
Lumache

Release 0.1

Graziella

Jun 08, 2022

CONTENTS

1	Contents	3
1.1	Explore the pangenome - PanTools manual	3

Lumache (/lu'make/) is a Python library for cooks and food lovers that creates recipes mixing random ingredients. It pulls data from the [Open Food Facts database](#) and offers a *simple* and *intuitive* API.

Check out the usage section for further information, including how to install the project.

Note: This project is under active development.

CONTENTS

1.1 Explore the pangenome - PanTools manual

PanTools manual

- [Home](#)
- [Install](#)
- [Construct pangenome](#)
- [Pangenome characterization](#)
- [Phylogeny](#)
- [Sequence alignments](#)
- [Explore the pangenome](#)
 - *Gene locations*
 - * *Required arguments*
 - * *Optional arguments*
 - * *Example command*
 - * *Output files*
 - *Find genes*
 - * *Find genes by name*
 - *Required arguments*
 - *Optional arguments*
 - *Example command*
 - *Output files*
 - * *Find genes by annotation*
 - *Required arguments*
 - *Optional arguments*
 - *Example command*
 - *Output files*
 - * *Find genes in region*
 - *Required arguments*

- *Optional arguments*
 - *Example input file*
 - *Example command*
 - *Output files*
- *Functional annotations*
 - * *Show GO*
 - *Required arguments*
 - *Example commands*
 - *Output file*
 - * *Compare GO*
 - *Required arguments*
 - *Example command*
 - *Output file*
- *Homology group information*
 - * *Required arguments*
 - * *Optional arguments*
 - * *Example command*
 - * *Output files*
- *Sequence alignments*
- *Matrix files*
 - * *Order matrix*
 - *Required argument*
 - *Optional argument*
 - *Example command*
 - *Output file*
 - * *Rename matrix*
 - *Required arguments*
 - *Optional arguments*
 - *Example command*
 - *Output file*
- *Retrieve regions, genomes or features*
 - * *Retrieve regions*
 - *Required arguments*
 - *Example command*
 - *Example input*
 - *Output file*

- * *Retrieve features*
 - *Required arguments*
- * *Optional arguments*
 - *Example command*
 - *Output files*

- [Read mapping](#)
- [Querying the database](#)
- [Differences between pangenome and panproteome](#)
- [Tutorial](#)

PanTools manual

- [Docs »](#)
- [Explore the pangenome](#)
-

Explore the pangenome

The functionalities on this page allow to actively explore the pangenome.

- Retrieve regions from the pangenome
- Retrieve sequences and functional annotations from homology groups
- Search for genes using a gene name, functional annotation or database node identifier
- Align homology groups or genomic regions
- GO enrichment analysis

Gene locations

Identify and compare gene clusters of neighbouring genes based on a set of homology groups. First, identifies the genomic position of genes in homology groups, retrieves the order of genes per genome and based on this construct the gene clusters. If homology groups with multiple genomes were selected, the gene cluster composition is compared between genomes. When a `--phenotype` is included, gene clusters can be found that only consist of groups of a certain phenotype.

For example, 100 groups were predicted as core in a pangenome of 5 genomes. The gene clusters are first identified per genome, whereafter it compares the gene order of one genome to all the other genomes. The result could be 75 groups with genes that are not only homologous but also share their gene neighbourhood. Another example, when accessory (present 2 in to 4 genomes) groups are given to this function in combination with a `--phenotype` (assigned to only two genomes), the function can return clusters that can only be found in the phenotype members.

Required arguments

- `--database-path/-dp` Path to the database.
- `--homology-groups/-hm` A text file with homology group node identifiers, separated by a comma.

Optional arguments

- `--phenotype/-ph` A phenotype name, used to identify gene clusters shared by all phenotype members.
- `--value` The number of allowed nucleotides between two neighbouring genes (default is 1 MB).
- `--gap-open/-go` When constructing the clusters, allow a number of genes for each cluster that are not originally part of the input groups (default is 0).
- `--core-threshold` Lower the threshold (%) for a group to be considered core/softcore (default is the total number of genomes found in the groups, not a percentage).
- `--skip/-sk` Exclude a selection of genomes.
- `--reference/-ref` Only include a selection of genomes.
- `--mode ignore-copies` Duplicated and co-localized genes no longer break up clusters.

Example command

```
$ pantools locate_genes -dp tomato_DB -hm phenotype_groups.csv
$ pantools locate_genes -dp tomato_DB -hm unique_groups.csv --value 5000 -go 1
$ pantools locate_genes -dp tomato_DB -hm accessory_groups.csv --core-threshold 95 -go 1
```

Output files

Output files are stored in *database_directory/locate_genes/*

- gene_clusters_by_position.txt**, the identified gene clusters ordered by their position in the genome.
- gene_clusters_by_size.txt**, the identified gene clusters ordered from largest to smallest.
- compare_gene_clusters**, the composition of found gene clusters is compared to the other genomes. For each cluster, it shows which parts match other clusters and which parts do not. The file is not created when homology groups only contain proteins of a single genome (unique).

When a `--phenotype` is included

- phenotype_clusters**, homology group node identifiers from phenotype shared and specific clusters.
- compare_gene_clusters_PHENOTYPE.txt**, the same information as **compare_gene_clusters** but now the gene cluster comparison is only done between phenotype members.

Find genes

Find genes by name

Find your genes of interest in the pangenome by using the gene name and extract the nucleotide and protein sequence. To be able to find a gene, every letter of the given input must match a gene name. The search is not case sensitive. Performing a search with 'sonic1' as query will not be able find 'sonic', but is able to find Sonic1, SONIC1 or sOnIc1. Including the `--mode 1` argument allows a more relaxed search and using 'sonic' will now also find gene name variations as 'sonic1', 'sonic3' etc..

Be aware, for this function to work it is important that genomes are annotated by a method that follows the rules for genetic nomenclature. Gene naming can be inconsistent when different tools are used for genome annotation, making this functionality ineffective.

This function is the same as `mlsa_find_genes` but uses a different output directory. Several warnings (shown in the other manual) can be generated during the search. These warnings are less relevant for this function as the genes are not required to be single copy-orthologous.

Required arguments

`--database-path/-dp` Path to the database.
`--name` One or multiple gene names, separated by a comma.

Optional arguments

`--skip/-sk` Exclude a selection of genomes.
`--reference/-ref` Exclude a selection of genomes.
`--mode extensive` Perform a more extensive gene search.

Example command

```
$ pantools find_genes_by_name -dp tomato_DB --name dnaX,gapA,recA
$ pantools find_genes_by_name -dp tomato_DB --name gapA --mode extensive
```

Output files

Output files are stored in `/database_directory/find_genes/by_name/`. For each gene name that was included, a nucleotide and protein and .FASTA file is created with sequences found in all genomes.

- **find_genes_by_name.log**, relevant information about the extracted genes: node identifier, gene location, homology group etc..

Find genes by annotation

Find genes of interest in the pangenome that share a functional annotation node and extract the nucleotide and protein sequence.

Required arguments

`--database-path/-dp` Path to the database.

Requires either **one** of the following arguments

`--node` One or multiple identifiers of function nodes (GO, InterPro, PFAM, TIGRFAM), separated by a comma.

`--name` One or multiple function identifiers (GO, InterPro, PFAM, TIGRFAM), separated by a comma.

Optional arguments

`--skip/-sk` Exclude a selection of genomes.

`--reference/-ref` Only include a selection of genomes.

Example command

```
$ pantools find_genes_by_annotation -dp tomato_DB --node 14928,25809
$ pantools find_genes_by_annotation -dp tomato_DB --name PF00005,GO:0000160,IPR000683,
↪TIGR02499
```

Output files

Output files are stored in `/database_directory/find_genes/by_annotation/`. For each function (node) that was included, a nucleotide and protein and .FASTA file is created with sequences from the genes that are connected to the node.

- **find_genes_by_annotation.log**, relevant information about the extracted genes: node identifier, gene location, homology group etc..

Find genes in region

Find genes of interest in the pangenome that can be (partially) found within a given region (partially). For each found gene, relevant information, the nucleotide sequence and protein sequence is extracted.

Required arguments

`--database-path/-dp` Path to the database.

`--regions-file/-rf` A text file containing genome locations with on each line: a genome number, sequence number, begin and end position, separated by a space.

Optional arguments

`--mode partial` Also retrieve genes that only partially overlap the input regions.

Example input file

Each line must have a genome number, sequence number, begin and end positions that are separated by a space.

```
195 1 477722 478426
71 10 17346 18056
138 47 159593 160300
```

Example command

```
$ pantools find_genes_in_region -dp tomato_DB -rf regions.txt
$ pantools find_genes_in_region -dp tomato_DB -rf regions.txt --mode partial
```

Output files

Output files are stored in `/database_directory/find_genes/in_region/`. For each region that was included, a nucleotide and protein and .FASTA file is created with sequences from the genes that are found within the region.

- **find_genes_in_region.log**, relevant information about the extracted genes: node identifier, gene location, homology group etc..

Functional annotations

The following functions can only be used when any type of functional annotation is [added to the database](#).

Show GO

For a selection of 'GO' nodes, retrieves connected 'mRNA' nodes, child and all parent GO terms that are higher in the GO hierarchy. This function follows the 'is_a' relationships of GO each node to their parent GO term until the 'biological process', 'molecular function' or 'cellular location' node is reached. This can be useful in case InterProScan annotations were included, as these only add the most specific GO terms of the hierarchy to a sequence.

Required arguments

`--database-path/-dp` Path to the database

Requires either **one** of the following arguments

`--node` One or multiple identifiers of 'GO' nodes, separated by a comma.

`--name` One or multiple GO term identifiers, separated by a comma.

Example commands

```
$ pantools show_go -dp tomato_DB --node 15078,15079
$ pantools show_go -dp tomato_DB --name GO:0000001,GO:0000002,GO:0008982
```

Output file

- **show_go.txt**, information of the selected GO node(s): the connected ‘mRNA’ nodes, the GO layer below, and all layers above.

Compare GO

Check if and how similar two given GO terms are. For both nodes, follows the ‘is_a’ relationships up to their parent GO terms until the ‘biological process’, ‘molecular function’ or ‘cellular location’ node is reached. After all parent terms are found, the shared GO terms and their location in the hierarchy is reported.

Required arguments

--database-path/-dp Path to the database.

Requires either **one** of the following arguments

--node Two node identifiers of ‘GO’ nodes, separated by a comma.

--name Two GO identifiers, separated by a comma.

Example command

```
$ pantools compare_go -dp tomato_DB --name GO:0032775,GO:0006313
$ pantools compare_go -dp tomato_DB --node 741487,741488
```

Output file

Output files are stored in *database_directory/function/*

- **compare_go.txt**, information of the two GO nodes: the connected ‘mRNA’ nodes, the GO layer below, all layers above and the shared GO terms between the two nodes.

Homology group information

Report all available information of one or multiple homology groups.

Required arguments

- database-path/-dp Path to the database.
- homology-groups/-hm A text file with homology group node identifiers, separated by a comma

Optional arguments

- label Name of function identifiers from GO, PFAM, InterPro or TIGRAM. To find Phobius (P) or SignalP (S) annotations, include: 'secreted' (P/S), 'receptor' (P/S), or 'transmembrane' (P).
- name One or multiple gene names, separated by a comma.
- skip/-sk Exclude a selection of genomes.
- reference/-ref Only include a selection of genomes.

Example command

```
$ pantools group_info -dp yeast_DB -hm core_groups.txt
$ pantools group_info -dp yeast_DB -hm core_groups.txt --label GO:0032775,GO:0006313 --
  ↪ name budC,estP
```

Output files

Output files are stored in *database_directory/alignments/grouping_v?/groups/*. For each homology group that was included, a nucleotide and protein and .FASTA file is created with sequences found in all genomes.

- **group_info.txt**, relevant information for each homology group: number of copies per genome, gene names, mRNA node identifiers, functions, protein sequence lengths, etc..
- **group_functions.txt**, full description of the functions found in homology groups

When function identifiers are included via --label

- **groups_with_function.txt**, homology group node identifiers from groups that match one of the input functions.

When gene names are included via --name

- **groups_with_name.txt**, homology group node identifiers from groups that match one of the input gene names.

Sequence alignments

The manual for PanTools' sequence alignment functionalities moved to a standalone page - [Sequence alignments](#)

Matrix files

Several functions generate tables in a CSV file format. as tables that the following functions can work with. For example, ANI scores, *k*-mer and gene distance used for constructing the Neighbour Joining [phylogenetic trees](#), and the identity and protein sequence similarity tables created by the [alignment functions](#).

Order matrix

Transforms the CSV table to easy to read file by ordering the values in ascending order from low to high or descending order when `--mode desc` is included in the command. If phenotype information is included in the header, a separate file with the range of found values is created for each phenotype. If this information is not present (only genome numbers in the header), use `rename_matrix` to change the headers.

Required argument

`--database-path/-dp` Path to the database.
`--input-file/-af` A CSV formatted matrix file.

Optional argument

`--skip/-sk` Skip over the values of a selection of genomes.
`--reference/-ref` Only include the values from a selection of genomes.
`--mode asc` or `--mode desc` Order the matrix in ascending or descending order (ascending is default).

Example command

```
$ pantools order_matrix -dp bacteria_DB -if bacteria_DB/ANI/fastANI/ANI_distance_matrix.  
↪ csv  
$ pantools order_matrix -dp bacteria_DB -if bacteria_DB/ANI/fastANI/ANI_distance_matrix.  
↪ csv --mode desc
```

Output file

Output is written to the same directory as the selected input file

- `'old file name' + '_ORDERED'`, ordered values of the original matrix file.

When phenotype information is present in the header

- `'old file name' + '_PHENOTYPE'`, range of values per phenotype.

Rename matrix

Rename the headers (first row and leftmost column) of CSV formatted matrix files. If no `--phenotype` is included, headers are changed to only contain genome numbers.

Required arguments

--database-path/-dp Path to the database.
 --input-file/-af a matrix file with numerical values.

Optional arguments

--phenotype/-ph A phenotype name, used to include phenotype information into the headers.
 --skip/sk Exclude a selection of genomes from the new matrix file. --reference/-ref Only include a selection of genomes in the new matrix file.
 --mode no-numbers Exclude genome numbers from the headers.

Example command

```
$ pantools rename_matrix -dp pecto_DB -phenotype species -if pecto_DB/ANI/fastANI/ANI_
↪ distance_matrix.csv
```

Output file

Output is written to the same directory as the selected input file.

- '*old file name*' + '*_RENAMED*', the original matrix file with changed headers.

Retrieve regions, genomes or features

The two following functions allow users to retrieve genomic regions from the pangenome.

Retrieve regions

Retrieve the full genome sequence or genomic regions from the pangenome.

Required arguments

--database-path/-dp Path to the database.
 --regions-file/-rf A text file containing genome locations with on each line: a genome number, sequence number, begin and end positions separated by a space.

Example command

```
$ pantools retrieve_regions -dp pecto_DB -rf regions.txt
```

Example input

To extract:

- Complete genome - Include a genome number
- An entire sequence - Include a genome number with sequence number
- A genomic region - Include a genome number, sequence number, begin and end positions that are separated by a space. Place a minus symbol behind the regions to extract the reverse complement sequence of the region.

```
1
1 1
1 1 1 10000
1 1 1000 1500 -
195 1 477722 478426
71 10 17346 18056 -
138 47 159593 160300 -
```

Output file

A single FASTA file is created for all given locations and is stored in the database directory.

Retrieve features

To retrieve the sequence of annotated features from the pangenome.

Required arguments

--database-path/-dp Path to the database.

--feature-type or -ft The feature name; for example 'gene', 'mRNA', 'exon', 'tRNA', etc.

Optional arguments

Use one of the following arguments to limit the sequence retrieval to a selection of genomes.

--skip/-sk Exclude a selection of genomes.

--reference/-ref Only include a selection of genomes.

Example command

```
$ pantools retrieve_features -dp pecto_DB --feature-type gene
$ pantools retrieve_features -dp pecto_DB --ft mRNA
```

Output files

For each genome a FASTA file containing the retrieved features will be stored in the database directory. For example, genes.1.fasta contains all the genes annotated in genome 1.

[Next](#) [Previous](#)

[« Previous](#) [Next »](#)

Lumache has its documentation hosted on [Read the Docs](#).