Pangloss Manual

Charley McCarthy

August 5, 2019

Contents

1 Citation

The corresponding paper for Pangloss is published as: Charley G. P. McCarthy & David A. Fitzpatrick (2019). "Pangloss: a tool for pan-genome analysis of microbial eukaryotes." *Genes*, 10(7):521. Link: https://doi.org/10.3390/genes10070521

It may also be worth your while reading our other paper on pangenome analysis of eukaryotes, which used a beta version of Pangloss goes into a bit more detail on the methodology and the pipeline itself: McCarthy C. G. P. and Fitzpatrick D. A. (2019). "Pan-genome analyses of model Fungal species." *Microbial Genomics*, 5(2). Link: https://doi.org/10.1099/mgen.0.000243

Finally, as Pangloss uses PanOCT to perform pangenome analysis it's probably best to cite its corresponding paper also: Fouts D. E. et al. (2012). "PanOCT: automated clustering of orthologs using conserved gene neighborhood for pan-genomic analysis of bacterial strains and closely related species." *Nucleic Acids Res.*, 40(22):e172. https://doi.org/10.1093/nar/gks757

2 Availability

Download from https://github.com/chmccarthy/Pangloss.

3 Authors

Charley McCarthy wrote this document.

4 Overview

Pangloss is a Python/R/Perl pipeline for pangenomic analysis of microbial eukaryotes. Pangloss consists of three major analytic components:

- A gene and gene location prediction pipeline using Exonerate (optional), GeneMark-ES and TransDecoder.
- Construction of a syntenic pangenome by clustering together syntenically-conserved orthologs using the Perl software PanOCT, followed by an optional refinement of that pangenome using reciprocal homology between clusters of syntenic orthologs.
- Functional annotation and visualization of PanOCT-derived pangenome data.

Background information for Pangloss can be found in the *Genes* paper, and it's worth consulting the *Microbial Genomics* paper and the PanOCT paper linked in the Citation section above. Detailed information on the processes of each of these three components can be found in the **Usage** section below.

5 Usage

You can obtain Pangloss by running the command (provided you've git installed) `git clone https://github.com/chmccarthy/Pangloss` or by downloading a zip file from GitHub at https://github.com/chmccarthy/Pangloss. Pangloss is located in the `Pangloss/` folder upon extraction and is run from the command-line as `python $PANGLOSS_INSTALL_PATH/Pangloss.py` followed by a series of arguments, e.g. going into the `Pangloss/` folder just typing `Pangloss.py` will give you:

```
usage:  Pangloss.py [-h] (--pred | --pred_only | --no_pred) [--no_exonerate]
[--qc] [--busco] [--no_blast] [--no_panoct] [--refine] [--ips] [--go]
[--yn00] [--plots] [--karyo] [--size] [--upset] CONFIG_FILE
```

The sections below will give you a brief walkthrough of how to use Pangloss and what the individual command-line arguments actually do.

5.1 How to run Pangloss

5.1.1 The config file (required)

When you download Pangloss you might notice a file called `config.ini` in the folder where `Pangloss.py` is located. Each Pangloss run requires a config file

to tell Pangloss where all the different software it requires is located, what parameters to set for certain software dependencies and where the input data is located. By default Pangloss will use the `config.ini` file in the `Pangloss/` folder, so it's up to you whether you want to either edit that file or specify a new config file on the command-line per run. Let's take a look at the first few lines of the default config file:

```
Paths for gene prediction dependencies: Exonerate, GeneMark-ES and
TransDecoder. Can be full or relative paths, so long as your $PATH
variable is set up appropriately.
[Gene_prediction_dependencies]
exonerate_path = exonerate
genemark_path = /home/charley/bin/gm_et_linux_64/gmes_petap.pl
transdecoder_predict_path = TransDecoder.Predict
transdecoder_longorfs_path = TransDecoder.LongOrfs
```

This section of the file provides Pangloss with the locations of the three dependencies for gene model prediction: Exonerate, GeneMark-ES and the two component scripts of TransDecoder. **Notice that the path for GeneMark-ES is absolute while the paths for Exonerate and TransDecoder are relative**. In this instance, GeneMark-ES is installed in my `/bin` folder on my server while Exonerate and TransDecoder are universally installed (or are specified in my `$PATH` from `/.profile` &c). For every dependency and for input data and output data, Pangloss supports either absolute or relative paths **so long as it matches how your `$PATH` is set up**. If you're unsure how your `$PATH` is set up, specify everything in terms of its absolute path.

5.1.2 Input data

I'm going to run Pangloss in a working folder I've made called `Test_Dataset/`. In `Test_Dataset/`, I've included the following files and folders (* indicates input for optional analyses):

```
genomes/ *go.obo *goslim_generic.obo
*saccharomycetales_odb9/ testconfig.ini
```

`testconfig.ini` will be our config file for this Pangloss run, and contains the paths of all the dependencies for Pangloss as well as the input data for this run. In that config file I've specified a folder `genomes/` which contains input data for gene model prediction and quality control in Pangloss. Inside `genomes/` I have five files (* indicates input for optional analyses):

```
*dubious.faa genomes.txt *reference.faa Strain_A.fna Strain_B.fna
```

The two input genomes for this Pangloss run are called `Strain_A.fna` and `Strain_B.fna` - these are two genome assemblies I've downloaded from NCBI

and simply renamed according to my own naming scheme. `genomes.txt` gives the paths to these two assemblies line-by-line relative to the working folder starting with the reference assembly for my species of interest (in this case, Strain A):

```
genomes/Strain_A.fna
genomes/Strain_B.fna
```

The other two files are FASTA-formatted protein sequence files. For optional gene model prediction using Exonerate, a "reference" protein sequence set is required. This can usually be obtained for your species of interest from places like NCBI, ENSEMBL, &c. In this case, I've called the protein set `reference.faa` and put it in the same folder as the two genome assemblies. Another protein sequence file, `dubious.faa` contains known protein sequences from a species that are actually "pseudogenes" or sequences which are non-functional despite being predicted as functional by genome annotation software. This file is used to filter out potential pseudogenes from a predicted protein sequence dataset for each input genome (see command-line arguments section below).

Moving back to the `Test_Dataset/` working folder you'll notice one other folder and two other files. `saccharomycetales_odb9/` is a BUSCO lineage dataset (see https://busco.ezlab.org/ for more information) used to assess predicted protein sequence set completeness. As with `genomes/`, the location of this folder is provided to Pangloss from the config file, **but the folder's location must be provided relative to your working directory or BUSCO will refuse to run analysis**. The simplest way to avoid this is just to have the BUSCO lineage dataset in the same working directory as your input data.

The two .obo files are for slimmed Gene Ontology or (GO-slim) enrichment analysis of a pangenome dataset using GOATools (see sections on GOATools below). go.obo is the general Gene Ontology available from the GO website (see http://geneontology.org/docs/download-ontology/ for more information), and goslim_generic.obo is a slimmed subset of the overall GO data intended to give a broader overview of functions in an organism (see http://geneontology.org/docs/go-subset-guide/ for more information). Different GO-slim subsets exist for different model organisms (e.g. yeast, fruit fly) but as my species of interest in not a model organism I can just use the generic subset.

5.1.3 Running Pangloss

Let's assume I have Pangloss available in my `$PATH`. In `Test_Dataset/`, I run Pangloss as such:

```
Pangloss.py --pred --busco --qc --refine --ips --go --yn00
--plots testconfig.ini
```

In this case,

1. I'm telling Pangloss I want to run full gene model and gene location prediction for each genome in my dataset (`--pred`) with filtering against my pseudogene dataset `genomes/dubious.faa` (`--qc`) and BUSCO completeness analysis of each predicted protein set (`--busco`). Gene model and gene location prediction datasets are written to the folder `gm_pred/` by defailt, which contains individual and concatenated nucleotide sequence, amino acid sequence and genomic location files for all input genomes in `gm_pred/sets/` and temporary files for the various prediction methods in `gm_pred/gmes/`, `gm_pred/td/` and `gm_pred/ref/`. BUSCO output is written to the separate folder `busco/`.

2. Pangloss will then perform an all-vs.-all BLASTp search for the entire pangenome protein sequence dataset (named `gm_pred/sets/allprot.db` by default) with a liberal e-value cutoff of 10^{-4}. I've attempted to parallize the all-vs.-all BLASTp search as much as possible in Pangloss without constantly writing separate output files, but it may be easier in some cases to perform gene prediction only in Pangloss, perform the all-vs.-all BLASTp search yourself and then pangenome construction and downstream analysis using Pangloss again (see below sections for the appropriate arguments to carry this out).

3. Once the BLASTp searches are done Pangloss will then perform pangenome construction using PanOCT. I've then also told Pangloss to also attempt to "refine" the constructed pan-genome for microsynteny loss due to assembly artefacts (`--refine`). Pangloss will write the various outputs of PanOCT (with and/or without further refinement) to the directory `panoct/` and within that directory generate nucleotide and amino acid sequence files for every cluster in both the refined and/or unrefined results in the directory `panoct/clusters/`.

4. Pangloss will then perform try to annotate the full pangenome dataset with Pfam, InterPro and Gene Ontology information using InterProScan (on Linux only, see sections below). Following that it will perform GO-slim enrichment analysis on the core and accessory genome datasets using GOATools, which is set to use a Fisher's exact test with 500 resampled p-values (see sections below on GOATools).

5. Pangloss will also try to run selection analysis on all refined or initial pangenome clusters using yn00. It does this by translating each cluster nucleotide file into its amino acid equivalent, aligning that amino acid set using MUSCLE and then untranslating that amino acid alignment into nucleotide format and running yn00 with the untranslated nucleotide alignment as input. Why the trouble of switching between nucleotides and amino acids? Nucleotide alignments generally don't respect the codon rule and tend to introduce gaps where it doesn't make evolutionary sense to do so. Yang & Nielsen selection analysis output from yn00 for each cluster

will be summarized in a file called `panoct/clusters/yn00summary.txt`. There may be some instances where yn00 can't run for certain alignment due to weird frameshifts or if the cluster is a singleton - these are usually given in summary file with a cluster size of None or 1. The translation/alignment/untranslation paradigm is essentially a reimplementation of PutGaps (see http://mcinerneylab.com/software/putgaps/).

6. Finally, Pangloss will generate a number of plots which visualize different features of the pangenomic data we've just generated. This will include a bar chart giving the occurrences of clusters of size 1 to n (n = total number of input genomes) and both the observed and total estimated numbers of clusters within the species pangenome, a ring chart of the proportions of core and accessory clusters in the pangenome dataset, an UpSet plot of the distribution of accessory genes across the various input genomes in the pangenome dataset and two karyotype plots for each input genome in the dataset; one labelled with the position of core and accessory genes and one labelled with the positions of genes coloured by the number of syntenic orthologs they have. The UpSet plot and bar chart plots will be written to the working folder, while karyotype plots will be written to a subfolder called `karyoplots/`.

More information (and exciting workflow diagrams) can be found in the Genes papers linked in Citations above.

5.2 Example commands

5.2.1 Run everything

```
Pangloss.py --pred --busco --qc --refine --ips --go --yn00
--plots testconfig.ini
```

5.2.2 Run gene prediction and QC steps only

```
Pangloss.py --pred_only --busco --qc testconfig.ini
```

5.2.3 Run PanOCT analysis with pre-generated data

```
Pangloss.py --no_pred --no_blast testconfig.ini
```

5.2.4 Run functional annotation and some visualization analyses on pre-generated PanOCT output

```
Pangloss.py --no_pred --no_blast --no_panoct --ips --go --upset --karyo
testconfig.ini
```

5.3 Command-line arguments for gene prediction

5.3.1 `--pred`, `--pred_only`, `--no_pred` (one required)

These three arguments control how gene prediction analysis is carried out within the overal Pangloss pipeline. One of these arguments is required for Pangloss to run, and each argument is mutually exclusive of the other two. `--pred` runs gene prediction as part of the overall pipeline, `--pred_only` runs only the gene prediction part of the pipeline (as well as downstream analyses if `--qc` and `--busco` are enabled) and quits once all predictions are complete, and `--no_pred` does not run gene prediction. The latter argument is useful when gene sequence and gene location data is already available (either from previous Pangloss runs or from other sources).

5.3.2 `--no_exonerate`

This argument disables Exonerate-based gene prediction, which speeds up the overall gene prediction process but means you may miss potential gene models not detected by either GeneMark-ES or TransDecoder.

5.3.3 `--qc`

This argument enables a custom "quality control" assessment of gene prediction for a dataset by searching known "dubious" genes or pseudogenes against your dataset using BLASTp, and removing genes that have $\geq$70% similarity to known dubious genes. Works best for model organisms with knowns sets of dubious genes usually available from a genomic resource website, like that from the Saccharomyces Genome Database for *Saccharomyces cerevisiae*.

5.3.4 `--busco`

This argument enables BUSCO completeness analysis of predicted gene sets for each genome in an input dataset. Installation instructions for BUSCO are available at https://gitlab.com/ezlab/busco. For completeness analysis of protein sequence data HMMER must also be installed (available from http://hmmer.org/). **Note**: you need to specify a separate config.ini file for BUSCO analysis (generally located in `$BUSCO_INSTALL_PATH/scripts/../config/`. You need to change the location of HMMsearch to where you have installed the HMMER suite (e.g. `/usr/local/bin`) in that file.

5.4 Command-line arguments for pangenome construction

5.4.1 `--no_blast`

This argument disables the all-vs.-all BLASTp search that Pangloss by default performs for the entire pangenome dataset. This is useful when BLASTp output has already been generated for a pangenome dataset (and in most cases, all-vs.-all BLASTp data is quicker to generate yourself using a HPC environment than

it is in Pangloss). Just make sure that the name and location of your BLASTp output matches what's in the configuration file.

5.4.2 `--no_panoct`

This argument disables pangenome construction from a given dataset using PanOCT. This argument is here for debugging purposes, but may be useful in cases where PanOCT data has already been produced in a previous run.

5.4.3 `--refine`

This argument enables in-house "refinement" of a pangenome constructed by PanOCT. This is based on reciprocal strain top-hit homology between all member genes of two accessory syntenic clusters. See the papers in Citation for more information.

5.5 Command-line arguments for characterization

5.5.1 `--ips`

This argument enables InterProScan analysis of a pangenome dataset by Pangloss, and will run Pfam, InterPro and Gene Ontology annotation analysis. This analysis requires InterProScan to be installed and available in your `PATH`. **Note:** InterProScan can only run on Linux distributions, due to its use of third-party binaries (see https://github.com/ebi-pf-team/interproscan/wiki for more information). Pangloss will skip over InterProScan analysis if `--ips` is passed on non-Linux operating systems.

5.5.2 `--go`

This argument enables GO-slim enrichment analysis of core and accessory genomes using GOATools. Enrichment analysis in Pangloss (*via* GOATools) uses Fisher's exact test (FET) with parent term propagation and false discovery rate correction using 500 sampled p-values ($p \geq 0.05$).

5.5.3 `--yn00`

This argument enables Yang & Nielsen (2000) selection analysis using yn00, which is part of the PAML package of phylogenetic tools. Pangloss will run yn00 on each syntenic cluster in a pangenome, and generate a summary of the number of pairwise alignments in each cluster (if any) which exhibit traits of positive selection, i.e. their d_N/d_S ratio is ≥ 1.

5.6 Command-line arguments for data visualization

5.6.1 `--plots`

This argument enables all visualization analyses described below, and is equivalent to passing `--karyo --size --upset` to Pangloss.

5.6.2 `--karyo`

This argument enables karyotype plot generation (by `Karyotype.R`) of core and accessory genome content along all genomic sequences (contigs, chromosomes, &c.) within each genome in a pangenome dataset. `--karyo` produces two karyotype plots: one in which gene locations are coloured by their parent component (core: green, accessory: red), and one in which the same locations are coloured by the number of genes in their parent syntenic cluster (red-to-green gradient with red representing singleton genes and green representing core genes). This analysis requires the R packages `KaryoploteR`, `Hmisc` and `regioneR`. **Note:** Although karyotype plots are generated for each genome in a dataset, plots for genomes that are assembled to chromosome-level or are otherwise highly continguous are generally the most informative/aesthetically pleasing. For more information, see http://bioconductor.org/packages/release/bioc/html/karyoploteR.html.

5.6.3 `--size`

This argument enables the generation of two simple size plots: a ring chart of the proportions of core and accessory syntenic clusters in a pangenome dataset (performed by `RingChart.R`) and a bar chart of the syntenic cluster size distribution within the same dataset (performed by `BarChart.R`). The latter also estimates the "true" number of syntenic clusters (i.e. the "true" pangenome size) within a dataset using the Chao lower bound method. The Chao estimate for pangenome size $\hat{N}$ is given by the equation

$$\hat{N} = N + \frac{y_1^2}{2y_2}$$

where N is the number of syntenic clusters in a pangenome, y_1 is the number of singleton clusters and y_2 is the number of doubleton (2-member) clusters. `RingChart.R` requires the R packages `ggplot2` and `ggrepel`. `BarChart.R` also requires `ggplot2`. The implementation of the Chao estimation method in `BarChart.R` is based on a previous implementation in the R package `micropan`.

5.6.4 `--upset`

This argument enables the generation of an UpSet plot representing the distribution of syntenic orthologs within the accessory genome component of a pangenome dataset. This is performed by `UpSet.R`, which requires the R packages `UpSetR` and on macOS `Cairo`. An UpSet plot is a way of visualizing

the intersections between sets as an alternative to Venn or Euler diagrams, which are generally limited to a certain amount of input sets. For more information on UpSet plots, see this pretty thorough explanation of the concept: https://caleydo.org/tools/upset/.

6 Requirements

Pangloss has been tested on macOS 10.13, Ubuntu 18.04.2 LTS and CentOS 7. Pangloss requires the latest versions of Python 2 (≥2.7.10), R (≥3.5.2) and Perl 5. Individual requirements for compontent analyses are detailed below:

Dependency	**Purpose**	**Download**
Biopython	FASTA processing, etc.	https://biopython.org/
Exonerate	Gene prediction using translated reference homologs.	https://github.com/nathanweeks/exonerate
GeneMark-ES	Gene prediction using Hidden Markov Models.	http://topaz.gatech.edu/GeneMark/license_download.cgi
TransDecoder	ORF detection in non-coding regions using position-weight matrices.	https://github.com/TransDecoder/TransDecoder
BLAST+	Sequence similarity search.	https://blast.ncbi.nlm.nih.gov/Blast.cgi?PAGE_TYPE=BlastDocs&DOC_TYPE=Download
BUSCO	Gene model set completeness analysis.	https://gitlab.com/ezlab/busco
yn00	Pairwise selection analysis of syntenic clusters.	http://abacus.gene.ucl.ac.uk/software/paml.html#download
MUSCLE	Alignment of genes in syntenic clusters.	https://www.drive5.com/muscle/downloads.htm
InterProScan	Pfam, InterPro, GO functional annotation.	https://github.com/ebi-pf-team/interproscan/wiki/HowToDownload
GOATools	GO-slim enrichment analysis.	https://github.com/tanghaibao/goatools
ggplot, ggrepel, UpSetR, KaryoploteR	Gene model set completeness analysis.	See Section 6.11

7 Installation

Pangloss is available as executable code from https://github.com/chmccarthy/Pangloss and PanOCT is included in the repository. Links to installation instructions and other useful info for the various dependencies of Pangloss (assuming Python, R and Perl are installed) are given below.

7.1 Biopython

Tested version: 1.73

Installation instructions for Biopython are available from https://biopython.org/wiki/Download. For most Linux and macOS environments, Biopython can be installed via `pip`, e.g. `pip install biopython`. Pangloss requires Biopython 1.73 (released December 2018) or later, as previous versions contain bugs in the relevant packages for handling data from Exonerate. Biopython is imported within Pangloss (e.g. `from Bio import SeqIO`, etc.).

7.2 Exonerate

Tested version: 2.4

Exonerate is no longer officially supported (it appears), but a continuation of Exonerate is hosted at https://github.com/nathanweeks/exonerate. Installation instructions from source are provided at the same address. Exonerate can also be installed from `apt-get` on most Linux distributions or through Homebrew on macOS via `brew install brewsci/bio/exonerate`. Exonerate should be available in your `PATH` as `exonerate` or specified in your config file.

7.3 GeneMark-ES

Tested version: 4.3.8

macOS and Linux versions of GeneMark-ES executables (and the licence keys necessary to run GeneMark-ES) are available at http://topaz.gatech.edu/GeneMark/license_download.cgi. See `INSTALL` file in GeneMark-ES folder for instructions on how to "install" licence key. GeneMark-ES requires the `YAML`, `Hash::Merge`, `Logger::Simple` and `Parallel::ForkManager` Perl modules which are all available via `cpanm`. GeneMark-ES should be available in your `PATH` as `gmes_petap.pl` or specified in your config file. Pangloss uses a modified version of `get_sequence_from_GTF.pl` due to inconsistencies in how the version provided with GeneMark-ES parses GFF files.

7.4 TransDecoder

Tested version: 5.5

TransDecoder is available as executable code from https://github.com/TransDecoder/TransDecoder. Both executable programs `TransDecoder.LongOrfs` and `TransDecoder.Predict` should be in your `PATH` or specified in your config file.

7.5 BLAST+

Tested version: 2.9.0

BLAST+ installation instructions are available from the NCBI at https://www.ncbi.nlm.nih.gov/books/NBK52640/. `blastp` should be available in your `PATH`.

7.6 BUSCO

Tested version: 3.1

Installation instructions for BUSCO are available at https://gitlab.com/ezlab/busco. For completedness analysis of protein sequence data HMMER must also be installed (available from http://hmmer.org/). Note that you need to specify a separate config.ini file for BUSCO analysis (generally located in `$BUSCO_INSTALL_PATH/scripts/../config/`) and you need to change the location of `HMMsearch` to where you have installed the HMMER suite (e.g. `/usr/local/bin`) in that file. `run_BUSCO.py` must be in your `PATH` or otherwise specified in the config file for Pangloss.

7.7 MUSCLE

Tested version: 3.8.31

MUSCLE binaries can be found at https://www.drive5.com/muscle/downloads.htm. MUSCLE should be in your `PATH` as `muscle` or as specified in your config file.

7.8 yn00

Tested version: 4.8 (PAML)

yn00 is part of the PAML package. PAML installation instructions are available from http://abacus.gene.ucl.ac.uk/software/paml.html#download - scroll down to the section entitled "UNIX/Linux and Mac OSX". yn00 should be in your `PATH` as `yn00` or otherwise specified in your config file.

7.9 InterProScan

Tested version: 5.34

Installation instructions for InterProScan are available at https://github.com/ebi-pf-team/interproscan/wiki/HowToDownload. `interproscan.sh` should be in your `PATH`. InterProScan can only run on Linux distributions, due to its use of third-party binaries.

7.10 GOATools

Tested version: 0.8.12

See https://pypi.org/project/goatools/ for installation instructions, this in turn should make `map_to_slim.py` and `find_enrichment.py` available in your `PATH`. FET analysis in GOAtools uses either the `fisher` or `Scipy.stats.fisher` Python modules - generally the former is quicker. Both should be available via `pip`.

7.11 ggplot, ggrepel, UpSetR, KaryoploteR

Tested versions: 3.2, 0.81, 1.4, 1.10.3 respectively

ggplot, ggrepel and UpSetR can all be installed from `install.packages` within R. UpSetR plots are rendered using Cairo, which is available via `install.packages`, although plot visualization through Cairo is currently disabled for Linux operating systems. KaryoploteR is a Bioconductor package. Instructions for Bioconductor installation are available from https://bioconductor.org/install/ and specific instructions for installing KaryoploteR are available from https://bioconductor.org/packages/release/bioc/html/karyoploteR.html.