time on them. You can use Jalview to modify the alignment and the annotations by hand if you want to. Via “File” (alignment window) → “Export images” you can put all this into nice graphics. Example:



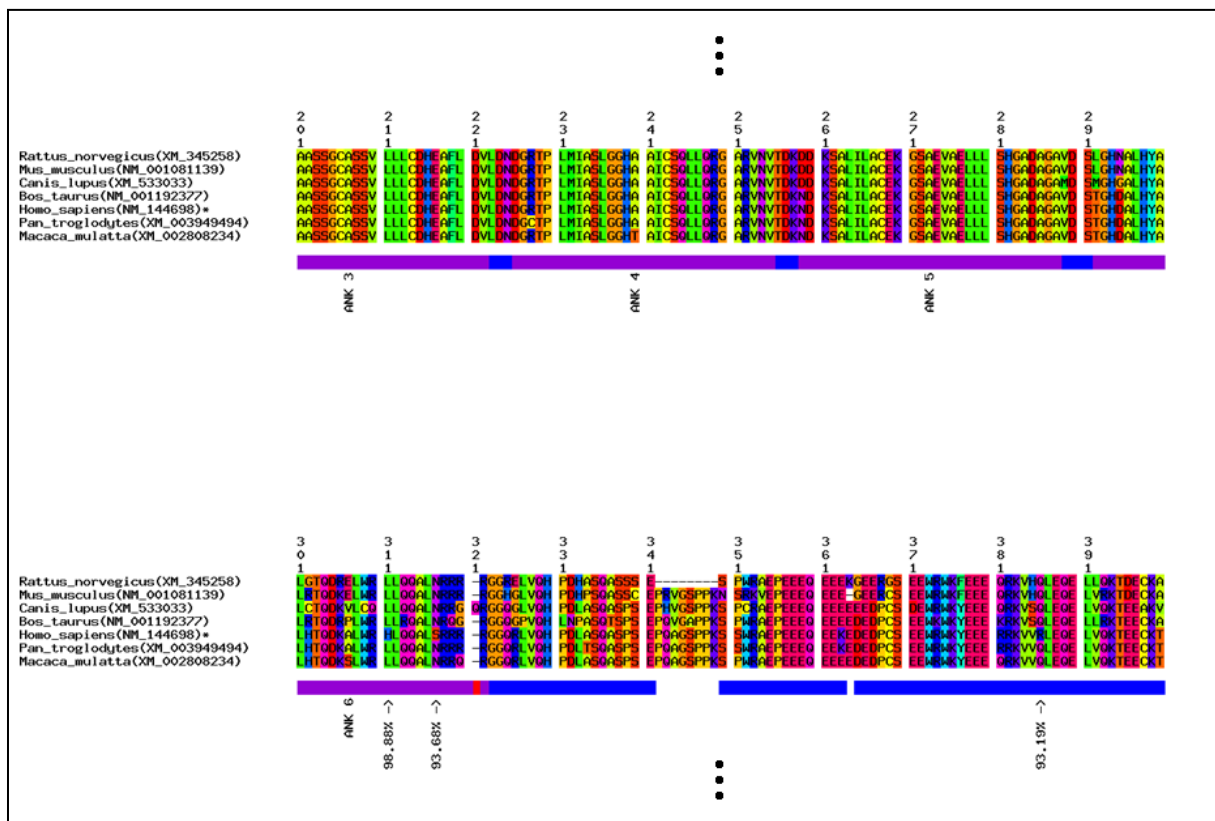
PNG images

For each alignment and any of these combinations an image is produced: protein- or nucleotide-level, annotated (with functional information) or not, interleaved presentation or as one stretch. This results in a set of eight PNG-images per alignment that are referenced in the result table. Two of

Nucleotide-level, interleaved, annotated (cutting):



Protein-level, interleaved, annotated (cutting):

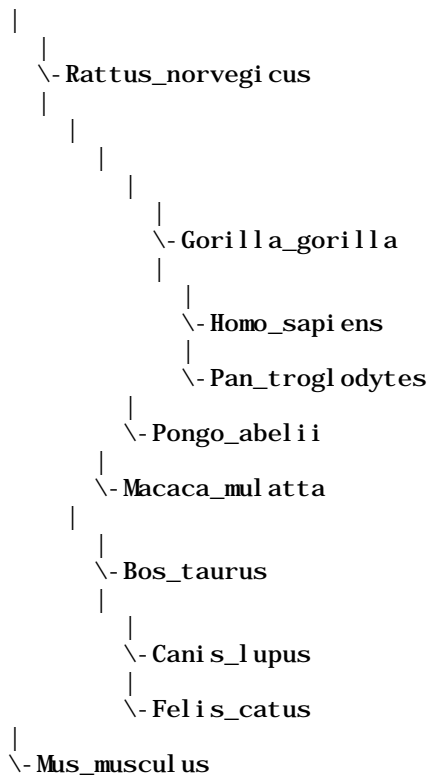


Trees

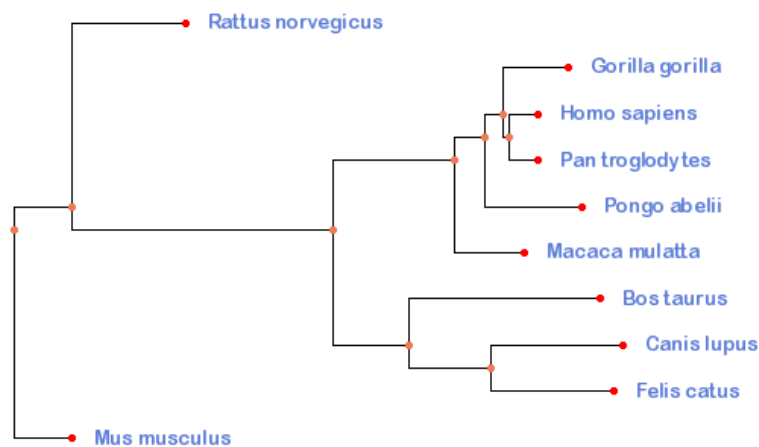
After a PosiGene run the subdirectory “trees/” (within the output directory) contains all trees that represent the whole data set of the run as well as files that were used to create them. The trees reconstructed by the pipeline are always in newick format. Among others the subdirectory trees/ will contain a tree file named by the used anchor species, e.g., tree_anchor_species_Homo_sapiens.newick. This reconstruction of the tree is done with the program “dnaml” of the phylip package and based on a concatenated alignment over the whole gene set that can also be found within trees/.

There are multiple ways for visualizing such newick tree files:

- A simple tab-tree output on the command line will be produced by the use of the -show_tree argument, e.g: `perl PosiGene.pl -show_tree=hominines/trees/tree_anchor_species_Homo_sapiens.newick:`

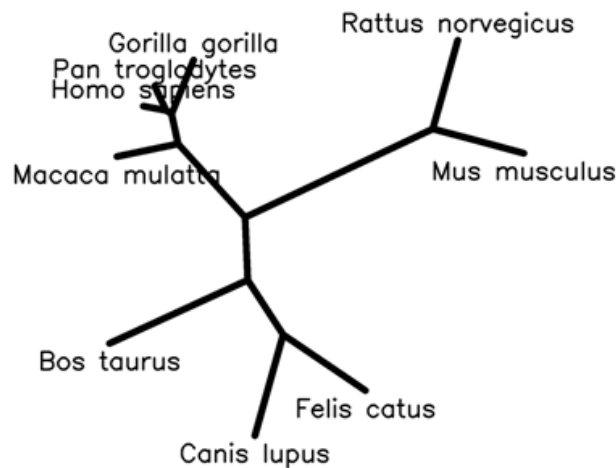


- External programs or webservices for newick interpretation, e.g. Phyfi: <http://cgi-www.cs.au.dk/cgi-chili/phyfi/go>. If one loads up the tree_anchor_species_Homo_sapiens.newick of our it looks like this:



- The `-view` argument shows among others a tree for a specific result line with Jalview visualizations. Phylogenetic trees shown this way are the same as the general ones stored in trees/. However, they are reduced to the species set that is actually present in the respective alignment.

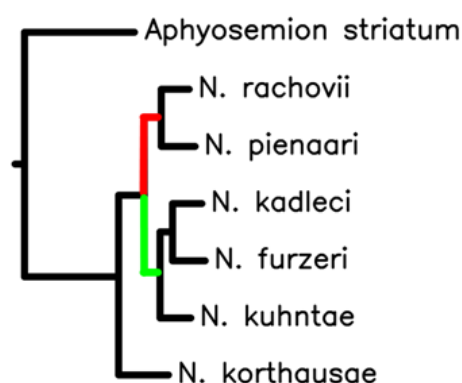
Please note that the trees reconstructed by the PosiGene pipeline are always unrooted – in contrast to the simplified trees that we have show in the Examples section. That is the reason for three child nodes of the pseudoroot left in the Phyfi-tree. Less misleading visualizations, that do not imply a root, can be done, e.g., with the program “drawtree” of the phylip package (not part of PosiGene):



If you want PosiGene to use a tree of your own instead of reconstructing it, pass a file path to a *.newick file with the argument `-tree (-t)`. It will be used for alignment calculation (Prank) and testing for positive selection. Please, ensure that the species names in the tree file are the same as used with the sequence input files. The tree can be rooted (PosiGene will internally make it unrooted).

Context species

PosiGene will set the p-value of any alignment to one, regardless of what CodeML calculated, if no species of the sister taxon of the tested branch is represented in that alignment. Example: If we would test the red marked branch of the following tree, then the alignment must include at least one (logical “or”) of the species under the green branch (*N. kadleci*, *N. furzeri*, *N. kuhntae*). Otherwise the p-value is set to one (the p-value calculated by CodeML occurs still in the result table as “raw p-value”). This is done to ensure that predicted positive selection really occurred on the tested branch and somewhere before that.



You can disable this by setting the argument `-context_species (-cs)` to any of the target species (here, e.g., `-cs=Notobranchius_rachovii`). Instead you can also give an own comma separated list of species that must be included in an alignment. However, then **all** your listed species have to be in the alignment (logical “and”), otherwise the p-value is set to one.

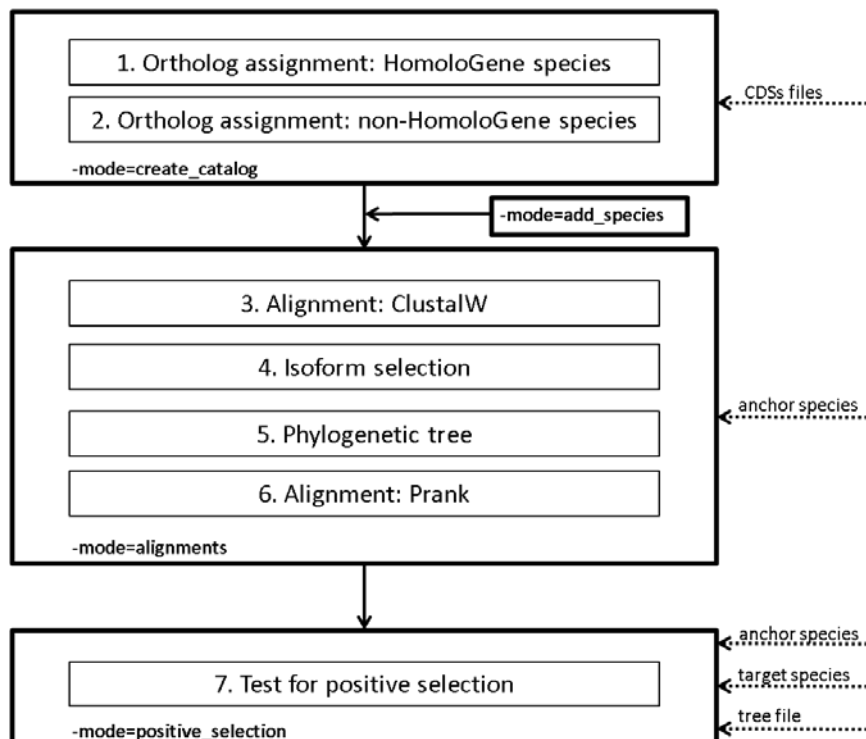
Modularization – How to accelerate (re-)analysis

The modularization allows reusing already calculated results in order to reduce calculation effort and run time for analyses on the same data set. E.g.: If two evolutionary branches on the same species set shall be scanned for PSGs then only the tests for positive selection have to be done twice but not all the other tasks. To explain how this works, we have to take a look on the basic structure of the PosiGene pipeline.

PosiGene is built from seven core scripts that are again summarized to three main modules: **create_catalog**, **alignments** and **positive_selection**. If you call PosiGene without the argument `-mode` as in the Examples the default mode **all** is used and the three main modules are executed in the above mentioned order. However, you also can use the `-mode` argument to call each of the three main modules separately. The order of the main modules must be maintained: “alignments” requires that “create_catalog” ran before on the respective output directory. “positive_selection” needs “create_catalog” and “alignments”. Given this, the main modules have the following mandatory arguments, if called separately:

- `create_catalog`: CDSs files → (`-hs` and `-nhs`) or (`-nhsbr` and `-rs`) or `-nhsbs`
- `alignments`: anchor species → `-as`
- `positive_selection`: (anchor species → `-as`) and (target species → `-ts`) and (tree file → `-t`)

In total these are the same arguments that have to be used for a full PosiGene run, with the exception of the tree file that is needed for “positive_selection”. Such a tree file can be found in the subdirectory “trees/” after the execution of “alignments”.



We see mainly three ways how this can be used to accelerate reanalysis of an existing output directory:

- Use `-mode=positive_selection` and a different specification of the `-target_species` (`-ts`) argument than before to search on another branch of the phylogenetic tree for PSGs.

The old result tables, individual gene results and visualizations will be overwritten unless you use the `-branch_name (-bn)` argument that includes a string of your choice in all output files.

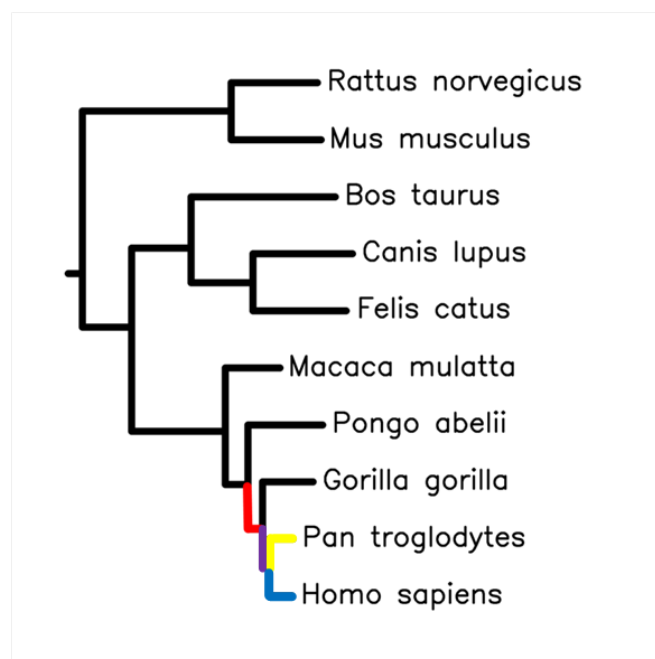
- Use `-mode=alignments` and a different specification of the `-anchor_species (-as)` than before to build alignments based on another anchor species. All results for before analyzed anchor species will remain. However, you have to call (again) “positive_selection” with the new anchor species to get result tables and visualizations for it.
- Use `-mode=add_species` and additional CDSs file(s) of **non-HomoloGene** species to increase the species set of the analysis: `-nhs` or `(-nhsbr and -rs)` or `-nhsbs`. You have to call (again) “alignments” and “positive selection” to get results.

Examples

Here we use the above mentioned points to extend our former Example II – HomoloGene and non-HomoloGene species. We assume that we already have ran the original command.

Different evolutionary branches

The original command scanned the last common ancestor of human, chimp and gorilla for PSGs. Now, we want additionally to analyze the human branch (*Homo sapiens*), the chimp branch (*Pan troglodytes*) and the last common ancestor of human and chimp as shown in the tree below:



The three commands we had to use after PosiGene had already a full run with the original command are:

```
perl PosiGene.pl -o=homini nes -mode=positive_selection -as=Mus_musculus
-ts=Homo_sapiens -bn=human -t=homini nes/trees/
tree_anchor_species_Mus_musculus.newick -tn=64
```

```
perl PosiGene.pl -o=homini nes -mode=positive_selection -as=Mus_musculus
-ts=Pan_troglodytes -bn=chimp -t=homini nes/trees/
tree_anchor_species_Mus_musculus.newick -tn=64
```

```
perl PosiGene.pl -o=homini nes -mode=positive_selection -as=Mus_musculus
-ts=Homo_sapiens, Pan_troglodytes -bn=LCA_human_chimp -t=homini nes/trees/
tree_anchor_species_Mus_musculus.newick -tn=64
```

The use of the `-branch_name (-bn)` argument ensures that our results are not overwritten: E.g., after the original command we would have a result table named according to our anchor species `hominines/result_tables/Mus_musculus.tsv` that stores the information for tests on the LCA of human, chimp and gorilla. After our three commands above we would have in the same directory also `Mus_musculus_human.tsv`, `Mus_musculus_Ptro.tsv` and `Mus_musculus_LCA_chimp_human.tsv`. Of course we could have used the `-branch_name (-bn)` argument in the original command to name the respective results differently, e.g. with `-bn=LCA_human_gorilla_chimp`.

Different anchor species

In the notes of our Example II – HomoloGene and non-HomoloGene species was mentioned that it could be better to use human instead of mouse (*Mus musculus*) as anchor species. However, if we already had a run with mouse as anchor species we do not have to repeat the `create_catalog` step, instead we could do the following:

```
perl PosiGene.pl -o=hominines -mode=alignments -as=Homo_sapiens
```

Then we can do the actual scanning for PSGs like in Different evolutionary branches – but with human as anchor species:

```
perl PosiGene.pl -o=hominines -mode=positive_selection -as=Homo_sapiens
-ts=Homo_sapiens, Pan_troglodytes, Gorilla_gorilla -bn=LCA_human_gorilla_chimp
-t=hominines/trees/tree_anchor_species_Mus_musculus.newick -tn=64
```

```
perl PosiGene.pl -o=hominines -mode=positive_selection -as=Homo_sapiens
-ts=Homo_sapiens -bn=human
-t=hominines/trees/tree_anchor_species_Mus_musculus.newick -tn=64
```

```
perl PosiGene.pl -o=hominines -mode=positive_selection -as=Homo_sapiens
-ts=Pan_troglodytes -bn=chimp
-t=hominines/trees/tree_anchor_species_Mus_musculus.newick -tn=64
```

```
perl PosiGene.pl -o=hominines -mode=positive_selection -as=Homo_sapiens
-ts=Homo_sapiens, Pan_troglodytes -bn=LCA_human_chimp
-t=hominines/trees/tree_anchor_species_Mus_musculus.newick -tn=64
```

Adding non-HomoloGene species

If you want to expand a species set for that you already have constructed an ortholog catalog, you can use the mode “`add_species`”. Here we would add three additional old world monkeys: Gibbon (*Nomascus leucogenys*), Drill (*Mandrillus leucophaeus*) and Baboon (*Papio anubis*). As before we assume that the respective CDSs file lie in `/usr/local/my_seq_data/`:

```
perl PosiGene.pl -o=hominines -mode=add_species -nhs=/usr/local/my_seq_data/Nomascus_
leucogenys.gbk, Mandrillus_leucophaeus:/usr/local/my_seq_data/drill.fasta, /usr/local/
my_seq_data/Papio_anubis.gbk -tn=64
```

After this the modules “`alignments`” and “`positive_selection`” have to be called as shown in Different anchor species and Different evolutionary branches to get results for the augmented species set. May existing results for an anchor species would be deleted if “`alignments`” is called again with the same anchor species. This affects, e.g., all visualizations that were done in the scope of that anchor species. If you want to keep those results use “`add_species`” on a copy of the respective output directory.

PosiGene arguments

Obligatory arguments

-anchor_species|-as → NAME of one species that is part of the species set passed with the input files. To each isoform of this species the best matching isoform of each other species will be aligned. Visualized domain information will also be based on the anchor species. Affected modes: all, alignments and positive_selection. Example: `-as=Mus_musculus`

-target_species|-ts → Comma separated list of species NAMES that are part of the species set passed with the input files. The last common ancestor of these species will be tested for positive selection. If only one species is passed with the argument, this species will be tested.

CDS input files

HomoloGene based ortholog assignment

-homologene_species|-hs → Comma separated list of paths to GenBank files (*.gb, *.gbk) containing sequences of species which are part of the HomoloGene database. Each specified file should contain sequences of exactly one species. You have to provide at least one file/species. Affected modes: all and create_catalog. Example: `-hs=/homes/user_A/bio_data/Homo_sapiens.gb, "/homes/userA/my_seqs/Mus_musculus_12-03-2013.gbk", "/data/bio_data/Rattus_rattus.gb"`

Optional:

-non_homologene_species|-nhs → Comma separated list of paths to GenBank files (*.gb, *.gbk) with sequences of species which are not part of the HomoloGene database. Alternatively fasta files (*.fa, *.fasta) with CDS sequences of non homologene species can be used with `species_name:path_To_fasta_file`. Each specified file should contain sequences of exactly one species. Affected modes: create_catalog, add_species and all. Example: `-nhs=/homes/user_A/bio_data/Cavia_porcellus.gb, /homes/user_A/bio_data/Cricetus_griseus.gb, /homes/user_A/bio_data/, "Ochotona princeps: /homes/user_A/bio_data/O_princeps.fa"`

Reference based ortholog assignment

-reference_species|-rs → Path to a GenBank (*.gb, *.gbk) or fasta (*.fa, *.fasta) file with sequences of a species that shall be used as reference for -nhsbr option. To tell the program that you specify a fasta file write `species_name:path_To_fasta_file` or use file extensions *.fa/*.fas/*.fasta. The reference species will NOT be added to the analysis itself. If you want it being part of the analysis simply add it to the -nhsbr or -nhsbs option. Affected modes: create_catalog, add_species and all.

-non_homologene_species_by_reference|-nhsbr → Same as option nhs. Difference is that ortholog-groups will be built by BLAST against a user-defined reference species. Affected modes: create_catalog, add_species and all.

Symbol based ortholog assignment

-non_homologene_species_by_symbol|-nhsbs → Same as option nhs. Difference is that ortholog-groups will be built straight forward by the gene symbols provided by the user (/gene/gene tag in *.gbk or second entry in *.fasta-header). Affected modes: create_catalog, add_species and all.

Optional arguments

Frequently used arguments

-output_dir|-o=./(default) → Directory where the program writes its results to. Attention: The output folder should either not contain any directories or not yet exist. Affected modes:

create_catalog, add_species, alignments, positive_selection and all. Example:

```
-o=/homes/user_A/my_positive_selection_results/
```

-thread_num|-tn|-cpus=8(default) -> Number of threads that will be used. We recommend to use as many threads as cpu cores are available.

-mode=all(default) → possible values: create_catalog, alignments, positive_selection, all, add_species, info, show_tree, view -> Specifies which steps will be performed by PosiGene. "create_catalog" builds an ortholog catalog based on user-provided sequences. "alignments" calculates alignments of each ortholog group, selects best matching isoform combinations and reconstructs the phylogenetic tree. "positive_selection" calculates the probabilities of each gene to be under positive selection in the branch the user selected. "all" performs the three previous mentioned steps consecutively. "add_species" adds sequences of one additional species belated to the ortholog catalog (to involve this species in positive selection analysis the steps "alignments" and "positive_selection" have to be performed additionally). "info" prints summarized information about what has been done on a chosen output directory so far. "show_tree" enables a simple command line representation of a user-chosen newick tree. "view" opens a Jalview based alignment visualization of a user-chosen .view file.

-branch_name=""(default) → If you want to run "positive_selection" several times on the same output directory to test different branches for positive selection or to use different settings, the program will normally overwrite the result files of previous runs unless you use this parameter which will be used in naming of result files of this run. Affected modes: positive_selection and all.

-view → Sets mode to "view" and visualizes the specified .view file with Jalview. Example:

```
-view=/homes/user_A/my_positive_selection_results/  
/individual_results/TTR/Bos_taurus/NM_173967_codeml/prank/prank.best.fas.translation.view
```

-show_tree → Sets mode to "show_tree" and prints the specified .newick file in tab-tree format to the screen. Example:

```
-view=/homes/user_A/my_positive_selection_results/trees/tree_ancor_species_Bos_taurus.  
newick
```

-info → Prints summarized information about what has been done on the chosen output directory (-o) so far (same as -mode=info).

-continue → If the program was interrupted by some reason (e.g. electricity fail) start it again with same parameters and use additionally this parameter. The program will show you which steps it has performed before it was interrupted and with which step it will proceed if you agree. Affected modes: create_catalog, alignments, positive_selection and all.

-tree_file|-t → Path to a file which contains an evolutionary species tree in newick format. The tree is used for prank alignments and positive selection tests. If you specify a tree for the modes "alignments" or "all" PosiGene will not reconstruct a tree on its own and instead use your tree. If you call the mode "positive_selection" the argument is obligatory. Affected modes: alignments, positive_selection and all. Example:

```
-t=/homes/user_A/my_trees/my_mammal_species_set.newick
```

Fine-tuning arguments

-homologene_file|-hf={PosiGene_directory}/homologene_106.data(default) → Use this parameter to let the program use another version of the Homologene database for the creation of the

orthologue catalog than the by default provided (current version 106). You can download the latest version of Homologene under: <ftp://ftp.ncbi.nih.gov/pub/HomoloGene/current/homologene.data>. Affected modes: create_catalog and all.

-blast_threshold=1E-04(default) → BLAST threshold that is used when adding a non-HomoloGene species to the ortholog catalog. Affected modes: create_catalog, add_species and all.

-use_prank=1(default) → ClustalW alignments are created to decide which isoforms from one gene are selected, so that each species is represented with one sequence in an alignment. By default this selection is later realigned with Prank. This will decrease the number of false positives due to misalignment in the final result but also cost calculation time. Set this argument to 0 to work directly with ClustalW subalignments. Affected modes: alignments and all.

-min_ident=70.00(default) → The most similar non-anchor species isoforms must have additionally at least this pairwise identity percentage against the respective anchor species isoform to be considered for further analysis. Affected modes: alignments and all.

-min_pairs_ident=50.00(default) → → The most similar non-anchor species isoforms (in respect to the respective anchor species isoform) must have additionally at least this pairwise identity percentage against each other sequence in the alignment. Sequences with most violations against this rule are removed from an alignment until all sequences follow the rule. Affected modes: alignments and all.

-context_species|cs=auto(default) → Comma separated list of species. All of these species must be included in the tested alignment. Otherwise the p-value is set to 1. The default “auto” ensures that at least one species from each sister taxon of the tested branch is represented in the alignment. Affected modes: positive_selection and all.

-min_outgroups=0(default) → This number of species that are both non-target and non-context must be represented in an alignment. Otherwise the p-value is set to 2. Affected modes: positive_selection and all.

-max_aln_gaps_soft=40(default) → If the percentage of gaps and filtered columns in an alignment is greater than this number, its p-value in the positive selection analysis will be set to 1. Affected modes: positive_selection and all.

-max_aln_gaps_hard=50(default) → If the percentage of gaps and filtered columns in an alignment is greater than this number, it will be removed from positive selection analysis (that means no result for this alignment). Affected modes: positive_selection and all.

-max_anchor_gaps_soft=33.33(default) → If the percentage of gaps and filtered columns in an anchor sequence is greater than this number, its p-value of the respective alignment will be set to 1 in the positive selection analysis. Affected modes: positive_selection and all.

-max_anchor_gaps_hard=20(default) → If the percentage of gaps and filtered columns in an anchor sequence is greater than this number, the respective alignment will be removed from positive selection analysis (that means no result for this alignment). Affected modes: positive_selection and all.

-min_seq_num_soft=4(default) → If a alignment has less than this number of sequences, its p-value will be set to 1 in the positive selection analysis. Cannot be lower than 3. Affected modes: positive_selection and all.

-min_seq_num_hard=3(default) → If a alignment has less than this number of sequences, it will be removed from positive selection analysis (that means no result for this alignment). Cannot be lower than 3. Affected modes: positive_selection and all.

-min_aln_length=60(default) → If a gap-filtered alignment has less than this number of nucleotides, it will be removed from positive selection analysis (that means no result for this alignment). Affected modes: positive_selection and all.

-min_KaKs|min_omega=0.85(default) → If the KaKs value of the tested branch in the alternative model is less than this argument, the p-value will be set to 1. Affected modes: positive_selection and all.

-max_KaKs|max_omega=150(default) → If the KaKs value of the tested branch in the alternative model is greater than this argument, the p-value will be set to 1. Affected modes: positive_selection and all.

-min_site_num=1(default) → If less than this number of positively selected sites are identified, the p value will be set to 1. Affected modes: positive_selection and all.

-min_site_significance=0.4(default) → The minimal demanded probability of a predicted positively selected site to be counted. Affected modes: positive_selection and all.

-site_excess=0.2(default) → If the fraction of predicted positively selected sites relative to the number of tested sites is equal or greater than this number, a site-excess is detected. Affected modes: positive_selection and all.

-site_if_excess_min=0.9(default) → The minimal demanded probability of a predicted positively selected site if a site-excess was detected. Affected modes: positive_selection and all.

-flank_size=2(default) → Affected modes: positive_selection and all. Region around filtered alignment parts as well as after start and before end where predicted positively selected sites are removed following a subsequent p-value adjustment.

-genetic_code=1(default) → The codon table that will be used. See <http://www.ncbi.nlm.nih.gov/Taxonomy/Utils/wprintgc.cgi>. Affected modes: positive_selection and all.

Appendix

Appendix I: Used Perl modules (part of the package)

- Bio::Align::AlignI
- Bio::Align::Graphics
- Bio::Align::Utilities
- Bio::AlignIO
- Bio::AlignIO::clustalw

- Bio::AlignIO::fasta
- Bio::AlignIO::phylip
- Bio::LocatableSeq
- Bio::Root::IO
- Bio::SearchIO::blasttable
- Bio::Seq
- Bio::SeqFeature::Generic
- Bio::SeqIO
- Bio::SeqIO::fasta
- Bio::SeqIO::GenBank
- Bio::SimpleAlign
- Bio::Tree::Node
- Bio::Tree::Tree
- Bio::TreeIO
- Bio::TreeIO::newick
- Bio::TreeIO::tabtree
- Cwd
- File::Basename
- File::Copy
- File::Path
- Getopt::Long
- POSIX
- Storable
- threads
- threads::shared
- Thread::Queue

Appendix II: Used Software (part of the package)

- NCBI Blast+ (2.2.28)
- ClustalW (2.0.10)
- Phylip (3.696)
- Prank (v. 140110)
- Gblocks (0.91b)
- PAML (4.7)
- Jalview (2.8.1)

Additionally used: HomoloGene database (Release 68)

Appendix III: HomoloGene species

- Homo sapiens
- Pan troglodytes
- Macaca mulatta
- Canis lupus familiaris
- Bos taurus
- Mus musculus
- Rattus norvegicus
- Gallus gallus
- Xenopus tropicalis
- Danio rerio
- Drosophila melanogaster

- *Anopheles gambiae*
- *Caenorhabditis elegans*
- *Saccharomyces cerevisiae*
- *Kluyveromyces lactis*
- *Eremothecium gossypii*
- *Schizosaccharomyces pombe*
- *Magnaporthe oryzae*
- *Neurospora crassa*
- *Arabidopsis thaliana*
- *Oryza sativa*