CONCOCT Documentation

Release 1.0.0

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

December 12, 2018

Contents

1 Features 3

2 Installation 5

3 Contribute 7

4 Support 9

5 Licence 11

6 Contents: 13
- 6.1 Installation . . . 13
- 6.2 Usage . . . 15
- 6.3 Complete Example V1.0 . . . 15
- 6.4 CONCOCT Scripts . . . 15

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

For any known issues with CONCOCT check the issue tracker: https://github.com/BinPro/CONCOCT/issues

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 mailing list located at: concoct-support@lists.sourceforge.net which you can subscribe to here.

CHAPTER 5

Licence

FreeBSD

CHAPTER 6

Contents:

Installation

Dependencies

Fundamental dependencies

```
python version 2.7 or  version 3
gcc - C compiler
gsl - GNU Scientific Library
gslcblas - GNU Scientific Library BLAS library
gomp - GNU OpenMP implementation
```

These items are prerequisities for the installation of concoct as described below. The installation procedure varies on different systems, and described in this README is only how to proceed with a linux (ubuntu) distribution.

We recommend using miniconda to install python. A c-compiler, e.g. `gcc`, is needed to compile the c parts of concoct that uses the GNU Scientific Library `gsl`. For linux (ubuntu) this is installed through:

```
apt-get install build-essential libgsl0-dev libgomp1
```

Making it work on Mac OSX

A bit of a hack. You have been warned:

```
conda install llvm gcc libgcc pip
export CC=/Users/johannes.alneberg/miniconda3/envs/concoct_py3/bin/gcc
export CXX=/Users/johannes.alneberg/miniconda3/envs/concoct_py3/bin/g++
conda install gsl
pip install -r requirements.txt
unset CC
unset CXX
pip install pysam
```

Python packages

```
cython>=0.19.2
numpy>=1.7.1
scipy>=0.12.0
```

```
pandas>=0.11.0
biopython>=1.62b
scikit-learn>=0.13.1
```

These are the python packages that need to be installed in order to run concoct. If you follow the installation instructions below, these will be installed automatically, but are listed here for transparency.

Optional dependencies

- For assembly, use your favorite, here is one
  - Megahit
- To create the input table (containing average coverage per sample and contig)
  - BEDTools version >= 2.15.0 (only genomeCoverageBed)
  - Picard tools version >= 1.110
  - samtools version >= 0.1.18
  - bowtie2 version >= 2.1.0
  - GNU parallel version >= 20130422
  - Python packages: `pysam>=0.6`
- For validation of clustering using single-copy core genes we recommend using:
  - CheckM

If you want to install these dependencies on your own server, you can take a look at doc/Dockerfile.all_dep for ideas on how to install them.

Installation

Here we describe two recommended ways of getting concoct to run on your computer/server. The first option, using Anaconda, should work for any *nix (e.g. Mac OS X or Linux) system even where you do not have 'sudo' rights (e.g. on a common computer cluster). The second option is suitable for a linux computer where you have root privileges and you prefer to use a virtual machine where all dependencies to run concoct are included. Docker does also run on Mac OS X through a virtual machine. For more information check out the Docker documentation.

Using Anaconda

This instruction shows how to install all dependencies (except the 'Fundamental dependencies' and the 'Optional dependencies' listed above) using an Anaconda environment. Anaconda is a tool to isolate your python installation, which allows you to have multiple parallel installations using different versions of different packages, and gives you a very convenient and fast way to install the most common scientific python packages. Anaconda is free but not open source, you can download Anaconda here. Installation instructions can be found here.

After installing Anaconda, create a new environment that will contain the concoct installation:

```
conda create -n concoct_env python=2.7
```

After choosing to proceed, run the suggested command:

```
source activate concoct_env
```

then install the concoct dependencies into this environment:

```
conda install cython numpy scipy biopython pandas pip scikit-learn
```

Finally, download the CONCOCT distribution from https://github.com/BinPro/CONCOCT/releases (stable) and extract the files, or clone the repository with github (potentially unstable). Resolve all dependencies, see above and then execute within the CONCOCT directory:

```
python setup.py install
```

Using Docker

If you have root access to a machine where you want to install concoct and you have storage for roughly 2G "virtual machine" then Docker provides a very nice way to get a Docker image with concoct and its dependencies installed. This way the only thing you install on your host system is Docker, the rest is contained in an Docker image. This allows you to install and run programs in that image without it affecting your host system. You should get to know Docker here. You need to get Docker installed and specially if you have Ubuntu. When Docker is installed you need to download and log into the concoct image.

We provide a Docker image:

binpro/concoct_latest contains CONCOCT and all its dependencies for the Complete Example V1.0 with the exception of the SCG evaluation.

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.

```
sudo 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.

Usage

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:

```
  File "<stdin>", line 2
SyntaxError: from __future__ imports must occur at the beginning of the file
```

Complete Example V1.0

We'd like to here give you a complete example walk through. However, the examples that were here previously were so outdated that they were directly unhelpful. Hopefully a new version of this page will appear here within a not so distant future.

CONCOCT Scripts

The scripts in the `CONCOCT/scripts` directory are not fully maintained. They implement methods that we apply after binning with CONCOCT. Eventually some of these methods might make it to a package of their own.

To test all scripts that have tests one could do:

```
cd CONCOCT/scripts/tests
nosetests
```

Before using a script it would be good to check if its test (in case it has one) is working for you:

```
cd CONCOCT/scripts/tests
nosetests -s test_script_name
```

Contents:

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)

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`.

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
                        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`.