CONCOCT Documentation

Release 1.0.0

Johannes Alneberg, Brynjar Smari Bjarnason, Ino de Bruijn, Mela

Aug 02, 2019

Contents

CONCOCT "bins" metagenomic contigs. Metagenomic binning is the process of clustering sequences into clusters corresponding to operational taxonomic units of some level.

CHAPTER 1

Features

CONCOCT does unsupervised binning of metagenomic contigs by using nucleotide composition - kmer frequencies - and coverage data for multiple samples. CONCOCT can accurately (up to species level) bin metagenomic contigs. For optimal performance:

- Map several samples against your assembled contigs.
- Cut longer contigs into 10 - 20 kb pieces prior to mapping.
- Evaluate your bins using single copy genes.

CHAPTER 2

Installation

For a comprehensive guide on how to install CONCOCT and all its dependencies, see *Installation*.

CHAPTER 3

Contribute

- Issue Tracker: github
- Source Code: github

CHAPTER 4

Support

If you are having issues, please let us know. We have a discussion thread on gitter:

CHAPTER 5

Known Issues

- Contig names consisting of digits only are not allowed. Please rename your contigs in both the fasta and the coverage table before proceeding.
- Contig sequences can only contain letters A,C,G or T. For example Ns are currently not allowed.
- Contigs need to be cut up prior to binning. This is covered in the *Basic Usage* page.

For a more up to date list of reported issues, check the issue tracker: https://github.com/BinPro/CONCOCT/issues

CHAPTER 6

Licence

FreeBSD

CHAPTER 7

Contents:

7.1 Installation

7.1.1 With Bioconda [Recommended]

The easiest and recommended way to install concoct is through Bioconda and conda in an isolated environment:

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

conda create -n concoct_env python=3 concoct
```

Note for Mac OSX users

Currently concoct on Mac OSX can only run in single threaded mode, which drastically increases the runtime. However, the Mac OSX installation of concoct can still be useful for testing purposes and is possible to install through conda as shown above.

7.1.2 Manual Installation

The conda installation should be enough for most users. However, if you want to modify the source code, a manual installation might be needed. An example of a manual installation on an Ubuntu system can be seen in the Travis CI config file.

7.1.3 Using Docker

We provide a Docker image: binpro/concoct_latest which contains CONCOCT and its dependencies for a basic workflow.

Assuming DOcker is installed, the following command will then download the image from the Docker image index, map the Data folder to the image and log you into the docker image.

```
docker run -v /home/USER/Data:/opt/Data -i -t binpro/concoct_latest bash
```

To test concoct you can then do:

```
$ cd /opt/CONCOCT_latest
$ nosetests
```

Which should execute all tests without errors.

7.1.4 Other Programs Needed

- For assembly, use your favourite. Here is a good one:
  - Megahit
- To create the input bam files, choose your favourite aligner, for example bowtie2 or bwa.
- For validation of clustering using single-copy core genes we recommend using:
  - CheckM

7.2 Basic Usage

This guide assumes you have your original contigs assembled into a file `original_contigs.fa` and that you have mapped reads from several samples to these contigs into `.bam` files. Note that the assembly can be constructed using either one single sample or several (usually all) samples. In either case, all sample reads should be mapped against the assembly to achieve the best binning performance.

The next step is then to cut contigs into smaller parts:

```
cut_up_fasta.py original_contigs.fa -c 10000 -o 0 --merge_last -b contigs_10K.bed >␣
↪contigs_10K.fa
```

Generate table with coverage depth information per sample and subcontig. This step assumes the directory 'mapping' contains sorted and indexed bam files where each sample has been mapped against the original contigs:

```
concoct_coverage_table.py contigs_10K.bed mapping/Sample*.sorted.bam > coverage_table.
↪tsv
```

Run concoct:

```
concoct --composition_file contigs_10K.fa --coverage_file coverage_table.tsv -b␣
↪concoct_output/
```

Merge subcontig clustering into original contig clustering:

```
merge_cutup_clustering.py concoct_output/clustering_gt1000.csv > concoct_output/
↪clustering_merged.csv
```

Extract bins as individual FASTA:

```
mkdir concoct_output/fasta_bins
extract_fasta_bins.py original_contigs.fa concoct_output/clustering_merged.csv --
↪output_path concoct_output/fasta_bins
```

(continues on next page)

(continued from previous page)

These bins should now be evaluated and filtered for completeness and contamination using for example CheckM or BUSCO.

7.3 Command Line Options

CONCOCT uses several command line options to control the clustering, here is a complete documentation of these. These can also be viewed by typing `concoct -h` on the command line:

```
usage: - [-h] [--coverage_file COVERAGE_FILE]
         [--composition_file COMPOSITION_FILE] [-c CLUSTERS] [-k KMER_LENGTH]
         [-t THREADS] [-l LENGTH_THRESHOLD] [-r READ_LENGTH]
         [--total_percentage_pca TOTAL_PERCENTAGE_PCA] [-b BASENAME] [-s SEED]
         [-i ITERATIONS] [--no_cov_normalization] [--no_total_coverage]
         [--no_original_data] [-o] [-d] [-v]

optional arguments:
  -h, --help            show this help message and exit
  --coverage_file COVERAGE_FILE
                        specify the coverage file, containing a table where
                        each row correspond to a contig, and each column
                        correspond to a sample. The values are the average
                        coverage for this contig in that sample. All values
                        are separated with tabs.
  --composition_file COMPOSITION_FILE
                        specify the composition file, containing sequences in
                        fasta format. It is named the composition file since
                        it is used to calculate the kmer composition (the
                        genomic signature) of each contig.
  -c CLUSTERS, --clusters CLUSTERS
                        specify maximal number of clusters for VGMM, default
                        400.
  -k KMER_LENGTH, --kmer_length KMER_LENGTH
                        specify kmer length, default 4.
  -t THREADS, --threads THREADS
                        Number of threads to use
  -l LENGTH_THRESHOLD, --length_threshold LENGTH_THRESHOLD
                        specify the sequence length threshold, contigs shorter
                        than this value will not be included. Defaults to
                        1000.
  -r READ_LENGTH, --read_length READ_LENGTH
                        specify read length for coverage, default 100
  --total_percentage_pca TOTAL_PERCENTAGE_PCA
                        The percentage of variance explained by the principal
                        components for the combined data.
  -b BASENAME, --basename BASENAME
                        Specify the basename for files or directory where
                        outputwill be placed. Path to existing directory or
                        basenamewith a trailing '/' will be interpreted as a
                        directory.If not provided, current directory will be
                        used.
  -s SEED, --seed SEED  Specify an integer to use as seed for clustering. 0
                        gives a random seed, 1 is the default seed and any
                        other positive integer can be used. Other values give
```

(continues on next page)

(continued from previous page)

```
                      ArgumentTypeError.
-i ITERATIONS, --iterations ITERATIONS
                      Specify maximum number of iterations for the VBGMM.
                      Default value is 500
--no_cov_normalization
                      By default the coverage is normalized with regards to
                      samples, then normalized with regards of contigs and
                      finally log transformed. By setting this flag you skip
                      the normalization and only do log transorm of the
                      coverage.
--no_total_coverage   By default, the total coverage is added as a new
                      column in the coverage data matrix, independently of
                      coverage normalization but previous to log
                      transformation. Use this tag to escape this behaviour.
--no_original_data    By default the original data is saved to disk. For big
                      datasets, especially when a large k is used for
                      compositional data, this file can become very large.
                      Use this tag if you don't want to save the original
                      data.
-o, --converge_out    Write convergence info to files.
-d, --debug           Debug parameters.
-v, --version         show program's version number and exit
```

7.4 CONCOCT Scripts

CONCOCT ships with some additional scripts which are very useful to e.g. create input files and to extract output fastas for concoct. These scripts are:

- `cut_up_fasta.py`
- `concoct_coverage_table.py`
- `merge_cutup_clustering.py`
- `extract_fasta_bins.py`

The repository CONCOCT contains additional scripts in the `CONCOCT/scripts` directory which are not fully maintained. They implement methods that we apply after binning with CONCOCT and it might be useful as a starting point or inspiration when creating your own scripts for downstream processing of the output files. Out of these scripts, the ones documented here are:

- `dnadiff_dist_matrix.py`
- `extract_scg_bins.py` [Deprecated]

Contents:

7.4.1 cut_up_fasta.py

Usage

The usage and help documentation of `cut_up_fasta.py` can be seen by running `cut_up_fasta.py -h`:

```
usage: - [-h] [-c CHUNK_SIZE] [-o OVERLAP_SIZE] [-m] [-b BEDFILE]
         contigs [contigs ...]

Cut up fasta file in non-overlapping or overlapping parts of equal length.

Optionally creates a BED-file where the cutup contigs are specified in terms
of the original contigs. This can be used as input to concoct_coverage_table.py.

positional arguments:
  contigs               Fasta files with contigs

optional arguments:
  -h, --help            show this help message and exit
  -c CHUNK_SIZE, --chunk_size CHUNK_SIZE
                        Chunk size
  -o OVERLAP_SIZE, --overlap_size OVERLAP_SIZE
                        Overlap size
  -m, --merge_last      Concatenate final part to last contig
  -b BEDFILE, --bedfile BEDFILE
                        BEDfile to be created with exact regions of the
                        original contigs corresponding to the newly created
                        contigs
```

Example

An example of how to run `cut_up_fasta.py`:

```
cut_up_fasta.py original_contigs.fa -c 10000 -o 0 --merge_last -b contigs_10K.bed >␣
↪contigs_10K.fa
```

This creates a fasta file and a BED file. The fasta file `contigs_10K.fa` contains the original contigs cut up into parts of length exactly 10K, except for the last contig part which is between 10K and 20K long. The BED file `contigs_10K.bed` contains a list of the contig parts created with coordinates in the original contigs.

7.4.2 concoct_coverage_table.py

Usage

The usage and help documentation of `concoct_coverage_table.py` can be seen by running `concoct_coverage_table.py -h`:

```
usage: - [-h] [--samplenames SAMPLENAMES] bedfile bamfiles [bamfiles ...]

A script to generate the input coverage table for CONCOCT using a BEDFile.
Output is written to stdout. The BEDFile defines the regions used as
subcontigs for concoct. This makes it possible to get the coverage for
subcontigs without specifically mapping reads against the subcontigs. @author:
inodb, alneberg

positional arguments:
  bedfile               Contigs BEDFile with four columns representing:
                        'Contig ID, Start Position, End Position and SubContig
                        ID' respectively. The Subcontig ID must contain the
                        pattern 'concoct_part_[0-9]*' while the contigs which
```

(continues on next page)

(continued from previous page)

```
                        are not cutup cannot contain this pattern. This file
                        can be generated by the cut_up_fasta.py script.
  bamfiles              BAM files with mappings to the original contigs.

optional arguments:
  -h, --help            show this help message and exit
  --samplenames SAMPLENAMES
                        File with sample names, one line each. Should be same
                        nr of bamfiles. Default sample names used are the file
                        names of the bamfiles, excluding the file extension.
```

Example

An example of how to run `concoct_coverage_table.py`:

```
concoct_coverage_table.py contigs_10K.bed mapping/Sample*.sorted.bam > coverage_table.
↪tsv
```

This creates a coverage table suitable as input for concoct as the *coverage_file* parameter. The `contigs_10K.bed` file is created from the `cut_up_fasta.py` script and the `bam`-files needs to be sorted and indexed.

7.4.3 merge_cutup_clustering.py

Usage

The usage and help documentation of `merge_cutup_clustering.py` can be seen by running `merge_cutup_clustering.py -h`:

```
usage: - [-h] cutup_clustering_result

With contigs cutup with cut_up_fasta.py as input, sees to that the consequtive
parts of the original contigs are merged. prints result to stdout. @author:
alneberg

positional arguments:
  cutup_clustering_result
                        Input cutup clustering result.

optional arguments:
  -h, --help            show this help message and exit
```

Example

An example of how to run `merge_cutup_clustering.py`:

```
merge_cutup_clustering.py concoct_output/clustering_gt1000.csv > concoct_output/
↪clustering_merged.csv
```

This merges the clustering `clustering_gt1000.csv` created by concoct by looking at cluster assignments per contig part and assigning a concensus cluster for the original contig. The output clustering_merged.csv contains a header line and contig_id and cluster_id per line, separated by a comma.

7.4.4 extract_fasta_bins.py

Usage

The usage and help documentation of `extract_fasta_bins.py` can be seen by running `extract_fasta_bins.py -h`:

```
usage: - [-h] [--output_path OUTPUT_PATH] fasta_file cluster_file

extract_fasta_bins.py Extract a fasta file for each cluster from a concoct
result file.

positional arguments:
  fasta_file            Input Fasta file.
  cluster_file          Concoct output cluster file

optional arguments:
  -h, --help            show this help message and exit
  --output_path OUTPUT_PATH
                        Directory where files will be printed
```

Example

An example of how to run `extract_fasta_bins.py`:

```
mkdir concoct_output/fasta_bins
extract_fasta_bins.py original_contigs.fa concoct_output/clustering_merged.csv --
↪output_path concoct_output/fasta_bins
```

This creates a fasta file for each cluster assigned by concoct. The clusters assigned need not to be complete or uncontaminated and should be investigated closer with e.g. CheckM.

7.4.5 dnadiff_dist_matrix.py

Usage

The usage and help documentation of `dnadiff_dist_matrix.py` can be seen by running `pyhton dnadiff_dist_matrix -h`:

```
usage: - [-h] [--min_coverage MIN_COVERAGE] [--fasta_names FASTA_NAMES]
         [--plot_image_extension PLOT_IMAGE_EXTENSION] [--skip_dnadiff]
         [--skip_matrix] [--skip_plot] [--cluster-threshold CLUSTER_THRESHOLD]
         output_folder fasta_files [fasta_files ...]

Output distance matrix between fasta files using dnadiff from MUMmer. Generates
dnadiff output files in folders:

output_folder/fastaname1_vs_fastaname2/
output_folder/fastaname1_vs_fastaname3/

etc

where fastaname for each fasta file can be supplied as an option to the script.
Otherwise they are just counted from 0 to len(fastafiles)
```

(continues on next page)

(continued from previous page)

```
The distance between each bin is computed using the 1-to-1 alignments of the
report files (not M-to-M):

1 - AvgIdentity if min(AlignedBases) >= min_coverage. Otherwise distance is 1.
Or 0 to itself.

Resulting matrix is printed to stdout and to output_folder/dist_matrix.tsv. The
rows and columns of the matrix follow the order of the supplied fasta files. The
names given to each fasta file are also outputted to the file
output_folder/fasta_names.tsv

A hierarchical clustering of the distance using euclidean average linkage
clustering is plotted. This can be deactivated by using --skip_plot. The
resulting heatmap is in output_folder/hclust_heatmap.pdf or
output_folder/hclust_dendrogram.pdf and the resulting clustering is presented
in output_folder/clustering.tsv. The image extension can be changed.

positional arguments:
  output_folder         Output folder
  fasta_files           fasta files to compare pairwise using MUMmer's dnadiff

optional arguments:
  -h, --help            show this help message and exit
  --min_coverage MIN_COVERAGE
                        Minimum coverage of bin in percentage to calculate
                        distance otherwise distance is 1. Default is 50.
  --fasta_names FASTA_NAMES
                        File with names for fasta file, one line each. Could
                        be sample names, bin names, genome names, whatever you
                        want. The names are used when storing the MUMmer
                        dnadiff results as in
                        output_folder/fastaname1_vs_fastaname2/. The names are
                        also used for the plots.
  --plot_image_extension PLOT_IMAGE_EXTENSION
                        Type of image to plotted e.g. pdf, png, svg.
  --skip_dnadiff        Skips running MUMmer and uses output_folder as given
                        input to calculate the distance matrix. Expects
                        dnadiff output as
                        output_folder/fastaname1_vs_fastaname2/out.report
  --skip_matrix         Skips Calculating the distance matrix.
  --skip_plot           Skips plotting the distance matrix. By default the
                        distance matrix is clustered hierarchically using
                        euclidean average linkage clustering. This step
                        requires seaborn and scipy.
  --cluster-threshold CLUSTER_THRESHOLD
                        The maximum within cluster distance allowed.
```

Example

An example of how to run `dnadiff_dist_matrix` on the test data:

```
cd CONCOCT/scripts
python dnadiff_dist_matrix.py test_dnadiff_out tests/test_data/bins/sample*.fa
```

This results in the following output files in the folder `test_dnadiff_out/`:

- `dist_matrix.stv` The distance matrix
- `fasta_names.tsv` The names given to each bin (or fasta file)
- `clustering.tsv` This file will give a cluster assignment for each bin (or fasta file)
- `hcust_dendrogram.pdf` Dendrogram of the clustering (click for example)
- `hcust_heatmap.pdf` Heatmap of the clustering (click for example)

Then there is also for each pairwise `dnadiff` alignment the following output files in a subfolder `fastaname1_vs_fastaname2/`:

```
out.1coords
out.1delta
out.cmd
out.delta
out.mcoords
out.mdelta
out.qdiff
out.rdiff
out.report
out.snps
out.unqry
out.unref
```

See MUMmer's own manual for an explanation of each file with `dnadiff --help`.

7.4.6 [Deprecated] extract_scg_bins.py

Usage

The usage and help documentation of `extract_scg_bins.py` can be seen by running `pyhton extract_scg_bins -h`:

```
usage: - [-h] --output_folder OUTPUT_FOLDER --scg_tsvs SCG_TSVS [SCG_TSVS ...]
         --fasta_files FASTA_FILES [FASTA_FILES ...] --names NAMES [NAMES ...]
         [--groups GROUPS [GROUPS ...]] [--max_missing_scg MAX_MISSING_SCG]
         [--max_multicopy_scg MAX_MULTICOPY_SCG]

Extract bins with given SCG (Single Copy genes) criteria. Criteria can be set
as a combination of the maximum number of missing SCGs and the maximum number
of multicopy SCGs. By default the script selects from pairs of scg_tsvs and
fasta_files, the pair that has the highest number of approved bins. In case
there are multiple with the max amount of approved bins, it takes the one that
has the highest sum of bases in those bins. If that is the same, it selects the
first one passed as argument.

One can also group the pairs of scg_tsvs and fasta_files with the --groups
option so one can for instance find the best binning per sample.

optional arguments:
  -h, --help            show this help message and exit
  --output_folder OUTPUT_FOLDER
                        Output folder
  --scg_tsvs SCG_TSVS [SCG_TSVS ...]
                        Single Copy Genes (SCG) tsvs as outpututted by
                        COG_table.py. Should have the same ordering as
```

(continues on next page)

(continued from previous page)

```
                        fasta_files.
  --fasta_files FASTA_FILES [FASTA_FILES ...]
                        Fasta files. Should have the same ordering as scg_tsvs
  --names NAMES [NAMES ...]
                        Names for each scg_tsv and fasta_file pair. This is
                        used as the prefix for the outputted bins.
  --groups GROUPS [GROUPS ...]
                        Select the best candidate for each group of scg_tsv
                        and fasta_file pairs. Number of group names given
                        should be equal to the number of scg_tsv and
                        fasta_file pairs. Identical group names indicate same
                        groups.
  --max_missing_scg MAX_MISSING_SCG
  --max_multicopy_scg MAX_MULTICOPY_SCG
```

Example

An example of how to run `extract_scg_bins` on the test data:

```
cd CONCOCT/scripts/tests/test_data
python extract_scg_bins.py \
    --output_folder test_extract_scg_bins_out \
    --scg_tsvs tests/test_data/scg_bins/sample0_gt300_scg.tsv \
               tests/test_data/scg_bins/sample0_gt500_scg.tsv \
    --fasta_files tests/test_data/scg_bins/sample0_gt300.fa \
                  tests/test_data/scg_bins/sample0_gt500.fa \
    --names sample0_gt300 sample0_gt500 \
    --max_missing_scg 2 --max_multicopy_scg 4 \
    --groups gt300 gt500
```

This results in the following output files in the folder `test_extraxt_scg_bins_out/`:

```
$ ls test_extract_scg_bins_out/
sample0_gt300_bin2.fa  sample0_gt500_bin2.fa
```

Only bin2 satisfies the given criteria for both binnings. If we want to get the best binning of the two, one can remove the `--groups` parameter (or give them the same group id). That would only output `sample0_gt500_bin2.fa`, because the sum of bases in the approved bins of `sample0_gt500` is higher than that of `sample0_gt300`.

CHAPTER 8

Indices and tables

- genindex
- modindex
- search