get_homologues-est manual

Bruno Contreras-Moreira
Laboratory of Computational Biology
Estación Experimental de Aula Dei/CSIC, Fundación ARAID
http://www.eead.csic.es/compbio
Zaragoza, España

Pablo Vinuesa
Center for Genomic Sciences
Universidad Nacional Autónoma de México
http://www.ccg.unam.mx/~vinuesa
Cuernavaca, México

February 1, 2016

Contents

1 Description

This document describes *get_homologues-est*, a fork of *get_homologues* for clustering homologous transcript sequences of strains/populations of the same species. This algorithm has been designed and tested with plant transcripts and CDS sequences, and uses BLASTN to compare DNA sequences. The main tasks for which this was conceived are:

1 Finding and translating coding regions (CDSs) within raw transcripts.

2 Clustering transcripts/CDS nucleotide sequences in homologous (possibly orthologous) groups, on the grounds of DNA sequence similarity.

3 Definition of pan- and core-transcriptomes by calculation of overlapping sets of CDSs.

The core algorithms of *get_homologues-est* have been adapted from *get_homologues*, and are therefore explained in *manual_get_homologues.pdf*. This document focuses mostly on EST-specific options.

2 Requirements and installation

2.1 Perl modules

A few Perl core modules are required by the *get_homologues-est.pl* script, which should be already installed on your system: Cwd, FindBin, File::Basename, File::Spec, Getopt::Std, Benchmark and Storable. In addition, the *Bio::Seq*, *Bio::SeqIO*, *Bio::Graphics* and *Bio::SeqFeature::Generic* modules from the Bioperl collection, and module Parallel::ForkManager are also required, and have been included in the *get_homologues-est* bundle for your convenience.

Should this version of BioPerl fail in your system (as diagnosed by *install.pl*) it might be necessary to install it from scratch. However, before trying to download it, you might want to check whether it is already living on your system, by typing on the terminal:

```
$ perl -MBio::Root::Version -e 'print $Bio::Root::Version::VERSION'
```

If you get a message `Can't locate Bio/Root/Version...` then you need to actually install it, which can sometimes become troublesome due to failed dependencies. For this reason usually the easiest way of installing it, provided that you have root privileges, it is to use the software manager of your Linux distribution (such as *synaptic/apt-get* in Ubuntu, *yum* in Fedora or *YaST* in openSUSE). If you prefer the terminal please use the *cpan* program with administrator privileges (`sudo` in Ubuntu):

```
$ cpan -i C/CJ/CJFIELDS/BioPerl-1.6.1.tar.gz
```

This form should be also valid:

```
$ perl -MCPAN -e 'install C/CJ/CJFIELDS/BioPerl-1.6.1.tar.gz'
```

Please check this tutorial if you need further help.

2.2 Required binaries

In order to properly read (optionally) compressed input files, *get_homologues-est* requires `gunzip` and `bunzip2`, which should be universally installed on most systems.

The Perl script *install.pl*, already mentioned in section 2, checks whether the included precompiled binaries for COGtriangles, hmmer, MCL and BLAST are in place and ready to be used by *get_homologues-est*. However, if any of these binaries fails to work in your system, perhaps due a different architecture or due to missing libraries, it will be necessary to obtain an appropriate version for your system or to compile them with your own compiler.

In order to compile *MCL* the GNU *gcc* compiler is required, although it should most certainly already be installed on your system. If not, you might install it by any of the alternatives listed in section 2.1. For instance, in Ubuntu this works well: `$ sudo apt-get install gcc` . The compilation steps are as follows:

```
$ cd bin/mcl-14-137;
$ ./configure‘;
$ make
```

To compile *COGtriangles* the GNU *g++* compiler is required. You should obtain it by any of the alternatives listed in section 2.1. The compilation would then include several steps:

```
$cd bin/COGsoft;
$cd COGlse; make;
$cd ../COGmakehash;make;
$cd ../COGreadblast;make;
$cd ../COGtriangles;make
```

Regarding BLAST, *get_homologues-est* uses BLAST+ binaries, which can be easily downloaded from the NCBI FTP site. The packed binaries are *blastp* and *makeblastdb* from version *ncbi-blast-2.2.27+*. If these do not work in your machine or your prefer to use older BLAST versions, then it will be necessary to edit file *lib/phyTools.pm*. First, environmental variable `$ENV{'BLAST_PATH'}` needs to be set to the right path in your system (inside subroutine `sub set_phyTools_env`).
Variables `$ENV{'EXE_BLASTP'}` and `$ENV{'EXE_FORMATDB'}` also need to be changed to the appropriate BLAST binaries, which are respectively *blastall* and *formatdb*.

2.3 Optional software dependencies

It is possible to make use of *get_homologues-est* on a computer farm or high-performance computing cluster managed by gridengine. In particular we have tested this feature with versions GE 6.0u8, 6.2u4, 2011.11p1 invoking the program with option `-m cluster`. For this command to work it might be necessary to edit the *get_homologues-est.pl* file and add the right path to set global variable `$SGEPATH`. To find out the installation path of your SGE installation you might try the next terminal command: `$ which qsub`
In case you have access to a multi-core computer you can follow the steps at
blog post to set up your own Grid Engine cluster and speed up your calculations.

For cluster-based operations three bundled Perl scripts are invoked:
`_cluster_makeHomolog.pl`, `_cluster_makeInparalog.pl` and `_cluster_makeOrtholog.pl` .

It is also possible to invoke Pfam domain scanning from *get_homologues-est*. This option requires the bundled binary *hmmscan*, which is part of the HMMER3 package, whose path is set in file *lib/phyTools.pm* (variable `$ENV{'EXE_HMMPFAM'}`). Should this binary not work in your system, a fresh install might be the solution, say in `/your/path/hmmer-3.1b2/`. In this case you'll have to edit file *lib/phyTools.pm* and modify the relevant:

```
if( ! defined($ENV{'EXE_HMMPFAM'}) )
{
$ENV{'EXE_HMMPFAM'} = '/your/path/hmmer-3.1b2/src/hmmscan --noali --acc --cut_ga ';
}
```

The Pfam HMM library is also required and the *install.pl* script should take care of it. However, you can manually download it from the appropriate Pfam FTP site. This file needs to be decompressed, either in the default *db* folder or in any other location, and then it should be formatted with the program *hmmpress*, which is also part of the *HMMER3* package. A valid command sequence could be:

```
$ cd db;
$ wget ftp://ftp.sanger.ac.uk/pub/databases/Pfam/current_release/Pfam-A.hmm.gz .;
$ gunzip Pfam-A.hmm.gz;
$ /your/path/hmmer-3.1b2/src/hmmpress Pfam-A.hmm
```

Finally, you'll need to edit file *lib/phyTools.pm* and modify the relevant line to:

```
if( ! defined($ENV{"PFAMDB"}) ){ $ENV{"PFAMDB"} = "db/Pfam-A.hmm"; }
```

In order to reduce the memory footprint of *get_homologues-est* it is possible to take advantage of the Berkeley_DB database engine, which requires Perl core module DB_File, which should be installed on all major Linux distributions.

The accompanying script *transcripts2cds.pl* should work out of the box, but the more efficient *transcripts2cdsCPP.pl* requires the installation of module Inline::CPP, which in turn requires Inline::C and g++, the GNU C++ compiler. The installation of these modules is known to be troublesome in some systems, but the standard way should work in most cases:

```
$ yum -y install gcc-c++ perl-Inline-C perl-Inline-CPP  # Redhat and derived distros

$ sudo apt-get -y install g++             # Ubuntu/Debian-based distros, and then cpan below

$ cpan -i Inline::C Inline::CPP           # will require administrator privileges (sudo)
```

Similarly, in order to take full advantage of the accompanying script *parse_pangenome_matrix.pl*, particularly for option `-p`, the installation of module GD is recommended. An easy way to install them, provided that you have administrator privileges, is with help from the software manager of your Linux distribution (such as *synaptic/apt-get* in Ubuntu, *yum* in Fedora or *YaST* in openSUSE).

This can usually be done on the terminal as well, in different forms:

```
$ sudo apt-get -y install libgd-gd2-perl  # Ubuntu/Debian-based distros

$ yum -y install perl-GD                  # Redhat and derived distros

$ zypper --assume-yes install perl-GD     # SuSE

$ cpan -i GD                              # will require administrator privileges (sudo)

$ perl -MCPAN -e 'install GD'             # will require administrator privileges (sudo)
```

The installation of perl-GD on macOSX systems is known to be troublesome.

The accompanying scripts *compare_clusters.pl*, *plot_pancore_matrix.pl*, *parse_pangenome_matrix.pl*, *plot_matrix_heatmap.sh*, *hcluster_matrix.sh* require the installation of the statistical software R, which usually is listed by software managers in all major Linux distributions. In some cases (some SuSE versions and some Redhat-like distros) it will be necessary to add a repository to your package manager. R can be installed from the terminal:

```
$ sudo apt-get -y install r-base r-base-dev      # Ubuntu/Debian-based distros

$ yum -y install R                               # RedHat and derived distros

$ zypper --assume-yes R-patched R-patched-devel  # Suse
```

Please visit CRAN to download and install R on macOSX systems, which is straightforward.

In addition to R itself, *plot_matrix_heatmap.sh* and *hcluster_matrix.sh* require some R packages to run, which can be easily installed from the R command line with:

```
> install.packages(c("ape", "gplots", "cluster"), dependencies=TRUE)
```

Finally, the script *compare_clusters.pl* might require the installation of program PARS from the PHYLIP suite, which should be already bundled with your copy of *get_homologues*.

3 User manual

This section describes the available options for the *get_homologues-est* software.

3.1 Input data

This program takes input sequences in FASTA format, which might be GZIP- or BZIP2-compressed, contained in a directory or folder containing several files with extension '.fna', which can have twin `.faa` files with translated amino acid sequences for the corresponding CDSs. File names matching the tag 'flcdna' are handled as full-length transcripts, and this information will be used downstream in order to estimate coverage. Global variable `$MINSEQLENGTH` controls the minimum length of sequences to be considered; the default value is 20.

3.2 Program options

Typing `$ ./get_homologues-est.pl -h` on the terminal will show the basic options:

```
-v print version, credits and checks installation
-d directory with input FASTA files (.fna , optionally .faa),  (use of pre-clustered sequences
   1 per sample, or subdirectories (subdir.clusters/subdir_)    ignores -c)
   with pre-clustered sequences (.faa/.fna ). Files matching
   tag 'flcdna' are handled as full-length transcripts.
   Allows for files to be added later.
   Creates output folder named 'directory_est_homologues'

Optional parameters:
-o only run BLAST/Pfam searches and exit                       (useful to pre-compute searches)
-i cluster redundant isoforms, including those that can be     (min overlap, default: -i 40,
   concatenated with no overhangs, and perform                  use -i 0 to disable)
   calculations with longest
-c report transcriptome composition analysis                   (follows order in -I file if enforced,
                                                                ignores -r,-e)
-R set random seed for genome composition analysis             (optional, requires -c, example -R 1234)
-s save memory by using BerkeleyDB; default parsing stores
   sequence hits in RAM
-m runmode [local|cluster]                                     (default: -m local)
-n nb of threads for BLASTN/HMMER/MCL in 'local' runmode       (default=2)
-I file with .fna files in -d to be included                   (takes all by default, requires -d)

Algorithms instead of default bidirectional best-hits (BDBH):
-M use orthoMCL algorithm (OMCL, PubMed=12952885)

Options that control sequence similarity searches:
-C min %coverage of shortest sequence in BLAST alignments      (range [1-100],default: -C 75)
-E max E-value                                                 (default: -E 1e-05 , max=0.01)
-D require equal Pfam domain composition                       (best with -m cluster or -n threads)
   when defining similarity-based orthology
-S min %sequence identity in BLAST query/subj pairs            (range [1-100],default: -S 1 [BDBH|OMCL]
-b compile core-transcriptome with minimum BLAST searches      (ignores -c [BDBH])

Options that control clustering:
-t report sequence clusters including at least t taxa          (default: t=numberOfTaxa,
                                                                t=0 reports all clusters [OMCL])
-L add redundant isoforms to clusters                          (optional, requires -i)
-r reference transcriptome .fna file                           (by default takes file with
                                                                least sequences; with BDBH sets
                                                                first taxa to start adding genes)
-e exclude clusters with inparalogues                          (by default inparalogues are
                                                                included)
-F orthoMCL inflation value                                    (range [1-5], default: -F 1.5 [OMCL])
-A calculate average identity of clustered sequences,          (optional, creates tab-separated matrix,
 uses blastn results                                            recommended with -t 0 [OMCL])
-z add soft-core to genome composition analysis                (optional, requires -c [OMCL])
```

The only required option is `-d`, which indicates an input folder, as seen in section 3.1. It is important to remark that in principle only files with extensions `.fna / .fa` and optionally `.faa` are considered when parsing the `-d` directory. By using `.faa` input files protein sequences can be used to scan Pfam domains and included in output clusters.

The use of an input folder or directory (`-d`) is recommended as it allows for new files to be added there in the future, reducing the computing required for updated analyses. For instance, if a user does a first analysis

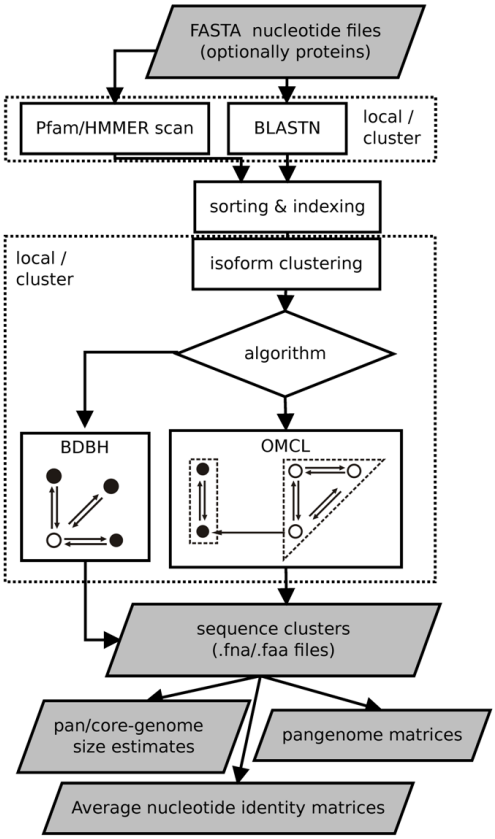

Figure 1: Flowchart of get_homologues-est.

name	option	
BDBH	default	Starting from a reference genome, keep adding genomes stepwise while storing the sequence clusters that result of merging the latest bidirectional best hits.
OMCL	`-M`	OrthoMCL v1.4, uses the Markov Cluster Algorithm to group sequences, with inflation (`-F`) controlling cluster granularity, as described in PubMed=12952885.

Table 1: List of available clustering algorithms. Note that the COG triangles algorithm is not supported.

with 5 input genomes today, it is possible to check how the resulting clusters would change when adding an extra 10 genomes tomorrow, by copying these new 10 `.faa` / `.gbk` input files to the pre-existing `-d` folder, so that all previous BLAST searches are re-used.

All remaining flags are options that can modify the default behavior of the program, which is to use the bidirectional best hit algorithm (BDBH) in order to compile clusters of potential orthologous DNA sequences, taking the smallest genome as a reference. By default nucleotide sequences are used to guide the clustering, thus relying on BLASTN searches.

Perhaps the most important optional parameter would be the choice of clustering algorithm (Table 1):

The remaining options are now reviewed:

- Apart from showing the credits, option `-v` can be helpful after installation, for it prints the enabled features of the program.
- `-o` is ideally used to submit to a computer cluster the required BLAST (and Pfam) searches, preparing a job for posterior analysis on a single computer.
- `-i` can be used to filter out short, redundant isoforms which overlap, with no overhangs, for a minimun length. By default this is set to `$MINREDOVERLAP=40` as in PubMed=12651724. This EST-specific feature

can be turned off by setting `-i 0`. Redundant isoforms will not be output unless `-L` is set.

- `-c` is used to request a pan- and core-genome analysis of the input sequences, which will be output as tab-separated files. The number of samples for the genome composition analysis is set to 20 by default, but this can be edited at the header of `get_homologues-est.pl` (check the `$NOFSAMPLESREPORT` variable). As *get_homologues-est* is meant to be used primarily for the study of transcripts/CDSs of the same species, it uses appropriate thresholds to define new accessory genes (`$MIN_PERSEQID_HOM=95`, `$MIN_COVERAGE_HOM=75`), which mean that genes/transcripts added to the pool must be $< 95\%$ identical in sequence to any previous sequence (as in PubMed=21572440)) with cover $> 75\%$. Note that these default values are different to those in *get_homologues*. When combined with flag `-t` (see below), the composition analysis will disregard clusters reported in an arbitrary number of strains. This feature can be used to filter out singletons or artifacts which might arise from *de novo* assembled transcriptomes.

- `-R` takes a number that will be used to seed the random generator used with option `-c`. By using the same seed in different `-c` runs the user ensures that genomes are sampled in the same order.

- `-s` can be used to reduce the memory footprint, provided that the Perl module BerkeleyDB is in place. This option usually makes *get_homologues-est* slower, but for very large datasets or in machines with little memory resources this might be the only way to complete a job.

- `-m` allows the choice of runmode, which can be either `-m local` (the default) or `-m cluster`. In the second case global variable `$SGEPATH` might need to be appropriately set, as explained in *manual_get_homologues.pdf*, as well as `$QUEUESETTINGS`, that specificies for instance a particular queue name for your cluster jobs.

- `-n` sets the number of threads/CPUs to dedicate to each BLAST/HMMER/mcl job run locally, which by default is 2.

- `-I list_file.txt` allows the user to restrict a *get_homologues-est* job to a subset of FASTA files included in the input `-d` folder. This flag can be used in conjunction with `-c` to control the order in which genomes are considered during pan- and core-transcriptome analyses. Taking the `sample_RNAseq` folder, a valid `list_file.txt` could contain these lines:

  ```
  Esterel.trinity.fna.bz2
  Franka.trinity.fna.bz2
  ```

- option `-C` sets the minimum percentage of coverage required to call two sequences best hits. As EST/transcripts are frequently truncated, by default coverage is calculated with respect to the shortest sequence in the pair, unless both of them come from a full-length collection (see 3.1).

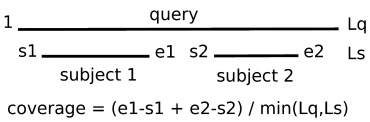

Figure 2: Coverage illustrated with the alignment of sequence 'query' to two aligned fragments of sequence 'subject'. Lq and Ls are the lengths of both sequences, and s1,e1,s2,e2 and Lq are alignment coordinates.

- `-E` sets the maximum expectation value (E-value) for BLASTN alignments. This value is by default set to 1e-05.

- `-D` is an extra restriction for calling best hits, that should have identical Pfam domain compositions. Note that this option requires scanning all input sequences for Pfam domains, and this task requires extra computing time, ideally on a computer cluster (`-m cluster`). While for BDBH domain filtering is done at the time bidirectional best hits are called, this processing step is performed only after the standard OMCL algorithms have completed, to preserve the algorithm features.

- `-S` can be passed to require a minimum % sequence identity for two sequences to be called best hits. The default value is set to 1.

- `-b` reduces the number of pairwise BLAST searches performed while compiling core-genomes with algorithm BDBH, reducing considerably memory and run-time requirements (for G genomes, 3G searches are launched instead of the default G^2). It comes at the cost of being less exhaustive in finding inparalogues, but in our bacterial benchmarks this potential, undesired effect was negligible.

- `-t` is used to control which sequence clusters should be reported. By default only clusters which include at least one sequence per genome are output. However, a value of `-t 2` would report all clusters containing sequences from at least 2 taxa. A especial case is `-t 0`, which will report all clusters found, even those with sequences from a single genome.

- `-r` allows the choice of any input sequence set (of course included in `-d` folder) as the reference, instead of the default smaller one. If possible, resulting clusters are named using CDS/transcript names from this genome, which can be used to select well annotated species for this purpose.

- `-e` excludes clusters with inparalogues, defined as sequences with best hits in its own genome. This option might be helpful to rule out clusters including several sequences from the same species, which might be of interest for users employing these clusters for primer design, for instance.

- `-F` is the inflation value that governs Markov Clustering in OMCL runs, as explained in PubMed=12952885. As a rule of thumb, low inflation values (`-F 1`)result in the inclusion of more sequences in fewer groups, whilst large values produce more, smaller clusters (`-F 4`).

- `-A` tells the program to produce a tab-separated file with average % sequence identity values among pairs of genomes, computed from sequences in the final set of clusters (see also option `-t`). By default these identities are derived from BLASTN alignments, and hence correspond to nucleotide sequence identities, to produce genomic average nucleotide sequence identities (ANI).

- `-z` can be called when performing a genome composition analysis with clustering algorithm OMCL. In addition to the core- and pan-genome tab-separated files mentioned earlier (see option `-c`), this flag requests a soft-core report, considering all sequence clusters present in a fraction of genomes defined by global variable `$SOFTCOREFRACTION`, with a default value of 0.95. This choice produces a composition report more robust to assembly or annotation errors than the core-genome.

3.3 Accompanying scripts

The following Perl and shell scripts are included in each release to assist in the interpretation of results generated by *get_homologues-est.pl*. See examples of use in *manual_get_homologues.pdf*:

- *compare_clusters.pl* primarily calculates the intersection between cluster sets, which can be used to select clusters supported by different algorithms or settings. This script can also produce pangenome matrices and Venn diagrams.
- *parse_pangenome_matrix.pl* is a script that can be used to analyze pan-genome sets, in order to find transcripts/genes present in a group A of strains which are absent in set B. This script can also be used for calculating and plotting cloud, shell and core genome compartments.
- *make_nr_pangenome_matrix.pl* is provided to post-process pangenome matrices in case the user wishes to remove redundant clusters, using either nucleotide or protein sequence identity cut-offs.
- *plot_pancore_matrix.pl*, a Perl script to plot pan/soft/core-genome sampling results and to fit regression curves with help from R functions.
- *check_BDBHs.pl* is a script that can be used, after a previous *get_homologues-est* run, to find out the bidirectional best hits of a sequence identifier chosen by the user. It can also retrieve the Pfam annotations of a sequence and its reciprocal best hits.
- *add_pancore_matrices.pl* can be used to add pan/core-matrices produced by previous *get_homologues.est -c -R* runs on the same set of genomes, with the aim of combining clusters.
- *plot_matrix_heatmap.sh* calculates ordered heatmaps with attached row and column dendrograms from tab-separated numeric matrices, which can be presence/absence pangenomic matrices or similarity / identity matrices as those produced by *get_homologues-est* with flag `-A`.
- *hcluster_matrix.sh* generates a distance matrix out of a tab-separated numeric matrix, which is then used to call R functions `hclust()` and `heatmap.2()` in order to produce a heatmap.

Apart from these, auxiliar *transcripts2cds.pl* script is bundled to assist in the analysis of transcripts. In particular, this script can be used to annotate potential Open Reading Frames (ORFs) contained within raw transcripts, which might be truncated or contain introns. This script uses TransDecoder, BLASTX and SWISSPROT, which should be installed by running: *./install.pl*

```
usage: ./transcripts2cds.pl [options] <input FASTA file(s) with transcript nucleotide sequences>

-h this message
-p check only 'plus' strand                                  (optional, default both strands)
-l min length for CDS                                        (optional, default=50 amino acid residues)
-g genetic code to use during translation                    (optional, default=1, example: -g 11)
-d run blastx against selected protein FASTA database file   (default=swissprot, example: -d db.faa)
-E max E-value during blastx search                          (default=1e-05)
-n number of threads for BLASTX jobs                         (default=2)

-G show available genetic codes and exit
```

4 A few examples

This section presents typical ways of running *get_homologues-est.pl* and the accompanying scripts with provided sample input data. Please check file *manual_get_homologues.pdf* for more examples, particularly for the auxiliary scripts, which are not explained in this document.

4.1 Extracting coding sequences from transcripts

This example takes the provided sample file `sample_transcripts.fna` to demonstrate how to annotate coding sequences contained in those sequences by calling `transcripts2cds.pl`. Note that `transcripts2cdsCPP.pl` is significantly faster, but requires an optional Perl module (see 2.3).

This is an optional pre-processing step which you might not want to do, as the software should be able to properly handle any nucleotides sequences suitable for BLASTN. However, coding sequences have the advantage that can be translated to amino acids and thus used to scan Pfam domains further down in the analysis (see option `-D`).

A simple command would be, which will discard sequences less than 50b long, and will aligned them to SWISSPROT proteins in order to annotate coding regions. In case of overlap, Transdecoder-defined and BLASTX-based coding regions are combined provided that a `$MINCONOVERLAP=90` overlap, with no mismatches, is found; otherwise the latter are given higher priority:

```
./transcripts2cdsCPP.pl -n 10 sample_transcripts.fna
```

The output should look like this (contained in file `sample_transcripts_output.txt`):

```
# ./transcripts2cdsCPP.pl -p 0 -m  -d /path/get_homs-est/db/swissprot -E 1e-05 -l 50 -g 1 -n 10
# input files(s):
# sample_transcripts.fna

## processing file sample_transcripts.fna ...
# running transdecoder...
# parsing transdecoder output (_sample_transcripts.fna_minl50.transdecoder.cds.gz) ...
# running blastx...
# parsing blastx output (_sample_transcripts.fna_eval1e-05.blastx.gz) ...
# calculating consensus sequences ...
# input transcripts = 9
# transcripts with ORFs = 7
# transcripts with no ORFs = 2
# output files: sample_transcripts.fna_minl50_eval1e-05.transcript.fna ,
# sample_transcripts.fna_minl50_eval1e-05.cds.fna ,
# sample_transcripts.fna_minl50_eval1e-05.cds.faa ,
# sample_transcripts.fna_minl50_eval1e-05.noORF.fna
```

The resulting CDS files can be then analyzed with *get_homologues-est.pl*.

Apart from the listed output files, which include translated protein sequences, temporary files are stored in the working directory, which of course can be removed, but will be re-used if the same job is re-run later, such as `_sample_transcripts.fna_eval1e-05.blastx.gz`, `_sample_transcripts.fna_minl50.transdecoder.cds.gz` and `_sample_transcripts.fna_minl50.transdecoder.pep.gz`.

4.2 Clustering orthologous transcripts from FASTA files, one per strain

This example takes the sample input folder `sample_transcripts_fasta`, which contains automatically assembled transcripts (Trinity) of three *Hordeum vulgare* strains (barley), plus a set of full-length cDNA collection of cultivar *Haruna Nijo*, to show to produce clusters of transcripts.

The next command uses the OMCL algorithm to cluster sequences, produces a composition report, including the soft-core, and finally computes an Average Nucleotide Identity matrix on the produced clusters. Note that redundant isoforms are filtered, keeping only the longest one (you can turn this feature off with `-i 0`):

```
$ ./get_homologues-est.pl -d sample_transcripts_fasta -M -c -z -A  .
```

The output should look like this (contained in file `sample_output_est.txt`):

```
# results_directory=/path/sample_transcripts_fasta_est_homologues
# parameters: MAXEVALUEBLASTSEARCH=0.01 MAXPFAMSEQS=250 BATCHSIZE=100

# checking input files...
# Esterel.trinity.fna.bz2 5892  median length = 506
# Franka.trinity.fna.bz2 6036  median length = 523
# Hs_Turkey-19-24.trinity.fna.bz2 6204  median length = 476
# flcdnas_Hnijo.fna.gz 28620 [full length sequences] median length = 1504

# 4 genomes, 46752 sequences

# taxa considered = 4 sequences = 46752 residues = 63954041

# mask=Esterel_alltaxa_algOMCL_e0_ (_algOMCL)
[...]

# re-using previous isoform clusters
# 42 sequences
# 65 sequences
# 61 sequences
# 2379 sequences

# creating indexes, this might take some time (lines=2.08e+05) ...

# construct_taxa_indexes: number of taxa found = 4
# number of file addresses/BLAST queries = 4.4e+04

# genome composition report (samples=20,permutations=24,seed=0)
# genomic composition parameters: MIN_PERSEQID_HOM=95 MIN_COVERAGE_HOM=75 SOFTCOREFRACTION=0.95
[...]

# file=sample_transcripts_fasta_est_homologues/core_genome_algOMCL.tab
genomes mean stddev | samples
0 8744 7017 | 4610 4610 4610 ...
1 1157 734 | 540 475 2042 ...
2 296 109 | 109 354 387 ...
3 91 0 | 91 91 91 ...

# file=sample_transcripts_fasta_est_homologues/soft-core_genome_algOMCL.tab
genomes mean stddev | samples
0 8744 7017 | 4610 4610 4610 ...
1 3538 2335 | 2462 2247 8205 ...
2 2219 1005 | 837 3505 2182 ...
3 716 109 | 903 658 625 ...
```

```
# clustering orthologous sequences

# looking for valid sequence clusters (n_of_taxa=4)...

# number_of_clusters = 91
# cluster_list = sample_transcripts_fasta_est_homologues/Esterel_alltaxa_algOMCL_e0_.cluster_list
# cluster_directory = sample_transcripts_fasta_est_homologues/Esterel_alltaxa_algOMCL_e0_

# average_nucleotide_identity_matrix_file =
# sample_transcripts_fasta_est_homologues/Esterel_alltaxa_algOMCL_e0_Avg_identity.tab
```

Notice that both core and soft-core sampling experiments are reported, considering genes found in all strains and also in 95% strains, respectively.

The produced Average Nucleotide Identity matrix looks like this:

genomes	Esterel	Franka	HsTurkey	flcdnasHnijo
Esterel	100	96.96	95.88	95.87
Franka	96.96	100	96.01	96.31
HsTurkey	95.88	96.01	100	93.25
flcdnasHnijo	95.87	96.31	93.25	100

Provided that optional R modules described in *manual_get_homologues.pdf* are installed, This matrix can be plotted with:

```
./plot_matrix_heatmap.sh -i sample_[...]/Esterel_alltaxa_algOMCL_e0_Avg_identity.tab
```

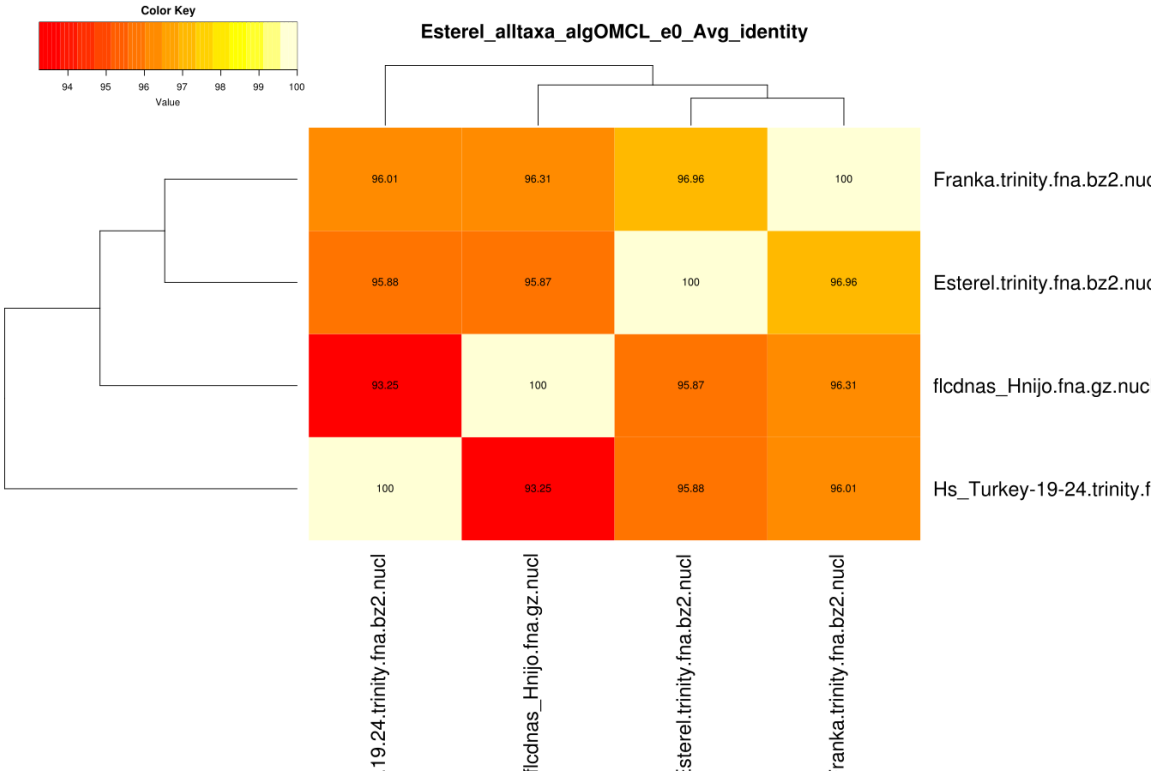

Figure 3: Plot of Average Nucleotide Identity matrix

If the previous command is changed by adding option `-t -2` only transcripts present in at least two strains will be considered, which are output in folder:

`sample_transcripts_fasta_est_homologues/Esterel_2taxa_algOMCL_e0_`
This second command produces a significantly different pan-genome composition matrix, which changes from:

```
# file=sample_transcripts_fasta_est_homologues/pan_genome_algOMCL.tab
genomes mean stddev | samples
0 8744 7017 | 4610 4610 4610 ...
1 15363 7026 | 8943 8955 22475 ...
2 22607 5513 | 13106 25909 24897 ...
3 28392 539 | 29155 28210 27562 ...
```

to

```
# file=sample_transcripts_fasta_est_homologues/pan_genome_2taxa_algOMCL.tab
genomes mean stddev | samples
0 9145 7695 | 4617 4617 4617 ...
1 10646 7019 | 6550 6245 6643 ...
2 11444 6689 | 8003 7308 6862 ...
3 11549 6687 | 8123 7410 6959 ...
```

Both matrices can be plotted with script *plot_pancore_matrix.pl*, with a command such as:
`./plot_pancore_matrix.pl -i sample_transcripts_fasta_est_homologues/pan_genome_algOMCL.tab -f pan`

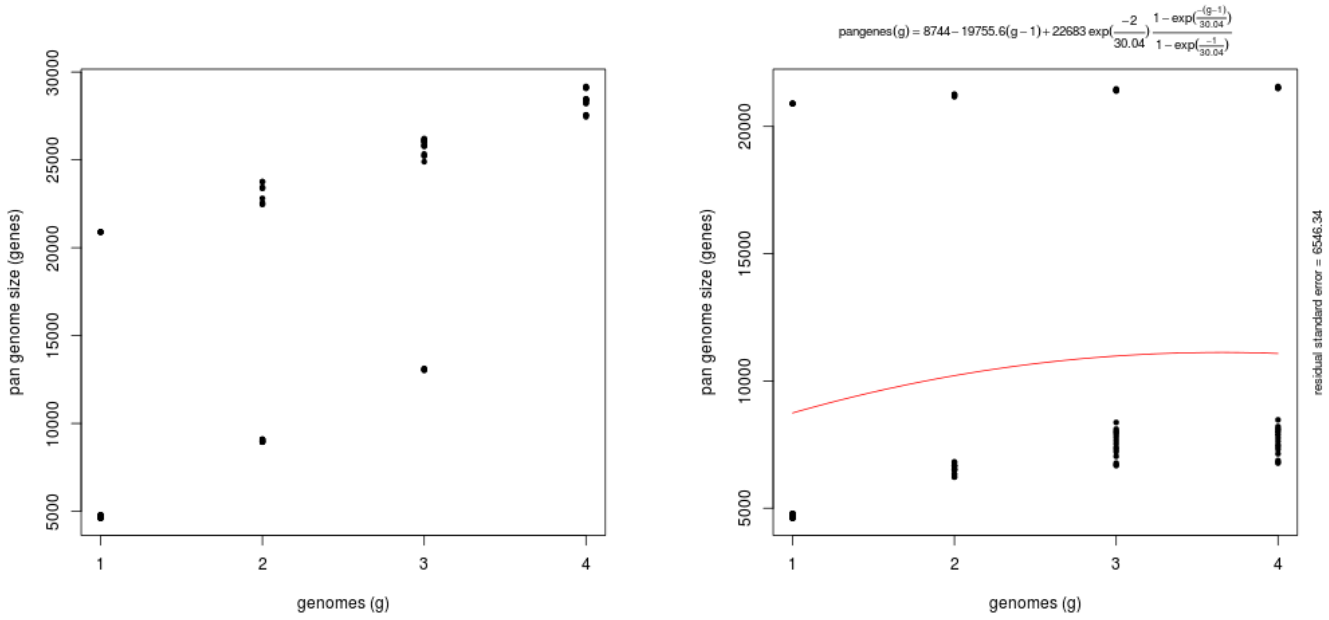

Figure 4: Pan-genome (-t 0, left) and pan-genome (-t 2, right) estimates based on random samples of 4 strains. As the left example illustrates, four strains are usually not enough to fit a Tettelin-like function.

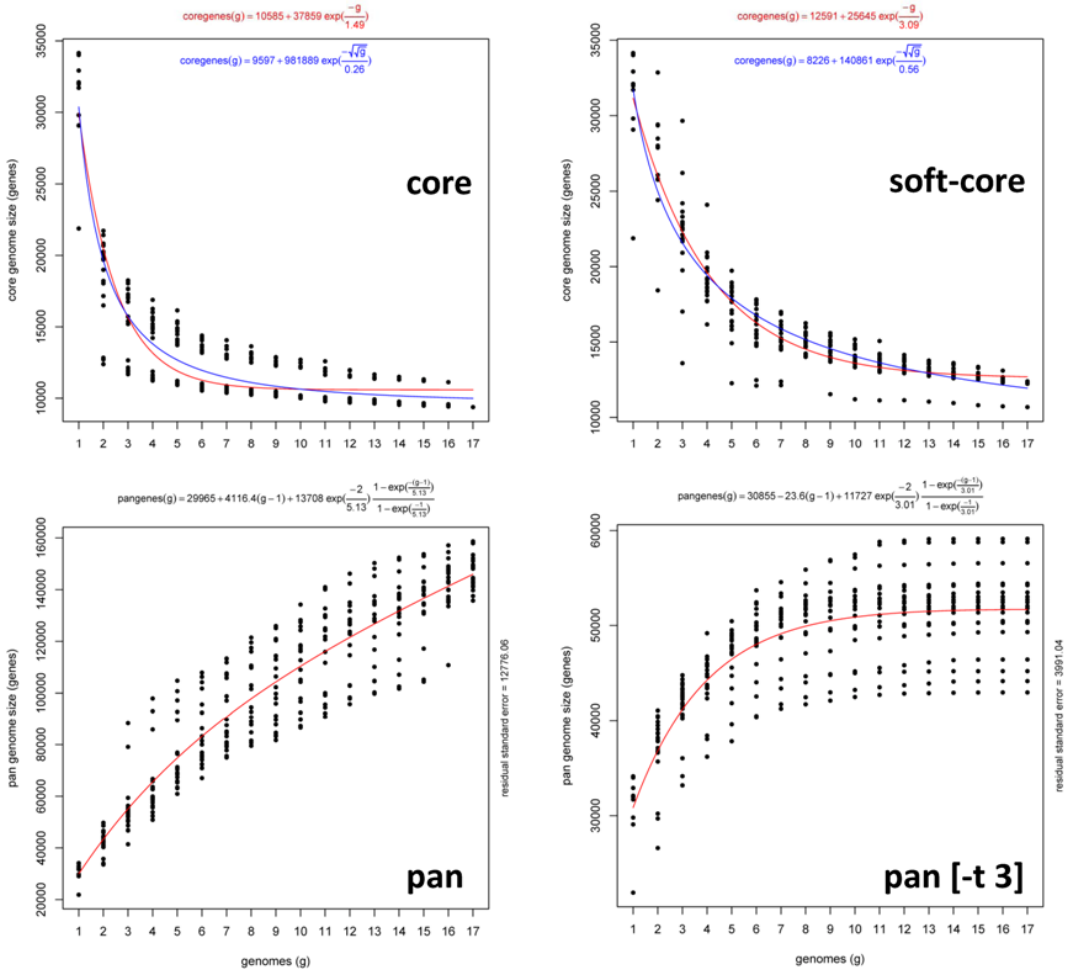

Figure 5: Core-genome, soft-core-genome and pan-genome composition analysis of 17 barley transcript sets. Note that the pan-genome simulation was done with all clusters (left) and with all clusters found in at least three transcriptomes (right), illustrating the effect of option -t 3, which might be useful to remove low confidence sequences.

4.3 Producing a nucleotide-based pangenome matrix

The clusters obtained in the previous section with option `-t 2` can be used to compile a pangenome matrix without singletons with this command:
`./compare_clusters.pl -d sample_[...]/Esterel_2taxa_algOMCL_e0_ -o outdir -n -m`

```
# number of input cluster directories = 1

# parsing clusters in sample_transcripts_fasta_est_homologues/Esterel_2taxa_algOMCL_e0_ ...
# cluster_list in place, will parse it (sample_[...]/Esterel_2taxa_algOMCL_e0_.cluster_list)
# number of clusters = 5259

# intersection output directory: outdir

# intersection size = 5259 clusters

# intersection list = outdir/intersection_t0.cluster_list

# pangenome_file = outdir/pangenome_matrix_t0.tab
# pangenome_phylip file = outdir/pangenome_matrix_t0.phylip
```

If the optional R modules described in *manual_get_homologues.pdf* are installed, such a pangenome matrix can be used to hierarchically cluster strains with this command:
`./hcluster_matrix.sh -i outdir/pangenome_matrix_t0.tab`

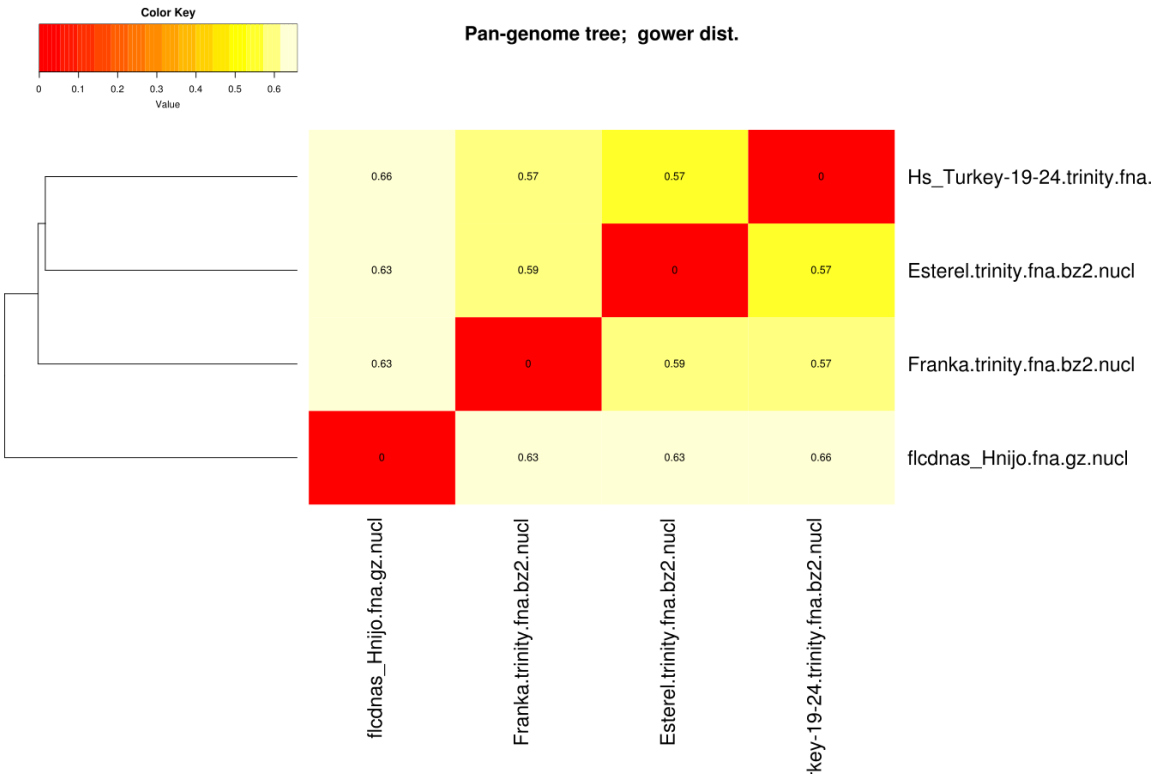

Figure 6: Hierarchical grouping of strains based on pangenome matrix.

4.4 Making and annotating a non-redundant pangenome matrix

Script `make_nr_pangenome_matrix.pl` produces a non-redundant pangenome matrix by comparing all clusters to each other, taking the median sequence in each cluster. If the original input of *get_homologues-est* comprised both DNA and protein sequences, the user can choose which kind of sequence will be used to compute redundancy. In terms of protein function probably it makes sense to use amino acid sequences for this task. On the contrary, it would seem more appropriate to use DNA sequences to measure diversity.

In this example a DNA-based non-redundand pangenome matrix is computed with BLASTN (option `-n`) asuming that sequence might be truncated (option `-e`) and using 10 processor cores and using a coverage cutoff of 50%:

```
./make_nr_pangenome_matrix.pl -m outdir/pangenome_matrix_t0.tab -n -N 10 -e -C 50

# input matrix contains 5259 clusters and 4 taxa

# filtering clusters ...
# 5259 clusters with taxa >= 1 and sequence length >= 0

# sorting clusters and extracting median sequence ...

# running makeblastdb with outdir/pangenome_matrix_t0_nr_t1_l0_e1_C50_S90.fna

# parsing blast result! (outdir/pangenome_matrix_t0_nr_t1_l0_e1_C50_S90.blast , 0.34MB)
# parsing file finished

# 5226 non-redundant clusters
# created: outdir/pangenome_matrix_t0_nr_t1_l0_e1_C50_S90.fna

# printing nr pangenome matrix ...
# created: outdir/pangenome_matrix_t0_nr_t1_l0_e1_C50_S90.tab
```

Note that the previous command can be modified to match clusters to a set of pre-computed clusters, such as clusters of orthologous sequences or sequences from Swissprot, so that the resulting matrix contains cross-references to those external clusters, and their annotations.

5 Frequently asked questions (FAQs)

Please see also the FAQs on emphmanual_get_homologues.pdf.

- What's the performance gain of v2?

  After evolving parts of the original code base, and fixing some bugs (see CHANGES.txt), both *get_homologues.pl* and *get_homologues-est.pl* have significantly improved their performance, as can be seen in the figure, which combines data from the original benchmark and new data generated after v2 was in place.

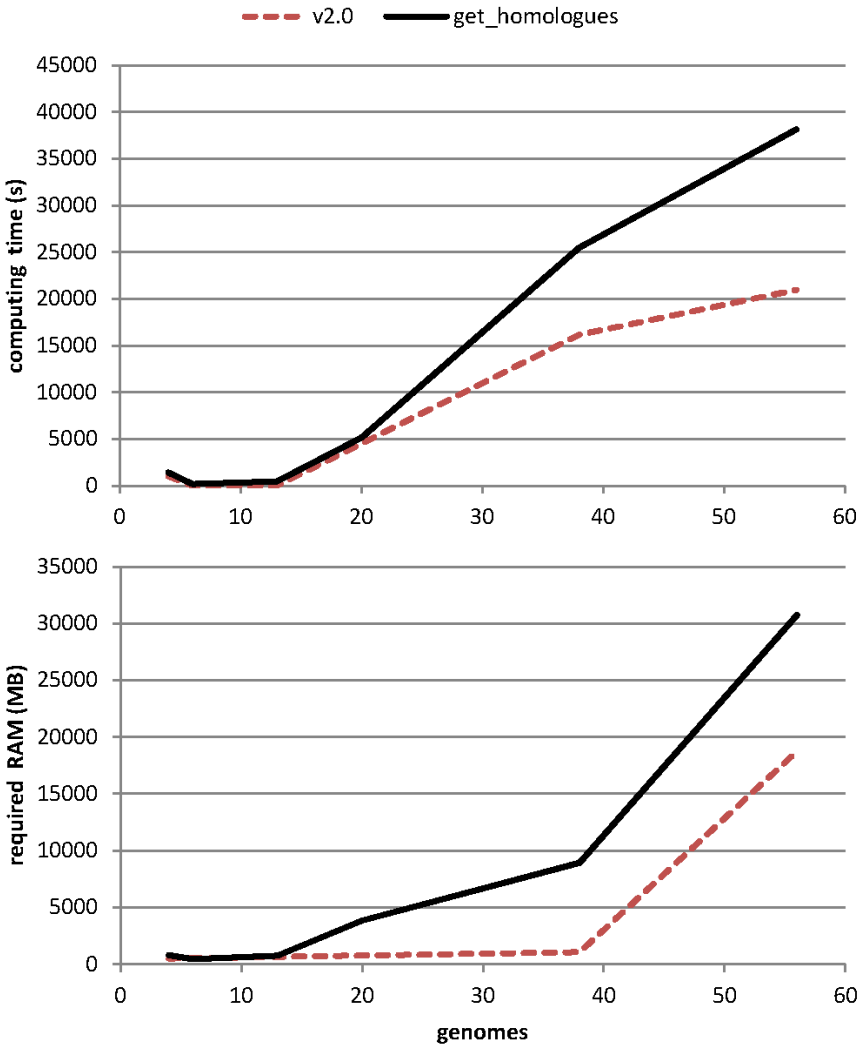

Figure 7: Computing time and RAM requirements of the original algorithm (OMCL, measured on 6 sequence sets) as compared to the updated v2 code (measured on 3 three sets).

- What are the main caveats when clustering transcripts/CDS sequences?

  get_homologues-est.pl has been mainly tested with plant sequences, using both CDS sets from whole-genome annotations and also transcripts from expression experiments. The main problems we have found so far are split genes, frequent artifacts in genome assemblies, incomplete genes which lack exons, for the same previous reasons, and retained introns, which are common among plant transcripts. These three common situations are illustrated on Figure 8.

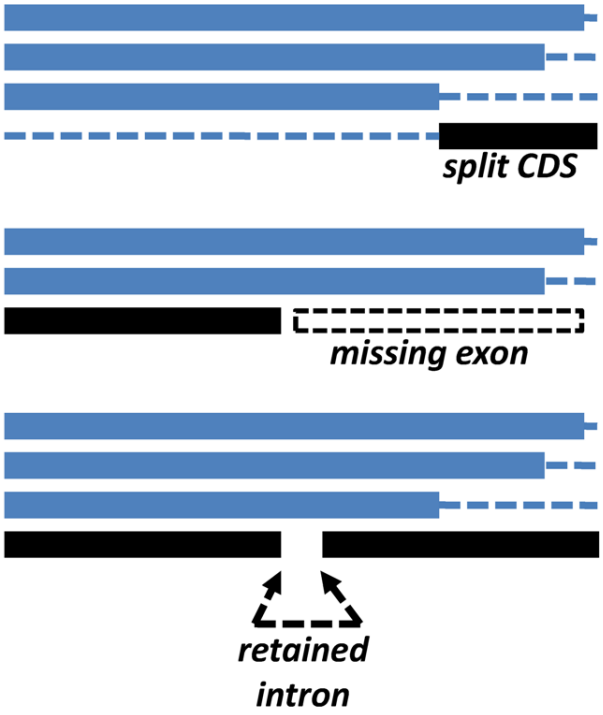

Figure 8: Common problems faced when clustering transcripts/CDS sequences.

6 Credits and references

get_homologues-est.pl is designed, created and maintained at the Laboratory of Computational Biology at Estación Experimental de Aula Dei/CSIC in Zaragoza (Spain) and at the Center for Genomic Sciences of Universidad Nacional Autónoma de México (CCG/UNAM).

The code was written mostly by Bruno Contreras-Moreira and Pablo Vinuesa, but it also includes code and binaries from OrthoMCL v1.4 (algorithm OMCL, `-M`), NCBI Blast+ and BioPerl 1.5.2

We ask the reader to cite the main reference describing the *get_homologues* software,

- Contreras-Moreira,B. and Vinuesa,P. (2013) GET_HOMOLOGUES, a versatile software package for scalable and robust microbial pangenome analysis. Appl.Environ.Microbiol. 79:7696-7701.

and also the original papers describing the included algorithms and databases, accordingly:

- Li L, Stoeckert CJ Jr, Roos DS (2003) OrthoMCL: identification of ortholog groups for eukaryotic genomes. Genome Res. 13(9):2178-89.
- Altschul SF, Madden TL, Schaffer AA, Zhang J, Zhang Z, Miller W and Lipman DJ (1997) Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. Nucl. Acids Res. 25(17): 3389-3402.
- Stajich JE, Block D, Boulez K, Brenner SE, Chervitz SA, Dagdigian C, Fuellen G, Gilbert JG, Korf I, Lapp H, Lehvslaiho H, Matsalla C, Mungall CJ, Osborne BI, Pocock MR, Schattner P, Senger M, Stein LD, Stupka E, Wilkinson MD, Birney E. (2002) The Bioperl toolkit: Perl modules for the life sciences. Genome Res. 12(10):1611-8.
- hmmscan :: search sequence(s) against a profile database HMMER 3.1b2 (Feb 2015) http://hmmer.org Copyright (C) 2015 Howard Hughes Medical Institute. Freely distributed under the GNU General Public License (GPLv3).

- Finn RD, Bateman A, Clements J, Coggill P, Eberhardt RY, Eddy SR, Heger A, Hetherington K, Holm L, Mistry J, Sonnhammer EL, Tate J, Punta M. (2014) Pfam: the protein families database. Nucleic Acids Res. 42:D222-30.

If you use the accompanying scripts the following references should also be cited:

- R Core Team (2013) R: A Language and Environment for Statistical Computing. http://www.R-project.org R Foundation for Statistical Computing, Vienna, Austria, ISBN3-900051-07-0