PopPUNK Documentation

Release 1.1.3

John Lees and Nicholas Croucher

Dec 18, 2020

Contents:

1 Installation **3**
- 1.1 Installing with conda (recommended) . . . 3
- 1.2 Installing with pip . . . 3
- 1.3 Clone the code . . . 4
- 1.4 Dependencies . . . 4

2 Options **5**

3 Quickstart **9**
- 3.1 Running PopPUNK . . . 9
- 3.2 Re-fitting the model . . . 13

4 Tutorial **17**
- 4.1 Creating a database . . . 18
- 4.2 Fitting the model . . . 20
- 4.3 Refining a model . . . 27
- 4.4 Applying a single distance threshold . . . 32
- 4.5 Assigning queries . . . 32
- 4.6 Creating external visualisations from a fitted model . . . 37
- 4.7 Using a previous model with a new database . . . 38

5 Troubleshooting **39**
- 5.1 Error/warning messages . . . 40
- 5.2 Choosing the right k-mer lengths . . . 41
- 5.3 Viewing the network with cytoscape . . . 42
- 5.4 Setting the perplexity parameter for t-SNE . . . 44
- 5.5 Dealing with poor quality data . . . 46
- 5.6 Removing samples from a database . . . 48
- 5.7 Memory/run-time issues . . . 49

6 Scripts **51**
- 6.1 Writing the pairwise distances to an output file . . . 51
- 6.2 Writing network components to an output file . . . 51
- 6.3 Calculating Rand indices . . . 52
- 6.4 Calculating silhouette indices . . . 52

7 Miscellaneous **53**

7.1 Rejected/alternative logos 54

8 Details **55**

9 Citation: **57**

10 Index: **59**

In straightforward cases, usage can be as simple as:

```
poppunk --easy-run --r-files references.txt --output poppunk_db
```

Where `references.txt` is a list of assembly fasta files, one per line. See *Quickstart* and the *Tutorial* for full details.

CHAPTER 1

Installation

The easiest way to install is through conda, which will also install the dependencies:

```
conda install poppunk
```

Then run with `poppunk`.

Important: PopPUNK requires python3 to run (which on many default Linux installations is run using `python3` rather than `python`).

1.1 Installing with conda (recommended)

If you do not have `conda` you can install it through miniconda and then add the necessary channels:

```
conda config --add channels defaults
conda config --add channels bioconda
conda config --add channels conda-forge
```

Then run:

```
conda install poppunk
```

1.2 Installing with pip

If you do not have conda, you can also install through pip:

```
python3 -m pip install poppunk
```

You will also need mash (v2 or higher) installed (see *Dependencies*).

1.3 Clone the code

You can also clone the github to run the latest version, which is executed by:

```
git clone https://github.com/johnlees/PopPUNK.git && cd PopPUNK
python3 poppunk-runner.py
```

This will also give access to the *Scripts*.

You will need to install the *Dependencies* yourself (you can still use conda or pip for this purpose).

1.4 Dependencies

We tested PopPUNK with the following packages:

- python3 (3.6.3)
- `DendroPy` (4.3.0)
- `hdbscan` (0.8.13)
- `matplotlib` (2.1.2)
- `networkx` (2.1)
- `numpy` (1.14.1)
- `numba` (0.36.2)
- `pandas` (0.22.0)
- `scikit-learn` (0.19.1)
- `scipy` (1.0.0)
- `sharedmem` (0.3.5)

`numba` may need `gcc >=v4.8` to install correctly through pip (if you are getting `OSError` or `'GLIBCXX_3.4.17' not found`).

You will need a mash installation which is v2.0 or higher

Optionally, you can use rapidnj if producing output with `--microreact` and `--rapidnj` options. We used v2.3.2.

CHAPTER 2

Options

Usage:

```
usage: PopPUNK [-h]
               (--easy-run | --create-db | --fit-model | --refine-model | --assign-query
↪| --use-model | --generate-viz)
               [--ref-db REF_DB] [--r-files R_FILES] [--q-files Q_FILES]
               [--distances DISTANCES]
               [--external-clustering EXTERNAL_CLUSTERING] --output OUTPUT
               [--plot-fit PLOT_FIT] [--full-db] [--update-db] [--overwrite]
               [--min-k MIN_K] [--max-k MAX_K] [--k-step K_STEP]
               [--sketch-size SKETCH_SIZE] [--max-a-dist MAX_A_DIST]
               [--ignore-length] [--K K] [--dbscan] [--D D]
               [--min-cluster-prop MIN_CLUSTER_PROP] [--pos-shift POS_SHIFT]
               [--neg-shift NEG_SHIFT] [--manual-start MANUAL_START]
               [--indiv-refine] [--no-local] [--model-dir MODEL_DIR]
               [--previous-clustering PREVIOUS_CLUSTERING] [--core-only]
               [--accessory-only] [--subset SUBSET] [--microreact]
               [--cytoscape] [--phandango] [--grapetree] [--rapidnj RAPIDNJ]
               [--perplexity PERPLEXITY] [--info-csv INFO_CSV] [--mash MASH]
               [--threads THREADS] [--no-stream] [--version]
```

Command line options

optional arguments:

-h, --help	show this help message and exit

Mode of operation:

--easy-run	Create clusters from assemblies with default settings
--create-db	Create pairwise distances database between reference sequences
--fit-model	Fit a mixture model to a reference database
--refine-model	Refine the accuracy of a fitted model

--assign-query Assign the cluster of query sequences without re- running the whole mixture model

--generate-viz Generate files for a visualisation from an existing database

--use-model Apply a fitted model to a reference database to restore database files

Input files:

--ref-db REF_DB Location of built reference database

--r-files R_FILES File listing reference input assemblies

--q-files Q_FILES File listing query input assemblies

--distances DISTANCES Prefix of input pickle of pre-calculated distances

--external-clustering EXTERNAL_CLUSTERING File with cluster definitions or other labels generated with any other method.

Output options:

--output OUTPUT Prefix for output files (required)

--plot-fit PLOT_FIT Create this many plots of some fits relating k-mer to core/accessory distances [default = 0]

--full-db Keep full reference database, not just representatives

--update-db Update reference database with query sequences

--overwrite Overwrite any existing database files

Kmer comparison options:

--min-k MIN_K Minimum kmer length [default = 9]

--max-k MAX_K Maximum kmer length [default = 29]

--k-step K_STEP K-mer step size [default = 4]

--sketch-size SKETCH_SIZE Kmer sketch size [default = 10000]

Quality control options:

--max-a-dist MAX_A_DIST Maximum accessory distance to permit [default = 0.5]

--ignore-length Ignore outliers in terms of assembly length [default = False]

Model fit options:

--K K Maximum number of mixture components [default = 2]

--dbscan Use DBSCAN rather than mixture model

--D D Maximum number of clusters in DBSCAN fitting [default = 100]

--min-cluster-prop MIN_CLUSTER_PROP Minimum proportion of points in a cluster in DBSCAN fitting [default = 0.0001]

Refine model options:

--pos-shift POS_SHIFT Maximum amount to move the boundary away from origin [default = 0.2]

--neg-shift NEG_SHIFT Maximum amount to move the boundary towards the origin [default = 0.4]

--manual-start MANUAL_START A file containing information for a start point. See documentation for help.

--indiv-refine Also run refinement for core and accessory individually

--no-local Do not perform the local optimization step (speed up on very large datasets)

Database querying options:

--model-dir MODEL_DIR Directory containing model to use for assigning queries to clusters [default = reference database directory]

--previous-clustering PREVIOUS_CLUSTERING Directory containing previous cluster definitions and network [default = use that in the directory containing the model]

--core-only Use a core-distance only model for assigning queries [default = False]

--accessory-only Use an accessory-distance only model for assigning queries [default = False]

Further analysis options:

--subset SUBSET File with list of sequences to include in visualisation (with –generate-viz only)

--microreact Generate output files for microreact visualisation

--cytoscape Generate network output files for Cytoscape

--phandango Generate phylogeny and TSV for Phandango visualisation

--grapetree Generate phylogeny and CSV for grapetree visualisation

--rapidnj RAPIDNJ Path to rapidNJ binary to build NJ tree for Microreact

--perplexity PERPLEXITY Perplexity used to calculate t-SNE projection (with –microreact) [default=20.0]

--info-csv INFO_CSV Epidemiological information CSV formatted for microreact (can be used with other outputs)

Other options:

--mash MASH Location of mash executable

--threads THREADS Number of threads to use [default = 1]

--no-stream Use temporary files for mash dist interfacing. Reduce memory use/increase disk use for large datasets

--version show program's version number and exit

CHAPTER 3

Quickstart

This guide briefly explains how PopPUNK can be run on a set of genomes. For a more detailed example see the *Tutorial*.

We will work with 128 *Listeria monocytogenes* genomes from Kremer et al which can be downloaded from figshare.

- *Running PopPUNK*
  - *Check the distance fits*
  - *Check the distance plot*
  - *Check the model fit*
- *Re-fitting the model*
  - *Creating interactive output*

3.1 Running PopPUNK

First download the example set above, then extract the assemblies and create a file with a list of their locations:

```
tar xf listeria_example.tar.bz2
ls *.contigs_velvet.fa > reference_list.txt
```

Now run PopPUNK:

```
poppunk --easy-run --r-files reference_list.txt --output lm_example --threads 4 --
↪plot-fit 5 --min-k 13 --full-db
```

This will:

1. Create a database of mash sketches

2. Use these to calculate core and accessory distances between samples (which are also stored as part of the database).
3. Fit a two-component Gaussian mixture model to these distances to attempt to find within-strain distances.
4. Use this fit to construct a network, from which clusters are defined

where the additional options:

- `--threads 4` increase speed by using more CPUs
- `--min-k 13` ensures the distances are not biased by random matches at lower k-mer lengths.
- `--plot-fit 5` plots five examples of the linear fit, to ensure `--min-k` was set high enough.
- `--full-db` does not remove redundant references at the end, so the model fit can be re-run.

> **Important:** The key step for getting good clusters is to get the right model fit to the distances. The algorithm is robust to most other parameters settings. See *Re-fitting the model* for details.

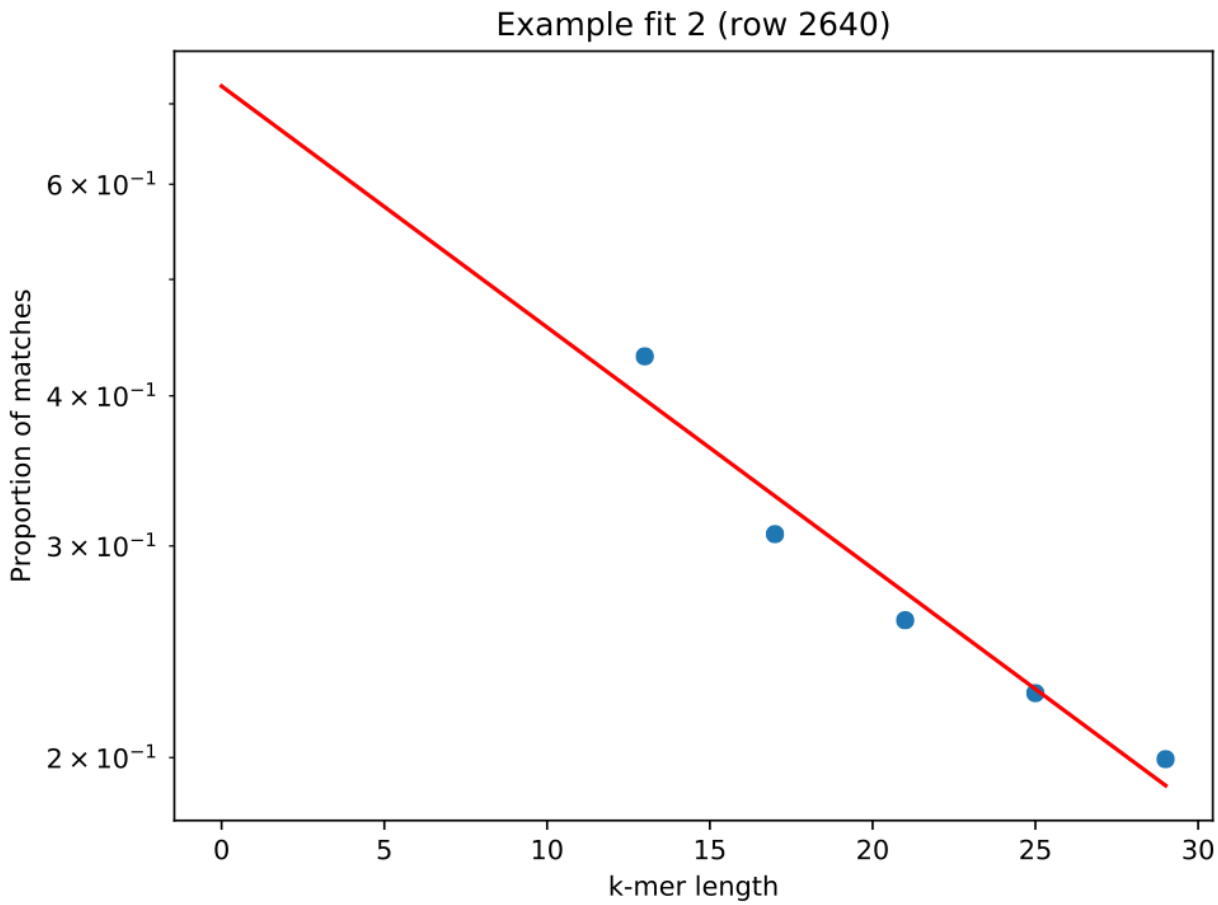
The cluster definitions are output to `lm_example/lm_example_clusters.csv`.

3.1.1 Check the distance fits

The first thing to do is check the relation between mash distances and core and accessory distances are correct:

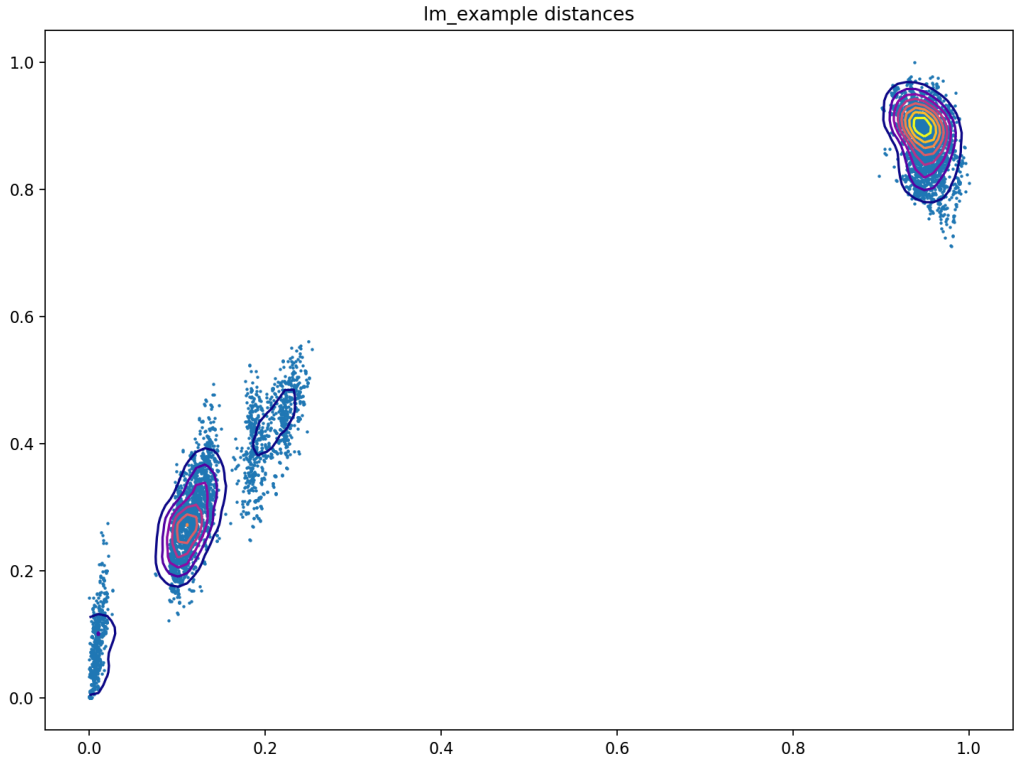
```
Creating mash database for k = 13
Random 13-mer probability: 0.04
Creating mash database for k = 21
Random 21-mer probability: 0.00
Creating mash database for k = 17
Random 17-mer probability: 0.00
Creating mash database for k = 25
Random 25-mer probability: 0.00
Creating mash database for k = 29
Random 29-mer probability: 0.00
```

This shows `--min-k` was set appropriately, as no random probabilities were greater than 0.05. Looking at one of the plots `lm_example/fit_example_1.pdf` shows a straight line fit, with the left most point not significantly above the fitted relationship:



3.1.2 Check the distance plot

A plot of core and accessory distances contains information about population structure, and about the evolution of core and accessory elements. Open `lm_example/lm_example_distanceDistribution.png`:



Each point is the distance between a pair of isolates in the collection. The x-axis shows core distances, the y-axis accessory distances. Lines are contours of density in regions where points overlap, running from blue (low density) to yellow (high density). Usually the highest density will be observed in the top-right most blob, where isolates from different clusters are being compared.

In this sample collection the top-right blob represents comparisons between lineage I and lineage II strains. The blob nearest the origin represents comparisons within the same strain. The other two blobs are comparisons between different strains within the same lineage. Overall there is a correlation between core and accessory divergence, and accessory divergence within a cluster is higher than the core divergence.

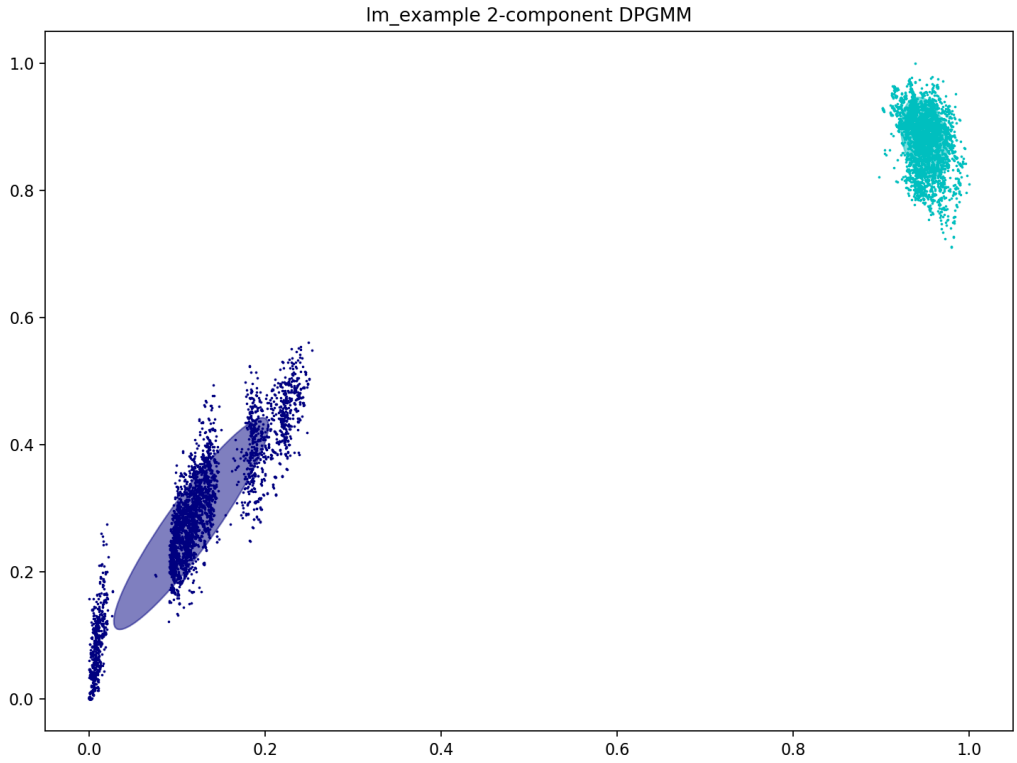
3.1.3 Check the model fit

A summary of the fit and model is output to `STDERR`:

```
Fit summary:
   Avg. entropy of assignment    0.0004
   Number of components used     2
Warning: trying to create very large network
Network summary:
   Components     2
   Density        0.5405
   Transitivity   1.0000
   Score          0.4595
```

This is a bad network score – a value of at least 0.8 would be expected for a good fit. A high density suggests the fit

was not specific enough, and too many points in the core-accessory plot have been included as within strain. Looking at the fit this proves to be true:



As only two components were used, the separate blobs on the plots were not able to be captured. The blob closest to the origin must be separated from the others for a good high-specificity fit. Inclusion of even a small number of points between different clusters rapidly increases cluster size and decreases number of clusters. In this example the initial fit clusters lineage I and lineage II separately, but merges sub-lineages (which we refer to as strains).

PopPUNK offers three ways to achieve this – two are discussed below.

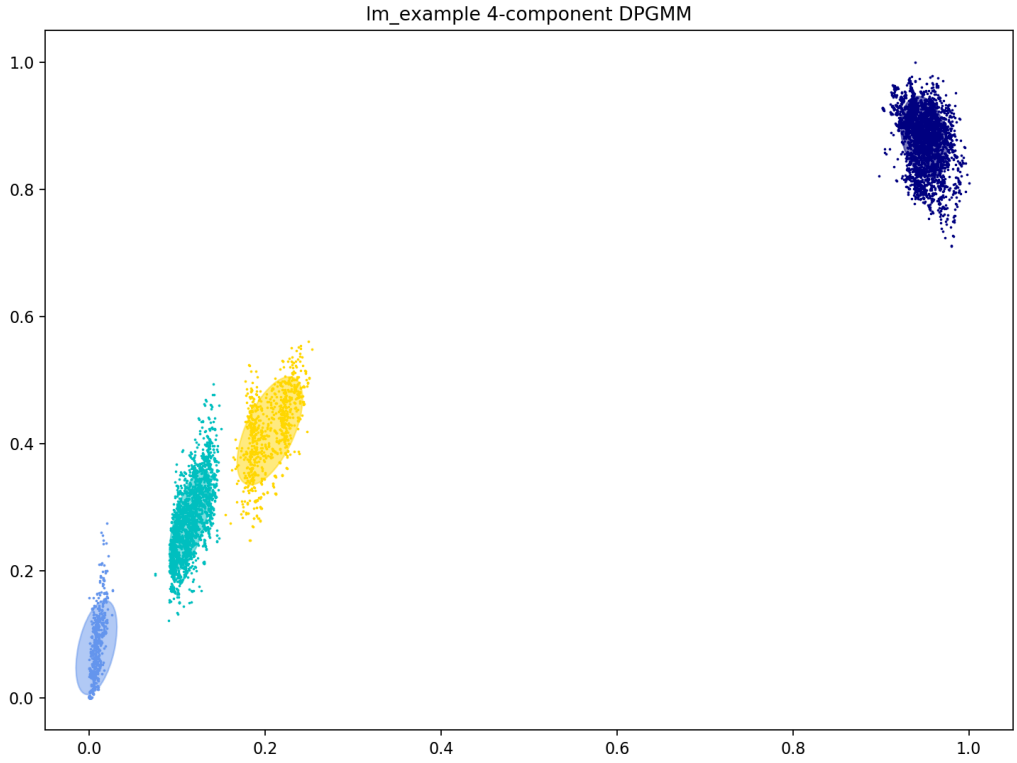
3.2 Re-fitting the model

To achieve a better model fit which finds the strains within the main lineages the blob of points near the origin needs to be separated from the other clusters. One can use the existing database to refit the model with minimal extra computation.

The first way to do this is to increase the number of mixture components to the number of blobs you roughly judge to be in the plot. In this case there are four:

```
poppunk --fit-model --distances lm_example/lm_example.dists --ref-db lm_example --
↪output lm_example --full-db --K 4
```

This correctly separates the blob at the origin – the 'within-strain' distances:



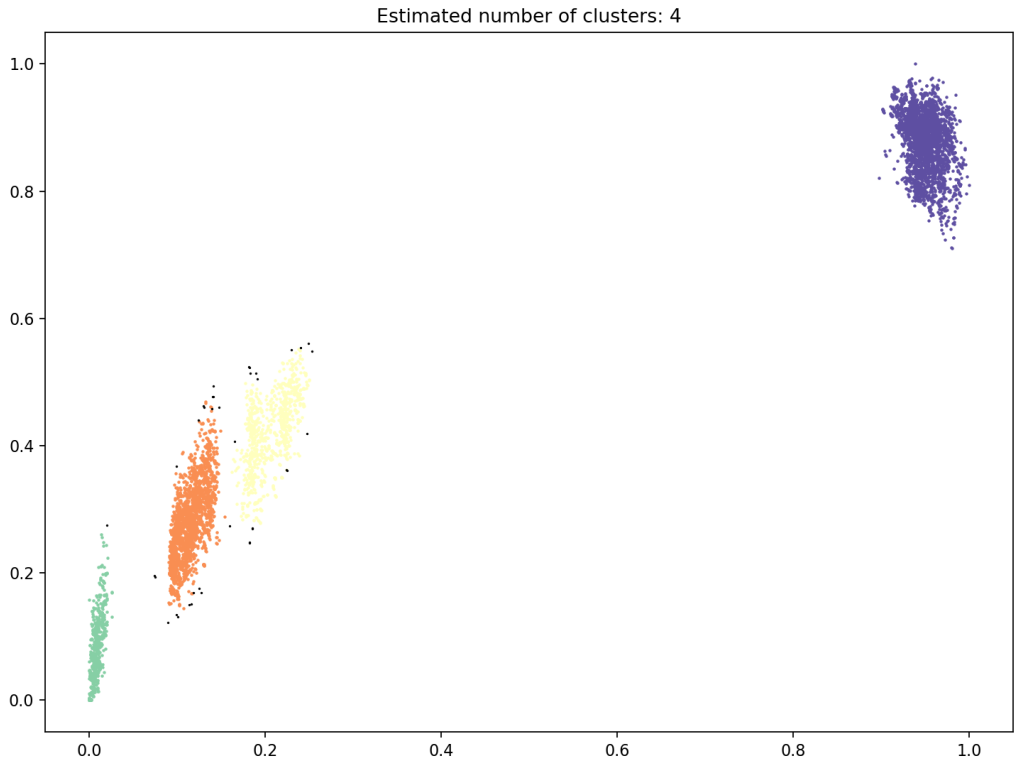
Which gives more clusters (network components) and a lower density, higher scoring network:

```
Fit summary:
   Avg. entropy of assignment    0.0076
   Number of components used     4
Network summary:
   Components      31
   Density         0.0897
   Transitivity    1.0000
   Score           0.9103
```

Alternatively DBSCAN can be used, which doesn't require the number of clusters to be specified:

```
poppunk --fit-model --distances lm_example/lm_example.dists --ref-db lm_example --
↪output lm_example --full-db --dbscan
```

This gives a very similar result:



with an almost identical network producing identical clusters:

```
Fit summary:
   Number of clusters      4
   Number of datapoints    8128
   Number of assignments   8128
Network summary:
   Components      31
   Density         0.0896
   Transitivity    0.9997
   Score           0.9103
```

The slight discrepancy is due to one within-strain point being classified as noise (small, black point on the plot). For datasets with more noise points from DBSCAN then model refinement should always be run after this step (see *Refining a model*).

3.2.1 Creating interactive output

Now that a good, high-specificity fit has been obtained you can add some extra flags to create output files for visualisation:

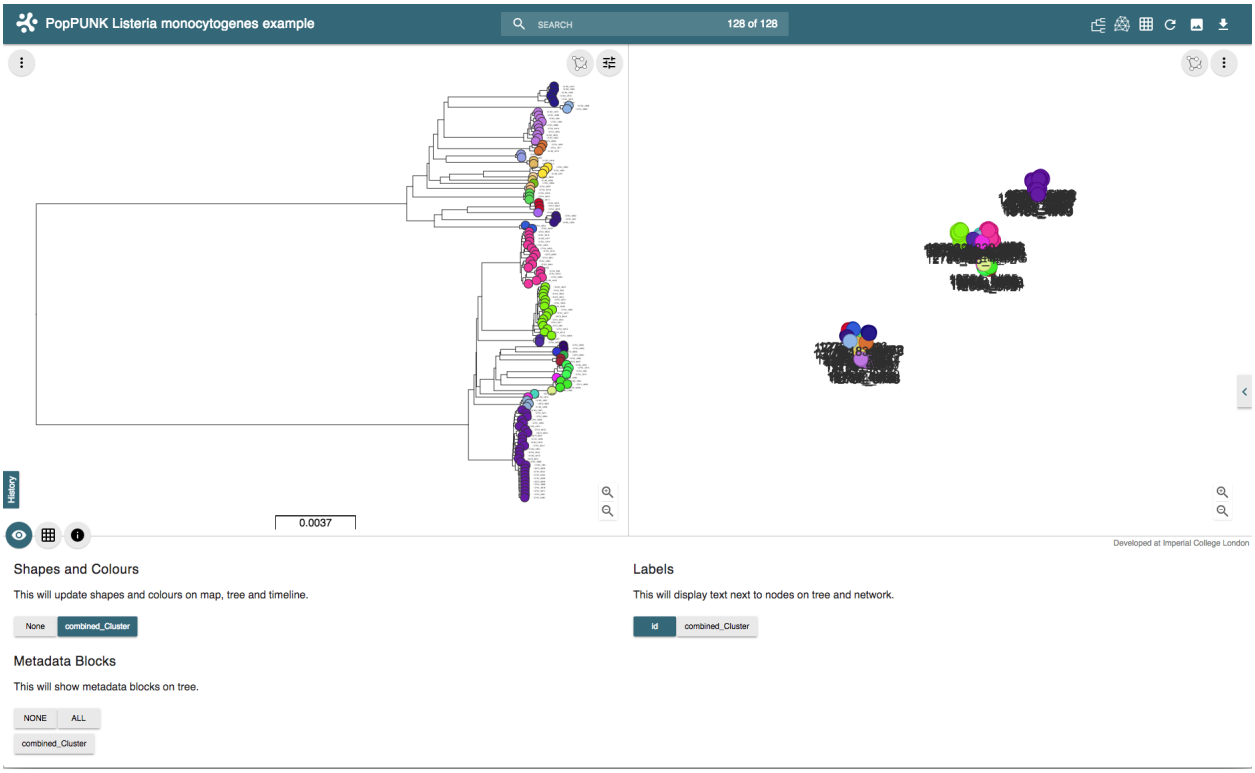
- `--microreact` – Files for Microreact (see below).
- `--rapidnj rapidnj` – Perform core NJ tree construction using rapidnj, which is much faster than the default implementation. The argument points to the rapidnj binary.

- `--cytoscape` – Files to view the network in Cytoscape.
- `--phandango` – Files to view the clustering in phandango.
- `--grapetree` – Files to view the clustering in GrapeTree.

As a brief example, in the `lm_example` folder find the files:

- `lm_example_phandango_clusters.csv`
- `lm_example_perplexity20.0_accessory_tsne.dot`
- `lm_example_core_NJ.nwk`

And drag-and-drop these into the browser at https://microreact.org/upload. This will produce a visualisation with a core genome phylogeny on the left, and an embedding of the accessory distances on the right. Each sample is coloured by its cluster:



The interactive version can be found at https://microreact.org/project/rJJ-cXOum.

CHAPTER 4

Tutorial

This tutorial will guide you through the use of the four modes of PopPUNK, explaining when to use each one. In places we refer to *Troubleshooting* which explains how to deal with common problems when the defaults don't quite work.

The first two steps can be run together in a single command using `--easy-run`, which in many cases will work without need for further modification.

In this tutorial we will work with the Salmonella genomes reviewed by Alikhan et al which can be downloaded from EnteroBase.

- *Creating a database*
  - *Output files*
  - *Relevant command line options*
- *Fitting the model*
  - *Using DBSCAN*
  - *Use of full-db*
  - *Providing previous cluster definitions*
  - *Output files*
  - *Relevant command line options*
- *Refining a model*
  - *Output files*
  - *Relevant command line options*
- *Applying a single distance threshold*
- *Assigning queries*

    - *Using core/accessory only*
    - *Output files*
    - *Relevant command line options*
- *Creating external visualisations from a fitted model*
- *Using a previous model with a new database*

4.1 Creating a database

To analyse a population from scratch (where PopPUNK hasn't been used before) the first step is to create a PopPUNK database, which is essentially a list of all the core and accessory distances between each pair of isolates in the collection.

The basic command to do this is as follows:

```
poppunk --create-db --r-files reference_list.txt --output strain_db --threads 2 --
↪plot-fit 5
```

Where `references.txt` is a list of fasta assemblies to analyse, created by, for example:

```
ls assemblies/*.fasta > reference_list.txt
```

The references will first be hashed at different k-mer lengths, then pairwise distances are calculated, which are finally converted into core and accessory distances:

```
PopPUNK (POPulation Partitioning Using Nucleotide Kmers)
Mode: Building new database from input sequences
Creating mash database for k = 13
Random 13-mer probability: 0.06
Creating mash database for k = 17
Random 17-mer probability: 0.00
Creating mash database for k = 15
Random 15-mer probability: 0.00
Creating mash database for k = 19
Random 19-mer probability: 0.00
Creating mash database for k = 21
Random 21-mer probability: 0.00
Creating mash database for k = 25
Random 25-mer probability: 0.00
Creating mash database for k = 23
Random 23-mer probability: 0.00
Creating mash database for k = 27
Random 27-mer probability: 0.00
Creating mash database for k = 29
Random 29-mer probability: 0.00
mash dist -p 2 ./strain_db/strain_db.13.msh ./strain_db/strain_db.13.msh 2> strain_db.
↪err.log
mash dist -p 2 ./strain_db/strain_db.15.msh ./strain_db/strain_db.15.msh 2> strain_db.
↪err.log
mash dist -p 2 ./strain_db/strain_db.17.msh ./strain_db/strain_db.17.msh 2> strain_db.
↪err.log
mash dist -p 2 ./strain_db/strain_db.19.msh ./strain_db/strain_db.19.msh 2> strain_db.
↪err.log
```

(continues on next page)

(continued from previous page)

```
mash dist -p 2 ./strain_db/strain_db.21.msh ./strain_db/strain_db.21.msh 2> strain_db.
↪err.log
mash dist -p 2 ./strain_db/strain_db.23.msh ./strain_db/strain_db.23.msh 2> strain_db.
↪err.log
mash dist -p 2 ./strain_db/strain_db.25.msh ./strain_db/strain_db.25.msh 2> strain_db.
↪err.log
mash dist -p 2 ./strain_db/strain_db.27.msh ./strain_db/strain_db.27.msh 2> strain_db.
↪err.log
mash dist -p 2 ./strain_db/strain_db.29.msh ./strain_db/strain_db.29.msh 2> strain_db.
↪err.log
Calculating core and accessory distances

Done
```

We would recommend using as many threads as available for maximum speed (even if #threads > #k-mers). To convert k-mer distances into core and accessory distances the following relationship is used:

$$\mathrm{pr}(a, b) = (1 - a)(1 - c)^k$$
$$\log(\mathrm{pr}(a, b)) = \log(1 - a) + k \cdot \log(1 - c)$$

Where $\mathrm{pr}(a, b)$ is the proportion of k-mers matching at length k between sequences a and b. In log-linear space this is linear by k-mer length, and a constrained least squared fit gives the accessory distance (the intercept) and the core distance (the slope).

Warning: A straight line fit is required for correct calculation of core and accessory distances. To inspect this the use of the `--plot-fit` options is generally recommended to inspect some of the regressions. Choice of min-k depends on this, and is discussed in *Choosing the right k-mer lengths*.

4.1.1 Output files

This will create two files *strain_db/strain_db.dists.npy* and *strain_db/strain_db.dists.pkl* which store the distances and strain names respectively. These are then used in *Fitting the model*.

There are also databases of sketches at each k-mer length (**.msh*) which can be re-used if the same data is fitted with a new range of k-mer lengths. Otherwise they should be recalculated by specifying `--overwrite`.

4.1.2 Relevant command line options

The following command line options can be used in this mode:

Mode of operation:

--create-db Create pairwise distances database between reference sequences

Input files:

--r-files R_FILES File listing reference input assemblies

Output options:

--output OUTPUT Prefix for output files (required)

--plot-fit PLOT_FIT Create this many plots of some fits relating k-mer to core/accessory distances [default = 0]

--overwrite Overwrite any existing database files

Quality control options:

--max-a-dist MAX_A_DIST Maximum accessory distance to permit [default = 0.5]

--ignore-length Ignore outliers in terms of assembly length [default = False]

Kmer comparison options:

--min-k MIN_K Minimum kmer length [default = 9]

--max-k MAX_K Maximum kmer length [default = 29]

--k-step K_STEP K-mer step size [default = 4]

--sketch-size SKETCH_SIZE Kmer sketch size [default = 10000]

Other options:

--mash MASH Location of mash executable

--threads THREADS Number of threads to use during database querying [default = 1]

--no-stream Use temporary files for mash dist interfacing. Reduce memory use/increase disk use for large datasets

4.2 Fitting the model

The basic command used to fit the model is as follows:

```
poppunk_runner.py --fit-model --distances strain_db/strain_db.dists --output strain_
↪db --full-db --ref-db strain_db --K 2
```

This will fit a mixture of up to three 2D Gaussians to the distribution of core and accessory distances:

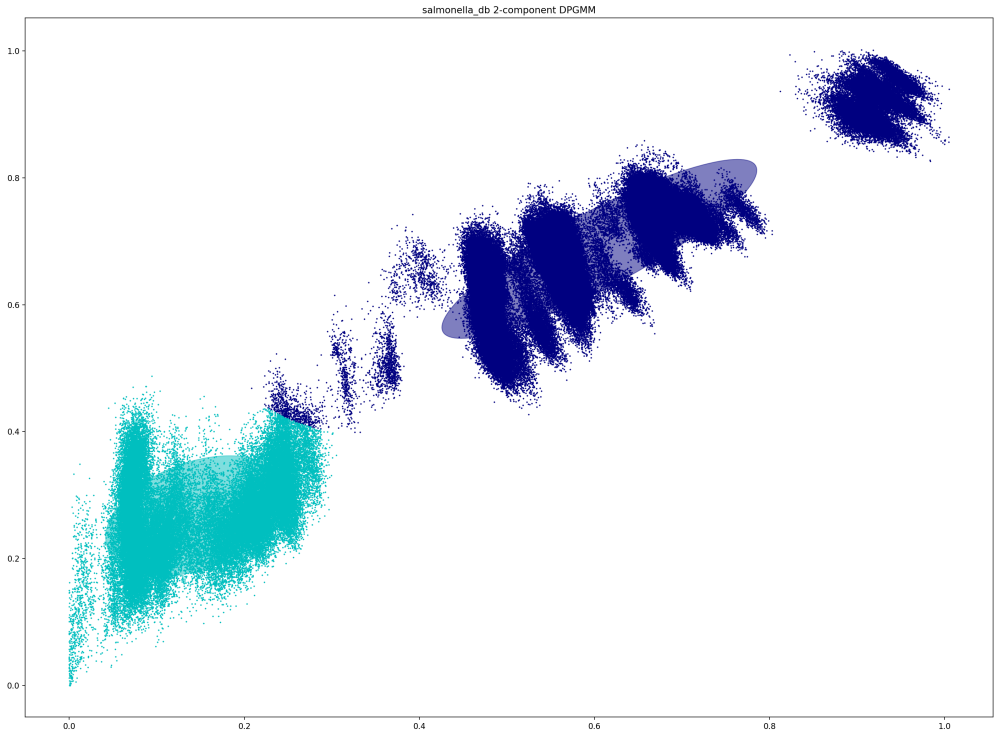
```
PopPUNK (POPulation Partitioning Using Nucleotide Kmers)
Mode: Fitting model to reference database

Fit summary:
   Avg. entropy of assignment        0.0042
   Number of components used 2
Network summary:
   Components        12
   Density   0.1852
   Transitivity      0.9941
   Score     0.8100

Done
```

The default is to fit two components, one for between-strain and one for within-strain distances. There are a number of summary statistics which you can use to assess the fit:

Statistic	Interpretation
Avg. entropy of assignment	How confidently each distance is assigned to a component. Closer to zero is more confident, and indicates less overlap of components, which may be indicative of less recombination overall.
Number of components used	The number of mixture components actually used, which may be less than the maximum allowed.
Components	The number of components in the network == the number of population clusters
Density	The proportion of edges in the network. 0 is no links, 1 is every link. Lower is better.
Transitivity	The transitivity of the network, between 0 and 1. Higher is better
Score	Network score based on density and transitivity. Higher is better.

Important: This is the most important part of getting a good estimation of population structure. In many cases choosing a sensible `--K` will get a fit with a good score, but in more complex cases PopPUNK allows alternative model fitting. See *Refining a model* for a discussion on how to improve the model fit.

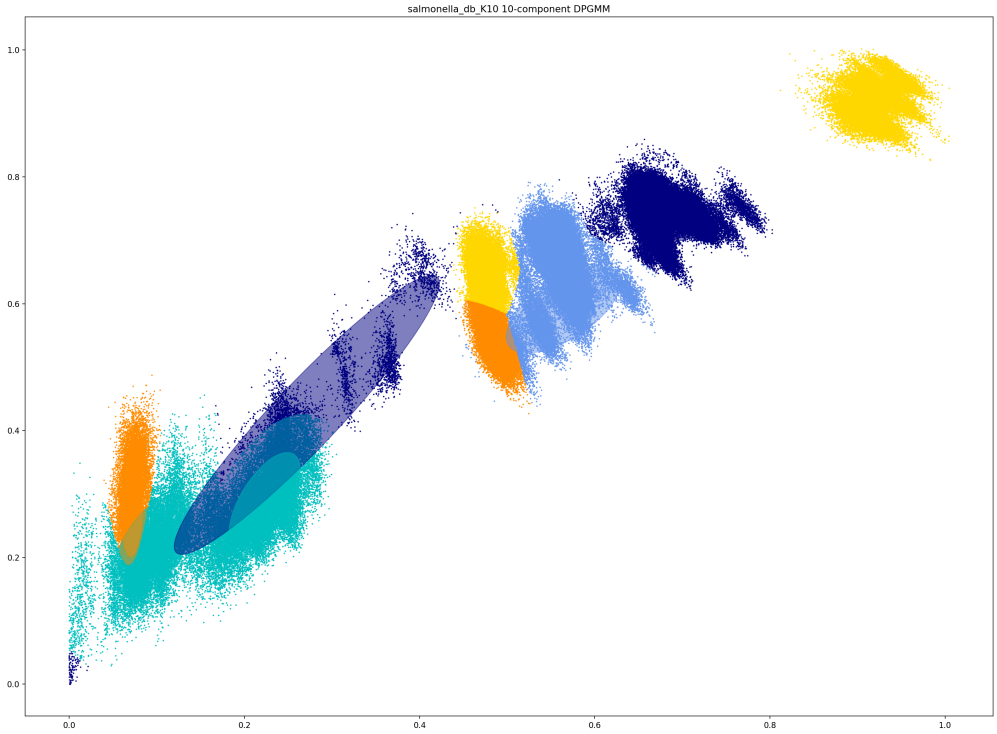
The most useful plot is *strain_db_DPGMM_fit.png* which shows the clustering:



This looks reasonable. The component closest to the origin is used to create a network where isolates determined to be within the same strain are linked by edges. The connected components of this network are then the population clusters.

In this case, allowing more components (`--K 10`) gives a worse fit as more complexity is introduced arbitrarily:

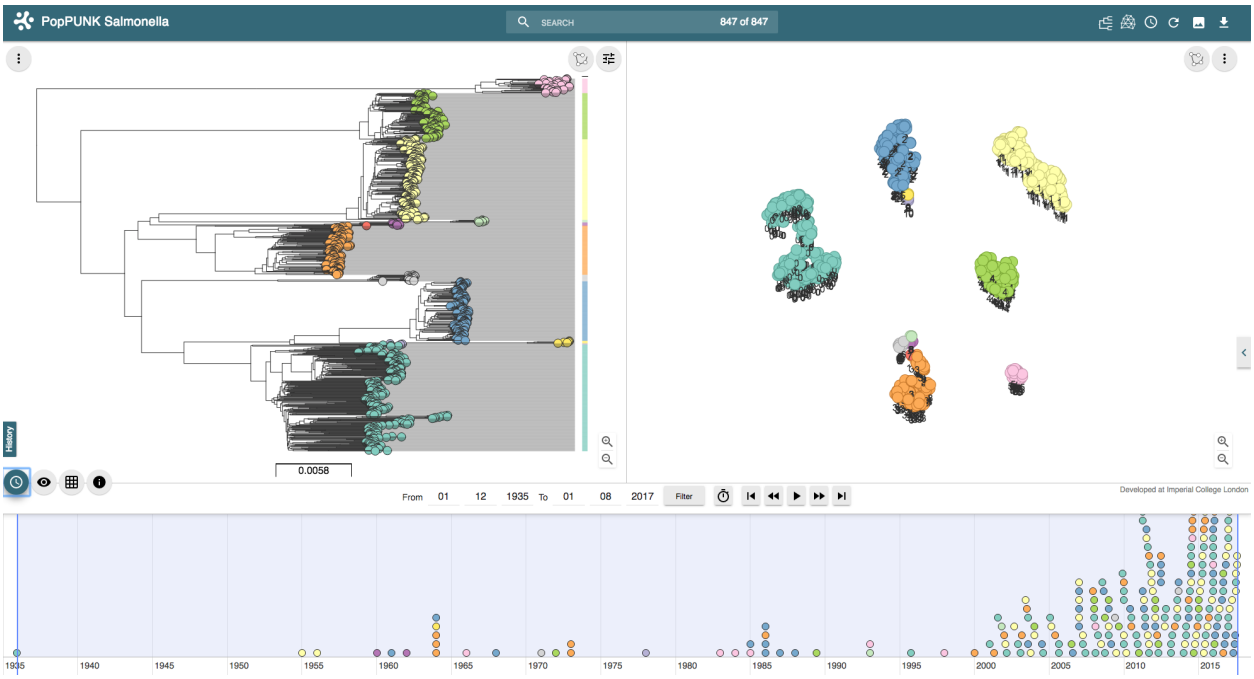
```
PopPUNK (POPulation Partitioning Using Nucleotide Kmers)
Mode: Fitting model to reference database

Fit summary:
     Avg. entropy of assignment      0.0053
     Number of components used       10
Network summary:
     Components      121
     Density 0.0534
     Transitivity    0.8541
     Score   0.8085

Done
```



In this case the fit is too conservative, and the network has a high number of components.

Once you have a good fit, run again with the `--microreact` option (and `--rapidnj` if you have rapidnj installed). This will create output files which can dragged and dropped into Microreact for visualisation of the results.

Drag the files *strain_db_microreact_clusters.csv*, *strain_db_perplexity5.0_accessory_tsne*, and *strain_db_core_NJ_microreact.nwk* onto Microreact. For this example, the output is at https://microreact.org/project/Skg0j9sjz (this also includes a CSV of additional metadata downloaded from EnteroBase and supplied to PopPUNK with `--info-csv`).



The left panel shows the tree from the core distances, and the right panel the embedding of accessory distances (at perplexity 30). Differences in clustering between the two can be informative of separate core and accessory evolution, but in this case they are correlated as expected for strains. Tips are coloured by the PopPUNK inferred cluster.

Note: t-SNE can be sensitive to the `--perplexity` parameter provided. This can be re-run as necessary by changing the parameter value. Use a value between 5 and 50, but see *Setting the perplexity parameter for t-SNE* for further discussion.

4.2.1 Using DBSCAN

Clustering can also be performed by using DBSCAN, which uses the HDBSCAN* library. Run the same `fit-model` command as above, but add the `--dbscan` option:

```
poppunk-runner.py --fit-model --distances strain_db/strain_db.dists --output strain_
↪db --full-db --ref-db strain_db --dbscan
```

The output is as follows:

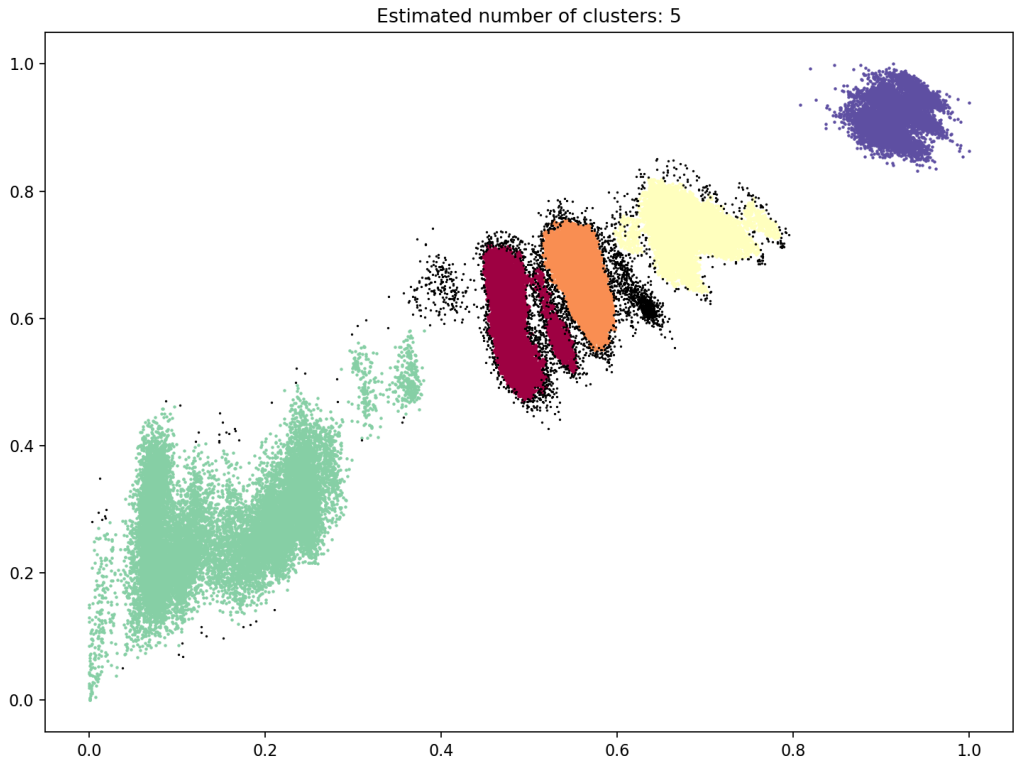
```
PopPUNK (POPulation Partitioning Using Nucleotide Kmers)
Mode: Fitting model to reference database

Fit summary:
     Number of clusters      5
     Number of datapoints    100000
     Number of assignments   100000
Network summary:
     Components      9
     Density 0.1906
     Transitivity    0.9979
     Score   0.8077
```

(continues on next page)

(continued from previous page)

```
Done
```

In this case the fit is quite similar to the mixture model:



The small black points are classified as noise, and are not used in the network construction.

Warning: If there are a lot of noise points (in black) then fit refinement will be subsequently required, as these points will not contribute to the network. See *Refining a model*.

4.2.2 Use of full-db

By default the `--full-db` option is off. When on this will keep every sample in the analysis in the database for future querying.

When off (the default) representative samples will be picked from each cluster by choosing only one reference sample from each clique (where all samples in a clqiue have a within-cluster link to all other samples in the clique). This can significantly reduce the database size for future querying without loss of accuracy. Representative samples are written out to a .refs file, and a new database is sketched for future distance comparison.

In the case of the example above, this reduces from 848 to 14 representatives (one for each of the twelve clusters, except for 3 and 6 which have two each).

If the program was run through using `--full-db`, references can be picked and a full directory with a PopPUNK model for query assignment created using the `poppunk_references` program:

```
poppunk_references --network strain_db/strain_db_graph.gpickle --ref-db strain_db --
↪distances strain_db/strain_db.dists \
--model strain_db --output strain_references --threads 4
```

Using the `--model` will also copy over the model fit, so that the entire PopPUNK database is in a single directory.

4.2.3 Providing previous cluster definitions

By using the option `--external-clustering` one can provide cluster names or labels that have been previously defined by any other method. This could include, for example, another clustering methods IDs, serotypes, clonal complexes and MLST assignments. The input is a CSV file which is formatted as follows:

```
sample,serotype,MLST
sample1,12,34
sample2,23F,1
```

This can contain any subset of the samples in the input, and additionally defined samples will be safely ignored.

PopPUNK will output a file `_external_clusters.csv` which has, for each sample in the input (either reference or query, depending on the mode it was run in), a list of of these labels which have been assigned to any sample in the PopPUNK cluster. Samples are expected to have a single label, but may receive multiple labels. Novel query clusters will not receive labels. An example of output:

```
sample,serotype,MLST
sample1,12,34
sample2,23F,1
sample3,15B;15C,21
sample4,NA,NA
```

4.2.4 Output files

- strain_db.search.out – the core and accessory distances between all pairs.
- strain_db_graph.gpickle – the network used to predict clusters.
- strain_db_DPGMM_fit.png – scatter plot of all distances, and mixture model fit and assignment.
- strain_db_DPGMM_fit_contours.png – contours of likelihood function fitted to data (blue low -> yellow high). The thick red line is the decision boundary between within- and between-strain components.
- strain_db_distanceDistribution.png – scatter plot of the distance distribution fitted by the model, and a kernel-density estimate.
- strain_db.csv – isolate names and the cluster assigned.
- strain_db.png – unclustered distribution of distances used in the fit (subsampled from total).
- strain_db.npz – save fit parameters.
- strain_db.refs – representative references in the new database (unless `--full-db` was used).

If `--dbscan` was used:

- strain_db_dbscan.png – scatter plot of all distances, and DBSCAN assignment.

If `--external-clustering` was used:

- strain_db_external_clusters.csv – a CSV file relating the samples to previous clusters provided in the input CSV.

If `--microreact` was used:

- strain_db_core_dists.csv – matrix of pairwise core distances.
- strain_db_acc_dists.csv – matrix of pairwise accessory distances.
- strain_db_core_NJ_microreact.nwk – neighbour joining tree using core distances (for microreact).
- strain_db_perplexity5.0_accessory_tsne.dot – t-SNE embedding of accessory distances at given perplexity (for microreact).
- strain_db_microreact_clusters.csv – cluster assignments plus any epi data added with the `--info-csv` option (for microreact).

If `--cytoscape` was used:

- strain_db_cytoscape.csv – cluster assignments plus any epi data added with the `--info-csv` option (for cytoscape).
- strain_db_cytoscape.graphml – XML representation of resulting network (for cytoscape).

4.2.5 Relevant command line options

The following command line options can be used in this mode:

Mode of operation:

--fit-model Fit a mixture model to a reference database

Input files:

--ref-db REF_DB Location of built reference database

--distances DISTANCES Prefix of input pickle of pre-calculated distances

--external-clustering EXTERNAL_CLUSTERING File with cluster definitions or other labels generated with any other method.

Output options:

--output OUTPUT Prefix for output files (required)

--full-db Keep full reference database, not just representatives

--overwrite Overwrite any existing database files

Quality control options:

--max-a-dist MAX_A_DIST Maximum accessory distance to permit [default = 0.5]

Model fit options:

--K K Maximum number of mixture components [default = 2]

--dbscan Use DBSCAN rather than mixture model

--D D Maximum number of clusters in DBSCAN fitting [default = 100]

--min-cluster-prop MIN_CLUSTER_PROP Minimum proportion of points in a cluster in DBSCAN fitting [default = 0.0001]

Further analysis options:

--microreact Generate output files for microreact visualisation

--cytoscape Generate network output files for Cytoscape

--phandango Generate phylogeny and TSV for Phandango visualisation

--grapetree Generate phylogeny and CSV for grapetree visualisation

--rapidnj RAPIDNJ Path to rapidNJ binary to build NJ tree for Microreact

--perplexity PERPLEXITY Perplexity used to calculate t-SNE projection (with –microreact) [default=20.0]

--info-csv INFO_CSV Epidemiological information CSV formatted for microreact (can be used with other outputs)

Other options:

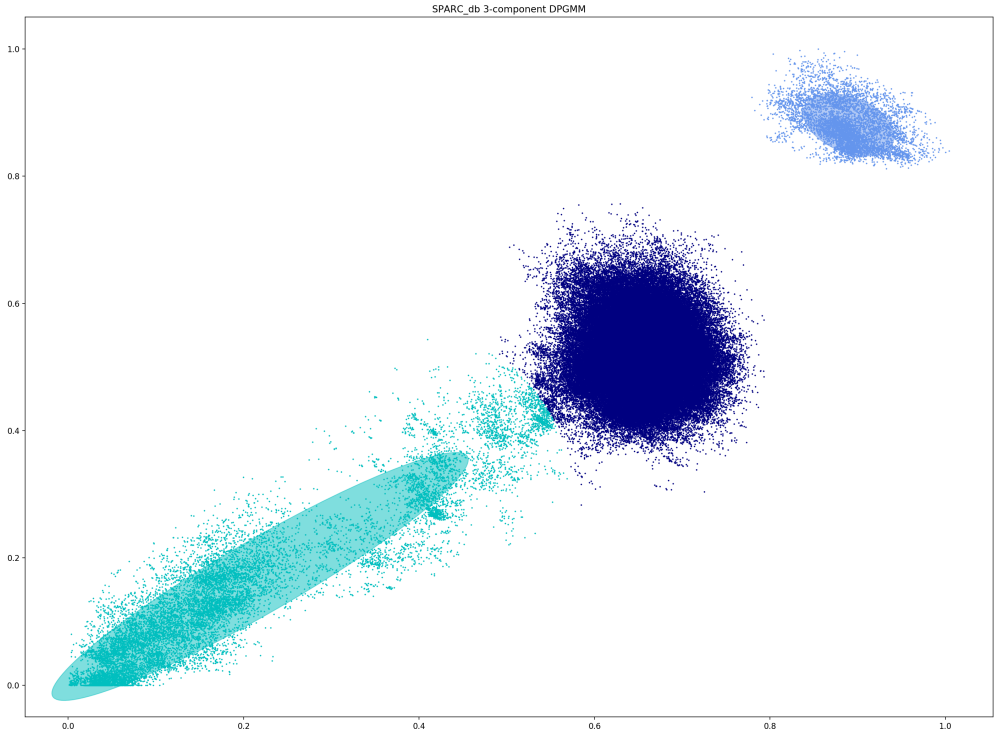
--mash MASH Location of mash executable

Note: If using the default mixture model threads will only be used if `--full-db` is *not* specified and sketching of the representatives is performed at the end.

4.3 Refining a model

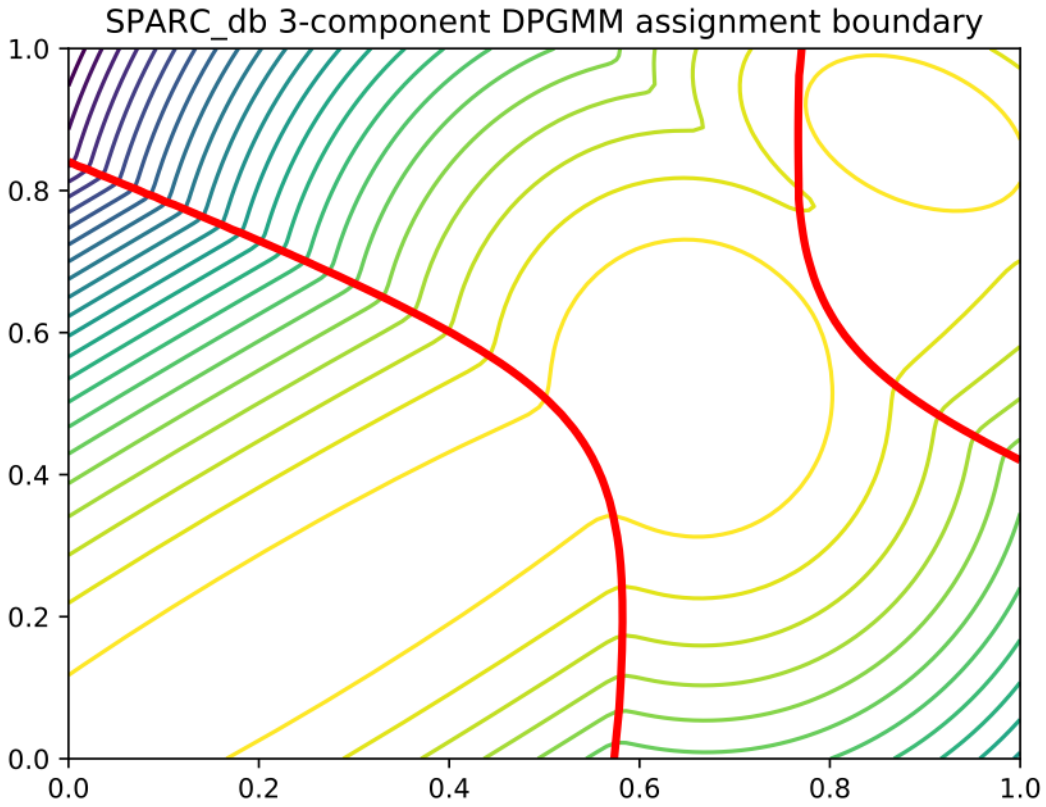
In species with a relatively high recombination rate the distinction between the within- and between-strain distributions may be blurred in core and accessory space. This does not give the mixture model enough information to draw a good boundary as the likelihood is very flat in this region.

See this example of 616 *S. pneumoniae* genomes with the DPGMM fit. These genomes were collected from Massachusetts, first reported here and can be accessed here.



Although the score of this fit looks ok (0.904), inspection of the network and microreact reveals that it is too liberal and clusters have been merged. This is because some of the blur between the origin and the central distribution has been included, and connected clusters together erroneously.

The likelihood of the model fit and the decision boundary looks like this:



Using the core and accessory distributions alone does not give much information about exactly where to put the boundary, and the only way to fix this would be by specifying strong priors on the weights of the distributions. Fortunately the network properties give information in the region, and we can use `--refine-fit` to tweak the existing fit and pick a better boundary.

Run:

```
poppunk --refine-model --distances strain_db/strain_db.dists --output strain_db --
↪full-db --ref-db strain_db --threads 4
```

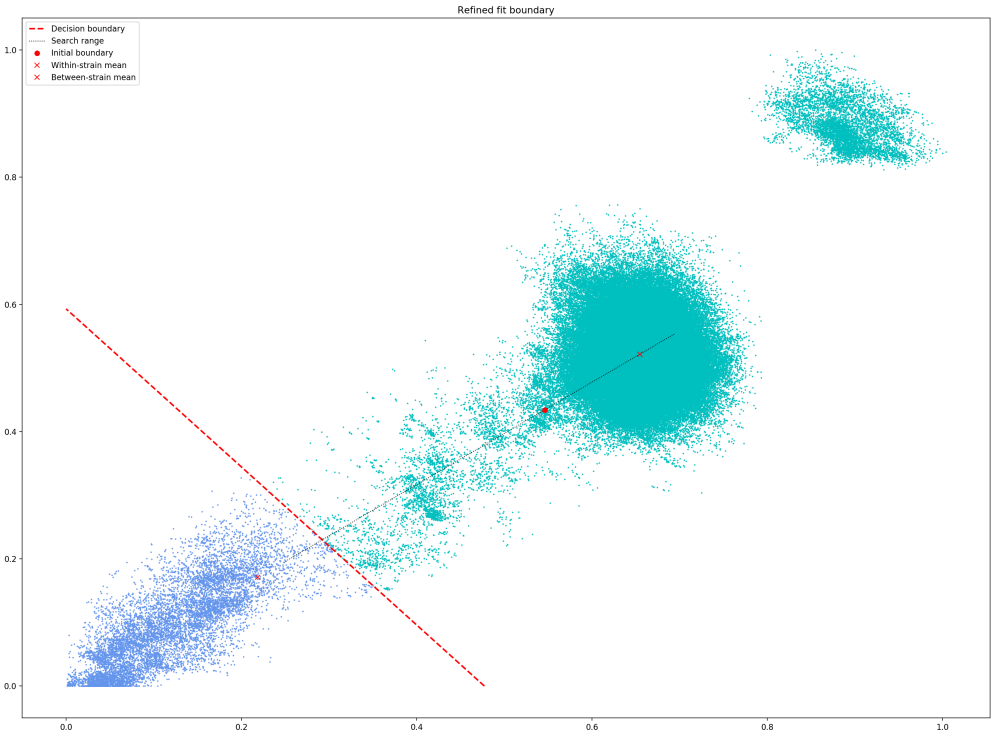
Briefly:

- A line between the within- and between-strain means is constructed
- The point on this line where samples go from being assigned as within-strain to between-strain is used as the starting point
- A line normal to the first line, passing through this point is constructed. The triangle formed by this line and the x- and y-axes is now the decision boundary. Points within this line are within-strain.
- The starting point is shifted by a distance along the first line, and a new decision boundary formed in the same way. The network is reconstructed.
- The shift of the starting point is optimised, as judged by the network score. First globally by a grid search, then locally near the global optimum.

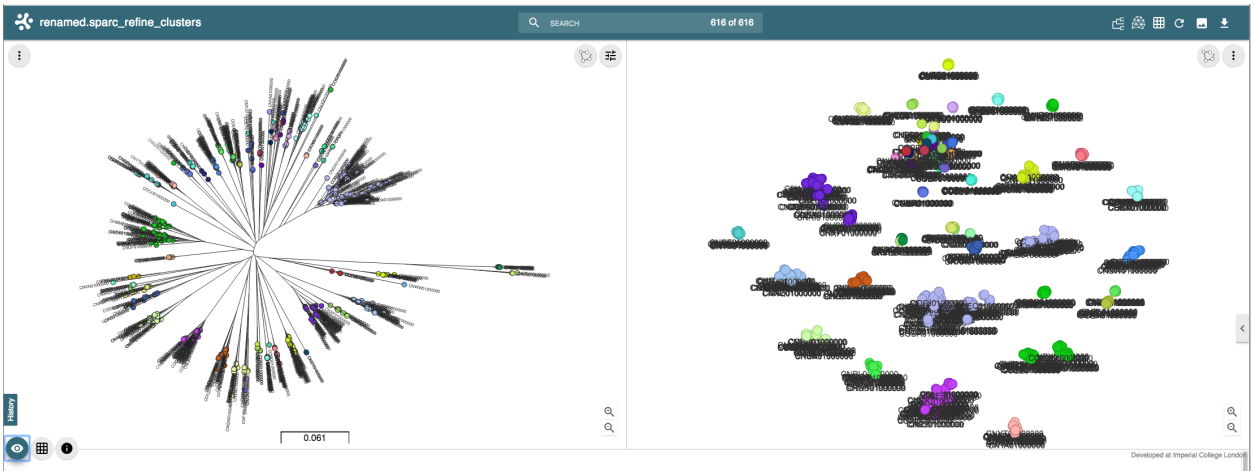
If the mixture model does not give any sort of reasonable fit to the points, see *Using fit refinement when mixture model totally fails* for details about how to set the starting parameters for this mode manually.

The score is a function of transitivity (which is expected to be high, as everything within a cluster should be the same strain as everything else in the cluster) and density (which should be low, as there are far fewer within- than between-strain links).

Here is the refined fit, which has a score of 0.939, and 62 rather than 32 components:



Which, looking at the microreact output, is much better:



The core and accessory distances can also be used on their own. Add the `--indiv-refine` option to refine the fit to these two distances independently (see *Using core/accessory only* for more information).

4.3.1 Output files

The files are as for `--fit-model` (*Output files*), and also include:

- strain_db_refined_fit.png – A plot of the new linear boundary, and core and accessory distances coloured by assignment to either side of this boundary.
- strain_db_refined_fit.npz – The saved parameters of the refined fit.

If `--indiv-refine` was used, a copy of the *_clusters.csv* and network *.gpickle* files for core and accessory only will also be produced.

4.3.2 Relevant command line options

The following command line options can be used in this mode:

Mode of operation:

--refine-model Refine the accuracy of a fitted model

Input files:

--ref-db REF_DB Location of built reference database

--distances DISTANCES Prefix of input pickle of pre-calculated distances

--external-clustering EXTERNAL_CLUSTERING File with cluster definitions or other labels generated with any other method.

Output options:

--output OUTPUT Prefix for output files (required)

--full-db Keep full reference database, not just representatives

--overwrite Overwrite any existing database files

Quality control options:

--max-a-dist MAX_A_DIST Maximum accessory distance to permit [default = 0.5]

Refine model options:

--pos-shift POS_SHIFT Maximum amount to move the boundary away from origin [default = 0.2]

--neg-shift NEG_SHIFT Maximum amount to move the boundary towards the origin [default = 0.4]

--manual-start MANUAL_START A file containing information for a start point. See documentation for help.

--indiv-refine Also run refinement for core and accessory individually

--no-local Do not perform the local optimization step (speed up on very large datasets)

Further analysis options:

--microreact Generate output files for microreact visualisation

--cytoscape Generate network output files for Cytoscape

--phandango Generate phylogeny and TSV for Phandango visualisation

--grapetree Generate phylogeny and CSV for grapetree visualisation

--rapidnj RAPIDNJ Path to rapidNJ binary to build NJ tree for Microreact

--perplexity PERPLEXITY Perplexity used to calculate t-SNE projection (with –microreact) [default=20.0]

--info-csv INFO_CSV Epidemiological information CSV formatted for microreact (can be used with other outputs)

Other options:

--mash MASH Location of mash executable

--threads THREADS Number of threads to use during database querying [default = 1]

Note: Threads are used for the global optimisation step only. If the local optimisation step is slow, turn it off with `--no-local`.

4.4 Applying a single distance threshold

If you want to find clusters beneath a genetic distance cutoff, but using a network which forms clusters by joining samples beneath this threshold, you can use `--threshold`. This will connect samples with core distances below the provided threshold:

```
poppunk --threshold 0.05 --distances strain_db/strain_db.dists --output strain_db --
↪full-db --ref-db strain_db
```

4.5 Assigning queries

Once a database has been built and a model fitted (either in one step with `--easy-run`, or having run both steps separately) new sequences can be assigned to a cluster using `--assign-queries`. This process is much quicker than building a database of all sequences from scratch, and will use the same model fit as before. Cluster names will not change, unless queries cause clusters to be merged (in which case they will be the previous cluster names, underscore separated).

Having created a file listing the new sequences to assign `query_list.txt`, the command to assign a cluster to new sequences is:

```
poppunk --assign-query --ref-db strain_db --q-files query_list.txt --output strain_
↪query --threads 3 --update-db
```

Where *strain_db* is the output of the previous `PopPUNK` commands, containing the model fit and distances.

Note: It is possible to specify a model fit in a separate directory from the distance sketches using `--model-dir`. Similarly a clustering and network can be specified using `--previous-clustering`.

First, distances between queries and sequences in the reference database will be calculated. The model fit (whether mixture model, DBSCAN or refined) will be loaded and used to determine matches to existing clusters:

```
PopPUNK (POPulation Partitioning Using Nucleotide Kmers)
Mode: Assigning clusters of query sequences

Creating mash database for k = 15
Random 15-mer probability: 0.00
Creating mash database for k = 13
Random 13-mer probability: 0.04
Creating mash database for k = 17
Random 17-mer probability: 0.00
Creating mash database for k = 19
Random 19-mer probability: 0.00
Creating mash database for k = 21
Random 21-mer probability: 0.00
Creating mash database for k = 23
Random 23-mer probability: 0.00
Creating mash database for k = 25
Random 25-mer probability: 0.00
Creating mash database for k = 27
Random 27-mer probability: 0.00
Creating mash database for k = 29
Random 29-mer probability: 0.00
mash dist -p 3 ./strain_db/strain_db.13.msh ./strain_query/strain_query.13.msh 2>␣
↪strain_db.err.log
mash dist -p 3 ./strain_db/strain_db.15.msh ./strain_query/strain_query.15.msh 2>␣
↪strain_db.err.log
mash dist -p 3 ./strain_db/strain_db.17.msh ./strain_query/strain_query.17.msh 2>␣
↪strain_db.err.log
mash dist -p 3 ./strain_db/strain_db.19.msh ./strain_query/strain_query.19.msh 2>␣
↪strain_db.err.log
mash dist -p 3 ./strain_db/strain_db.21.msh ./strain_query/strain_query.21.msh 2>␣
↪strain_db.err.log
mash dist -p 3 ./strain_db/strain_db.23.msh ./strain_query/strain_query.23.msh 2>␣
↪strain_db.err.log
mash dist -p 3 ./strain_db/strain_db.25.msh ./strain_query/strain_query.25.msh 2>␣
↪strain_db.err.log
mash dist -p 3 ./strain_db/strain_db.27.msh ./strain_query/strain_query.27.msh 2>␣
↪strain_db.err.log
mash dist -p 3 ./strain_db/strain_db.29.msh ./strain_query/strain_query.29.msh 2>␣
↪strain_db.err.log
Calculating core and accessory distances
Loading DBSCAN model
```

If query sequences were found which didn't match an existing cluster they will start a new cluster. `PopPUNK` will check whether any of these novel clusters should be merged, based on the model fit:

```
Found novel query clusters. Calculating distances between them:
Creating mash database for k = 13
Random 13-mer probability: 0.04
Creating mash database for k = 15
Random 15-mer probability: 0.00
Creating mash database for k = 17
Random 17-mer probability: 0.00
Creating mash database for k = 19
Random 19-mer probability: 0.00
Creating mash database for k = 21
Random 21-mer probability: 0.00
Creating mash database for k = 23
```

(continues on next page)

(continued from previous page)

```
Random 23-mer probability: 0.00
Creating mash database for k = 25
Random 25-mer probability: 0.00
Creating mash database for k = 27
Random 27-mer probability: 0.00
Creating mash database for k = 29
Random 29-mer probability: 0.00
mash dist -p 3 ././strain_dbij_sqnjr_tmp/./strain_dbij_sqnjr_tmp.13.msh ././strain_
↪dbij_sqnjr_tmp/./strain_dbij_sqnjr_tmp.13.msh 2> ./strain_dbij_sqnjr_tmp.err.log
mash dist -p 3 ././strain_dbij_sqnjr_tmp/./strain_dbij_sqnjr_tmp.15.msh ././strain_
↪dbij_sqnjr_tmp/./strain_dbij_sqnjr_tmp.15.msh 2> ./strain_dbij_sqnjr_tmp.err.log
mash dist -p 3 ././strain_dbij_sqnjr_tmp/./strain_dbij_sqnjr_tmp.17.msh ././strain_
↪dbij_sqnjr_tmp/./strain_dbij_sqnjr_tmp.17.msh 2> ./strain_dbij_sqnjr_tmp.err.log
mash dist -p 3 ././strain_dbij_sqnjr_tmp/./strain_dbij_sqnjr_tmp.19.msh ././strain_
↪dbij_sqnjr_tmp/./strain_dbij_sqnjr_tmp.19.msh 2> ./strain_dbij_sqnjr_tmp.err.log
mash dist -p 3 ././strain_dbij_sqnjr_tmp/./strain_dbij_sqnjr_tmp.21.msh ././strain_
↪dbij_sqnjr_tmp/./strain_dbij_sqnjr_tmp.21.msh 2> ./strain_dbij_sqnjr_tmp.err.log
mash dist -p 3 ././strain_dbij_sqnjr_tmp/./strain_dbij_sqnjr_tmp.23.msh ././strain_
↪dbij_sqnjr_tmp/./strain_dbij_sqnjr_tmp.23.msh 2> ./strain_dbij_sqnjr_tmp.err.log
mash dist -p 3 ././strain_dbij_sqnjr_tmp/./strain_dbij_sqnjr_tmp.25.msh ././strain_
↪dbij_sqnjr_tmp/./strain_dbij_sqnjr_tmp.25.msh 2> ./strain_dbij_sqnjr_tmp.err.log
mash dist -p 3 ././strain_dbij_sqnjr_tmp/./strain_dbij_sqnjr_tmp.27.msh ././strain_
↪dbij_sqnjr_tmp/./strain_dbij_sqnjr_tmp.27.msh 2> ./strain_dbij_sqnjr_tmp.err.log
mash dist -p 3 ././strain_dbij_sqnjr_tmp/./strain_dbij_sqnjr_tmp.29.msh ././strain_
↪dbij_sqnjr_tmp/./strain_dbij_sqnjr_tmp.29.msh 2> ./strain_dbij_sqnjr_tmp.err.log
Calculating core and accessory distances
```

At this point, cluster assignments for the query sequences are written to a CSV file. Finally, if new clusters were created due to the queries, the database will be updated to reflect this if `--update-db` was used:

```
Creating mash database for k = 13
Random 13-mer probability: 0.04
Overwriting db: ./strain_query/strain_query.13.msh
Creating mash database for k = 15
Random 15-mer probability: 0.00
Overwriting db: ./strain_query/strain_query.15.msh
Creating mash database for k = 17
Random 17-mer probability: 0.00
Overwriting db: ./strain_query/strain_query.17.msh
Creating mash database for k = 19
Random 19-mer probability: 0.00
Overwriting db: ./strain_query/strain_query.19.msh
Creating mash database for k = 21
Random 21-mer probability: 0.00
Overwriting db: ./strain_query/strain_query.21.msh
Creating mash database for k = 23
Random 23-mer probability: 0.00
Overwriting db: ./strain_query/strain_query.23.msh
Creating mash database for k = 25
Random 25-mer probability: 0.00
Overwriting db: ./strain_query/strain_query.25.msh
Creating mash database for k = 27
Random 27-mer probability: 0.00
Overwriting db: ./strain_query/strain_query.27.msh
Creating mash database for k = 29
Random 29-mer probability: 0.00
```

(continues on next page)

(continued from previous page)

```
Overwriting db: ./strain_query/strain_query.29.msh
Writing strain_query/strain_query.13.joined.msh...
Writing strain_query/strain_query.15.joined.msh...
Writing strain_query/strain_query.17.joined.msh...
Writing strain_query/strain_query.19.joined.msh...
Writing strain_query/strain_query.21.joined.msh...
Writing strain_query/strain_query.23.joined.msh...
Writing strain_query/strain_query.25.joined.msh...
Writing strain_query/strain_query.27.joined.msh...
Writing strain_query/strain_query.29.joined.msh...

Done
```

Note: For future uses of `--assign-query`, the database now stored in `strain-query` should be used as the `--ref-db` argument.

4.5.1 Using core/accessory only

In some cases, such as analysis within a lineage, it may be desirable to use only core or accessory distances to classify further queries. This can be achieved by using the `--core-only` or `--accessory-only` options with a fit produced by *Refining a model*. The default is to use the x-axis intercept of the boundary as the core distance cutoff (y-axis for accessory). However, if planning on using this mode we recommend running the refinement with the `--indiv-refine` options, which will allow these boundaries to be placed independently, allowing the best fit in each case:

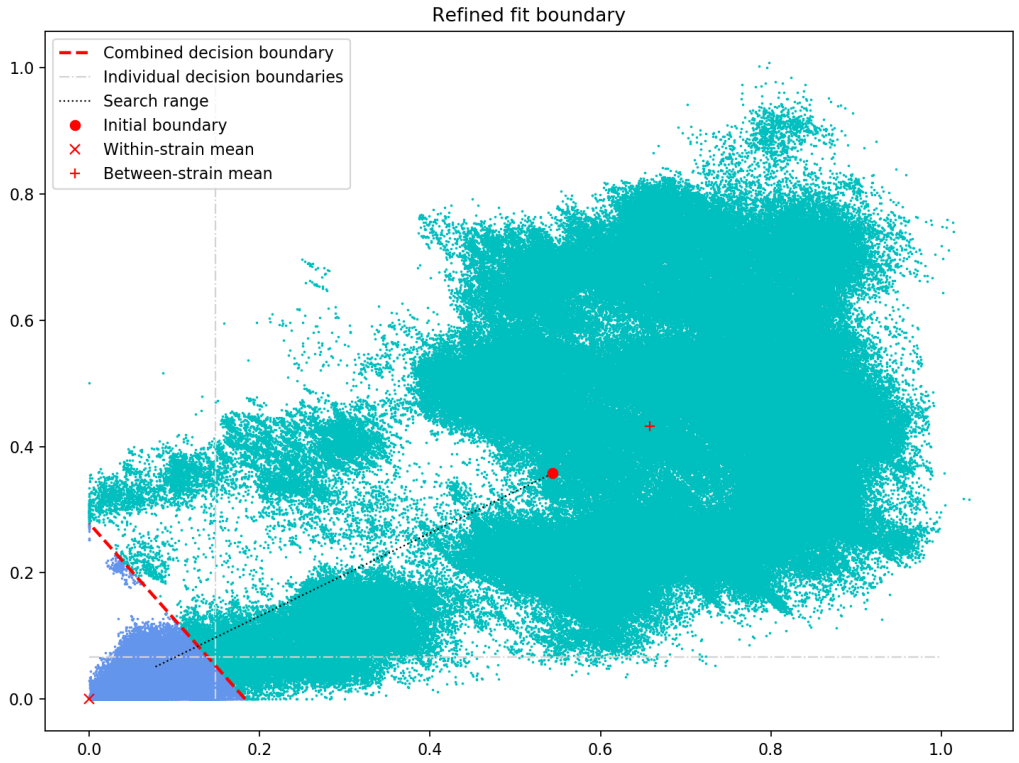
```
poppunk --refine-model --distances strain_db/strain_db.dists --output strain_db --
↪full-db --indiv-refine --ref-db strain_db --threads 4
PopPUNK (POPulation Partitioning Using Nucleotide Kmers)
Mode: Refining model fit using network properties

Loading BGMM 2D Gaussian model
Initial boundary based network construction
Decision boundary starts at (0.54,0.36)
Trying to optimise score globally
Trying to optimise score locally
Refining core and accessory separately
Initial boundary based network construction
Decision boundary starts at (0.54,0.36)
Trying to optimise score globally
Trying to optimise score locally
Initial boundary based network construction
Decision boundary starts at (0.54,0.36)
Trying to optimise score globally
Trying to optimise score locally
Network summary:
     Components      132
     Density 0.0889
     Transitivity    0.9717
     Score   0.8853
Network summary:
     Components      114
     Density 0.0955
     Transitivity    0.9770
```

(continues on next page)

(continued from previous page)

```
     Score    0.8837
Network summary:
     Components       92
     Density 0.0937
     Transitivity     0.9327
     Score    0.8453
writing microreact output:
Building phylogeny
Running t-SNE

Done
```

There are three different networks, and the core and accessory boundaries will also be shown on the *refined_fit.png* plot as dashed gray lines:



4.5.2 Output files

The main output is *strain_query/strain_query_clusters.csv*, which contains the cluster assignments of the query sequences, ordered by frequency.

If `--update-db` was used a full updated database will be written to `--output`, which consists of sketches at each k-mer length (**.msh*), a *search.out* file of distances, and a *.gpickle* of the network.

4.5.3 Relevant command line options

The following command line options can be used in this mode:

Mode of operation:

--assign-query Assign the cluster of query sequences without re- running the whole mixture model

Input files:

--ref-db REF_DB Location of built reference database

--q-files Q_FILES File listing query input assemblies

--external-clustering EXTERNAL_CLUSTERING File with cluster definitions or other labels generated with any other method.

Output options:

--output OUTPUT Prefix for output files (required)

--update-db Update reference database with query sequences

Quality control options:

--max-a-dist MAX_A_DIST Maximum accessory distance to permit [default = 0.5]

--ignore-length Ignore outliers in terms of assembly length [default = False]

Database querying options:

--model-dir MODEL_DIR Directory containing model to use for assigning queries to clusters [default = reference database directory]

--previous-clustering PREVIOUS_CLUSTERING Directory containing previous cluster definitions and network [default = use that in the directory containing the model]

--core-only Use a core-distance only model for assigning queries [default = False]

--accessory-only Use an accessory-distance only model for assigning queries [default = False]

Other options:

--mash MASH Location of mash executable

--threads THREADS Number of threads to use [default = 1]

--no-stream Use temporary files for mash dist interfacing. Reduce memory use/increase disk use for large datasets

--version show program's version number and exit

4.6 Creating external visualisations from a fitted model

Visualisations for external software (Microreact etc) will be created in a mode calling `--fit-model`, `--refine-model` or `--assign-query` if any of the following options were added:

- `--microreact`
- `--cytoscape`

- `--phandango`
- `--grapetree`

Additionally, if `--refine-model`, `--indiv-refine` and `--cytoscape` are all specified, the networks for core and accessory distances only will also be output.

To create these outputs for visualisation after the initial command has been run use the `--generate-viz` mode, with the same options as the original run (plus any specific to the visualisation). In this mode you may also specify a file containing a list of samples to include in the visualisation with `--subset`.

Note: Only a single network will be used in this mode if core and accessory distance restricted models have also been produced. To visualise these instead of the combined fit use `--core-only` or `--accessory-only`.

4.7 Using a previous model with a new database

If you have a model which has been fitted which you wish to apply this to a new reference database, you may do this with `--use-model`. This will take a fitted model, apply it to distances from `--create-db` and produce a network, assignment and reference database for future use with `--assign-query`.

Note: Generally, to use an existing model with new data it is better to `--assign-query` (see *Assigning queries*). This mode can be used when the model, reference database and network are out of sync due to accidentally overwriting one or losing track of versions.

Options are the same as `--fit-model` for GMM and DBSCAN models or `--refine-model` for refined models.

CHAPTER 5

Troubleshooting

This page deals with common issues in running the analysis. For issues with installing or running the software please raise an issue on github.

- *Error/warning messages*
  - *Errors in graph.py*
  - *Trying to create a very large network*
  - *Row name mismatch*
  - *Samples are missing from the final network*
  - *Old cluster split across multiple new clusters*
- *Choosing the right k-mer lengths*
  - *Using fit refinement when mixture model totally fails*
- *Viewing the network with cytoscape*
- *Setting the perplexity parameter for t-SNE*
- *Dealing with poor quality data*
  - *With the distances*
  - *With the network*
- *Removing samples from a database*
- *Memory/run-time issues*

5.1 Error/warning messages

5.1.1 Errors in graph.py

If you get an `AttributeError`:

```
AttributeError: 'Graph' object has no attribute 'node'
```

Then your `networkx` package is out of date. Its version needs to be at >=v2.0.

5.1.2 Trying to create a very large network

When using `--refine-model` you may see the message:

```
Warning: trying to create very large network
```

One or more times. This is triggered if 5×10^5 edges or greater than 40% of the maximum possible number of edges have been added into the network. This suggests that the boundary is too large including too many links as within sample. This isn't necessarily a problem as it can occur at the edge of the optimisation range, so will not be the final optimised result. However, if you have a large number of samples it may make this step run very slowly and/or use a lot of memory. If that is the case, decrease `--pos-shift`.

5.1.3 Row name mismatch

PopPUNK may throw:

```
RuntimeError: Row name mismatch. Old: 6999_2#17.fa,6259_5#6.fa
New: 6952_7#16.fa,6259_5#6.fa
```

This is an error where the mash output order does not match the order in stored databases (`.pkl`). Most likely, the input files are from different runs, possibly due to using `--overwrite`. Run again, giving each step its own output directory.

5.1.4 Samples are missing from the final network

When running `--assign-query` an error such as:

```
WARNING: Samples 7553_5#54.fa,6999_5#1.fa are missing from the final network
```

Means that samples present in `--distances` and or `--ref-db` are not present in the loaded network. This should be considered an error as it will likely lead to other errors and warnings. Make sure the provided network is the one created by applying the `--model-dir` to `--distances`, and that the same output directory has not been used and overwriten by different steps or inputs.

5.1.5 Old cluster split across multiple new clusters

When running `--assign-query`, after distances have been calculated and queries are being assigned warnings such as:

```
WARNING: Old cluster 1 split across multiple new clusters
```

Mean that a single cluster in the original clustering is now split into more than one cluster. This means something has gone wrong, as the addition of new queries should only be able to merge existing clusters, not cause them to split.

Most likely, the `--previous-clustering` directory is inconsistent with the `--ref-db` and/or `--model-dir`. Make sure the clusters are those created from the network being used to assign new queries.

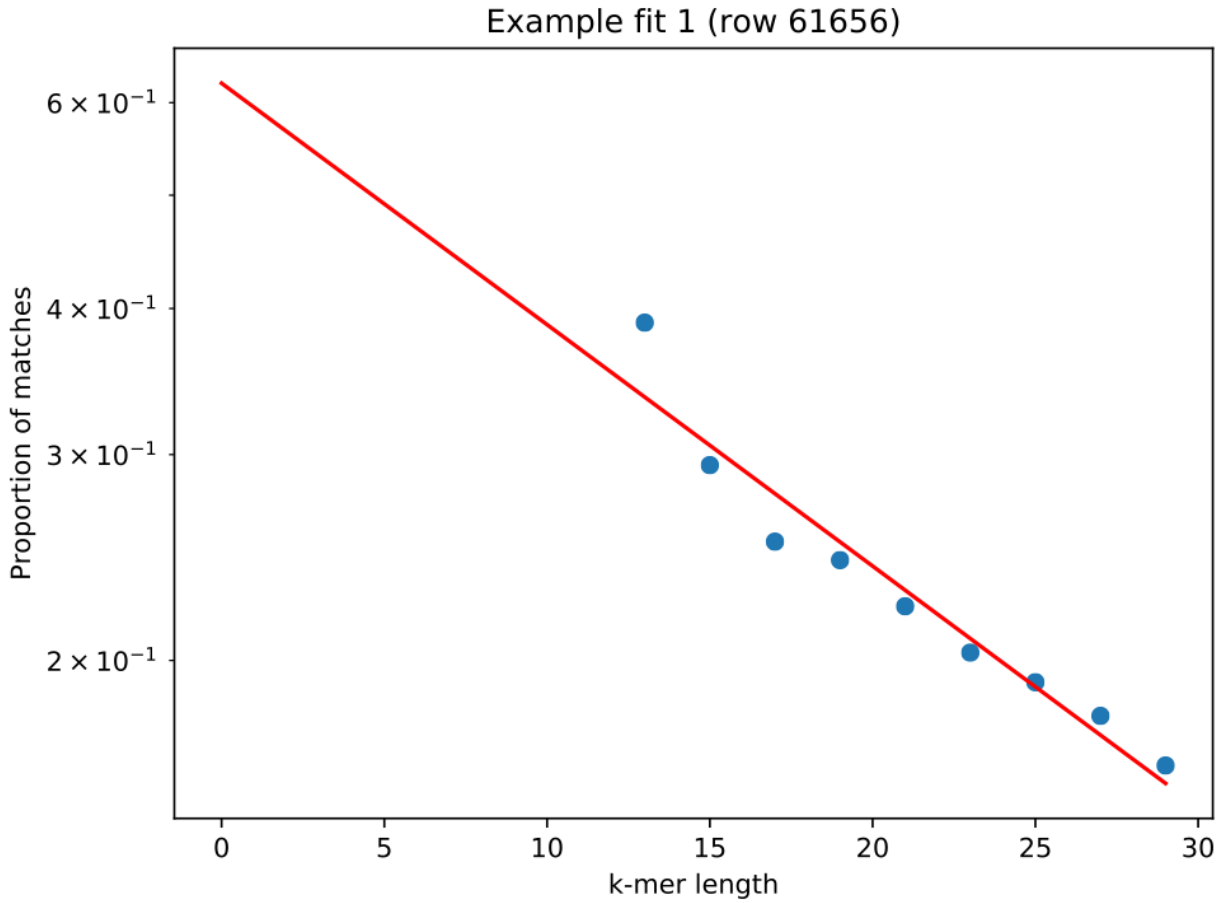
If you want to change cluster names or assign queries to your own cluster definitions you can use the `--external-clustering` argument instead.

5.2 Choosing the right k-mer lengths

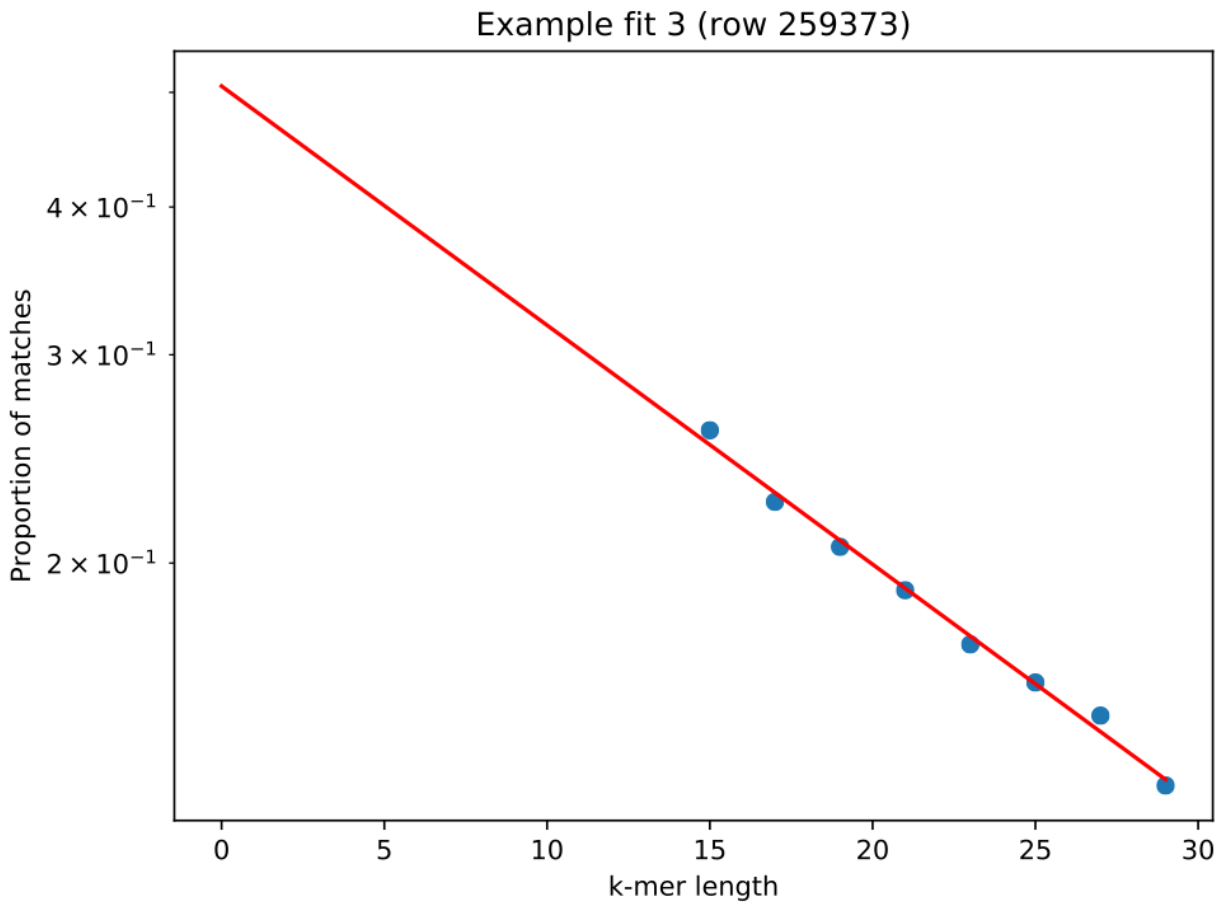
When using in the `--create-db` mode a straight line fit is required. Make sure to run with the `--plot-fit` option, which will randomly choose a number of sample pairs to plot the relation between k-mer distances and core and accessory fits.

To get a sensitive estimate of accessory distance independent from core distance, a small a k-mer size as possible needs to be included in the fit. However, for longer genomes too small a k-mer size will result in biased estimates of distances as small k-mers will match at random.

Here is an example of a fit with `--k-step 2 --min-k 13`:



The genome being fitted is 4.6Mb long, which at 13-mer matches gives a 6% chance of random matches (this information is written to `STDERR`), resulting in the left-most point being over-estimated. Using exactly the same command, but changing `--min-k 15` fixes the issue:



A `--kmer-step` of four is usually sufficient, but drop this to two or three to give the best accuracy.

5.2.1 Using fit refinement when mixture model totally fails

If the mixture model does not give any sort of reasonable fit to the points, you can manually provide a file with `--manual-start` to give the starting parameters to `--refine-fit` mode. The format of this file is as follows:

```
mean0 0,0
mean1 0.5,0.6
start_point 0.3
```

A key, followed by its value (space separated).

`mean0` and `mean1` define the points (x,y) to draw the line between, and `start_point` is the distance along this line to draw the initial boundary (which is normal to the line).

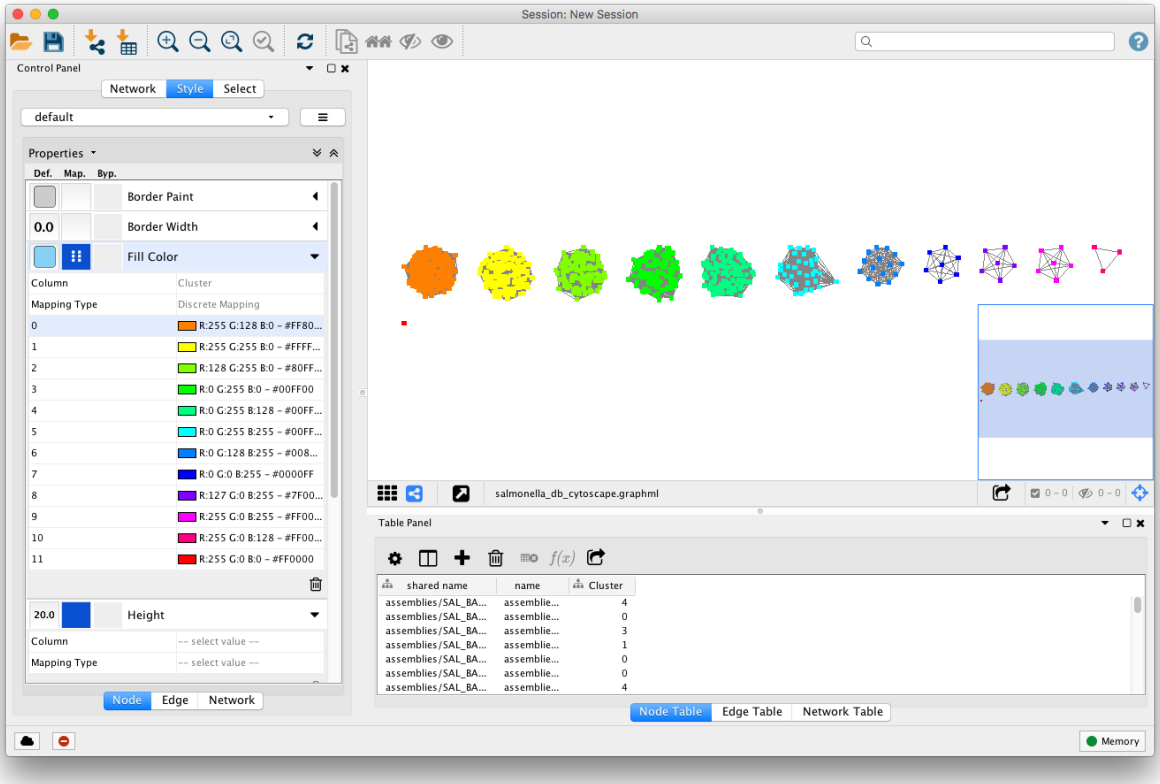
5.3 Viewing the network with cytoscape

If you add the `--cytoscape` option when running `--fit-model` _cytoscape.graphml and _cytoscape.csv files will be written to the output directory.

Open cytoscape and drag and drop the .graphml file onto the window to import the network. Import -> table -> file to load the CSV. Click 'Select None' then add the 'id' column as a key, and any required metadata columns (at least the

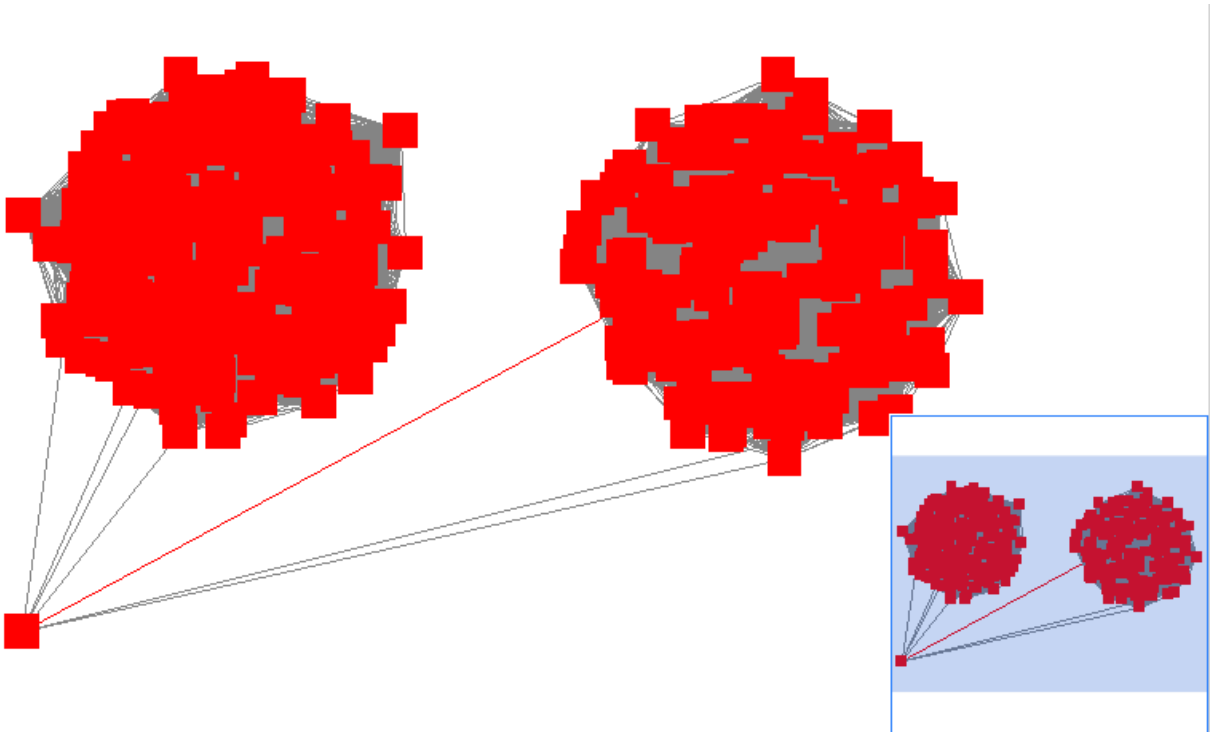
'Cluster' column) as attributes. Make sure 'Node Table Columns' is selected as the data type.

Click on 'Style' and change the node fill colour to be by cluster, the mapping type as discrete, then right click to autogenerate a colour scheme. You can also modify the node size here. In the *Tutorial* example, the components are nicely separated and the network has high transitivity:



In some cases, edges which are between strain links may have been erroneously included in the network. This could be due to poor model fit, or a poor quality sequence. Use Tools -> NetworkAnalyzer -> Analyze Network to compute information for each node and edge. It may help to analyze connected components separately. They can be split under Tools -> NetworkAnalyzer -> Subnetwork Creation.

Here is an example where an errant node is connecting two clusters into one large cluster, which should be split:



The incorrect node in question has a low CluteringCoefficient and high Stress. The EdgeBetweeness of its connections are also high. Sorting the node and edge tables by these columns can find individual problems such as this.

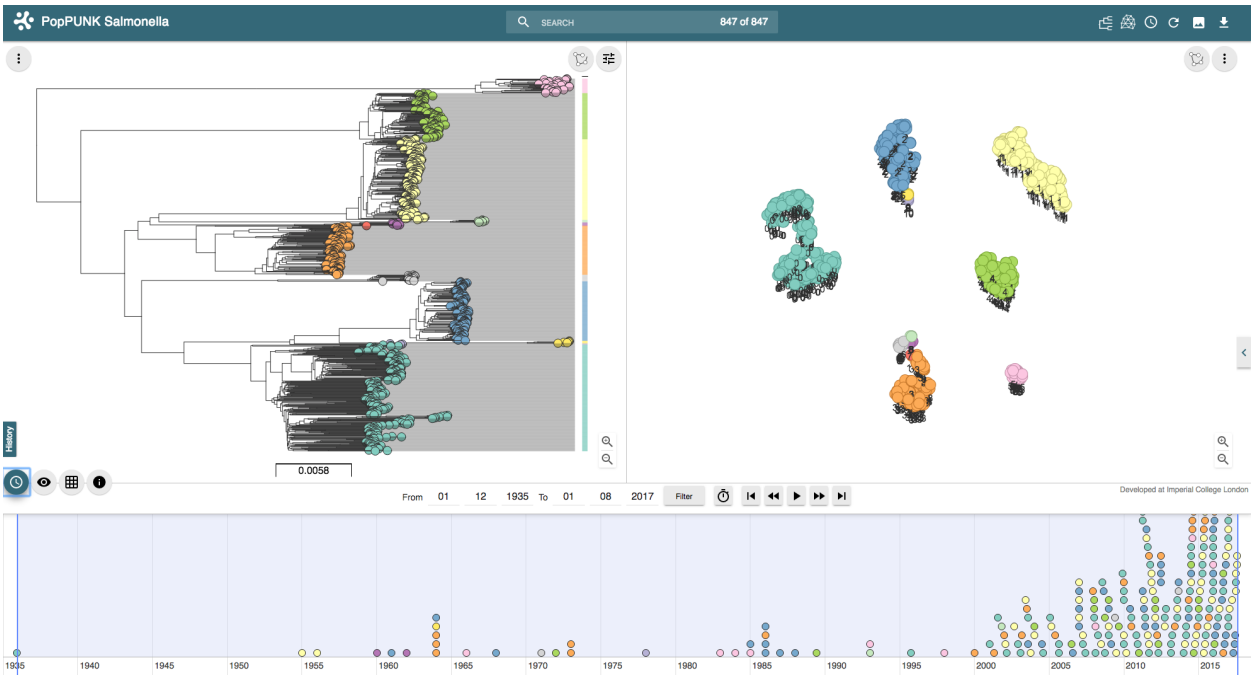
5.4 Setting the perplexity parameter for t-SNE

In t-SNE an embedding of the accessory genome distances is found which represents local structure of the data. Isolates with similar accessory content will visually appear in clusters together.

The perplexity sets a guess about the number of close neighbours each point has, and is a trade-off between local and global structure. t-SNE is reasonably robust to changes in the perplexity parameter (set with `--perplexity` when creating microreact output with `--microreact` in the‘‘–fit-model‘‘ mode), however we would recommend trying a few values to get a good embedding for the accessory distances.

There is a good discussion of the effect of perplexity here and the sklearn documentation shows some examples of the effect of changing perplexity.

In the *Tutorial* example, a perplexity of 30 gives clear clustering of the accessory genome content, condordant with the core genome structure (data):



With a lower perplexity of 5, the clustering is too loose, and the strain structure cannot clearly be seen (data):

30 is a good default, but you may wish to try other values, particularly with larger or smaller datasets. You can re-run the t-SNE using the `poppunk_tsne` command, providing the distances from the previous run:

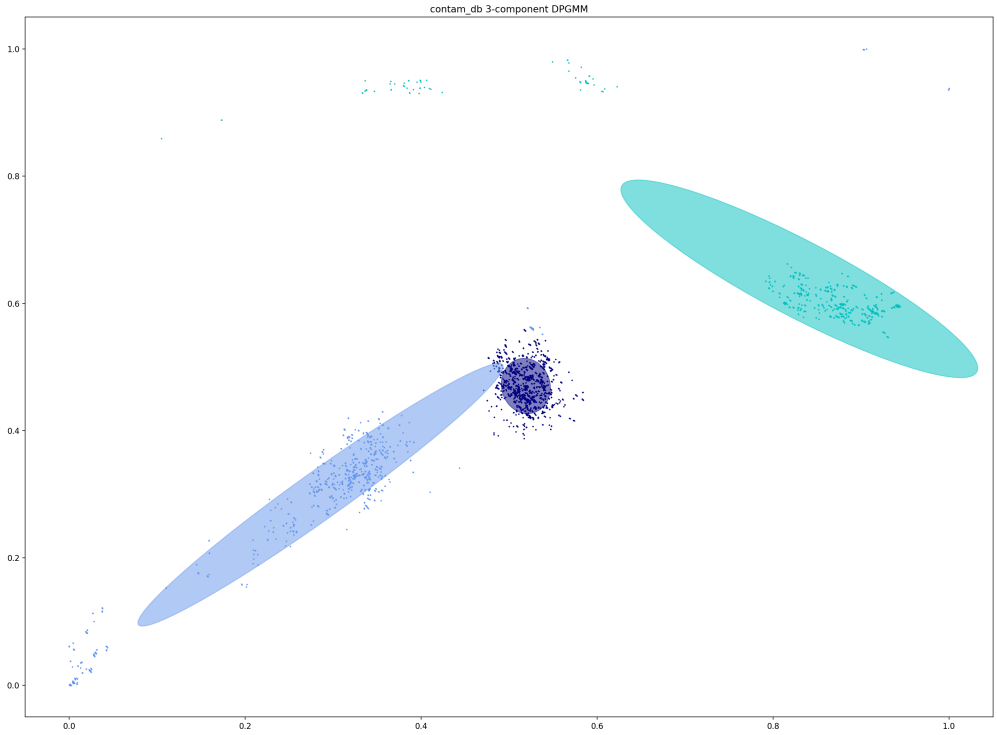
```
poppunk_tsne --distances strain_db/strain_db.dists --output strain_db \
--perplexity 20 --verbosity 1
```

5.5 Dealing with poor quality data

In this example we analyse 76 *Haemophilus influenzae* isolates. One isolate, 14412_4_15, is contaminated with 12% of reads being *Haemophilus parainfluenzae* and a total assembly length of 3.8Mb. It should be removed before input, but its presence can also be found with `PopPUNK`.

5.5.1 With the distances

A fit with three mixture components overestimates the number of between strain links, and gives a network with a poor score (0.6849) and only five components:



Distances in the top left of the plot, with low core distances and high accessory distances, are due to the contaminated contigs in the isolate. Finding which isolates contribute to these distances reveals a clear culprit:

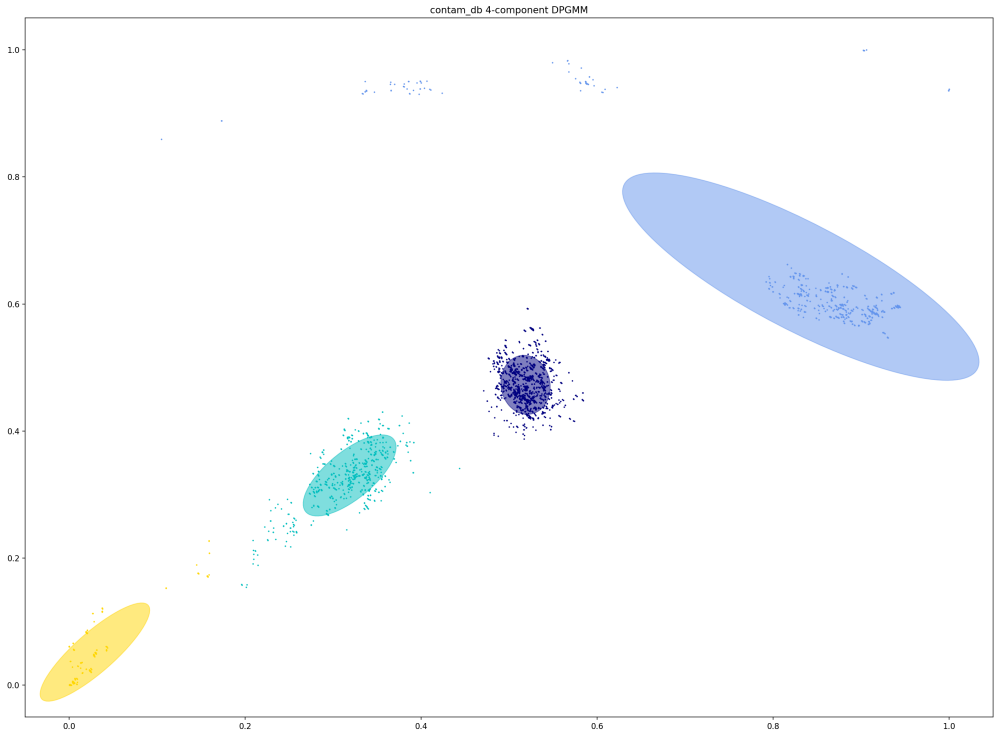
```
awk '$3<0.02 && $4 > 0.3 {print $1}' contam_db/contam_db.search.out | cut -f 1 | sort␣
↪| uniq -c
   1 14412_3_81
   1 14412_3_82
   1 14412_3_83
   1 14412_3_84
   1 14412_3_88
   1 14412_3_89
   1 14412_3_91
   1 14412_3_92
```

(continues on next page)

(continued from previous page)

```
  1 14412_4_1
  1 14412_4_10
 28 14412_4_15
```

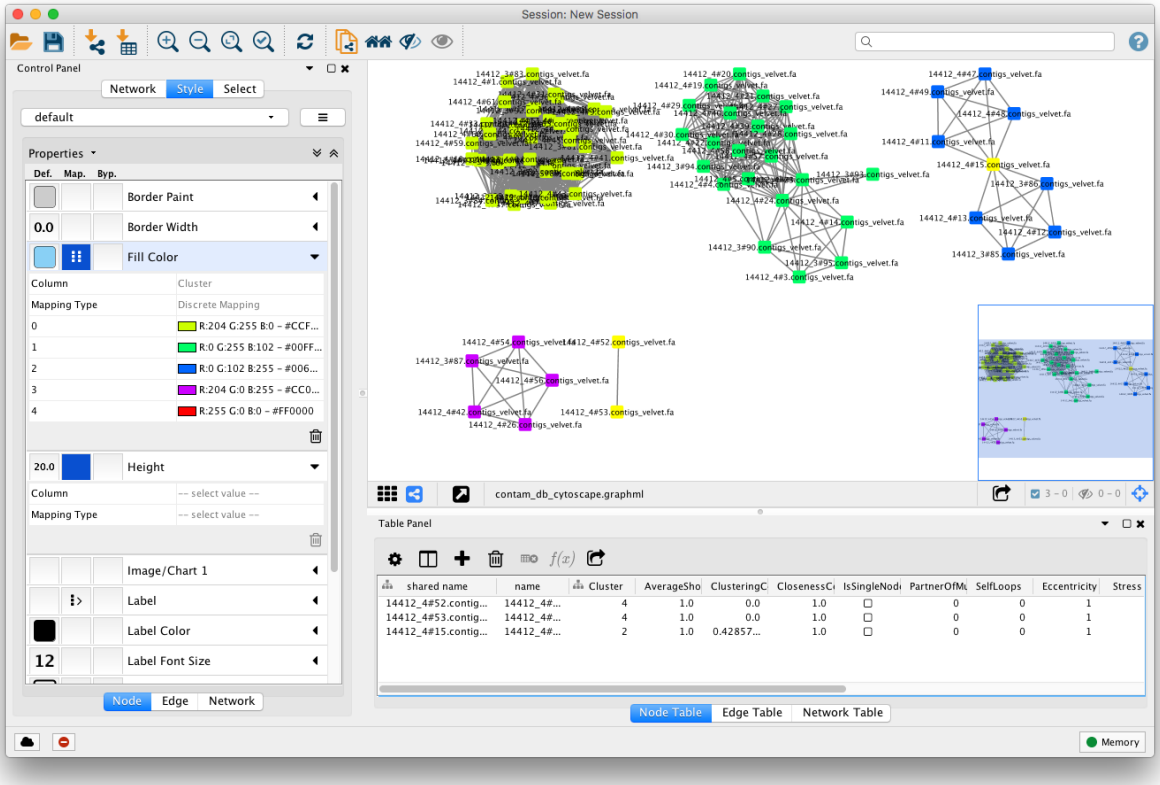
In this case it is sufficent to increase the number of mixture components to four, which no longer includes these inflated distances. This gives a score of 0.9401 and 28 components:



The best thing to do is to remove the poor quality isolate, or if possible remove the contaminated reads/contigs from the assembly.

5.5.2 With the network

Alternatively, the network itself can be inspected with `--cytoscape`. Using the approach detailed in *Viewing the network with cytoscape* gives the following view:



The contaminated node appears when ordering by ClusteringCoefficient, Stress or TopologicalCoefficient, and its edges appear when ordering by EdgeBetweeness. It can be seen highlighted in the top right component, connecting two clusters which otherwise have no links. It can be removed, and components recalculated in cytoscape directly, though removal from the PopPUNK database is best.

The second largest cluster is also suspicious, where there are few triangles (low transitivity) and the nodes involved have high Stress. This is indicative of a bad fit overall, rather than a single problem sample.

5.6 Removing samples from a database

You can use the `prune_poppunk` command to remove samples from a database, for example those found to be of poor quality (see *Dealing with poor quality data*). Create a file `remove.txt` with the names of the samples you wish to remove, one per line, and run:

```
prune_poppunk --remove remove.txt --distances strain_db/strain_db.dists --output␣
↪pruned_db
```

This will remove the samples from the `strain_db.dists` files, from which `--model-fit` can be run again.

If you would like to create the mash sketches again, which is recommended if you plan to use `--full-db` and/or assign future query sequences, add the `--resketch` argument:

```
prune_poppunk --remove remove.txt --distances strain_db/strain_db.dists --output␣
↪pruned_db --resketch --ref-db strain_db --threads 4
```

5.7 Memory/run-time issues

For larger datasets resource use may be challenging. So far the largest dataset we've analysed was around 12000 genomes, which used modest computational resources. Here are some tips based on these experiences:

- Add `--threads` – they are used fairly efficiently throughout.
- When running `--create-db` with many threads, add the `--no-stream` option. This will trade-off memory for disk usage, as it seems that many threaded `mash dist` output cannot be processed as fast as it is produced.
- In `--refine-model` set `--pos-shift 0` to avoid creating huge networks with close to N^2 edges. Mixture models normally need to be pruned.
- In `--refine-model` you may add the `--no-local` option to skip that step and decrease run-time, though gains are likely marginal.
- Use `--rapid-nj`, if producing MicroReact output.

Another option for scaling is to run `--create-db` with a smaller initial set (not using the `--full-db` command), then use `--assign-query` to add to this.

CHAPTER 6

Scripts

Brief documentation on the helper scripts included in the package in the `/scripts` directory. To use these scripts you will need to have a clone of the git repository, or they should also be installed with the prefix 'poppunk' (e.g to run `extract_distances.py`, run the command `poppunk_extract_distances.py`).

- *Writing the pairwise distances to an output file*
- *Writing network components to an output file*
- *Calculating Rand indices*
- *Calculating silhouette indices*

6.1 Writing the pairwise distances to an output file

By default PopPUNK does not write the calculated π_n and a distances out, as this contains $\frac{1}{2}n * (n-1)$ rows, which gives a multi Gb file for large datasets.

However, if needed, there is a script available to extract these distances as a text file:

```
python scripts/extract_distances.py --distances strain_db.dists --output strain_db.
↪dists.out
```

6.2 Writing network components to an output file

Visualisation of large networks with cytoscape may become challenging. It is possible to extract individual components/clusters for visualisation as follows:

```
python scripts/extract_components.py strain_db_graph.gpickle strain_db
```

6.3 Calculating Rand indices

This script allows the clusters formed by different runs/fits/modes of PopPUNK to be compared to each other. 0 indicates the clusterings are totally discordant, and 1 indicates they are identical.

Run:

```
python scripts/calculate_rand_indices.py --input poppunk_gmm_clusters.csv,poppunk_
↪dbscan_cluster.csv
```

The script will calculate the Rand index and the adjusted Rand index between all pairs of files provided (comma separated) to the `--input` argument. These will be written to the file `rand.out`, which can be changed using `--output`.

The `--subset` argument can be used to restrict comparisons to include only specific samples listed in the provided file.

6.4 Calculating silhouette indices

This script can be used to find how well the clusters project into core-accessory space by calculating the silhoutte index, which measures how close samples are to others in their own cluster compared to samples from other clusters. The silhoutte index is calculated for every sample and takes a value between -1 (poorly matched) to +1 (well matched). The script reports the average of these indices across all samples, using Euclidean distances between the (normalised) core and accessory divergences calculated by PopPUNK.

To run:

```
python scripts/calculate_silhouette.py --distances strain_db.dists --cluster-csv␣
↪strain_db_clusters.csv
```

The following additonal options are available for use with external clusterings (e.g. from hierBAPS):

- `--cluster-col` the (1-indexed) column index containing the cluster assignment
- `--id-col` the (1-indexed) column index containing the sample names
- `--sub` a string to remove from sample names to match them to those in `--distances`

CHAPTER 7

Miscellaneous

7.1 Rejected/alternative logos

CHAPTER 8

Details

A full description of the method can be found in the paper.

`PopPUNK` uses the fast k-mer distance estimation enabled by mash to calculate core and accessory distances between all pairs of isolates of bacteria in a collection. By clustering these distances into 'within-strain' and 'between-strain' distances a network of within-strain comparisons can be constructed. The use of a network has a number of convenient properties, the first being that the connected components represent a cluster of strains.

As well as identifying strains, the pairwise distance distribution also helps with assembly quality control (particularly in the case of contaminated contigs) and may be informative of the level of recombination in the population. The network representation also allows definition of representative isolates by sampling one example from each clique, and calculation of various statistics which can show how good the clustering is.

The advantages of this approach are broadly that:

- It is fast, and scalable to 10^4 genomes in a single run.
- Assigning new query sequences to a cluster using an existing database is scalable even beyond this.
- Databases can be updated online (as sequences arrive).
- Online updating is equivalent to building databases from scratch.
- Databases can be kept small and managable by only keeping representative isolates.
- There is no bin cluster. Outlier isolates will be in their own cluster.
- Pre-processing, such as generation of an alignment, is not required.
- The definition of clusters is biologically relevant to how bacteria evolve.
- There is a lot of quantitative and graphical output to assist with clustering.
- A direct import into microreact is available, as well as cytoscape, grapetree and phandango.
- Everything is available within a single python executable.

CHAPTER 9

Citation:

If you find PopPUNK useful, please cite as:

Lees JA, Harris SR, Tonkin-Hill G, Gladstone RA, Lo SW, Weiser JN, Corander J, Bentley SD, Croucher NJ. Fast and flexible bacterial genomic epidemiology with PopPUNK. *Genome Research* **29**:1-13 (2019). doi:10.1101/gr.241455.118

CHAPTER 10

Index:

- genindex
- search