gcMapExplorer Documentation

Release 1.0.25

Rajendra Kumar

Jul 30, 2018

Contents

It is a platform to visualize and analyze the contact maps that are generated from Hi-C experiments. This package is developed by considering the huge size of contact maps at very fine resolution. It contains

- *Graphical User Interface Applications* - Several windows like applications to perform tasks.
- **Command Line Interface - Several commands to perform tasks.**
  - *Commands to import Hi-C data*
  - *Commands to convert bigWig/wig/bed to h5*
  - *Commands to normalize Hi-C map*
- Application Programming Interface - It can be used to perform analysis by any mathematical operations through programming.

For Discussion and Questions, visit this forum

CHAPTER 1

Features

- Support for **huge contact maps** - Use of Disk instead of RAM - Matrices/arrays are stored in Disks - mathematical operations by directly reading/writing from/to Disks, **without loading them into RAM**
- A browser with rich interfaces for **Comparative** and **Interactive** visualization of **two dimensional contact maps** along with **genomic datasets** such as produced by DNase-seq, ChIP-seq, RNA-seq etc.
- Contact maps can be **zoomed in/out** from finest resolution to whole chromosome level.
- Rich customizations of **color scale for contact maps** visualization
- Rich customizations of **X- and Y- axis properties**.
- **Normalization of contact maps by**
  - **Iterative Correction** (IC)
  - **Knight-Ruiz Matrix Balancing** (KR)
  - **Vanilla-Coverage** (VC)
  - **Distance-Frequency**
- A **new file format** based on HDF5 for genome contact map and genomic track datasets.
  - **Portable**, **platform independent** and can be read through C/C++, JAVA, Python and R programming language.
  - **Very fast to read** - fast browsing of contact maps and genomic datasets
- Another file format for chromosomal contact map - much faster than above format to read/write but not compact. Suitable for performing calculations.
- A GUI interface and commands to convert Coordinate Sparse, Pair Coordinate Sparse, HOMER Interaction matrix, Bin-Contact formats into the new gcmap and ccmap formats.
- Interface and commands to convert bigWig/wig/bed file to genomic track dataset h5 file.
- Interface and commands for contact map Normalizations.
- Publication ready images at one click.

CHAPTER 2

Citation

Rajendra Kumar, Haitham Sobhy, Per Stenberg and Ludvig Lizana. Genome Contact Map Explorer - A platform for the comparison, interactive visualization and analysis of genome contact maps. *Nucleic Acids Res.* (2017).

CHAPTER 3

Screen-shots

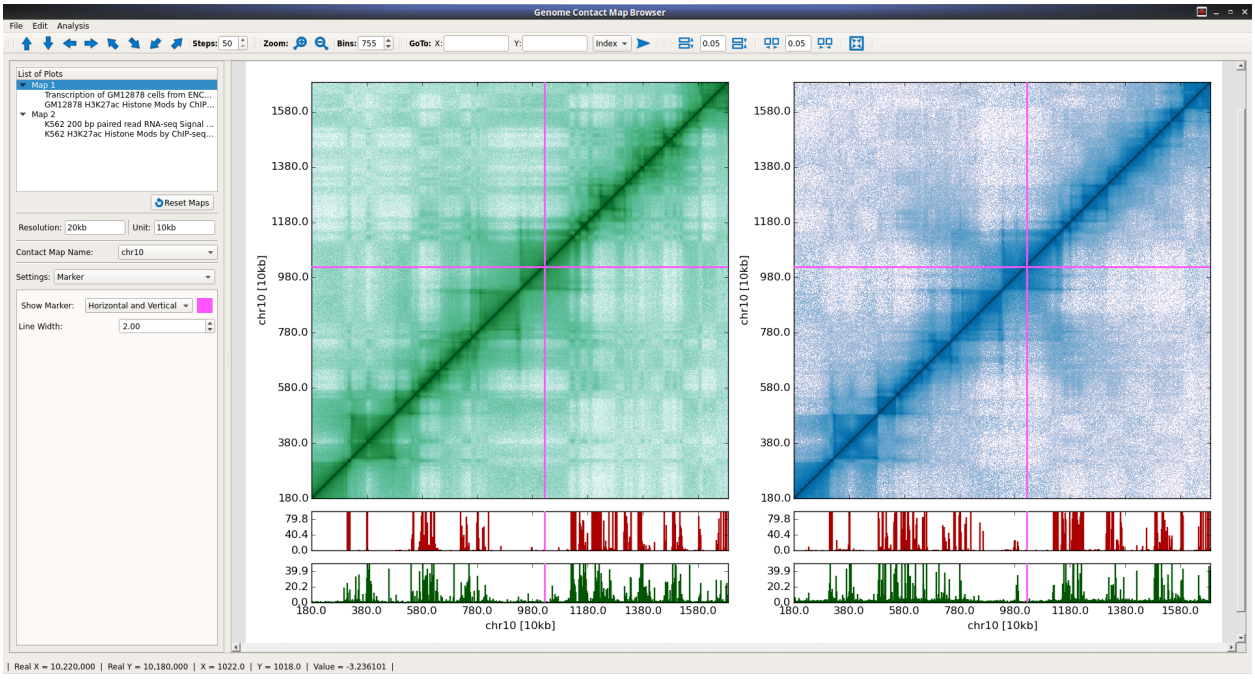


Fig. 1: Genome Contact Map browser

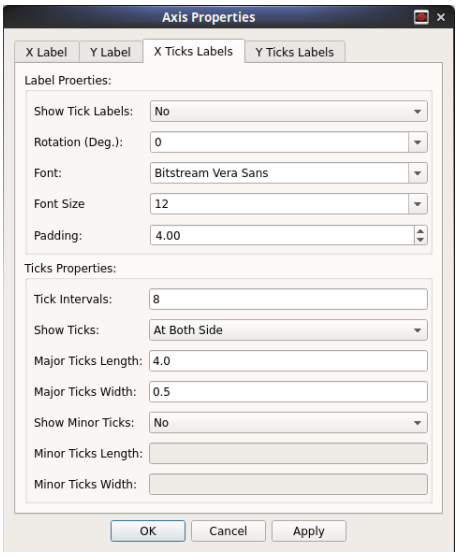


Fig. 2: Axis Properties interface in Browser

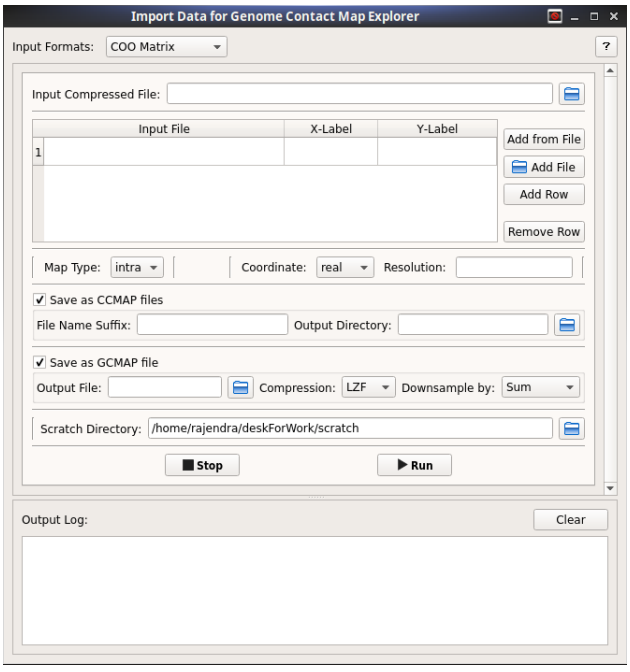


Fig. 3: gcmap Importer Interface

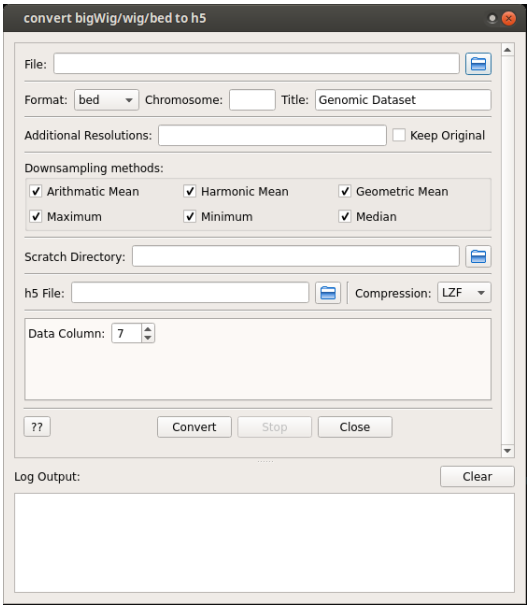


Fig. 4: genomic track dataset converter Interface

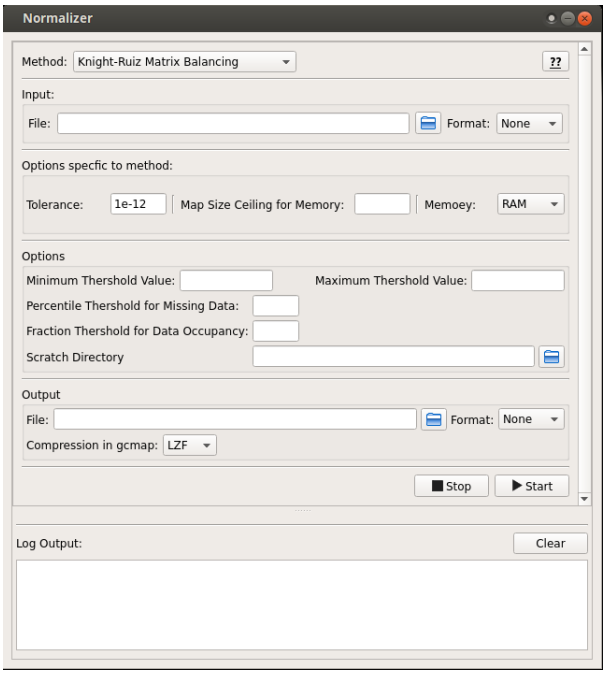


Fig. 5: Contact map normalization Interface

3.1 Contents

3.1.1 Requirements and Installation

Requirements

gcMapExplorer is written in Python3, therefore, it **requires Python3** for installation. It also requires several external Python packages.

Package Required during installation: It has to be installed before gcMapExplorer installation.

-

Package required after installation: These packages are installed automatically during gcMapExplorer installation.

-
-
-
-
-

Package required to install manually:

- - It needs to be installed manually. In case of **Python-3.5**, it can be installed automatically from PyPI.

Installation Steps on Linux

1. Python3 is available through package managers such as **yum** (Fedora, CentOs), **YaST** (OpenSuse) and **apt-get** (Ubuntu, Linux Mint). For example on ubuntu: run `sudo apt-get install python3` command to install Python3.
2. Install Cython by `pip3 install Cython` command.
3. Similar to Python3, PyQt5 is available through package managers. For example on ubuntu: run `sudo apt-get install python3-pyqt5` command to install Python3.
4. Install **gcMapExplorer** by `pip3 install gcMapExplorer` command.

Installation Steps on MacOS

1. Python3 is available through package manager. After installing Homebrew, run `brew install python3` command to install Python3.
2. Install Cython by `pip3 install Cython` command.
3. Similar to Python3, PyQt5 is available through . Run `brew install pyqt5 --with-python3` command to install pyqt5.
4. Install **gcMapExplorer** by `pip3 install gcMapExplorer` command.

Installation Steps on Windows OS

1. Download and install . Note that WinPython should include PyQt5.
2. Open WinPython directory (Default is in C:/ drive) and click on **"WinPython Command Prompt"**. It will open a command prompt terminal.
3. Run `pip install gcMapExplorer` in command prompt terminal to install **gcMapExplorer**

Note: To execute gcMapExplorer command, simple command prompt terminal (from Start Menu) might not work. Use **"WinPython Command Prompt"** present in WinPython directory to launch or execute gcMapExplorer.

Updating gcMapExplorer

To update the gcMapExplorer package use following command:

```
pip install --upgrade --no-deps gcMapExplorer
```

OR

```
pip3 install --upgrade --no-deps gcMapExplorer
```

`--upgrade` flag is used to update the package and `--no-deps` prevents update of dependent packages like numpy, scipy, matplotlib etc.

Configuring gcMapExplorer

Presently two types of global options are required.

- Scratch Directory to dump intermediate temporary files
- Location to `bigWigInfo` and `bigWigToWig` tool

These options are stored in a configuration file. During first use of gcMapExplorer, Scratch Directory is set to by default temporary directory depending on OS. Location to external tools can be set by either manually opening and editing configuration file or through `gcMapExplorer` Python module (see below).

For quick look to configuration file and cleaning scratch directory, `gcMapExplorer config` command can be used:

config

Description: To print configuration file and clean scratch directory

This can be used to print configuration file and its content.

Usage:

```
gcMapExplorer config [-h] [-cs] [-pc]
```

-cs,--clean-scratch

Clean scratch directory gcMapexplorer try to remove all temporary files present in scracth directory. However, due to crash and errors, sometimes these files cannot be removed. Therefore, by this simple command, all temporary files from the scracth directory is removed.

Note: Temporary files could be huge and therefore it is advised to run this command periodically.

Note: Do not use this command when any of the gcMapExplorer tools or module are in execution process.

-pc,--print-config

Print configuration file. It will print location configuration file and its content.

Configuration using `gcMapExplorer` Python modules:

- Print configuration file: *`gcMapExplorer.config.printConfig()`*
- Get configuration as dictionary: *`gcMapExplorer.config.getConfig()`*
- Update configuration file: *`gcMapExplorer.config.updateConfig()`*
- Clean scratch directory: *`gcMapExplorer.config.cleanScratch()`*

3.1.2 How to use gcMapExplorer?

Several interfaces are available as following.

- *Graphical User Interface Applications* - Several windows like applications to perform tasks.
- **Command Line Interface - Several commands to perform tasks.**
  - *Commands to import Hi-C data*
  - *Commands to convert bigWig/wig/bed to h5*
  - *Commands to normalize Hi-C map*
- Application Programming Interface - It can be used to perform analysis by any mathematical operations through programming. Tutorial files with Jupyter-Notebooks can be downloaded from

Usage

Run `gcMapExplorer` command on terminal to get list of all sub-commands.

Following sub-commands are available:

Table 1: Graphical User Interface Applications

Command	Function
browser	Interactive Browser for genomic contact maps
cmapImporter	Interface to import contact maps and datasets
cmapNormalizer	Interface to normalize contact maps
h5Converter	Interface to convert bigWig/wig/bed file to h5 file

Table 2: Commands to import Hi-C data

Command	Function
coo2cmap	Import COO sparse matrix format to ccmap or gcmap
pairCoo2cmap	Import map from files similar to paired COO format
homer2cmap	Import HOMER Hi-C interaction matrix to ccmap or gcmap
bc2cmap	Import Bin-Contact format files to ccmap or gcmap
hic2gcmap	Import hic to gcmap

Table 3: Commands to convert bigWig/wig/bed to h5

Command	Function
bigwig2h5	Convert a bigWig file to HDF5 format h5 file
wig2h5	Convert a wig file to HDF5 format h5 file
bed2h5	Convert a bed file to HDF5 format h5 file
encode2h5	Download and convert ENCODE datasets to HDF5 format h5 files

Table 4: Commands to normalize Hi-C map

Command	Function
normKR	Normalization using Knight-Ruiz matrix balancing
normVC	Normalization using Vanilla-Coverage method
normIC	Normalization using Iterative Correction
normMCFS	Scale maps using Median/Mean Contact Frequency

Table 5: Commands for Analysis

Command	Function
corrBWcmaps	Calculate correlation between contact maps

Table 6: Configuration utility

Command	Function
config	To print configuration file and clean scratch directory

Command help

Run `gcMapExplorer <sub-commands> -h` command.

For example:

- `gcMapExplorer normKR -h`
- `gcMapExplorer coo2cmap -h`

3.1.3 Genome contact map browser

It is an application to browse genome contact maps along with respective genomic track datasets. It acts as an interactive visualizer, where user can browse the data by dragging and zooming.

To launch browser, execute following command:

```
gcMapExplorer browser
```

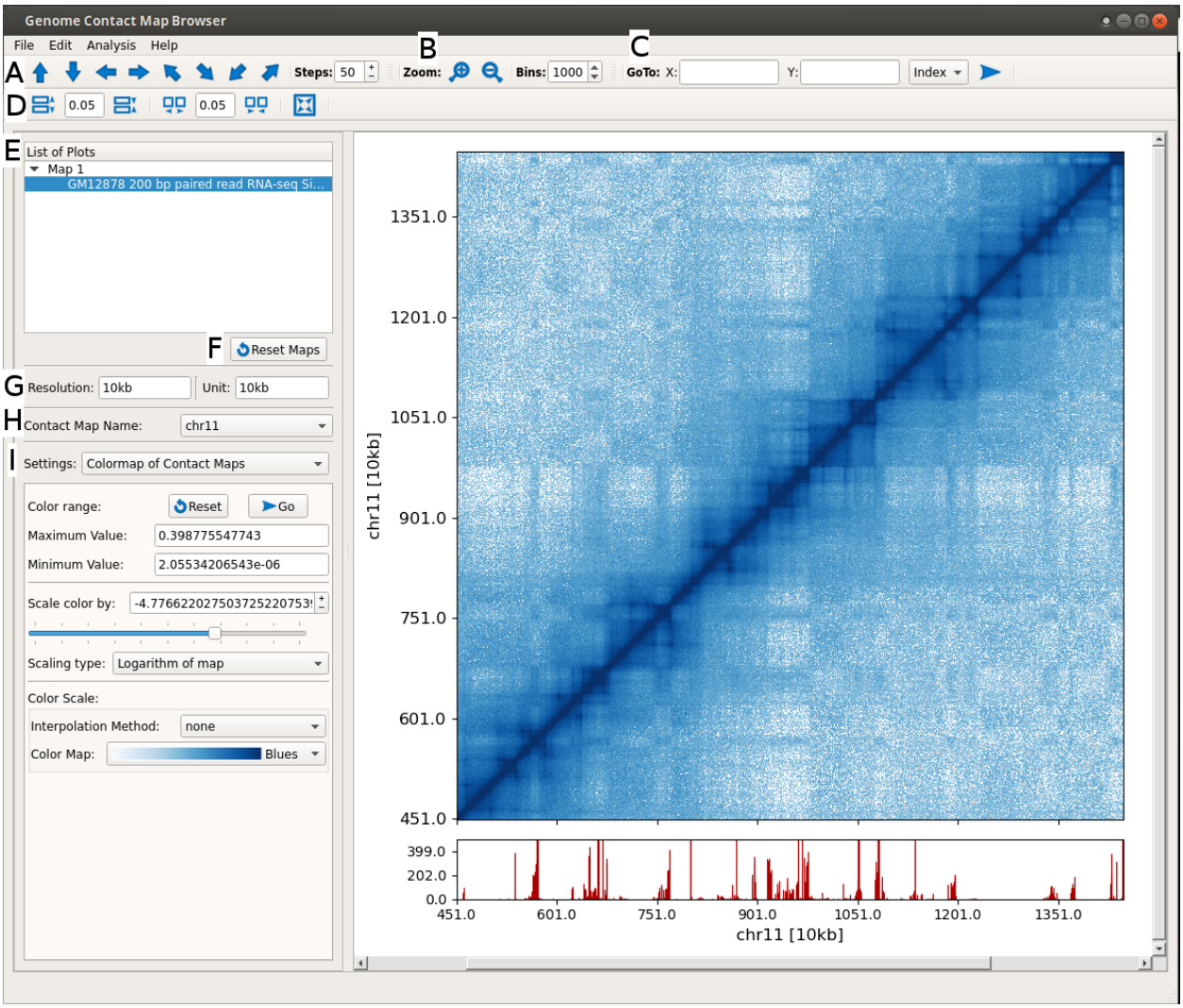


Fig. 6: Genome Contact Map browser

1. Click on arrow to move the maps in respective directions. The genomic datasets also follow automatically.
2. Click on these buttons to zoom in and out. The genomic datasets also follow automatically.
3. Use this to go to on specific coordinate. Input can be real or indexed coordinate.
4. Change width-spacing between plots.
5. List of all plots. Active plot is selected. One can select a plot to make it active.
6. Reset all maps to original view.
7. Display information of current resolution and unit shown in plots.
8. Name of contact map. Usually chromosome name.
9. Various options. Presently shown for color mapping and scaling of maps. It can be used to modify colormaps, color scaling range and color scaling type.

Add a genomic track dataset

Click on `File` -> `Add genomic dataset to ...` and select the contact map for which new dataset is need to be added. A new window will appear where user may select a compatible file by clicking on `Open` button. Afterwards, different box will appear depending on the input file format.

- h5 file: User is prompted to select the dataset.

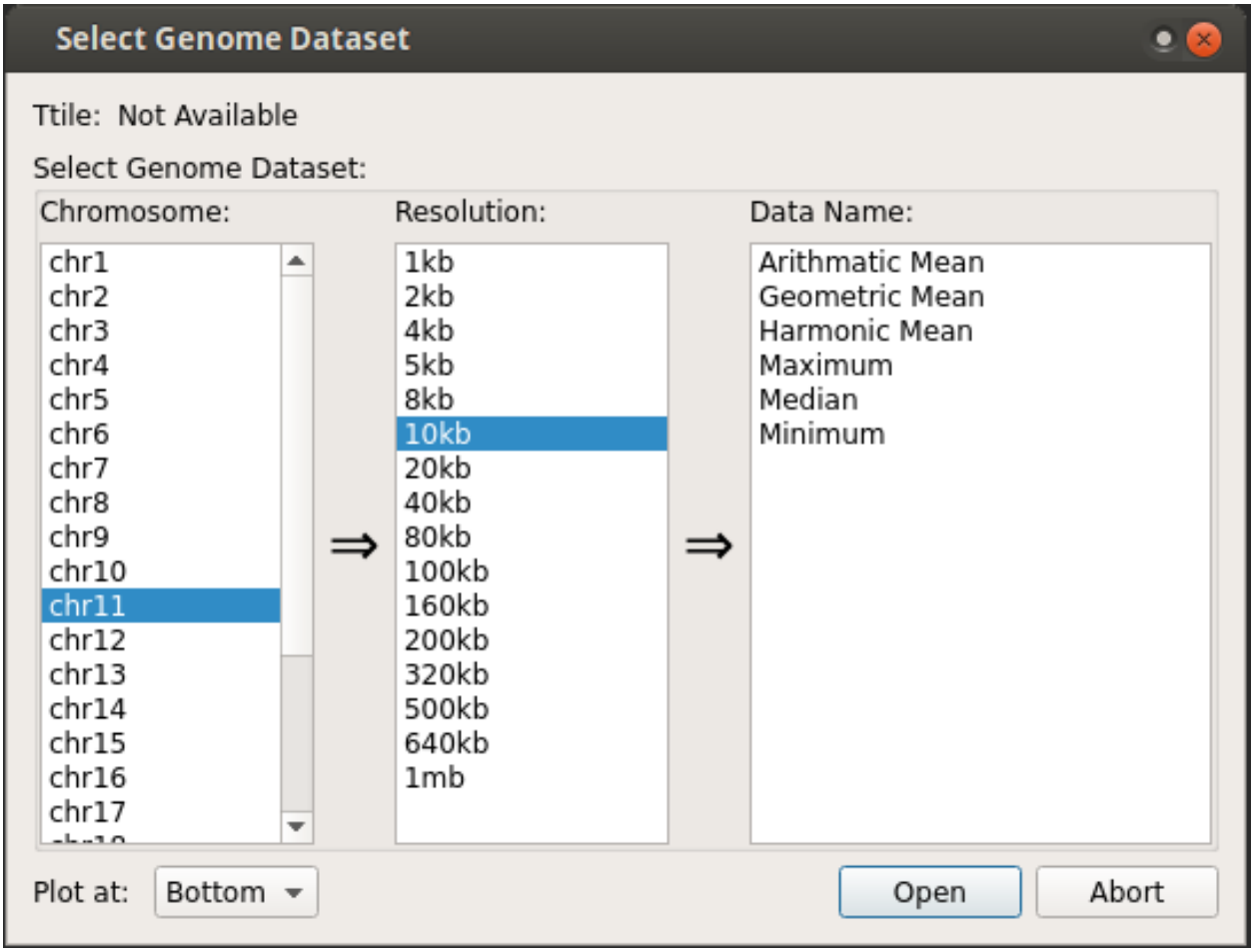


Fig. 7: Select genomic track dataset

By default, the genomic track is added at the bottom of map. However, one can select `top` in the above dialog box to plot the track at the top of the contact map.

- **bigWig/wig/bed file** : Since, only h5 files are compatible with browser, a window will appear as similar to h5Converter to convert these files to h5 on the fly. By default, the generated h5 file will be deleted after use because `Remove` is checked. To save the generated h5 file, uncheck `Remove` option and change location of output file.

  Once dataset is converted, user is prompted to select the dataset as shown above for h5 input file.

  To reduce the waiting time for conversion process, only dataset for the required chromosome is converted. When later another chromosome is selected in `browser`, h5Converter will again appear and dataset will be again converted for the selected chromosome. Once dataset for a chromosome is converted, it remains stored in the h5 file. Therefore, when this chromosome is re-selected in `browser`, no conversion is required and dataset is plotted in `browser` instantly.

Note: The options in h5Converter are only editable at first time. Later when another chromosome is selected in `browser`, this box will again appear, however, options will not be editable.

Change page size and orientation

Click on `Edit` -> `Change Page Size` and select desired page size. In case of custom page size, click on `Custom`. A dialog will open, where user can specify new page size.

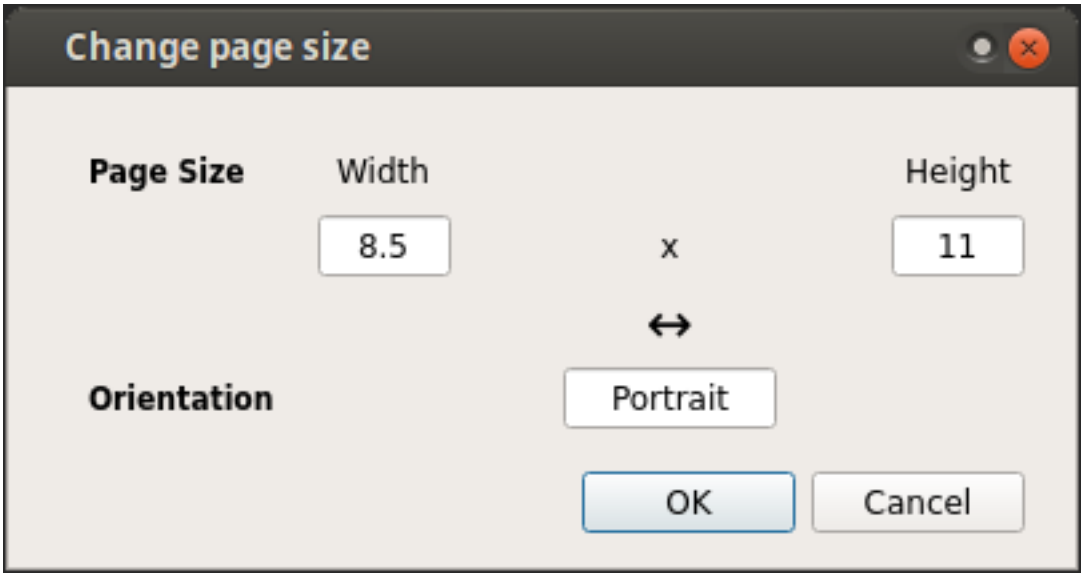


Fig. 8: Change Page Size Dialog

Save as image

Click on `File` -> `Save Plot`, choose file name with acceptable extension and click on `Save` button to save the plot as image file.

Change Genomic track plot setting

Select a `Genomic Dataset Y-Scaling` option in `Settings` at right panel. Below several options will appear. These include color, line width, maximum and minimum limit along Y-axis.

Change marker setting

Select a `Marker` option in `Settings` at right panel. Below options will appear to modify the marker settings.

User defined colormap

Although several colormap is already included in the browser. One may generate own colormap and later modify it using the implemented option. To open this, click on `Edit` -> `Add/Modify colormap`.

This box can be used to create or modify the colormaps. The shown colormap can be saved as a text file for later use.

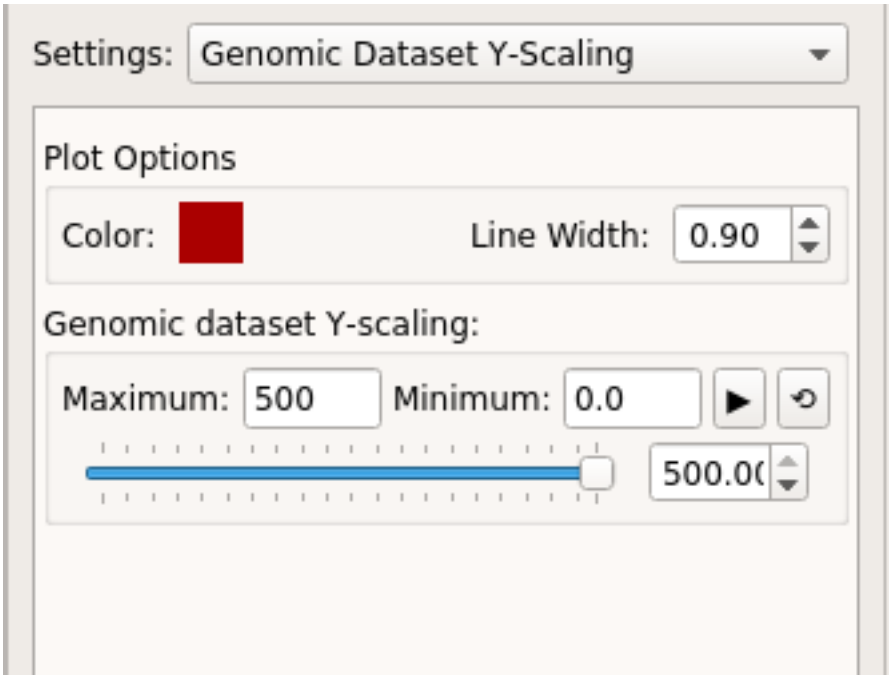


Fig. 9: Change Genomic track plot options

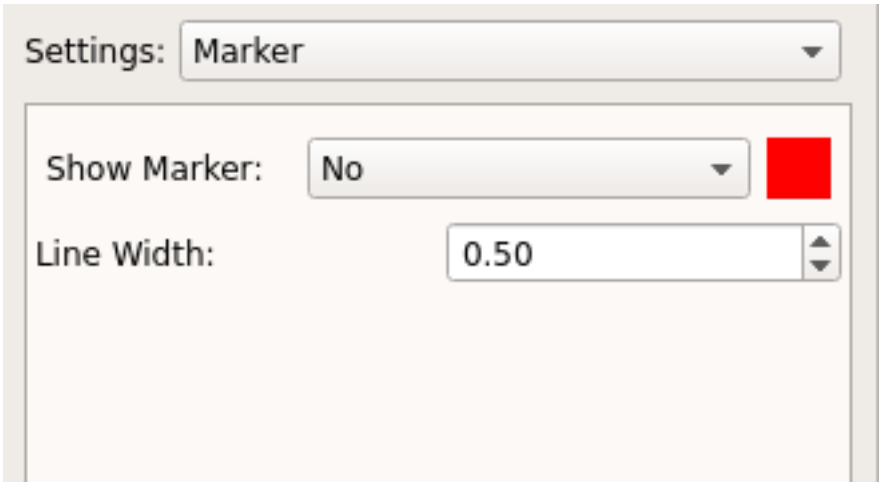


Fig. 10: Change marker settings

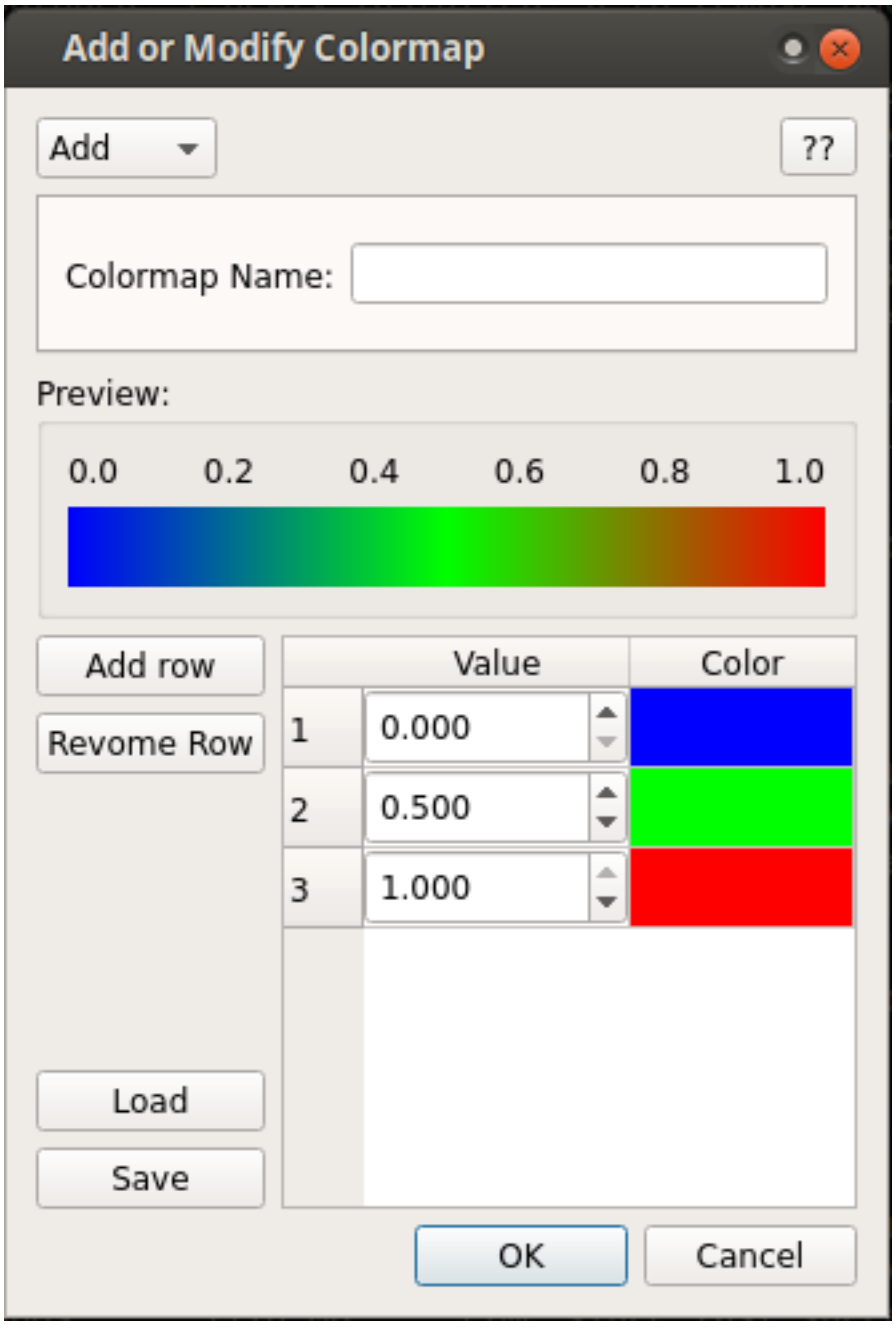


Fig. 11: Add/Modify colormap

Modify Axis Properties

Properties of both X and Y axis are highly customizable. User may customize most of the properties such as font, tick-lengths, axis-labels etc. To open this box, right click on plot and choose the axis properties.

3.1.4 File formats

gcMapExplorer uses three format of files. Both gcmap and genomic track file are in format while ccmap file is in format.

`gcmap` file

The `gcmap` file is in format to store Genome Contact Map (gcmap). It is implemented by considering both portability and readability. A single file may contains maps of various resolutions of all chromosomes. It also contains properties, which are attributes, of each map.

Structure of `gcmap` file

As format supports Hierarchical Data Model, therefore we implemented the contact maps in format way. Overall structure format is as follows:

```
HDF5
|
├──────── chr1 ─── Attributes : ['xlabel', 'ylabel', 'compression']
|           |
|           ├─────── 10kb  ( 2D Numpy Array ) ─── Attributes : ['minvalue', 'maxvalue',
↪ 'xshape', 'yshape', 'binsize']
|           ├─────── 20kb  ( 2D Numpy Array ) ─── Attributes : ['minvalue', 'maxvalue',
↪ 'xshape', 'yshape', 'binsize']
|           ├─────── 40kb  ( 2D Numpy Array ) ─── Attributes : ['minvalue', 'maxvalue',
↪ 'xshape', 'yshape', 'binsize']
|           ├─────── 60kb  ( 2D Numpy Array ) ─── Attributes : ['minvalue', 'maxvalue',
↪ 'xshape', 'yshape', 'binsize']
|           ├─────── 80kb  ( 2D Numpy Array ) ─── Attributes : ['minvalue', 'maxvalue',
↪ 'xshape', 'yshape', 'binsize']
|           ├─────── 160kb ( 2D Numpy Array ) ─── Attributes : ['minvalue', 'maxvalue',
↪ 'xshape', 'yshape', 'binsize']
|           ├─────── 320kb ( 2D Numpy Array ) ─── Attributes : ['minvalue', 'maxvalue',
↪ 'xshape', 'yshape', 'binsize']
|           ├─────── 640kb ( 2D Numpy Array ) ─── Attributes : ['minvalue', 'maxvalue',
↪ 'xshape', 'yshape', 'binsize']
|           |
|           ├─────── 10kb-bNoData  ( 1D Numpy Array )
|           ├─────── 20kb-bNoData  ( 1D Numpy Array )
|           ├─────── 40kb-bNoData  ( 1D Numpy Array )
|           ├─────── 60kb-bNoData  ( 1D Numpy Array )
|           ├─────── 80kb-bNoData  ( 1D Numpy Array )
|           ├─────── 160kb-bNoData ( 1D Numpy Array )
|           ├─────── 320kb-bNoData ( 1D Numpy Array )
|           └─────── 640kb-bNoData ( 1D Numpy Array )
|
├──────── chr2 ─── Attributes : ['xlabel', 'ylabel', 'compression']
|           |
```

(continues on next page)

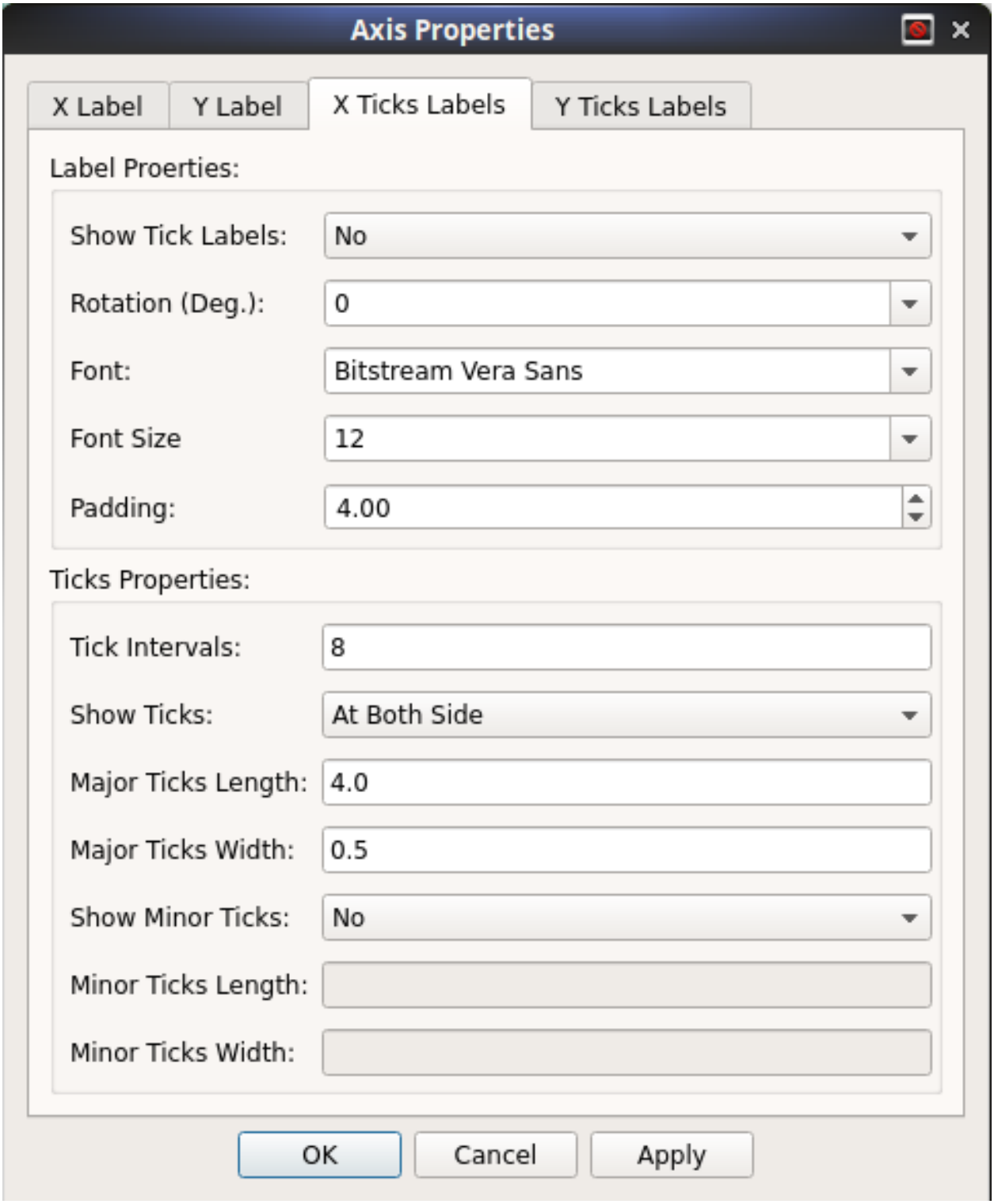


Fig. 12: Axis Properties interface in Browser

(continued from previous page)

```
│            ├──────── 10kb  ( 2D Numpy Array ) ─── Attributes : ['minvalue', 'maxvalue',
↪ 'xshape', 'yshape', 'binsize']
│            ├──────── 20kb  ( 2D Numpy Array ) ─── Attributes : ['minvalue', 'maxvalue',
↪ 'xshape', 'yshape', 'binsize']
│            ├──────── 40kb  ( 2D Numpy Array ) ─── Attributes : ['minvalue', 'maxvalue',
↪ 'xshape', 'yshape', 'binsize']
│            ├──────── 60kb  ( 2D Numpy Array ) ─── Attributes : ['minvalue', 'maxvalue',
↪ 'xshape', 'yshape', 'binsize']
│            ├──────── 80kb  ( 2D Numpy Array ) ─── Attributes : ['minvalue', 'maxvalue',
↪ 'xshape', 'yshape', 'binsize']
│            ├──────── 160kb ( 2D Numpy Array ) ─── Attributes : ['minvalue', 'maxvalue',
↪ 'xshape', 'yshape', 'binsize']
│            ├──────── 320kb ( 2D Numpy Array ) ─── Attributes : ['minvalue', 'maxvalue',
↪ 'xshape', 'yshape', 'binsize']
│            ├──────── 640kb ( 2D Numpy Array ) ─── Attributes : ['minvalue', 'maxvalue',
↪ 'xshape', 'yshape', 'binsize']
│            │
│            ├──────── 10kb-bNoData  ( 1D Numpy Array )
│            ├──────── 20kb-bNoData  ( 1D Numpy Array )
│            ├──────── 40kb-bNoData  ( 1D Numpy Array )
│            ├──────── 60kb-bNoData  ( 1D Numpy Array )
│            ├──────── 80kb-bNoData  ( 1D Numpy Array )
│            ├──────── 160kb-bNoData ( 1D Numpy Array )
│            ├──────── 320kb-bNoData ( 1D Numpy Array )
│            └──────── 640kb-bNoData ( 1D Numpy Array )
:
:
:
└──────── ...
```

Compression

In gcmap file, contact map is stored as compressed 2D matrix. Presently, two compression method are allowed in the gcmap file:

-
- GZIP

By default, is used to compress arrays. This method is very fast, and allow the rapid contact map reading. However, the size reduction is moderate in comparison with GZIP compression method.

> **Warning:** method is only available through **Python h5py** module, and therefore, this file cannot be read by another programming language through standard library. For portability, use GZIP compression method, which is available in standard HDF5 library.

Portability and Readability

The `gcmap` file with **GZIP** compressed arrays can be read and write from any programming language. For C/C++/Java, a standard HDF5 library is available from group. For R programming language, and are available.

Both GZIP and compression reduces the file size significantly as compare to respective flat text file. Therefore, this file is also suitable for storage and transfer.

Convert Hi-C data to gcmap

Hi-C data are available in several different formats. Presently, following formats can be converted to gcmap using implemented tools.

- COO sparse matrix
- Paired COO sparse matrix
- Homer Hi-C interaction matrix
- Bin-Contact pair files
- Hic files

Following tools are available for the conversion

coo2cmap - convert COO sparse matrix format to ccmap or gcmap

As shown below in example, in this format, first and second column is location on chromosome and third column is the respective value:

```
20000000        20000000        2692.0
20000000        20100000        885.0
20100000        20100000        6493.0
20000000        20200000        15.0
20100000        20200000        52.0
20200000        20200000        2.0
20000000        20300000        18.0
20100000        20300000        40.0
```

NOTE that, above location is real value. However, with `-idx/--index` option, these two same column will be considered as index value. index should always start from zero for absolute beginning of chromosome.e.g. for 10kb, 0-10000 should have index of zero, 10000-20000 have index of one. If this is file format,resolution should be provided with `-r/--resolution` option.

Usage:

```
usage: gcMapExplorer coo2cmap [-h] [-i input.txt] [-ic input.tar.gz]
                              [-mt intra] [-r 10kb] [-idx]
                              [-ccm 10kb_RawObserved] [-od OUTDIR]
                              [-gcm inOut.gcmap] [-cmeth lzf] [-dmeth sum]
                              [-wd /home/rajendra/deskForWork/scratch]
```

Optional arguments:

```
-h, --help            show this help message and exit
-i input.txt, --input input.txt
                      Meta input file containing input contact map files list with␣
↪respective
                      xlabel and ylabel. xlabel should be always provided. In case of␣
↪intra-
                      chromosomal map, only xlabel is sufficient because both x and y␣
↪axis are of
                      same chromosome. However for inter-chromosomal map, both xlabel␣
↪and ylabel
                      should be provided. Example format:
```

(continues on next page)

(continued from previous page)

```
                      100kb_resolution_intrachromosomal/chr1/MAPQGE30/chr1_100kb.
↪RAWobserved    chr1
                      100kb_resolution_intrachromosomal/chr5/MAPQGE30/chr5_100kb.
↪RAWobserved    chr5
                      100kb_resolution_intrachromosomal/chr15/MAPQGE30/chr15_100kb.
↪RAWobserved  chr15
                      100kb_resolution_intrachromosomal/chr20/MAPQGE30/chr20_100kb.
↪RAWobserved  chr20
                      100kb_resolution_intrachromosomal/chr21/MAPQGE30/chr21_100kb.
↪RAWobserved  chr21
                      100kb_resolution_intrachromosomal/chr22/MAPQGE30/chr22_100kb.
↪RAWobserved  chr22

-ic input.tar.gz, --input-compressed input.tar.gz
                      Input compressed archive file containing all the listed contact␣
↪maps.
                      Presently, only "tar.gz" and "zip" compressed files are␣
↪supported.

                      If -i/--input is not provided, all files from compressed file␣
↪will be tried for
                      processing.

-mt intra, --mapType intra
                       Type of listed contact maps: "intra" or "inter" chromosomal␣
↪map.

-r 10kb, --resolution 10kb
                      Resolution of all maps. It is an optional argument. Note that,␣
↪if this
                      option is not provided, resolution will be automatically␣
↪determined from the
                      contact map file. However, in case of -idx/--index option,␣
↪resolution
                      should be provided as resolution cannot be determined from␣
↪input contact map
                      file.

-idx, --index         It determines whether contact map files have real coordinate of␣
↪chromosome
                      or index number. If this option is enabled, -r/--resolution␣
↪option should be
                      provided.

-ccm 10kb_RawObserved, --ccmap 10kb_RawObserved
                       Use this to convert all contact maps to ccmap format files.␣
↪Provide suffix
                      of ccmap file names with this option and it will enable the␣
↪conversion.

                      Output ccmap file name is generated automatically as follows;
                      if xlabel is not equal to ylabel: <xlabel>_<ylabel>_<suffix>.
↪ccmap
                      else: <xlabel>_<suffix>.ccmap

                      Note that -od/--out-dir option is also required because all␣
↪ccmaps will be
```

(continues on next page)

(continued from previous page)

```
                      saved in this directory.

-od OUTDIR, --out-dir OUTDIR
                      Directory where all ccmap files will be saved.
-gcm inOut.gcmap, --gcmap inOut.gcmap
                      Provide gcmap file to convert all contact maps into one gcmap␣
↪file.
                      File name should contain full path because -od/--out-dir is not␣
↪considered
                      for this conversion.

-cmeth lzf, --compression-method lzf
                      Data compression method in gcmap file.
-dmeth sum, --downsample-method sum
                      Downsampling method to coarsen the resolution in gcmap file.␣
↪The option is
                      intended to use with -gcm/--gcmap option. Three accepted␣
↪methods are
                              sum  : sum of values,
                              mean : Average of values and
                              max  : Maximum of values.

                      This option generates all coarser maps where resolutions will␣
↪be coarsened by
                      a factor of two, consecutively. e.g.: In case of 10 kb input␣
↪resolution,
                      downsampled maps of "20kb", "40kb", "80kb", "160kb", "320kb"␣
↪etc. will be
                      generated until, map size is less than 500.

-wd /home/rajendra/deskForWork/scratch, --work-dir /home/rajendra/deskForWork/scratch
                      Directory where temporary files will be stored.
```

pairCoo2cmap - paired COO sparse matrix to ccmap or gcmap

This format is very similar to COO format with an additional information of chromosome. Therefore, maps for all chromosome could be contained in a single file.

This type of format appeared with this publication (GEO datasets).

Following file format can be read as a text file, where first and second column is location on chromosome and third column is the value:

```
chr4    60000   75000   chr4    60000   75000   0.1163470887070292
chr4    60000   75000   chr4    105000  120000  0.01292745430078102
chr4    60000   75000   chr4    435000  450000  0.01292745430078102
chr4    75000   90000   chr4    75000   90000   0.05170981720312409
chr4    75000   90000   chr4    345000  360000  0.01292745430078102
chr4    90000   105000  chr4    90000   105000  0.01292745430078102
.
.
.
.
.
.
```

Usage:

```
usage: gcMapExplorer pairCoo2cmap [-h] [-i maps.txt] [-ccm RawObserved]
                                  [-od OUTDIR] [-gcm inOut.gcmap] [-cmeth lzf]
                                  [-dmeth sum]
                                  [-wd /home/rajendra/deskForWork/scratch]
```

Optional arguments:

```
-h, --help            show this help message and exit
-i maps.txt, --input maps.txt
                       Input file name.

-ccm RawObserved, --ccmap RawObserved
                       Use this to convert all contact maps to ccmaps file. Provide␣
↪suffix of
                       ccmap file names with this option and it will enable the␣
↪conversion.

                       Output ccmap file name is generated automatically as follows;
                       <chromosome>_<resolution>_<suffix>.ccmap

                       Note that -od/--out-dir option is also required because all␣
↪ccmaps will be
                       saved in this directory.

-od OUTDIR, --out-dir OUTDIR
                       Directory where all ccmap files will be saved.
-gcm inOut.gcmap, --gcmap inOut.gcmap
                       Provide gcmap file to convert all contact maps into one gcmap␣
↪file.
                       File name should contain full path because -od/--out-dir is not␣
↪considered
                       for thi conversion.

-cmeth lzf, --compression-method lzf
                       Data compression method for gcmap file.
-dmeth sum, --downsample-method sum
                       Downsampling method to coarsen the resolution in gcmap file.␣
↪The option is
                       intended to use with -gcm/--gcmap option. Three accepted␣
↪methods are
                               sum  : sum of values,
                               mean : Average of values and
                               max  : Maximum of values.

                       This option generates all coarser maps where resolutions will␣
↪be coarsened by
                       a factor of two, consecutively. e.g.: In case of 10 kb input␣
↪resolution,
                       downsampled maps of "20kb", "40kb", "80kb", "160kb", "320kb"␣
↪etc. will be
                       generated until, map size is less than 500.

-wd /home/rajendra/deskForWork/scratch, --work-dir /home/rajendra/deskForWork/scratch
                       Directory where temporary files will be stored.
```

homer2cmap - HOMER Hi-C matrix to ccmap or gcmap

HOMER package contains modules to analyze genome wide interaction data. It creates Hi-C matrix in a specific format as shown as shown here.

This format contains contact map in a matrix format.

Usage:

```
usage: gcMapExplorer homer2cmap [-h] [-i matrix.txt] [-ccm RawObserved]
                                [-od OUTDIR] [-gcm inOut.gcmap] [-cmeth lzf]
                                [-dmeth sum]
                                [-wd /home/rajendra/deskForWork/scratch]
```

Optional arguments:

```
-h, --help            show this help message and exit
-i matrix.txt, --input matrix.txt
                       File containing HOMER Hi-C ineraction matrix format contact map

-ccm RawObserved, --ccmap RawObserved
                       Use this to convert all contact maps to ccmaps file. Provide␣
↪suffix of
                      ccmap file names with this option and it will enable the␣
↪conversion.

                      Output ccmap file name is generated automatically as follows;
                      <chromosome>_<resolution>_<suffix>.ccmap

                      Note that -od/--out-dir option is also required because all␣
↪ccmaps will be
                      saved in this directory.

-od OUTDIR, --out-dir OUTDIR
                      Directory where all ccmap files will be saved.
-gcm inOut.gcmap, --gcmap inOut.gcmap
                      Provide gcmap file to convert all contact maps into one gcmap␣
↪file.
                      File name should contain full path because -od/--out-dir is not␣
↪considered
                      for thi conversion.

-cmeth lzf, --compression-method lzf
                      Data compression method for gcmap file.
-dmeth sum, --downsample-method sum
                      Downsampling method to coarsen the resolution in gcmap file.␣
↪The option is
                      intended to use with -gcm/--gcmap option. Three accepted␣
↪methods are
                              sum  : sum of values,
                              mean : Average of values and
                              max  : Maximum of values.

                      This option generates all coarser maps where resolutions will␣
↪be coarsened by
                      a factor of two, consecutively. e.g.: In case of 10 kb input␣
↪resolution,
                      downsampled maps of "20kb", "40kb", "80kb", "160kb", "320kb"␣
↪etc. will be
```

(continues on next page)

(continued from previous page)

```
                      generated until, map size is less than 500.

-wd /home/rajendra/deskForWork/scratch, --work-dir /home/rajendra/deskForWork/scratch
                      Directory where temporary files will be stored.
```

bc2cmap -Bin-Contact files pair to ccmap or gcmap

In this format, two separate files are available. One file contains bins information and other contains contact frequency.

These types of files are present in following GEO data:

- http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE61471
- http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE34453

This format contains a pair of file:

BIN file:

```
cbin        chr     from.coord      to.coord        count
1   2L      0       160000  747
2   2L      160000  320000  893
3   2L      320000  480000  1056
4   2L      480000  640000  1060
5   2L      640000  800000  978
6   2L      800000  960000  926
.
.
.
```

CONTACT file in list format:

```
cbin1       cbin2   expected_count  observed_count
1   1       40.245201       21339
1   2       83.747499       5661
1   3       92.12501        1546
1   4       93.401273       864
1   5       87.265472       442
.
.
.
```

Both BIN and CONTACT files are **necessary** for the conversion.

Usage:

```
usage: gcMapExplorer bc2cmap [-h] [-ib nm_none_160000.bins]
                             [-ic nm_none_160000.n_contact] [-ccm RawObserved]
                             [-od OUTDIR] [-gcm inOut.gcmap] [-cmeth lzf]
                             [-dmeth sum]
                             [-wd /home/rajendra/deskForWork/scratch]
```

Optional arguments:

```
-h, --help            show this hel
p message and exit
-ib nm_none_160000.bins, --input-bin nm_none_160000.bins
```

(continues on next page)

(continued from previous page)

```
                        Input BIN file as shown above.

-ic nm_none_160000.n_contact, --input-contact nm_none_160000.n_contact
                        Input CONTACT file as shown above.

-ccm RawObserved, --ccmap RawObserved
                        Use this to convert all contact maps to ccmaps file. Provide␣
↪suffix of
                        ccmap file names with this option and it will enable the␣
↪conversion.

                        Output ccmap file name is generated automatically as follows;
                        <chromosome>_<resolution>_<suffix>.ccmap

                        Note that -od/--out-dir option is also required because all␣
↪ccmaps will be
                        saved in this directory.

-od OUTDIR, --out-dir OUTDIR
                        Directory where all ccmap files will be saved.
-gcm inOut.gcmap, --gcmap inOut.gcmap
                        Provide gcmap file to convert all contact maps into one gcmap␣
↪file.
                        File name should contain full path because -od/--out-dir is not␣
↪considered
                        for thi conversion.

-cmeth lzf, --compression-method lzf
                        Data compression method for gcmap file.
-dmeth sum, --downsample-method sum
                        Downsampling method to coarsen the resolution in gcmap file.␣
↪The option is
                        intended to use with -gcm/--gcmap option. Three accepted␣
↪methods are
                                sum  : sum of values,
                                mean : Average of values and
                                max  : Maximum of values.

                        This option generates all coarser maps where resolutions will␣
↪be coarsened by
                        a factor of two, consecutively. e.g.: In case of 10 kb input␣
↪resolution,
                        downsampled maps of "20kb", "40kb", "80kb", "160kb", "320kb"␣
↪etc. will be
                        generated until, map size is less than 500.

-wd /home/rajendra/deskForWork/scratch, --work-dir /home/rajendra/deskForWork/scratch
                        Directory where temporary files will be stored.
```

hic2gcmap - convert hic file to gcmap

Hic files as described in[1] can be converted to gcmap.

Usage:

[1] Durand, Neva C. *et al*. Juicer Provides a One-Click System for Analyzing Loop-Resolution Hi-C Experiments, Cell Systems, Volume 3, Issue 1, p. 95-98.

```
usage: gcMapExplorer hic2gcmap [-h] [-c A B | -l] [--compression C] [-r R]
                               [-n N] [--downsampling D]
                               input [output]
```

Arguments:

```
positional arguments:
  input                 hic input file
  output                output file or directory

optional arguments:
  -h, --help            show this help message and exit
  -c A B, --chromosomes A B
                        a pair of chromosomes A B
  -l, --list            list all available chromosomes
  --compression C       compression type, choose between lzf, gzip (default:
                        lzf)
  -r R, --resolution R  the resolution, as an integer or as kb (default:
                        finest)
  -n N, --norm N        the type of norm to use, choose between VC, VC_SQRT,
                        KR, none (default: none)
  --downsampling D      the downsampling method to use, choose between sum,
                        mean, max, none (default: sum)
```

Examples: Import all chromosome pairs at the finest available resolution:

```
gcMapExplorer hic2gcmap input.hic
```

List all available chromosomes:

```
gcMapExplorer hic2gcmap input.hic --list
```

Import chromosome pair X X and save output to `output.gcmap`:

```
gcMapExplorer hic2gcmap input.hic ouput.gcmap -c X X
```

Same as above but save to a generated filename in `outdir/` using finest resulution 10kb:

```
gcMapExplorer hic2gcmap input.hic outdir/ -c X X -r 10kb
```

References:

cmapImporter - An application to import ccmap or gcmap

It is an application for converting external Hi-C format into either gcmap or ccmap format.

It can be launched by following command:

```
gcMapExplorer cmapImporter
```

This interface contains in-built help (**Click on** `??` **button**) to understand the functionality of interfaces.

Convert using `gcMapExplorer` Python modules:

- COO sparse matrix : *gcMapExplorer.lib.importer.CooMatrixHandler*
- Paired COO sparse matrix : *gcMapExplorer.lib.importer.PairCooMatrixHandler*

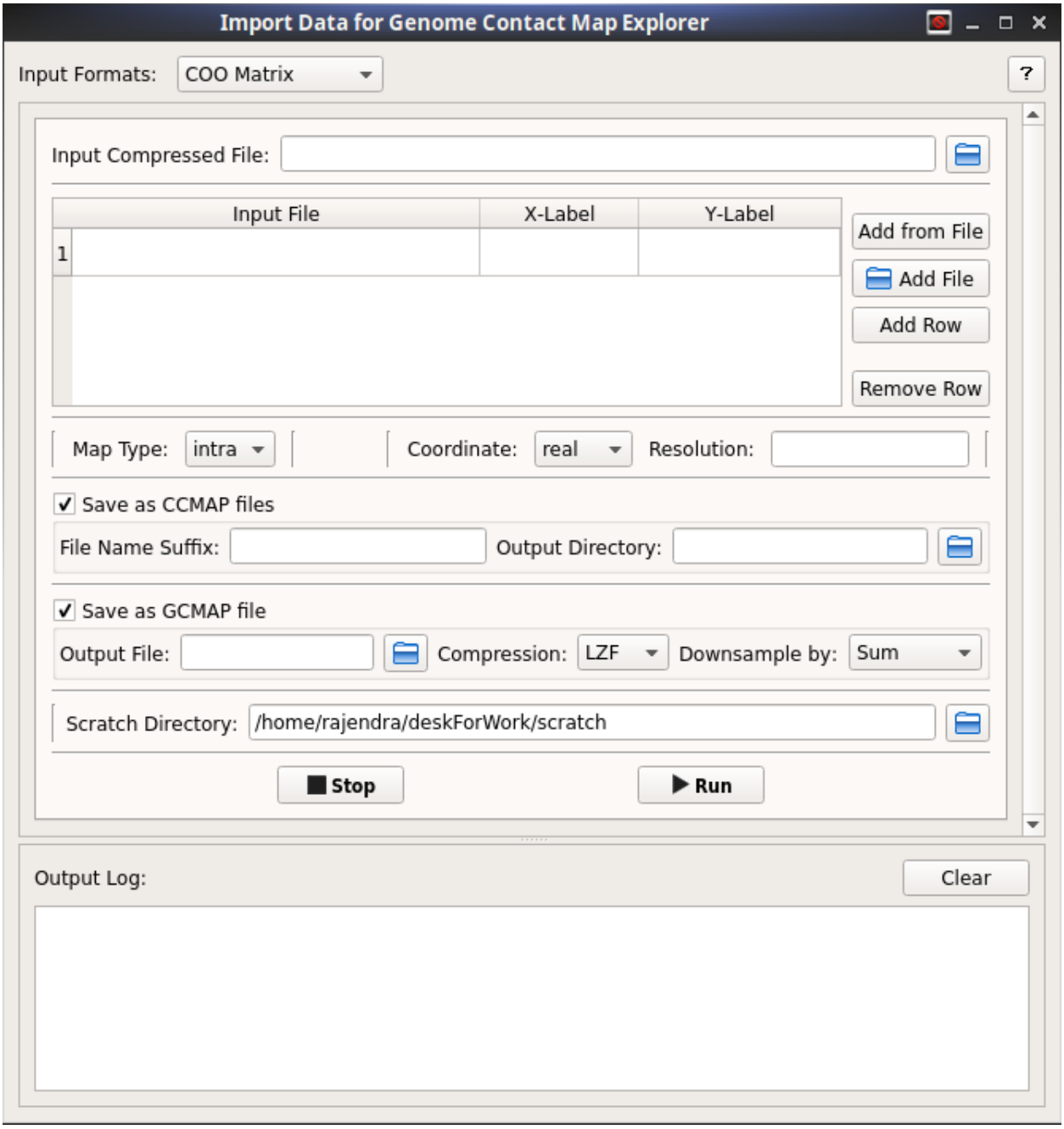


Fig. 13: cmapImporter Interface

- Homer Hi-C interaction matrix : *gcMapExplorer.lib.importer.HomerInputHandler*
- Bin-Contact pair files : *gcMapExplorer.lib.importer.BinsNContactFilesHandler*

See also:

A tutorial to convert external Hi-C maps into ccmap or gcmap using `gcMapExplorer` module are shown here.

`*.ccmap` and `*.npbin` files

This package implements the Chromosome Contact Map (ccmap) data in a specific format, which contains two inter-related file

- `*.ccmap`: It is a text file and contains meta-data for the Chromosome Contact Map.
- `*.npbin` or `*.npbin.gz`: This file contains the Chromosome Contact Map data, which is a memory mapped 2D matrix.

In contrast to `gcmap` file, the `*.ccmap` and `*.npbin` paired files contain only one contact map. To perform mathematical operations directly using `gcmap` is slow (see here), therefore, gcMapExplorer uses `*.ccmap` and `*.npbin` paired files during mathematical calculations.

Why two files?

Contact map is a two-dimensional matrix. Size of matrix can be very huge for large chromosome at high resolutions. Memory required to handle such huge matrices could be very large and sometimes beyond the available hardware. Therefore, we used a matrix mapped to the file `*.npbin` (compressed `*.npbin.gz`), which is stored in external disk. For each Contact map, we also need to store some properties like its title/name, size, minimum and maximum values, columns/rows with missing data, and path to memory mapped matrix file. These properties are stored in `*.ccmap` file.

Advantages of memory mapped matrix file

- It is a binary indexed file and any particular region of the matrix can be rapidly accessed.
- This file is generated using , and therefore, it can be used as . See also for more about numpy array.
- Because, it can be used as a numpy array, operations can be performed to access the data.
- All mathematical operations available in and modules can be directly performed.

Contents of `*.ccmap` file

`*.ccmap` is a text file and its content is shown as example:

```
{
    "title":null,
    "path2matrix":"chr22_100kb_normKR.npbin",
    "xlabel":null,
    "minvalue":"7.207987891888479e-06",
    "bLog":false,
    "state":"saved",
    "maxvalue":"0.28213343024253845",
    "binsize":100000,
    "shape":[
```

(continues on next page)

(continued from previous page)


```
        "513",
        "513"
    ],
    "matrix":null,
    "yticks":[
        "0",
        "51300000"
    ],
    "bNoData":"1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
    "xticks":[
        "0",
        "51300000"
    ],
    "ylabel":null,
    "dtype":"float32"
}
```


Following properties are included in Contact Map metadata file:

- `title`: Title of the data. Used to display in browser.
- `path2matrix`: Path to `*.npbin` or `*.npbin.gz` file
- `bLog`: Whether values in matrix is Logarithm values
- `maxvalue`: Maximum value
- `minvalue`: Minimum value
- `binsize`: Resolution of data.
- `shape`: Shape of matrix along X and Y axis
- `xticks`: Upper and lower limits of X-axis
- `yticks`: Upper and lower limits of Y-axis
- `xlabel`: Label for x-axis
- `ylabel`: Label for y-axis
- `matrix`: See `gcMapExplorer.lib.CCMAP.matrix`
- `state`: See `gcMapExplorer.lib.CCMAP.state`
- `dtype`: Data type for memory mapped matrix file. e.g. float, float32, float64 etc.
- `bNoData`: Whether data is missing for entire row/column

Genomic track HDF5 (.h5) file

To enable rapid visualization of genomic track datasets along with contact map, we used format file to store these datasets at various resolutions. HDF5 is a binary indexed file and therefore, entire datasets or a portion of dataset can be accessed rapidly.

HDF5 library is available for C, C++, R, Java and Python programming language, and therefore these files can be directly read through these languages.

Downsampling or Coarsening of datasets

Genomic contact map can be of different resolutions, and therefore, resolution of corresponding genomic track datasets should match during the visualization/analysis. Therefore, genomic dataset need to be downsampled or coarsened. However, there are several possible methods for downsampling that can be suitable for different purposes. Therefore, we have implemented six different methods as follows,

- Arithmetic mean
- Geometric mean
- Harmonic mean
- Median
- Maximum
- Minimum

When a file is opened in the `browser`, user receives a prompt for selection of downsampling method, and subsequently, the selected data is loaded into the `browser`.

Structure of genomic track (.h5) file

Format: `/<Chromosome>/<Resolution>/<1D Numpy Array>`

```
HDF5 ──────────────────────────> title
  ├──────── chr1
  │           ├──── 1kb
  │           │       ├──────── amean  ( Arithmetic mean) (type: 1D Array)
  │           │       ├──────── median ( Median value   ) (type: 1D Array)
  │           │       ├──────── hmean  ( Harmonic mean  ) (type: 1D Array)
  │           │       ├──────── gmean  ( Geometric mean ) (type: 1D Array)
  │           │       ├──────── min    ( Minimum value  ) (type: 1D Array)
  │           │       └──────── max    ( Maximum value  ) (type: 1D Array)
  │           │
  │           ├────  5kb
  │           │       ├──────── amean  ( Arithmetic mean) (type: 1D Array)
  │           │       ├──────── median ( Median value   ) (type: 1D Array)
  │           │       ├──────── hmean  ( Harmonic mean  ) (type: 1D Array)
  │           │       ├──────── gmean  ( Geometric mean ) (type: 1D Array)
  │           │       ├──────── min    ( Minimum value  ) (type: 1D Array)
  │           │       └──────── max    ( Maximum value  ) (type: 1D Array)
  │           │
  │           └────  ...
  │
  ├──────── chr2
  │           ├──── 1kb
  │           │       ├──────── amean  ( Arithmetic mean) (type: 1D Array)
  │           │       ├──────── median ( Median value   ) (type: 1D Array)
  │           │       ├──────── hmean  ( Harmonic mean  ) (type: 1D Array)
  │           │       ├──────── gmean  ( Geometric mean ) (type: 1D Array)
  │           │       ├──────── min    ( Minimum value  ) (type: 1D Array)
  │           │       └──────── max    ( Maximum value  ) (type: 1D Array)
  │           │
  │           └────  ..
  │
  :
  :
  :
  └──────── ...
```

Compression

In h5 file, dataset is stored as an 1D array. Presently, two compression methods are allowed in the h5 file:

-
- GZIP

By default, is used to compress arrays. This method is very fast, and allow the reading.

> **Warning:** method is only available through **Python h5py** module, and therefore, this file cannot be read by another programming language through standard library.
>
> For portability, use GZIP compression method, which is available in standard HDF5 library.

Convert bigWig/wig/bed to genomic track h5 file

To convert bigWig/wig/bed files to genomic track files a GUI application and several commands are available.

h5Converter

This is a GUI application, which can be used to convert bigWig/wig/bed file into `gcMapExplorer browser` compatible h5 file. To open this interface, use following command:

```
gcMapExplorer h5Converter
```

This interface contains in-built help (**Click on** `??` **button**) to understand the functionality of interfaces.

bigwig2h5

Description:

```
Import a bigWig file to HDF5 format h5 file
===========================================

bigWig file can be converted into gcMapExplorer compatible HDF5 file using
this tool. This HDF5 file can be loaded into gcMapExplorer browser for
interactive visualization.

Requirements
============
1) bigWigToWig : It converts binary bigWig file to ascii Wig file.
2) bigWigInfo : It fetches the information about chromosomes from bigWig file.

Both tools can be downloaded from http://hgdownload.cse.ucsc.edu/admin/exe/
for linux and Mac platform. However, these tools are not yet available for
Windows OS.

Path to these tools can be set using gcMapExplorer configure utility or can be
given with the command.

Resolutions
===========
```

(continues on next page)

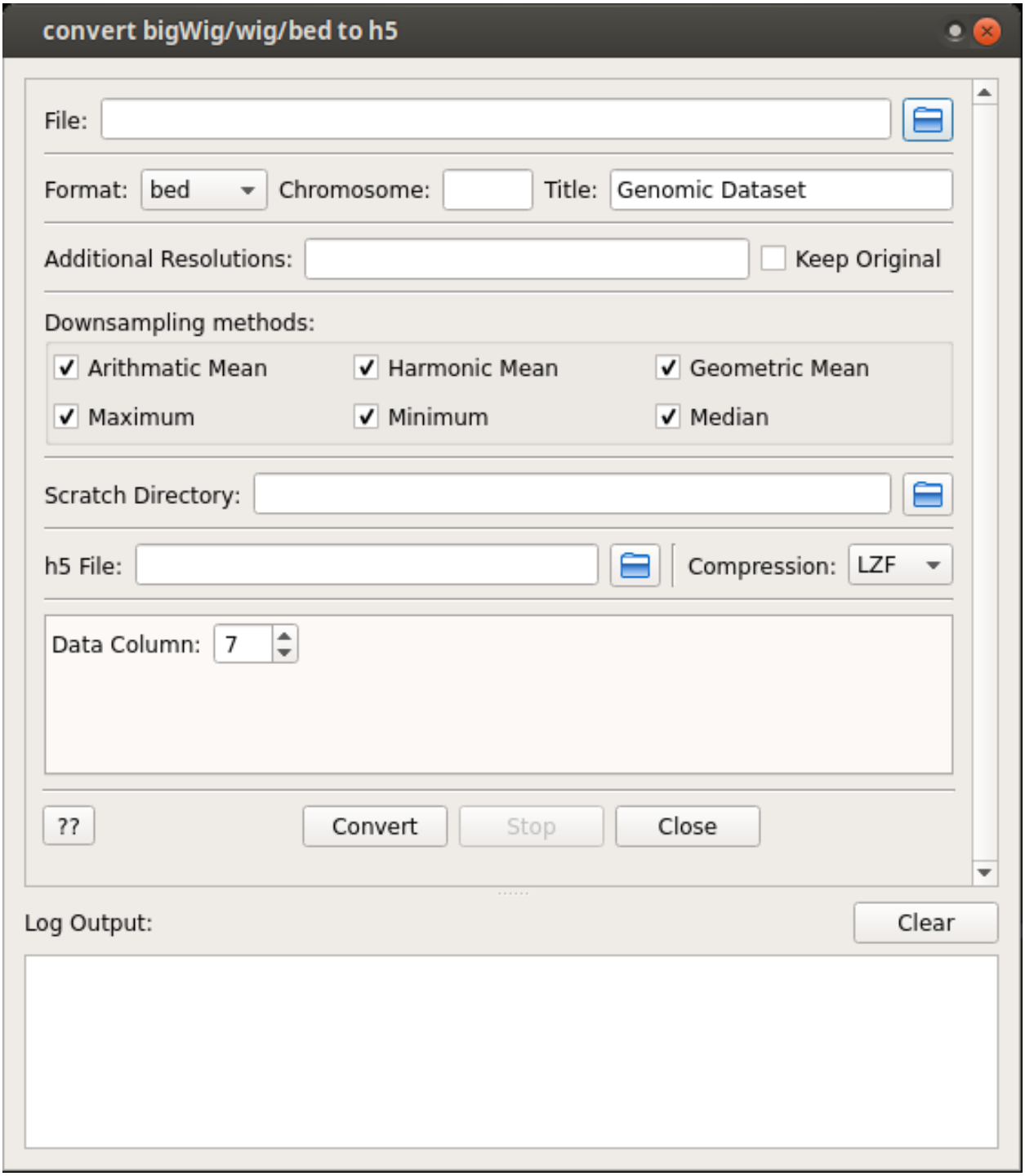


Fig. 14: h5Converter Interface

(continued from previous page)

```
By default, original data are downsampled to following resolutions:  '1kb',
'2kb', '4kb', '5kb', '8kb', '10kb', '20kb', '40kb', '80kb', '100kb', '160kb',
'200kb', '320kb', '500kb', '640kb',  and '1mb'.

The data are downsampled at this stage only to speed up the visualization
process as downsampling might slow down the interactive visualization.

Downsampling/Coarsening method
==============================
Presently, six methods are implemented:
1) min    -> Minimum value
2) max    -> Maximum value
3) amean  -> Arithmetic mean or average
4) hmean  -> Harmonic mean
5) gmean  -> Geometric mean
6) median -> Median

All these methods are used by default.
See below help for "-dm/--downsample-method" option to change the methods.

To keep original 1 base resolution data
=======================================
By default, the output h5 file does not contain original 1-base resolution
data to reduce the file size. To keep the original data in h5 file, used
-ko/--keep-original flag.
```

Usage:

```
usage: gcMapExplorer bigwig2h5 [-h] [-i input.bigWig] [-t "Genomic Dataset"]
                               [-b2w bigWigToWig] [-binfo bigWigInfo]
                               [-r "List of Resolutions"] [-icn CHROMNAME]
                               [-dm "List of downsampling method"]
                               [-cmeth lzf] [-o out.h5] [-ow] [-ko]
                               [-wd /home/rajendra/deskForWork/scratch]
```

Optional arguments:

```
-h, --help            show this help message and exit
-i input.bigWig, --input input.bigWig
                      Input bigWig file.

-t "Genomic Dataset", --title "Genomic Dataset"
                      Title of the dataset.
-b2w bigWigToWig, --bigWigToWig bigWigToWig
                      Path to bigWigToWig tool.

                      This is not necessary when bigWigToWig path is already set using␣
↪gcMapExplorer
                      configure utility.

                      It can be downloaded from http://hgdownload.cse.ucsc.edu/admin/
↪exe/
                      for linux and Mac platform.

-binfo bigWigInfo, --bigWigInfo bigWigInfo
                       Path to bigWigInfo tool.
```

(continues on next page)

(continued from previous page)

```
                    This is not necessary when bigWigInfo path is already set using␣
↪gcMapExplorer
                    configure utility.

                    It can be downloaded from http://hgdownload.cse.ucsc.edu/admin/
↪exe/
                    for linux and Mac platform.

-r "List of Resolutions", --resolutions "List of Resolutions"
                    Additional input resolutions other than these resolutions: 1kb',
↪'2kb',
                    '4kb', '5kb', '8kb', '10kb', '20kb', '40kb', '80kb', '100kb',
↪'160kb','200kb',
                    '320kb', '500kb', '640kb',  and '1mb'.

                    Resolutions should be provided in comma separated values. For␣
↪Example:
                    -r "25kb, 50kb, 75kb"

-icn CHROMNAME, --input-chromosome CHROMNAME
                    Input Chromosome Name.
                    If this is provided, only this chromosome data is extracted and␣
↪stored in h5
                    file.

-dm "List of downsampling method", --downsample-method "List of downsampling method"
                    Methods to coarse or downsample the data for converting from 1-
↪base
                    to coarser resolutions. If this option is not provided, all six␣
↪methods (see
                    above) will be considered. User may use only subset of these␣
↪methods.
                    For example: -dm "max, amean" can be used for downsampling by␣
↪only these
                    two methods.

-cmeth lzf, --compression-method lzf
                    Data compression method in h5 file.
-o out.h5, --out out.h5
                    Output h5 file.

                    If file is already present, it will replace the data. Therefore,␣
↪be careful
                    if a file with same name is present.

-ow, --overwrite      If a output file is already present, overwrite the datasets in␣
↪the output
                    file.

-ko, --keep-original  To copy original 1-base resolution data in h5 file. This will␣
↪increase the
                    file size significantly.

-wd /home/rajendra/deskForWork/scratch, --work-dir /home/rajendra/deskForWork/scratch
                    Directory where temporary files will be stored.
```

wig2h5

Description:

```
Import a wig file to HDF5 format h5 file
=========================================

wig file can be converted into gcMapExplorer compatible HDF5 file using
this tool. This HDF5 file can be loaded into gcMapExplorer browser for
interactive visualization.

This tool does not require any external program.

Resolutions
===========
By default, original data are downsampled to following resolutions:  '1kb',
'2kb', '4kb', '5kb', '8kb', '10kb', '20kb', '40kb', '80kb', '100kb', '160kb',
'200kb', '320kb', '500kb', '640kb',  and '1mb'.

The data are downsampled at this stage only to speed up the visualization
process as downsampling might slow down the interactive visualization.

Downsampling/Coarsening method
==============================
Presently, six methods are implemented:
1) min    -> Minimum value
2) max    -> Maximum value
3) amean  -> Arithmetic mean or average
4) hmean  -> Harmonic mean
5) gmean  -> Geometric mean
6) median -> Median

All these methods are used by default.
See below help for "-dm/--downsample-method" option to change the methods.

To keep original 1 base resolution data
=======================================
By default, the output h5 file does not contain original 1-base resolution
data to reduce the file size. To keep the original data in h5 file, used
-ko/--keep-original flag.
```

Usage:

```
usage: gcMapExplorer wig2h5 [-h] [-i input.wig] [-t "Genomic Dataset"]
                            [-r "List of Resolutions"]
                            [-dm "List of downsampling method"]
                            [-icn CHROMNAME] [-cmeth lzf] [-o out.h5] [-ow]
                            [-ko] [-idf index.json]
                            [-wd /home/rajendra/deskForWork/scratch]
```

Optional arguments:

```
-h, --help            show this help message and exit
-i input.wig, --input input.wig
                      Input wig file.

-t "Genomic Dataset", --title "Genomic Dataset"
```

(continues on next page)

(continued from previous page)

```
                      Title of the dataset.
-r "List of Resolutions", --resolutions "List of Resolutions"
                      Additional input resolutions other than these resolutions: 1kb',
↪ '2kb',
                      '4kb', '5kb', '8kb', '10kb', '20kb', '40kb', '80kb', '100kb',
↪'160kb','200kb',
                      '320kb', '500kb', '640kb',  and '1mb'.

                      Resolutions should be provided in comma separated values. For␣
↪Example:
                      -r "25kb, 50kb, 75kb"

-dm "List of downsampling method", --downsample-method "List of downsampling method"
                      Methods to coarse or downsample the data for converting from 1-
↪base
                      to coarser resolutions. If this option is not provided, all six␣
↪methods (see
                      above) will be considered. User may use only subset of these␣
↪methods.
                      For example: -dm "max, amean" can be used for downsampling by␣
↪only these
                      two methods.

-icn CHROMNAME, --input-chromosome CHROMNAME
                      Input Chromosome Name.
                      If this is provided, only this chromosome data is extracted and␣
↪stored in h5
                      file.

-cmeth lzf, --compression-method lzf
                      Data compression method in h5 file.
-o out.h5, --out out.h5
                      Output h5 file.

                      If file is already present, it will replace the data. Therefore,
↪ be careful
                      if a file with same name is present.

-ow, --overwrite      If a output file is already present, overwrite the datasets in␣
↪the output
                      file.

-ko, --keep-original  To copy original 1-base resolution data in h5 file. This will␣
↪increase the
                      file size significantly.

-idf index.json, --index-file index.json
                      Index file in json format.
                      A file in json format containing indices (position in wig file)␣
↪and sizes of
                      chromosomes. If this file is not present and given as input, a␣
↪new file will be
                      generated. If this file is present, indices and sizes will be␣
↪taken from this
                      file. If index and size of input chromosome is not present in␣
↪json file, these
                      will be determined from wig file and stored in same json file.␣
↪This file could
```

(continues on next page)

(continued from previous page)

```
                        be very helpful in case when same wig file has to be read many␣
↪times because
                        step to determine index and size of chromosome is skipped.

-wd /home/rajendra/deskForWork/scratch, --work-dir /home/rajendra/deskForWork/scratch
                        Directory where temporary files will be stored.
```

bed2h5

Description:

```
Import a bed file to HDF5 format h5 file
============================================

bed file can be converted into gcMapExplorer compatible HDF5 file using
this tool. This HDF5 file can be loaded into gcMapExplorer browser for
interactive visualization.

This tool does not require any external program.

Resolutions
===========
By default, original data are downsampled to following resolutions:  '1kb',
'2kb', '4kb', '5kb', '8kb', '10kb', '20kb', '40kb', '80kb', '100kb', '160kb',
'200kb', '320kb', '500kb', '640kb',  and '1mb'.

The data are downsampled at this stage only to speed up the visualization
process as downsampling might slow down the interactive visualization.

Downsampling/Coarsening method
==============================
Presently, six methods are implemented:
1) min    -> Minimum value
2) max    -> Maximum value
3) amean  -> Arithmetic mean or average
4) hmean  -> Harmonic mean
5) gmean  -> Geometric mean
6) median -> Median

All these methods are used by default.
See below help for "-dm/--downsample-method" option to change the methods.

To keep original 1 base resolution data
=======================================
By default, the output h5 file does not contain original 1-base resolution
data to reduce the file size. To keep the original data in h5 file, used
-ko/--keep-original flag.
```

Usage:

```
usage: gcMapExplorer bed2h5 [-h] [-i input.bed] [-t "Genomic Dataset"]
                            [-dtc 7] [-r "List of Resolutions"]
                            [-dm "List of downsampling method"]
                            [-icn CHROMNAME] [-cmeth lzf] [-o out.h5] [-ow]
```

(continues on next page)

(continued from previous page)

```
                              [-ko] [-idf index.json]
                              [-wd /home/rajendra/deskForWork/scratch]
```

Optional arguments:

```
-h, --help            show this help message and exit
-i input.bed, --input input.bed
                      Input wig file.

-t "Genomic Dataset", --title "Genomic Dataset"
                      Title of the dataset.
-dtc 7, --data-column 7
                      The column number, which is considered as data column. Column␣
↪number
                      could vary and depends on BED format. For example:
                      1) ENCODE broadPeak format (BED 6+3): 7th column
                      2) ENCODE gappedPeak format (BED 12+3): 13th column
                      3) ENCODE narrowPeak format (BED 6+4): 7th column
                      4) ENCODE RNA elements format (BED 6+3): 7th column

-r "List of Resolutions", --resolutions "List of Resolutions"
                      Additional input resolutions other than these resolutions: 1kb',
↪ '2kb',
                      '4kb', '5kb', '8kb', '10kb', '20kb', '40kb', '80kb', '100kb',
↪'160kb','200kb',
                      '320kb', '500kb', '640kb',  and '1mb'.

                      Resolutions should be provided in comma separated values. For␣
↪Example:
                      -r "25kb, 50kb, 75kb"

-dm "List of downsampling method", --downsample-method "List of downsampling method"
                      Methods to coarse or downsample the data for converting from 1-
↪base
                      to coarser resolutions. If this option is not provided, all six␣
↪methods (see
                      above) will be considered. User may use only subset of these␣
↪methods.
                      For example: -dm "max, amean" can be used for downsampling by␣
↪only these
                      two methods.

-icn CHROMNAME, --input-chromosome CHROMNAME
                      Input Chromosome Name.
                      If this is provided, only this chromosome data is extracted and␣
↪stored in h5
                      file.

-cmeth lzf, --compression-method lzf
                      Data compression method in h5 file.
-o out.h5, --out out.h5
                      Output h5 file.

                      If file is already present, it will replace the data. Therefore,
↪ be careful
                      if a file with same name is present.
```

(continues on next page)

(continued from previous page)

```
-ow, --overwrite      If a output file is already present, overwrite the datasets in␣
↪the output
                      file.

-ko, --keep-original  To copy original 1-base resolution data in h5 file. This will␣
↪increase the
                      file size significantly.

-idf index.json, --index-file index.json
                      Index file in json format.
                      A file in json format containing indices (position in bed file)␣
↪and sizes of
                      chromosomes. If this file is not present and given as input, a␣
↪new file will be
                      generated. If this file is present, indices and sizes will be␣
↪taken from this
                      file. If index and size of input chromosome is not present in␣
↪json file, these
                      will be determined from bed file and stored in same json file.␣
↪This file could
                      be very helpful in case when same bed file has to be read many␣
↪times because
                      step to determine index and size of chromosome is skipped.

-wd /home/rajendra/deskForWork/scratch, --work-dir /home/rajendra/deskForWork/scratch
                      Directory where temporary files will be stored.
```

encode2h5

Description:

```
Download and Convert ENCODE datasets to h5 files
================================================

It can be used to download and convert multiple datasets from ENCODE Experiment
matrix (https://www.encodeproject.org/matrix/?type=Experiment).
Presently, only bigWig files are downloaded and then converted.

At first search the datasets on https://www.encodeproject.org/matrix/?type=Experiment␣
↪.
Then click on download button on top of the page. A text file will be downloaded.
This text file can be used as input in this program. All bigWig files will be
downloaded and converted to gcMapExplorer compatible hdf5 format.

NOTE: At first a metafile is automatically downloaded and then files
      are filtered according to bigWig format and Assembly. Subsequently,
      if several replicates are present, only datasets with combined
      replicates are considered. In case if two replicates are present
      and combined replicates are not present, replicates will be combined with
      '-mtc/--method-to-combine' option.

NOTE: Because downloading and conversion might take very long time, it also
      generates a checkpoint file in the output directory. Therefore,
      in case of crash or abrupt exit, the process can be continued from the
      last file.
```

(continues on next page)

(continued from previous page)

```
Name of output files:

    (1) For ChIP-seq assay:
        a. signal-<Experiment target>-<Experiment accession>-<File accessions.h
        b. fold-<Experiment target>-<Experiment accession>-<File accessions>.h5

    (2) For RNA-seq:
        a. uniq-reads-<date>-<Experiment accession>-<File accessions>.h
        b. plus-uniq-reads-<date>-<Experiment accession>-<File accessions>.h
        c. minus-uniq-reads-<date>-<Experiment accession>-<File accessions>.h
        d. all-reads-<date>-<Experiment accession>-<File accessions>.h5
        e. plus-all-reads-<date>-<Experiment accession>-<File accessions>.h5
        f. minus-all-reads-<date>-<Experiment accession>-<File accessions>.h5
        g. signal-<date>-<Experiment accession>-<File accession>.h5

    (2) For DNase-seq:
        a. uniq-reads-signal-<date>-<Experiment accession>-<File accessions>.h
        b. raw-signal-<date>-<Experiment accession>-<File accessions>.h
        c. all-reads-signal-<date>-<Experiment accession>-<File accessions>.h
        d. signal-<date>-<Experiment accession>-<File accessions>.h5

    Note that name of cell-line is not included here. Therefore, use the
    directory name as a identifiers for cell-lines or species. The Experiment
    and File accession can be used to back-track about the dataset on ENCODE
    website.

Requirements
============
1) bigWigToWig : It converts binary bigWig file to ascii Wig file.
2) bigWigInfo : It fetches the information about chromosomes from bigWig file.

Both tools can be downloaded from http://hgdownload.cse.ucsc.edu/admin/exe/
for linux and Mac platform. However, these tools are not yet available for
Windows OS.

Path to these tools can be set using gcMapExplorer configure utility or can be
given with the command.

Resolutions
===========
By default, original data are downsampled to following resolutions:  '1kb',
'2kb', '4kb', '5kb', '8kb', '10kb', '20kb', '40kb', '80kb', '100kb', '160kb',
'200kb', '320kb', '500kb', '640kb',  and '1mb'.

The data are downsampled at this stage only to speed up the visualization
process as downsampling might slow down the interactive visualization.

Downsampling/Coarsening method
==============================
Presently, six methods are implemented:
1) min    -> Minimum value
2) max    -> Maximum value
3) amean  -> Arithmetic mean or average
4) hmean  -> Harmonic mean
5) gmean  -> Geometric mean
6) median -> Median
```

(continues on next page)

(continued from previous page)

```
All these methods are used by default.
See below help for "-dm/--downsample-method" option to change the methods.

To keep original 1 base resolution data
=======================================
By default, the output h5 file does not contain original 1-base resolution
data to reduce the file size. To keep the original data in h5 file, used
-ko/--keep-original flag.
```

Usage:

```
usage: gcMapExplorer encode2h5 [-h] [-i input.txt] [-amb hg19] [-asy ChIP-seq]
                               [-b2w bigWigToWig] [-binfo bigWigInfo]
                               [-r "List of Resolutions"]
                               [-dm "List of downsampling method"]
                               [-cmeth lzf] [-mtc mean] [-od outDir] [-ko]
                               [-wd /home/rajendra/deskForWork/scratch]
```

Optional arguments:

```
-h, --help            show this help message and exit
-i input.txt, --input input.txt
                      Input text file.
                      At first search the datasets on https://www.encodeproject.org/
↪matrix/?type=Experiment.
                      Then click on download button on top of the page. A text file␣
↪will be downloaded.
                      This text file can be used as input in this program.

-amb hg19, --assembly hg19
                       Name of reference genome.
                      Example: hg19, GRCh38 etc.

-asy ChIP-seq, --assay ChIP-seq
                       Name of assay.
                      Presently, four assays are implemented:
                      'ChIP-seq', 'RNA-seq', 'DNase-seq' and 'FAIRE-seq'.

-b2w bigWigToWig, --bigWigToWig bigWigToWig
                      Path to bigWigToWig tool.

                      This is not necessary when bigWigToWig path is already set␣
↪using gcMapExplorer
                      configure utility.

                      It can be downloaded from http://hgdownload.cse.ucsc.edu/admin/
↪exe/
                      for linux and Mac platform.

                      If it is not present in configuration file, the input path␣
↪should
                      be provided. It will be stored in configuration file for later␣
↪use.

-binfo bigWigInfo, --bigWigInfo bigWigInfo
                       Path to bigWigInfo tool.
```

(continues on next page)

(continued from previous page)

```
                      This is not necessary when bigWigInfo path is already set using␣
↪gcMapExplorer
                      configure utility.

                      It can be downloaded from http://hgdownload.cse.ucsc.edu/admin/
↪exe/
                      for linux and Mac platform.

                      If it is not present in configuration file, the input path␣
↪should
                      be provided. It will be stored in configuration file for later␣
↪use.

-r "List of Resolutions", --resolutions "List of Resolutions"
                      Additional input resolutions other than these resolutions: 1kb',
↪ '2kb',
                      '4kb', '5kb', '8kb', '10kb', '20kb', '40kb', '80kb', '100kb',
↪'160kb','200kb',
                      '320kb', '500kb', '640kb',  and '1mb'.

                      Resolutions should be provided in comma separated values. For␣
↪Example:
                      -r "25kb, 50kb, 75kb"

-dm "List of downsampling method", --downsample-method "List of downsampling method"
                      Methods to coarse or downsample the data for converting from 1-
↪base
                      to coarser resolutions. If this option is not provided, all six␣
↪methods (see
                      above) will be considered. User may use only subset of these␣
↪methods.
                      For example: -dm "max, amean" can be used for downsampling by␣
↪only these
                      two methods.

-cmeth lzf, --compression-method lzf
                      Data compression method in h5 file.
-mtc mean, --method-to-combine mean
                      Methods to combine data from more than two input file.␣
↪Presently, three
                      methods can be used: 'mean', 'max' and 'min' for average,␣
↪maximum and minimum
                      value, respectively.

-od outDir, --outDir outDir
                       Directory to save all h5 files. It is an essential input.

-ko, --keep-original  To copy original 1-base resolution data in h5 file. This will␣
↪increase the
                      file size significantly.

-wd /home/rajendra/deskForWork/scratch, --work-dir /home/rajendra/deskForWork/scratch
                      Directory where temporary files will be stored.
```

Convert using `gcMapExplorer` Python modules:

- bigWig file: *gcMapExplorer.lib.genomicsDataHandler.BigWigHandler*

- wig file: *gcMapExplorer.lib.genomicsDataHandler.WigHandler*
- bed file: *gcMapExplorer.lib.genomicsDataHandler.BEDHandler*
- ENCODE datasets : *gcMapExplorer.lib.genomicsDataHandler.EncodeDatasetsConverter*

3.1.5 Normalization of Hi-C maps

To normalize the Hi-C maps, several methods are implemented.

- **Iterative Correction** (IC)[1] This method normalize the raw contact map by removing biases from experimental procedure. This is an method of matrix balancing, however, in the normalized, sum of rows and columns are **not** equal to one.
- **Knight-Ruiz Matrix Balancing** (KR)[2] The Knight-Ruiz (KR) matrix balancing is a fast algorithm to normalize a symmetric matrix. A doubly stochastic matrix is obtained after this normalization. In this matrix, sum of rows and columns are equal to one.
- **Vanilla-Coverage** (VC)[3] This method was first used for inter-chromosomal map. Later it was used for intra-chromosomal map by Rao et al., 2014. This is a simple method where at first each element is divided by sum of respective row and subsequently divided by sum of respective column.
- **Median Contact Frequency Scaling** (MCFS) This method can be used to normalize contact map using Median contact values for particular distance between two locations/coordinates. At first, Median distance contact frequency for each distance is calculated. Subsequently, the observed contact frequency is divided by median contact frequency obtained for distance between the two locations.

To perform these normalizations, following tools are implemented:

cmapNormalizer - An application for Hi-C map normalization

Presently this application contains all the three normalization methods. It can be launch by following command:

```
gcMapExplorer cmapNormalizer
```

This interface contains in-built help (**Click on** `??` **button**) to understand the functionality of interfaces.

normKR - Normalization by Knight-Ruiz matrix balancing

The Knight-Ruiz (KR) matrix balancing is a fast algorithm to normalize a symmetric matrix. A doubly stochastic matrix is obtained after this normalization. In this matrix, sum of rows and columns are equal to one.

Usage:

```
usage: gcMapExplorer normKR [-h] [-i input.gcmap] [-fi gcmap]
                            [-o output.gcmap] [-fo gcmap] [-t 1e-12] [-m RAM]
                            [-vmax VMAX] [-vmin VMIN] [-mscm 20000] [-ptnd␣
↪99]
                            [-tdo 0.8] [-cmeth lzf]
                            [-wd /home/rajendra/deskForWork/scratch]
```

Optional arguments:

[1] Imakaev *et al*. Iterative correction of Hi-C data reveals hallmarks of chromosome organization. Nature Methods 9, 999–1003 (2012).
[2] Knight P and D. Ruiz. A fast algorithm for matrix balancing. IMA J Numer Anal (2013) 33 (3): 1029-1047.
[3] Lieberman-Aiden *et al*. Comprehensive mapping of long-range interactions reveals folding principles of the human genome. Science (2009) 326 : 289-293.

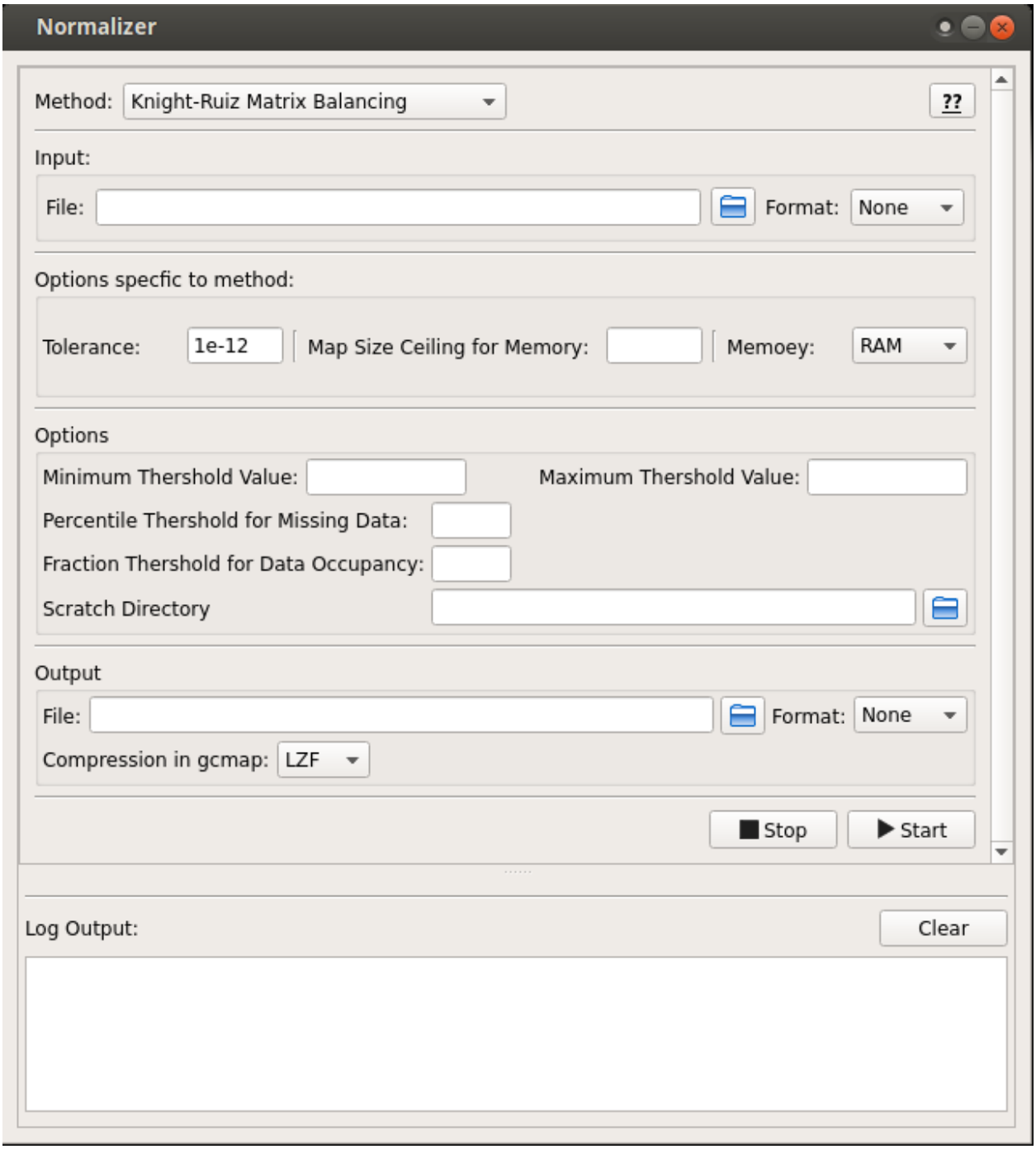


Fig. 15: Contact map normalization Interface

```
-h, --help            show this help message and exit
-i input.gcmap, --input input.gcmap
                       Input ccmap or gcmap file.

-fi gcmap, --format-input gcmap
                       Input format: 'ccmap' or 'gcmap'.

-o output.gcmap, --output output.gcmap
                       Output ccmap or gcmap file.

                      When input file is ccmap, output file can be gcmap. However,␣
↪when a input file
                      is gcmap, output file will be only in gcmap.

-fo gcmap, --format-output gcmap
                       Output format: 'ccmap' or 'gcmap'.

                      When input file is ccmap, output file can be gcmap. However,␣
↪when a input file
                      is gcmap, output file will be only in gcmap.

-t 1e-12, --tolerance 1e-12
                       Tolerance for matrix balancing.
                      Smaller tolerance increases accuracy in sums of rows and␣
↪columns.

-m RAM, --memory RAM   The memory used for calculation. Acceptable keywords are 'RAM'␣
↪or 'HDD'.

                      In case of RAM, memory is used for the calculation. In case of␣
↪Disk, all
                      intermediate steps will use DIsk Drive to store intermediate␣
↪data.

                      This option is ONLY VALID when input file is in ccmap format.

-vmax VMAX, --maximum-value VMAX
                       Minimum threshold value for normalization.
                      If contact frequency is less than or equal to this threshold␣
↪value,
                      this value is discarded during normalization.

-vmin VMIN, --minimum-value VMIN
                       Maximum threshold value for normalization.
                      If contact frequency is greater than or equal to this threshold␣
↪value,
                      this value is discarded during normalization.

-mscm 20000, --map-size-ceiling-for-memory 20000
                       Maximum size of contact map allowed for calculation using RAM.
                      If map size or shape is larger than this value, normalization␣
↪will be
                      performed using disk (HDD). This option is ONLY VALID when␣
↪input file is in
                      gcmap format.

-ptnd 99, --percentile-threshold-no-data 99
```

(continues on next page)

(continued from previous page)

```
                        It can be used to filter the map, where rows/columns with␣
↪largest numbers
                      of missing data can be discarded. Its value should be between 1␣
↪and 100.
                      This options discard the rows and columns which are above this␣
↪percentile.
                      For example: if this value is 99, those rows or columns will be␣
↪discarded which
                      contains larger than number of zeros (missing data) at 99␣
↪percentile.

                      To calculate percentile, all blank rows are removed, then in␣
↪all rows, number
                      of zeros are counted. Afterwards, number of zeros at input␣
↪percentile is
                      obtained. In next step, if a row contain number of zeros larger␣
↪than this
                      percentile value, the whole row and column is assigned to have␣
↪missing data.
                      This percentile indicates highest numbers of zeros (missing␣
↪data) in given
                      rows/columns.

-tdo 0.8, --threshold-data-occupancy 0.8
                        It can be used to filter the map, where rows/columns with␣
↪largest numbers
                      of missing data can be discarded.This ratio is:
                        (number of bins with data) / (total number of bins in the␣
↪given row/column)

                      For example: if -tdo = 0.8, then all rows containing more than␣
↪20% of
                      missing data will be discarded.

-cmeth lzf, --compression-method lzf
                      Data compression method for output gcmap file.
-wd /home/rajendra/deskForWork/scratch, --work-dir /home/rajendra/deskForWork/scratch
                      Directory where temporary files will be stored.
```

normVC - Normalization by Vanilla-Coverage

This method was first used for inter-chromosomal map. Later it was used for intra-chromosomal map by Rao et al., 2014 (http://dx.doi.org/10.1016/j.cell.2014.11.021). This is a simple method where at first each element is divided by sum of respective row and subsequently divided by sum of respective column.

For more details, see publications:

1. https://doi.org/10.1126/science.1181369
2. http://dx.doi.org/10.1016/j.cell.2014.11.021

Usage:

```
usage: gcMapExplorer normVC [-h] [-i input.gcmap] [-fi gcmap]
                            [-o output.gcmap] [-fo gcmap] [-sq] [-vmax VMAX]
                            [-vmin VMIN] [-ptnd 99] [-tdo 0.8] [-cmeth lzf]
                            [-wd /home/*****/scratch]
```

Optional arguments:

```
-h, --help            show this help message and exit
-i input.gcmap, --input input.gcmap
                      Input ccmap or gcmap file.

-fi gcmap, --format-input gcmap
                      Input format: 'ccmap' or 'gcmap'.

-o output.gcmap, --output output.gcmap
                      Output ccmap or gcmap file.

                     When input file is ccmap, output file can be gcmap. However, when␣
↪a input file
                     is gcmap, output file will be only in gcmap.

-fo gcmap, --format-output gcmap
                      Output format: 'ccmap' or 'gcmap'.

                     When input file is ccmap, output file can be gcmap. However, when␣
↪a input file
                     is gcmap, output file will be only in gcmap.

-sq, --sqroot           Square-root of normalized map

-vmax VMAX, --maximum-value VMAX
                      Minimum threshold value for normalization.
                     If contact frequency is less than or equal to this threshold␣
↪value,
                     this value is discarded during normalization.

-vmin VMIN, --minimum-value VMIN
                      Maximum threshold value for normalization.
                     If contact frequency is greater than or equal to this threshold␣
↪value,
                     this value is discarded during normalization.

-ptnd 99, --percentile-threshold-no-data 99
                      It can be used to filter the map, where rows/columns with␣
↪largest numbers
                     of missing data can be discarded. Its value should be between 1␣
↪and 100.
                     This options discard the rows and columns which are above this␣
↪percentile.
                     For example: if this value is 99, those rows or columns will be␣
↪discarded which
                     contains larger than number of zeros (missing data) at 99␣
↪percentile.

                     To calculate percentile, all blank rows are removed, then in all␣
↪rows, number
                     of zeros are counted. Afterwards, number of zeros at input␣
↪percentile is
                     obtained. In next step, if a row contain number of zeros larger␣
↪than this
                     percentile value, the whole row and column is assigned to have␣
↪missing data.
                     This percentile indicates highest numbers of zeros (missing data)␣
↪in given
```

(continues on next page)

(continued from previous page)

```
                    rows/columns.

-tdo 0.8, --threshold-data-occupancy 0.8
                     It can be used to filter the map, where rows/columns with␣
↪largest numbers
                    of missing data can be discarded.This ratio is:
                      (number of bins with data) / (total number of bins in the given␣
↪row/column)

                    For example: if -tdo = 0.8, then all rows containing more than 20
↪% of
                    missing data will be discarded.

-cmeth lzf, --compression-method lzf
                    Data compression method for output gcmap file.
-wd /home/*****/scratch, --work-dir /home/*****/scratch
                    Directory where temporary files will be stored.
```

normIC - Normalization by Iterative Correction

This method normalize the raw contact map by removing biases from experimental procedure. This is an method of matrix balancing, however, in the normalized, sum of rows and columns are **not** equal to one.

Usage:

```
usage: gcMapExplorer normIC [-h] [-i input.gcmap] [-fi gcmap]
                            [-o output.gcmap] [-fo gcmap] [-t 0.0001]
                            [-vmax VMAX] [-vmin VMIN] [-c 500] [-ptnd 99]
                            [-tdo 0.8] [-cmeth lzf]
                            [-wd /home/rajendra/deskForWork/scratch]
```

Optional arguments:

```
-h, --help            show this help message and exit
-i input.gcmap, --input input.gcmap
                       Input ccmap or gcmap file.

-fi gcmap, --format-input gcmap
                       Input format: 'ccmap' or 'gcmap'.

-o output.gcmap, --output output.gcmap
                       Output ccmap or gcmap file.

                      When input file is ccmap, output file can be gcmap. However,␣
↪when a input file
                      is gcmap, output file will be only in gcmap.

-fo gcmap, --format-output gcmap
                       Input format: 'ccmap' or 'gcmap'.

                      When input file is ccmap, output file can be gcmap. However,␣
↪when a input file
                      is gcmap, output file will be only in gcmap.

-t 0.0001, --tolerance 0.0001
```

(continues on next page)

(continued from previous page)

```
                        Tolerance for matrix balancing.
                       Smaller tolerance increases accuracy in sums of rows and␣
↪columns.

-vmax VMAX, --maximum-value VMAX
                        Minimum threshold value for normalization.
                       If contact frequency is less than or equal to this threshold␣
↪value,
                       this value is discarded during normalization.

-vmin VMIN, --minimum-value VMIN
                        Maximum threshold value for normalization.
                       If contact frequency is greater than or equal to this threshold␣
↪value,
                       this value is discarded during normalization.

-c 500, --iteration 500
                        Number of iteration to stop the normalization.

-ptnd 99, --percentile-threshold-no-data 99
                        It can be used to filter the map, where rows/columns with␣
↪largest numbers
                       of missing data can be discarded. Its value should be between 1␣
↪and 100.
                       This options discard the rows and columns which are above this␣
↪percentile.
                       For example: if this value is 99, those rows or columns will be␣
↪discarded which
                       contains larger than number of zeros (missing data) at 99␣
↪percentile.

                       To calculate percentile, all blank rows are removed, then in␣
↪all rows, number
                       of zeros are counted. Afterwards, number of zeros at input␣
↪percentile is
                       obtained. In next step, if a row contain number of zeros larger␣
↪than this
                       percentile value, the whole row and column is assigned to have␣
↪missing data.
                       This percentile indicates highest numbers of zeros (missing␣
↪data) in given
                       rows/columns.

-tdo 0.8, --threshold-data-occupancy 0.8
                        It can be used to filter the map, where rows/columns with␣
↪largest numbers
                       of missing data can be discarded.This ratio is:
                         (number of bins with data) / (total number of bins in the␣
↪given row/column)

                       For example: if -tdo = 0.8, then all rows containing more than␣
↪20% of
                       missing data will be discarded.

-cmeth lzf, --compression-method lzf
                       Data compression method for output gcmap file.
-wd /home/rajendra/deskForWork/scratch, --work-dir /home/rajendra/deskForWork/scratch
```

(continues on next page)

(continued from previous page)

```
                        Directory where temporary files will be stored.
```

normMCFS - Scale maps using Median/Mean Contact Frequency

This method can be used to normalize contact map with expected values. These expected values could be either Median or Average contact values for particular distance between two locations/coordinates. At first, Median/Average distance contact frequency for each distance is calculated. Subsequently, the observed contact frequency is either divided ('o/e') or subtracted ('o-e') by median/average contact frequency obtained for distance between the two locations.

Usage:

```
usage: gcMapExplorer normMCFS [-h] [-i input.gcmap] [-fi gcmap]
                              [-o output.gcmap] [-fo gcmap] [-s median]
                              [-vmax VMAX] [-vmin VMIN] [-st o/e] [-ptnd 99]
                              [-tdo 0.8] [-cmeth lzf]
                              [-wd /home/rajendra/deskForWork/scratch]
```

Optional arguments:

```
-h, --help            show this help message and exit
-i input.gcmap, --input input.gcmap
                       Input ccmap or gcmap file.

-fi gcmap, --format-input gcmap
                       Input format: 'ccmap' or 'gcmap'.

-o output.gcmap, --output output.gcmap
                       Output ccmap or gcmap file.

                      When input file is ccmap, output file can be gcmap. However,␣
↪when a input file
                      is gcmap, output file will be only in gcmap.

-fo gcmap, --format-output gcmap
                       Input format: 'ccmap' or 'gcmap'.

                      When input file is ccmap, output file can be gcmap. However,␣
↪when a input file
                      is gcmap, output file will be only in gcmap.

-s median, --stats median
                       Statistics to be considered for scaling.
                      It may be either "mean" or "median". By default, it is "median".

-vmax VMAX, --maximum-value VMAX
                       Minimum threshold value for normalization.
                      If contact frequency is less than or equal to this threshold␣
↪value,
                      this value is discarded during normalization.

-vmin VMIN, --minimum-value VMIN
                       Maximum threshold value for normalization.
                      If contact frequency is greater than or equal to this threshold␣
↪value,
                      this value is discarded during normalization.
```

(continues on next page)

(continued from previous page)

```
-st o/e, --stype o/e   Type of scaling.
                       It may be either 'o/e' or 'o-e'. In case of 'o/e',
                       Observed/Expected will be calculated while (Observed - Expected)
                       will be calculated for 'o-e'.

-ptnd 99, --percentile-threshold-no-data 99
                        It can be used to filter the map, where rows/columns with␣
↪largest numbers
                       of missing data can be discarded. Its value should be between 1␣
↪and 100.
                       This options discard the rows and columns which are above this␣
↪percentile.
                       For example: if this value is 99, those rows or columns will be␣
↪discarded which
                       contains larger than number of zeros (missing data) at 99␣
↪percentile.

                       To calculate percentile, all blank rows are removed, then in␣
↪all rows, number
                       of zeros are counted. Afterwards, number of zeros at input␣
↪percentile is
                       obtained. In next step, if a row contain number of zeros larger␣
↪than this
                       percentile value, the whole row and column is assigned to have␣
↪missing data.
                       This percentile indicates highest numbers of zeros (missing␣
↪data) in given
                       rows/columns.

-tdo 0.8, --threshold-data-occupancy 0.8
                        It can be used to filter the map, where rows/columns with␣
↪largest numbers
                       of missing data can be discarded.This ratio is:
                         (number of bins with data) / (total number of bins in the␣
↪given row/column)

                       For example: if -tdo = 0.8, then all rows containing more than␣
↪20% of
                       missing data will be discarded.

-cmeth lzf, --compression-method lzf
                       Data compression method for output gcmap file.
-wd /home/rajendra/deskForWork/scratch, --work-dir /home/rajendra/deskForWork/scratch
                       Directory where temporary files will be stored.
```

References

3.1.6 Frequently asked questions

gcMapExplorer browser:

- *How to solve - When multiple maps are loaded, tick-labels are overlapped?*

gcMapExplorer browser

How to solve - When multiple maps are loaded, tick-labels are overlapped?

Several default settings could be change to resolve this issue.

- By changing page size and orientation: When page size is increased, the space between maps also increases. Page size can be changed to either standard A4, A3 A2 and A1 size or any custom size through menu bar.
- By changing axis properties: Properties such as font-size of axis-labels, tick intervals, labels rotation etc can be changed to increase the visibility of labels. Axis properties dialog can be accessed by right clicking on any map or plot. This dialog contains all necessary options to change the display style of axis elements.

By combining these options, display quality of axis-labels can be changed in case of multiple maps.

3.1.7 Download Example Datasets

Contact map - gcmap/ccmap format

Followings are the `gcmap` format files available for the download. These data were taken from , which was published along with this .

GM12878 Cell Types

- Raw Observed - Finest 10 kb resolution - LZF compressed :
- Raw Observed - Finest 10 kb resolution - GZIP compressed :
- KR Balanced - Finest 10 kb resolution - LZF compressed :
- KR Balanced - Finest 10 kb resolution - GZIP compressed :
- Iteratively Corrected - Finest 10 kb resolution - LZF compressed :
- Iteratively Corrected - Finest 10 kb resolution - GZIP compressed :
- Average Contacts by Distance Normalized - Finest 10 kb resolution - LZF compressed :
- Average Contacts by Distance Normalized - Finest 10 kb resolution - GZIP compressed :

K562 Cell Types

- Raw Observed - Finest 10 kb resolution - LZF compressed :
- Raw Observed - Finest 10 kb resolution - GZIP compressed :
- KR Balanced - Finest 10 kb resolution - LZF compressed :
- KR Balanced - Finest 10 kb resolution - GZIP compressed :
- Iteratively Corrected - Finest 10 kb resolution - LZF compressed :
- Iteratively Corrected - Finest 10 kb resolution - GZIP compressed :
- Average Contacts by Distance Normalized - Finest 10 kb resolution - LZF compressed :
- Average Contacts by Distance Normalized - Finest 10 kb resolution - GZIP compressed :

Genomic datasets - h5 format

Followings dataset in h5 formats are available for the download. These were downloaded and converted from .

GM12878 Cell Types

- RNA-seq :
- Histone modifications H3K9ac :
- Histone modifications H3K27ac :
- Histone modifications H3K36me3 :
- Histone modifications H3K79me2 :
- Histone modifications H3K4me3 :
- CTCF Binding Site Raw Signal :

K562 Cell Types

- RNA-seq :
- Histone modifications H3K9ac :
- Histone modifications H3K27ac :
- Histone modifications H3K36me3 :
- Histone modifications H3K79me2 :
- Histone modifications H3K4me3 :
- CTCF Binding Site Raw Signal :

3.1.8 Summary of Python Modules

config module

updateConfig(section, option, value)	Update configuration file
getConfig()	To get the present configuration.
printConfig()	Print configuration file
cleanScratch()	Clean scratch directory.

ccmap module

ccmap.CCMAP.copy([fill])	To create a new copy of CCMAP object
ccmap.CCMAP.get_ticks([binsize])	To get xticks and yticks for the matrix
ccmap.CCMAP.make_readable()	Enable reading the numpy array binary file.
ccmap.CCMAP.make_unreadable()	Disable reading the numpy array binary file from local file system
ccmap.CCMAP.make_writable()	Create new numpy array binary file on local file system and enable reading/writing to this file

Continued on next page

Table 8 – continued from previous page

ccmap.CCMAP.make_editable()	Enable editing numpy array binary file
ccmap.jsonify(ccMapObj)	Changes data type of attributes in CCMAP object for .
ccmap.dejsonify(ccMapObj[, json_dict])	Change back the data type of attributes in CCMAP object.
ccmap.save_ccmap(ccMapObj, outfile[, ...])	Save CCMAP object on file
ccmap.load_ccmap(infile[, workDir])	Load CCMAP object from an input file
ccmap.export_cmap(ccmap, outfile[, ...])	To export `.ccmap` as text file
ccmap.checkCCMapObjectOrFile(ccMap[, workDir])	Check whether ccmap is a object or file
ccmap.downSampleCCMap(cmap[, level, method, ...])	Downsample or coarsen the contact map
ccmap.getOutputShapeFor2DMapDownsampling(...)	Helper function to determine output shape of map for downsampling
ccmap.downSample2DMap(inMatrix[, outMatrix, ...])	Downsample or coarsen the matrix

ccmapHelpers module

ccmapHelpers.MemoryMappedArray	Convenient wrapper for numpy memory mapped array file
ccmapHelpers.MemoryMappedArray.copy	Copy this numpy memory mapped array and generate new
ccmapHelpers.MemoryMappedArray.copy_from	Copy values from source `MemoryMappedArray`
ccmapHelpers.MemoryMappedArray.copy_to	Copy values to destination `MemoryMappedArray`
ccmapHelpers.get_nonzeros_index(matrix[, ...])	To get a numpy array of bool values for all rows/columns which have **NO** missing data
ccmapHelpers.remove_zeros(matrix[, ...])	To remove rows/columns with missing data (zero values)

util module

util.resolutionToBinsize(resolution)	Return the bin size from the resolution unit
util.binsizeToResolution(binsize)	Return the resolution unit from the bin size
util.sorted_nicely(inputList)	Sorts the given given list in the way that is expected.
util.locate_significant_digit_after_decimal(...)	Get location at which significant digit start after decimal
util.kth_diag_indices(k, a)	Get diagonal indices of 2D array 'a' offset by 'k'
util.detectOutliers1D(points[, thresh])	Returns a boolean array with `True` if points are outliers and False otherwise.
util.getRandomName([size, chars])	Random name generator
util.MapNotFoundError(value)	
util.ResolutionNotFoundError(value)	

gcmap module

gcmap.GCMAP(hdf5[, mapName, chromAtX, ...])	To access Genome wide contact map.
gcmap.GCMAP.checkMapExist([mapName, ...])	Check if a map is exist in the file

Continued on next page

Table 11 – continued from previous page

gcmap.GCMAP.changeMap([mapName, chromAtX, ...])	Change the map for another chromosome
gcmap.GCMAP.changeResolution(resolution)	Try to change contact map of a given resolution.
gcmap.GCMAP.toFinerResolution()	Try to change contact map to next finer resolution
gcmap.GCMAP.toCoarserResolution()	Try to change contact map to next coarser resolution
gcmap.GCMAP.loadSmallestMap([resolution])	Load smallest sized contact map
gcmap.GCMAP.genMapNameList([sortBy])	Generate list of contact maps available in gcmap file
gcmap.GCMAP.performDownSampling([method])	Downsample recursively and store the maps
gcmap.GCMAP.downsampleMapToResolution(resolution)	Downsample the current map to a particular resolution
gcmap.GCMAP.downsampleAllMapToResolution()	Downsample all maps to a particular resolution
gcmap.loadGCMapAsCCMap(filename[, mapName, ...])	Load a map from gcmap file as a *gcMapExplorer.lib.ccmap.CCMAP*.
gcmap.addCCMap2GCMap(cmap, filename[, ...])	Add *gcMapExplorer.lib.ccmap.CCMAP* to a gcmap file
gcmap.changeGCMapCompression(infile, ...[, ...])	Change compression method in GCMAP file

importer module

importer.CooMatrixHandler([inputFiles, ...])	To import ccmap from files similar to sparse matrix in Coordinate (COO) format
importer.CooMatrixHandler.save_ccmaps([...])	To Save all Hi-C maps
importer.CooMatrixHandler.save_gcmap(outputFile)	To Save all Hi-C maps as a gcmap file
importer.CooMatrixHandler.setLabels(xlabels, ...)	To set xlabels and ylabels for contact maps
importer.CooMatrixHandler.setOutputFileList(...)	To set list of output files
importer.PairCooMatrixHandler(inputFile[, ...])	To import ccmap from files similar to paired sparse matrix Coordinate (COO) format
importer.PairCooMatrixHandler.setGCMapOptions([...])	Set options for output gcmap file
importer.PairCooMatrixHandler.runConversion()	Perform conversion and save to ccmap and/or gcmap file.
importer.HomerInputHandler([inputFiles, ...])	To import ccmap from Hi-C maps generated by HOMER
importer.HomerInputHandler.save_ccmaps(outdir)	Import and save ccmap file
importer.HomerInputHandler.save_gcmap(outputFile)	To Save all Hi-C maps as a gcmap file
importer.BinsNContactFilesHandler(binFile, ...)	To import Hi-C map from bin and contact file in list format
importer.BinsNContactFilesHandler.save_ccmaps(outdir)	Import and save ccmap file
importer.BinsNContactFilesHandler.save_gcmap(...)	To Save all Hi-C maps as a gcmap file
importer.gen_map_from_locations_value(i, j, ...)	To generate CCMAP object from three lists – i, j, value

normalizer module

normalizer.NormalizeKnightRuizOriginal(ccMapObj)	Original Knight-Ruiz algorithm for matrix balancing
normalizer.normalizeCCMapByKR(ccMap[, ...])	Normalize a ccmap using Knight-Ruiz matrix balancing method.
normalizer.normalizeGCMapByKR(...[, ...])	Normalize a gcmap using Knight-Ruiz matrix balancing method.
normalizer.normalizeCCMapByIC(ccMap[, tol, ...])	Normalize a ccmap by Iterative correction method
normalizer.normalizeGCMapByIC(...[, vmin, ...])	Normalize a gcmap using Iterative Correction.
normalizer.normalizeCCMapByMCFS(ccMap[, ...])	Scale ccmap using Median Contact Frequency
normalizer.normalizeGCMapByMCFS(...[, ...])	Scale all maps in gcmap using Median Contact Frequency
normalizer.normalizeCCMapByVCNorm(ccMap[, ...])	Normalize ccmap using Vanilla-Coverage method
normalizer.normalizeGCMapByVCNorm(...[, ...])	Normalize all maps using Vanilla-Coverage method

cmstats module

cmstats.correlateCMaps(ccMapObjOne, ccMapObjTwo)	To calculate correlation between two Hi-C maps
cmstats.correlateGCMaps(gcmapOne, gcmapTwo)	To calculate correlation between common Hi-C maps from two gcmap files
cmstats.getAvgContactByDistance(ccmaps[, ...])	To calculate average contact as a function of distance

corrMatrix module

corrMatrix.calculateCorrMatrix(...[, maskvalue])	Calculate correlation matrix from a 2D numpy array.
corrMatrix.calculateCovMatrix(...[, maskvalue])	Calculate covariance matrix from a 2D numpy array.
corrMatrix.calculateCorrelation(ndarray x, ...)	Calculate correlation between two 1D numpy array.
corrMatrix.calculateCovariance(ndarray x, ...)	Calculate covariance between two 1D numpy array.
corrMatrix.calculateCorrMatrixForCCMap(...)	Calculate correlation matrix of a contact map.
corrMatrix.calculateCorrMatrixForGCMaps(...)	Calculate Correlation matrix for all maps present in input gcmap file It calculates correlation between all rows and columns of contact map.

statDist module

statDist.calculateTransitionProbabilityMatrix(A)	Core function to calculate transition probability matrix.
statDist.transitionProbabilityMatrixForCCMap(...)	To calculate transition probability matrix.

Continued on next page

Table 16 – continued from previous page

statDist.transitionProbabilityMatrixForGCMap()	To calculate transition matrices using a gcmap file.
statDist.statDistrByEigenDecompForCCMap()	Calculate stationary distribution from a ccmap file or object.
statDist.statDistrByEigenDecompForGCMap()	Calculate stationary distribution using transition matrices from gcmap file for given resolution.
statDist.stationaryDistributionByEigenDecomp()	To calculate stationary distribution from probability transition matrix.

genomicsDataHandler module

genomicsDataHandler.HDF5Handler(filename[, ...])	Handler for genomic data HDF5 file.
genomicsDataHandler.HDF5Handler.setTitle(title)	Set title of the dataset
genomicsDataHandler.HDF5Handler.getChromList()	To get list of all chromosomes present in hdf5 file
genomicsDataHandler.HDF5Handler.getResolutionList(chrom)	To get all resolutions for given chromosome from hdf5 file
genomicsDataHandler.HDF5Handler.getDataNameList(...)	List of all available arrays by respective coarse method name for given chromosome and resolution
genomicsDataHandler.HDF5Handler.buildDataTree()	Build data dictionary from the input hdf5 file
genomicsDataHandler.BigWigHandler(filenames)	To handle bigWig files and to convert it to h5 file
genomicsDataHandler.BigWigHandler.getBigWigInfo()	Retrieve chromosome names and their sizes
genomicsDataHandler.BigWigHandler.bigWigtoWig([...])	To generate Wig file
genomicsDataHandler.BigWigHandler.saveAsH5(...)	Save data to h5 file.
genomicsDataHandler.WigHandler(filenames[, ...])	To convert Wig files to hdf5 file
genomicsDataHandler.WigHandler.parseWig()	To parse Wig files
genomicsDataHandler.WigHandler.setChromosome(...)	Set the target chromosome for reading and extracting from wig file
genomicsDataHandler.WigHandler.saveAsH5(hdf5Out)	To convert Wig files to hdf5 file
genomicsDataHandler.WigHandler.getRawWigDataAsDictionary([...])	To get a entire dictionary of data from Wig file
genomicsDataHandler.BEDHandler(filenames[, ...])	To convert BED files to hdf5/h5 file
genomicsDataHandler.BEDHandler.parseBed()	To parse bed files
genomicsDataHandler.BEDHandler.setChromosome(...)	Set the target chromosome for reading and extracting from bed file
genomicsDataHandler.BEDHandler.saveAsH5(hdf5Out)	To convert bed files to hdf5 file
genomicsDataHandler.EncodeDatasetsConverter(...)	Download and convert datasets from ENCODE Experiments matrix

Continued on next page

Table 17 – continued from previous page

genomicsDataHandler.EncodeDatasetsConverter.saveAsH5(outDir)	Download the files and convert to gcMapExplorer compatible hdf5 file.
genomicsDataHandler.TextFileHandler(...[, ...])	To import a genomic data from column text file format
genomicsDataHandler.TextFileHandler.readData()	Read data from input file
genomicsDataHandler.TempNumpyArrayFiles([...])	To handle temporary numpy array files
genomicsDataHandler.TempNumpyArrayFiles.updateArraysByBigWig(...)	Update/resize all array files using given bigWig file
genomicsDataHandler.TempNumpyArrayFiles.updateArraysByChromSize(...)	Update/resize an array file using given chromosome and its size
genomicsDataHandler.TempNumpyArrayFiles.addChromSizeInfo(...)	Update chromosome sizes using new bigWig file
genomicsDataHandler.TempNumpyArrayFiles.generateAllTempNumpyFiles()	Generate all memory mapped numpy array files
genomicsDataHandler.TempNumpyArrayFiles.generateTempNumpyFile(key)	Generate a memory mapped numpy array file
genomicsDataHandler.TempNumpyArrayFiles.fillAllArraysWithZeros()	Fill all arrays with zeros.

3.1.9 Contents

Please download tutorial files

How to Import external HI-C map data?

1. From a matrix coordinate format text file

As shown below in example, in this format, first and second column is location on chromosome and third column is the respective value:

```
20000000    20000000    2692.0
20000000    20100000    885.0
20100000    20100000    6493.0
20000000    20200000    15.0
20100000    20200000    52.0
20200000    20200000    2.0
20000000    20300000    18.0
20100000    20300000    40.0
.
.
.
.
```

(continues on next page)

(continued from previous page)

```
.
.
```

Hi-C maps data with the above format are available with this article and can be downlaoded here.

At first, we import gcMapExplorer.lib module

All neccessary modules are avaiable in gcMapExplorer.lib module

```
In [1]: from gcMapExplorer import lib as gmlib
```

This module has methods to read and save ccmap file as shown below in the exmaple.

Remove old files if any present in output directories

```
In [2]: %%bash

        for f in ./cmaps/binContact/*; do
            [ -e "$f" ] && rm $f
        done

        for f in ./cmaps/CooMatrix/*; do
            [ -e "$f" ] && rm $f
        done

        for f in ./cmaps/homer/*; do
            [ -e "$f" ] && rm $f
        done
```

We read Hi-C file as follows:

```
In [3]: cooReader = gmlib.importer.CooMatrixHandler('./data/CooMatrixFormat/chr15_100kb.RAWobserved')
```

See also:

Function *gcMapExplorer.lib.importer.CooMatrixHandler()* for more details.

Now, save the Hi-C map as ccmap:

We save imported Hi-C map in `cmaps` directory as `chr15_100kb_Raw_from_text.ccmap` file. To reduce the storage memory, map file is compressed in gzip format.

```
In [4]: cooReader.save_ccmaps('cmaps/CooMatrix/chr15_100kb_Raw_from_text.ccmap', xlabels='chr15')
        del cooReader       # Delete object and generated any temporary files
```

```
INFO:CooMatrixHandler: Reading file: [./data/CooMatrixFormat/chr15_100kb.RAWobserved]...


INFO:CooMatrixHandler:     ... Finished reading file: [./data/CooMatrixFormat/chr15_100kb.RAWobserved
INFO:genMapFromLists: Total number of data in input file: 318258
INFO:genMapFromLists:Minimum base-pair: 20000000 and Maximum base-pair: 102500000 are present in inpu
INFO:genMapFromLists:Shape of overall map: (1026, 1026)

INFO:save_ccmap: Saving ccmap to file [cmaps/CooMatrix/chr15_100kb_Raw_from_text.ccmap] and [/home/ra
INFO:save_ccmap: Compressing [/home/rajendra/workspace/genome_3d_organization/tutorials_modules/cmaps
INFO:save_ccmap:       Finished!!!
```

See also:

- Function *gcMapExplorer.lib.importer.CooMatrixHandler.save_ccmaps()* for more details.

- Function *gcMapExplorer.lib.ccmap.save_ccmap()* for more details.

Importing from a tar archive

If a Hi-C map data file is present inside a tar archive, the map file can be directly imported as follows:

```
In [5]: tarfile = 'data/CooMatrixFormat/100kb_resolution_intrachromosomal.tar.gz'            # Input
        mapfile = '100kb_resolution_intrachromosomal/chr15/MAPQGE30/chr15_100kb.RAWobserved' # Map fi
        cooReader = gmlib.importer.CooMatrixHandler(mapfile, tarfile)
```

where, `data/100kb_resolution_intrachromosomal.tar.gz` is input tar archive and `100kb_resolution_intrachromosomal/chr15/MAPQGE30/chr15_100kb.RAWobserved` is a map file inside the archive.

Now, save the Hi-C map as ccmap: as already shown above.

```
In [6]: cooReader.save_ccmaps('cmaps/CooMatrix/chr15_100kb_raw_from_archive.ccmap', xlabels='chr15')

INFO:CooMatrixHandler: Extracting-Reading [100kb_resolution_intrachromosomal/chr15/MAPQGE30/chr15_100


INFO:CooMatrixHandler:   ...Finished extracting and reading [100kb_resolution_intrachromosomal/chr15/
INFO:genMapFromLists: Total number of data in input file: 318258
INFO:genMapFromLists:Minimum base-pair: 20000000 and Maximum base-pair: 102500000 are present in inpu
INFO:genMapFromLists:Shape of overall map: (1026, 1026)

INFO:save_ccmap: Saving ccmap to file [cmaps/CooMatrix/chr15_100kb_raw_from_archive.ccmap] and [/home
INFO:save_ccmap: Compressing [/home/rajendra/workspace/genome_3d_organization/tutorials_modules/cmaps
INFO:save_ccmap:       Finished!!!
```

Convert several files from a tar archive

`100kb_resolution_intrachromosomal.tar.gz` file contains six Hi-C map data files. Through a `for` loop, these files can be imported and saved. Path to these files in the tar archive are as follows:

```
100kb_resolution_intrachromosomal/chr1/MAPQGE30/chr1_100kb.RAWobserved
100kb_resolution_intrachromosomal/chr5/MAPQGE30/chr5_100kb.RAWobserved
100kb_resolution_intrachromosomal/chr15/MAPQGE30/chr15_100kb.RAWobserved
100kb_resolution_intrachromosomal/chr20/MAPQGE30/chr20_100kb.RAWobserved
100kb_resolution_intrachromosomal/chr21/MAPQGE30/chr21_100kb.RAWobserved
100kb_resolution_intrachromosomal/chr22/MAPQGE30/chr22_100kb.RAWobserved
```

These file names have a pattern, and we utilize this pattern to form a name inside `for` loop.

```
In [7]: tarfile = 'data/CooMatrixFormat/100kb_resolution_intrachromosomal.tar.gz'

        chroms = [1, 5, 15, 20, 21, 22]      # List of chromosomes

        # Loop for each chromosome
        inputFileList = []
        outputFileList = []
        xlabels = []
        for chrom in chroms:
            mapfile = '100kb_resolution_intrachromosomal/chr{0}/MAPQGE30/chr{0}_100kb.RAWobserved' .f
            inputFileList.append(mapfile)
```

```
            output_file = 'cmaps/CooMatrix/chr{0}_100kb_RawObserved.ccmap' .format(chrom)      # Outpu
            outputFileList.append(output_file)

            xlabels.append( 'chr{0}'.format(chrom) )

        cooReader = gmlib.importer.CooMatrixHandler(inputFileList, tarfile)
        cooReader.save_ccmaps(outputFileList, xlabels=xlabels)
```

```
INFO:CooMatrixHandler: Extracting-Reading [100kb_resolution_intrachromosomal/chr1/MAPQGE30/chr1_100k


INFO:CooMatrixHandler:   ...Finished extracting and reading [100kb_resolution_intrachromosomal/chr1/M
INFO:genMapFromLists: Total number of data in input file: 2435300
INFO:genMapFromLists:Minimum base-pair: 0 and Maximum base-pair: 249200000 are present in input data
INFO:genMapFromLists:Shape of overall map: (2493, 2493)

INFO:save_ccmap: Saving ccmap to file [cmaps/CooMatrix/chr1_100kb_RawObserved.ccmap] and [/home/rajen
INFO:save_ccmap: Compressing [/home/rajendra/workspace/genome_3d_organization/tutorials_modules/cmaps
INFO:save_ccmap:       Finished!!!

INFO:CooMatrixHandler: Extracting-Reading [100kb_resolution_intrachromosomal/chr5/MAPQGE30/chr5_100k


INFO:CooMatrixHandler:   ...Finished extracting and reading [100kb_resolution_intrachromosomal/chr5/M
INFO:genMapFromLists: Total number of data in input file: 1533205
INFO:genMapFromLists:Minimum base-pair: 0 and Maximum base-pair: 180800000 are present in input data
INFO:genMapFromLists:Shape of overall map: (1809, 1809)

INFO:save_ccmap: Saving ccmap to file [cmaps/CooMatrix/chr5_100kb_RawObserved.ccmap] and [/home/rajen
INFO:save_ccmap: Compressing [/home/rajendra/workspace/genome_3d_organization/tutorials_modules/cmaps
INFO:save_ccmap:       Finished!!!

INFO:CooMatrixHandler: Extracting-Reading [100kb_resolution_intrachromosomal/chr15/MAPQGE30/chr15_100


INFO:CooMatrixHandler:   ...Finished extracting and reading [100kb_resolution_intrachromosomal/chr15/
INFO:genMapFromLists: Total number of data in input file: 318258
INFO:genMapFromLists:Minimum base-pair: 20000000 and Maximum base-pair: 102500000 are present in inpu
INFO:genMapFromLists:Shape of overall map: (1026, 1026)

INFO:save_ccmap: Saving ccmap to file [cmaps/CooMatrix/chr15_100kb_RawObserved.ccmap] and [/home/raje
INFO:save_ccmap: Compressing [/home/rajendra/workspace/genome_3d_organization/tutorials_modules/cmaps
INFO:save_ccmap:       Finished!!!

INFO:CooMatrixHandler: Extracting-Reading [100kb_resolution_intrachromosomal/chr20/MAPQGE30/chr20_100


INFO:CooMatrixHandler:   ...Finished extracting and reading [100kb_resolution_intrachromosomal/chr20/
INFO:genMapFromLists: Total number of data in input file: 179488
INFO:genMapFromLists:Minimum base-pair: 0 and Maximum base-pair: 62900000 are present in input data
INFO:genMapFromLists:Shape of overall map: (630, 630)

INFO:save_ccmap: Saving ccmap to file [cmaps/CooMatrix/chr20_100kb_RawObserved.ccmap] and [/home/raje
INFO:save_ccmap: Compressing [/home/rajendra/workspace/genome_3d_organization/tutorials_modules/cmaps
INFO:save_ccmap:       Finished!!!

INFO:CooMatrixHandler: Extracting-Reading [100kb_resolution_intrachromosomal/chr21/MAPQGE30/chr21_100
INFO:CooMatrixHandler:   ...Finished extracting and reading [100kb_resolution_intrachromosomal/chr21/
INFO:genMapFromLists: Total number of data in input file: 60664
```

```
INFO:genMapFromLists:Minimum base-pair: 9400000 and Maximum base-pair: 48100000 are present in input
INFO:genMapFromLists:Shape of overall map: (482, 482)



INFO:save_ccmap: Saving ccmap to file [cmaps/CooMatrix/chr21_100kb_RawObserved.ccmap] and [/home/raje
INFO:save_ccmap: Compressing [/home/rajendra/workspace/genome_3d_organization/tutorials_modules/cmaps
INFO:save_ccmap:       Finished!!!

INFO:CooMatrixHandler: Extracting-Reading [100kb_resolution_intrachromosomal/chr22/MAPQGE30/chr22_100
INFO:CooMatrixHandler:   ...Finished extracting and reading [100kb_resolution_intrachromosomal/chr22/
INFO:genMapFromLists: Total number of data in input file: 59429
INFO:genMapFromLists:Minimum base-pair: 16000000 and Maximum base-pair: 51200000 are present in input
INFO:genMapFromLists:Shape of overall map: (513, 513)



INFO:save_ccmap: Saving ccmap to file [cmaps/CooMatrix/chr22_100kb_RawObserved.ccmap] and [/home/raje
INFO:save_ccmap: Compressing [/home/rajendra/workspace/genome_3d_organization/tutorials_modules/cmaps
INFO:save_ccmap:       Finished!!!
```

Now, in `cmaps/CooMatrix` directory, all files from archive are saved. These files can be used either with browser to visualize or for further analysis.

Convert to `gcmap` file

The contact map files can be converted to `gcmap` format file.

See also:

- Function *gcMapExplorer.lib.importer.CooMatrixHandler.save_gcmap()* for more details.

```
In [8]: tarfile = 'data/CooMatrixFormat/100kb_resolution_intrachromosomal.tar.gz'

        chroms = [1, 5, 15, 20, 21, 22]      # List of chromosomes

        # Loop for each chromosome
        inputFileList = []
        outputFileList = []
        xlabels = []
        for chrom in chroms:
            mapfile = '100kb_resolution_intrachromosomal/chr{0}/MAPQGE30/chr{0}_100kb.RAWobserved' .f
            inputFileList.append(mapfile)

            output_file = 'cmaps/CooMatrix/chr{0}_100kb_RawObserved.ccmap' .format(chrom)      # Outpu
            outputFileList.append(output_file)

            xlabels.append( 'chr{0}'.format(chrom) )

        cooReader = gmlib.importer.CooMatrixHandler(inputFileList, tarfile)
        cooReader.save_gcmap('cmaps/CooMatrix/rawObserved_100kb.gcmap', xlabels=xlabels, coarsingMeth
```

```
INFO:CooMatrixHandler: Extracting-Reading [100kb_resolution_intrachromosomal/chr1/MAPQGE30/chr1_100kb



INFO:CooMatrixHandler:   ...Finished extracting and reading [100kb_resolution_intrachromosomal/chr1/M
INFO:genMapFromLists: Total number of data in input file: 2435300
INFO:genMapFromLists:Minimum base-pair: 0 and Maximum base-pair: 249200000 are present in input data
```

```
INFO:genMapFromLists:Shape of overall map: (2493, 2493)

INFO:addCCMap2GCMap: Opened file [cmaps/CooMatrix/rawObserved_100kb.gcmap] for reading writing..
INFO:addCCMap2GCMap: Adding data to [cmaps/CooMatrix/rawObserved_100kb.gcmap] for [chr1] ...
INFO:addCCMap2GCMap:     ...Finished adding data for [chr1] ...
INFO:addCCMap2GCMap: Generating downsampled maps for [chr1] ...
INFO:addCCMap2GCMap:     ... Finished downsampling for [chr1] ...
INFO:addCCMap2GCMap: Closed file [cmaps/CooMatrix/rawObserved_100kb.gcmap]...
INFO:CooMatrixHandler: Extracting-Reading [100kb_resolution_intrachromosomal/chr5/MAPQGE30/chr5_100k


INFO:CooMatrixHandler:   ...Finished extracting and reading [100kb_resolution_intrachromosomal/chr5/M
INFO:genMapFromLists: Total number of data in input file: 1533205
INFO:genMapFromLists:Minimum base-pair: 0 and Maximum base-pair: 180800000 are present in input data
INFO:genMapFromLists:Shape of overall map: (1809, 1809)

INFO:addCCMap2GCMap: Opened file [cmaps/CooMatrix/rawObserved_100kb.gcmap] for reading writing..
INFO:addCCMap2GCMap: Adding data to [cmaps/CooMatrix/rawObserved_100kb.gcmap] for [chr5] ...
INFO:addCCMap2GCMap:     ...Finished adding data for [chr5] ...
INFO:addCCMap2GCMap: Generating downsampled maps for [chr5] ...
INFO:addCCMap2GCMap:     ... Finished downsampling for [chr5] ...
INFO:addCCMap2GCMap: Closed file [cmaps/CooMatrix/rawObserved_100kb.gcmap]...
INFO:CooMatrixHandler: Extracting-Reading [100kb_resolution_intrachromosomal/chr15/MAPQGE30/chr15_100


INFO:CooMatrixHandler:   ...Finished extracting and reading [100kb_resolution_intrachromosomal/chr15/
INFO:genMapFromLists: Total number of data in input file: 318258
INFO:genMapFromLists:Minimum base-pair: 20000000 and Maximum base-pair: 102500000 are present in inpu
INFO:genMapFromLists:Shape of overall map: (1026, 1026)

INFO:addCCMap2GCMap: Opened file [cmaps/CooMatrix/rawObserved_100kb.gcmap] for reading writing..
INFO:addCCMap2GCMap: Adding data to [cmaps/CooMatrix/rawObserved_100kb.gcmap] for [chr15] ...
INFO:addCCMap2GCMap:     ...Finished adding data for [chr15] ...
INFO:addCCMap2GCMap: Generating downsampled maps for [chr15] ...
INFO:addCCMap2GCMap:     ... Finished downsampling for [chr15] ...
INFO:addCCMap2GCMap: Closed file [cmaps/CooMatrix/rawObserved_100kb.gcmap]...
INFO:CooMatrixHandler: Extracting-Reading [100kb_resolution_intrachromosomal/chr20/MAPQGE30/chr20_100


INFO:CooMatrixHandler:   ...Finished extracting and reading [100kb_resolution_intrachromosomal/chr20/
INFO:genMapFromLists: Total number of data in input file: 179488
INFO:genMapFromLists:Minimum base-pair: 0 and Maximum base-pair: 62900000 are present in input data
INFO:genMapFromLists:Shape of overall map: (630, 630)

INFO:addCCMap2GCMap: Opened file [cmaps/CooMatrix/rawObserved_100kb.gcmap] for reading writing..
INFO:addCCMap2GCMap: Adding data to [cmaps/CooMatrix/rawObserved_100kb.gcmap] for [chr20] ...
INFO:addCCMap2GCMap:     ...Finished adding data for [chr20] ...
INFO:addCCMap2GCMap: Generating downsampled maps for [chr20] ...
INFO:addCCMap2GCMap:     ... Finished downsampling for [chr20] ...
INFO:addCCMap2GCMap: Closed file [cmaps/CooMatrix/rawObserved_100kb.gcmap]...
INFO:CooMatrixHandler: Extracting-Reading [100kb_resolution_intrachromosomal/chr21/MAPQGE30/chr21_100
INFO:CooMatrixHandler:   ...Finished extracting and reading [100kb_resolution_intrachromosomal/chr21/
INFO:genMapFromLists: Total number of data in input file: 60664
INFO:genMapFromLists:Minimum base-pair: 9400000 and Maximum base-pair: 48100000 are present in input
INFO:genMapFromLists:Shape of overall map: (482, 482)
```

```
INFO:addCCMap2GCMap: Opened file [cmaps/CooMatrix/rawObserved_100kb.gcmap] for reading writing..
INFO:addCCMap2GCMap: Adding data to [cmaps/CooMatrix/rawObserved_100kb.gcmap] for [chr21] ...
INFO:addCCMap2GCMap:     ...Finished adding data for [chr21] ...
INFO:addCCMap2GCMap: Generating downsampled maps for [chr21] ...
INFO:addCCMap2GCMap:     ... Finished downsampling for [chr21] ...
INFO:addCCMap2GCMap: Closed file [cmaps/CooMatrix/rawObserved_100kb.gcmap]...
INFO:CooMatrixHandler: Extracting-Reading [100kb_resolution_intrachromosomal/chr22/MAPQGE30/chr22_10
INFO:CooMatrixHandler:   ...Finished extracting and reading [100kb_resolution_intrachromosomal/chr22/
INFO:genMapFromLists: Total number of data in input file: 59429
INFO:genMapFromLists:Minimum base-pair: 16000000 and Maximum base-pair: 51200000 are present in input
INFO:genMapFromLists:Shape of overall map: (513, 513)
```

```
INFO:addCCMap2GCMap: Opened file [cmaps/CooMatrix/rawObserved_100kb.gcmap] for reading writing..
INFO:addCCMap2GCMap: Adding data to [cmaps/CooMatrix/rawObserved_100kb.gcmap] for [chr22] ...
INFO:addCCMap2GCMap:     ...Finished adding data for [chr22] ...
INFO:addCCMap2GCMap: Generating downsampled maps for [chr22] ...
INFO:addCCMap2GCMap:     ... Finished downsampling for [chr22] ...
INFO:addCCMap2GCMap: Closed file [cmaps/CooMatrix/rawObserved_100kb.gcmap]...
```

2. From HOMER Hi-C interaction matrix format

HOMER package contains modules to analyze genome wide interaction data. It creates Hi-C matrix in a specific format as shown in this link.

Covert to ccmap

An example input file `human_INL_sample1_matrix_1Mb_raw.txt` is present in `data/HomerFormat` directory. Below, we read it and convert it to `.ccmap` formats. The input file contains several chromosomes, therefore, several `.ccmap` files will be generated for each respective chromosome.

Ouput `.ccmap` files with `suffix='_sample1'` will be saved in `cmaps/homer` directory.

See also:

Class *gcMapExplorer.lib.importer.HomerInputHandler()* for more details.

```
In [9]: # Initialize
        homer_reader = gmlib.importer.HomerInputHandler('data/HomerFormat/human_INL_sample1_matrix_1M

        # Convert and save
        homer_reader.save_ccmaps('cmaps/homer', suffix='_sample1')

        # Delete all temporary files, neccessary, automatically deleted
        del homer_reader
```

```
INFO:HomerInputHandler: Getting chromosome list and resolution from Input Files ...
INFO:HomerInputHandler: Resolution: 1mb
INFO:HomerInputHandler: Following chromsomes found in input files:
                            chr1
                            chr2
                            chr3
                            chr4
                            chr5
                            chr6
                            chr7
                            chr8
```

```
                              chr9
                              chr10
                              chr11
                              chr12
                              chr13
                              chr14
                              chr15
                              chr16
                              chr17
                              chr18
                              chr19
                              chr20
                              chr21
                              chr22
                              chrMT
                              chrX
                              chrY
INFO:HomerInputHandler: Reading [data/HomerFormat/human_INL_sample1_matrix_1Mb_raw.txt] file ...
INFO:HomerInputHandler:           ... Finished reading [data/HomerFormat/human_INL_sample1_matrix_1Mb
INFO:CooMatrixHandler: Reading file: [/home/rajendra/deskForWork/scratch/chr1_f77vvnvf.tmp]...
INFO:CooMatrixHandler:     ... Finished reading file: [/home/rajendra/deskForWork/scratch/chr1_f77vvn
INFO:genMapFromLists: Total number of data in input file: 40344
INFO:genMapFromLists:Minimum base-pair: 0 and Maximum base-pair: 249000000 are present in input data
INFO:genMapFromLists:Shape of overall map: (250, 250)



INFO:save_ccmap: Saving ccmap to file [cmaps/homer/chr1_1mb__sample1.ccmap] and [/home/rajendra/works
INFO:save_ccmap: Compressing [/home/rajendra/workspace/genome_3d_organization/tutorials_modules/cmaps
INFO:save_ccmap:       Finished!!!

INFO:CooMatrixHandler: Reading file: [/home/rajendra/deskForWork/scratch/chr2_vkkm8epw.tmp]...
INFO:CooMatrixHandler:     ... Finished reading file: [/home/rajendra/deskForWork/scratch/chr2_vkkm8e
INFO:genMapFromLists: Total number of data in input file: 46886
INFO:genMapFromLists:Minimum base-pair: 0 and Maximum base-pair: 243000000 are present in input data
INFO:genMapFromLists:Shape of overall map: (244, 244)



INFO:save_ccmap: Saving ccmap to file [cmaps/homer/chr2_1mb__sample1.ccmap] and [/home/rajendra/works
INFO:save_ccmap: Compressing [/home/rajendra/workspace/genome_3d_organization/tutorials_modules/cmaps
INFO:save_ccmap:       Finished!!!

INFO:CooMatrixHandler: Reading file: [/home/rajendra/deskForWork/scratch/chr3_9ykqxl7v.tmp]...
INFO:CooMatrixHandler:     ... Finished reading file: [/home/rajendra/deskForWork/scratch/chr3_9ykqxl
INFO:genMapFromLists: Total number of data in input file: 33308
INFO:genMapFromLists:Minimum base-pair: 0 and Maximum base-pair: 197000000 are present in input data
INFO:genMapFromLists:Shape of overall map: (198, 198)



INFO:save_ccmap: Saving ccmap to file [cmaps/homer/chr3_1mb__sample1.ccmap] and [/home/rajendra/works
INFO:save_ccmap: Compressing [/home/rajendra/workspace/genome_3d_organization/tutorials_modules/cmaps
INFO:save_ccmap:       Finished!!!

INFO:CooMatrixHandler: Reading file: [/home/rajendra/deskForWork/scratch/chr4_2rrmxtgs.tmp]...
INFO:CooMatrixHandler:     ... Finished reading file: [/home/rajendra/deskForWork/scratch/chr4_2rrmxt
INFO:genMapFromLists: Total number of data in input file: 29054
```

```
INFO:genMapFromLists:Minimum base-pair: 0 and Maximum base-pair: 191000000 are present in input data
INFO:genMapFromLists:Shape of overall map: (192, 192)



INFO:save_ccmap: Saving ccmap to file [cmaps/homer/chr4_1mb__sample1.ccmap] and [/home/rajendra/works
INFO:save_ccmap: Compressing [/home/rajendra/workspace/genome_3d_organization/tutorials_modules/cmaps
INFO:save_ccmap:       Finished!!!

INFO:CooMatrixHandler: Reading file: [/home/rajendra/deskForWork/scratch/chr5_54n2e8ij.tmp]...
INFO:CooMatrixHandler:     ... Finished reading file: [/home/rajendra/deskForWork/scratch/chr5_54n2e8
INFO:genMapFromLists: Total number of data in input file: 26286
INFO:genMapFromLists:Minimum base-pair: 0 and Maximum base-pair: 180000000 are present in input data
INFO:genMapFromLists:Shape of overall map: (181, 181)



INFO:save_ccmap: Saving ccmap to file [cmaps/homer/chr5_1mb__sample1.ccmap] and [/home/rajendra/works
INFO:save_ccmap: Compressing [/home/rajendra/workspace/genome_3d_organization/tutorials_modules/cmaps
INFO:save_ccmap:       Finished!!!

INFO:CooMatrixHandler: Reading file: [/home/rajendra/deskForWork/scratch/chr6_52cmdvyx.tmp]...
INFO:CooMatrixHandler:     ... Finished reading file: [/home/rajendra/deskForWork/scratch/chr6_52cmdv
INFO:genMapFromLists: Total number of data in input file: 25032
INFO:genMapFromLists:Minimum base-pair: 0 and Maximum base-pair: 171000000 are present in input data
INFO:genMapFromLists:Shape of overall map: (172, 172)



INFO:save_ccmap: Saving ccmap to file [cmaps/homer/chr6_1mb__sample1.ccmap] and [/home/rajendra/works
INFO:save_ccmap: Compressing [/home/rajendra/workspace/genome_3d_organization/tutorials_modules/cmaps
INFO:save_ccmap:       Finished!!!

INFO:CooMatrixHandler: Reading file: [/home/rajendra/deskForWork/scratch/chr7_5ormk7rj.tmp]...
INFO:CooMatrixHandler:     ... Finished reading file: [/home/rajendra/deskForWork/scratch/chr7_5ormk7
INFO:genMapFromLists: Total number of data in input file: 20748
INFO:genMapFromLists:Minimum base-pair: 0 and Maximum base-pair: 159000000 are present in input data
INFO:genMapFromLists:Shape of overall map: (160, 160)



INFO:save_ccmap: Saving ccmap to file [cmaps/homer/chr7_1mb__sample1.ccmap] and [/home/rajendra/works
INFO:save_ccmap: Compressing [/home/rajendra/workspace/genome_3d_organization/tutorials_modules/cmaps
INFO:save_ccmap:       Finished!!!

INFO:CooMatrixHandler: Reading file: [/home/rajendra/deskForWork/scratch/chr8_fedpovvz.tmp]...
INFO:CooMatrixHandler:     ... Finished reading file: [/home/rajendra/deskForWork/scratch/chr8_fedpov
INFO:genMapFromLists: Total number of data in input file: 18371
INFO:genMapFromLists:Minimum base-pair: 0 and Maximum base-pair: 146000000 are present in input data
INFO:genMapFromLists:Shape of overall map: (147, 147)



INFO:save_ccmap: Saving ccmap to file [cmaps/homer/chr8_1mb__sample1.ccmap] and [/home/rajendra/works
INFO:save_ccmap: Compressing [/home/rajendra/workspace/genome_3d_organization/tutorials_modules/cmaps
INFO:save_ccmap:       Finished!!!

INFO:CooMatrixHandler: Reading file: [/home/rajendra/deskForWork/scratch/chr9_qbg3ysw1.tmp]...
```

```
INFO:CooMatrixHandler:     ... Finished reading file: [/home/rajendra/deskForWork/scratch/chr9_qbg3ys
INFO:genMapFromLists: Total number of data in input file: 11414
INFO:genMapFromLists:Minimum base-pair: 0 and Maximum base-pair: 141000000 are present in input data
INFO:genMapFromLists:Shape of overall map: (142, 142)



INFO:save_ccmap: Saving ccmap to file [cmaps/homer/chr9_1mb__sample1.ccmap] and [/home/rajendra/works
INFO:save_ccmap: Compressing [/home/rajendra/workspace/genome_3d_organization/tutorials_modules/cmaps
INFO:save_ccmap:       Finished!!!

INFO:CooMatrixHandler: Reading file: [/home/rajendra/deskForWork/scratch/chr10_van15vqd.tmp]...
INFO:CooMatrixHandler:     ... Finished reading file: [/home/rajendra/deskForWork/scratch/chr10_van15
INFO:genMapFromLists: Total number of data in input file: 15560
INFO:genMapFromLists:Minimum base-pair: 0 and Maximum base-pair: 135000000 are present in input data
INFO:genMapFromLists:Shape of overall map: (136, 136)



INFO:save_ccmap: Saving ccmap to file [cmaps/homer/chr10_1mb__sample1.ccmap] and [/home/rajendra/work
INFO:save_ccmap: Compressing [/home/rajendra/workspace/genome_3d_organization/tutorials_modules/cmaps
INFO:save_ccmap:       Finished!!!

INFO:CooMatrixHandler: Reading file: [/home/rajendra/deskForWork/scratch/chr11_fnxjyqlz.tmp]...
INFO:CooMatrixHandler:     ... Finished reading file: [/home/rajendra/deskForWork/scratch/chr11_fnxjy
INFO:genMapFromLists: Total number of data in input file: 15429
INFO:genMapFromLists:Minimum base-pair: 0 and Maximum base-pair: 134000000 are present in input data
INFO:genMapFromLists:Shape of overall map: (135, 135)

INFO:save_ccmap: Saving ccmap to file [cmaps/homer/chr11_1mb__sample1.ccmap] and [/home/rajendra/work
INFO:save_ccmap: Compressing [/home/rajendra/workspace/genome_3d_organization/tutorials_modules/cmaps


INFO:save_ccmap:       Finished!!!

INFO:CooMatrixHandler: Reading file: [/home/rajendra/deskForWork/scratch/chr12_5_4a9ygv.tmp]...
INFO:CooMatrixHandler:     ... Finished reading file: [/home/rajendra/deskForWork/scratch/chr12_5_4a9
INFO:genMapFromLists: Total number of data in input file: 14928
INFO:genMapFromLists:Minimum base-pair: 0 and Maximum base-pair: 133000000 are present in input data
INFO:genMapFromLists:Shape of overall map: (134, 134)

INFO:save_ccmap: Saving ccmap to file [cmaps/homer/chr12_1mb__sample1.ccmap] and [/home/rajendra/work
INFO:save_ccmap: Compressing [/home/rajendra/workspace/genome_3d_organization/tutorials_modules/cmaps


INFO:save_ccmap:       Finished!!!

INFO:CooMatrixHandler: Reading file: [/home/rajendra/deskForWork/scratch/chr13_gaajxa0_.tmp]...
INFO:CooMatrixHandler:     ... Finished reading file: [/home/rajendra/deskForWork/scratch/chr13_gaajx
INFO:genMapFromLists: Total number of data in input file: 8675
INFO:genMapFromLists:Minimum base-pair: 19000000 and Maximum base-pair: 115000000 are present in inpu
INFO:genMapFromLists:Shape of overall map: (116, 116)

INFO:save_ccmap: Saving ccmap to file [cmaps/homer/chr13_1mb__sample1.ccmap] and [/home/rajendra/work
INFO:save_ccmap: Compressing [/home/rajendra/workspace/genome_3d_organization/tutorials_modules/cmaps


INFO:save_ccmap:       Finished!!!
```

```
INFO:CooMatrixHandler: Reading file: [/home/rajendra/deskForWork/scratch/chr14_4vrfy9r7.tmp]...
INFO:CooMatrixHandler:     ... Finished reading file: [/home/rajendra/deskForWork/scratch/chr14_4vrfy
INFO:genMapFromLists: Total number of data in input file: 7245
INFO:genMapFromLists:Minimum base-pair: 19000000 and Maximum base-pair: 107000000 are present in inpu
INFO:genMapFromLists:Shape of overall map: (108, 108)

INFO:save_ccmap: Saving ccmap to file [cmaps/homer/chr14_1mb__sample1.ccmap] and [/home/rajendra/work
INFO:save_ccmap: Compressing [/home/rajendra/workspace/genome_3d_organization/tutorials_modules/cmaps
INFO:save_ccmap:       Finished!!!

INFO:CooMatrixHandler: Reading file: [/home/rajendra/deskForWork/scratch/chr15_6ewiakeq.tmp]...
INFO:CooMatrixHandler:     ... Finished reading file: [/home/rajendra/deskForWork/scratch/chr15_6ewia
INFO:genMapFromLists: Total number of data in input file: 6249
INFO:genMapFromLists:Minimum base-pair: 20000000 and Maximum base-pair: 102000000 are present in inpu
INFO:genMapFromLists:Shape of overall map: (103, 103)



INFO:save_ccmap: Saving ccmap to file [cmaps/homer/chr15_1mb__sample1.ccmap] and [/home/rajendra/work
INFO:save_ccmap: Compressing [/home/rajendra/workspace/genome_3d_organization/tutorials_modules/cmaps
INFO:save_ccmap:       Finished!!!

INFO:CooMatrixHandler: Reading file: [/home/rajendra/deskForWork/scratch/chr16__1yly3n6.tmp]...
INFO:CooMatrixHandler:     ... Finished reading file: [/home/rajendra/deskForWork/scratch/chr16__1yly
INFO:genMapFromLists: Total number of data in input file: 5629
INFO:genMapFromLists:Minimum base-pair: 0 and Maximum base-pair: 90000000 are present in input data
INFO:genMapFromLists:Shape of overall map: (91, 91)

INFO:save_ccmap: Saving ccmap to file [cmaps/homer/chr16_1mb__sample1.ccmap] and [/home/rajendra/work
INFO:save_ccmap: Compressing [/home/rajendra/workspace/genome_3d_organization/tutorials_modules/cmaps
INFO:save_ccmap:       Finished!!!

INFO:CooMatrixHandler: Reading file: [/home/rajendra/deskForWork/scratch/chr17_nt5_zegm.tmp]...
INFO:CooMatrixHandler:     ... Finished reading file: [/home/rajendra/deskForWork/scratch/chr17_nt5_z
INFO:genMapFromLists: Total number of data in input file: 5650
INFO:genMapFromLists:Minimum base-pair: 0 and Maximum base-pair: 81000000 are present in input data
INFO:genMapFromLists:Shape of overall map: (82, 82)

INFO:save_ccmap: Saving ccmap to file [cmaps/homer/chr17_1mb__sample1.ccmap] and [/home/rajendra/work
INFO:save_ccmap: Compressing [/home/rajendra/workspace/genome_3d_organization/tutorials_modules/cmaps



INFO:save_ccmap:       Finished!!!

INFO:CooMatrixHandler: Reading file: [/home/rajendra/deskForWork/scratch/chr18_xcgf_1cx.tmp]...
INFO:CooMatrixHandler:     ... Finished reading file: [/home/rajendra/deskForWork/scratch/chr18_xcgf_
INFO:genMapFromLists: Total number of data in input file: 5581
INFO:genMapFromLists:Minimum base-pair: 0 and Maximum base-pair: 78000000 are present in input data
INFO:genMapFromLists:Shape of overall map: (79, 79)

INFO:save_ccmap: Saving ccmap to file [cmaps/homer/chr18_1mb__sample1.ccmap] and [/home/rajendra/work
INFO:save_ccmap: Compressing [/home/rajendra/workspace/genome_3d_organization/tutorials_modules/cmaps
INFO:save_ccmap:       Finished!!!

INFO:CooMatrixHandler: Reading file: [/home/rajendra/deskForWork/scratch/chr19_4662o8wv.tmp]...
INFO:CooMatrixHandler:     ... Finished reading file: [/home/rajendra/deskForWork/scratch/chr19_4662o
```

```
INFO:genMapFromLists: Total number of data in input file: 3012
INFO:genMapFromLists:Minimum base-pair: 0 and Maximum base-pair: 59000000 are present in input data
INFO:genMapFromLists:Shape of overall map: (60, 60)

INFO:save_ccmap: Saving ccmap to file [cmaps/homer/chr19_1mb__sample1.ccmap] and [/home/rajendra/work
INFO:save_ccmap: Compressing [/home/rajendra/workspace/genome_3d_organization/tutorials_modules/cmaps
INFO:save_ccmap:       Finished!!!

INFO:CooMatrixHandler: Reading file: [/home/rajendra/deskForWork/scratch/chr20_8g0vjmpm.tmp]...
INFO:CooMatrixHandler:     ... Finished reading file: [/home/rajendra/deskForWork/scratch/chr20_8g0v
INFO:genMapFromLists: Total number of data in input file: 3563
INFO:genMapFromLists:Minimum base-pair: 0 and Maximum base-pair: 62000000 are present in input data
INFO:genMapFromLists:Shape of overall map: (63, 63)




INFO:save_ccmap: Saving ccmap to file [cmaps/homer/chr20_1mb__sample1.ccmap] and [/home/rajendra/work
INFO:save_ccmap: Compressing [/home/rajendra/workspace/genome_3d_organization/tutorials_modules/cmaps
INFO:save_ccmap:       Finished!!!

INFO:CooMatrixHandler: Reading file: [/home/rajendra/deskForWork/scratch/chr21_y1cttm1r.tmp]...
INFO:CooMatrixHandler:     ... Finished reading file: [/home/rajendra/deskForWork/scratch/chr21_y1ctt
INFO:genMapFromLists: Total number of data in input file: 1266
INFO:genMapFromLists:Minimum base-pair: 9000000 and Maximum base-pair: 48000000 are present in input
INFO:genMapFromLists:Shape of overall map: (49, 49)

INFO:save_ccmap: Saving ccmap to file [cmaps/homer/chr21_1mb__sample1.ccmap] and [/home/rajendra/work
INFO:save_ccmap: Compressing [/home/rajendra/workspace/genome_3d_organization/tutorials_modules/cmaps
INFO:save_ccmap:       Finished!!!

INFO:CooMatrixHandler: Reading file: [/home/rajendra/deskForWork/scratch/chr22_yqqbttxz.tmp]...
INFO:CooMatrixHandler:     ... Finished reading file: [/home/rajendra/deskForWork/scratch/chr22_yqqbt
INFO:genMapFromLists: Total number of data in input file: 1222
INFO:genMapFromLists:Minimum base-pair: 16000000 and Maximum base-pair: 51000000 are present in input
INFO:genMapFromLists:Shape of overall map: (52, 52)

INFO:save_ccmap: Saving ccmap to file [cmaps/homer/chr22_1mb__sample1.ccmap] and [/home/rajendra/work
INFO:save_ccmap: Compressing [/home/rajendra/workspace/genome_3d_organization/tutorials_modules/cmaps
INFO:save_ccmap:       Finished!!!

INFO:CooMatrixHandler: Reading file: [/home/rajendra/deskForWork/scratch/chrMT_e250ze2w.tmp]...
INFO:CooMatrixHandler:     ... Finished reading file: [/home/rajendra/deskForWork/scratch/chrMT_e250z
INFO:genMapFromLists: Total number of data in input file: 1
INFO:genMapFromLists:Minimum base-pair: 0 and Maximum base-pair: 0 are present in input data
INFO:genMapFromLists:Shape of overall map: (1, 1)

INFO:save_ccmap: Saving ccmap to file [cmaps/homer/chrMT_1mb__sample1.ccmap] and [/home/rajendra/work
INFO:save_ccmap: Compressing [/home/rajendra/workspace/genome_3d_organization/tutorials_modules/cmaps
INFO:save_ccmap:       Finished!!!

INFO:CooMatrixHandler: Reading file: [/home/rajendra/deskForWork/scratch/chrX_bdb_wu8f.tmp]...
INFO:CooMatrixHandler:     ... Finished reading file: [/home/rajendra/deskForWork/scratch/chrX_bdb_wu
INFO:genMapFromLists: Total number of data in input file: 20634
INFO:genMapFromLists:Minimum base-pair: 0 and Maximum base-pair: 155000000 are present in input data
INFO:genMapFromLists:Shape of overall map: (156, 156)
```

```
INFO:save_ccmap: Saving ccmap to file [cmaps/homer/chrX_1mb__sample1.ccmap] and [/home/rajendra/works
INFO:save_ccmap: Compressing [/home/rajendra/workspace/genome_3d_organization/tutorials_modules/cmaps
INFO:save_ccmap:       Finished!!!

INFO:CooMatrixHandler: Reading file: [/home/rajendra/deskForWork/scratch/chrY_do37o7a1.tmp]...
INFO:CooMatrixHandler:     ... Finished reading file: [/home/rajendra/deskForWork/scratch/chrY_do37o
INFO:genMapFromLists: Total number of data in input file: 18
INFO:genMapFromLists:Minimum base-pair: 3000000 and Maximum base-pair: 59000000 are present in input
INFO:genMapFromLists:Shape of overall map: (60, 60)

INFO:save_ccmap: Saving ccmap to file [cmaps/homer/chrY_1mb__sample1.ccmap] and [/home/rajendra/works
INFO:save_ccmap: Compressing [/home/rajendra/workspace/genome_3d_organization/tutorials_modules/cmaps
INFO:save_ccmap:       Finished!!!

INFO:HomerInputHandler: Saved ['cmaps/homer/chr1_1mb__sample1.ccmap', 'cmaps/homer/chr2_1mb__sample1
```

Convert from zip file to ccmap file

An example input zip file `human_INL.zip` is present in `data/HomerFormat` directory. This zip file contains two text files. Below, we read, combine and convert them to `.ccmap` formats. The input file contains several chromosomes, therefore, several `.ccmap` files will be generated for each respective chromosome.

Ouput `.ccmap` files with `suffix='_combined'` will be saved in `cmaps/homer` directory.

```
In [10]: # Name of input ZIP file
         inputCompressedFile = 'data/HomerFormat/human_INL.zip'

         # List of files inside zip archive
         files = ['human_INL_sample1_matrix_1Mb_raw.txt', 'human_INL_sample2_matrix_1Mb_raw.txt']

         # Initialize
         homer_reader = gmlib.importer.HomerInputHandler(files, inputCompressedFile)
         homer_reader.save_ccmaps('cmaps/homer', suffix='_combined')

         # Delete all temporary files, not neccessary, automatically deleted after
         del homer_reader
```

```
INFO:HomerInputHandler: Getting chromosome list and resolution from Input Files ...
INFO:HomerInputHandler: Resolution: 1mb
INFO:HomerInputHandler: Following chromsomes found in input files:
                              chr1
                              chr2
                              chr3
                              chr4
                              chr5
                              chr6
                              chr7
                              chr8
                              chr9
                              chr10
                              chr11
                              chr12
```

```
                            chr13
                            chr14
                            chr15
                            chr16
                            chr17
                            chr18
                            chr19
                            chr20
                            chr21
                            chr22
                            chrMT
                            chrX
                            chrY
INFO:HomerInputHandler: Reading [human_INL_sample1_matrix_1Mb_raw.txt] file ...
INFO:HomerInputHandler:           ... Finished reading [human_INL_sample1_matrix_1Mb_raw.txt] file ..
INFO:HomerInputHandler: Reading [human_INL_sample2_matrix_1Mb_raw.txt] file ...
INFO:HomerInputHandler:           ... Finished reading [human_INL_sample2_matrix_1Mb_raw.txt] file ..
INFO:CooMatrixHandler: Reading file: [/home/rajendra/deskForWork/scratch/chr1_3sk6otug.tmp]...
INFO:CooMatrixHandler:     ... Finished reading file: [/home/rajendra/deskForWork/scratch/chr1_3sk6ot
INFO:genMapFromLists: Total number of data in input file: 73792
INFO:genMapFromLists:Minimum base-pair: 0 and Maximum base-pair: 249000000 are present in input data
INFO:genMapFromLists:Shape of overall map: (250, 250)



INFO:save_ccmap: Saving ccmap to file [cmaps/homer/chr1_1mb__combined.ccmap] and [/home/rajendra/work
INFO:save_ccmap: Compressing [/home/rajendra/workspace/genome_3d_organization/tutorials_modules/cmaps
INFO:save_ccmap:       Finished!!!

INFO:CooMatrixHandler: Reading file: [/home/rajendra/deskForWork/scratch/chr2_truhub8_.tmp]...
INFO:CooMatrixHandler:     ... Finished reading file: [/home/rajendra/deskForWork/scratch/chr2_truhub
INFO:genMapFromLists: Total number of data in input file: 84760
INFO:genMapFromLists:Minimum base-pair: 0 and Maximum base-pair: 243000000 are present in input data
INFO:genMapFromLists:Shape of overall map: (244, 244)



INFO:save_ccmap: Saving ccmap to file [cmaps/homer/chr2_1mb__combined.ccmap] and [/home/rajendra/work
INFO:save_ccmap: Compressing [/home/rajendra/workspace/genome_3d_organization/tutorials_modules/cmaps
INFO:save_ccmap:       Finished!!!

INFO:CooMatrixHandler: Reading file: [/home/rajendra/deskForWork/scratch/chr3_aw9a_mmf.tmp]...
INFO:CooMatrixHandler:     ... Finished reading file: [/home/rajendra/deskForWork/scratch/chr3_aw9a_m
INFO:genMapFromLists: Total number of data in input file: 61102
INFO:genMapFromLists:Minimum base-pair: 0 and Maximum base-pair: 197000000 are present in input data
INFO:genMapFromLists:Shape of overall map: (198, 198)



INFO:save_ccmap: Saving ccmap to file [cmaps/homer/chr3_1mb__combined.ccmap] and [/home/rajendra/work
INFO:save_ccmap: Compressing [/home/rajendra/workspace/genome_3d_organization/tutorials_modules/cmaps
INFO:save_ccmap:       Finished!!!

INFO:CooMatrixHandler: Reading file: [/home/rajendra/deskForWork/scratch/chr4_9beyxm8x.tmp]...
INFO:CooMatrixHandler:     ... Finished reading file: [/home/rajendra/deskForWork/scratch/chr4_9beyxm
INFO:genMapFromLists: Total number of data in input file: 52272
INFO:genMapFromLists:Minimum base-pair: 0 and Maximum base-pair: 191000000 are present in input data
INFO:genMapFromLists:Shape of overall map: (192, 192)
```

```
INFO:save_ccmap: Saving ccmap to file [cmaps/homer/chr4_1mb__combined.ccmap] and [/home/rajendra/work
INFO:save_ccmap: Compressing [/home/rajendra/workspace/genome_3d_organization/tutorials_modules/cmaps
INFO:save_ccmap:       Finished!!!

INFO:CooMatrixHandler: Reading file: [/home/rajendra/deskForWork/scratch/chr5_3s7i8xdr.tmp]...
INFO:CooMatrixHandler:     ... Finished reading file: [/home/rajendra/deskForWork/scratch/chr5_3s7i8x
INFO:genMapFromLists: Total number of data in input file: 47594
INFO:genMapFromLists:Minimum base-pair: 0 and Maximum base-pair: 180000000 are present in input data
INFO:genMapFromLists:Shape of overall map: (181, 181)



INFO:save_ccmap: Saving ccmap to file [cmaps/homer/chr5_1mb__combined.ccmap] and [/home/rajendra/work
INFO:save_ccmap: Compressing [/home/rajendra/workspace/genome_3d_organization/tutorials_modules/cmaps
INFO:save_ccmap:       Finished!!!

INFO:CooMatrixHandler: Reading file: [/home/rajendra/deskForWork/scratch/chr6_ky_ory9p.tmp]...
INFO:CooMatrixHandler:     ... Finished reading file: [/home/rajendra/deskForWork/scratch/chr6_ky_ory
INFO:genMapFromLists: Total number of data in input file: 46347
INFO:genMapFromLists:Minimum base-pair: 0 and Maximum base-pair: 171000000 are present in input data
INFO:genMapFromLists:Shape of overall map: (172, 172)



INFO:save_ccmap: Saving ccmap to file [cmaps/homer/chr6_1mb__combined.ccmap] and [/home/rajendra/work
INFO:save_ccmap: Compressing [/home/rajendra/workspace/genome_3d_organization/tutorials_modules/cmaps
INFO:save_ccmap:       Finished!!!

INFO:CooMatrixHandler: Reading file: [/home/rajendra/deskForWork/scratch/chr7_i3bykuzk.tmp]...
INFO:CooMatrixHandler:     ... Finished reading file: [/home/rajendra/deskForWork/scratch/chr7_i3byku
INFO:genMapFromLists: Total number of data in input file: 38192
INFO:genMapFromLists:Minimum base-pair: 0 and Maximum base-pair: 159000000 are present in input data
INFO:genMapFromLists:Shape of overall map: (160, 160)



INFO:save_ccmap: Saving ccmap to file [cmaps/homer/chr7_1mb__combined.ccmap] and [/home/rajendra/work
INFO:save_ccmap: Compressing [/home/rajendra/workspace/genome_3d_organization/tutorials_modules/cmaps
INFO:save_ccmap:       Finished!!!

INFO:CooMatrixHandler: Reading file: [/home/rajendra/deskForWork/scratch/chr8_d9rpwhnf.tmp]...
INFO:CooMatrixHandler:     ... Finished reading file: [/home/rajendra/deskForWork/scratch/chr8_d9rpwh
INFO:genMapFromLists: Total number of data in input file: 34554
INFO:genMapFromLists:Minimum base-pair: 0 and Maximum base-pair: 146000000 are present in input data
INFO:genMapFromLists:Shape of overall map: (147, 147)



INFO:save_ccmap: Saving ccmap to file [cmaps/homer/chr8_1mb__combined.ccmap] and [/home/rajendra/work
INFO:save_ccmap: Compressing [/home/rajendra/workspace/genome_3d_organization/tutorials_modules/cmaps
INFO:save_ccmap:       Finished!!!

INFO:CooMatrixHandler: Reading file: [/home/rajendra/deskForWork/scratch/chr9_jfoof4qc.tmp]...
INFO:CooMatrixHandler:     ... Finished reading file: [/home/rajendra/deskForWork/scratch/chr9_jfoof4
INFO:genMapFromLists: Total number of data in input file: 21457
```

```
INFO:genMapFromLists:Minimum base-pair: 0 and Maximum base-pair: 141000000 are present in input data
INFO:genMapFromLists:Shape of overall map: (142, 142)



INFO:save_ccmap: Saving ccmap to file [cmaps/homer/chr9_1mb__combined.ccmap] and [/home/rajendra/work
INFO:save_ccmap: Compressing [/home/rajendra/workspace/genome_3d_organization/tutorials_modules/cmaps
INFO:save_ccmap:       Finished!!!

INFO:CooMatrixHandler: Reading file: [/home/rajendra/deskForWork/scratch/chr10_7yjs3x_z.tmp]...
INFO:CooMatrixHandler:     ... Finished reading file: [/home/rajendra/deskForWork/scratch/chr10_7yjs3
INFO:genMapFromLists: Total number of data in input file: 29188
INFO:genMapFromLists:Minimum base-pair: 0 and Maximum base-pair: 135000000 are present in input data
INFO:genMapFromLists:Shape of overall map: (136, 136)



INFO:save_ccmap: Saving ccmap to file [cmaps/homer/chr10_1mb__combined.ccmap] and [/home/rajendra/wor
INFO:save_ccmap: Compressing [/home/rajendra/workspace/genome_3d_organization/tutorials_modules/cmaps
INFO:save_ccmap:       Finished!!!

INFO:CooMatrixHandler: Reading file: [/home/rajendra/deskForWork/scratch/chr11_wwpdr1zq.tmp]...
INFO:CooMatrixHandler:     ... Finished reading file: [/home/rajendra/deskForWork/scratch/chr11_wwpdr
INFO:genMapFromLists: Total number of data in input file: 28920
INFO:genMapFromLists:Minimum base-pair: 0 and Maximum base-pair: 134000000 are present in input data
INFO:genMapFromLists:Shape of overall map: (135, 135)



INFO:save_ccmap: Saving ccmap to file [cmaps/homer/chr11_1mb__combined.ccmap] and [/home/rajendra/wor
INFO:save_ccmap: Compressing [/home/rajendra/workspace/genome_3d_organization/tutorials_modules/cmaps
INFO:save_ccmap:       Finished!!!

INFO:CooMatrixHandler: Reading file: [/home/rajendra/deskForWork/scratch/chr12_vtt8g0s9.tmp]...
INFO:CooMatrixHandler:     ... Finished reading file: [/home/rajendra/deskForWork/scratch/chr12_vtt8g
INFO:genMapFromLists: Total number of data in input file: 27766
INFO:genMapFromLists:Minimum base-pair: 0 and Maximum base-pair: 133000000 are present in input data
INFO:genMapFromLists:Shape of overall map: (134, 134)



INFO:save_ccmap: Saving ccmap to file [cmaps/homer/chr12_1mb__combined.ccmap] and [/home/rajendra/wor
INFO:save_ccmap: Compressing [/home/rajendra/workspace/genome_3d_organization/tutorials_modules/cmaps
INFO:save_ccmap:       Finished!!!

INFO:CooMatrixHandler: Reading file: [/home/rajendra/deskForWork/scratch/chr13_v3q8__1n.tmp]...
INFO:CooMatrixHandler:     ... Finished reading file: [/home/rajendra/deskForWork/scratch/chr13_v3q8_
INFO:genMapFromLists: Total number of data in input file: 16584
INFO:genMapFromLists:Minimum base-pair: 19000000 and Maximum base-pair: 115000000 are present in inpu
INFO:genMapFromLists:Shape of overall map: (116, 116)

INFO:save_ccmap: Saving ccmap to file [cmaps/homer/chr13_1mb__combined.ccmap] and [/home/rajendra/wor
INFO:save_ccmap: Compressing [/home/rajendra/workspace/genome_3d_organization/tutorials_modules/cmaps


INFO:save_ccmap:       Finished!!!

INFO:CooMatrixHandler: Reading file: [/home/rajendra/deskForWork/scratch/chr14_03luy2_j.tmp]...
```

```
INFO:CooMatrixHandler:     ... Finished reading file: [/home/rajendra/deskForWork/scratch/chr14_03luy
INFO:genMapFromLists: Total number of data in input file: 13904
INFO:genMapFromLists:Minimum base-pair: 19000000 and Maximum base-pair: 107000000 are present in inpu
INFO:genMapFromLists:Shape of overall map: (108, 108)

INFO:save_ccmap: Saving ccmap to file [cmaps/homer/chr14_1mb__combined.ccmap] and [/home/rajendra/wor
INFO:save_ccmap: Compressing [/home/rajendra/workspace/genome_3d_organization/tutorials_modules/cmaps
INFO:save_ccmap:       Finished!!!

INFO:CooMatrixHandler: Reading file: [/home/rajendra/deskForWork/scratch/chr15_oqkuow8g.tmp]...
INFO:CooMatrixHandler:     ... Finished reading file: [/home/rajendra/deskForWork/scratch/chr15_oqkuo
INFO:genMapFromLists: Total number of data in input file: 12006
INFO:genMapFromLists:Minimum base-pair: 20000000 and Maximum base-pair: 102000000 are present in inpu
INFO:genMapFromLists:Shape of overall map: (103, 103)



INFO:save_ccmap: Saving ccmap to file [cmaps/homer/chr15_1mb__combined.ccmap] and [/home/rajendra/wor
INFO:save_ccmap: Compressing [/home/rajendra/workspace/genome_3d_organization/tutorials_modules/cmaps
INFO:save_ccmap:       Finished!!!

INFO:CooMatrixHandler: Reading file: [/home/rajendra/deskForWork/scratch/chr16_uo9j9gle.tmp]...
INFO:CooMatrixHandler:     ... Finished reading file: [/home/rajendra/deskForWork/scratch/chr16_uo9j9
INFO:genMapFromLists: Total number of data in input file: 10808
INFO:genMapFromLists:Minimum base-pair: 0 and Maximum base-pair: 90000000 are present in input data
INFO:genMapFromLists:Shape of overall map: (91, 91)

INFO:save_ccmap: Saving ccmap to file [cmaps/homer/chr16_1mb__combined.ccmap] and [/home/rajendra/wor
INFO:save_ccmap: Compressing [/home/rajendra/workspace/genome_3d_organization/tutorials_modules/cmaps
INFO:save_ccmap:       Finished!!!

INFO:CooMatrixHandler: Reading file: [/home/rajendra/deskForWork/scratch/chr17_lwa34i9w.tmp]...
INFO:CooMatrixHandler:     ... Finished reading file: [/home/rajendra/deskForWork/scratch/chr17_lwa34
INFO:genMapFromLists: Total number of data in input file: 10918
INFO:genMapFromLists:Minimum base-pair: 0 and Maximum base-pair: 81000000 are present in input data
INFO:genMapFromLists:Shape of overall map: (82, 82)



INFO:save_ccmap: Saving ccmap to file [cmaps/homer/chr17_1mb__combined.ccmap] and [/home/rajendra/wor
INFO:save_ccmap: Compressing [/home/rajendra/workspace/genome_3d_organization/tutorials_modules/cmaps
INFO:save_ccmap:       Finished!!!

INFO:CooMatrixHandler: Reading file: [/home/rajendra/deskForWork/scratch/chr18_uk88aw7n.tmp]...
INFO:CooMatrixHandler:     ... Finished reading file: [/home/rajendra/deskForWork/scratch/chr18_uk88a
INFO:genMapFromLists: Total number of data in input file: 10852
INFO:genMapFromLists:Minimum base-pair: 0 and Maximum base-pair: 78000000 are present in input data
INFO:genMapFromLists:Shape of overall map: (79, 79)

INFO:save_ccmap: Saving ccmap to file [cmaps/homer/chr18_1mb__combined.ccmap] and [/home/rajendra/wor
INFO:save_ccmap: Compressing [/home/rajendra/workspace/genome_3d_organization/tutorials_modules/cmaps
INFO:save_ccmap:       Finished!!!

INFO:CooMatrixHandler: Reading file: [/home/rajendra/deskForWork/scratch/chr19_g1xkdve7.tmp]...
INFO:CooMatrixHandler:     ... Finished reading file: [/home/rajendra/deskForWork/scratch/chr19_g1xkd
INFO:genMapFromLists: Total number of data in input file: 5892
INFO:genMapFromLists:Minimum base-pair: 0 and Maximum base-pair: 59000000 are present in input data
```

```
INFO:genMapFromLists:Shape of overall map: (60, 60)



INFO:save_ccmap: Saving ccmap to file [cmaps/homer/chr19_1mb__combined.ccmap] and [/home/rajendra/wor
INFO:save_ccmap: Compressing [/home/rajendra/workspace/genome_3d_organization/tutorials_modules/cmaps
INFO:save_ccmap:       Finished!!!

INFO:CooMatrixHandler: Reading file: [/home/rajendra/deskForWork/scratch/chr20_17aojv7z.tmp]...
INFO:CooMatrixHandler:     ... Finished reading file: [/home/rajendra/deskForWork/scratch/chr20_17ao
INFO:genMapFromLists: Total number of data in input file: 6974
INFO:genMapFromLists:Minimum base-pair: 0 and Maximum base-pair: 62000000 are present in input data
INFO:genMapFromLists:Shape of overall map: (63, 63)

INFO:save_ccmap: Saving ccmap to file [cmaps/homer/chr20_1mb__combined.ccmap] and [/home/rajendra/wor
INFO:save_ccmap: Compressing [/home/rajendra/workspace/genome_3d_organization/tutorials_modules/cmaps
INFO:save_ccmap:       Finished!!!

INFO:CooMatrixHandler: Reading file: [/home/rajendra/deskForWork/scratch/chr21_rrw6x8vw.tmp]...
INFO:CooMatrixHandler:     ... Finished reading file: [/home/rajendra/deskForWork/scratch/chr21_rrw6x
INFO:genMapFromLists: Total number of data in input file: 2474
INFO:genMapFromLists:Minimum base-pair: 9000000 and Maximum base-pair: 48000000 are present in input
INFO:genMapFromLists:Shape of overall map: (49, 49)

INFO:save_ccmap: Saving ccmap to file [cmaps/homer/chr21_1mb__combined.ccmap] and [/home/rajendra/wor
INFO:save_ccmap: Compressing [/home/rajendra/workspace/genome_3d_organization/tutorials_modules/cmaps
INFO:save_ccmap:       Finished!!!

INFO:CooMatrixHandler: Reading file: [/home/rajendra/deskForWork/scratch/chr22_ufsr5nr7.tmp]...
INFO:CooMatrixHandler:     ... Finished reading file: [/home/rajendra/deskForWork/scratch/chr22_ufsr5
INFO:genMapFromLists: Total number of data in input file: 2436
INFO:genMapFromLists:Minimum base-pair: 16000000 and Maximum base-pair: 51000000 are present in input
INFO:genMapFromLists:Shape of overall map: (52, 52)

INFO:save_ccmap: Saving ccmap to file [cmaps/homer/chr22_1mb__combined.ccmap] and [/home/rajendra/wor
INFO:save_ccmap: Compressing [/home/rajendra/workspace/genome_3d_organization/tutorials_modules/cmaps
INFO:save_ccmap:       Finished!!!



INFO:CooMatrixHandler: Reading file: [/home/rajendra/deskForWork/scratch/chrMT_wyc60lnm.tmp]...
INFO:CooMatrixHandler:     ... Finished reading file: [/home/rajendra/deskForWork/scratch/chrMT_wyc60
INFO:genMapFromLists: Total number of data in input file: 2
INFO:genMapFromLists:Minimum base-pair: 0 and Maximum base-pair: 0 are present in input data
INFO:genMapFromLists:Shape of overall map: (1, 1)

INFO:save_ccmap: Saving ccmap to file [cmaps/homer/chrMT_1mb__combined.ccmap] and [/home/rajendra/wor
INFO:save_ccmap: Compressing [/home/rajendra/workspace/genome_3d_organization/tutorials_modules/cmaps
INFO:save_ccmap:       Finished!!!

INFO:CooMatrixHandler: Reading file: [/home/rajendra/deskForWork/scratch/chrX_eyxkravm.tmp]...
INFO:CooMatrixHandler:     ... Finished reading file: [/home/rajendra/deskForWork/scratch/chrX_eyxkra
INFO:genMapFromLists: Total number of data in input file: 37926
INFO:genMapFromLists:Minimum base-pair: 0 and Maximum base-pair: 155000000 are present in input data
INFO:genMapFromLists:Shape of overall map: (156, 156)
```

```
INFO:save_ccmap: Saving ccmap to file [cmaps/homer/chrX_1mb__combined.ccmap] and [/home/rajendra/work
INFO:save_ccmap: Compressing [/home/rajendra/workspace/genome_3d_organization/tutorials_modules/cmaps
INFO:save_ccmap:       Finished!!!

INFO:CooMatrixHandler: Reading file: [/home/rajendra/deskForWork/scratch/chrY_a249pvip.tmp]...
INFO:CooMatrixHandler:     ... Finished reading file: [/home/rajendra/deskForWork/scratch/chrY_a249pv
INFO:genMapFromLists: Total number of data in input file: 29
INFO:genMapFromLists:Minimum base-pair: 3000000 and Maximum base-pair: 59000000 are present in input
INFO:genMapFromLists:Shape of overall map: (60, 60)

INFO:save_ccmap: Saving ccmap to file [cmaps/homer/chrY_1mb__combined.ccmap] and [/home/rajendra/work
INFO:save_ccmap: Compressing [/home/rajendra/workspace/genome_3d_organization/tutorials_modules/cmaps
INFO:save_ccmap:       Finished!!!

INFO:HomerInputHandler: Saved ['cmaps/homer/chr1_1mb__combined.ccmap', 'cmaps/homer/chr2_1mb__combine
```

Convert to gcmap

An example input file `human_INL_sample1_matrix_1Mb_raw.txt` is present in `data/HomerFormat` directory. Below, we read it and convert it to `.gcmap` format. The input file contains several chromosomes, and all contact maps will be added to `gcmap` file.

Ouput `human_INL_sample1_matrix_1Mb_raw.gcmap` files will be saved in `cmaps/homer` directory.

```
In [11]: # Initialize
         homer_reader = gmlib.importer.HomerInputHandler('data/HomerFormat/human_INL_sample1_matrix_1

         # Convert and save
         homer_reader.save_gcmap('cmaps/homer/human_INL_sample1_matrix_1Mb_raw.gcmap',
                                                              coarsingMethod='sum', compression='lzf')

         # Delete all temporary files, neccessary, automatically deleted
         del homer_reader
```

```
INFO:HomerInputHandler: Getting chromosome list and resolution from Input Files ...
INFO:HomerInputHandler: Resolution: 1mb
INFO:HomerInputHandler: Following chromsomes found in input files:
                            chr1
                            chr2
                            chr3
                            chr4
                            chr5
                            chr6
                            chr7
                            chr8
                            chr9
                            chr10
                            chr11
                            chr12
                            chr13
                            chr14
                            chr15
                            chr16
                            chr17
```

```
                            chr18
                            chr19
                            chr20
                            chr21
                            chr22
                            chrMT
                            chrX
                            chrY
INFO:HomerInputHandler: Reading [data/HomerFormat/human_INL_sample1_matrix_1Mb_raw.txt] file ...
INFO:HomerInputHandler:           ... Finished reading [data/HomerFormat/human_INL_sample1_matrix_1Mb
INFO:CooMatrixHandler: Reading file: [/home/rajendra/deskForWork/scratch/chr1_ozde0va4.tmp]...
INFO:CooMatrixHandler:     ... Finished reading file: [/home/rajendra/deskForWork/scratch/chr1_ozde0v
INFO:genMapFromLists: Total number of data in input file: 40344
INFO:genMapFromLists:Minimum base-pair: 0 and Maximum base-pair: 249000000 are present in input data
INFO:genMapFromLists:Shape of overall map: (250, 250)



INFO:addCCMap2GCMap: Opened file [cmaps/homer/human_INL_sample1_matrix_1Mb_raw.gcmap] for reading wri
INFO:addCCMap2GCMap: Adding data to [cmaps/homer/human_INL_sample1_matrix_1Mb_raw.gcmap] for [chr1] .
INFO:addCCMap2GCMap:     ...Finished adding data for [chr1] ...
INFO:addCCMap2GCMap: Generating downsampled maps for [chr1] ...
INFO:addCCMap2GCMap:     ... Finished downsampling for [chr1] ...
INFO:addCCMap2GCMap: Closed file [cmaps/homer/human_INL_sample1_matrix_1Mb_raw.gcmap]...
INFO:CooMatrixHandler: Reading file: [/home/rajendra/deskForWork/scratch/chr2_cpdiirk7.tmp]...
INFO:CooMatrixHandler:     ... Finished reading file: [/home/rajendra/deskForWork/scratch/chr2_cpdiir
INFO:genMapFromLists: Total number of data in input file: 46886
INFO:genMapFromLists:Minimum base-pair: 0 and Maximum base-pair: 243000000 are present in input data
INFO:genMapFromLists:Shape of overall map: (244, 244)



INFO:addCCMap2GCMap: Opened file [cmaps/homer/human_INL_sample1_matrix_1Mb_raw.gcmap] for reading wri
INFO:addCCMap2GCMap: Adding data to [cmaps/homer/human_INL_sample1_matrix_1Mb_raw.gcmap] for [chr2] .
INFO:addCCMap2GCMap:     ...Finished adding data for [chr2] ...
INFO:addCCMap2GCMap: Generating downsampled maps for [chr2] ...
INFO:addCCMap2GCMap:     ... Finished downsampling for [chr2] ...
INFO:addCCMap2GCMap: Closed file [cmaps/homer/human_INL_sample1_matrix_1Mb_raw.gcmap]...
INFO:CooMatrixHandler: Reading file: [/home/rajendra/deskForWork/scratch/chr3_6rz30dtu.tmp]...
INFO:CooMatrixHandler:     ... Finished reading file: [/home/rajendra/deskForWork/scratch/chr3_6rz30d
INFO:genMapFromLists: Total number of data in input file: 33308
INFO:genMapFromLists:Minimum base-pair: 0 and Maximum base-pair: 197000000 are present in input data
INFO:genMapFromLists:Shape of overall map: (198, 198)



INFO:addCCMap2GCMap: Opened file [cmaps/homer/human_INL_sample1_matrix_1Mb_raw.gcmap] for reading wri
INFO:addCCMap2GCMap: Adding data to [cmaps/homer/human_INL_sample1_matrix_1Mb_raw.gcmap] for [chr3] .
INFO:addCCMap2GCMap:     ...Finished adding data for [chr3] ...
INFO:addCCMap2GCMap: Generating downsampled maps for [chr3] ...
INFO:addCCMap2GCMap:     ... Finished downsampling for [chr3] ...
INFO:addCCMap2GCMap: Closed file [cmaps/homer/human_INL_sample1_matrix_1Mb_raw.gcmap]...
INFO:CooMatrixHandler: Reading file: [/home/rajendra/deskForWork/scratch/chr4_eat16ys2.tmp]...
INFO:CooMatrixHandler:     ... Finished reading file: [/home/rajendra/deskForWork/scratch/chr4_eat16y
INFO:genMapFromLists: Total number of data in input file: 29054
INFO:genMapFromLists:Minimum base-pair: 0 and Maximum base-pair: 191000000 are present in input data
INFO:genMapFromLists:Shape of overall map: (192, 192)
```

```
INFO:addCCMap2GCMap: Opened file [cmaps/homer/human_INL_sample1_matrix_1Mb_raw.gcmap] for reading wri
INFO:addCCMap2GCMap: Adding data to [cmaps/homer/human_INL_sample1_matrix_1Mb_raw.gcmap] for [chr4]
INFO:addCCMap2GCMap:     ...Finished adding data for [chr4] ...
INFO:addCCMap2GCMap: Generating downsampled maps for [chr4] ...
INFO:addCCMap2GCMap:     ... Finished downsampling for [chr4] ...
INFO:addCCMap2GCMap: Closed file [cmaps/homer/human_INL_sample1_matrix_1Mb_raw.gcmap]...
INFO:CooMatrixHandler: Reading file: [/home/rajendra/deskForWork/scratch/chr5_tg7a2qrx.tmp]...
INFO:CooMatrixHandler:     ... Finished reading file: [/home/rajendra/deskForWork/scratch/chr5_tg7a2
INFO:genMapFromLists: Total number of data in input file: 26286
INFO:genMapFromLists:Minimum base-pair: 0 and Maximum base-pair: 180000000 are present in input data
INFO:genMapFromLists:Shape of overall map: (181, 181)


INFO:addCCMap2GCMap: Opened file [cmaps/homer/human_INL_sample1_matrix_1Mb_raw.gcmap] for reading wri
INFO:addCCMap2GCMap: Adding data to [cmaps/homer/human_INL_sample1_matrix_1Mb_raw.gcmap] for [chr5]
INFO:addCCMap2GCMap:     ...Finished adding data for [chr5] ...
INFO:addCCMap2GCMap: Generating downsampled maps for [chr5] ...
INFO:addCCMap2GCMap:     ... Finished downsampling for [chr5] ...
INFO:addCCMap2GCMap: Closed file [cmaps/homer/human_INL_sample1_matrix_1Mb_raw.gcmap]...
INFO:CooMatrixHandler: Reading file: [/home/rajendra/deskForWork/scratch/chr6_82bba4ff.tmp]...
INFO:CooMatrixHandler:     ... Finished reading file: [/home/rajendra/deskForWork/scratch/chr6_82bba
INFO:genMapFromLists: Total number of data in input file: 25032
INFO:genMapFromLists:Minimum base-pair: 0 and Maximum base-pair: 171000000 are present in input data
INFO:genMapFromLists:Shape of overall map: (172, 172)


INFO:addCCMap2GCMap: Opened file [cmaps/homer/human_INL_sample1_matrix_1Mb_raw.gcmap] for reading wri
INFO:addCCMap2GCMap: Adding data to [cmaps/homer/human_INL_sample1_matrix_1Mb_raw.gcmap] for [chr6]
INFO:addCCMap2GCMap:     ...Finished adding data for [chr6] ...
INFO:addCCMap2GCMap: Generating downsampled maps for [chr6] ...
INFO:addCCMap2GCMap:     ... Finished downsampling for [chr6] ...
INFO:addCCMap2GCMap: Closed file [cmaps/homer/human_INL_sample1_matrix_1Mb_raw.gcmap]...
INFO:CooMatrixHandler: Reading file: [/home/rajendra/deskForWork/scratch/chr7_zykfac9h.tmp]...
INFO:CooMatrixHandler:     ... Finished reading file: [/home/rajendra/deskForWork/scratch/chr7_zykfac
INFO:genMapFromLists: Total number of data in input file: 20748
INFO:genMapFromLists:Minimum base-pair: 0 and Maximum base-pair: 159000000 are present in input data
INFO:genMapFromLists:Shape of overall map: (160, 160)


INFO:addCCMap2GCMap: Opened file [cmaps/homer/human_INL_sample1_matrix_1Mb_raw.gcmap] for reading wri
INFO:addCCMap2GCMap: Adding data to [cmaps/homer/human_INL_sample1_matrix_1Mb_raw.gcmap] for [chr7]
INFO:addCCMap2GCMap:     ...Finished adding data for [chr7] ...
INFO:addCCMap2GCMap: Generating downsampled maps for [chr7] ...
INFO:addCCMap2GCMap:     ... Finished downsampling for [chr7] ...
INFO:addCCMap2GCMap: Closed file [cmaps/homer/human_INL_sample1_matrix_1Mb_raw.gcmap]...
INFO:CooMatrixHandler: Reading file: [/home/rajendra/deskForWork/scratch/chr8_gap1ob5e.tmp]...
INFO:CooMatrixHandler:     ... Finished reading file: [/home/rajendra/deskForWork/scratch/chr8_gap1ob
INFO:genMapFromLists: Total number of data in input file: 18371
INFO:genMapFromLists:Minimum base-pair: 0 and Maximum base-pair: 146000000 are present in input data
INFO:genMapFromLists:Shape of overall map: (147, 147)


INFO:addCCMap2GCMap: Opened file [cmaps/homer/human_INL_sample1_matrix_1Mb_raw.gcmap] for reading wri
```

```
INFO:addCCMap2GCMap: Adding data to [cmaps/homer/human_INL_sample1_matrix_1Mb_raw.gcmap] for [chr8]
INFO:addCCMap2GCMap:     ...Finished adding data for [chr8] ...
INFO:addCCMap2GCMap: Generating downsampled maps for [chr8] ...
INFO:addCCMap2GCMap:     ... Finished downsampling for [chr8] ...
INFO:addCCMap2GCMap: Closed file [cmaps/homer/human_INL_sample1_matrix_1Mb_raw.gcmap]...
INFO:CooMatrixHandler: Reading file: [/home/rajendra/deskForWork/scratch/chr9_uehz864s.tmp]...
INFO:CooMatrixHandler:     ... Finished reading file: [/home/rajendra/deskForWork/scratch/chr9_uehz86
INFO:genMapFromLists: Total number of data in input file: 11414
INFO:genMapFromLists:Minimum base-pair: 0 and Maximum base-pair: 141000000 are present in input data
INFO:genMapFromLists:Shape of overall map: (142, 142)

INFO:addCCMap2GCMap: Opened file [cmaps/homer/human_INL_sample1_matrix_1Mb_raw.gcmap] for reading wri
INFO:addCCMap2GCMap: Adding data to [cmaps/homer/human_INL_sample1_matrix_1Mb_raw.gcmap] for [chr9]
INFO:addCCMap2GCMap:     ...Finished adding data for [chr9] ...
INFO:addCCMap2GCMap: Generating downsampled maps for [chr9] ...
INFO:addCCMap2GCMap:     ... Finished downsampling for [chr9] ...
INFO:addCCMap2GCMap: Closed file [cmaps/homer/human_INL_sample1_matrix_1Mb_raw.gcmap]...
INFO:CooMatrixHandler: Reading file: [/home/rajendra/deskForWork/scratch/chr10_he_lg6i2.tmp]...
INFO:CooMatrixHandler:     ... Finished reading file: [/home/rajendra/deskForWork/scratch/chr10_he_lg
INFO:genMapFromLists: Total number of data in input file: 15560
INFO:genMapFromLists:Minimum base-pair: 0 and Maximum base-pair: 135000000 are present in input data
INFO:genMapFromLists:Shape of overall map: (136, 136)



INFO:addCCMap2GCMap: Opened file [cmaps/homer/human_INL_sample1_matrix_1Mb_raw.gcmap] for reading wri
INFO:addCCMap2GCMap: Adding data to [cmaps/homer/human_INL_sample1_matrix_1Mb_raw.gcmap] for [chr10]
INFO:addCCMap2GCMap:     ...Finished adding data for [chr10] ...
INFO:addCCMap2GCMap: Generating downsampled maps for [chr10] ...
INFO:addCCMap2GCMap:     ... Finished downsampling for [chr10] ...
INFO:addCCMap2GCMap: Closed file [cmaps/homer/human_INL_sample1_matrix_1Mb_raw.gcmap]...
INFO:CooMatrixHandler: Reading file: [/home/rajendra/deskForWork/scratch/chr11_sgvef97s.tmp]...
INFO:CooMatrixHandler:     ... Finished reading file: [/home/rajendra/deskForWork/scratch/chr11_sgvef
INFO:genMapFromLists: Total number of data in input file: 15429
INFO:genMapFromLists:Minimum base-pair: 0 and Maximum base-pair: 134000000 are present in input data
INFO:genMapFromLists:Shape of overall map: (135, 135)

INFO:addCCMap2GCMap: Opened file [cmaps/homer/human_INL_sample1_matrix_1Mb_raw.gcmap] for reading wri
INFO:addCCMap2GCMap: Adding data to [cmaps/homer/human_INL_sample1_matrix_1Mb_raw.gcmap] for [chr11]
INFO:addCCMap2GCMap:     ...Finished adding data for [chr11] ...
INFO:addCCMap2GCMap: Generating downsampled maps for [chr11] ...
INFO:addCCMap2GCMap:     ... Finished downsampling for [chr11] ...
INFO:addCCMap2GCMap: Closed file [cmaps/homer/human_INL_sample1_matrix_1Mb_raw.gcmap]...


INFO:CooMatrixHandler: Reading file: [/home/rajendra/deskForWork/scratch/chr12_t4cag_zp.tmp]...
INFO:CooMatrixHandler:     ... Finished reading file: [/home/rajendra/deskForWork/scratch/chr12_t4cag
INFO:genMapFromLists: Total number of data in input file: 14928
INFO:genMapFromLists:Minimum base-pair: 0 and Maximum base-pair: 133000000 are present in input data
INFO:genMapFromLists:Shape of overall map: (134, 134)

INFO:addCCMap2GCMap: Opened file [cmaps/homer/human_INL_sample1_matrix_1Mb_raw.gcmap] for reading wri
INFO:addCCMap2GCMap: Adding data to [cmaps/homer/human_INL_sample1_matrix_1Mb_raw.gcmap] for [chr12]
INFO:addCCMap2GCMap:     ...Finished adding data for [chr12] ...
INFO:addCCMap2GCMap: Generating downsampled maps for [chr12] ...
INFO:addCCMap2GCMap:     ... Finished downsampling for [chr12] ...
INFO:addCCMap2GCMap: Closed file [cmaps/homer/human_INL_sample1_matrix_1Mb_raw.gcmap]...
INFO:CooMatrixHandler: Reading file: [/home/rajendra/deskForWork/scratch/chr13_yw69ibjp.tmp]...
```

```
INFO:CooMatrixHandler:     ... Finished reading file: [/home/rajendra/deskForWork/scratch/chr13_yw69
INFO:genMapFromLists: Total number of data in input file: 8675
INFO:genMapFromLists:Minimum base-pair: 19000000 and Maximum base-pair: 115000000 are present in inpu
INFO:genMapFromLists:Shape of overall map: (116, 116)

INFO:addCCMap2GCMap: Opened file [cmaps/homer/human_INL_sample1_matrix_1Mb_raw.gcmap] for reading wri
INFO:addCCMap2GCMap: Adding data to [cmaps/homer/human_INL_sample1_matrix_1Mb_raw.gcmap] for [chr13]
INFO:addCCMap2GCMap:     ...Finished adding data for [chr13] ...
INFO:addCCMap2GCMap: Generating downsampled maps for [chr13] ...
INFO:addCCMap2GCMap:     ... Finished downsampling for [chr13] ...
INFO:addCCMap2GCMap: Closed file [cmaps/homer/human_INL_sample1_matrix_1Mb_raw.gcmap]...
INFO:CooMatrixHandler: Reading file: [/home/rajendra/deskForWork/scratch/chr14_0isd7_l1.tmp]...
INFO:CooMatrixHandler:     ... Finished reading file: [/home/rajendra/deskForWork/scratch/chr14_0isd
INFO:genMapFromLists: Total number of data in input file: 7245
INFO:genMapFromLists:Minimum base-pair: 19000000 and Maximum base-pair: 107000000 are present in inpu
INFO:genMapFromLists:Shape of overall map: (108, 108)

INFO:addCCMap2GCMap: Opened file [cmaps/homer/human_INL_sample1_matrix_1Mb_raw.gcmap] for reading wri
INFO:addCCMap2GCMap: Adding data to [cmaps/homer/human_INL_sample1_matrix_1Mb_raw.gcmap] for [chr14]
INFO:addCCMap2GCMap:     ...Finished adding data for [chr14] ...
INFO:addCCMap2GCMap: Generating downsampled maps for [chr14] ...
INFO:addCCMap2GCMap:     ... Finished downsampling for [chr14] ...
INFO:addCCMap2GCMap: Closed file [cmaps/homer/human_INL_sample1_matrix_1Mb_raw.gcmap]...
INFO:CooMatrixHandler: Reading file: [/home/rajendra/deskForWork/scratch/chr15_2shc6o7t.tmp]...
INFO:CooMatrixHandler:     ... Finished reading file: [/home/rajendra/deskForWork/scratch/chr15_2shc6
INFO:genMapFromLists: Total number of data in input file: 6249
INFO:genMapFromLists:Minimum base-pair: 20000000 and Maximum base-pair: 102000000 are present in inpu
INFO:genMapFromLists:Shape of overall map: (103, 103)

INFO:addCCMap2GCMap: Opened file [cmaps/homer/human_INL_sample1_matrix_1Mb_raw.gcmap] for reading wri
INFO:addCCMap2GCMap: Adding data to [cmaps/homer/human_INL_sample1_matrix_1Mb_raw.gcmap] for [chr15]
INFO:addCCMap2GCMap:     ...Finished adding data for [chr15] ...
INFO:addCCMap2GCMap: Generating downsampled maps for [chr15] ...
INFO:addCCMap2GCMap:     ... Finished downsampling for [chr15] ...
INFO:addCCMap2GCMap: Closed file [cmaps/homer/human_INL_sample1_matrix_1Mb_raw.gcmap]...



INFO:CooMatrixHandler: Reading file: [/home/rajendra/deskForWork/scratch/chr16_vg29q2l6.tmp]...
INFO:CooMatrixHandler:     ... Finished reading file: [/home/rajendra/deskForWork/scratch/chr16_vg29q
INFO:genMapFromLists: Total number of data in input file: 5629
INFO:genMapFromLists:Minimum base-pair: 0 and Maximum base-pair: 90000000 are present in input data
INFO:genMapFromLists:Shape of overall map: (91, 91)

INFO:addCCMap2GCMap: Opened file [cmaps/homer/human_INL_sample1_matrix_1Mb_raw.gcmap] for reading wri
INFO:addCCMap2GCMap: Adding data to [cmaps/homer/human_INL_sample1_matrix_1Mb_raw.gcmap] for [chr16]
INFO:addCCMap2GCMap:     ...Finished adding data for [chr16] ...
INFO:addCCMap2GCMap: Generating downsampled maps for [chr16] ...
INFO:addCCMap2GCMap:     ... Finished downsampling for [chr16] ...
INFO:addCCMap2GCMap: Closed file [cmaps/homer/human_INL_sample1_matrix_1Mb_raw.gcmap]...
INFO:CooMatrixHandler: Reading file: [/home/rajendra/deskForWork/scratch/chr17_hioju8k0.tmp]...
INFO:CooMatrixHandler:     ... Finished reading file: [/home/rajendra/deskForWork/scratch/chr17_hioju
INFO:genMapFromLists: Total number of data in input file: 5650
INFO:genMapFromLists:Minimum base-pair: 0 and Maximum base-pair: 81000000 are present in input data
INFO:genMapFromLists:Shape of overall map: (82, 82)
```

```
INFO:addCCMap2GCMap: Opened file [cmaps/homer/human_INL_sample1_matrix_1Mb_raw.gcmap] for reading wri
INFO:addCCMap2GCMap: Adding data to [cmaps/homer/human_INL_sample1_matrix_1Mb_raw.gcmap] for [chr17]
INFO:addCCMap2GCMap:     ...Finished adding data for [chr17] ...
INFO:addCCMap2GCMap: Generating downsampled maps for [chr17] ...
INFO:addCCMap2GCMap:     ... Finished downsampling for [chr17] ...
INFO:addCCMap2GCMap: Closed file [cmaps/homer/human_INL_sample1_matrix_1Mb_raw.gcmap]...
INFO:CooMatrixHandler: Reading file: [/home/rajendra/deskForWork/scratch/chr18_6i32yh98.tmp]...
INFO:CooMatrixHandler:     ... Finished reading file: [/home/rajendra/deskForWork/scratch/chr18_6i32y
INFO:genMapFromLists: Total number of data in input file: 5581
INFO:genMapFromLists:Minimum base-pair: 0 and Maximum base-pair: 78000000 are present in input data
INFO:genMapFromLists:Shape of overall map: (79, 79)
```

```
INFO:addCCMap2GCMap: Opened file [cmaps/homer/human_INL_sample1_matrix_1Mb_raw.gcmap] for reading wri
INFO:addCCMap2GCMap: Adding data to [cmaps/homer/human_INL_sample1_matrix_1Mb_raw.gcmap] for [chr18]
INFO:addCCMap2GCMap:     ...Finished adding data for [chr18] ...
INFO:addCCMap2GCMap: Generating downsampled maps for [chr18] ...
INFO:addCCMap2GCMap:     ... Finished downsampling for [chr18] ...
INFO:addCCMap2GCMap: Closed file [cmaps/homer/human_INL_sample1_matrix_1Mb_raw.gcmap]...
INFO:CooMatrixHandler: Reading file: [/home/rajendra/deskForWork/scratch/chr19_6vxo52rp.tmp]...
INFO:CooMatrixHandler:     ... Finished reading file: [/home/rajendra/deskForWork/scratch/chr19_6vxo5
INFO:genMapFromLists: Total number of data in input file: 3012
INFO:genMapFromLists:Minimum base-pair: 0 and Maximum base-pair: 59000000 are present in input data
INFO:genMapFromLists:Shape of overall map: (60, 60)

INFO:addCCMap2GCMap: Opened file [cmaps/homer/human_INL_sample1_matrix_1Mb_raw.gcmap] for reading wri
INFO:addCCMap2GCMap: Adding data to [cmaps/homer/human_INL_sample1_matrix_1Mb_raw.gcmap] for [chr19]
INFO:addCCMap2GCMap:     ...Finished adding data for [chr19] ...
INFO:addCCMap2GCMap: Generating downsampled maps for [chr19] ...
INFO:addCCMap2GCMap:     ... Finished downsampling for [chr19] ...
INFO:addCCMap2GCMap: Closed file [cmaps/homer/human_INL_sample1_matrix_1Mb_raw.gcmap]...
INFO:CooMatrixHandler: Reading file: [/home/rajendra/deskForWork/scratch/chr20_75fs4w_z.tmp]...
INFO:CooMatrixHandler:     ... Finished reading file: [/home/rajendra/deskForWork/scratch/chr20_75fs4
INFO:genMapFromLists: Total number of data in input file: 3563
INFO:genMapFromLists:Minimum base-pair: 0 and Maximum base-pair: 62000000 are present in input data
INFO:genMapFromLists:Shape of overall map: (63, 63)

INFO:addCCMap2GCMap: Opened file [cmaps/homer/human_INL_sample1_matrix_1Mb_raw.gcmap] for reading wri
INFO:addCCMap2GCMap: Adding data to [cmaps/homer/human_INL_sample1_matrix_1Mb_raw.gcmap] for [chr20]
INFO:addCCMap2GCMap:     ...Finished adding data for [chr20] ...
INFO:addCCMap2GCMap: Generating downsampled maps for [chr20] ...
INFO:addCCMap2GCMap:     ... Finished downsampling for [chr20] ...
INFO:addCCMap2GCMap: Closed file [cmaps/homer/human_INL_sample1_matrix_1Mb_raw.gcmap]...
INFO:CooMatrixHandler: Reading file: [/home/rajendra/deskForWork/scratch/chr21_k7h2y6a8.tmp]...
INFO:CooMatrixHandler:     ... Finished reading file: [/home/rajendra/deskForWork/scratch/chr21_k7h2y
INFO:genMapFromLists: Total number of data in input file: 1266
INFO:genMapFromLists:Minimum base-pair: 9000000 and Maximum base-pair: 48000000 are present in input
INFO:genMapFromLists:Shape of overall map: (49, 49)

INFO:addCCMap2GCMap: Opened file [cmaps/homer/human_INL_sample1_matrix_1Mb_raw.gcmap] for reading wri
INFO:addCCMap2GCMap: Adding data to [cmaps/homer/human_INL_sample1_matrix_1Mb_raw.gcmap] for [chr21]
INFO:addCCMap2GCMap:     ...Finished adding data for [chr21] ...
INFO:addCCMap2GCMap: Generating downsampled maps for [chr21] ...
INFO:addCCMap2GCMap:     ... Finished downsampling for [chr21] ...
INFO:addCCMap2GCMap: Closed file [cmaps/homer/human_INL_sample1_matrix_1Mb_raw.gcmap]...
INFO:CooMatrixHandler: Reading file: [/home/rajendra/deskForWork/scratch/chr22_lkz9p2ak.tmp]...
```

```
INFO:CooMatrixHandler:     ... Finished reading file: [/home/rajendra/deskForWork/scratch/chr22_lkz9p
INFO:genMapFromLists: Total number of data in input file: 1222
INFO:genMapFromLists:Minimum base-pair: 16000000 and Maximum base-pair: 51000000 are present in input

INFO:addCCMap2GCMap: Opened file [cmaps/homer/human_INL_sample1_matrix_1Mb_raw.gcmap] for reading wri
INFO:addCCMap2GCMap: Adding data to [cmaps/homer/human_INL_sample1_matrix_1Mb_raw.gcmap] for [chr22]
INFO:addCCMap2GCMap:     ...Finished adding data for [chr22] ...
INFO:addCCMap2GCMap: Generating downsampled maps for [chr22] ...
INFO:addCCMap2GCMap:     ... Finished downsampling for [chr22] ...




INFO:addCCMap2GCMap: Closed file [cmaps/homer/human_INL_sample1_matrix_1Mb_raw.gcmap]...
INFO:CooMatrixHandler: Reading file: [/home/rajendra/deskForWork/scratch/chrMT_wrr9inav.tmp]...
INFO:CooMatrixHandler:     ... Finished reading file: [/home/rajendra/deskForWork/scratch/chrMT_wrr9i
INFO:genMapFromLists: Total number of data in input file: 1
INFO:genMapFromLists:Minimum base-pair: 0 and Maximum base-pair: 0 are present in input data
INFO:genMapFromLists:Shape of overall map: (1, 1)

INFO:addCCMap2GCMap: Opened file [cmaps/homer/human_INL_sample1_matrix_1Mb_raw.gcmap] for reading wri
INFO:addCCMap2GCMap: Adding data to [cmaps/homer/human_INL_sample1_matrix_1Mb_raw.gcmap] for [chrMT]
INFO:addCCMap2GCMap:     ...Finished adding data for [chrMT] ...
INFO:addCCMap2GCMap: Generating downsampled maps for [chrMT] ...
INFO:addCCMap2GCMap:     ... Finished downsampling for [chrMT] ...
INFO:addCCMap2GCMap: Closed file [cmaps/homer/human_INL_sample1_matrix_1Mb_raw.gcmap]...
INFO:CooMatrixHandler: Reading file: [/home/rajendra/deskForWork/scratch/chrX_z02f32p7.tmp]...
INFO:CooMatrixHandler:     ... Finished reading file: [/home/rajendra/deskForWork/scratch/chrX_z02f32
INFO:genMapFromLists: Total number of data in input file: 20634
INFO:genMapFromLists:Minimum base-pair: 0 and Maximum base-pair: 155000000 are present in input data
INFO:genMapFromLists:Shape of overall map: (156, 156)




INFO:addCCMap2GCMap: Opened file [cmaps/homer/human_INL_sample1_matrix_1Mb_raw.gcmap] for reading wri
INFO:addCCMap2GCMap: Adding data to [cmaps/homer/human_INL_sample1_matrix_1Mb_raw.gcmap] for [chrX]
INFO:addCCMap2GCMap:     ...Finished adding data for [chrX] ...
INFO:addCCMap2GCMap: Generating downsampled maps for [chrX] ...
INFO:addCCMap2GCMap:     ... Finished downsampling for [chrX] ...
INFO:addCCMap2GCMap: Closed file [cmaps/homer/human_INL_sample1_matrix_1Mb_raw.gcmap]...
INFO:CooMatrixHandler: Reading file: [/home/rajendra/deskForWork/scratch/chrY_kkzm18rx.tmp]...
INFO:CooMatrixHandler:     ... Finished reading file: [/home/rajendra/deskForWork/scratch/chrY_kkzm18
INFO:genMapFromLists: Total number of data in input file: 18
INFO:genMapFromLists:Minimum base-pair: 3000000 and Maximum base-pair: 59000000 are present in input

INFO:addCCMap2GCMap: Opened file [cmaps/homer/human_INL_sample1_matrix_1Mb_raw.gcmap] for reading wri
INFO:addCCMap2GCMap: Adding data to [cmaps/homer/human_INL_sample1_matrix_1Mb_raw.gcmap] for [chrY]
INFO:addCCMap2GCMap:     ...Finished adding data for [chrY] ...
INFO:addCCMap2GCMap: Generating downsampled maps for [chrY] ...
INFO:addCCMap2GCMap:     ... Finished downsampling for [chrY] ...
INFO:addCCMap2GCMap: Closed file [cmaps/homer/human_INL_sample1_matrix_1Mb_raw.gcmap]...
```

3. From Bin-Contact format

These types of files are present in following GEO data: * http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE61471 * http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE34453

This format contains a pair of file: * bin file:

```
cbin    chr from.coord  to.coord    count
1   2L  0   160000  747
2   2L  160000  320000  893
3   2L  320000  480000  1056
4   2L  480000  640000  1060
5   2L  640000  800000  978
6   2L  800000  960000  926
.
.
.
```

- Contact file in list format

```
cbin1   cbin2   expected_count  observed_count
1   1   40.245201   21339
1   2   83.747499   5661
1   3   92.12501    1546
1   4   93.401273   864
1   5   87.265472   442
.
.
.
```

Convert to ccmap

A pair of example input files `nm_none_160000.bins` and `nm_none_160000.n_contact` is present in `data/binContactFormat` directory. Below, we read it and convert it to `.ccmap` formats. The input file contains several chromosomes, therefore, several `.ccmap` files will be generated for each respective chromosome.

Ouput `.ccmap` files will be saved in `cmaps/binContact` directory.

See also:

Class *gcMapExplorer.lib.importer.BinsNContactFilesHandler()* for more details.

```
In [12]: # File names
         binFile = 'data/binContactFormat/nm_none_160000.bins'
         contactFile = 'data/binContactFormat/nm_none_160000.n_contact'

         # Initialize
         binContactReader = gmlib.importer.BinsNContactFilesHandler(binFile, contactFile)

         # Save ccmaps
         binContactReader.save_ccmaps('cmaps/binContact')
```

```
INFO:BinsNContactFilesHandler:  Chromosome Size:
                                                  4 : 1280000
                                                 3L : 24640000
                                                 2L : 23040000
                                                 2R : 21280000
                                                  X : 22560000
```

```
                                               3R : 28000000

INFO:BinsNContactFilesHandler:  Chromosome Bins info:
                                                     4: {'min': 607, 'max': 614}
                                                    3L: {'min': 278, 'max': 431}
                                                    2L: {'min': 1, 'max': 144}
                                                    2R: {'min': 145, 'max': 277}
                                                     X: {'min': 615, 'max': 755}
                                                    3R: {'min': 432, 'max': 606}

INFO:BinsNContactFilesHandler: Generating temporary numpy array file [/home/rajendra/deskForWork/scra
INFO:BinsNContactFilesHandler: Finished.

INFO:BinsNContactFilesHandler: Generating temporary numpy array file [/home/rajendra/deskForWork/scra
INFO:BinsNContactFilesHandler: Finished.

INFO:BinsNContactFilesHandler: Generating temporary numpy array file [/home/rajendra/deskForWork/scra
INFO:BinsNContactFilesHandler: Finished.

INFO:BinsNContactFilesHandler: Generating temporary numpy array file [/home/rajendra/deskForWork/scra
INFO:BinsNContactFilesHandler: Finished.

INFO:BinsNContactFilesHandler: Generating temporary numpy array file [/home/rajendra/deskForWork/scra
INFO:BinsNContactFilesHandler: Finished.

INFO:BinsNContactFilesHandler: Generating temporary numpy array file [/home/rajendra/deskForWork/scra
INFO:BinsNContactFilesHandler: Finished.

INFO:BinsNContactFilesHandler: Reading contact file ...

INFO:BinsNContactFilesHandler:  Generating Hi-C Map for [2L] ...

INFO:genMapFromLists: Total number of data in input file: 20737
INFO:genMapFromLists:Minimum base-pair: 160000 and Maximum base-pair: 23040000 are present in input d
INFO:genMapFromLists:Shape of overall map: (145, 145)

INFO:BinsNContactFilesHandler: Finished

INFO:BinsNContactFilesHandler:  Generating Hi-C Map for [2R] ...

INFO:genMapFromLists: Total number of data in input file: 17689
INFO:genMapFromLists:Minimum base-pair: 160000 and Maximum base-pair: 21280000 are present in input d
INFO:genMapFromLists:Shape of overall map: (134, 134)

INFO:BinsNContactFilesHandler: Finished

INFO:BinsNContactFilesHandler:  Generating Hi-C Map for [3L] ...

INFO:genMapFromLists: Total number of data in input file: 23716
INFO:genMapFromLists:Minimum base-pair: 160000 and Maximum base-pair: 24640000 are present in input d
INFO:genMapFromLists:Shape of overall map: (155, 155)

INFO:BinsNContactFilesHandler: Finished

INFO:BinsNContactFilesHandler:  Generating Hi-C Map for [3R] ...

INFO:genMapFromLists: Total number of data in input file: 30625
INFO:genMapFromLists:Minimum base-pair: 160000 and Maximum base-pair: 28000000 are present in input d
INFO:genMapFromLists:Shape of overall map: (176, 176)
```

```
INFO:BinsNContactFilesHandler: Finished

INFO:BinsNContactFilesHandler:  Generating Hi-C Map for [4] ...

INFO:genMapFromLists: Total number of data in input file: 64
INFO:genMapFromLists:Minimum base-pair: 160000 and Maximum base-pair: 1280000 are present in input da
INFO:genMapFromLists:Shape of overall map: (9, 9)

INFO:BinsNContactFilesHandler: Finished

INFO:BinsNContactFilesHandler:  Generating Hi-C Map for [X] ...
INFO:genMapFromLists: Total number of data in input file: 19880
INFO:genMapFromLists:Minimum base-pair: 160000 and Maximum base-pair: 22560000 are present in input c
INFO:genMapFromLists:Shape of overall map: (142, 142)

INFO:BinsNContactFilesHandler: Finished

INFO:BinsNContactFilesHandler: Finished reading contact file.

INFO:BinsNContactFilesHandler:   Hi-C Maps Summary:
                                                Chromosome    Size            Max.    Min.
                                                         4    (9, 9)  18961.0 1182.0
                                                        3L    (155, 155)      25431.0 3.0
                                                        2L    (145, 145)      24438.0 6.0
                                                        2R    (134, 134)      20234.0 1.0
                                                         X    (142, 142)      11447.0 1.0
                                                        3R    (176, 176)      22142.0 11.0

INFO:save_ccmap: Saving ccmap to file [cmaps/binContact/chr4_160kb.ccmap] and [/home/rajendra/workspa
INFO:save_ccmap: Compressing [/home/rajendra/workspace/genome_3d_organization/tutorials_modules/cmaps
INFO:save_ccmap:       Finished!!!

INFO:save_ccmap: Saving ccmap to file [cmaps/binContact/chr3L_160kb.ccmap] and [/home/rajendra/worksp
INFO:save_ccmap: Compressing [/home/rajendra/workspace/genome_3d_organization/tutorials_modules/cmaps
INFO:save_ccmap:       Finished!!!

INFO:save_ccmap: Saving ccmap to file [cmaps/binContact/chr2L_160kb.ccmap] and [/home/rajendra/worksp
INFO:save_ccmap: Compressing [/home/rajendra/workspace/genome_3d_organization/tutorials_modules/cmaps
INFO:save_ccmap:       Finished!!!

INFO:save_ccmap: Saving ccmap to file [cmaps/binContact/chr2R_160kb.ccmap] and [/home/rajendra/worksp
INFO:save_ccmap: Compressing [/home/rajendra/workspace/genome_3d_organization/tutorials_modules/cmaps
INFO:save_ccmap:       Finished!!!

INFO:save_ccmap: Saving ccmap to file [cmaps/binContact/chrX_160kb.ccmap] and [/home/rajendra/workspa
INFO:save_ccmap: Compressing [/home/rajendra/workspace/genome_3d_organization/tutorials_modules/cmaps
INFO:save_ccmap:       Finished!!!

INFO:save_ccmap: Saving ccmap to file [cmaps/binContact/chr3R_160kb.ccmap] and [/home/rajendra/worksp
INFO:save_ccmap: Compressing [/home/rajendra/workspace/genome_3d_organization/tutorials_modules/cmaps
INFO:save_ccmap:       Finished!!!
```

Convert to gcmap

A pair of example input files `nm_none_160000.bins` and `nm_none_160000.n_contact` is present in `data/binContactFormat` directory. Below, we read it and convert it to `.gcmap` formats. The input file contains several chromosomes, all contact map will be added to the output `gcmap`.

Ouput `raw_160kb.gcmap` files will be saved in `cmaps/binContact` directory.

```
In [13]: # Save gcmap
         binContactReader.save_gcmap('cmaps/binContact/raw_160kb.gcmap', coarsingMethod='sum', compre
```

```
INFO:addCCMap2GCMap: Opened file [cmaps/binContact/raw_160kb.gcmap] for reading writing..
INFO:addCCMap2GCMap: Adding data to [cmaps/binContact/raw_160kb.gcmap] for [chr4] ...
INFO:addCCMap2GCMap:     ...Finished adding data for [chr4] ...
INFO:addCCMap2GCMap: Generating downsampled maps for [chr4] ...
INFO:addCCMap2GCMap:     ... Finished downsampling for [chr4] ...
INFO:addCCMap2GCMap: Closed file [cmaps/binContact/raw_160kb.gcmap]...
INFO:addCCMap2GCMap: Opened file [cmaps/binContact/raw_160kb.gcmap] for reading writing..
INFO:addCCMap2GCMap: Adding data to [cmaps/binContact/raw_160kb.gcmap] for [chr3L] ...
INFO:addCCMap2GCMap:     ...Finished adding data for [chr3L] ...
INFO:addCCMap2GCMap: Generating downsampled maps for [chr3L] ...
INFO:addCCMap2GCMap:     ... Finished downsampling for [chr3L] ...
INFO:addCCMap2GCMap: Closed file [cmaps/binContact/raw_160kb.gcmap]...
INFO:addCCMap2GCMap: Opened file [cmaps/binContact/raw_160kb.gcmap] for reading writing..
INFO:addCCMap2GCMap: Adding data to [cmaps/binContact/raw_160kb.gcmap] for [chr2L] ...
INFO:addCCMap2GCMap:     ...Finished adding data for [chr2L] ...
INFO:addCCMap2GCMap: Generating downsampled maps for [chr2L] ...
INFO:addCCMap2GCMap:     ... Finished downsampling for [chr2L] ...
INFO:addCCMap2GCMap: Closed file [cmaps/binContact/raw_160kb.gcmap]...
INFO:addCCMap2GCMap: Opened file [cmaps/binContact/raw_160kb.gcmap] for reading writing..
INFO:addCCMap2GCMap: Adding data to [cmaps/binContact/raw_160kb.gcmap] for [chr2R] ...
INFO:addCCMap2GCMap:     ...Finished adding data for [chr2R] ...
INFO:addCCMap2GCMap: Generating downsampled maps for [chr2R] ...
INFO:addCCMap2GCMap:     ... Finished downsampling for [chr2R] ...
INFO:addCCMap2GCMap: Closed file [cmaps/binContact/raw_160kb.gcmap]...
INFO:addCCMap2GCMap: Opened file [cmaps/binContact/raw_160kb.gcmap] for reading writing..
INFO:addCCMap2GCMap: Adding data to [cmaps/binContact/raw_160kb.gcmap] for [chrX] ...
INFO:addCCMap2GCMap:     ...Finished adding data for [chrX] ...
INFO:addCCMap2GCMap: Generating downsampled maps for [chrX] ...
INFO:addCCMap2GCMap:     ... Finished downsampling for [chrX] ...
INFO:addCCMap2GCMap: Closed file [cmaps/binContact/raw_160kb.gcmap]...
INFO:addCCMap2GCMap: Opened file [cmaps/binContact/raw_160kb.gcmap] for reading writing..
INFO:addCCMap2GCMap: Adding data to [cmaps/binContact/raw_160kb.gcmap] for [chr3R] ...
INFO:addCCMap2GCMap:     ...Finished adding data for [chr3R] ...
INFO:addCCMap2GCMap: Generating downsampled maps for [chr3R] ...
INFO:addCCMap2GCMap:     ... Finished downsampling for [chr3R] ...
INFO:addCCMap2GCMap: Closed file [cmaps/binContact/raw_160kb.gcmap]...
```

How to normalize Hi-C map?

To normalize a Hi-C map, several methods have been implemented.

- Matrix balancing algorithm by Knight and Ruiz
- Matrix balancing by Iterative Correction
- Normalized using Median Contact Frequency Scaling (MCFS)

Note: Following tutorial needs `*.ccmap` files, generated in *previous tutorial*.

See also:

Module *gcMapExplorer.lib.normalizer* for all implemented normalization methods in detail.

Remove old files if any present in normalize directory

```
In [1]: %%bash

        for f in ./normalized/*; do
            [ -e "$f" ] && rm $f
        done
```

Matrix balancing algorithm by Knight and Ruiz

import gcMapExplorer.lib module

```
In [2]: import gcMapExplorer.lib as gmlib
        import numpy as np
        import os
```

Directly Normalize and save ccmap file

```
In [3]: # Name of ccmap file with path
        raw_ccmap_file = 'cmaps/CooMatrix/chr22_100kb_RawObserved.ccmap'

        # Name of output file
        outFile = 'normalized/chr22_100kb_normKR_direct.ccmap'

        # Perform normalization and save to output file
        gmlib.normalizer.normalizeCCMapByKR(raw_ccmap_file, outFile=outFile, memory='RAM')
```

```
INFO:normalizer: KR Normalization will be done through RAM.
INFO:normalizer: KR Normalization is in process for chr22 map...
INFO:normalizer:        ...Finished KR Normalization for chr22 map...
INFO:save_ccmap: Saving ccmap to file [normalized/chr22_100kb_normKR_direct.ccmap] and [/home/rajendr
INFO:save_ccmap: Compressing [/home/rajendra/workspace/genome_3d_organization/tutorials_modules/norma
INFO:save_ccmap:       Finished!!!
```

See also:

Function *gcMapExplorer.lib.normalizer.normalizeCCMapByKR()* for more details.

Normalize and save Hi-C map thorough CCMAP object

- Load the ccmap as CCMAP object
- Normalize it
- Save as ccmap file

```
In [4]: # Load the ccmap file as CCMAP object
        raw_ccmap = gmlib.ccmap.load_ccmap(raw_ccmap_file)

        # Perform normalization and save to output file
        norm_ccmap = gmlib.normalizer.normalizeCCMapByKR(raw_ccmap_file, memory='RAM')
```

```
        # Save ccmap file
        gmlib.ccmap.save_ccmap(norm_ccmap, 'normalized/chr22_100kb_normKR.ccmap', compress=True)

        # Remove CCMAP object from memory and generated temporary files
        del raw_ccmap
        del norm_ccmap
```

```
INFO:normalizer: KR Normalization will be done through RAM.
INFO:normalizer: KR Normalization is in process for chr22 map...
INFO:normalizer:        ...Finished KR Normalization for chr22 map...
INFO:save_ccmap: Saving ccmap to file [normalized/chr22_100kb_normKR.ccmap] and [/home/rajendra/works
INFO:save_ccmap: Compressing [/home/rajendra/workspace/genome_3d_organization/tutorials_modules/norma
INFO:save_ccmap:       Finished!!!
```

See also:

Function *gcMapExplorer.lib.ccmap.load_ccmap()* for more details.

Whether using RAM and HDD yield same result

Here, we test whether using RAM and HDD yield same results. We also calculate total time taken for normalization.

```
In [5]: raw_ccmap = gmlib.ccmap.load_ccmap('cmaps/CooMatrix/chr1_100kb_RawObserved.ccmap')

        print('Time using RAM: ')
        %timeit norm_ram = gmlib.normalizer.normalizeCCMapByKR(raw_ccmap, memory='RAM')

        print('Time using HDD: ')
        %timeit norm_hdd = gmlib.normalizer.normalizeCCMapByKR(raw_ccmap, memory='HDD')

        # Again renormalize
        norm_hdd = gmlib.normalizer.normalizeCCMapByKR(raw_ccmap, memory='HDD')
        norm_ram = gmlib.normalizer.normalizeCCMapByKR(raw_ccmap, memory='RAM')
        norm_ram.make_readable()
        norm_hdd.make_readable()

        print('If matrix from RAM and HDD are similar: ', np.allclose(norm_ram.matrix, norm_hdd.matri
        del raw_ccmap
        del norm_ram
        del norm_hdd
```

```
INFO:normalizer: KR Normalization will be done through RAM.
INFO:normalizer: KR Normalization is in process for chr1 map...
```

```
Time using RAM:
```

```
INFO:normalizer:        ...Finished KR Normalization for chr1 map...
INFO:normalizer: KR Normalization will be done through RAM.
INFO:normalizer: KR Normalization is in process for chr1 map...
INFO:normalizer:        ...Finished KR Normalization for chr1 map...
INFO:normalizer: KR Normalization will be done through RAM.
INFO:normalizer: KR Normalization is in process for chr1 map...
INFO:normalizer:        ...Finished KR Normalization for chr1 map...
INFO:normalizer: KR Normalization will be done through RAM.
INFO:normalizer: KR Normalization is in process for chr1 map...
INFO:normalizer:        ...Finished KR Normalization for chr1 map...
INFO:normalizer: KR Normalization will be done through RAM.
INFO:normalizer: KR Normalization is in process for chr1 map...
INFO:normalizer:        ...Finished KR Normalization for chr1 map...
INFO:normalizer: KR Normalization will be done through RAM.
INFO:normalizer: KR Normalization is in process for chr1 map...
```

```
INFO:normalizer:        ...Finished KR Normalization for chr1 map...
INFO:normalizer: KR Normalization will be done through RAM.
INFO:normalizer: KR Normalization is in process for chr1 map...
INFO:normalizer:        ...Finished KR Normalization for chr1 map...
INFO:normalizer: KR Normalization will be done through RAM.
INFO:normalizer: KR Normalization is in process for chr1 map...
INFO:normalizer:        ...Finished KR Normalization for chr1 map...
INFO:normalizer: KR Normalization will be done through HDD.
INFO:normalizer: KR Normalization is in process for chr1 map...
```

```
6.38 s ± 827 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
Time using HDD:
```

```
INFO:normalizer:        ...Finished KR Normalization for chr1 map...
INFO:normalizer: KR Normalization will be done through HDD.
INFO:normalizer: KR Normalization is in process for chr1 map...
INFO:normalizer:        ...Finished KR Normalization for chr1 map...
INFO:normalizer: KR Normalization will be done through HDD.
INFO:normalizer: KR Normalization is in process for chr1 map...
INFO:normalizer:        ...Finished KR Normalization for chr1 map...
INFO:normalizer: KR Normalization will be done through HDD.
INFO:normalizer: KR Normalization is in process for chr1 map...
INFO:normalizer:        ...Finished KR Normalization for chr1 map...
INFO:normalizer: KR Normalization will be done through HDD.
INFO:normalizer: KR Normalization is in process for chr1 map...
INFO:normalizer:        ...Finished KR Normalization for chr1 map...
INFO:normalizer: KR Normalization will be done through HDD.
INFO:normalizer: KR Normalization is in process for chr1 map...
INFO:normalizer:        ...Finished KR Normalization for chr1 map...
INFO:normalizer: KR Normalization will be done through HDD.
INFO:normalizer: KR Normalization is in process for chr1 map...
INFO:normalizer:        ...Finished KR Normalization for chr1 map...
INFO:normalizer: KR Normalization will be done through HDD.
INFO:normalizer: KR Normalization is in process for chr1 map...
INFO:normalizer:        ...Finished KR Normalization for chr1 map...
INFO:normalizer: KR Normalization will be done through HDD.
INFO:normalizer: KR Normalization is in process for chr1 map...
```

```
20 s ± 988 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
```

```
INFO:normalizer:        ...Finished KR Normalization for chr1 map...
INFO:normalizer: KR Normalization will be done through RAM.
INFO:normalizer: KR Normalization is in process for chr1 map...
INFO:normalizer:        ...Finished KR Normalization for chr1 map...
```

```
If matrix from RAM and HDD are similar:  True
```

Normalize and save all ccmaps

```
In [6]: chroms = [1, 5, 15, 20, 21]      # List of chromosomes

        # Loop for each chromosome
        for chrom in chroms:
            input_file = 'cmaps/CooMatrix/chr{0}_100kb_RawObserved.ccmap' .format(chrom)
            output_file = 'normalized/chr{0}_100kb_normKR.ccmap' .format(chrom)

            raw_ccmap = gmlib.ccmap.load_ccmap(input_file)
            norm_ccmap = gmlib.normalizer.normalizeCCMapByKR(raw_ccmap, memory='RAM', workDir=os.getc
            gmlib.ccmap.save_ccmap(norm_ccmap, output_file, compress=True)
```

```
            del raw_ccmap     # Remove CCMAP object from memory and any related temporary files
            del norm_ccmap    # Remove CCMAP object from memory and any related temporary files
```

```
INFO:normalizer: KR Normalization will be done through RAM.
INFO:normalizer: KR Normalization is in process for chr1 map...
INFO:normalizer:        ...Finished KR Normalization for chr1 map...
INFO:save_ccmap: Saving ccmap to file [normalized/chr1_100kb_normKR.ccmap] and [/home/rajendra/worksp
INFO:save_ccmap: Compressing [/home/rajendra/workspace/genome_3d_organization/tutorials_modules/norma
INFO:save_ccmap:       Finished!!!

INFO:normalizer: KR Normalization will be done through RAM.
INFO:normalizer: KR Normalization is in process for chr5 map...
INFO:normalizer:        ...Finished KR Normalization for chr5 map...
INFO:save_ccmap: Saving ccmap to file [normalized/chr5_100kb_normKR.ccmap] and [/home/rajendra/worksp
INFO:save_ccmap: Compressing [/home/rajendra/workspace/genome_3d_organization/tutorials_modules/norma
INFO:save_ccmap:       Finished!!!

INFO:normalizer: KR Normalization will be done through RAM.
INFO:normalizer: KR Normalization is in process for chr15 map...
INFO:normalizer:        ...Finished KR Normalization for chr15 map...
INFO:save_ccmap: Saving ccmap to file [normalized/chr15_100kb_normKR.ccmap] and [/home/rajendra/works
INFO:save_ccmap: Compressing [/home/rajendra/workspace/genome_3d_organization/tutorials_modules/norma
INFO:save_ccmap:       Finished!!!

INFO:normalizer: KR Normalization will be done through RAM.
INFO:normalizer: KR Normalization is in process for chr20 map...
INFO:normalizer:        ...Finished KR Normalization for chr20 map...
INFO:save_ccmap: Saving ccmap to file [normalized/chr20_100kb_normKR.ccmap] and [/home/rajendra/works
INFO:save_ccmap: Compressing [/home/rajendra/workspace/genome_3d_organization/tutorials_modules/norma
INFO:save_ccmap:       Finished!!!

INFO:normalizer: KR Normalization will be done through RAM.
INFO:normalizer: KR Normalization is in process for chr21 map...
INFO:normalizer:        ...Finished KR Normalization for chr21 map...
INFO:save_ccmap: Saving ccmap to file [normalized/chr21_100kb_normKR.ccmap] and [/home/rajendra/works
INFO:save_ccmap: Compressing [/home/rajendra/workspace/genome_3d_organization/tutorials_modules/norma
INFO:save_ccmap:       Finished!!!
```

All normalized ccmap files are saved in `output` directory.

Normalize all maps from a gcmap file using KR method

Maps stored in a gcmap files can be normalized and stored in another gcmap files. Tolerance is increased to `1e-5` value so that normalized values can be later comapred with original algorithm.

```
In [7]: # Input raw gcmap file
        raw_gcmap_file = 'cmaps/CooMatrix/rawObserved_100kb.gcmap'

        # Name of output gcmap file
        normKR_gcmap_file = 'normalized/normKR_100kb.gcmap'

        # Perform normalization and save to output file
        gmlib.normalizer.normalizeGCMapByKR(raw_gcmap_file, normKR_gcmap_file, tol=1e-4)
```

```
INFO:normalizer: KR Normalization will be done through RAM.
INFO:normalizer: KR Normalization is in process for chr21 map...
INFO:normalizer:        ...Finished KR Normalization for chr21 map...
INFO:addCCMap2GCMap: Opened file [normalized/normKR_100kb.gcmap] for reading writing..
```

```
INFO:addCCMap2GCMap: Adding data to [normalized/normKR_100kb.gcmap] for [chr21] ...
INFO:addCCMap2GCMap:     ...Finished adding data for [chr21] ...
INFO:addCCMap2GCMap: Generating downsampled maps for [chr21] ...
INFO:addCCMap2GCMap:     ... Finished downsampling for [chr21] ...
INFO:addCCMap2GCMap: Closed file [normalized/normKR_100kb.gcmap]...
INFO:normalizer: KR Normalization will be done through RAM.
INFO:normalizer: KR Normalization is in process for chr22 map...
INFO:normalizer:        ...Finished KR Normalization for chr22 map...
INFO:addCCMap2GCMap: Opened file [normalized/normKR_100kb.gcmap] for reading writing..
INFO:addCCMap2GCMap: Adding data to [normalized/normKR_100kb.gcmap] for [chr22] ...
INFO:addCCMap2GCMap:     ...Finished adding data for [chr22] ...
INFO:addCCMap2GCMap: Generating downsampled maps for [chr22] ...
INFO:addCCMap2GCMap:     ... Finished downsampling for [chr22] ...
INFO:addCCMap2GCMap: Closed file [normalized/normKR_100kb.gcmap]...
INFO:normalizer: KR Normalization will be done through RAM.
INFO:normalizer: KR Normalization is in process for chr20 map...
INFO:normalizer:        ...Finished KR Normalization for chr20 map...
INFO:addCCMap2GCMap: Opened file [normalized/normKR_100kb.gcmap] for reading writing..
INFO:addCCMap2GCMap: Adding data to [normalized/normKR_100kb.gcmap] for [chr20] ...
INFO:addCCMap2GCMap:     ...Finished adding data for [chr20] ...
INFO:addCCMap2GCMap: Generating downsampled maps for [chr20] ...
INFO:addCCMap2GCMap:     ... Finished downsampling for [chr20] ...
INFO:addCCMap2GCMap: Closed file [normalized/normKR_100kb.gcmap]...
INFO:normalizer: KR Normalization will be done through RAM.
INFO:normalizer: KR Normalization is in process for chr15 map...
INFO:normalizer:        ...Finished KR Normalization for chr15 map...
INFO:addCCMap2GCMap: Opened file [normalized/normKR_100kb.gcmap] for reading writing..
INFO:addCCMap2GCMap: Adding data to [normalized/normKR_100kb.gcmap] for [chr15] ...
INFO:addCCMap2GCMap:     ...Finished adding data for [chr15] ...
INFO:addCCMap2GCMap: Generating downsampled maps for [chr15] ...
INFO:addCCMap2GCMap:     ... Finished downsampling for [chr15] ...
INFO:addCCMap2GCMap: Closed file [normalized/normKR_100kb.gcmap]...
INFO:normalizer: KR Normalization will be done through RAM.
INFO:normalizer: KR Normalization is in process for chr5 map...
INFO:normalizer:        ...Finished KR Normalization for chr5 map...
INFO:addCCMap2GCMap: Opened file [normalized/normKR_100kb.gcmap] for reading writing..
INFO:addCCMap2GCMap: Adding data to [normalized/normKR_100kb.gcmap] for [chr5] ...
INFO:addCCMap2GCMap:     ...Finished adding data for [chr5] ...
INFO:addCCMap2GCMap: Generating downsampled maps for [chr5] ...
INFO:addCCMap2GCMap:     ... Finished downsampling for [chr5] ...
INFO:addCCMap2GCMap: Closed file [normalized/normKR_100kb.gcmap]...
INFO:normalizer: KR Normalization will be done through RAM.
INFO:normalizer: KR Normalization is in process for chr1 map...
INFO:normalizer:        ...Finished KR Normalization for chr1 map...
INFO:addCCMap2GCMap: Opened file [normalized/normKR_100kb.gcmap] for reading writing..
INFO:addCCMap2GCMap: Adding data to [normalized/normKR_100kb.gcmap] for [chr1] ...
INFO:addCCMap2GCMap:     ...Finished adding data for [chr1] ...
INFO:addCCMap2GCMap: Generating downsampled maps for [chr1] ...
INFO:addCCMap2GCMap:     ... Finished downsampling for [chr1] ...
INFO:addCCMap2GCMap: Closed file [normalized/normKR_100kb.gcmap]...
```

See also:

Function *gcMapExplorer.lib.normalizer.normalizeGCMapByKR()* for more details.

Normalize by Iterative correction method

This method normalize the raw contact map by removing biases from experimental procedure. For more details, see this publication.

See also:

Function *gcMapExplorer.lib.normalizer.normalizeCCMapByIC()* for more details.

```
In [8]: chroms = [1, 5, 15, 20, 21]      # List of chromosomes

        # Loop for each chromosome
        for chrom in chroms:
            input_file = 'cmaps/CooMatrix/chr{0}_100kb_RawObserved.ccmap' .format(chrom)
            output_file = 'normalized/chr{0}_100kb_IC.ccmap' .format(chrom)

            raw_ccmap = gmlib.ccmap.load_ccmap(input_file)
            norm_ccmap = gmlib.normalizer.normalizeCCMapByIC(raw_ccmap)
            gmlib.ccmap.save_ccmap(norm_ccmap, output_file, compress=True)

            del raw_ccmap     # Remove CCMAP object from memory and any related temporary files
            del norm_ccmap    # Remove CCMAP object from memory and any related temporary files
```

```
INFO:normalizer: Iterative Correction is in process for chr1 map...
INFO:normalizer:        ...Finished Iterative Correction for chr1 map...
INFO:save_ccmap: Saving ccmap to file [normalized/chr1_100kb_IC.ccmap] and [/home/rajendra/workspace/
INFO:save_ccmap: Compressing [/home/rajendra/workspace/genome_3d_organization/tutorials_modules/norma
INFO:save_ccmap:       Finished!!!

INFO:normalizer: Iterative Correction is in process for chr5 map...
INFO:normalizer:        ...Finished Iterative Correction for chr5 map...
INFO:save_ccmap: Saving ccmap to file [normalized/chr5_100kb_IC.ccmap] and [/home/rajendra/workspace/
INFO:save_ccmap: Compressing [/home/rajendra/workspace/genome_3d_organization/tutorials_modules/norma
INFO:save_ccmap:       Finished!!!

INFO:normalizer: Iterative Correction is in process for chr15 map...
INFO:normalizer:        ...Finished Iterative Correction for chr15 map...
INFO:save_ccmap: Saving ccmap to file [normalized/chr15_100kb_IC.ccmap] and [/home/rajendra/workspace
INFO:save_ccmap: Compressing [/home/rajendra/workspace/genome_3d_organization/tutorials_modules/norma
INFO:save_ccmap:       Finished!!!

INFO:normalizer: Iterative Correction is in process for chr20 map...
INFO:normalizer:        ...Finished Iterative Correction for chr20 map...
INFO:save_ccmap: Saving ccmap to file [normalized/chr20_100kb_IC.ccmap] and [/home/rajendra/workspace
INFO:save_ccmap: Compressing [/home/rajendra/workspace/genome_3d_organization/tutorials_modules/norma
INFO:save_ccmap:       Finished!!!

INFO:normalizer: Iterative Correction is in process for chr21 map...
INFO:normalizer:        ...Finished Iterative Correction for chr21 map...
INFO:save_ccmap: Saving ccmap to file [normalized/chr21_100kb_IC.ccmap] and [/home/rajendra/workspace
INFO:save_ccmap: Compressing [/home/rajendra/workspace/genome_3d_organization/tutorials_modules/norma
INFO:save_ccmap:       Finished!!!
```

Normalize all maps from a gcmap file using IC method

Maps stored in a gcmap files can be normalized and stored in another gcmap files.

Note: Here we used high tolerance and large number of iteration.

```
In [9]: # Input raw gcmap file
        raw_gcmap_file = 'cmaps/CooMatrix/rawObserved_100kb.gcmap'

        # Name of output gcmap file
        normIC_gcmap_file = 'normalized/normIC_100kb.gcmap'

        # Perform normalization and save to output file
        # Not that here we used high tolerance and large number of iteration
        gmlib.normalizer.normalizeGCMapByIC(raw_gcmap_file, normIC_gcmap_file, tol=1e-4, iteration=30
```

```
INFO:normalizer: Iterative Correction is in process for chr21 map...
INFO:normalizer:        ...Finished Iterative Correction for chr21 map...
INFO:addCCMap2GCMap: Opened file [normalized/normIC_100kb.gcmap] for reading writing..
INFO:addCCMap2GCMap: Adding data to [normalized/normIC_100kb.gcmap] for [chr21] ...
INFO:addCCMap2GCMap:     ...Finished adding data for [chr21] ...
INFO:addCCMap2GCMap: Generating downsampled maps for [chr21] ...
INFO:addCCMap2GCMap:     ... Finished downsampling for [chr21] ...
INFO:addCCMap2GCMap: Closed file [normalized/normIC_100kb.gcmap]...
INFO:normalizer: Iterative Correction is in process for chr22 map...
INFO:normalizer:        ...Finished Iterative Correction for chr22 map...
INFO:addCCMap2GCMap: Opened file [normalized/normIC_100kb.gcmap] for reading writing..
INFO:addCCMap2GCMap: Adding data to [normalized/normIC_100kb.gcmap] for [chr22] ...
INFO:addCCMap2GCMap:     ...Finished adding data for [chr22] ...
INFO:addCCMap2GCMap: Generating downsampled maps for [chr22] ...
INFO:addCCMap2GCMap:     ... Finished downsampling for [chr22] ...
INFO:addCCMap2GCMap: Closed file [normalized/normIC_100kb.gcmap]...
INFO:normalizer: Iterative Correction is in process for chr20 map...
INFO:normalizer:        ...Finished Iterative Correction for chr20 map...
INFO:addCCMap2GCMap: Opened file [normalized/normIC_100kb.gcmap] for reading writing..
INFO:addCCMap2GCMap: Adding data to [normalized/normIC_100kb.gcmap] for [chr20] ...
INFO:addCCMap2GCMap:     ...Finished adding data for [chr20] ...
INFO:addCCMap2GCMap: Generating downsampled maps for [chr20] ...
INFO:addCCMap2GCMap:     ... Finished downsampling for [chr20] ...
INFO:addCCMap2GCMap: Closed file [normalized/normIC_100kb.gcmap]...
INFO:normalizer: Iterative Correction is in process for chr15 map...
INFO:normalizer:        ...Finished Iterative Correction for chr15 map...
INFO:addCCMap2GCMap: Opened file [normalized/normIC_100kb.gcmap] for reading writing..
INFO:addCCMap2GCMap: Adding data to [normalized/normIC_100kb.gcmap] for [chr15] ...
INFO:addCCMap2GCMap:     ...Finished adding data for [chr15] ...
INFO:addCCMap2GCMap: Generating downsampled maps for [chr15] ...
INFO:addCCMap2GCMap:     ... Finished downsampling for [chr15] ...
INFO:addCCMap2GCMap: Closed file [normalized/normIC_100kb.gcmap]...
INFO:normalizer: Iterative Correction is in process for chr5 map...
INFO:normalizer:        ...Finished Iterative Correction for chr5 map...
INFO:addCCMap2GCMap: Opened file [normalized/normIC_100kb.gcmap] for reading writing..
INFO:addCCMap2GCMap: Adding data to [normalized/normIC_100kb.gcmap] for [chr5] ...
INFO:addCCMap2GCMap:     ...Finished adding data for [chr5] ...
INFO:addCCMap2GCMap: Generating downsampled maps for [chr5] ...
INFO:addCCMap2GCMap:     ... Finished downsampling for [chr5] ...
INFO:addCCMap2GCMap: Closed file [normalized/normIC_100kb.gcmap]...
INFO:normalizer: Iterative Correction is in process for chr1 map...
INFO:normalizer:        ...Finished Iterative Correction for chr1 map...
INFO:addCCMap2GCMap: Opened file [normalized/normIC_100kb.gcmap] for reading writing..
INFO:addCCMap2GCMap: Adding data to [normalized/normIC_100kb.gcmap] for [chr1] ...
INFO:addCCMap2GCMap:     ...Finished adding data for [chr1] ...
INFO:addCCMap2GCMap: Generating downsampled maps for [chr1] ...
INFO:addCCMap2GCMap:     ... Finished downsampling for [chr1] ...
INFO:addCCMap2GCMap: Closed file [normalized/normIC_100kb.gcmap]...
```

See also:

Function *gcMapExplorer.lib.normalizer.normalizeGCMapByIC()* for more details.

Normalize by Median Contact Frequency Scaling (MCFS)

This method can be used to scale Hi-C map using median contact values for respective distance between two locations/coordinates. At first, median distance contact frequency for each distance is calculated, and subsequently, the observed contact frequency is scaled (divided) by respective median distance contact frequency.

See also:

Function *gcMapExplorer.lib.normalizer.normalizeCCMapByMCFS()* for more details.

```
In [10]: chroms = [1, 5, 15, 20, 21]      # List of chromosomes

         # Loop for each chromosome
         for chrom in chroms:
             input_file = 'cmaps/CooMatrix/chr{0}_100kb_RawObserved.ccmap' .format(chrom)
             output_file = 'normalized/chr{0}_100kb_MCFS.ccmap' .format(chrom)

             raw_ccmap = gmlib.ccmap.load_ccmap(input_file)
             norm_ccmap = gmlib.normalizer.normalizeCCMapByMCFS(raw_ccmap)
             gmlib.ccmap.save_ccmap(norm_ccmap, output_file, compress=True)

             del raw_ccmap     # Remove CCMAP object from memory and any related temporary files
             del norm_ccmap    # Remove CCMAP object from memory and any related temporary files
INFO:normalizer: Median Contact Frequency Scaling is in process for chr1 map...
INFO:normalizer:        ...Finished Median Contact Frequency Scaling for chr1 map...
INFO:save_ccmap: Saving ccmap to file [normalized/chr1_100kb_MCFS.ccmap] and [/home/rajendra/workspac
INFO:save_ccmap: Compressing [/home/rajendra/workspace/genome_3d_organization/tutorials_modules/norma
INFO:save_ccmap:       Finished!!!

INFO:normalizer: Median Contact Frequency Scaling is in process for chr5 map...
INFO:normalizer:        ...Finished Median Contact Frequency Scaling for chr5 map...
INFO:save_ccmap: Saving ccmap to file [normalized/chr5_100kb_MCFS.ccmap] and [/home/rajendra/workspac
INFO:save_ccmap: Compressing [/home/rajendra/workspace/genome_3d_organization/tutorials_modules/norma
INFO:save_ccmap:       Finished!!!

INFO:normalizer: Median Contact Frequency Scaling is in process for chr15 map...
INFO:normalizer:        ...Finished Median Contact Frequency Scaling for chr15 map...
INFO:save_ccmap: Saving ccmap to file [normalized/chr15_100kb_MCFS.ccmap] and [/home/rajendra/workspa
INFO:save_ccmap: Compressing [/home/rajendra/workspace/genome_3d_organization/tutorials_modules/norma
INFO:save_ccmap:       Finished!!!

INFO:normalizer: Median Contact Frequency Scaling is in process for chr20 map...
INFO:normalizer:        ...Finished Median Contact Frequency Scaling for chr20 map...
INFO:save_ccmap: Saving ccmap to file [normalized/chr20_100kb_MCFS.ccmap] and [/home/rajendra/workspa
INFO:save_ccmap: Compressing [/home/rajendra/workspace/genome_3d_organization/tutorials_modules/norma
INFO:save_ccmap:       Finished!!!

INFO:normalizer: Median Contact Frequency Scaling is in process for chr21 map...
INFO:normalizer:        ...Finished Median Contact Frequency Scaling for chr21 map...
INFO:save_ccmap: Saving ccmap to file [normalized/chr21_100kb_MCFS.ccmap] and [/home/rajendra/workspa
INFO:save_ccmap: Compressing [/home/rajendra/workspace/genome_3d_organization/tutorials_modules/norma
INFO:save_ccmap:       Finished!!!
```

Normalize all maps from a gcmap file using MCFS method

Maps stored in a gcmap files can be normalized and stored in another gcmap files.

```
In [11]: # Name of output gcmap file
         normMCFS_gcmap_file = 'normalized/normMCFS_100kb.gcmap'


         # Perform scaling and save to output file
         gmlib.normalizer.normalizeGCMapByMCFS(raw_gcmap_file, normMCFS_gcmap_file)
```

```
INFO:normalizer: Median Contact Frequency Scaling is in process for chr21 map...
INFO:normalizer:        ...Finished Median Contact Frequency Scaling for chr21 map...
INFO:addCCMap2GCMap: Opened file [normalized/normMCFS_100kb.gcmap] for reading writing..
INFO:addCCMap2GCMap: Adding data to [normalized/normMCFS_100kb.gcmap] for [chr21 - 100kb] ...
INFO:addCCMap2GCMap:     ...Finished adding data for [chr21] ...
INFO:addCCMap2GCMap: Closed file [normalized/normMCFS_100kb.gcmap]...
INFO:normalizer: Median Contact Frequency Scaling is in process for chr22 map...
INFO:normalizer:        ...Finished Median Contact Frequency Scaling for chr22 map...
INFO:addCCMap2GCMap: Opened file [normalized/normMCFS_100kb.gcmap] for reading writing..
INFO:addCCMap2GCMap: Adding data to [normalized/normMCFS_100kb.gcmap] for [chr22 - 100kb] ...
INFO:addCCMap2GCMap:     ...Finished adding data for [chr22] ...
INFO:addCCMap2GCMap: Closed file [normalized/normMCFS_100kb.gcmap]...
INFO:normalizer: Median Contact Frequency Scaling is in process for chr22 map...
INFO:normalizer:        ...Finished Median Contact Frequency Scaling for chr22 map...
INFO:addCCMap2GCMap: Opened file [normalized/normMCFS_100kb.gcmap] for reading writing..
INFO:addCCMap2GCMap: Adding data to [normalized/normMCFS_100kb.gcmap] for [chr22 - 200kb] ...
INFO:addCCMap2GCMap:     ...Finished adding data for [chr22] ...
INFO:addCCMap2GCMap: Closed file [normalized/normMCFS_100kb.gcmap]...
INFO:normalizer: Median Contact Frequency Scaling is in process for chr20 map...
INFO:normalizer:        ...Finished Median Contact Frequency Scaling for chr20 map...
INFO:addCCMap2GCMap: Opened file [normalized/normMCFS_100kb.gcmap] for reading writing..
INFO:addCCMap2GCMap: Adding data to [normalized/normMCFS_100kb.gcmap] for [chr20 - 100kb] ...
INFO:addCCMap2GCMap:     ...Finished adding data for [chr20] ...
INFO:addCCMap2GCMap: Closed file [normalized/normMCFS_100kb.gcmap]...
INFO:normalizer: Median Contact Frequency Scaling is in process for chr20 map...
INFO:normalizer:        ...Finished Median Contact Frequency Scaling for chr20 map...
INFO:addCCMap2GCMap: Opened file [normalized/normMCFS_100kb.gcmap] for reading writing..
INFO:addCCMap2GCMap: Adding data to [normalized/normMCFS_100kb.gcmap] for [chr20 - 200kb] ...
INFO:addCCMap2GCMap:     ...Finished adding data for [chr20] ...
INFO:addCCMap2GCMap: Closed file [normalized/normMCFS_100kb.gcmap]...
INFO:normalizer: Median Contact Frequency Scaling is in process for chr15 map...
INFO:normalizer:        ...Finished Median Contact Frequency Scaling for chr15 map...
INFO:addCCMap2GCMap: Opened file [normalized/normMCFS_100kb.gcmap] for reading writing..
INFO:addCCMap2GCMap: Adding data to [normalized/normMCFS_100kb.gcmap] for [chr15 - 100kb] ...
INFO:addCCMap2GCMap:     ...Finished adding data for [chr15] ...
INFO:addCCMap2GCMap: Closed file [normalized/normMCFS_100kb.gcmap]...
INFO:normalizer: Median Contact Frequency Scaling is in process for chr15 map...
INFO:normalizer:        ...Finished Median Contact Frequency Scaling for chr15 map...
INFO:addCCMap2GCMap: Opened file [normalized/normMCFS_100kb.gcmap] for reading writing..
INFO:addCCMap2GCMap: Adding data to [normalized/normMCFS_100kb.gcmap] for [chr15 - 200kb] ...
INFO:addCCMap2GCMap:     ...Finished adding data for [chr15] ...
INFO:addCCMap2GCMap: Closed file [normalized/normMCFS_100kb.gcmap]...
INFO:normalizer: Median Contact Frequency Scaling is in process for chr15 map...
INFO:normalizer:        ...Finished Median Contact Frequency Scaling for chr15 map...
INFO:addCCMap2GCMap: Opened file [normalized/normMCFS_100kb.gcmap] for reading writing..
INFO:addCCMap2GCMap: Adding data to [normalized/normMCFS_100kb.gcmap] for [chr15 - 400kb] ...
INFO:addCCMap2GCMap:     ...Finished adding data for [chr15] ...
```

```
INFO:addCCMap2GCMap: Closed file [normalized/normMCFS_100kb.gcmap]...
INFO:normalizer: Median Contact Frequency Scaling is in process for chr5 map...
INFO:normalizer:        ...Finished Median Contact Frequency Scaling for chr5 map...
INFO:addCCMap2GCMap: Opened file [normalized/normMCFS_100kb.gcmap] for reading writing..
INFO:addCCMap2GCMap: Adding data to [normalized/normMCFS_100kb.gcmap] for [chr5 - 100kb] ...
INFO:addCCMap2GCMap:     ...Finished adding data for [chr5] ...
INFO:addCCMap2GCMap: Closed file [normalized/normMCFS_100kb.gcmap]...
INFO:normalizer: Median Contact Frequency Scaling is in process for chr5 map...
INFO:normalizer:        ...Finished Median Contact Frequency Scaling for chr5 map...
INFO:addCCMap2GCMap: Opened file [normalized/normMCFS_100kb.gcmap] for reading writing..
INFO:addCCMap2GCMap: Adding data to [normalized/normMCFS_100kb.gcmap] for [chr5 - 200kb] ...
INFO:addCCMap2GCMap:     ...Finished adding data for [chr5] ...
INFO:addCCMap2GCMap: Closed file [normalized/normMCFS_100kb.gcmap]...
INFO:normalizer: Median Contact Frequency Scaling is in process for chr5 map...
INFO:normalizer:        ...Finished Median Contact Frequency Scaling for chr5 map...
INFO:addCCMap2GCMap: Opened file [normalized/normMCFS_100kb.gcmap] for reading writing..
INFO:addCCMap2GCMap: Adding data to [normalized/normMCFS_100kb.gcmap] for [chr5 - 400kb] ...
INFO:addCCMap2GCMap:     ...Finished adding data for [chr5] ...
INFO:addCCMap2GCMap: Closed file [normalized/normMCFS_100kb.gcmap]...
INFO:normalizer: Median Contact Frequency Scaling is in process for chr1 map...
INFO:normalizer:        ...Finished Median Contact Frequency Scaling for chr1 map...
INFO:addCCMap2GCMap: Opened file [normalized/normMCFS_100kb.gcmap] for reading writing..
INFO:addCCMap2GCMap: Adding data to [normalized/normMCFS_100kb.gcmap] for [chr1 - 100kb] ...
INFO:addCCMap2GCMap:     ...Finished adding data for [chr1] ...
INFO:addCCMap2GCMap: Closed file [normalized/normMCFS_100kb.gcmap]...
INFO:normalizer: Median Contact Frequency Scaling is in process for chr1 map...
INFO:normalizer:        ...Finished Median Contact Frequency Scaling for chr1 map...
INFO:addCCMap2GCMap: Opened file [normalized/normMCFS_100kb.gcmap] for reading writing..
INFO:addCCMap2GCMap: Adding data to [normalized/normMCFS_100kb.gcmap] for [chr1 - 200kb] ...
INFO:addCCMap2GCMap:     ...Finished adding data for [chr1] ...
INFO:addCCMap2GCMap: Closed file [normalized/normMCFS_100kb.gcmap]...
INFO:normalizer: Median Contact Frequency Scaling is in process for chr1 map...
INFO:normalizer:        ...Finished Median Contact Frequency Scaling for chr1 map...
INFO:addCCMap2GCMap: Opened file [normalized/normMCFS_100kb.gcmap] for reading writing..
INFO:addCCMap2GCMap: Adding data to [normalized/normMCFS_100kb.gcmap] for [chr1 - 400kb] ...
INFO:addCCMap2GCMap:     ...Finished adding data for [chr1] ...
INFO:addCCMap2GCMap: Closed file [normalized/normMCFS_100kb.gcmap]...
INFO:normalizer: Median Contact Frequency Scaling is in process for chr1 map...
INFO:normalizer:        ...Finished Median Contact Frequency Scaling for chr1 map...
INFO:addCCMap2GCMap: Opened file [normalized/normMCFS_100kb.gcmap] for reading writing..
INFO:addCCMap2GCMap: Adding data to [normalized/normMCFS_100kb.gcmap] for [chr1 - 800kb] ...
INFO:addCCMap2GCMap:     ...Finished adding data for [chr1] ...
INFO:addCCMap2GCMap: Closed file [normalized/normMCFS_100kb.gcmap]...
```

See also:

Function *gcMapExplorer.lib.normalizer.normalizeGCMapByIC()* for more details.

How to access Hi-C map from .ccmap file?

`.ccmap` is a text file and associated `*.npbin` or `*.npbin.gz` is memory mapped matrix file.

Note: Following example needs `*.ccmap` file, generated in *previous tutorial*.

See also:

About `*.ccmap` and `*.npbin` (compressed `*.npbin.gz`) files *here*.

At first, we import modules:

- gcMapExplorer.lib
- numpy for statistics
- matplotlib for plotting

```
In [1]: import gcMapExplorer.lib as gmlib
        import numpy as np
        import matplotlib.pyplot as plt

        # To show inline plots
        %matplotlib inline
        plt.style.use('ggplot')              # Theme for plotting
```

Load a `.ccmap` file

```
In [2]: ccmap = gmlib.ccmap.load_ccmap('normalized/chr15_100kb_normKR.ccmap')
```

Print some properties of Hi-C data

```
In [3]: print('shape: ', ccmap.shape)              # Shape of matrix along X and Y axis
        print('Minimum value: ', ccmap.minvalue)  # Maximum value in Hi-C data
        print('Maximum value: ', ccmap.maxvalue)  # Minimum value in Hi.C data
        print('data-type: ', ccmap.dtype)         #  Data type for memory mapped matrix file
        print('path to matrix file:', ccmap.path2matrix)
```

```
shape:  (1026, 1026)
Minimum value:  4.66654819319956e-06
Maximum value:  0.8729197978973389
data-type:  float32
path to matrix file: /home/rajendra/deskForWork/scratch/npBinary_g20uc8yo.tmp
```

Reading `*.npbin` file

```
In [4]: ccmap.make_readable()         # npbin file is now readable
```

Now, Hi-C matrix is available as `ccmap.matrix`.

Overview of Hi-C matrix

```
In [5]: print(ccmap.matrix)
```

```
[[ 0.          0.          0.         ...,  0.          0.          0.        ]
 [ 0.          0.          0.         ...,  0.          0.          0.        ]
 [ 0.          0.          0.         ...,  0.          0.          0.        ]
 ...,
 [ 0.          0.          0.         ...,  0.27888188  0.13513692
   0.07627077]
 [ 0.          0.          0.         ...,  0.13513692  0.45601341  0.        ]
 [ 0.          0.          0.         ...,  0.07627077  0.          0.        ]]
```

Using numpy module

- We can use numpy module to compare properties from `.ccmap` file and from `.npbin` file.
- To find maximum and minimum of matrix, numpy functions amin and amax can be used.

```
In [6]: print('shape: ', ccmap.shape, ccmap.matrix.shape)                # Shape of matrix along X ar
        print('Minimum value: ', ccmap.minvalue, np.amin(ccmap.matrix))  # Minimum value in Hi-C data
        print('Maximum value: ', ccmap.maxvalue, np.amax(ccmap.matrix))  # Maximum value in Hi.C data


shape:  (1026, 1026) (1026, 1026)
Minimum value:  4.66654819319956e-06 0.0
Maximum value:  0.8729197978973389 0.87292
```

Remove rows/columns with missing data

```
In [7]: bData = ~ccmap.bNoData                              # Stores whther rows/columns has missing data
        new_matrix = (ccmap.matrix[bData,:])[:,bData]       # Getting new matrix after removing row/colum
        index_bData = np.nonzero(bData)[0]                   # Getting indices of original matrix after r

        print('Original shape: ', ccmap.matrix.shape)       # Shape of original matrix
        print('New shape: ', new_matrix.shape)               # Shape of new matrix

Original shape:  (1026, 1026)
New shape:  (822, 822)
```

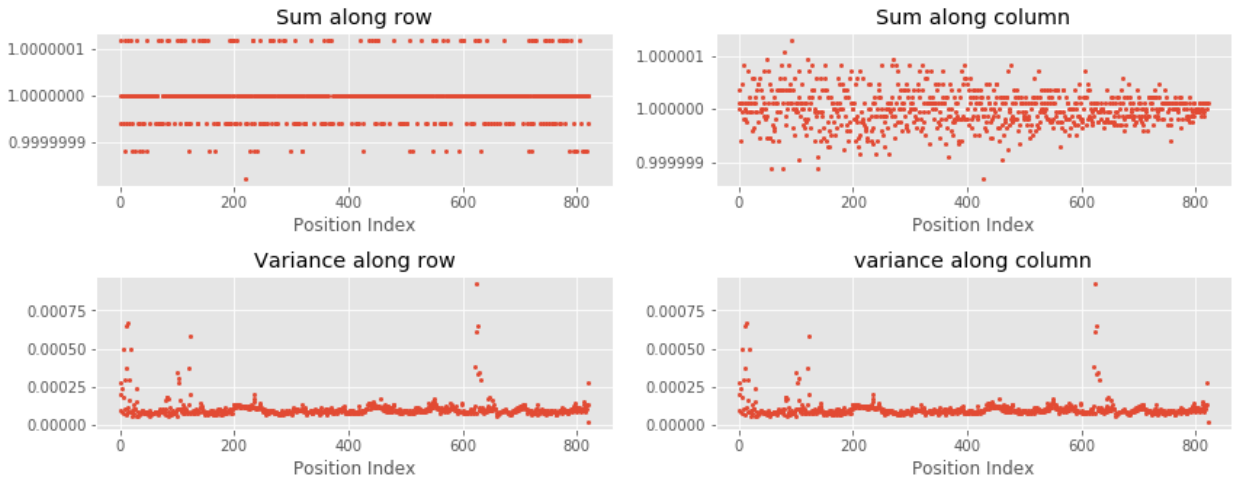
Warning: Above operation cannot be performed on raw Hi-C map because `CCMAP.bNoData` is only avaialable after normalization.

Whether matrix is balanced after KR normalization?

If matrix is balanced, sum of rows and coloumns should be one and variance should be almost equal. Sum and variance of rows and columns can be easily calculated using numpy.sum and numpy.var functions, respectively.

```
In [8]: r_sum = np.sum(new_matrix, axis = 0)                  # Sum along row using numpy.sum
        r_var = np.var(new_matrix, axis = 0)                  # Variance along row using numpy.var

        c_sum = np.sum(new_matrix, axis = 1)                  # Sum along column using numpy.sum
        c_var = np.var(new_matrix, axis = 1)                  # Variance along column using numpy.var

        # Plot the values for visual representations
        fig = plt.figure(figsize=(14,5))                                  # Figure size
        fig.subplots_adjust(hspace=0.6)                                   # Space between plots

        ax1 = fig.add_subplot(2,2,1)                                      # Axes first plot
        ax1.set_title('Sum along row')                                    # Title first plot
        ax1.set_xlabel('Position Index')                                  # X-label
```

```
ax2 = fig.add_subplot(2,2,2)                                    # Axes second plot
ax2.set_title('Sum along column')
ax2.set_xlabel('Position Index')

ax3 = fig.add_subplot(2,2,3)                                    # Axes third plot
ax3.set_title('Variance along row')
ax3.set_xlabel('Position Index')

ax4 = fig.add_subplot(2,2,4)                                    # Axes fourth plot
ax4.set_title('variance along column')
ax4.set_xlabel('Position Index')

ax1.plot(r_sum, marker='o', lw=0, ms=2)                         # Plot in first axes
ax2.plot(c_sum, marker='o', lw=0, ms=2)                         # Plot in second axes
ax3.plot(r_var, marker='o', lw=0, ms=2)                         # Plot in third axes
ax4.plot(c_var, marker='o', lw=0, ms=2)                         # Plot in fourth axes

ax1.get_yaxis().get_major_formatter().set_useOffset(False)      # Prevent ticks auto-formattin
ax2.get_yaxis().get_major_formatter().set_useOffset(False)
ax3.get_yaxis().get_major_formatter().set_useOffset(False)
ax4.get_yaxis().get_major_formatter().set_useOffset(False)

plt.show()
```



Result

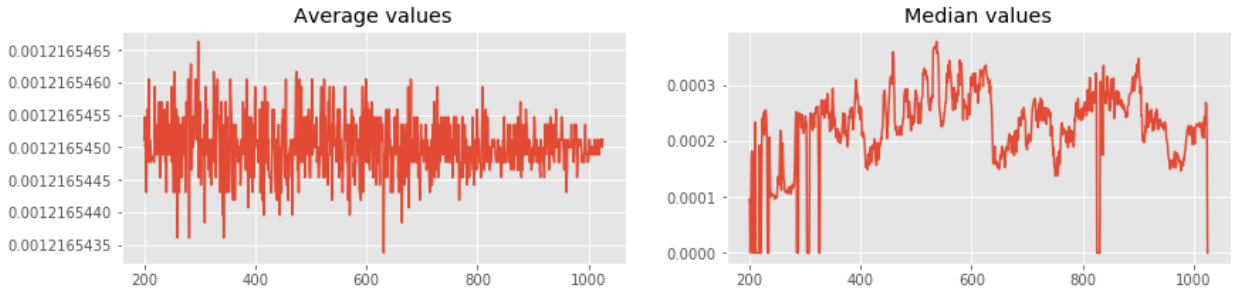
As can be seen in the above plots, sums of rows and columns are equal to one. Additionally, variances of rows and columns are also almost equal.

Using more numpy modules

Lets plot average and median of each rows using numpy.mean and numpy.median.

```
In [9]: averages = np.mean(new_matrix, axis = 1)            # Calculating mean using numpy.mean
        medians = np.median(new_matrix, axis = 1)           # Calculating median using numpy.median

        # Plot the values for visual representations
        fig = plt.figure(figsize=(14,3))                    # Figure size
```

```
ax1 = fig.add_subplot(1,2,1)                                    # Axes first plot
ax1.set_title('Average values')                                  # Title first plot
ax1.get_yaxis().get_major_formatter().set_useOffset(False)      # Prevent ticks auto-formattin

ax2 = fig.add_subplot(1,2,2)                                    # Axes second plot
ax2.set_title('Median values')
ax2.get_yaxis().get_major_formatter().set_useOffset(False)

# in below both plots, x-axis is index from original matrix to preserve original location
ax1.plot(index_bData, averages)    # Plot in first axes
ax2.plot(index_bData, medians)     # Plot in second axes

plt.show()
```



Using masked array with Hi-C map

Large Hi-C map data contains missing values. During analysis, sometimes we need to ignore these values. numpy.ma module can be used to perform mathematical operations after masking these missing values.

See also:

Module numpy.ma

At first, we import modules:

- gcMapExplorer.ccmap for loading `.ccmap` files
- numpy for array operations
- numpy.ma for masked array module
- scipy.stats.mstats for correlation calculation using masked array

```
In [1]: import gcMapExplorer.lib as gmlib
        import numpy as np
        import os
        from numpy import ma
        from scipy import stats

        import matplotlib.pyplot as plt

        # To show inline plots
        %matplotlib inline
        plt.style.use('ggplot')              # Theme for plotting
```

To calculate correlation between Hi-C maps

Load two Hi-C data

```
In [2]: ccmapOne = gmlib.ccmap.load_ccmap('cmaps/CooMatrix/chr15_100kb_RawObserved.ccmap')
        ccmapOne.make_readable()

        ccmapTwo = gmlib.ccmap.load_ccmap('normalized/chr15_100kb_normKR.ccmap')
        ccmapTwo.make_readable()
```

Determine smallest shape

Matrix size can be different, therefore smallest size is determined to calculate element-wise correlation.

```
In [3]: if ccmapOne.shape[0] <= ccmapTwo.shape[0]:
            smallest_shape = ccmapOne.shape[0]
        else:
            smallest_shape = ccmapTwo.shape[0]
```

Generate masks

- At first, generate mask from two matrices seprately, and then combine it.
- Also, use slicing operations to derive subset of matrix with smallest shape.
- During masking, lower-triangular part is also masked with diagonal offset of five. Because the values at Hi-C map diagnoal is usually large, correlation could be high due to these large values. Moreover, we are usually intereseted in off-diagonal regions of the Hi-C maps.

```
In [4]: m1 = ccmapOne.matrix[:smallest_shape, :smallest_shape] <= ccmapOne.minvalue   # Mask all mini
        m2 = ccmapTwo.matrix[:smallest_shape, :smallest_shape] <= ccmapTwo.minvalue   # Mask all mini
        mask = ( m1 & m2 )                                    # Combine both masks
        mask[np.tril_indices_from(mask, k=5)] = True    # Also, Mask lower-triangle of matrix with fi
```

Warning: For huge matrices, above operations could be memory/RAM consuming and script may crash.

See also:

- How to perform slicing?
- How to generate boolean mask?
- How to get indices of lower-triangle of an array?

Generate masked matrix

Now, using `numpy.ma` module, generate masked matrices

```
In [5]: maskedMatrixOne = ma.array(ccmapOne.matrix[:smallest_shape, :smallest_shape], mask=mask)
        maksedMatrixTwo = ma.array(ccmapTwo.matrix[:smallest_shape, :smallest_shape], mask=mask)
```

Calculate correlation coefficiant

- Pearson correlation coefficiant using scipy.stats.mstats.pearsonr

```
In [6]: corr, pvalue = stats.pearsonr(maskedMatrixOne.compressed(), maksedMatrixTwo.compressed())

        print('Pearson Correlation: ', corr)

        corr, pvalue = stats.spearmanr(maskedMatrixOne.compressed(), maksedMatrixTwo.compressed())
```

```
        print('Spearman Correlation: ', corr)
```

```
Pearson Correlation:  0.580994
Spearman Correlation:  0.837034386661
```

Crorrelation using gcMapExplorer.lib.cmstats module

Shown above is a simple step-by-step example to calculate correlation-coefficiant using masked array. However, above method may fail in case of huge matrices. Therefore, use the implemented function `gcMapExplorer.cmstats.correlateCMaps` function

See also:

- About *gcMapExplorer.lib.cmstats.correlateCMaps()* in more details

```
In [7]: corr, pvalue = gmlib.cmstats.correlateCMaps(ccmapOne, ccmapTwo, diagonal_offset=5)

        print('Pearson Correlation: ', corr)

        corr, pvalue = gmlib.cmstats.correlateCMaps(ccmapOne, ccmapTwo, corrType='spearman', diagonal
        print('Spearman Correlation: ', corr)
```

```
Pearson Correlation:  0.580994
Spearman Correlation:  0.837035735987
```

Both above shown step-by-step example and implemented functions yielded similar correlation values.

Block-wise Crorrelation

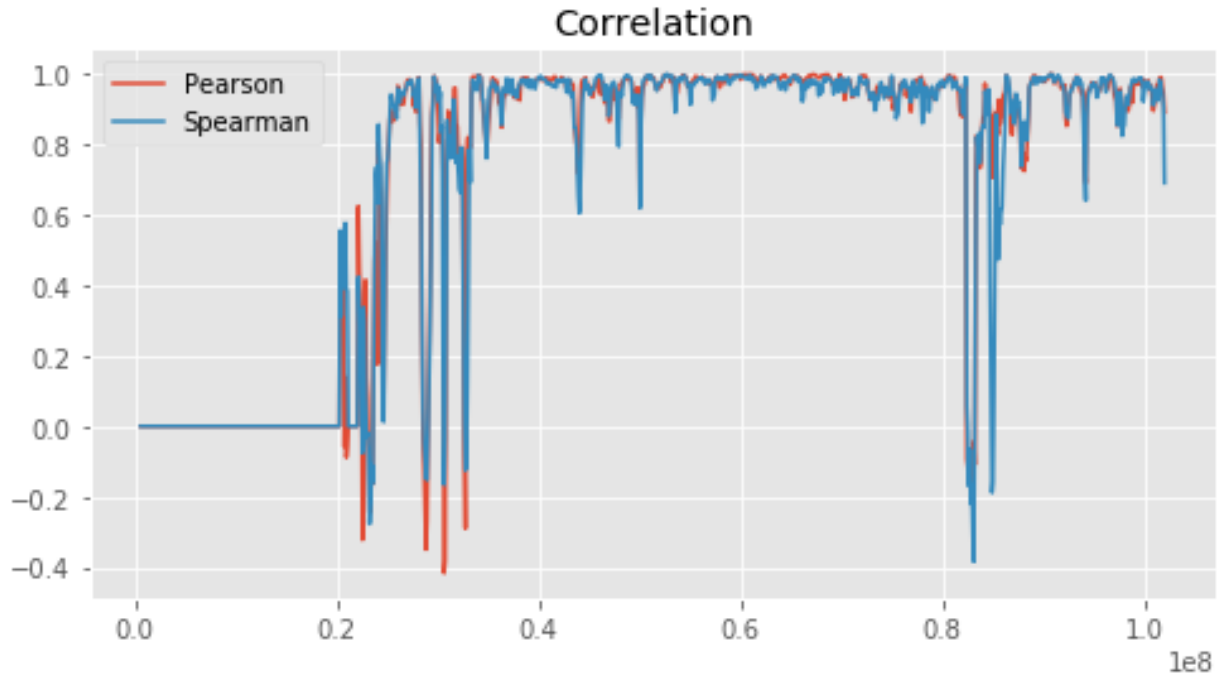
To identify local difference between two maps, block-wise coorelation could be more helpful. A block is created and slided along the diagonal, and for each new position, correlation is calculated.

```
In [8]: # Pearson correlation
        pearson, p_centers = gmlib.cmstats.correlateCMaps(ccmapOne, ccmapTwo, diagonal_offset=2, bloc
                                                          slideStepSize=1, outFile='pearson.txt', work


        # Spearman correlation
        spearman, s_centers = gmlib.cmstats.correlateCMaps(ccmapOne, ccmapTwo, corrType='spearman', c
                                                           blockSize='1mb', slideStepSize=1, outFil
                                                           workDir=os.getcwd())
```

```
INFO:correlateCMaps: Block-wise correlation with [1mb] block-size
INFO:correlateCMaps: Number of Blocks: 102
INFO:correlateCMaps: Size of each Block in bins: 10
INFO:correlateCMaps: Number of Overlapping bins between sliding blocks:  9
INFO:correlateCMaps: Block-wise correlation with [1mb] block-size
INFO:correlateCMaps: Number of Blocks: 102
INFO:correlateCMaps: Size of each Block in bins: 10
INFO:correlateCMaps: Number of Overlapping bins between sliding blocks:  9
```

```
In [9]: # Plot the values for visual representations
        fig = plt.figure(figsize=(8,4))                                    # Figure size

        ax = fig.add_subplot(1,1,1)                                        # Axes first plot
        ax.set_title('Correlation')                                        # Title first plot

        ax.plot(p_centers, pearson, label='Pearson')                       # Plot for Pearson correlation
```

```
        ax.plot(s_centers, spearman, label='Spearman')                # Plot for Spearman correlation

        ax.get_xaxis().get_major_formatter().set_useOffset(False)     # Prevent ticks auto-formatting

        plt.legend(loc=2)

        plt.show()
```



Calculating correlation from gcmap

Correlation between maps present in two gcmap files could be readily calculated as follows. Below is the example where at first correlation between raw and KR normalized maps are calculated. Further below, correlation between raw and IC maps are calculated.

```
In [10]: gcMapOne = 'cmaps/CooMatrix/rawObserved_100kb.gcmap'
         gcMapTwo = 'normalized/normKR_100kb.gcmap'
         gcMapThree = 'normalized/normIC_100kb.gcmap'

         mapList, corrsKR, pvaluesKR = gmlib.cmstats.correlateGCMaps(gcMapOne, gcMapTwo)
         mapList, corrsIC, pvaluesIC = gmlib.cmstats.correlateGCMaps(gcMapOne, gcMapThree)

         print('Pearson Correlation Raw vs KR/IC normalized')
         print('===========================================')
         print('\nChromosome     Corr Raw-KR     Corr Raw-IC')
         print('-----------------------------------------------------')
         for i in range(len(mapList)):
             print('{0:5} {1:16.5} {2:15.5}'.format( mapList[i], corrsKR[i], corrsIC[i]))
         print('-----------------------------------------------------')
```

```
INFO:correlateGCMaps: Performing calculation for chr1 ...
INFO:correlateGCMaps:     Finished calculation for chr1 ...
```

```
INFO:correlateGCMaps: Performing calculation for chr5 ...
INFO:correlateGCMaps:     Finished calculation for chr5 ...

INFO:correlateGCMaps: Performing calculation for chr15 ...
INFO:correlateGCMaps:     Finished calculation for chr15 ...

INFO:correlateGCMaps: Performing calculation for chr20 ...
INFO:correlateGCMaps:     Finished calculation for chr20 ...

INFO:correlateGCMaps: Performing calculation for chr21 ...
INFO:correlateGCMaps:     Finished calculation for chr21 ...

INFO:correlateGCMaps: Performing calculation for chr22 ...
INFO:correlateGCMaps:     Finished calculation for chr22 ...

INFO:correlateGCMaps: Performing calculation for chr1 ...
INFO:correlateGCMaps:     Finished calculation for chr1 ...

INFO:correlateGCMaps: Performing calculation for chr5 ...
INFO:correlateGCMaps:     Finished calculation for chr5 ...

INFO:correlateGCMaps: Performing calculation for chr15 ...
INFO:correlateGCMaps:     Finished calculation for chr15 ...

INFO:correlateGCMaps: Performing calculation for chr20 ...
INFO:correlateGCMaps:     Finished calculation for chr20 ...

INFO:correlateGCMaps: Performing calculation for chr21 ...
INFO:correlateGCMaps:     Finished calculation for chr21 ...

INFO:correlateGCMaps: Performing calculation for chr22 ...
INFO:correlateGCMaps:     Finished calculation for chr22 ...


Pearson Correlation Raw vs KR/IC normalized
===========================================

Chromosome     Corr Raw-KR     Corr Raw-IC
-----------------------------------------------------
chr1             0.729         0.72935
chr5           0.77334         0.77344
chr15           0.7977          0.7977
chr20          0.96336         0.96336
chr21          0.82557         0.82557
chr22          0.89011         0.89011
-----------------------------------------------------
```

```
In [11]: gcMapOne = 'cmaps/CooMatrix/rawObserved_100kb.gcmap'
         gcMapTwo = 'normalized/normKR_100kb.gcmap'
         gcMapThree = 'normalized/normIC_100kb.gcmap'

         mapList, corrsKR, pvaluesKR = gmlib.cmstats.correlateGCMaps(gcMapOne, gcMapTwo, corrType='sp
         mapList, corrsIC, pvaluesIC = gmlib.cmstats.correlateGCMaps(gcMapOne, gcMapThree, corrType=

         print('Spearman Correlation Raw vs KR/IC normalized')
         print('===========================================')
         print('\nChromosome     Corr Raw-KR     Corr Raw-IC')
         print('-----------------------------------------------------')
         for i in range(len(mapList)):
             print('{0:5} {1:16.5} {2:15.5}'.format( mapList[i], corrsKR[i], corrsIC[i]))
```

```
        print('-------------------------------------------------------')
```

```
INFO:correlateGCMaps: Performing calculation for chr1 ...
INFO:correlateGCMaps:     Finished calculation for chr1 ...

INFO:correlateGCMaps: Performing calculation for chr5 ...
INFO:correlateGCMaps:     Finished calculation for chr5 ...

INFO:correlateGCMaps: Performing calculation for chr15 ...
INFO:correlateGCMaps:     Finished calculation for chr15 ...

INFO:correlateGCMaps: Performing calculation for chr20 ...
INFO:correlateGCMaps:     Finished calculation for chr20 ...

INFO:correlateGCMaps: Performing calculation for chr21 ...
INFO:correlateGCMaps:     Finished calculation for chr21 ...

INFO:correlateGCMaps: Performing calculation for chr22 ...
INFO:correlateGCMaps:     Finished calculation for chr22 ...

INFO:correlateGCMaps: Performing calculation for chr1 ...
INFO:correlateGCMaps:     Finished calculation for chr1 ...

INFO:correlateGCMaps: Performing calculation for chr5 ...
INFO:correlateGCMaps:     Finished calculation for chr5 ...

INFO:correlateGCMaps: Performing calculation for chr15 ...
INFO:correlateGCMaps:     Finished calculation for chr15 ...

INFO:correlateGCMaps: Performing calculation for chr20 ...
INFO:correlateGCMaps:     Finished calculation for chr20 ...

INFO:correlateGCMaps: Performing calculation for chr21 ...
INFO:correlateGCMaps:     Finished calculation for chr21 ...

INFO:correlateGCMaps: Performing calculation for chr22 ...
INFO:correlateGCMaps:     Finished calculation for chr22 ...


Spearman Correlation Raw vs KR/IC normalized
============================================

Chromosome     Corr Raw-KR     Corr Raw-IC
-------------------------------------------------------
chr1           0.91239         0.91239
chr5           0.94794         0.94794
chr15          0.83996         0.83996
chr20           0.9446          0.9446
chr21          0.86788         0.86788
chr22          0.82516         0.82516
-------------------------------------------------------
```

Export .ccmap as text file

Presently only COO list format is implemented in `gcMapExplorer.ccmap.export_ccmap` function. In COO format, lists of (row, column, value) as three tab seprated columns are written in output file.

See also:

Module `gcMapExplorer.ccmap.export_ccmap()`

```
In [1]: import gcMapExplorer.lib as gmlib
```

```
In [2]: # Input files and path
        inputPath='normalized/'
        files=  ['chr5_100kb_normKR.ccmap', 'chr15_100kb_normKR.ccmap',
                 'chr20_100kb_normKR.ccmap', 'chr21_100kb_normKR.ccmap']

        #Output files and path
        outputPath = 'export/'
        outputs=['chr5_100kb.txt',    'chr15_100kb.txt',  'chr20_100kb.txt',  'chr21_100kb.txt']


        for i, o in zip(files, outputs):
            ccmap = gmlib.ccmap.load_ccmap(inputPath+i)
            gmlib.ccmap.export_cmap(ccmap, outputPath+o, doNotWriteZeros=True)
```

How to access Hi-C map from .gcmap file?

`.gcmap` is a HDF5 format file.

Note: Following example needs `*.ccmap` file, generated in *previous tutorial*.

At first, we import modules:

- gcMapExplorer.lib
- numpy for statistics
- matplotlib for plotting

```
In [1]: import gcMapExplorer.lib as gmlib
        import numpy as np
        import matplotlib.pyplot as plt

        # To show inline plots
        %matplotlib inline
        plt.style.use('ggplot')              # Theme for plotting
```

Load a `.gcmap` file

- At first load a map of chromosome from gcmap file using GCMAP class.
- Also, load it as ccmap to compare.

```
In [2]: #filename = 'cmaps/CooMatrix/rawObserved_100kb.gcmap'
        filename = 'normalized/normKR_100kb.gcmap'

        # Load through GCMAP class
        gcmap = gmlib.gcmap.GCMAP(filename, mapName='chr21')

        # Load as a CCMAP class
        ccmap = gmlib.gcmap.loadGCMapAsCCMap(filename, mapName='chr21')
```

Print some properties of Hi-C data

```
In [3]: for key in gcmap.__dict__:
            print(key, ' : ', gcmap.__dict__[key])

minvalue  :  7.40758559914e-06
resolution  :  100kb
finestResolution  :  100kb
matrix  :  <HDF5 dataset "100kb": shape (482, 482), type "<f4">
binsize  :  100000
xlabel  :  chr21
yticks  :  [0, 48200000]
mapNameList  :  None
bLog  :  False
groupName  :  chr21
fileOpened  :  True
binsizes  :  [100000]
hdf5  :  <HDF5 file "normKR_100kb.gcmap" (mode r+)>
ylabel  :  chr21
bNoData  :  [ True  True  True  True  True  True  True  True  True  True  True  True
  True  True  True  True  True  True  True  True  True  True  True  True
  True  True  True  True  True  True  True  True  True  True  True  True
  True  True  True  True  True  True  True  True  True  True  True  True
  True  True  True  True  True  True  True  True  True  True  True  True
  True  True  True  True  True  True  True  True  True  True  True  True
  True  True  True  True  True  True  True  True  True  True  True  True
  True  True  True  True  True  True  True  True  True  True False False
 False False False False False False False False False False False False
 False False False False  True  True  True  True  True  True  True  True
  True  True  True  True  True  True  True  True  True  True  True  True
  True  True  True  True  True  True  True  True  True  True  True False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False]
dtype  :  float32
```

```
shape   :   (482, 482)
title   :   chr21_vs_chr21
xticks   :   [0, 48200000]
mapType   :   intra
maxvalue   :   0.897768378258
```

Reading contact map

Contact matrix is available as `gcmap.matrix` as similar to that of `ccmap.matrix`.

```
In [4]: print(gcmap.matrix[:])

        ccmap.make_readable()
        print(ccmap.matrix)
```

```
[[ 0.          0.          0.         ...,  0.          0.          0.        ]
 [ 0.          0.          0.         ...,  0.          0.          0.        ]
 [ 0.          0.          0.         ...,  0.          0.          0.        ]
 ...,
 [ 0.          0.          0.         ...,  0.33700117  0.13242508
   0.0245942 ]
 [ 0.          0.          0.         ...,  0.13242508  0.32248676
   0.10111635]
 [ 0.          0.          0.         ...,  0.0245942   0.10111635
   0.7121914 ]]
[[ 0.          0.          0.         ...,  0.          0.          0.        ]
 [ 0.          0.          0.         ...,  0.          0.          0.        ]
 [ 0.          0.          0.         ...,  0.          0.          0.        ]
 ...,
 [ 0.          0.          0.         ...,  0.33700117  0.13242508
   0.0245942 ]
 [ 0.          0.          0.         ...,  0.13242508  0.32248676
   0.10111635]
 [ 0.          0.          0.         ...,  0.0245942   0.10111635
   0.7121914 ]]
```

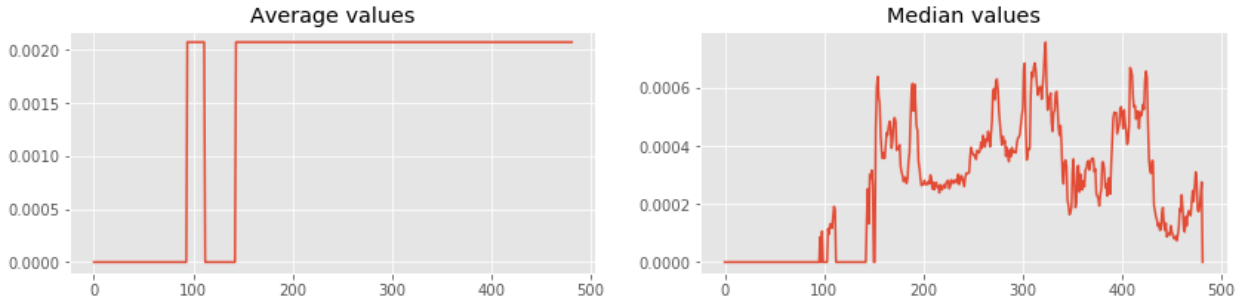
As can be seen in the above plot, sum of rows/columns are approximately one. It means that the matrix is balanced.

Using numpy modules

Lets plot average and median of each rows using numpy.mean and numpy.median.

```
In [5]: averages = np.mean(gcmap.matrix, axis = 1)            # Calculating mean using numpy.mean
        medians = np.median(gcmap.matrix, axis = 0)           # Calculating median using numpy.median

        # Plot the values for visual representations
        fig = plt.figure(figsize=(14,3))                                  # Figure size

        ax1 = fig.add_subplot(1,2,1)                                      # Axes first plot
        ax1.set_title('Average values')                                    # Title first plot
        ax1.get_yaxis().get_major_formatter().set_useOffset(False)        # Prevent ticks auto-formattin

        ax2 = fig.add_subplot(1,2,2)                                      # Axes second plot
        ax2.set_title('Median values')
```

```
        ax2.get_yaxis().get_major_formatter().set_useOffset(False)

        # in below both plots, x-axis is index from original matrix to preserve original location
        ax1.plot(averages)    # Plot in first axes
        ax2.plot(medians)     # Plot in second axes

        plt.show()
```



Execution Time Comparison between `np.ndarray`, `ccmap.matrix` and `gcmap.matrix`

```
In [6]: cmap = np.asarray( ccmap.matrix[:] )

        print('cmap Type:', type(cmap))
        print('ccmap Type:', type(ccmap.matrix))
        print('gcmap Type:', type(gcmap.matrix))

        print(' ')

        %timeit np.sum(gcmap.matrix, axis = 0)              # Sum along row using numpy.sum
        %timeit np.sum(ccmap.matrix, axis = 0)              # Sum along row using numpy.sum
        %timeit np.sum(cmap, axis = 0)              # Sum along row using numpy.sum

        print(' ')

        %timeit np.sum(gcmap.matrix, axis = 1)              # Sum along column using numpy.sum
        %timeit np.sum(ccmap.matrix, axis = 1)              # Sum along column using numpy.sum
        %timeit np.sum(cmap, axis = 1)              # Sum along column using numpy.sum

        print(' ')

        %timeit np.mean(gcmap.matrix, axis = 1)             # Calculating mean using numpy.mean
        %timeit np.mean(ccmap.matrix, axis = 1)             # Calculating mean using numpy.mean
        %timeit np.mean(cmap, axis = 1)             # Calculating mean using numpy.mean

        print(' ')

        %timeit np.median(gcmap.matrix, axis = 0)            # Calculating median using numpy.median
        %timeit np.median(ccmap.matrix, axis = 0)            # Calculating median using numpy.median
        %timeit np.median(cmap, axis = 0)            # Calculating median using numpy.median

        del ccmap

cmap Type: <class 'numpy.ndarray'>
```

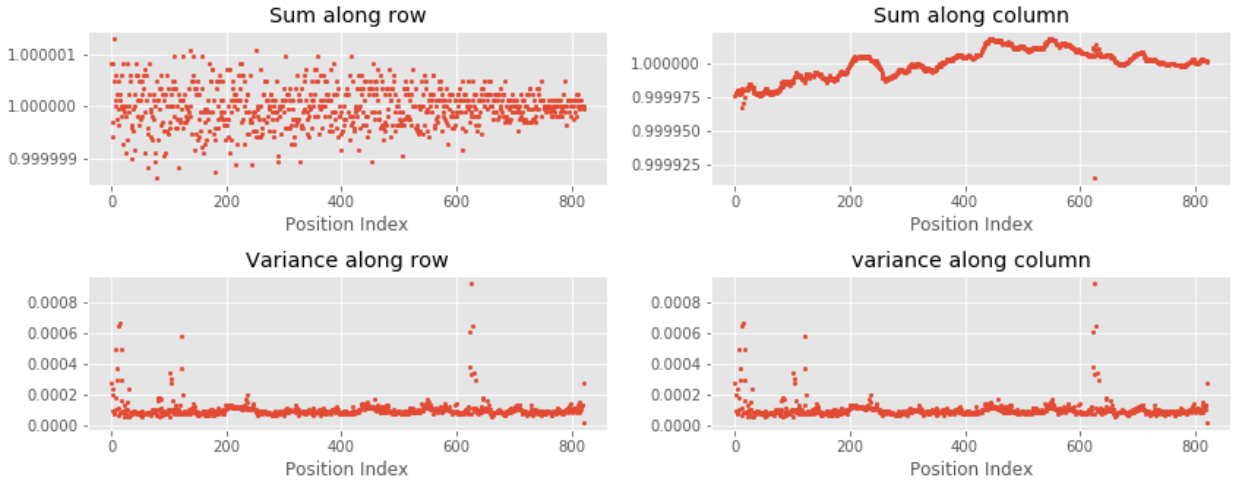
```
ccmap Type: <class 'numpy.core.memmap.memmap'>
gcmap Type: <class 'h5py._hl.dataset.Dataset'>

309 µs ± 25.9 µs per loop (mean ± std. dev. of 7 runs, 1000 loops each)
47.4 µs ± 5.96 µs per loop (mean ± std. dev. of 7 runs, 10000 loops each)
41.3 µs ± 193 ns per loop (mean ± std. dev. of 7 runs, 10000 loops each)

328 µs ± 10.9 µs per loop (mean ± std. dev. of 7 runs, 1000 loops each)
88.1 µs ± 8.83 µs per loop (mean ± std. dev. of 7 runs, 10000 loops each)
78.2 µs ± 3.28 µs per loop (mean ± std. dev. of 7 runs, 10000 loops each)

317 µs ± 380 ns per loop (mean ± std. dev. of 7 runs, 1000 loops each)
90.3 µs ± 19.5 µs per loop (mean ± std. dev. of 7 runs, 10000 loops each)
81.6 µs ± 2.43 µs per loop (mean ± std. dev. of 7 runs, 10000 loops each)

1.81 ms ± 73 µs per loop (mean ± std. dev. of 7 runs, 1000 loops each)
1.56 ms ± 91.8 µs per loop (mean ± std. dev. of 7 runs, 1000 loops each)
1.57 ms ± 166 µs per loop (mean ± std. dev. of 7 runs, 1000 loops each)
```

Whether matrix is balanced after ICE normalization?

In the ICE normalization, if matrix is balanced, sums of rows and coloumns should be equal and variances should be almost equal. In contrast to KR, sums and variances of rows and columns could be extremely large, therefore a direct comparison with KR normalization is impractical. Here, we renormalize IC matrix with sum of rows such that sum of rows and column become equal to one (see line number 9 below).

```
In [7]: ccmapIC = gmlib.gcmap.loadGCMapAsCCMap('normalized/normIC_100kb.gcmap', mapName='chr15')
        ccmapIC.make_readable()
        bData = ~ccmapIC.bNoData                          # Stores whther rows/columns has missing da
        new_matrix = (ccmapIC.matrix[bData,:])[:,bData]   # Getting new matrix after removing rows/co


        # Renormalize ICE matrix with sum of rows so that sum of rows become one
        # It is neccessary to compare with the KR normalization
        new_matrix = new_matrix / np.sum(new_matrix, axis = 1)

        r_sum = np.sum(new_matrix, axis = 0)              # Sum along row using numpy.sum
        r_var = np.var(new_matrix, axis = 0)              # Variance along row using numpy.var

        c_sum = np.sum(new_matrix, axis = 1)              # Sum along column using numpy.sum
        c_var = np.var(new_matrix, axis = 1)              # Variance along column using numpy.var

        # Plot the values for visual representations
        fig = plt.figure(figsize=(14,5))                                  # Figure size
        fig.subplots_adjust(hspace=0.6)                                   # Space between plots

        ax1 = fig.add_subplot(2,2,1)                                      # Axes first plot
        ax1.set_title('Sum along row')                                    # Title first plot
        ax1.set_xlabel('Position Index')                                  # X-label

        ax2 = fig.add_subplot(2,2,2)                                      # Axes second plot
        ax2.set_title('Sum along column')
        ax2.set_xlabel('Position Index')

        ax3 = fig.add_subplot(2,2,3)                                      # Axes third plot
        ax3.set_title('Variance along row')
```

```
ax3.set_xlabel('Position Index')

ax4 = fig.add_subplot(2,2,4)                                    # Axes fourth plot
ax4.set_title('variance along column')
ax4.set_xlabel('Position Index')

ax1.plot(r_sum, marker='o', lw=0, ms=2)                         # Plot in first axes
ax2.plot(c_sum, marker='o', lw=0, ms=2)                         # Plot in second axes
ax3.plot(r_var, marker='o', lw=0, ms=2)                         # Plot in third axes
ax4.plot(c_var, marker='o', lw=0, ms=2)                         # Plot in fourth axesL

ax1.get_yaxis().get_major_formatter().set_useOffset(False)      # Prevent ticks auto-formattin
ax2.get_yaxis().get_major_formatter().set_useOffset(False)
ax3.get_yaxis().get_major_formatter().set_useOffset(False)
ax4.get_yaxis().get_major_formatter().set_useOffset(False)

plt.show()

del ccmapIC                 # Remove temporary files
```



Result

As can be seen in the above plots, sums of rows and columns are equal to one. Additionally, variances of rows and columns are also almost equal.

Comparison between original KR algorithm and gcMapExplorer implementation

Here, we compare the results from original KR matrix balancing algorithm with the gcMapExplorer implmentation.

Note: Since orignal argorithm is implemented in **MATLAB**, here we used **Octave** for calculation. Also, **oct2py** python module should be installed on the system.

Note:

- Below we use already normalized maps that were calculated during *previous tutorial*.

Importing required modules:

- gcMapExplorer.lib
- numpy for statistics

- matplotlib for plotting
- oct2py acts as bridge between Python and Octave, to run original algorithm

In [1]:
```
import gcMapExplorer.lib as gmlib
import numpy as np
from oct2py import octave

import matplotlib as mpl
import matplotlib.pyplot as plt

# To show inline plots
%matplotlib inline
plt.style.use('ggplot')              # Theme for plotting
```

Here we write **original code** in `norm_KR.m` file in current directory, which will be used below for calculation.

In [2]:
```
def write_orig_matlab_code():

    code = """function [x,res] = norm_KR(A,tol,x0,delta,Delta,fl)
  % BNEWT A balancing algorithm for symmetric matrices
  %
  % X = norm_KR(A) attempts to find a vector X such that
  % diag(X)*A*diag(X) is close to doubly stochastic. A must
  % be symmetric and nonnegative.
  %
  % X0: initial guess. TOL: error tolerance.
  % delta/Delta: how close/far balancing vectors can get
  % to/from the edge of the positive cone.
  % We use a relative measure on the size of elements.
  % FL: intermediate convergence statistics on/off.
  % RES: residual error, measured by norm(diag(x)*A*x - e).

  % Initialise
  n = size(A,1); e = ones(n,1); res=[];
  if nargin < 6, fl = 0; end
  if nargin < 5, Delta = 3; end
  if nargin < 4, delta = 0.1; end
  if nargin < 3, x0 = e; end
  if nargin < 2, tol = 1e-4; end

  % Inner stopping criterion parameters.
  g=0.9; etamax = 0.1;
  eta = etamax; stop_tol = tol*.5;
  x = x0; rt = tol^2; v = x.*(A*x); rk = 1 - v;
  rho_km1 = rk'*rk; rout = rho_km1; rold = rout;
  MVP = 0; % We'll count matrix vector products.
  i = 0; % Outer iteration count.
  if fl == 1, fprintf('it in. it res'), end

  while rout > rt % Outer iteration
    i = i + 1; k = 0; y = e;
    innertol = max([eta^2*rout,rt]);

    while rho_km1 > innertol %Inner iteration by CG
      k = k + 1;
      if k == 1
        Z = rk./v; p=Z; rho_km1 = rk'*Z;
```

```
      else
        beta=rho_km1/rho_km2;
        p=Z + beta*p;
      end
      % Update search direction efficiently.
      w = x.*(A*(x.*p)) + v.*p;
      alpha = rho_km1/(p'*w);
      ap = alpha*p;

      % Test distance to boundary of cone.
      ynew = y + ap;
      if min(ynew) <= delta

        if delta == 0
          break
        end

        ind = find(ap < 0);
        gamma = min((delta - y(ind))./ap(ind));
        y = y + gamma*ap;
        break
      end

      if max(ynew) >= Delta
        ind = find(ynew > Delta);
        gamma = min((Delta-y(ind))./ap(ind));
        y = y + gamma*ap;
        break
      end

      y = ynew;
      rk = rk - alpha*w; rho_km2 = rho_km1;
      Z = rk./v; rho_km1 = rk'*Z;

    end

    x = x.*y; v = x.*(A*x);
    rk = 1 - v; rho_km1 = rk'*rk; rout = rho_km1;
    MVP = MVP + k + 1;

    % Update inner iteration stopping criterion.
    rat = rout/rold; rold = rout; res_norm = sqrt(rout);
    eta_o = eta; eta = g*rat;
    if g*eta_o^2 > 0.1
      eta = max([eta,g*eta_o^2]);
    end

    eta = max([min([eta,etamax]),stop_tol/res_norm]);
    if fl == 1
      fprintf('%3d %6d %.3e', i,k, r_norm);
      res=[res; r_norm];
    end

  end

  fprintf('Matrix-vector products = %6d', MVP)
"""

    fout = open('norm_KR.m', 'w')
```

```
            fout.write(code)
            fout.close()
```

Wrapper function to normalize map using original algorithm

Follwing function perform several steps:

- Extract a new matrix from input matrix by removing any rows and columns with missing data
- Normalize it using **Octave**
- Construct normalized matrix from normalization vector
- Expand normalized matrix by re-inserting rows and column with missing data

```
In [3]: def get_norm_KR_matlab(ccmap):
            # Discard rows and columns with missing data
            ccmap.make_readable()
            bNonZeros = gmlib.ccmapHelpers.get_nonzeros_index(ccmap.matrix)
            A = (ccmap.matrix[bNonZeros,:])[:,bNonZeros]

            # Calculate normalization vector using original MATLAB code
            vector = octave.norm_KR(A[:])

            # Construct KR normalized matrix using normalization vector
            outA = vector.T * (A * vector)

            # Initialize ouput ccmap object
            normCCMap = ccmap.copy(fill=0.0)
            normCCMap.make_editable()
            normCCMap.bNoData = ~bNonZeros

            # Store normalized values to output ccmap
            dsm_i = 0
            ox = normCCMap.matrix.shape[0]
            idx_fill = np.nonzero( ~normCCMap.bNoData )
            for i in range(ox):
                if not normCCMap.bNoData[i]:
                    normCCMap.matrix[i, idx_fill] = outA[dsm_i]
                    normCCMap.matrix[idx_fill, i] = outA[dsm_i]
                    dsm_i += 1

            # Get the maximum and minimum value except zero
            ma = np.ma.masked_equal(outA, 0.0, copy=False)
            normCCMap.minvalue = ma.min()
            normCCMap.maxvalue = ma.max()

            normCCMap.make_unreadable()
            ccmap.make_unreadable()

            return normCCMap
```

Normalization using original Matlab code

Here, we read raw maps from a gcmap file, normalize it using above function and save the normalized maps to output gcmap file.

```
In [4]: gcMapInputFile = 'cmaps/CooMatrix/rawObserved_100kb.gcmap'
        gcMapOutFile = 'norm_comparison/normKR_matlab.gcmap'
        write_orig_matlab_code()

        # Get list of maps in ascending order
        gcmap = gmlib.gcmap.GCMAP(gcMapInputFile)
        gcmap.loadSmallestMap()
        mapList = gcmap.mapNameList.copy()
        del gcmap

        for mapName in mapList:
            print('Calculating for {0}...'.format(mapName))
            ccMap = gmlib.gcmap.loadGCMapAsCCMap(gcMapInputFile, mapName=mapName)
            normKR_ccmap = get_norm_KR_matlab(ccMap)

            if normKR_ccmap is not None:
                gmlib.gcmap.addCCMap2GCMap(normKR_ccmap, gcMapOutFile,generateCoarse=True, coarsingMe

            del ccMap
            del normKR_ccmap
```

```
Calculating for chr21...
Matrix-vector products =     36

INFO:addCCMap2GCMap: Opened file [norm_comparison/normKR_matlab.gcmap] for reading writing..
INFO:addCCMap2GCMap: Data for chr21 is already present in [norm_comparison/normKR_matlab.gcmap], repl
INFO:addCCMap2GCMap: Adding data to [norm_comparison/normKR_matlab.gcmap] for [chr21] ...
INFO:addCCMap2GCMap:     ...Finished adding data for [chr21] ...
INFO:addCCMap2GCMap: Generating downsampled maps for [chr21] ...
INFO:addCCMap2GCMap:     ... Finished downsampling for [chr21] ...
INFO:addCCMap2GCMap: Closed file [norm_comparison/normKR_matlab.gcmap]...

Calculating for chr22...
Matrix-vector products =     31

INFO:addCCMap2GCMap: Opened file [norm_comparison/normKR_matlab.gcmap] for reading writing..
INFO:addCCMap2GCMap: Data for chr22 is already present in [norm_comparison/normKR_matlab.gcmap], repl
INFO:addCCMap2GCMap: Adding data to [norm_comparison/normKR_matlab.gcmap] for [chr22] ...
INFO:addCCMap2GCMap:     ...Finished adding data for [chr22] ...
INFO:addCCMap2GCMap: Generating downsampled maps for [chr22] ...
INFO:addCCMap2GCMap:     ... Finished downsampling for [chr22] ...
INFO:addCCMap2GCMap: Closed file [norm_comparison/normKR_matlab.gcmap]...

Calculating for chr20...
Matrix-vector products =     29

INFO:addCCMap2GCMap: Opened file [norm_comparison/normKR_matlab.gcmap] for reading writing..
INFO:addCCMap2GCMap: Data for chr20 is already present in [norm_comparison/normKR_matlab.gcmap], repl
INFO:addCCMap2GCMap: Adding data to [norm_comparison/normKR_matlab.gcmap] for [chr20] ...
INFO:addCCMap2GCMap:     ...Finished adding data for [chr20] ...
INFO:addCCMap2GCMap: Generating downsampled maps for [chr20] ...
INFO:addCCMap2GCMap:     ... Finished downsampling for [chr20] ...
INFO:addCCMap2GCMap: Closed file [norm_comparison/normKR_matlab.gcmap]...

Calculating for chr15...
Matrix-vector products =     37

INFO:addCCMap2GCMap: Opened file [norm_comparison/normKR_matlab.gcmap] for reading writing..
INFO:addCCMap2GCMap: Data for chr15 is already present in [norm_comparison/normKR_matlab.gcmap], repl
INFO:addCCMap2GCMap: Adding data to [norm_comparison/normKR_matlab.gcmap] for [chr15] ...
INFO:addCCMap2GCMap:     ...Finished adding data for [chr15] ...
INFO:addCCMap2GCMap: Generating downsampled maps for [chr15] ...
```

```
INFO:addCCMap2GCMap:     ... Finished downsampling for [chr15] ...
INFO:addCCMap2GCMap: Closed file [norm_comparison/normKR_matlab.gcmap]...

Calculating for chr5...
Matrix-vector products =    102

INFO:addCCMap2GCMap: Opened file [norm_comparison/normKR_matlab.gcmap] for reading writing..
INFO:addCCMap2GCMap: Data for chr5 is already present in [norm_comparison/normKR_matlab.gcmap], repla
INFO:addCCMap2GCMap: Adding data to [norm_comparison/normKR_matlab.gcmap] for [chr5] ...
INFO:addCCMap2GCMap:     ...Finished adding data for [chr5] ...
INFO:addCCMap2GCMap: Generating downsampled maps for [chr5] ...
INFO:addCCMap2GCMap:     ... Finished downsampling for [chr5] ...
INFO:addCCMap2GCMap: Closed file [norm_comparison/normKR_matlab.gcmap]...

Calculating for chr1...
Matrix-vector products =    105

INFO:addCCMap2GCMap: Opened file [norm_comparison/normKR_matlab.gcmap] for reading writing..
INFO:addCCMap2GCMap: Data for chr1 is already present in [norm_comparison/normKR_matlab.gcmap], repla
INFO:addCCMap2GCMap: Adding data to [norm_comparison/normKR_matlab.gcmap] for [chr1] ...
INFO:addCCMap2GCMap:     ...Finished adding data for [chr1] ...
INFO:addCCMap2GCMap: Generating downsampled maps for [chr1] ...
INFO:addCCMap2GCMap:     ... Finished downsampling for [chr1] ...
INFO:addCCMap2GCMap: Closed file [norm_comparison/normKR_matlab.gcmap]...
```

Correlation Calculation

Below we calculate correlation between above normalized maps (original algorithm) and previously normalized maps (gcMapExplorer implementation).

```
In [5]: gcMapOne = 'norm_comparison/normKR_matlab.gcmap'
        gcMapTwo = 'normalized/normKR_100kb.gcmap'

        mapList, corrs, pvalues = gmlib.cmstats.correlateGCMaps(gcMapOne, gcMapTwo, corrType='pearson

        print('Pearson Correlation gcMapExplorer-KR vs original-KR')
        print('=================================================')
        print('\nChromosome     Corr      P-values')
        print('-------------------------------------------------------')
        for i in range(len(mapList)):
            print('{0:5} {1:12.5} {2:12.5}'.format( mapList[i], corrs[i], pvalues[i]))
        print('-------------------------------------------------------')
```

```
INFO:correlateGCMaps: Performing calculation for chr1 ...
INFO:correlateGCMaps:     Finished calculation for chr1 ...

INFO:correlateGCMaps: Performing calculation for chr5 ...
INFO:correlateGCMaps:     Finished calculation for chr5 ...

INFO:correlateGCMaps: Performing calculation for chr15 ...
INFO:correlateGCMaps:     Finished calculation for chr15 ...

INFO:correlateGCMaps: Performing calculation for chr20 ...
INFO:correlateGCMaps:     Finished calculation for chr20 ...

INFO:correlateGCMaps: Performing calculation for chr21 ...
INFO:correlateGCMaps:     Finished calculation for chr21 ...

INFO:correlateGCMaps: Performing calculation for chr22 ...
```

```
INFO:correlateGCMaps:     Finished calculation for chr22 ...

Pearson Correlation gcMapExplorer-KR vs original-KR
====================================================

Chromosome     Corr      P-values
----------------------------------------------------------
chr1           1.0          0.0
chr5           1.0          0.0
chr15          1.0          0.0
chr20          1.0          0.0
chr21          1.0          0.0
chr22          1.0          0.0
----------------------------------------------------------
```

Results

As can be seen above output table, both normalized maps correlate perfectly, therefore, gcMapExplorer implemented normalization yielded almost same result as of the original algorithm.

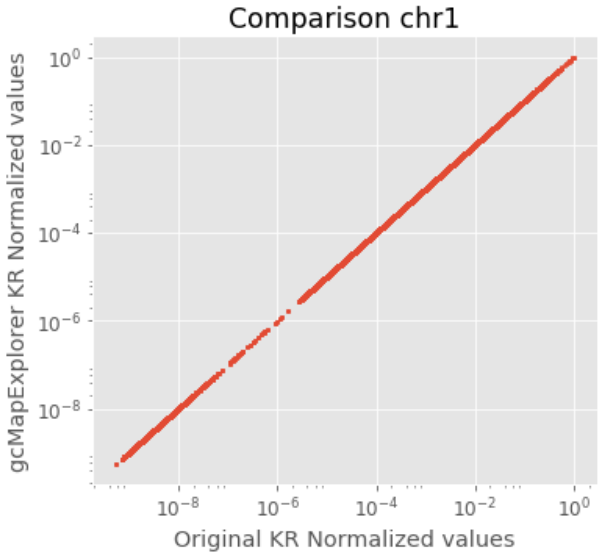
To plot comparison between values from original and gcMapExplorer method

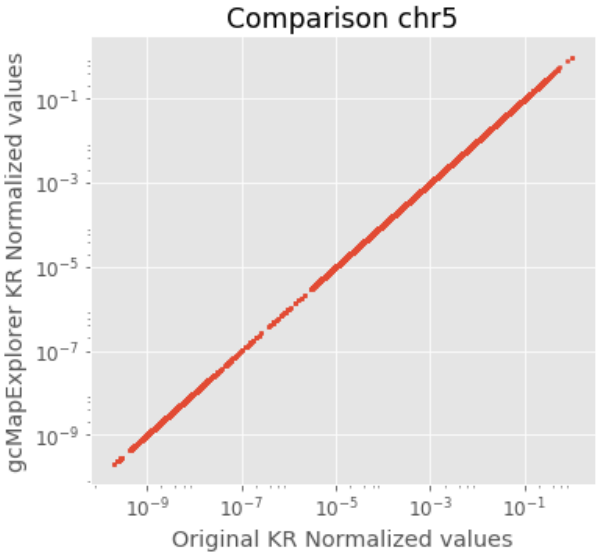
Below function is used to plot values from a single map.

```
In [6]: def plot_comparison_for_chromosome(mapName, fig, idx):
            # Read normalized map from original algorithm
            matlabNormCCmap = gmlib.gcmap.loadGCMapAsCCMap('norm_comparison/normKR_matlab.gcmap', map

            # Read normalized map from gcMapExplorer implemented code
            gcMapExpNormCCmap = gmlib.gcmap.loadGCMapAsCCMap('normalized/normKR_100kb.gcmap', mapName

            # Make ccmap matrix file available for reading
            matlabNormCCmap.make_readable()
            gcMapExpNormCCmap.make_readable()

            # Mask any bins that have zero value, since both map are calculated from same raw map,
            # same bins will have missing data
            matlabNormCCmapMasked = np.ma.masked_equal(matlabNormCCmap.matrix, 0.0, copy=False)
            gcMapExpNormCCmapMasked = np.ma.masked_equal(gcMapExpNormCCmap.matrix, 0.0, copy=False)

            ax = fig.add_subplot(3,2,idx)                                    # Axes plot
            ax.set_yscale("log", basey=10, nonposy='clip')                   # Set Y-axis to log scale
            ax.set_xscale("log", basey=10, nonposy='clip')                   # Set X-axis to log scale
            ax.set_title('Comparison {0}'.format(mapName))                   # Title plot

            # Plot for Pearson correlation
            # Plot only those bins that do not have missing data (> zero)
            ax.scatter(matlabNormCCmapMasked.compressed(), gcMapExpNormCCmapMasked.compressed(), mark

            ax.set_xlabel('Original KR Normalized values')
            ax.set_ylabel('gcMapExplorer KR Normalized values')

            del matlabNormCCmap
            del gcMapExpNormCCmap
```

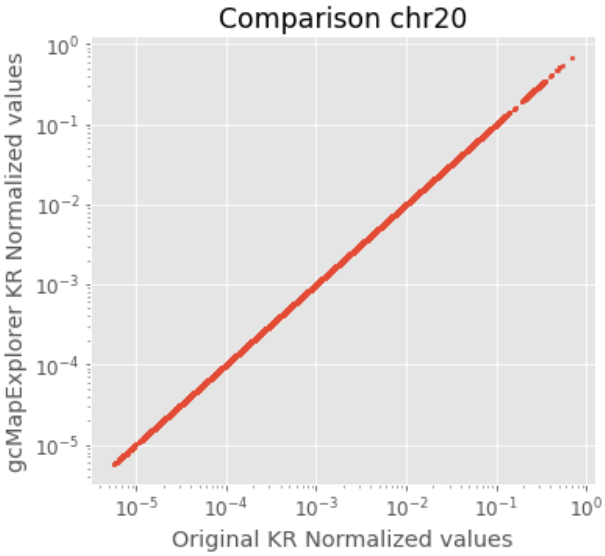
Now, we plot normalized values for some chromosome maps using above written function, which also shows that the obtaind values are similar from both implementations.

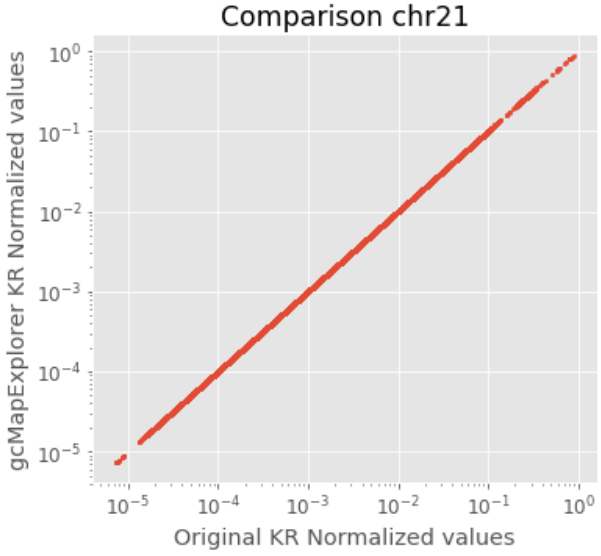
```
In [7]: fig = plt.figure(figsize=(13,19))                                    # Figure size
        fig.subplots_adjust(hspace=0.3, wspace=0.25)                          # Space between sub-plots
        mpl.rcParams.update({'font.size': 12})                               # Font-size

        # Plot for chromosomes
        plot_comparison_for_chromosome('chr1', fig, 1)                       # For chr1
        plot_comparison_for_chromosome('chr5', fig, 2)                       # For chr5
        plot_comparison_for_chromosome('chr15', fig, 3)                      # For chr15
        plot_comparison_for_chromosome('chr20', fig, 4)                      # For chr20
        plot_comparison_for_chromosome('chr21', fig, 5)                      # For chr21
        plot_comparison_for_chromosome('chr22', fig, 6)                      # For chr22

        plt.savefig('compare_KR.png', dpi=300)
        plt.show()
```

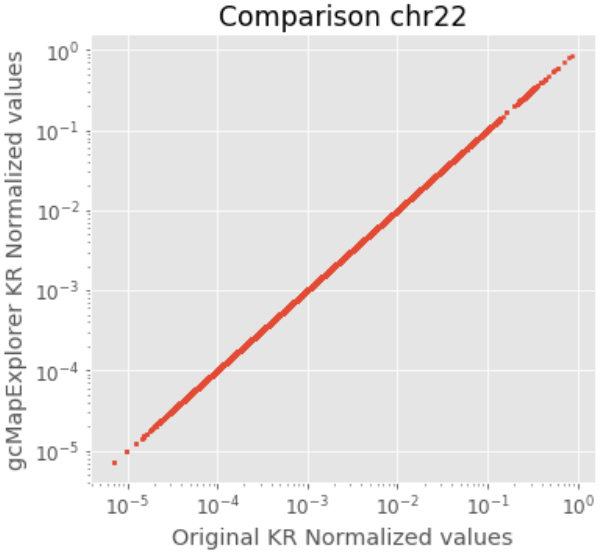

Comparison chr1
gcMapExplorer KR Normalized values
Original KR Normalized values


Comparison chr5
gcMapExplorer KR Normalized values
Original KR Normalized values


Comparison chr15
gcMapExplorer KR Normalized values
Original KR Normalized values


Comparison chr20
gcMapExplorer KR Normalized values
Original KR Normalized values


Comparison chr21
gcMapExplorer KR Normalized values
Original KR Normalized values


Comparison chr22
gcMapExplorer KR Normalized values
Original KR Normalized values

Comparison between original IC algorithm and gcMapExplorer implementation

Here, we compare the results from original IC matrix balancing algorithm with the gcMapExplorer implmentation.

Note: To run the below calculation, iced should be installed.

Note:

- Below we use already normalized maps that were calculated during *previous tutorial*.

Importing required modules:

- gcMapExplorer.lib
- numpy for statistics
- matplotlib for plotting
- iced to run original ICE algorithm

```
In [1]: import gcMapExplorer.lib as gmlib
        import numpy as np
        import iced

        import matplotlib as mpl
        import matplotlib.pyplot as plt

        # To show inline plots
        %matplotlib inline
        plt.style.use('ggplot')              # Theme for plotting
```

Wrapper function to normalize map using original algorithm

Follwing function perform several steps:

- Extract a new matrix from input matrix by removing any rows and columns with missing data
- Normalize it using iced module
- Construct normalized matrix from normalization vector
- Expand normalized matrix by re-inserting rows and column with missing data

```
In [2]: def normIC_from_iced(ccmap):
            # Discard rows and columns with missing data
            ccmap.make_readable()
            bNonZeros = gmlib.ccmapHelpers.get_nonzeros_index(ccmap.matrix)
            A = (ccmap.matrix[bNonZeros,:])[:,bNonZeros]

            # Calculate normalization using iced module
            outA = iced.normalization.ICE_normalization(A[:], eps=1e-4, max_iter=3000)

            # Initialize ouput ccmap object
            normCCMap = ccmap.copy(fill=0.0)
            normCCMap.make_editable()
            normCCMap.bNoData = ~bNonZeros

            # Store normalized values to output ccmap
            dsm_i = 0
            ox = normCCMap.matrix.shape[0]
```

```
            idx_fill = np.nonzero( ~normCCMap.bNoData )
            for i in range(ox):
                if not normCCMap.bNoData[i]:
                    normCCMap.matrix[i, idx_fill] = outA[dsm_i]
                    normCCMap.matrix[idx_fill, i] = outA[dsm_i]
                    dsm_i += 1

            # Get the maximum and minimum value except zero
            ma = np.ma.masked_equal(outA, 0.0, copy=False)
            normCCMap.minvalue = ma.min()
            normCCMap.maxvalue = ma.max()

            normCCMap.make_unreadable()
            ccmap.make_unreadable()

            return normCCMap
```

Normalization using original algorithm from iced module

Here, we read raw maps from a gcmap file, normalize it using above function and save the normalized maps to output gcmap file.

```
In [3]: gcMapInputFile = 'cmaps/CooMatrix/rawObserved_100kb.gcmap'
        gcMapOutFile = 'norm_comparison/normIC_iced.gcmap'

        # Get list of maps in ascending order
        gcmap = gmlib.gcmap.GCMAP(gcMapInputFile)
        gcmap.loadSmallestMap()
        mapList = gcmap.mapNameList.copy()
        del gcmap

        for mapName in mapList:
            print('Calculating for {0}...'.format(mapName))
            ccMap = gmlib.gcmap.loadGCMapAsCCMap(gcMapInputFile, mapName=mapName)
            normKR_ccmap = normIC_from_iced(ccMap)

            if normKR_ccmap is not None:
                gmlib.gcmap.addCCMap2GCMap(normKR_ccmap, gcMapOutFile,generateCoarse=True, coarsingMe

            del ccMap
            del normKR_ccmap
```

```
INFO:addCCMap2GCMap: Opened file [norm_comparison/normIC_iced.gcmap] for reading writing..
INFO:addCCMap2GCMap: Data for chr21 is already present in [norm_comparison/normIC_iced.gcmap], replac
INFO:addCCMap2GCMap: Adding data to [norm_comparison/normIC_iced.gcmap] for [chr21] ...
INFO:addCCMap2GCMap:     ...Finished adding data for [chr21] ...
INFO:addCCMap2GCMap: Generating downsampled maps for [chr21] ...
INFO:addCCMap2GCMap:     ... Finished downsampling for [chr21] ...
INFO:addCCMap2GCMap: Closed file [norm_comparison/normIC_iced.gcmap]...

Calculating for chr21...
Calculating for chr22...

INFO:addCCMap2GCMap: Opened file [norm_comparison/normIC_iced.gcmap] for reading writing..
INFO:addCCMap2GCMap: Data for chr22 is already present in [norm_comparison/normIC_iced.gcmap], replac
INFO:addCCMap2GCMap: Adding data to [norm_comparison/normIC_iced.gcmap] for [chr22] ...
INFO:addCCMap2GCMap:     ...Finished adding data for [chr22] ...
```

```
INFO:addCCMap2GCMap: Generating downsampled maps for [chr22] ...
INFO:addCCMap2GCMap:     ... Finished downsampling for [chr22] ...
INFO:addCCMap2GCMap: Closed file [norm_comparison/normIC_iced.gcmap]...

Calculating for chr20...

INFO:addCCMap2GCMap: Opened file [norm_comparison/normIC_iced.gcmap] for reading writing..
INFO:addCCMap2GCMap: Data for chr20 is already present in [norm_comparison/normIC_iced.gcmap], replac
INFO:addCCMap2GCMap: Adding data to [norm_comparison/normIC_iced.gcmap] for [chr20] ...
INFO:addCCMap2GCMap:     ...Finished adding data for [chr20] ...
INFO:addCCMap2GCMap: Generating downsampled maps for [chr20] ...
INFO:addCCMap2GCMap:     ... Finished downsampling for [chr20] ...
INFO:addCCMap2GCMap: Closed file [norm_comparison/normIC_iced.gcmap]...

Calculating for chr15...

INFO:addCCMap2GCMap: Opened file [norm_comparison/normIC_iced.gcmap] for reading writing..
INFO:addCCMap2GCMap: Data for chr15 is already present in [norm_comparison/normIC_iced.gcmap], replac
INFO:addCCMap2GCMap: Adding data to [norm_comparison/normIC_iced.gcmap] for [chr15] ...
INFO:addCCMap2GCMap:     ...Finished adding data for [chr15] ...
INFO:addCCMap2GCMap: Generating downsampled maps for [chr15] ...
INFO:addCCMap2GCMap:     ... Finished downsampling for [chr15] ...
INFO:addCCMap2GCMap: Closed file [norm_comparison/normIC_iced.gcmap]...

Calculating for chr5...

INFO:addCCMap2GCMap: Opened file [norm_comparison/normIC_iced.gcmap] for reading writing..
INFO:addCCMap2GCMap: Data for chr5 is already present in [norm_comparison/normIC_iced.gcmap], replaci
INFO:addCCMap2GCMap: Adding data to [norm_comparison/normIC_iced.gcmap] for [chr5] ...
INFO:addCCMap2GCMap:     ...Finished adding data for [chr5] ...
INFO:addCCMap2GCMap: Generating downsampled maps for [chr5] ...
INFO:addCCMap2GCMap:     ... Finished downsampling for [chr5] ...
INFO:addCCMap2GCMap: Closed file [norm_comparison/normIC_iced.gcmap]...

Calculating for chr1...

INFO:addCCMap2GCMap: Opened file [norm_comparison/normIC_iced.gcmap] for reading writing..
INFO:addCCMap2GCMap: Data for chr1 is already present in [norm_comparison/normIC_iced.gcmap], replaci
INFO:addCCMap2GCMap: Adding data to [norm_comparison/normIC_iced.gcmap] for [chr1] ...
INFO:addCCMap2GCMap:     ...Finished adding data for [chr1] ...
INFO:addCCMap2GCMap: Generating downsampled maps for [chr1] ...
INFO:addCCMap2GCMap:     ... Finished downsampling for [chr1] ...
INFO:addCCMap2GCMap: Closed file [norm_comparison/normIC_iced.gcmap]...
```

Correlation Calculation

Below we calculate correlation between above normalized maps (original algorithm) and previously normalized maps (gcMapExplorer implementation).

```
In [4]: gcMapOne = 'norm_comparison/normIC_iced.gcmap'    # Normalization using original algorithm
        gcMapTwo = 'normalized/normIC_100kb.gcmap'        # Normalization using gcMapExplorer module

        mapList, corrs, pvalues = gmlib.cmstats.correlateGCMaps(gcMapOne, gcMapTwo, corrType='pearson

        print('Pearson Correlation gcMapExplorer-IC vs original-IC')
        print('=======================================================')
        print('\nChromosome     Corr      P-values')
        print('-------------------------------------------------------')
        for i in range(len(mapList)):
            print('{0:5} {1:12.5} {2:12.5}'.format( mapList[i], corrs[i], pvalues[i]))
        print('-------------------------------------------------------')
```

```
INFO:correlateGCMaps: Performing calculation for chr1 ...
INFO:correlateGCMaps:     Finished calculation for chr1 ...

INFO:correlateGCMaps: Performing calculation for chr5 ...
INFO:correlateGCMaps:     Finished calculation for chr5 ...

INFO:correlateGCMaps: Performing calculation for chr15 ...
INFO:correlateGCMaps:     Finished calculation for chr15 ...

INFO:correlateGCMaps: Performing calculation for chr20 ...
INFO:correlateGCMaps:     Finished calculation for chr20 ...

INFO:correlateGCMaps: Performing calculation for chr21 ...
INFO:correlateGCMaps:     Finished calculation for chr21 ...

INFO:correlateGCMaps: Performing calculation for chr22 ...
INFO:correlateGCMaps:     Finished calculation for chr22 ...


Pearson Correlation gcMapExplorer-IC vs original-IC
===================================================

Chromosome      Corr      P-values
-------------------------------------------------------
chr1            1.0          0.0
chr5            1.0          0.0
chr15           1.0          0.0
chr20           1.0          0.0
chr21           1.0          0.0
chr22           1.0          0.0
-------------------------------------------------------
```

Results

As can be seen above output table, both normalized maps correlate perfectly, therefore, gcMapExplorer implemented normalization yielded almost same result as of the original algorithm.

To plot comparison between values from original and gcMapExplorer method

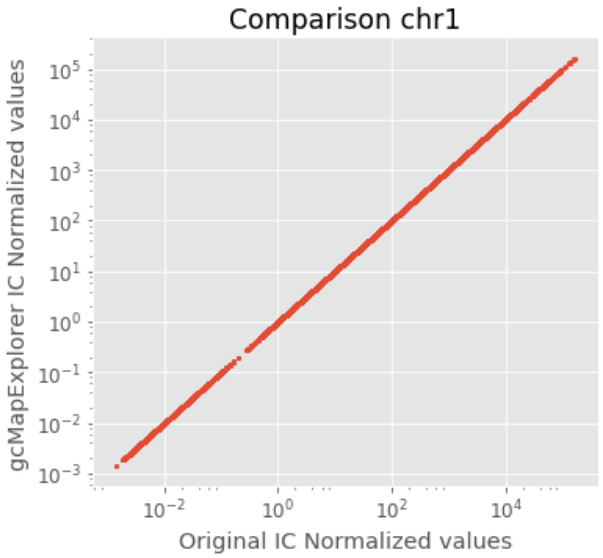
Below function is used to plot values from a single map.

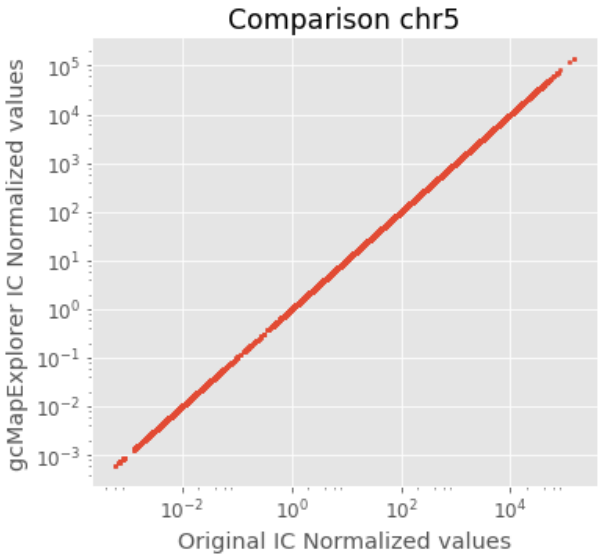
```
In [5]: def plot_comparison_for_chromosome(mapName, fig, idx):
            # Read normalized map from original algorithm
            origICNormCCmap = gmlib.gcmap.loadGCMapAsCCMap('norm_comparison/normIC_iced.gcmap', mapNa

            # Read normalized map from gcMapExplorer implemented code
            gcMapExpNormCCmap = gmlib.gcmap.loadGCMapAsCCMap('normalized/normIC_100kb.gcmap', mapName

            # Make ccmap matrix file available for reading
            origICNormCCmap.make_readable()
            gcMapExpNormCCmap.make_readable()

            # Mask any bins that have zero value, since both map are calculated from same raw map,
            # same bins will have missing data
            origICNormCCmapMasked = np.ma.masked_equal(origICNormCCmap.matrix, 0.0, copy=False)
            gcMapExpNormCCmapMasked = np.ma.masked_equal(gcMapExpNormCCmap.matrix, 0.0, copy=False)
```

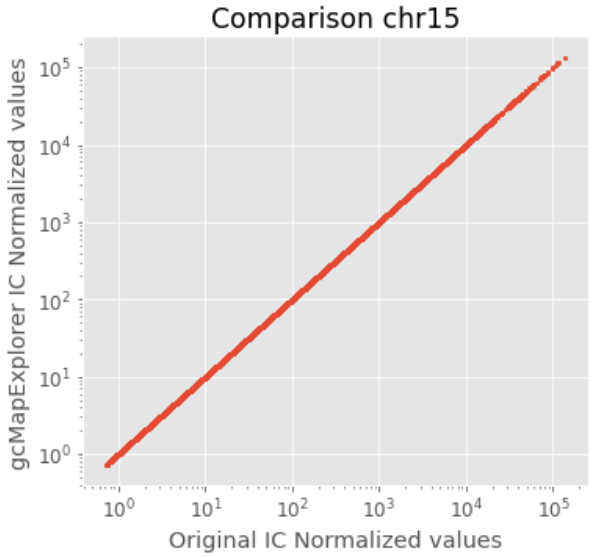
```
        ax = fig.add_subplot(3,2,idx)                                    # Axes plot
        ax.set_yscale("log", basey=10, nonposy='clip')                   # Set Y-axis to log scale
        ax.set_xscale("log", basey=10, nonposy='clip')                   # Set X-axis to log scale
        ax.set_title('Comparison {0}'.format(mapName))                   # Title plot

        # Plot for Pearson correlation
        # Plot only those bins that do not have missing data (> zero)
        ax.scatter(origICNormCCmapMasked.compressed(), gcMapExpNormCCmapMasked.compressed(), mar

        ax.set_xlabel('Original IC Normalized values')
        ax.set_ylabel('gcMapExplorer IC Normalized values')

        del origICNormCCmap
        del gcMapExpNormCCmap
```

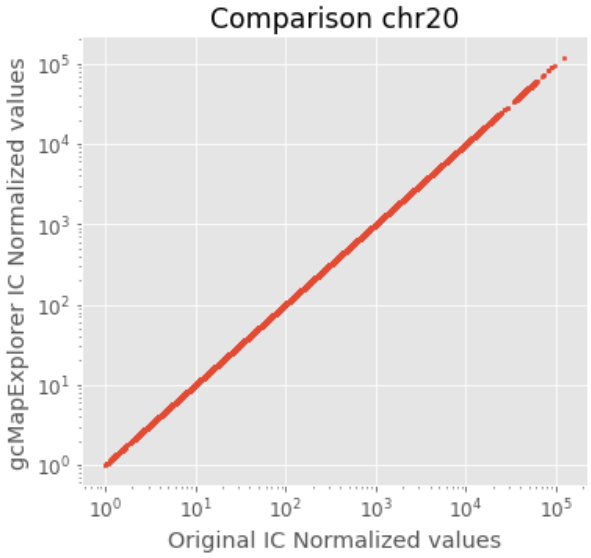
Now, we plot normalized values for some chromosome maps using above written function, which also shows that the obtaind values are similar from both implementations.

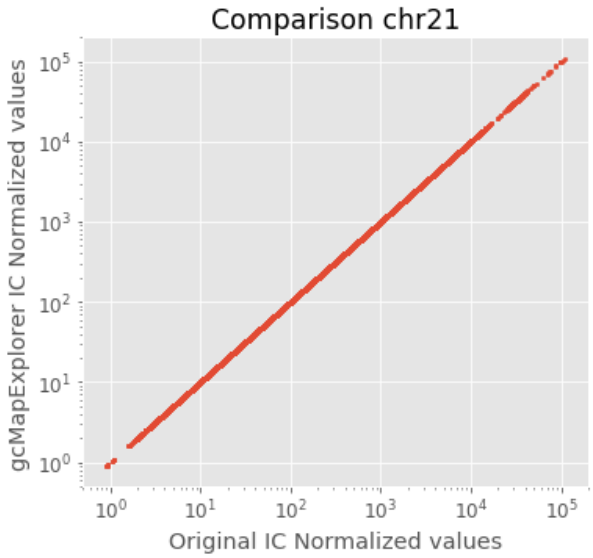
```
In [6]: fig = plt.figure(figsize=(13,19))                                # Figure size
        fig.subplots_adjust(hspace=0.3, wspace=0.25)                     # Space between sub-plots
        mpl.rcParams.update({'font.size': 12})                           # Font-size

        # Plot for chromosomes
        plot_comparison_for_chromosome('chr1', fig, 1)                   # For chr1
        plot_comparison_for_chromosome('chr5', fig, 2)                   # For chr5
        plot_comparison_for_chromosome('chr15', fig, 3)                  # For chr15
        plot_comparison_for_chromosome('chr20', fig, 4)                  # For chr20
        plot_comparison_for_chromosome('chr21', fig, 5)                  # For chr21
        plot_comparison_for_chromosome('chr22', fig, 6)                  # For chr22

        # plt.savefig('compare_IC.png', dpi=300)
        plt.show()
```

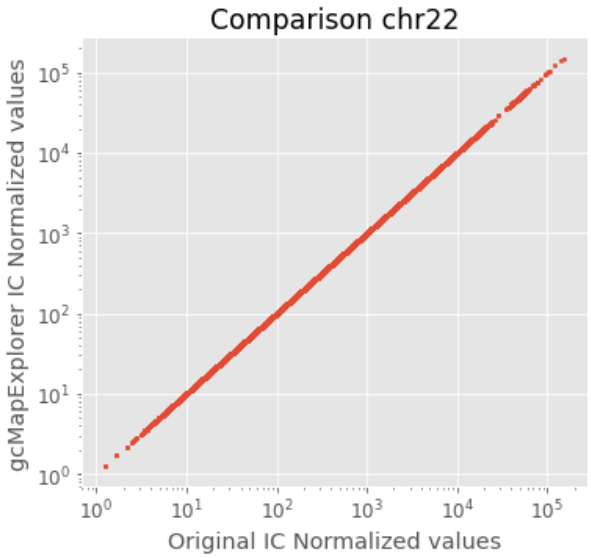

Comparison chr1
gcMapExplorer IC Normalized values
Original IC Normalized values


Comparison chr5
gcMapExplorer IC Normalized values
Original IC Normalized values


Comparison chr15
gcMapExplorer IC Normalized values
Original IC Normalized values


Comparison chr20
gcMapExplorer IC Normalized values
Original IC Normalized values


Comparison chr21
gcMapExplorer IC Normalized values
Original IC Normalized values


Comparison chr22
gcMapExplorer IC Normalized values
Original IC Normalized values

3.1.10 Documentation of Python Modules

ccmap module

ccmap.CCMAP.copy([fill])	To create a new copy of CCMAP object
ccmap.CCMAP.get_ticks([binsize])	To get xticks and yticks for the matrix
ccmap.CCMAP.make_readable()	Enable reading the numpy array binary file.
ccmap.CCMAP.make_unreadable()	Disable reading the numpy array binary file from local file system
ccmap.CCMAP.make_writable()	Create new numpy array binary file on local file system and enable reading/writing to this file
ccmap.CCMAP.make_editable()	Enable editing numpy array binary file
ccmap.jsonify(ccMapObj)	Changes data type of attributes in CCMAP object for .
ccmap.dejsonify(ccMapObj[, json_dict])	Change back the data type of attributes in CCMAP object.
ccmap.save_ccmap(ccMapObj, outfile[, ...])	Save CCMAP object on file
ccmap.load_ccmap(infile[, workDir])	Load CCMAP object from an input file
ccmap.export_cmap(ccmap, outfile[, ...])	To export `.ccmap` as text file
ccmap.checkCCMapObjectOrFile(ccMap[, workDir])	Check whether ccmap is a object or file
ccmap.downSampleCCMap(cmap[, level, method, ...])	Downsample or coarsen the contact map
ccmap.getOutputShapeFor2DMapDownsampling()	Helper function to determine output shape of map for downsampling
ccmap.downSample2DMap(inMatrix[, outMatrix, ...])	Downsample or coarsen the matrix

ccmap.CCMAP class

class CCMAP(*dtype='float32'*)

This class contains variables to store Hi-C Data.

The class is instantiated by two methods:

```
>>> ccMapObj = gcMapExplorer.lib.ccmap.CCMAP()
>>> ccMapObj = gcMapExplorer.lib.ccmap.CCMAP(dtype='float32')
```

Parameters **dtype** (*str, Optional*) – Data type for matrix. [Default='float32']

path2matrix

str – Path to numpy array binary file on local file system

yticks

list – Minimum and maximum locations along Y-axis. e.g. `yticks=[0, 400000]`

xticks

list – Minimum and maximum locations along X-axis. e.g. `xticks=[0, 400000]`

binsize

int – Resolution of data. In case of 10kb resolution, binsize is 10000.

title

str – Title of the data

xlabel
str – Title for X-axis

ylabel
str – Title for Y-axis

shape
tuple – Overall shape of matrix

minvalue
float – Minimum value in matrix

maxvalue
float – Maximum value in matrix

matrix
numpy.memmap – A memmap object pointing to matrix.

HiC map data is saved as a numpy array binary file on local file system. This file can be only read after mapping to a numpy memmap object. After mapping, `matrix` can be used as a numpy array. The file name is randomly generated as `npBinary_XXXXXXXXXX.tmp`, where `X` can be a alphanumeric charecter. Please see details in .

When ccmap is saved, this file is renamed with '.npbin' extension.

bNoData
numpy.ndarray – A boolean numpy array of matrix shape

bLog
bool – If values in matrix are in log

state
str – State of CCMAP object

This keyword stores the state of the object. The state ensures when the numpy array binary file should be deleted from the local file system.

Three keywords are used:

- **temporary :** When object is created, it is in temporary state. After executing the script, numpy array binary file is automatically deleted from the local file system.
- **saved:** When a temporary or new object is saved, the numpy array binary file is copied to the destination directory and state is changed to saved. However, after saving, state become temporary. This method ensures that the saved copy is not deleted and only temporary copy is deleted after executing the script.

  When a already saved object is loaded, it is in saved state. The numpy array binary file is read from the original location and remains saved at the original location after executing the script.
- **compressed:** When object is saved and numpy array binary file is simultaneously compressed, it is saved as compressed state. When this object is loaded, the numpy array binary file is decompressed into the working directory. This decompressed file is automatically deleted after execution of script while compressed file remains saved at the original location.

dtype
str – Data type of matrix

copy (*fill=None*)
To create a new copy of CCMAP object

This method can be used to create a new copy of *gcMapExplorer.lib.ccmap.CCMAP*. A new numpy array binary file will be created and all values from old file will be copied.

Parameters **fill** (*float*) – Fill map with the value. If not given, map values will be copied.

get_ticks(*binsize=None*)
To get xticks and yticks for the matrix

Parameters **binsize** (*int*) – Number of base in each bin or pixel or box of contact map.

Returns

- **xticks** (*numpy.array*) – 1D array containing positions along X-axis
- **yticks** (*numpy.array*) – 1D array containing positions along X-axis

make_editable()
Enable editing numpy array binary file

make_readable()
Enable reading the numpy array binary file.

Matrix file is saved on local file system. This file can be only read after mapping to a memmap object. This method maps the numpy memmap object to self.matrix variable. After using this method, *gcMapExplorer.lib.ccmap.CCMAP.matrix* can be used directly as similar to numpy array. Please see details in

make_unreadable()
Disable reading the numpy array binary file from local file system

make_writable()
Create new numpy array binary file on local file system and enable reading/writing to this file

Note: If a matrix file with similar name is already present, old file will be backed up.

ccmap module

jsonify(*ccMapObj*)
Changes data type of attributes in CCMAP object for .

Before saving the CCMAP object, its attributes data types are necessary to change because few data types are not supported by json.

Therefore, it is converted into other data types which are supported by json. These are the following attributes which are changed:

Attributes	Original	modified
bNoData	numpy boolean array	string of 0 and 1
xticks	list of integer	list of string
yticks	list of integer	list of string
minvalue	float	string
maxvalue	float	string
binsize	integer	string
shape	tuple of integer	list of string
dtype	Numpy dtype	string

Warning: If a object is passed through this method, it should be again passed through *gcMapExplorer.lib.ccmap.dejsonify()* for any further use. Otherwise, this object cannot be used in any other methods because of the attributes data type modifications.

Parameters **ccMapObj** (*gcMapExplorer.lib.ccmap.CCMAP*) – A CCMAP object

Returns

Return type None

dejsonify(*ccMapObj*, *json_dict=None*)

Change back the data type of attributes in CCMAP object.

Before loading the CCMAP object, its attributes data types are necessary to change back.

Therefore, it is converted into original data types as shown in a table (see *gcMapExplorer.lib.ccmap.jsonify()*)

Parameters

- **ccMapObj** (*gcMapExplorer.lib.ccmap.CCMAP*) – A CCMAP object
- **json_dict** (*dict, Optional*) – A directory obtained after loading the saved file through json.

save_ccmap(*ccMapObj*, *outfile*, *compress=False*, *logHandler=None*)

Save CCMAP object on file

CCMAP object can be saved as file for easy use. is used to save the object. The binary numpy array file is copied in the destination directory. If `compress=True`, the array file will be compressed in gzip format.

Note:

- Compression significantly reduces the array file size. However, its loading is slow during initiation when file is decompressed.
- After loading, the decompressed binary numpy array file takes additional memory on local file system.

Parameters

- **ccMapObj** (*gcMapExplorer.lib.ccmap.CCMAP*) – A CCMAP object, which has to be saved
- **outfile** (*str*) – Name of output file including path to the directory/folder where file should be saved.
- **compress** (*bool*) – If `True`, numpy array file will be compressed.

Returns

Return type None

load_ccmap(*infile*, *workDir=None*)

Load CCMAP object from an input file

CCMAP object can be created from the input file, which was earlier saved using *gcMapExplorer.lib.ccmap.save_ccmap()*. If the binary numpy array is compressed, this file is automatically extracted in the current working directory. After completion of the execution, this decompressed file will be automatically deleted. The compressed saved file will be remained unchanged.

Parameters

- **infile** (*str*) – Name of the inputp file including path to the directory/folder where file is saved.
- **workDir** (*str*) – Name of working directory, where temporary files will be kept.If `workDir = None`, file will be generated in OS based temporary directory.

Returns **ccMapObj** – A CCMAP object

Return type *gcMapExplorer.lib.ccmap.CCMAP*

export_cmap(*ccmap*, *outfile*, *doNotWriteZeros=True*)

To export `.ccmap` as text file

This function export `.ccmap` as coordinate list (COO) format sparse matrix file. In COO format, lists of (row, column, value) as three tab separated columns are written in output file.

Parameters

- **ccmap** (*gcMapExplorer.lib.ccmap.CCMAP*) – An instance of *gcMapExplorer.lib.ccmap.CCMAP*, which is need to be exported.
- **outfile** (*str*) – Output file name.
- **doNotWriteZeros** (*bool*) – Do not write Zero values. It reduces memory of file.

checkCCMapObjectOrFile(*ccMap*, *workDir=None*)

Check whether ccmap is a object or file

It can be used to check whether input is a *gcMapExplorer.lib.ccmap.CCMAP* or a ccmap file.

It returns the *gcMapExplorer.lib.ccmap.CCMAP* and input type name: i.e. `File` or `Object` as an identification keyword for the input.

In case if `ccMap` argument is a filename, this file will be opened as a *gcMapExplorer.lib.ccmap.CCMAP* object and will be returned with `ccmapType` as `File`.

In case if `ccMap` argument is a *gcMapExplorer.lib.ccmap.CCMAP* object, this file, same object will be returned with `ccmapType` as `Object`.

Parameters

- **ccMap** (*gcMapExplorer.lib.ccmap.CCMAP* or str) – CCMAP object or ccmap file.
- **workDir** (*str*) – Path to the directory where temporary intermediate files are generated. If `None`, files are generated in the temporary directory according to the main configuration.

Returns

- **ccMapObj** (*gcMapExplorer.lib.ccmap.CCMAP*) – CCMAP object
- **ccmapType** (*str*) – 'File' or 'Object'

downSampleCCMap(*cmap*, *level=2*, *method='sum'*, *workDir=None*)

Downsample or coarsen the contact map

It can be used to downsample the contact map by any given factor.

Parameters

- **level** (*int*) – The factor by which map has to be downsampled. For example, if input map has resolution of `10 kb` and `level = 4`, the ouput map will have `40 kb` resolution.
- **method** (*str*) – Method of downsampling. Three accepted methods are `sum`: sum all values, `mean`: Average of all values and `max`: Maximum of all values.

Returns **ccMapObj** – CCMAP object

Return type *gcMapExplorer.lib.ccmap.CCMAP*

getOutputShapeFor2DMapDownsampling(*inputShape*, *level=2*)
Helper function to determine output shape of map for downsampling

Parameters

- **level** (*int*) – The factor by which map has to be downsampled. For example, if input map has resolution of `10 kb` and `level = 4`, the ouput map will have `40 kb` resolution.
- **inputShape** (*(int, int)*) – Shape of input map as tuple. Preferably, CCMAP.shape.

Returns Output shape after downsampling the map

Return type outputShape (int, int)

downSample2DMap(*inMatrix*, *outMatrix=None*, *level=2*, *method='sum'*)
Downsample or coarsen the matrix

It can be used to downsample the matrix by any given factor. It is the core function used to downsample ccmap and gcmap.

Parameters

- **inMatrix** (*numpy.ndarray*) – Input 2D matrix of numpy array type.
- **outMatrix** (*numpy.ndarray*) – Output 2D numpy array in which all output values will be return. Note that shape of output array should be same as will be in downsampled array. To get the shape of output array priror to downsampling, use *gcMapExplorer.lib.ccmap.getOutputShapeFor2DMapDownsampling()*. In case if it is `None`, a output matrix will be returned.
- **level** (*int*) – The factor by which map has to be downsampled. For example, if input map has resolution of `10 kb` and `level = 4`, the ouput map will have `40 kb` resolution.
- **method** (*str*) – Method of downsampling. Three accepted methods are `sum`: sum all values, `mean`: Average of all values and `max`: Maximum of all values.

Returns **outMatrix** – In case if argument is `outMatrix=None`, output matrix will be returned.

Return type numpy.ndarray

ccmapHelpers module

ccmapHelpers.MemoryMappedArray	Convenient wrapper for numpy memory mapped array file
ccmapHelpers.MemoryMappedArray.copy	Copy this numpy memory mapped array and generate new
ccmapHelpers.MemoryMappedArray.copy_from	Copy values from source `MemoryMappedArray`
ccmapHelpers.MemoryMappedArray.copy_to	Copy values to destination `MemoryMappedArray`
ccmapHelpers.get_nonzeros_index(matrix[, ...])	To get a numpy array of bool values for all rows/columns which have **NO** missing data
ccmapHelpers.remove_zeros(matrix[, ...])	To remove rows/columns with missing data (zero values)

gcMapExplorer.ccmapHelpers

get_nonzeros_index(*matrix*, *threshold_percentile=None*, *threshold_data_occup=None*, *filterByDiagonal=False*)

To get a numpy array of bool values for all rows/columns which have **NO** missing data

Parameters

- **matrix** (numpy.memmap or *gcMapExplorer.lib.ccmap.CCMAP.matrix*) – Input matrix
- **percentile_threshold_no_data** (*int*) – It can be used to filter the map, where rows/columns with largest numbers of missing data can be discarded. `percentile_threshold_no_data` should be between 1 and 100. This options discard the rows and columns which are above this percentile. For example: if this value is 99, those row or columns will be discarded which contains larger than number of zeros (missing data) at 99 percentile.

  To calculate percentile, all blank rows are removed, then in all rows, number of zeros are counted. Afterwards, number of zeros at *percentile_threshold_no_data* percentile is obtained. In next step, if a row contain number of zeros larger than this percentile value, the whole row and column is assigned to have missing data. This percentile indicates highest numbers of zeros (missing data) in given rows/columns.
- **threshold_data_occup** (*float*) – It can be used to filter the map, where rows/columns with largest numbers of missing data can be discarded. This ratio is (number of bins with data) / (total number of bins in the given row/column). For example: if *threshold_data_occup = 0.8*, then all rows containing more than 20% of missing data will be discarded.

  Note that this parameter is suitable for low resolution data because maps are likely to be much less sparse.

Returns **bData** – 1D-array containing `True` and `False` values. * If `True`: row/column has data above the threshold * If `False`: row/column has no data under the threshold

Return type numpy.array[bool]

remove_zeros(*matrix*, *threshold_percentile=None*, *threshold_data_occup=None*, *workDir=None*)

To remove rows/columns with missing data (zero values)

Parameters

- **matrix** (numpy.memmap or *gcMapExplorer.lib.ccmap.CCMAP.matrix*) – Input matrix
- **percentile_threshold_no_data** (*int*) – It can be used to filter the map, where rows/columns with largest numbers of missing data can be discarded. `percentile_threshold_no_data` should be between 1 and 100. This options discard the rows and columns which are above this percentile. For example: if this value is 99, those row or columns will be discarded which contains larger than number of zeros (missing data) at 99 percentile.

  To calculate percentile, all blank rows are removed, then in all rows, number of zeros are counted. Afterwards, number of zeros at *percentile_threshold_no_data* percentile is obtained. In next step, if a row contain number of zeros larger than this percentile value, the whole row and column is assigned to have missing data. This percentile indicates highest numbers of zeros (missing data) in given rows/columns.
- **threshold_data_occup** (*float*) – It can be used to filter the map, where rows/columns with largest numbers of missing data can be discarded. This ratio is (num-

ber of bins with data) / (total number of bins in the given row/column). For example: if *threshold_data_occup = 0.8*, then all rows containing more than 20% of missing data will be discarded.

Note that this parameter is suitable for low resolution data because maps are likely to be much less sparse.

- **workDir** (*str*) – Path to the directory where temporary intermediate files are generated. If `None`, files are generated in the temporary directory according to the OS type.

Returns

- **A** (*MemoryMappedArray*) – *MemoryMappedArray* instance containing new truncated array as memory mapped file
- **bNoData** (*numpy.array[bool]*) – 1D-array containing `True` and `False` values. * If `True`: row/column has no data under the threshold * If `False`: row/column has data above the threshold

MemoryMappedArray class

class MemoryMappedArray

Convenient wrapper for numpy memory mapped array file

For more details, see here: (See:).

path2matrix

str – Path to numpy memory mapped array file

arr

numpy.memmap – Pointer to memory mapped numpy array

workDir

str – Path to the directory where temporary intermediate files are generated. If `None`, files are generated in the temporary directory according to the OS type.

dtype

str – Data type of array

Parameters

- **shape** (*tuple*) – Shape of array
- **fill** (*int or float (Optional)*) – Fill array with this value
- **dtype** (*str*) – Data type of array

copy

Copy this numpy memory mapped array and generate new

Returns **out** – A new *MemoryMappedArray* instance with copied arrays

Return type *MemoryMappedArray*

copy_from

Copy values from source *MemoryMappedArray*

Parameters **src** (*MemoryMappedArray*) – Source memory mapped arrays for new values

Returns

Return type None

Raises `ValueError` – if src is not of *MemoryMappedArray* instance

copy_to

Copy values to destination *MemoryMappedArray*

Parameters **dest** (*MemoryMappedArray*) – Destination memory mapped arrays

Returns

Return type None

Raises `ValueError` – if dest is not of *MemoryMappedArray* instance

KnightRuizNorm class

class KnightRuizNorm

A modified Knight-Ruiz algorithm for matrix balancing

The original ported Knight-Ruiz algorithm is modified to implement the normalization using both memory/RAM and disk. It allows the normalization of small Hi-C maps to huge maps that could not be accommodated in RAM.

Parameters

- **A** (numpy.ndarray or `MemoryMappedArray`) – Input matrix.

  Note:

  – Matrix should not contain any row or column with all zero values (missing data for row/column). This type of matrix can be obtained from `remove_zeros()`.

  – If `memory='HDD'`, `A` should be `MemoryMappedArray`

- **memory** (*str*) – Accepted keywords are `RAM` and `HDD`:

  – `RAM`: All intermediate arrays are generated in memory(RAM). This version is faster, however, it requires RAM depending on the input matrix size.

  – `HDD`: All intermediate arrays are generated as memory mapped array files on hard-disk.

- **workDir** (*str*) – Path to the directory where temporary intermediate files are generated. If `None`, files are generated in the temporary directory according to the OS type.

run

Perform Knight-Ruiz normalization

Parameters

- **A** (numpy.ndarray or `MemoryMappedArray.arr`) – Input matrix.

  Note:

  – Matrix should not contain any row or column with all zero values (missing data for row/column). This type of matrix can be obtained from `remove_zeros()`.

  Warning: If A was `MemoryMappedArray` in *KnightRuizNorm*. Here `A` should be `MemoryMappedArray.arr` instead of `MemoryMappedArray`.

- **fl** (*int*) – Its value should be zero

- **OutMatrix** (*gcMapExplorer.lib.ccmap.CCMAP.matrix*) – Output matrix of Hi-C map to which normalized matrix is returned.
- **bNoData** (*numpy.ndarray[bool]*) – A numpy.array containing bool to show if rows/columns have missing data. It can be obtained from `remove_zeros()`.

util module

This module contains some small and useful utility functions and classes.

util.resolutionToBinsize(resolution)	Return the bin size from the resolution unit
util.binsizeToResolution(binsize)	Return the resolution unit from the bin size
util.sorted_nicely(inputList)	Sorts the given given list in the way that is expected.
util.locate_significant_digit_after_decimal(value)	Get location at which significant digit start after decimal
util.kth_diag_indices(k, a)	Get diagonal indices of 2D array 'a' offset by 'k'
util.detectOutliers1D(points[, thresh])	Returns a boolean array with `True` if points are outliers and False otherwise.
util.getRandomName([size, chars])	Random name generator
util.MapNotFoundError(value)	
util.ResolutionNotFoundError(value)	

Exception classes

class MapNotFoundError(*value*)

class ResolutionNotFoundError(*value*)

Small utility functions

exception MapNotFoundError(*value*)

exception ResolutionNotFoundError(*value*)

binsizeToResolution(*binsize*)

Return the resolution unit from the bin size

It is a convenient function to convert binsize into resolution unit. It has a support of base (b), kilobase (kb), megabase (mb) and gigabase (gb) unit. It also convert binsize to decimal resolution unit as shown below in examples.

Parameters **binsize** (*int*) – bin size

Returns **resolution** – resolution unit

Return type str

Examples

```
>>> binsizeToResolution(1)
'1b'
>>> binsizeToResolution(10)
'10b'
>>> binsizeToResolution(10000)
```

(continues on next page)

(continued from previous page)

```
'10kb'
>>> binsizeToResolution(100000)
'100kb'
>>> binsizeToResolution(125500)
'125.5kb'
>>> binsizeToResolution(1000000)
'1mb'
>>> binsizeToResolution(1634300)
'1.6343mb'
```

detectOutliers1D (*points*, *thresh=3.5*)

Returns a boolean array with `True` if points are outliers and False otherwise.

Parameters

- **points** (*numpy.ndarray*) – An numobservations by numdimensions array of observations
- **thresh** (*float*) – The modified z-score to use as a threshold. Observations with a modified z-score (based on the median absolute deviation) greater than this value will be classified as outliers.

Returns **outBool** – A numobservations-length boolean array.

Return type numpy.ndarray

References

Boris Iglewicz and David Hoaglin (1993), "Volume 16: How to Detect and Handle Outliers", The ASQC Basic References in Quality Control: Statistical Techniques, Edward F. Mykytka, Ph.D., Editor.

detectOutliersMasked1D (*points*, *thresh=3.5*)

Returns a masked array where outliers are masked with preserved input mask.

Parameters

- **points** (*numpy.ma.ndarray*) – An numobservations by numdimensions array of observations
- **thresh** (*float*) – The modified z-score to use as a threshold. Observations with a modified z-score (based on the median absolute deviation) greater than this value will be classified as outliers.

Returns **maskArray** – A numobservations-length masked array.

Return type numpy.ma.ndarray

References

Boris Iglewicz and David Hoaglin (1993), "Volume 16: How to Detect and Handle Outliers", The ASQC Basic References in Quality Control: Statistical Techniques, Edward F. Mykytka, Ph.D., Editor.

getRandomName (*size=10*, *chars='abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789'*)

Random name generator

Parameters **size** (*int*) – Number of alphabets in the name.

Returns **name** – Randomly generated name.

Return type str

kth_diag_indices(*k*, *a*)
Get diagonal indices of 2D array 'a' offset by 'k'

Parameters

- **k** (*int*) – Diagonal offset
- **a** (*numpy.ndarray*) – Input numpy 2D array

Returns **indices** – It contain indences of elements that are at the diagonal offset by 'k'.

Return type tuple of two numpy.ndarray

locate_significant_digit_after_decimal(*value*)
Get location at which significant digit start after decimal

Parameters **value** (*float*) – Input value

Returns **value** – Number of zeros after which digit start in a small decimal number.

Return type int

resolutionToBinsize(*resolution*)
Return the bin size from the resolution unit

It is a convenient function to convert resolution unit to binsize. It has a support of base (b), kilobase (kb), megabase (mb) and gigabase (gb) unit. It also convert decimal resolution unit as shown below in examples.

Parameters **resolution** (*str*) – resolution in b, kb, mb or gb.

Returns **binsize** – bin size

Return type int

Examples

```
>>> resolutionToBinsize('1b')
1
>>> resolutionToBinsize('10b')
10
>>> resolutionToBinsize('1kb')
1000
>>> resolutionToBinsize('16kb')
16000
>>> resolutionToBinsize('1.23kb')
1230
>>> resolutionToBinsize('1.6mb')
1600000
>>> resolutionToBinsize('1.457mb')
1457000
```

sorted_nicely(*inputList*)
Sorts the given given list in the way that is expected.

Parameters **inputList** (*list*) – The input list to be sorted.

Returns **outputList** – The sorted list

Return type list

gcmap module

gcmap.GCMAP(hdf5[, mapName, chromAtX, ...])	To access Genome wide contact map.
gcmap.GCMAP.checkMapExist([mapName, ...])	Check if a map is exist in the file
gcmap.GCMAP.changeMap([mapName, chromAtX, ...])	Change the map for another chromosome
gcmap.GCMAP.changeResolution(resolution)	Try to change contact map of a given resolution.
gcmap.GCMAP.toFinerResolution()	Try to change contact map to next finer resolution
gcmap.GCMAP.toCoarserResolution()	Try to change contact map to next coarser resolution
gcmap.GCMAP.loadSmallestMap([resolution])	Load smallest sized contact map
gcmap.GCMAP.genMapNameList([sortBy])	Generate list of contact maps available in gcmap file
gcmap.GCMAP.performDownSampling([method])	Downsample recursively and store the maps
gcmap.GCMAP.downsampleMapToResolution(resolution)	Downsample the current map to a particular resolution
gcmap.GCMAP.downsampleAllMapToResolution()	Downsample all maps to a particular resolution
gcmap.loadGCMapAsCCMap(filename[, mapName, ...])	Load a map from gcmap file as a *gcMapExplorer.lib.ccmap.CCMAP*.
gcmap.addCCMap2GCMap(cmap, filename[, ...])	Add *gcMapExplorer.lib.ccmap.CCMAP* to a gcmap file
gcmap.changeGCMapCompression(infile, ...[, ...])	Change compression method in GCMAP file

GCMAP class

class GCMAP(*hdf5*, *mapName=None*, *chromAtX=None*, *chromAtY=None*, *resolution=None*)

To access Genome wide contact map.

It is similar to *gcMapExplorer.lib.ccmap.CCMAP* and contains same attributes. Therefore, both *gcMapExplorer.lib.ccmap.CCMAP* and GCMAP can be used in same way to access attributes. It also contains additional attributes because it uses HDF5 file to read the maps on demand.

Structure of gcmap file:

```
HDF5
│
├──────── chr1 ─── Attributes : ['xlabel', 'ylabel', 'compression']
│          │
│          ├──────── 10kb  ( 2D Numpy Array ) ─── Attributes : ['minvalue',
↪ 'maxvalue', 'xshape', 'yshape', 'binsize']
│          ├──────── 20kb  ( 2D Numpy Array ) ─── Attributes : ['minvalue',
↪ 'maxvalue', 'xshape', 'yshape', 'binsize']
│          ├──────── 40kb  ( 2D Numpy Array ) ─── Attributes : ['minvalue',
↪ 'maxvalue', 'xshape', 'yshape', 'binsize']
│          ├──────── 60kb  ( 2D Numpy Array ) ─── Attributes : ['minvalue',
↪ 'maxvalue', 'xshape', 'yshape', 'binsize']
│          ├──────── 80kb  ( 2D Numpy Array ) ─── Attributes : ['minvalue',
↪ 'maxvalue', 'xshape', 'yshape', 'binsize']
│          ├──────── 160kb ( 2D Numpy Array ) ─── Attributes : ['minvalue',
↪ 'maxvalue', 'xshape', 'yshape', 'binsize']
│          ├──────── 320kb ( 2D Numpy Array ) ─── Attributes : ['minvalue',
↪ 'maxvalue', 'xshape', 'yshape', 'binsize']
│          ├──────── 640kb ( 2D Numpy Array ) ─── Attributes : ['minvalue',
↪ 'maxvalue', 'xshape', 'yshape', 'binsize']
│          │
│          ├──────── 10kb-bNoData  ( 1D Numpy Array )
```

(continues on next page)

(continued from previous page)

```
│              ├──────── 20kb-bNoData  ( 1D Numpy Array )
│              ├──────── 40kb-bNoData  ( 1D Numpy Array )
│              ├──────── 60kb-bNoData  ( 1D Numpy Array )
│              ├──────── 80kb-bNoData  ( 1D Numpy Array )
│              ├──────── 160kb-bNoData ( 1D Numpy Array )
│              ├──────── 320kb-bNoData ( 1D Numpy Array )
│              └──────── 640kb-bNoData ( 1D Numpy Array )
│
├──────────── chr2 ──── Attributes : ['xlabel', 'ylabel', 'compression']
│              │
│              ├──────── 10kb  ( 2D Numpy Array ) ──── Attributes : ['minvalue',
↪ 'maxvalue', 'xshape', 'yshape', 'binsize']
│              ├──────── 20kb  ( 2D Numpy Array ) ──── Attributes : ['minvalue',
↪ 'maxvalue', 'xshape', 'yshape', 'binsize']
│              ├──────── 40kb  ( 2D Numpy Array ) ──── Attributes : ['minvalue',
↪ 'maxvalue', 'xshape', 'yshape', 'binsize']
│              ├──────── 60kb  ( 2D Numpy Array ) ──── Attributes : ['minvalue',
↪ 'maxvalue', 'xshape', 'yshape', 'binsize']
│              ├──────── 80kb  ( 2D Numpy Array ) ──── Attributes : ['minvalue',
↪ 'maxvalue', 'xshape', 'yshape', 'binsize']
│              ├──────── 160kb ( 2D Numpy Array ) ──── Attributes : ['minvalue',
↪ 'maxvalue', 'xshape', 'yshape', 'binsize']
│              ├──────── 320kb ( 2D Numpy Array ) ──── Attributes : ['minvalue',
↪ 'maxvalue', 'xshape', 'yshape', 'binsize']
│              ├──────── 640kb ( 2D Numpy Array ) ──── Attributes : ['minvalue',
↪ 'maxvalue', 'xshape', 'yshape', 'binsize']
│              │
│              ├──────── 10kb-bNoData  ( 1D Numpy Array )
│              ├──────── 20kb-bNoData  ( 1D Numpy Array )
│              ├──────── 40kb-bNoData  ( 1D Numpy Array )
│              ├──────── 60kb-bNoData  ( 1D Numpy Array )
│              ├──────── 80kb-bNoData  ( 1D Numpy Array )
│              ├──────── 160kb-bNoData ( 1D Numpy Array )
│              ├──────── 320kb-bNoData ( 1D Numpy Array )
│              └──────── 640kb-bNoData ( 1D Numpy Array )
:
:
:
└──────── ...
```

Note:

- Reading the entire map from HDF5 could be time taking for very large map. Therefore, use this class in cases like read small region of map or read only once.
- To perform calculation, use *gcMapExplorer.lib.gcmap.loadGCMapAsCCMap()* as it returns *gcMapExplorer.lib.ccmap.CCMAP*.

The class is instantiated by two methods:

```
>>> ccMapObj = gcMapExplorer.lib.ccmap.CCMAP(hdf5, 'chr22')     # To read␣
↪chr22 vs chr22 map
>>> ccMapObj.matrix[200:400, 200:400]  # Read region between 200 to 400 of␣
↪chr22 vs chr22 map.
```

Parameters

- **hdf5** (*str or h5py.File*) – Either gcmap file name or h5py file object, which is an entry point to HDF5 file.
- **mapName** (*str*) – Name of map. It could be chromosome name in case of intra-chromosomal map. e.g.: `chr1` or `chr2`.
- **chromAtX** (*str*) – chromosome at X-axis. In case of intra-chromosomal map, this is not required because both at X and Y axis same chromosome is present.
- **chromAtX** – chromosome at Y-axis. In case of intra-chromosomal map, this is not required because both at X and Y axis same chromosome is present. If `chromAtY = None`, both x-axis and y-axis contains same chromosome and map is of 'intra' of 'cis' type.
- **resolution** (*str*) – Input resolution to read from file.

yticks
list – Minimum and maximum locations along Y-axis. e.g. `yticks=[0, 400000]`

xticks
list – Minimum and maximum locations along X-axis. e.g. `xticks=[0, 400000]`

binsize
int – Resolution of data. In case of 10kb resolution, binsize is 10000.

title
str – Title of the data

xlabel
str – Title for X-axis, which is chromosome name along X-axis

ylabel
str – Title for Y-axis, which is chromosome name along Y-axis

shape
tuple – Overall shape of matrix

minvalue
float – Minimum value in matrix

maxvalue
float – Maximum value in matrix

matrix
h5py.Dataset – A HDF5 Dataset object pointing to matrix/map. See here for more details:

bNoData
numpy.ndarray – A boolean numpy array of matrix shape

bLog
bool – If values in matrix are in log

dtype
str – Data type of matrix/map

mapType
str – Type of map: `intra` or `inter` chromosomal map. If chromosome along X- and Y- axis is same, then map is intra-chromosomal, otherwise map is inter-chromosomal.

hdf5
h5py.File – HDF5 file object instance

fileOpened
bool – Whether a file is opened inside object or a HDF5 file object is provided to object. When a file is opened by object, it is closed before object is destroyed.

groupName
str – Name of current contact map or group name in HDF5 file

resolution
str – Resolution of current contact map

finestResolution
str – Finest available resolution of current contact map

binsizes
list – List of binsizes available for current contact map

mapNameList
list – List of all available contact maps in gcmap file

changeMap (*mapName=None*, *chromAtX=None*, *chromAtY=None*, *resolution=None*)
Change the map for another chromosome

It can be used to change the map. For example, to access the map of 'chr20' instead of 'chr22', use this function.

Parameters

- **mapName** (*str*) – Name of map. It could be chromosome name in case of intra-chromosomal map. e.g.: `chr1` or `chr2`.
- **chromAtX** (*str*) – chromosome at X-axis. In case of intra-chromosomal map, this is not required because both at X and Y axis same chromosome is present.
- **chromAtX** – chromosome at Y-axis. In case of intra-chromosomal map, this is not required because both at X and Y axis same chromosome is present. If `chromAtY = None`, both x-axis and y-axis contains same chromosome and map is of 'intra' of 'cis' type.
- **resolution** (*str*) – Input resolution to read from file.

For example:

```
>>> ccMapObj = gcMapExplorer.lib.ccmap.CCMAP(hdf5, 'chr22')     # To read␣
↪chr22 vs chr22 map
>>> ccMapObj.matrix[200:400, 200:400]  # To access region between 200 to␣
↪400 of chr22 vs chr22 map.
>>> ccMapObj.changeMap('chr20')        # Changed to read chr20 vs chr20␣
↪map
>>> ccMapObj.matrix[200:400, 200:400]  # Now, to access region between␣
↪200 to 400 of chr20 vs chr20 map.
```

changeResolution (*resolution*)
Try to change contact map of a given resolution.

Parameters **resolution** (*str*) – Resolution to change.

Returns **success** – If change was successful `True` otherwise `False`.

Return type bool

checkMapExist (*mapName=None*, *chromAtX=None*, *chromAtY=None*, *resolution=None*)
Check if a map is exist in the file

It can be used to check if a map is exist in the file.

Parameters

- **mapName** (*str*) – Name of map. It could be chromosome name in case of intra-chromosomal map. e.g.: `chr1` or `chr2`.
- **chromAtX** (*str*) – chromosome at X-axis. In case of intra-chromosomal map, this is not required because both at X and Y axis same chromosome is present.
- **chromAtX** – chromosome at Y-axis. In case of intra-chromosomal map, this is not required because both at X and Y axis same chromosome is present. If `chromAtY = None`, both x-axis and y-axis contains same chromosome and map is of 'intra' of 'cis' type.
- **resolution** (*str*) – Input resolution to read from file.

Returns **doExist** – If map is present then `True` otherwise `False`.

Return type bool

downsampleAllMapToResolution(*resolution*, *method='sum'*)

Downsample all maps to a particular resolution

By default, maps with only few resolutions are generated. For example, when finest resolution is 5 kb, the downsampled map with 10kb, 20kb, 40kb, 80kb, 160kb etc are generated. If other resolution (e.g. 100kb) is required, then it can be used.

Parameters

- **resolution** (*str*) – Resolution to downsample.
- **method** (*str*) – Method of downsampling. Three accepted methods are `sum`: sum all values, `mean`: Average of all values and `max`: Maximum of all values.

Returns **success** – `True` or `False`

Return type bool

downsampleMapToResolution(*resolution*, *method='sum'*)

Downsample the current map to a particular resolution

By default, maps with only few resolutions are generated. For example, when finest resolution is 5 kb, the downsampled map with 10kb, 20kb, 40kb, 80kb, 160kb etc are generated. If other resolution (e.g. 100kb) is required, then it can be used.

Note: It only downsample for current map. To downsample all maps, look `gcmap.GCMAP.downsampleAllMapToResolution()`.

Parameters

- **resolution** (*str*) – Resolution to downsample
- **method** (*str*) – Method of downsampling. Three accepted methods are `sum`: sum all values, `mean`: Average of all values and `max`: Maximum of all values.

Returns **success** – `True` or `False`

Return type bool

genMapNameList (*sortBy='name'*)
Generate list of contact maps available in gcmap file

The maps can be either sorted by name or by size. The listed maps are in `GCMAP.mapNameList`.

Parameters **sortBy** (*str*) – Accepted keywords are `name` and `size`.

get_ticks (*binsize=None*)
To get xticks and yticks for the matrix

Parameters **binsize** (*int*) – Number of base in each bin or pixel or box of contact map.

Returns

- **xticks** (*numpy.array*) – 1D array containing positions along X-axis
- **yticks** (*numpy.array*) – 1D array containing positions along X-axis

loadSmallestMap (*resolution=None*)
Load smallest sized contact map

Parameters **resolution** (*str*) – Resolution to change. When it is not provided, finest resolution of smallest maps is loaded.

performDownSampling (*method='sum'*)
Downsample recursively and store the maps

It Downsample the maps and automatically add it to same input `gcmap` file. Downsampling works recursively, and downsampled maps are generated until map has a size of less than 500.

Parameters **method** (*str*) – Method of downsampling. Three accepted methods are `sum`: sum all values, `mean`: Average of all values and `max`: Maximum of all values.

toCoarserResolution ()
Try to change contact map to next coarser resolution

Returns **success** – If change was successful `True` otherwise `False`.

Return type bool

toFinerResolution ()
Try to change contact map to next finer resolution

Returns **success** – If change was successful `True` otherwise `False`.

Return type bool

gcMapExplorer.gcmap

loadGCMapAsCCMap (*filename*, *mapName=None*, *chromAtX=None*, *chromAtY=None*, *resolution=None*, *workDir=None*)
Load a map from gcmap file as a *gcMapExplorer.lib.ccmap.CCMAP*.

Parameters

- **filename** (*str*) – Either a gcmap file or h5py.File instance or GCMAP.hdf5 from which contact map data will be read.
- **mapName** (*str*) – Name of contact map. e.g.: `chr1` or `chr2`.
- **chromAtX** (*str*) – Name of chromosome at X-axis
- **chromAtY** (*str*) – Name of chromosome at Y-axis. If `chromAtY = None`, both x-axis and y-axis contains same chromosome and map is of 'intra' of 'cis' type.

- **resolution** (*str*) – Resolution of required map. If contact map of input resolution is not found, `None` will be returned.
- **workDir** (*str*) – Name of directory where temporary files will be kept. These files will be automatically deleted.

Returns object

Return type None or *gcMapExplorer.lib.ccmap.CCMAP*

addCCMap2GCMap(*cmap*, *filename*, *scaleoffset=None*, *compression='lzf'*, *generateCoarse=True*, *coarseningMethod='sum'*, *replaceCMap=True*, *logHandler=None*)

Add *gcMapExplorer.lib.ccmap.CCMAP* to a gcmap file

Parameters

- **cmap** (*gcMapExplorer.lib.ccmap.CCMAP*) – An instance of *gcMapExplorer.lib.ccmap.CCMAP*, which will be added to gcmap file
- **filename** (*str*) – Name of `gcmap` file or h5py.File instance or GCMAP.hdf5 to which output data will be written.
- **scaleoffset** (*int*) – For integer data, this specifies the number of bits to retain in hdf5 file. In case of `0` value, HDF5 automatically compute the number of bits required for lossless compression of the chunk. For floating-point data, indicates the number of digits after the decimal point to retain. This can help to reduce the final file size. In case of `None` data will be stored without any loss of precision.
- **compression** (*str*) – Compression method. Presently allowed : `lzf` for LZF compression and `gzip` for GZIP compression.
- **generateCoarse** (*bool*) – Also generates all coarser maps where resolutions will be coarsen by a factor of two, consecutively. e.g.: In case of 10 kb input resolution, downsampled maps of `20kb`, `40kb`, `80kb`, `160kb`, `320kb` etc. will be generated until, map size is less than 500.
- **coarseningMethod** (*str*) – Method of downsampling. Three accepted methods are `sum`: sum all values, `mean`: Average of all values and `max`: Maximum of all values.
- **replaceCMap** (*bool*) – Replace entire old ccmap data including resolutions and coarsen data.

Returns **success** – If addition was successful `True` otherwise `False`.

Return type bool

changeGCMapCompression(*infile*, *outfile*, *compression*, *logHandler=None*)

Change compression method in GCMAP file

Change compression method in GCMAP file. Currently LZF and GZIP compression is allowed. LZF is fast and moderately compressing algorithm. However, GZIP is slower with large compressing ratio.

> **Warning:** GCMAP with `gzip` compression can be universally read from any programming language using HDF5 library, however `LZF` compression can be only decompressed using Python h5py package.

Parameters

- **infile** (*str*) – Input GCMAP file
- **outfile** (*str*) – Output GCMAP file
- **compression** (*str*) – Method of compression: *lzf*‘ or `gzip`

importer module

importer.CooMatrixHandler([inputFiles, ...])	To import ccmap from files similar to sparse matrix in Coordinate (COO) format
importer.CooMatrixHandler.save_ccmaps([...])	To Save all Hi-C maps
importer.CooMatrixHandler.save_gcmap(outputFile)	To Save all Hi-C maps as a gcmap file
importer.CooMatrixHandler.setLabels(xlabels, ...)	To set xlabels and ylabels for contact maps
importer.CooMatrixHandler.setOutputFileList(...)	To set list of output files
importer.PairCooMatrixHandler(inputFile[, ...])	To import ccmap from files similar to paired sparse matrix Coordinate (COO) format
importer.PairCooMatrixHandler.setGCMapOptions([...])	Set options for output gcmap file
importer.PairCooMatrixHandler.runConversion()	Perform conversion and save to ccmap and/or gcmap file.
importer.HomerInputHandler([inputFiles, ...])	To import ccmap from Hi-C maps generated by HOMER
importer.HomerInputHandler.save_ccmaps(outdir)	Import and save ccmap file
importer.HomerInputHandler.save_gcmap(outputFile)	To Save all Hi-C maps as a gcmap file
importer.BinsNContactFilesHandler(binFile, ...)	To import Hi-C map from bin and contact file in list format
importer.BinsNContactFilesHandler.save_ccmaps(outdir)	Import and save ccmap file
importer.BinsNContactFilesHandler.save_gcmap(...)	To Save all Hi-C maps as a gcmap file
importer.gen_map_from_locations_value(i, j, ...)	To generate CCMAP object from three lists – i, j, value

CooMatrixHandler class

class CooMatrixHandler(*inputFiles=None*, *inputCompressedFile=None*, *mapType='intra'*, *resolution=None*, *coordinate='real'*, *workDir=None*, *logHandler=None*)

To import ccmap from files similar to sparse matrix in Coordinate (COO) format

Two types of coordinates are accepted:

- with `coordinate='real'` pair of absolute binned locations on chromosome
- with `coordinate='index'` row and column index of matrix. index **should** always start from zero for absolute beginning of chromosome. e.g. for 10kb, 0-10000 should have index of zero, 10000-20000 have index of one. If this is file format, resolution should be provided explicitly.

> **Warning:** Input file should contain matrix for **only one** chromosome.

Following file format can be read as a text file, where first and second column is location on chromosome and third column is the value:

```
20000000        20000000        2692.0
20000000        20100000        885.0
20100000        20100000        6493.0
20000000        20200000        15.0
20100000        20200000        52.0
20200000        20200000        2.0
20000000        20300000        18.0
20100000        20300000        40.0
.
.
.
.
.
.
```

To Instantiate this class, three scenarios are possible:

- `Text in Archive`: Both a list of input files and a compressed file is given. It means look for input files in compressed file.
- `Text`: Only a list of input files is given. It means, read data directly from input file.
- `Archive`: Only a compressed file is given. It means, read all files present in compressed files.

Parameters

- **inputFiles** (*str or list*) – name of a input file or list of input files
- **inputCompressedFile** (*str*) – name of input tar archive
- **mapType** (*str*) – Type of HiC map
  - `intra`: for intra-chromosomal map
  - `inter`: for inter-chromosomal map
- **resolution** (*str*) – resolution of HiC map. Example: '100kb', '10kb' or '25kb' etc. If `None`, resolution will be determined from the input file.
- **workDir** (*str*) – Directory where temporary files will be stored.

inputFileList
list – List of input files. It could be `None` when not provided.

inputType
str – Type of input. Three types: `Text`, `Text in Archive` and `Archive` are determined from user input.

inputCompressedFile
str – Name of input compressed file. It could be `None` when not provided.

outputFileList
str – List of Output files

compressType
str – Format of compressed file. It could be either `.tar` or `.zip` or `None`

compressHandle
ZipFile or TarFile – An object to handle compressed file. It could be ZipFile or TarFile instance depending on compressed format.

mapType
str – Types of HiC map * `intra`: for intra-chromosomal map * `inter`: for inter-chromosomal map

coordinate
str – Coordinate type in input text file. It could be either `real` for real locations or `index` for rows and column indices.

resolution
str – resolution of HiC map. Example: '100kb', '10kb' or '25kb' etc. If `None`, resolution will be determined from the input file.

If `coordinate='index'`, resolution is essential for further processing.

workDir
str – Directory where temporary files will be stored. If not provided, directory name will be taken from configuration file.

save_ccmaps(*outputFiles=None*, *xlabels=None*, *ylabels=None*, *compress=True*)
To Save all Hi-C maps

This function reads input files one by one and save it as a `.ccmap` file.

Parameters

- **outputFiles** (*str or list*) – Name of a output file or list of output files. For each input file, a respective output file will be generated, therefore, number of input and output files should match.
- **xlabels** (*str or list*) – Name of the data along X-axis or list of names of the data for respective input files.
- **ylabels** (*str or list*) – Name of the data along y-axis or list of names of the data for respective input files. if it is `None`, ylabels will be same as xlabels.
- **compress** (*bool*) – If `True`, numpy array (matrix) file will be compressed to reduce storage memory.

save_gcmap(*outputFile*, *xlabels=None*, *ylabels=None*, *coarseningMethod='sum'*, *compression='lzf'*)
To Save all Hi-C maps as a gcmap file

This function reads input files one by one and save it as a `.gcmap` file.

Parameters

- **outputFile** (*str*) – Name of a output gcmap file.
- **xlabels** (*str or list*) – Name of the data along X-axis or list of names of the data for respective input files.
- **ylabels** (*str or list*) – Name of the data along y-axis or list of names of the data for respective input files. if it is `None`, ylabels will be same as xlabels.
- **coarseningMethod** (*str*) – Method of downsampling. Three accepted methods are `sum`: sum all values, `mean`: Average of all values and `max`: Maximum of all values.
- **compression** (*str*) – Compression method. Presently allowed : `lzf` for LZF compression and `gzip` for GZIP compression.

setLabels(*xlabels*, *ylabels*)
To set xlabels and ylabels for contact maps

xlabel and ylabel act as a title of data along X-axis and Y-axis respectively.

Parameters

- **xlabels** (*str or list*) – Name of the data along X-axis or list of names of the data for respective input files.
- **ylabels** (*str or list*) – Name of the data along y-axis or list of names of the data for respective input files.

setOutputFileList(*outputFiles*)

To set list of output files

Parameters **outputFiles** (*str or list*) – Name of a output file or list of output files. For each input file, a respective output file will be generated, therefore, number of input and output files should match.

PairCooMatrixHandler class

class PairCooMatrixHandler(*inputFile*, *ccmapOutDir=None*, *ccmapSuffix=None*, *gcmapOut=None*, *workDir=None*, *logHandler=None*)

To import ccmap from files similar to paired sparse matrix Coordinate (COO) format

This format is very similar to COO format with additional information of chromosome. Therefore, maps for all chromosome could be contained in a single file.

This type of format appeared with following publication:

- http://dx.doi.org/10.1016/j.cell.2015.10.026 — https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE72512

Following file format can be read as a text file, where first and second column is location on chromosome and third column is the value:

```
chr4    60000   75000   chr4    60000   75000   0.1163470887070292
chr4    60000   75000   chr4    105000  120000  0.01292745430078102
chr4    60000   75000   chr4    435000  450000  0.01292745430078102
chr4    75000   90000   chr4    75000   90000   0.05170981720312409
chr4    75000   90000   chr4    345000  360000  0.01292745430078102
chr4    90000   105000  chr4    90000   105000  0.01292745430078102
.
.
.
.
.
.
```

Parameters

- **inputFile** (*str*) – name of a input file
- **ccmapOutDir** (*str*) – name of directory where all ccmap file will be stored.
- **ccmapSuffix** (*str*) – Suffix for ccmap file name.
- **gcmapOut** (*str*) – Name of output gcmap file.
- **workDir** (*str*) – Directory where temporary files will be stored.

inputFile

str – name of a input file

ccmapOutDir

str – name of directory where all ccmap file will be stored.

ccmapSuffix
str – Suffix for ccmap file name.

gcmapOut
str – Name of output gcmap file.

gcmapOutOptions
dict – Dictionary for gcmap output options.

workDir
str – Directory where temporary files will be stored.

Examples

```
>>> pair_map_handle = PairCooMatrixHandler('GSM1863750_tethered_rep1_contacts.txt', gcmapOut='GSM1863750_tethered_rep1_contacts.gcmap')
>>> pair_map_handle.setGCMapOptions()
>>> pair_map_handle.runConversion()
```

runConversion()
Perform conversion and save to ccmap and/or gcmap file.

Read the input file, process the data, and convert it to ccmap or gcmap file. For output gcmap, *PairCooMatrixHandler.setGCMapOptions()* should be called to set the necessary options.

setGCMapOptions(*compression='lzf'*, *generateCoarse=True*, *coarseningMethod='sum'*, *replaceCMap=True*)
Set options for output gcmap file

Parameters

- **compression** (*str*) – Compression method. Presently allowed : `lzf` for LZF compression and `gzip` for GZIP compression.
- **generateCoarse** (*bool*) – Also generates all coarser maps where resolutions will be coarsen by a factor of two, consecutively. e.g.: In case of 10 kb input resolution, downsampled maps of `20kb`, `40kb`, `80kb`, `160kb`, `320kb` etc. will be generated until, map size is less than 500.
- **coarseningMethod** (*str*) – Method of downsampling. Three accepted methods are `sum`: sum all values, `mean`: Average of all values and `max`: Maximum of all values.
- **replaceCMap** (*bool*) – Replace entire old ccmap data including resolutions and coarsened data.

HomerInputHandler class

class HomerInputHandler(*inputFiles=None*, *inputCompressedFile=None*, *workDir=None*, *logHandler=None*)
To import ccmap from Hi-C maps generated by HOMER

HOMER package generates the Hi-C interaction matrices in text file. This Hi-C interaction file can be imported using this class. HOMER format interaction matrix file may contain data for all the chromosomes while this class separately read and save matrix of each chromosome.

To Instantiate this class, three scenarios are possible:

- `Text in Archive`: Both a list of input files and a compressed file is given. It means look for input files in compressed file.
- `Text`: Only a list of input files is given. It means, read data directly from input file.
- `Archive`: Only a compressed file is given. It means, read all files present in compressed files.

Parameters

- **inputFiles** (*str or list*) – Name of a input file or list of input files. If `None`, all files from compressed files are used as input files.
- **inputCompressedFile** (*str*) – Name of input compressed file. Accepted formats: `tar.gz`, `tar.bz2` and `zip`.
- **workDir** (*str*) – Directory where temporary files will be stored. If it is not provided, this value is taken from configuration file.

inputFileList
list – List of input files. It could be `None` when not provided.

inputType
str – Type of input. Three types: `Text`, `Text in Archive` and `Archive` are determined from user input.

inputCompressedFile
str – Name of input compressed file. It could be `None` when not provided.

compressType
str – Format of compressed file. It could be either `.tar` or `.zip` or `None`

compressHandle
ZipFile or TarFile – An object to handle compressed file. It could be ZipFile or TarFile instance depending on compressed format.

workDir
str – Directory where temporary files will be stored.

fIns
list[output file stream] – Input file stream for each input files

chromList
list[str] – List of chromosome found in input files

resolution
str – Resolution of map

fTmpOutNames
list[str] – List of temporary output files where data for each chromosomes are extracted separately

fTmpOut
list[output file stream] – List of output file streams for respective temporary output files

save_ccmaps(*outdir*, *suffix=None*)
Import and save ccmap file

Read input files, save data temporarily in text file for each chromosome and import these data to native ccmap format using *CooMatrixHandler* class.

Note:

- Output file names will be automatically generated as `<chromosome>_<resolution>.ccmap` format. e.g. `chr12_10kb.ccmap`.
- A suffix can be added to all files as `<chromosome>_<resolution>_<suffix>.ccmap` format. e.g. if `suffix='_RawObserved'`, file name is `chr12_10kb_RawObserved.ccmap`.

Parameters

- **outdir** (*str*) – Path to directory where all ccmaps have to be saved
- **suffix** (*str*) – Any suffix to file name

save_gcmap(*outputFile*, *coarseningMethod='sum'*, *compression='lzf'*)

To Save all Hi-C maps as a gcmap file

This function reads input files one by one and save it as a `.gcmap` file.

Parameters

- **outputFile** (*str*) – Name of a output gcmap file.
- **coarseningMethod** (*str*) – Method of downsampling. Three accepted methods are `sum`: sum all values, `mean`: Average of all values and `max`: Maximum of all values.
- **compression** (*str*) – Compression method. Presently allowed : `lzf` for LZF compression and `gzip` for GZIP compression.

BinsNContactFilesHandler class

class BinsNContactFilesHandler(*binFile*, *contactFile*, *workDir=None*, *logHandler=None*)

To import Hi-C map from bin and contact file in list format

These types of files are appeared in following GEO data:

- http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE61471
- http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE34453

Parameters

- **binFile** (*str*) – Bin file
- **contactFile** (*str*) – Contact file in list format
- **workDir** (*str*) – Directory where temporary files will be stored. If it is not provided, this value is taken from configuration file.

binFile
str – Bin file

contactFile
str – Contact file in list format

ChromSize
dict – Dictionary of Chromosome size

ChromBinsInfo
dict – Dictionary of min (start) and max (end) bin number for each Chromosome

npyBinFileList
dict – Dictionary containing tuple (memmap stream, Temporary numpy array file name)

binsize
int – Bin size of Hi-C map

ccmaps
dict – Dictionary of CCMAP instances for each Chromosome

save_ccmaps(*outdir*, *suffix=None*)
Import and save ccmap file

Read input files, save data temporarily for each chromosome and import these data to native ccmap format.

..note::

- Output file names will be automatically generated as `<chromosome>_<resolution>.ccmap` format. e.g. `chr12_10kb.ccmap`.
- A suffix can be added to all files as `<chromosome>_<resolution><suffix>.ccmap` format. e.g. if `suffix='_RawObserved'`, file name is `chr12_10kb_RawObserved.ccmap`.

Parameters

- **outdir** (*str*) – Path to directory where all ccmaps have to be saved
- **suffix** (*str*) – Any suffix to file name

save_gcmap(*outputFile*, *coarseningMethod='sum'*, *compression='lzf'*)
To Save all Hi-C maps as a gcmap file

This function reads input files one by one and save it as a `.gcmap` file.

Parameters

- **outputFile** (*str*) – Name of a output gcmap file.
- **coarseningMethod** (*str*) – Method of downsampling. Three accepted methods are `sum`: sum all values, `mean`: Average of all values and `max`: Maximum of all values.
- **compression** (*str*) – Compression method. Presently allowed : `lzf` for LZF compression and `gzip` for GZIP compression.

Other functions of importer module

gen_map_from_locations_value(*i*, *j*, *value*, *resolution=None*, *mapType='intra'*, *workDir=None*, *logHandler=None*)
To generate CCMAP object from three lists – i, j, value

Parameters

- **i** (*list[int]*) – List of first location from each pair
- **j** (*list[int]*) – List of second location from each pair
- **resolution** (*str*) – Resolution of Hi-C map
- **mapType** (*str*) – Hi-C map type: `intra` or `inter` chromosomal map
- **value** (*list[float]*) – List of values for respective location
- **workDir** (*str*) – Directory where temporary files will be stored. If it is not provided, this value is taken from configuration file.

Returns **ccMapObj** – A CCMAP object

Return type *gcMapExplorer.lib.ccmap.CCMAP*

normalizer module

normalizer.NormalizeKnightRuizOriginal(ccMapObj[, ...])	Original Knight-Ruiz algorithm for matrix balancing
normalizer.normalizeCCMapByKR(ccMap[, ...])	Normalize a ccmap using Knight-Ruiz matrix balancing method.
normalizer.normalizeGCMapByKR(...[, ...])	Normalize a gcmap using Knight-Ruiz matrix balancing method.
normalizer.normalizeCCMapByIC(ccMap[, tol, ...])	Normalize a ccmap by Iterative correction method
normalizer.normalizeGCMapByIC(...[, vmin, ...])	Normalize a gcmap using Iterative Correction.
normalizer.normalizeCCMapByMCFS(ccMap[, ...])	Scale ccmap using Median Contact Frequency
normalizer.normalizeGCMapByMCFS(...[, ...])	Scale all maps in gcmap using Median Contact Frequency
normalizer.normalizeCCMapByVCNorm(ccMap[, ...])	Normalize ccmap using Vanilla-Coverage method
normalizer.normalizeGCMapByVCNorm(...[, ...])	Normalize all maps using Vanilla-Coverage method

NormalizeKnightRuizOriginal(*ccMapObj*, *tol=1e-12*, *x0=None*, *delta=0.1*, *Delta=3*, *fl=0*)

Original Knight-Ruiz algorithm for matrix balancing

Ported from a matlab script given in the supporting information of the following paper:

- P.A. Knight and D. Ruiz (2013). A fast algorithm for matrix balancing (2013). IMA Journal of Numerical Analysis, 33, 1029-1047"
- Matrix must be symmetric and non-negative
- For input matrix A, this function find a vector X such that diag(X)*A*diag(X) is close to doubly stochastic.

Warning:

- This is original ported code and kept here for comparison and testing.
- Do not use it because for large matrix it may end up with consuming all the memory for large matrix.

Parameters **ccMapObj** (*gcMapExplorer.lib.ccmap.CCMAP*) – A CCMAP object containing observed contact frequency

Returns **normCCMap** – Normalized Contact map.

Return type CCMAP

normalizeCCMapByIC(*ccMap*, *tol=0.0001*, *vmin=None*, *vmax=None*, *outFile=None*, *iteration=500*, *percentile_threshold_no_data=None*, *threshold_data_occup=None*, *workDir=None*)

Normalize a ccmap by Iterative correction method

This method normalize the raw contact map by removing biases from experimental procedure. For more details, see this publication.

Parameters

- **ccMap** (*gcMapExplorer.lib.ccmap.CCMAP* or ccmap file.) – A CCMAP object containing observed contact frequency or a ccmap file
- **tol** (*float*) – Tolerance value. The relative increment in the results before declaring convergence.
- **vmin** (*float*) – Minimum threshold value for normalization. If contact frequency is less than or equal to this threshold value, this value is discarded during normalization.
- **vmax** (*float*) – Maximum threshold value for normalization. If contact frequency is greater than or equal to this threshold value, this value is discarded during normalization.
- **outFile** (*str*) – Name of output ccmap file, to save directly the normalized map as a ccmap file. In case of this option, `None` will return.
- **iteration** (*int*) – Number of iteration to stop the normalization.
- **percentile_threshold_no_data** (*int*) – It can be used to filter the map, where rows/columns with largest numbers of missing data can be discarded. `percentile_threshold_no_data` should be between 1 and 100. This options discard the rows and columns which are above this percentile. For example: if this value is 99, those row or columns will be discarded which contains larger than number of zeros (missing data) at 99 percentile.

  To calculate percentile, all blank rows are removed, then in all rows, number of zeros are counted. Afterwards, number of zeros at *percentile_threshold_no_data* percentile is obtained. In next step, if a row contain number of zeros larger than this percentile value, the whole row and column is assigned to have missing data. This percentile indicates highest numbers of zeros (missing data) in given rows/columns.
- **threshold_data_occup** (*float*) – It can be used to filter the map, where rows/columns with largest numbers of missing data can be discarded. This ratio is (number of bins with data) / (total number of bins in the given row/column). For example: if *threshold_data_occup = 0.8*, then all rows containing more than 20% of missing data will be discarded.

  Note that this parameter is suitable for low resolution data because maps are likely to be much less sparse.
- **workDir** (*str*) – Path to the directory where temporary intermediate files are generated. If `None`, files are generated in the temporary directory according to the main configuration.

Returns **normCCMap** – Normalized Contact map. When `outFile` is provided, `None` is returned. In case of any other error, `None` is returned.

Return type *gcMapExplorer.lib.ccmap.CCMAP* or `None`

normalizeCCMapByKR(*ccMap*, *memory='RAM'*, *tol=1e-12*, *outFile=None*, *vmin=None*, *vmax=None*, *percentile_threshold_no_data=None*, *threshold_data_occup=None*, *workDir=None*)

Normalize a ccmap using Knight-Ruiz matrix balancing method.

Note:

- This function uses a modified version of original ported code given in *NormalizeKnightRuizOriginal()*.
- **Please refer to:** P.A. Knight and D. Ruiz (2013). A fast algorithm for matrix balancing (2013). IMA Journal of Numerical Analysis, 33, 1029-1047

Parameters

- **ccMap** (*gcMapExplorer.lib.ccmap.CCMAP* or ccmap file) – A CCMAP object containing observed contact frequency or a ccmap file.
- **memory** (*str*) – Accepted keywords are `RAM` and `HDD`:
  - `RAM`: All intermediate arrays are generated in memory(RAM). This version is faster, however, it requires RAM depending on the input matrix size.
  - `HDD`: All intermediate arrays are generated as memory mapped array files on hard-disk.
- **tol** (*float*) – Tolerance for matrix balancing. Smaller tolerance increases accuracy in sums of rows and columns.
- **outFile** (*str*) – Name of output ccmap file, to save directly the normalized map as a ccmap file. In case of this option, `None` will return.
- **vmin** (*float*) – Minimum threshold value for normalization. If contact frequency is less than or equal to this threshold value, this value is discarded during normalization.
- **vmax** (*float*) – Maximum threshold value for normalization. If contact frequency is greater than or equal to this threshold value, this value is discarded during normalization.
- **percentile_threshold_no_data** (*int*) – It can be used to filter the map, where rows/columns with largest numbers of missing data can be discarded. `percentile_threshold_no_data` should be between 1 and 100. This options discard the rows and columns which are above this percentile. For example: if this value is 99, those row or columns will be discarded which contains larger than number of zeros (missing data) at 99 percentile.

  To calculate percentile, all blank rows are removed, then in all rows, number of zeros are counted. Afterwards, number of zeros at *percentile_threshold_no_data* percentile is obtained. In next step, if a row contain number of zeros larger than this percentile value, the whole row and column is assigned to have missing data. This percentile indicates highest numbers of zeros (missing data) in given rows/columns.
- **threshold_data_occup** (*float*) – It can be used to filter the map, where rows/columns with largest numbers of missing data can be discarded. This ratio is (number of bins with data) / (total number of bins in the given row/column). For example: if *threshold_data_occup = 0.8*, then all rows containing more than 20% of missing data will be discarded.

  Note that this parameter is suitable for low resolution data because maps are likely to be much less sparse.
- **workDir** (*str*) – Path to the directory where temporary intermediate files are generated. If `None`, files are generated in the temporary directory according to the main configuration.

Returns **ccMapObj** – Normalized Contact map. When `outFile` is provided, `None` is returned. In case of any other error, `None` is returned.

Return type *gcMapExplorer.lib.ccmap.CCMAP* or `None`

normalizeCCMapByMCFS(*ccMap*, *stats='median'*, *vmin=None*, *vmax=None*, *stype='o/e'*, *outFile=None*, *scaleUpInput=False*, *percentile_threshold_no_data=None*, *threshold_data_occup=None*, *workDir=None*)

Scale ccmap using Median Contact Frequency

This method can be used to normalize contact map with expected values. These expected values could be either Median or Average contact values for particular distance between two locations/coordinates. At first, Median/Average distance contact frequency for each distance is calculated. Subsequently, the observed contact frequency is either divided ('o/e') or subtracted ('o-e') by median/average contact frequency obtained for distance between the two locations.

Parameters

- **ccMap** (*gcMapExplorer.lib.ccmap.CCMAP* or ccmap file) – A CCMAP object containing observed contact frequency or a ccmap file
- **stats** (*str*) – Statistics to be calculated along diagonals: It may be either "mean" or "median". By default, it is "median".
- **vmin** (*float*) – Minimum threshold value for normalization. If contact frequency is less than or equal to this threshold value, this value is discarded during normalization.
- **vmax** (*float*) – Maximum threshold value for normalization. If contact frequency is greater than or equal to this threshold value, this value is discarded during normalization.
- **stype** (*str*) – Type of scaling. It may be either 'o/e' or 'o-e'. In case of 'o/e', Observed/Expected will be calculated while (Observed - Expected) will be calculated for 'o-e'.
- **outFile** (*str*) – Name of output ccmap file, to save directly the normalized map as a ccmap file. In case of this option, `None` will return.
- **scaleUpInput** (*bool*) – Scale up the input map by multiplying it with constant value. This constant value is precision of minimum value multiplied by 10. This scale up changes the minimum value to a integer value and accordingly whole map is changed. It is beneficial when input map contains very small value as generated from KR normalization.
- **percentile_threshold_no_data** (*int*) – It can be used to filter the map, where rows/columns with largest numbers of missing data can be discarded. `percentile_threshold_no_data` should be between 1 and 100. This options discard the rows and columns which are above this percentile. For example: if this value is 99, those row or columns will be discarded which contains larger than number of zeros (missing data) at 99 percentile.

  To calculate percentile, all blank rows are removed, then in all rows, number of zeros are counted. Afterwards, number of zeros at *percentile_threshold_no_data* percentile is obtained. In next step, if a row contain number of zeros larger than this percentile value, the whole row and column is assigned to have missing data. This percentile indicates highest numbers of zeros (missing data) in given rows/columns.
- **threshold_data_occup** (*float*) – It can be used to filter the map, where rows/columns with largest numbers of missing data can be discarded. This ratio is (number of bins with data) / (total number of bins in the given row/column). For example: if *threshold_data_occup = 0.8*, then all rows containing more than 20% of missing data will be discarded.

  Note that this parameter is suitable for low resolution data because maps are likely to be much less sparse.
- **workDir** (*str*) – Path to the directory where temporary intermediate files are generated. If `None`, files are generated in the temporary directory according to the main configuration.

Returns **ccMapObj** – Normalized Contact map. When `outFile` is provided, `None` is returned. In case of any other error, `None` is returned.

Return type *gcMapExplorer.lib.ccmap.CCMAP* or `None`

normalizeCCMapByVCNorm(*ccMap*, *sqroot=False*, *vmin=None*, *vmax=None*, *outFile=None*, *percentile_threshold_no_data=None*, *threshold_data_occup=None*, *workDir=None*)

Normalize ccmap using Vanilla-Coverage method

This method was first used in ' Lieberman-Aiden et al., 2009 <http://dx.doi.org/10.1126/science.1181369>'_ for inter-chromosomal map. Later it was used for intra-chromosomal map by Rao et al., 2014.

Parameters

- **ccMap** (*gcMapExplorer.lib.ccmap.CCMAP* or ccmap file) – A CCMAP object containing observed contact frequency or a ccmap file
- **sqroot** (*bool*) – If `True`, square-root of normalized map is calculated.
- **vmin** (*float*) – Minimum threshold value for normalization. If contact frequency is less than or equal to this threshold value, this value is discarded during normalization.
- **vmax** (*float*) – Maximum threshold value for normalization. If contact frequency is greater than or equal to this threshold value, this value is discarded during normalization.
- **outFile** (*str*) – Name of output ccmap file, to save directly the normalized map as a ccmap file. In case of this option, `None` will return.
- **percentile_threshold_no_data** (*int*) – It can be used to filter the map, where rows/columns with largest numbers of missing data can be discarded. `percentile_threshold_no_data` should be between 1 and 100. This options discard the rows and columns which are above this percentile. For example: if this value is 99, those row or columns will be discarded which contains larger than number of zeros (missing data) at 99 percentile.

  To calculate percentile, all blank rows are removed, then in all rows, number of zeros are counted. Afterwards, number of zeros at *percentile_threshold_no_data* percentile is obtained. In next step, if a row contain number of zeros larger than this percentile value, the whole row and column is assigned to have missing data. This percentile indicates highest numbers of zeros (missing data) in given rows/columns.
- **threshold_data_occup** (*float*) – It can be used to filter the map, where rows/columns with largest numbers of missing data can be discarded. This ratio is (number of bins with data) / (total number of bins in the given row/column). For example: if *threshold_data_occup = 0.8*, then all rows containing more than 20% of missing data will be discarded.

  Note that this parameter is suitable for low resolution data because maps are likely to be much less sparse.
- **workDir** (*str*) – Path to the directory where temporary intermediate files are generated. If `None`, files are generated in the temporary directory according to the main configuration.

Returns **ccMapObj** – Normalized Contact map. When `outFile` is provided, `None` is returned. In case of any other error, `None` is returned.

Return type *gcMapExplorer.lib.ccmap.CCMAP* or `None`

normalizeGCMapByIC(*gcMapInputFile*, *gcMapOutFile*, *vmin=None*, *vmax=None*, *tol=1e-12*, *iteration=500*, *percentile_threshold_no_data=None*, *threshold_data_occup=None*, *compression='lzf'*, *workDir=None*, *logHandler=None*)

Normalize a gcmap using Iterative Correction.

This method normalize the raw contact map by removing biases from experimental procedure. For more details, see this publication.

Parameters

- **gcMapInputFile** (*str*) – Name of input gcmap file.
- **gcMapOutFile** (*str*) – Name of output gcmap file.
- **vmin** (*float*) – Minimum threshold value for normalization. If contact frequency is less than or equal to this threshold value, this value is discarded during normalization.

- **vmax** (*float*) – Maximum threshold value for normalization. If contact frequency is greater than or equal to this threshold value, this value is discarded during normalization.
- **tol** (*float*) – Tolerance value. The relative increment in the results before declaring convergence.
- **iteration** (*int*) – Number of iteration to stop the normalization.
- **percentile_threshold_no_data** (*int*) – It can be used to filter the map, where rows/columns with largest numbers of missing data can be discarded. `percentile_threshold_no_data` should be between 1 and 100. This options discard the rows and columns which are above this percentile. For example: if this value is 99, those row or columns will be discarded which contains larger than number of zeros (missing data) at 99 percentile.

  To calculate percentile, all blank rows are removed, then in all rows, number of zeros are counted. Afterwards, number of zeros at *percentile_threshold_no_data* percentile is obtained. In next step, if a row contain number of zeros larger than this percentile value, the whole row and column is assigned to have missing data. This percentile indicates highest numbers of zeros (missing data) in given rows/columns.
- **threshold_data_occup** (*float*) – It can be used to filter the map, where rows/columns with largest numbers of missing data can be discarded. This ratio is (number of bins with data) / (total number of bins in the given row/column). For example: if *threshold_data_occup = 0.8*, then all rows containing more than 20% of missing data will be discarded.

  Note that this parameter is suitable for low resolution data because maps are likely to be much less sparse.
- **compression** (*str*) – Compression method in output gcmap file. Presently allowed : `lzf` for LZF compression and `gzip` for GZIP compression.
- **workDir** (*str*) – Path to the directory where temporary intermediate files are generated. If `None`, files are generated in the temporary directory according to the main configuration.

Returns

Return type None

See also:

gcMapExplorer.lib.normalizer.normalizeCCMapByIC()

normalizeGCMapByKR(*gcMapInputFile*, *gcMapOutFile*, *mapSizeCeilingForMemory=20000*, *vmin=None*, *vmax=None*, *tol=1e-12*, *percentile_threshold_no_data=None*, *threshold_data_occup=None*, *compression='lzf'*, *workDir=None*, *logHandler=None*)

Normalize a gcmap using Knight-Ruiz matrix balancing method.

Parameters

- **gcMapInputFile** (*str*) – Name of input gcmap file.
- **gcMapOutFile** (*str*) – Name of output gcmap file.
- **mapSizeCeilingForMemory** (*int*) – Maximum size of contact map allowed for calculation using RAM. If map size or shape is larger than this value, normalization will be performed using disk (HDD).
- **vmin** (*float*) – Minimum threshold value for normalization. If contact frequency is less than or equal to this threshold value, this value is discarded during normalization.
- **vmax** (*float*) – Maximum threshold value for normalization. If contact frequency is greater than or equal to this threshold value, this value is discarded during normalization.

- **tol** (*float*) – Tolerance for matrix balancing. Smaller tolerance increases accuracy in sums of rows and columns.
- **percentile_threshold_no_data** (*int*) – It can be used to filter the map, where rows/columns with largest numbers of missing data can be discarded. `percentile_threshold_no_data` should be between 1 and 100. This options discard the rows and columns which are above this percentile. For example: if this value is 99, those row or columns will be discarded which contains larger than number of zeros (missing data) at 99 percentile.

  To calculate percentile, all blank rows are removed, then in all rows, number of zeros are counted. Afterwards, number of zeros at *percentile_threshold_no_data* percentile is obtained. In next step, if a row contain number of zeros larger than this percentile value, the whole row and column is assigned to have missing data. This percentile indicates highest numbers of zeros (missing data) in given rows/columns.
- **threshold_data_occup** (*float*) – It can be used to filter the map, where rows/columns with largest numbers of missing data can be discarded. This ratio is (number of bins with data) / (total number of bins in the given row/column). For example: if *threshold_data_occup = 0.8*, then all rows containing more than 20% of missing data will be discarded.

  Note that this parameter is suitable for low resolution data because maps are likely to be much less sparse.
- **compression** (*str*) – Compression method in output gcmap file. Presently allowed : `lzf` for LZF compression and `gzip` for GZIP compression.
- **workDir** (*str*) – Path to the directory where temporary intermediate files are generated. If `None`, files are generated in the temporary directory according to the main configuration.

Returns

Return type None

See also:

gcMapExplorer.lib.normalizer.normalizeCCMapByKR()

normalizeGCMapByMCFS(*gcMapInputFile*, *gcMapOutFile*, *stats='median'*, *vmin=None*, *vmax=None*, *stype='o/e'*, *scaleUpInput=False*, *percentile_threshold_no_data=None*, *threshold_data_occup=None*, *compression='lzf'*, *workDir=None*, *logHandler=None*)

Scale all maps in gcmap using Median Contact Frequency

This method can be used to normalize contact map with expected values. These expected values could be either Median or Average contact values for particular distance between two locations/coordinates. At first, Median/Average distance contact frequency for each distance is calculated. Subsequently, the observed contact frequency is either divided ('o/e') or subtracted ('o-e') by median/average contact frequency obtained for distance between the two locations.

Parameters

- **gcMapInputFile** (*str*) – Name of input gcmap file.
- **gcMapOutFile** (*str*) – Name of output gcmap file.
- **stats** (*str*) – Statistics to be calculated along diagonals: It may be either "mean" or "median". By default, it is "median".
- **vmin** (*float*) – Minimum threshold value for normalization. If contact frequency is less than or equal to this threshold value, this value is discarded during normalization.

- **vmax** (*float*) – Maximum threshold value for normalization. If contact frequency is greater than or equal to this threshold value, this value is discarded during normalization.
- **stype** (*str*) – Type of scaling. It may be either 'o/e' or 'o-e'. In case of 'o/e', Observed/Expected will be calculated while (Observed - Expected) will be calculated for 'o-e'.
- **scaleUpInput** (*bool*) – Scale up the input map by multiplying it with constant value. This constant value is precision of minimum value multiplied by 10. This scale up changes the minimum value to a integer value and accordingly whole map is changed. It is beneficial when input map contains very small value as generated from KR normalization.
- **percentile_threshold_no_data** (*int*) – It can be used to filter the map, where rows/columns with largest numbers of missing data can be discarded. `percentile_threshold_no_data` should be between 1 and 100. This options discard the rows and columns which are above this percentile. For example: if this value is 99, those row or columns will be discarded which contains larger than number of zeros (missing data) at 99 percentile.

  To calculate percentile, all blank rows are removed, then in all rows, number of zeros are counted. Afterwards, number of zeros at *percentile_threshold_no_data* percentile is obtained. In next step, if a row contain number of zeros larger than this percentile value, the whole row and column is assigned to have missing data. This percentile indicates highest numbers of zeros (missing data) in given rows/columns.
- **threshold_data_occup** (*float*) – It can be used to filter the map, where rows/columns with largest numbers of missing data can be discarded. This ratio is (number of bins with data) / (total number of bins in the given row/column). For example: if *threshold_data_occup = 0.8*, then all rows containing more than 20% of missing data will be discarded.

  Note that this parameter is suitable for low resolution data because maps are likely to be much less sparse.
- **compression** (*str*) – Compression method in output gcmap file. Presently allowed : `lzf` for LZF compression and `gzip` for GZIP compression.
- **workDir** (*str*) – Path to the directory where temporary intermediate files are generated. If `None`, files are generated in the temporary directory according to the main configuration.

Returns

Return type None

See also:

gcMapExplorer.lib.normalizer.normalizeCCMapByMCFS()

normalizeGCMapByVCNorm(*gcMapInputFile*, *gcMapOutFile*, *sqroot=False*, *vmin=None*, *vmax=None*, *percentile_threshold_no_data=None*, *threshold_data_occup=None*, *compression='lzf'*, *workDir=None*, *logHandler=None*)

Normalize all maps using Vanilla-Coverage method

This method was first used in ' Lieberman-Aiden et al., 2009 <http://dx.doi.org/10.1126/science.1181369>'_ for inter-chromosomal map. Later it was used for intra-chromosomal map by Rao et al., 2014.

Parameters

- **gcMapInputFile** (*str*) – Name of input gcmap file.
- **gcMapOutFile** (*str*) – Name of output gcmap file.
- **sqroot** (*bool*) – If `True`, square-root of normalized map is calculated.

- **vmin** (*float*) – Minimum threshold value for normalization. If contact frequency is less than or equal to this threshold value, this value is discarded during normalization.
- **vmax** (*float*) – Maximum threshold value for normalization. If contact frequency is greater than or equal to this threshold value, this value is discarded during normalization.
- **scaleUpInput** (*bool*) – Scale up the input map by multiplying it with constant value. This constant value is precision of minimum value multiplied by 10. This scale up changes the minimum value to a integer value and accordingly whole map is changed. It is beneficial when input map contains very small value as generated from KR normalization.
- **percentile_threshold_no_data** (*int*) – It can be used to filter the map, where rows/columns with largest numbers of missing data can be discarded. `percentile_threshold_no_data` should be between 1 and 100. This options discard the rows and columns which are above this percentile. For example: if this value is 99, those row or columns will be discarded which contains larger than number of zeros (missing data) at 99 percentile.

  To calculate percentile, all blank rows are removed, then in all rows, number of zeros are counted. Afterwards, number of zeros at *percentile_threshold_no_data* percentile is obtained. In next step, if a row contain number of zeros larger than this percentile value, the whole row and column is assigned to have missing data. This percentile indicates highest numbers of zeros (missing data) in given rows/columns.
- **threshold_data_occup** (*float*) – It can be used to filter the map, where rows/columns with largest numbers of missing data can be discarded. This ratio is (number of bins with data) / (total number of bins in the given row/column). For example: if *threshold_data_occup = 0.8*, then all rows containing more than 20% of missing data will be discarded.

  Note that this parameter is suitable for low resolution data because maps are likely to be much less sparse.
- **compression** (*str*) – Compression method in output gcmap file. Presently allowed : `lzf` for LZF compression and `gzip` for GZIP compression.
- **workDir** (*str*) – Path to the directory where temporary intermediate files are generated. If `None`, files are generated in the temporary directory according to the main configuration.

Returns

Return type None

See also:

gcMapExplorer.lib.normalizer.normalizeCCMapByMCFS()

cmstats module

cmstats.correlateCMaps(ccMapObjOne, ccMapObjTwo)	To calculate correlation between two Hi-C maps
cmstats.correlateGCMaps(gcmapOne, gcmapTwo)	To calculate correlation between common Hi-C maps from two gcmap files
cmstats.getAvgContactByDistance(ccmaps[, ...])	To calculate average contact as a function of distance

correlateCMaps(*ccMapObjOne*, *ccMapObjTwo*, *ignore_triangular=True*, *diagonal_offset=1*, *corrType='pearson'*, *blockSize=None*, *slideStepSize=1*, *cutoffPercentile=None*, *workDir=None*, *outFile=None*, *logHandler=None*)

To calculate correlation between two Hi-C maps

This function can be used to calculate either Pearson or Spearman rank-order correlation between two Hi-C maps. It also ignore lower-triangular matrix with diagnonal offset to avoid duplicate and large values.

Parameters

- **ccMapObjOne** (*gcMapExplorer.lib.ccmap.CCMAP*) – First *gcMapExplorer.lib.ccmap.CCMAP* instance containing Hi-C data
- **ccMapObjTwo** (*gcMapExplorer.lib.ccmap.CCMAP*) – Second *gcMapExplorer.lib.ccmap.CCMAP* instance containing Hi-C data
- **ignore_triangular** (*bool*) – Whether entire matrix is considered or only one half triangular region of matrixis considered.
- **diagonal_offset** (*int*) – If `ignore_triangular=True`, it is used to determine how much bins are ignored from the diagonal in one half triangular region of matrix. `diagonal_offset = 0` is the main diagonal, `diagonal_offset > 0` means ignore this many bins from the diagonal.
- **corrType** (*str*) – Correlation type. For Pearson and Spearman rank-order correlation, use `pearson` and `spearman`, respectively.
- **blockSize** (*str*) – To calculate block-wise correlations by sliding block of given size along diagonals. It should be in resolution. For example, `1mb`, `500kb`, `5mb`, `2.5mb` etc. If `None`, correlation of whole map is calculated. Sliding step of block depends on `slideStepSize`.
- **slideStepSize** (*int*) – Step-size in bins by which blocks will be shifted for block-wise correlation. If slideStepSize is large then blocks might not be overlapped.
- **cutoffPercentile** (*float*) – Cutoff percentile to discard values during correlation calculation. If a cutoff percentile is given, values less than this percentile value will not be considered during correlation calculation.
- **workDir** (*str*) – Name of working directory, where temporary files will be kept.If `workDir = None`, file will be generated in OS based temporary directory.
- **outFile** (*str*) – Name of output file. Only written for block-wise correlation.

Returns

- **corr** (*float or list*) – Correlation coefficient
- **pvalue/centers** (*float or list*) – If `blockSize=None` 2-tailed p-value is returned. For block-wise correlation, list of block-center is returned.

See also:

- scipy.stats.pearsonr for Pearson correlation.
- scipy.stats.spearmanr for Spearman rank-order correlation.

correlateGCMaps(*gcmapOne*, *gcmapTwo*, *outFile=None*, *blockSize=None*, *slideStepSize=1*, *name=None*, *cutoffPercentile=None*, *ignore_triangular=True*, *diagonal_offset=1*, *corrType='pearson'*, *workDir=None*, *logHandler=None*)

To calculate correlation between common Hi-C maps from two gcmap files

This function can be used to calculate either Pearson or Spearman rank-order correlation between common maps present in two gcmap files. It also ignore lower-triangular matrix with diagonal offset to avoid duplicate and large values.

Note: If block-wise correlation calculation will be initiated by `blockSize` option, a `outFile` and `name` is also required for further processing. The block-wise correlation will stored in output HDF5 format file.

Parameters

- **gcmapOne** (*str*) – First gcmap file.
- **gcmapTwo** (*str*) – Second gcmap file
- **outFile** (*str*) – Name of output file. Only written for block-wise correlation.
- **blockSize** (*str*) – To calculate block-wise correlations by sliding block of given size along diagonals. It should be in resolution. For example, `1mb`, `500kb`, `5mb`, `2.5mb` etc. If `None`, correlation of whole map is calculated. Sliding step of block depends on `slideStepSize`.
- **slideStepSize** (*int*) – Step-size in bins by which blocks will be shifted for block-wise correlation. If slideStepSize is large then blocks might not be overlapped.
- **name** (*str*) – Title of dataset in HDF5 output file. If `blockSize` option is used, `name` is an essential argument.
- **cutoffPercentile** (*float*) – Cutoff percentile to discard values during correlation calculation. If a cutoff percentile is given, values less than this percentile value will not be considered during correlation calculation.
- **ignore_triangular** (*bool*) – Whether entire matrix is considered or only one half triangular region of matrixis considered.
- **diagonal_offset** (*int*) – If `ignore_triangular=True`, it is used to determine how much bins are ignored from the diagonal in one half triangular region of matrix. `diagonal_offset = 0` is the main diagonal, `diagonal_offset > 0` means ignore this many bins from the diagonal.
- **corrType** (*str*) – Correlation type. For Pearson and Spearman rank-order correlation, use `pearson` and `spearman`, respectively.
- **workDir** (*str*) – Name of working directory, where temporary files will be kept.If `workDir = None`, file will be generated in OS based temporary directory.

Returns

- **mapList** (*list*) – List of chromosomes
- **corrs** (*list*) – Correlation coefficient of each chromosome
- **pvalue** (*list*) – 2-tailed p-value for correlation coefficient of each chromosome.

getAvgContactByDistance(*ccmaps*, *stats='median'*, *removeOutliers=False*, *outliersThreshold=3.5*)

To calculate average contact as a function of distance

Parameters

- **ccmaps** (*gcMapExplorer.lib.ccmap.CCMAP* or list[*gcMapExplorer.lib.ccmap.CCMAP*]) – Input contact maps
- **stats** (*str*) – Statistics for scaling. Accepted methods are `mean` and `median`.

- **removeOutliers** (*bool*) – If `True`, outliers will be removed before calculating input statistics.
- **outliersThreshold** (*float*) – The modified z-score to use as a threshold. Observations with a modified z-score (based on the median absolute deviation) greater than this value will be classified as outliers.

Returns

- **avg_contacts** (*numpy.array*)
- *A one-dimensional numpy array containing average contacts, where index is distance between two locations for given resolution/binsize.*
- For example, if `ccmap.binsize=100000` and `avg_contacts[4]=1234.56`, then at distance of 400000 b, average contact is `1234.56`.

statDist module

This module contains functions and methods for calculating stationary distribution by assuming markov-chain.

This module contains functions to calculate probability transition matrix and subsequently to calculate stationary distribution.

Example

Below is a simple example to calculate stationary distribution. It consists of three major steps:

- Calculate median-subtracted normalized matrix
- Calculate probability transition matrix
- Calculate stationary distribution

```
# median-subtracted normalized matrix, stype should be 'o-e', Other arguments can be␣
↪changed.
gmlib.normalizer.normalizeGCMapByMCFS('input_raw.gcmap', 'mcfs_O-E.gcmap',␣
↪stats='median', stype='o-e')

# Probability transition matrix, calculate matrix at '40kb' resolution.
gmlib.statDist.transitionProbabilityMatrixForGCMap('mcfs_O-E.gcmap', 'prob_mat.gcmap',
↪ '40kb')

# Stationary distribution at '40kb' resolution
gmlib.statDist.statDistrByEigenDecompForGCMap('out_prob_mat.gcmap', 'stat_dist.h5',␣
↪'40kb')
```

Summary

statDist.calculateTransitionProbabilityMatrix(A)	Core function to calculate transition probability matrix.
statDist.transitionProbabilityMatrixForCCMap(ccMap)	To calculate transition probability matrix.
statDist.transitionProbabilityMatrixForGCMap()	To calculate transition matrices using a gcmap file.
statDist.statDistrByEigenDecompForCCMap(ccMap)	Calculate stationary distribution from a ccmap file or object.

Continued on next page

Table 25 – continued from previous page

statDist.statDistrByEigenDecompForGCMap()	Calculate stationary distribution using transition matrices from gcmap file for given resolution.
statDist.stationaryDistributionByEigenDecomp()	To calculate stationary distribution from probability transition matrix.

Documentation

MatrixPowerRAM(*M*, *n*)

Raise a square matrix to the (integer) power *n*.

Note: Ported from numpy/matrixlib/defmatrix.py. This function was rewritten to speed-up the calculation. In place of numpy dot function, a mdot function is used, which directly uses blas dgemm function.

For positive integers *n*, the power is computed by repeated matrix squaring and matrix multiplications. If `n == 0`, the identity matrix of the same shape as M is returned. If `n < 0`, the inverse is computed and then raised to the `abs(n)`.

Parameters

- **M** (*ndarray or matrix object*) – Matrix to be "powered." Must be square, i.e. `M.shape == (m, m)`, with *m* a positive integer.
- **n** (*int*) – The exponent can be any integer or long integer, positive, negative, or zero.

Returns **M**n** – The return value is the same shape and type as *M*; if the exponent is positive or zero then the type of the elements is the same as those of *M*. If the exponent is negative the elements are floating-point.

Return type ndarray or matrix object

Raises `LinAlgError` – If the matrix is not numerically invertible.

calculateTransitionProbabilityMatrix(*A*, *Out=None*, *bNoData=None*)

Core function to calculate transition probability matrix.

It is a core function to calculate transition probability matrix.

Parameters

- **A** (numpy.memmap or *gcMapExplorer.lib.ccmap.CCMAP.matrix* or numpy.ndarray) – Input map or matrix
- **Out** (numpy.memmap or *gcMapExplorer.lib.ccmap.CCMAP.matrix* or numpy.ndarray) – Ouput transition matrix. In case if it is `None`, ouput matrix will be returned.
- **bNoData** (*numpy.array[bool]*) – 1D-array containing `True` and `False` values. Its size should be equal to input array row/column size. Row/Column with `False` value will be considered during the calculation.

Returns **Out** – Ouput transition matrix. In case if `Out` is passed, `None` will be returned.

Return type numpy.ndarray or `None`.

statDistrByEigenDecompForCCMap(*ccMap*, *chrom=None*, *hdf5Handle=None*, *compression='lzf'*, *workDir=None*)

Calculate stationary distribution from a ccmap file or object.

It uses eigendecomposition method to calculate stationary distribution from transition matrix.

Parameters

- **ccMap** (*gcMapExplorer.lib.ccmap.CCMAP* or ccmap file) – A CCMAP object containing observed contact frequency or a ccmap file
- **chrom** (*str*) – Name of chromosome – necessary as used in output HDF5 file.
- **hdf5Handle** (*gcMapExplorer.lib.genomicsDataHandler.HDF5Handler*) – If it is provided, stationary distribution will be directly added to this file. In case of `None`, stationary distribution will be returned as a 1D array.
- **compression** (*str*) –

  Compression method in output HDF5 file. Presently allowed `[lzf]` for LZF compression and `gzip` for GZIP compression.
- **workDir** (*str*) – Path to the directory where temporary intermediate files are generated. If `None`, files are generated in the temporary directory according to the main configuration.

Returns **sdist** – If `hdf5Handle` is provided `None` is returned otherwise stationary distribution as 1D array is returned.

Return type numpy.1darray or `None`

statDistrByEigenDecompForGCMap(*gcMapInputFile*, *outFile*, *resolution*, *overwrite=False*, *compression='lzf'*, *workDir=None*)

Calculate stationary distribution using transition matrices from gcmap file for given resolution.

It uses eigendecomposition method to calculate stationary distribution from transition matrix.

Note: Use same input resolution as used during calculation of transition matrix using gcmap file.

Parameters

- **gcMapInputFile** (*str*) – Name of input gcmap file.
- **outFile** (*str*) – Name of output HDF5 file to store calculated stationary distribution.
- **resolution** (*str*) – Input resolution at which transition matrix was calculated. And stationary distribution will be calculated.
- **compression** (*str*) – Compression method in output HDF5 file. Presently allowed : `lzf` for LZF compression and `gzip` for GZIP compression.
- **workDir** (*str*) – Path to the directory where temporary intermediate files are generated. If `None`, files are generated in the temporary directory according to the main configuration.

stationaryDistributionByEigenDecomp(*prob_matrix*, *bNoData*)

To calculate stationary distribution from probability transition matrix.

Stationary distribution is calculated using probability transition matrix with eigendecomposition.

Parameters

- **prob_matrix** (numpy.memmap or *gcMapExplorer.lib.ccmap.CCMAP.matrix* or numpy.ndarray) – Input probability transition matrix
- **bNoData** (*numpy.array[bool]*) – 1D-array containing `True` and `False` values. Its size should be equal to input matrix row/column size. Row/Column with `False` value will be considered during the calculation. Row/Column with `True` value will not be considered during calculation and in these locations, minimum stationary distribution will be filled in the output.

Returns **sdist** – Ouput stationary distribution as 1D numpy array.

Return type numpy.ndarray

transitionProbabilityMatrixForCCMap(*ccMap*, *outFile=None*, *percentile_threshold_no_data=None*, *threshold_data_occup=None*, *workDir=None*)

To calculate transition probability matrix.

This method can be used to calculate transition probability matrix. This is similar to markov-chain transition matrix.

Note This transition matrix is not symmetric, because each row represents stochastic row vector, which contains contact probability of this bin with every other bins and sum of row is always equal to one. See here: .

Parameters

- **ccMap** (*gcMapExplorer.lib.ccmap.CCMAP* or ccmap file) – A CCMAP object containing observed contact frequency or a ccmap file
- **outFile** (*str*) – Name of output ccmap file, to save directly the map as a ccmap file. In case of this option, `None` will return.
- **percentile_threshold_no_data** (*int*) – It can be used to filter the map, where rows/columns with largest numbers of missing data can be discarded. `percentile_threshold_no_data` should be between 1 and 100. This options discard the rows and columns which are above this percentile. For example: if this value is 99, those row or columns will be discarded which contains larger than number of zeros (missing data) at 99 percentile.

  To calculate percentile, all blank rows are removed, then in all rows, number of zeros are counted. Afterwards, number of zeros at *percentile_threshold_no_data* percentile is obtained. In next step, if a row contain number of zeros larger than this percentile value, the whole row and column is assigned to have missing data. This percentile indicates highest numbers of zeros (missing data) in given rows/columns.
- **threshold_data_occup** (*float*) – It can be used to filter the map, where rows/columns with largest numbers of missing data can be discarded. This ratio is (number of bins with data) / (total number of bins in the given row/column). For example: if *threshold_data_occup = 0.8*, then all rows containing more than 20% of missing data will be discarded.

  Note that this parameter is suitable for low resolution data because maps are likely to be much less sparse.

Returns **normCCMap** – Transition matrix. When `outFile` is provided, `None` is returned. In case of any other error, `None` is returned.

Return type *gcMapExplorer.lib.ccmap.CCMAP* or `None`

transitionProbabilityMatrixForGCMap(*gcMapInputFile*, *gcMapOutFile*, *resolution*, *compression='lzf'*, *percentile_threshold_no_data=None*, *threshold_data_occup=None*, *workDir=None*, *logHandler=None*)

To calculate transition matrices using a gcmap file.

It can be used to calculate transition matrices (markov-chain) for all maps present in a gcmap file for given resolution.

Note: Matrices will be calculated for only input resolution. For coarser resolutions, data will be downsampled, and therefore in output gcmap, only matrices corresponding to input resolution is correct. Other coarsed resolutionsm matrices are only for visualization purpose.

Parameters

- **gcMapInputFile** (*str*) – Name of input gcmap file.
- **gcMapOutFile** (*str*) – Name of output gcmap file.
- **resolution** (*str*) – Input resolution at which transition matrix will be calculated.
- **compression** (*str*) – Compression method in output gcmap file. Presently allowed : `lzf` for LZF compression and `gzip` for GZIP compression.
- **percentile_threshold_no_data** (*int*) – It can be used to filter the map, where rows/columns with largest numbers of missing data can be discarded. `percentile_threshold_no_data` should be between 1 and 100. This options discard the rows and columns which are above this percentile. For example: if this value is 99, those row or columns will be discarded which contains larger than number of zeros (missing data) at 99 percentile.

  To calculate percentile, all blank rows are removed, then in all rows, number of zeros are counted. Afterwards, number of zeros at *percentile_threshold_no_data* percentile is obtained. In next step, if a row contain number of zeros larger than this percentile value, the whole row and column is assigned to have missing data. This percentile indicates highest numbers of zeros (missing data) in given rows/columns.
- **threshold_data_occup** (*float*) – It can be used to filter the map, where rows/columns with largest numbers of missing data can be discarded. This ratio is (number of bins with data) / (total number of bins in the given row/column). For example: if *threshold_data_occup = 0.8*, then all rows containing more than 20% of missing data will be discarded.

  Note that this parameter is suitable for low resolution data because maps are likely to be much less sparse.
- **workDir** (*str*) – Path to the directory where temporary intermediate files are generated. If `None`, files are generated in the temporary directory according to the main configuration.

corrMatrix module

corrMatrix.calculateCorrMatrix(... [, maskvalue])	Calculate correlation matrix from a 2D numpy array.
corrMatrix.calculateCovMatrix(... [, maskvalue])	Calculate covariance matrix from a 2D numpy array.
corrMatrix.calculateCorrelation(ndarray x, ...)	Calculate correlation between two 1D numpy array.
corrMatrix.calculateCovariance(ndarray x, ...)	Calculate covariance between two 1D numpy array.
corrMatrix.calculateCorrMatrixForCCMap(...)	Calculate correlation matrix of a contact map.
corrMatrix.calculateCorrMatrixForGCMaps(...)	Calculate Correlation matrix for all maps present in input gcmap file It calculates correlation between all rows and columns of contact map.

calculateCorrMatrix(*ndarray in_array*, *ndarray out=None*, *maskvalue=None*)

Calculate correlation matrix from a 2D numpy array. During calculation, array values equal to maskvalue will not be considered.

Parameters

- **in_array** (*numpy.ndarray*) – Input numpy array
- **out** (*numpy.ndarray*) – If it is None, output array is returned.
- **maskvalue** (*float*) – If this value is given, all elements of input array with this equal value is masked during the calculation.

Returns **out** – In case of out=None, ouput array will be returned. Otherwise, None is returned

Return type numpy.ndarray or None

calculateCovMatrix(*ndarray in_array*, *ndarray out=None*, *maskvalue=None*)

Calculate covariance matrix from a 2D numpy array. During calculation, array values equal to maskvalue will not be considered.

Parameters

- **in_array** (*numpy.ndarray*) – Input numpy array
- **out** (*numpy.ndarray*) – If it is None, output array is returned.
- **maskvalue** (*float*) – If this value is given, all elements of input array with this equal value is masked during the calculation.

Returns **out** – In case of out=None, ouput array will be returned. Otherwise, None is returned

Return type numpy.ndarray or None

calculateCorrelation(*ndarray x*, *ndarray y*, *maskvalue=None*)

Calculate correlation between two 1D numpy array. During calculation, array values equal to maskvalue will not be considered.

Parameters

- **x** (*numpy.ndarray*) – First input numpy array
- **y** (*numpy.ndarray*) – Second input numpy array
- **maskvalue** (*float*) – If this value is given, all elements of input array with this equal value is masked during the calculation.

Returns **result** – Correlation coefficient

Return type float

calculateCovariance(*ndarray x*, *ndarray y*, *maskvalue=None*)

Calculate covariance between two 1D numpy array. During calculation, array values equal to maskvalue will not be considered.

Parameters

- **x** (*numpy.ndarray*) – First input numpy array
- **y** (*numpy.ndarray*) – Second input numpy array
- **maskvalue** (*float*) – If this value is given, all elements of input array with this equal value is masked during the calculation.

Returns **result** – Covariance value

Return type float

calculateCorrMatrixForCCMap(*inputCCMap*, *logspace=False*, *maskvalue=0.0*, *vmin=None*, *vmax=None*, *outFile=None*, *workDir=None*)

Calculate correlation matrix of a contact map. It calculates correlation between all rows and columns of contact map.

Parameters

- **ccMap** (*gcMapExplorer.lib.ccmap.CCMAP* or ccmap file) – A CCMAP object containing observed contact frequency or a ccmap file.
- **logspace** (*bool*) – If its value is `True`, at first map is converted as logarithm of map and subsequently correlation will be calculated.
- **maskvalue** (*float*) – Do not consider bins with this value during calculation. By default here it is zero because bins with zero is considered to be have missing data.
- **vmin** (*float*) – Minimum threshold value for normalization. If contact frequency is less than or equal to this threshold value, this value is discarded during normalization.
- **vmax** (*float*) – Maximum threshold value for normalization. If contact frequency is greater than or equal to this threshold value, this value is discarded during normalization.
- **outFile** (*str*) – Name of output ccmap file, to save directly the correlation matrix as a ccmap file. In case of this option, `None` will return.
- **workDir** (*str*) – Path to the directory where temporary intermediate files are generated. If `None`, files are generated in the temporary directory according to the main configuration.

Returns **ccMapObj** – Normalized Contact map. When `outFile` is provided, `None` is returned. In case of any other error, `None` is returned.

Return type *gcMapExplorer.lib.ccmap.CCMAP* or `None`

calculateCorrMatrixForGCMaps(*gcMapInputFile*, *gcMapOutFile*, *logspace=False*, *maskvalue=0.0*, *vmin=None*, *vmax=None*, *replaceMatrix=False*, *compression='lzf'*, *workDir=None*)

Calculate Correlation matrix for all maps present in input gcmap file It calculates correlation between all rows and columns of contact map.

Parameters

- **gcMapInputFile** (*str*) – Input gcmap file.
- **gcMapOutFile** (*str*) – Output gcmap file.
- **logspace** (*bool*) – If its value is `True`, at first map is converted as logarithm of map and subsequently correlation will be calculated.
- **maskvalue** (*float*) – Do not consider bins with this value during calculation. By default here it is zero because bins with zero is considered to be have missing data.
- **vmin** (*float*) – Minimum threshold value for normalization. If contact frequency is less than or equal to this threshold value, this value is discarded during normalization.
- **vmax** (*float*) – Maximum threshold value for normalization. If contact frequency is greater than or equal to this threshold value, this value is discarded during normalization.
- **replaceMatrix** (*bool*) – If its value is `True`, the map will be replaced in output file. Otherwise, if a map is present, calculation will be skipped.
- **compression** (*str*) – Compression method in output gcmap file. Presently allowed : `lzf` for LZF compression and `gzip` for GZIP compression.
- **workDir** (*str*) – Path to the directory where temporary intermediate files are generated. If `None`, files are generated in the temporary directory according to the main configuration.

Returns

Return type None

genomicsDataHandler module

This module is developed to visualize and analyze Genomics data with respect to Hi-C maps. This module contains method to convert bigWig and Wig file to hdf5 file.

The hdf5 file gives us flexibility to access the data for given range of location of a specific chromosome at particular resolution.

List of class

class HDF5Handler

HDF5Handler(filename[, title])	Handler for genomic data HDF5 file.
HDF5Handler.setTitle(title)	Set title of the dataset
HDF5Handler.getChromList()	To get list of all chromosomes present in hdf5 file
HDF5Handler.getResolutionList(chrom[, dataName])	To get all resolutions for given chromosome from hdf5 file
HDF5Handler.getDataNameList(chrom, resolution)	List of all available arrays by respective coarse method name for given chromosome and resolution
HDF5Handler.hasChromosome(chrom)	To get list of all chromosomes present in hdf5 file
HDF5Handler.hasResolution(chrom, resolution)	To check if given resolution for given chromosome is present
HDF5Handler.hasDataName(chrom, resolution, ...)	To check if given data for given resolution for given chromosome is present
HDF5Handler.buildDataTree()	Build data dictionary from the input hdf5 file
HDF5Handler.addDataByArray(Chrom, ...[, ...])	Add array to the hdf5 file for given chromosome, resolution and data name.

class HDF5Handler(*filename*, *title=None*)

Handler for genomic data HDF5 file.

This class acts like a handler and can be used to read, write, and modify genomic data file in HDF5 format. This is a binary file and is compressed using `zlib` method to reduce the storage memory.

Structure of HDF5 file: `/<Chromosome>/<Resolution>/<1D Numpy Array>`

```
HDF5 ──────────────────────────> title
  ├──────── chr1
  │           ├──── 1kb
  │           │       ├──────── amean  ( Arithmetic mean) (type: 1D␣
↪Numpy Array)
  │           │       ├──────── median ( Median value   ) (type: 1D␣
↪Numpy Array)
  │           │       ├──────── hmean  ( Harmonic mean  ) (type: 1D␣
↪Numpy Array)
  │           │       ├──────── gmean  ( Geometric mean ) (type: 1D␣
↪Numpy Array)
  │           │       ├──────── min    ( Minimum value  ) (type: 1D␣
↪Numpy Array)
```

(continues on next page)

(continued from previous page)

```
  │             │          └──────── max     ( Maximum value  ) (type: 1D␣
↪Numpy Array)
  │             │
  │             ├─────  5kb
  │             │          ├──────── amean   ( Arithmetic mean) (type: 1D␣
↪Numpy Array)
  │             │          ├──────── median  ( Median value   ) (type: 1D␣
↪Numpy Array)
  │             │          ├──────── hmean   ( Harmonic mean  ) (type: 1D␣
↪Numpy Array)
  │             │          ├──────── gmean   ( Geometric mean ) (type: 1D␣
↪Numpy Array)
  │             │          ├──────── min     ( Minimum value  ) (type: 1D␣
↪Numpy Array)
  │             │          └──────── max     ( Maximum value  ) (type: 1D␣
↪Numpy Array)
  │             │
  │             └─────  ...
  │
  ├──────────── chr2
  │             ├─────  1kb
  │             │          ├──────── amean   ( Arithmetic mean) (type: 1D␣
↪Numpy Array)
  │             │          ├──────── median  ( Median value   ) (type: 1D␣
↪Numpy Array)
  │             │          ├──────── hmean   ( Harmonic mean  ) (type: 1D␣
↪Numpy Array)
  │             │          ├──────── gmean   ( Geometric mean ) (type: 1D␣
↪Numpy Array)
  │             │          ├──────── min     ( Minimum value  ) (type: 1D␣
↪Numpy Array)
  │             │          └──────── max     ( Maximum value  ) (type: 1D␣
↪Numpy Array)
  │             └─────  ..
  :
  :
  :
  └─────── ...
```

filename
str – HDF5 file name

title
str – Title of the data

hdf5
h5py.File – input/output stream to HDF5 file

data
dict – This dictionary is generated by *HDF5Handler.buildDataTree()*. This dictionary gives access to all data arrays.

Parameters

- **filename** (*str*) – HDF5 file name. e.g.: `abcxyz.h5`
- **title** (*str*) – title or name of the data

Examples

```
from gcMapExplorer import lib as gmlib
import numpy as np

# Load available file
hdf5Hand = gmlib.genomicsDataHandler.HDF5Handler('abcxyz.h5')

# Open the file
hdf5Hand.open()

# Build the data structure
hdf5Hand.buildDataTree()

# Print shape and maximum value of chr1->1kb->mean array
print(hdf5Hand.data['chr1']['1kb']['amean'].shape, np.amax(hdf5Hand.
↪data['chr1']['1kb']['mean']))
```

addDataByArray (*Chrom*, *resolution*, *data_name*, *value_array*, *compression='lzf'*)

Add array to the hdf5 file for given chromosome, resolution and data name. It can be used either to add new data array or to replace existing data.

Parameters

- **Chrom** (*str*) – Chromosome Name
- **resolution** (*str*) – Resolution of data
- **data_name** (*str*) – Name of data.
- **value_array** (*numpy.ndarray*) – An array containing values.

buildDataTree ()

Build data dictionary from the input hdf5 file

To retrieve the data from hdf5 file, this function should be used to built the dictionary *HDF5Handler.data*. This dictionary gives access directly to data of any chromosome with specific resolution.

close ()

close hdf5 file

getChromList ()

To get list of all chromosomes present in hdf5 file

Returns **chroms** – List of all chromosomes present in hdf5 file

Return type list

getDataNameList (*chrom*, *resolution*)

List of all available arrays by respective coarse method name for given chromosome and resolution

Parameters

- **chrom** (*str*) – chromosome name
- **resolution** (*str*) – resolution

Returns **nameList** – List of arrays by name of dataset

Return type list[str]

Raises KeyError – If chromosome not found in hdf5 file. If input resolution keyword is not found for input chromosome.

getResolutionList(*chrom*, *dataName=None*)
To get all resolutions for given chromosome from hdf5 file

Parameters

- **chrom** (*str*) – chromosome name
- **dataName** (*str*) – Options to get list of all resolution list for given data name

Returns **resolutionList** – A list of all available resolutions for the given chromosome

Return type list[str]

Raises KeyError – If chromosome not found in hdf5 file

hasChromosome(*chrom*)
To get list of all chromosomes present in hdf5 file

Parameters **chrom** (*str*) – Chromosome name to be look up in file.

Returns **gotChromosome** – If queried chromosome present in file `True` otherwise `False`.

Return type bool

hasDataName(*chrom*, *resolution*, *dataName*)
To check if given data for given resolution for given chromosome is present

Parameters

- **chrom** (*str*) – Chromosome name to be look up in file.
- **resolution** (*str*) – Data Resolution for queried Chromosome
- **dataName** (*str*) – Name of data to be queried in given Chromosome.

Returns **gotDataName** – If queried data in given chromosome at given resolution is present in file `True` otherwise `False`.

Return type bool

hasResolution(*chrom*, *resolution*, *dataName=None*)
To check if given resolution for given chromosome is present

Parameters

- **chrom** (*str*) – Chromosome name to be look up in file.
- **resolution** (*str*) – Data Resolution for queried Chromosome
- **dataName** (*str*) – Options to check if resolution for given data name is present

Returns **gotResolution** – If queried resolution of given chromosome present in file `True` otherwise `False`.

Return type bool

open()
open hdf5 file

setTitle(*title*)
Set title of the dataset

It can be used to set or replace the title of the dataset. If file is not yet opened, title will be stored to file when file will be opened.

Parameters **title** (*str*) – The title of dataset.

class BigWigHandler

BigWigHandler(filenames[, ...])	To handle bigWig files and to convert it to h5 file
BigWigHandler.getBigWigInfo()	Retrieve chromosome names and their sizes
BigWigHandler.bigWigtoWig([outfilenames])	To generate Wig file
BigWigHandler.saveAsH5(filename[, ...])	Save data to h5 file.

class BigWigHandler(*filenames*, *pathTobigWigToWig=None*, *pathTobigWigInfo=None*, *chromName=None*, *methodToCombine='mean'*, *workDir=None*, *maxEntryWrite=10000000*)

To handle bigWig files and to convert it to h5 file

This class can be used to convert bigWig file to h5 file. It can also be used to combine several bigWig files that are originated from replicated experiments.

> **Warning:** Presently `bigWigToWig` and `bigWigInfo` is not available for Windows OS. Therefore, this class will fail in this OS.

bigWigFileNames
str or list[str] – List of bigWig file names including path

pathTobigWigToWig
str – Path to `bigWigToWig` program. It can be downloaded from http://hgdownload.cse.ucsc.edu/admin/exe/ for MacOSX and Linux. If path to program is already present in configuration file, it will be taken from the configuration.

If it is not present in configuration file, the input path **should** be provided. It will be stored in configuration file for later use.

pathTobigWigInfo
str – Path to `bigWigInfo` program. It can be downloaded from http://hgdownload.cse.ucsc.edu/admin/exe/ for MacOSX and Linux. If path to program is already present in configuration file, it will be taken from the configuration.

If it is not present in configuration file, the input path **should** be provided. It will be stored in configuration file for later use.

WigFileNames
str – List of Wig file names, either automatically generated or given by user

chromName
str – Name of input target chromosome. If this is provided, only this chromosome data is extracted and stored in h5 file.

wigHandle
WigHandler – WigHandler instance to parse Wig file and save data as hdf5 file

chromSizeInfo
dict – A dictionary containing chromosome size information

methodToCombine
str – method to combine bigWig/Wig files, Presently, accepted keywords are: `mean`, `min` and `max`

maxEntryWrite
int – Number of lines read from Wig file at an instant, after this, data is dumped in temporary numpy array file

Parameters

- **filenames** (*str or list[str]*) – A bigWig file or list of bigWig files including path
- **pathTobigWigToWig** (*str*) – Path to `bigWigToWig` program. It can be downloaded from http://hgdownload.cse.ucsc.edu/admin/exe/ for MacOSX and Linux. If path to program is already present in configuration file, it will be taken from the configuration.

  If it is not present in configuration file, the input path **should** be provided. It will be stored in configuration file for later use.
- **pathTobigWigInfo** (*str*) – Path to `bigWigInfo` program. It can be downloaded from http://hgdownload.cse.ucsc.edu/admin/exe/ for MacOSX and Linux. If path to program is already present in configuration file, it will be taken from the configuration.

  If it is not present in configuration file, the input path **should** be provided. It will be stored in configuration file for later use.
- **chromName** (*str*) – Name of input target chromosome. If this is provided, only this chromosome data is extracted and stored in h5 file.
- **methodToCombine** (*str*) – method to combine bigWig/Wig files, Presently, accepted keywords are: `mean`, `min` and `max`
- **maxEntryWrite** (*int*) – Number of lines read from Wig file at an instant, after this, data is dumped in temporary numpy array file. To reduce memory (RAM) occupancy, reduce this number because large numbers need large RAM.

_bigWigtoWig(*bigWigFileName*, *outfilename*)
Base method to generate Wig file from a bigWig file

Use *BigWigHandler.bigWigtoWig()* to automatically convert all bigWig files to Wig files.

> **Warning:** **Private method**. Use it at your own risk. It is used internally in *BigWigHandler.bigWigtoWig()*

Parameters

- **bigWigFileName** (*str*) – Input bigWig file names.
- **outfilename** (*str*) – Name of output Wig file.

_checkBigWigInfoProgram(*pathTobigWigInfo*)
Check if bigWigInfo program is available or accessible.

If program is not available in configuration file, the given path will be stored in the file after checking its accessibility.

The path is stored in *gcMapExplorer.lib.genomicsDataHandler.BigWigHandler.pathTobigWigInfo*

Parameters **pathTobigWigInfo** (*str*) – Path to bigWigInfo program

_checkBigWigToWigProgram(*pathTobigWigToWig*)
Check if bigWigToWig program is available or accessible.

If program is not available in configuration file, the given path will be stored in the file after checking its accessibility.

The path is stored in *gcMapExplorer.lib.genomicsDataHandler.BigWigHandler.pathTobigWigToWig*

Parameters **pathTobigWigToWig** (*str*) – Path to bigWigToWig program

_getBigWigInfo(*filename*)
Base method to Retrieve chromosome names and their sizes

- Chromosome size information is stored for a given bigWig file. If size of chromosome is already present in dictionary, largest size is stored in dictionary.
- Use *BigWigHandler.getBigWigInfo()* to automatically retrieve chromosome size information from all bigWig files.

Warning: **Private method.** Use it at your own risk. It is used internally in *BigWigHandler.getBigWigInfo()*

Parameters **filename** (*str*) – Input bigWig file

bigWigtoWig(*outfilenames=None*)
To generate Wig file

It uses bigWigToWig program to convert bigWig to Wig file. It uses *BigWigHandler.chromSizeInfo* to extract the listed chromosome data.

If outfilenames are provided, wig files are generated with these names. Otherwise, Wig file names are generated randomly and listed in *BigWigHandler.WigFileNames*. If these files are generated with random names, these will be deleted after execution.

Parameters **outfilenames** (*str or list of strip*) – List of Wig file names. If `None`, names are automatically generated, files are temporarily created and after execution, all files are deleted.

getBigWigInfo()
Retrieve chromosome names and their sizes

BigWigInfo program is executed on all listed bigWig files and chromosomes name with respective size is stored in *BigWigHandler.chromSizeInfo* variable. From the several listed bigWig files, only largest size of chromosomes are considered.

If *BigWigHandler.chromName* is provided, only target chromosome information is kept in *BigWigHandler.chromSizeInfo* dictionary.

saveAsH5(*filename*, *tmpNumpyArrayFiles=None*, *title=None*, *resolutions=None*, *coarsening_methods=None*, *compression='lzf'*, *keep_original=False*)
Save data to h5 file.

Parameters

- **filename** (*str*) – Output hdf5 file name with h5 extension.
- **tmpNumpyArrayFiles** (*TempNumpyArrayFiles* (optional)) – Usually not required. This *TempNumpyArrayFiles* instance stores the temporary numpy array files information. To convert large number of bigWig files, its use increases the conversion

speed significantly because new temporary array files takes time to generate and frequent generation of these files can be avoided.

- **title** (*str (optional)*) – Title of the data
- **resolutions** (*list of str*) – Additional input resolutions other than these default resolutions: 1kb', '2kb', '4kb', '5kb', '8kb', '10kb', '20kb', '40kb', '80kb', '100kb', '160kb','200kb', '320kb', '500kb', '640kb', and '1mb'.

  For Example: use `resolutions=['25kb', '50kb', '75kb']` to add additional 25kb, 50kb and 75kb resolution data.
- **coarsening_methods** (*list of str*) – Methods to coarse or downsample the data for converting from 1-base to coarser resolutions. Presently, five methods are implemented.
  - `'min'` -> Minimum value
  - `'max'` -> Maximum value
  - `'amean'` -> Arithmetic mean or average
  - `'hmean'` -> Harmonic mean
  - `'gmean'` -> Geometric mean
  - `'median'` -> Median

  In case of `None`, all five methods will be considered. User may use only subset of these methods. For example: `coarse_method=['max', 'amean']` can be used for downsampling by only these two methods.
- **compression** (*str*) – data compression method in HDF5 file : `lzf` or `gzip` method.
- **keep_original** (*bool*) – Whether original data present in bigwig file should be incorporated in HDF5 file. This will significantly increase size of HDF5 file.

Examples

```
from gcMapExplorer.lib import genomicsDataHandler as gdh

# start BigWigHandler to combine and convert two bigWig files
bigwig = gdh.BigWigHandler(['first.bigWig', 'second.bigWig'], './bigWigToWig',
↪ './bigWigInfo')

# Save hdf5 file with two additional resolutions
# and only two downsampling method.
bigwig.saveAsH5('converted.h5', resolutions=['25kb', '50kb'], coarsening_
↪methods=['max', 'amean'])
```

class WigHandler

WigHandler(filenames[, chromSizeInfo, ...])	To convert Wig files to hdf5 file
WigHandler.parseWig()	To parse Wig files
WigHandler.setChromosome(chromName)	Set the target chromosome for reading and extracting from wig file
WigHandler.saveAsH5(hdf5Out[, title, ...])	To convert Wig files to hdf5 file

Continued on next page

Table 29 – continued from previous page

WigHandler.getRawWigDataAsDictionary([dictOut])	To get a entire dictionary of data from Wig file

class WigHandler(*filenames*, *chromSizeInfo=None*, *chromName=None*, *indexFile=None*, *tmpNumpyArrayFiles=None*, *methodToCombine='mean'*, *workDir=None*, *maxEntryWrite=10000000*)

To convert Wig files to hdf5 file

It parses wig files and save all data to a hdf5 file for given resolutions.

WigFileNames
list[str] – List of input Wig files.

Note: In case if *WigHandler.chromName* is provided, only one wig file is accepted.

chromName
str – Name of target chromosome name need to be extracted from wig file.

chromSizeInfo
dict – A dictionary containing chromosome size information.

_chromPointerInFile
dict – A dictionary containing position index of each chromosome in wig file.

indexFile
str – A file in json format containing indices (position in wig file) and sizes of chromosomes. If this file is not present and given as input, a new file will be generated. If this file is present, indices and sizes will be taken from this file. If index and size of input chromosome is not present in json file, these will be determined from wig file and stored in same json file. This file could be very helpful in case when same wig file has to be read many times because step to determine index and size of chromosome is skipped.

methodToCombine
str – method to combine bigWig/Wig files, Presently, accepted keywords are: `mean`, `min` and `max`

tmpNumpyArrayFiles
TempNumpyArrayFiles – This *TempNumpyArrayFiles* instance stores the temporary numpy array files information.

isWigParsed
bool – Whether Wig files are already parsed.

maxEntryWrite
int – Number of lines read from Wig file at an instant, after this, data is dumped in temporary numpy array file

Parameters

- **filenames** (*str or list(str)*) – List of input Wig files.

  Note: In case if *WigHandler.chromName* is provided, only one wig file is accepted.

- **chromName** (*str*) – Name of target chromosome name need to be extracted from wig file.
- **chromSizeInfo** (*dict*) – A dictionary containing chromosome size information. Generated by *BigWigHandler.getBigWigInfo()*.
- **indexFile** (*str*) – A file in json format containing indices (position in wig file) and sizes of chromosomes. If this file is not present and given as input, a new file will be generated.

If this file is present, indices and sizes will be taken from this file. If index and size of input chromosome is not present in json file, these will be determined from wig file and stored in same json file. This file could be very helpful in case when same wig file has to be read many times because step to determine index and size of chromosome is skipped.

- **tmpNumpyArrayFiles** (*TempNumpyArrayFiles*) – This *TempNumpyArrayFiles* instance stores the temporary numpy array files information.
- **methodToCombine** (*str*) – method to combine bigWig/Wig files, Presently, accepted keywords are: `mean`, `min` and `max`
- **maxEntryWrite** (*int*) – Number of lines read from Wig file at an instant, after this, data is dumped in temporary numpy array file. To reduce memory (RAM) occupancy, reduce this number because large numbers need large RAM.

_FillDataInNumpyArrayFile(*ChromTitle*, *location_list*, *value_list*)
Fill the extracted data from Wig file to temporary numpy array file

> **Warning:** **Private method**. Use it at your own risk. It is used internally in *WigHandler._parseWig()*.

Parameters

- **ChromTitle** (*str*) – Name of chromosome
- **location_list** (*list of int*) – List of locations for given chromosome
- **value_list** (*list of float*) – List of values for respecitve chromosome location

_PerformDataCoarsening(*Chrom*, *resolution*, *coarsening_method*)
Base method to perform Data coarsening.

This method read temporary Numpy array files and perform data coarsening using the given input method.

> **Warning:** **Private method**. Use it at your own risk. It is used internally in *WigHandler._StoreInHdf5File()*.

Parameters

- **Chrom** (*str*) – Chromosome name
- **resolution** (*str*) – resolution in word.
- **coarsening_method** (*str*) – Name of method to use for data coarsening. Accepted keywords: min, max, median, amean, gmean and hmean.

_StoreInHdf5File(*hdf5Out*, *title*, *resolutions=None*, *coarsening_methods=None*, *compression='lzf'*, *keep_original=False*)
Base method to store coarsed data in hdf5 file.

At first data is coarsened and subsequently stored in h5 file.

> **Warning:** **Private method**. Use it at your own risk. It is used internally in *WigHandler.saveAsH5()*.

Parameters

- **hdf5Out** (str or *HDF5Handler*) – Name of output hdf5 file or instance of *HDF5Handler*
- **title** (*str*) – Title of data
- **resolutions** (*list of str*) – Additional input resolutions other than these default resolutions: 1kb', '2kb', '4kb', '5kb', '8kb', '10kb', '20kb', '40kb', '80kb', '100kb', '160kb','200kb', '320kb', '500kb', '640kb', and '1mb'.

  For Example: use `resolutions=['25kb', '50kb', '75kb']` to add additional 25kb, 50kb and 75kb resolution data.
- **coarsening_methods** (*list of str*) – Methods to coarse or downsample the data for converting from 1-base to coarser resolutions. Presently, five methods are implemented. * `'min'` -> Minimum value * `'max'` -> Maximum value * `'amean'` -> Arithmetic mean or average * `'hmean'` -> Harmonic mean * `'gmean'` -> Geometric mean * `'median'` -> Median In case of `None`, all five methods will be considered. User may use only subset of these methods. For example: `coarse_method=['max', 'amean']` can be used for downsampling by only these two methods.
- **compression** (*str*) – data compression method in HDF5 file : `lzf` or `gzip` method.
- **keep_original** (*bool*) – Whether original data present in wig file should be incorporated in HDF5 file. This will significantly increase size of HDF5 file.

_getChromSizeInfo(*wigFileName*, *inputChrom=None*)

Get chromosome size and index wig file

This method parses a Wig file, extracts chromosome size and index it for each chromosome.

It sets *WigHandler._chromPointerInFile* and *WigHandler.chromSizeInfo*.

> **Warning:** **Private method**. Use it at your own risk. It is used internally durng initialization and in *WigHandler.setChromosome()*.

Parameters

- **wigFileName** (*str*) – Name of Wig File
- **inputChrom** (*str*) – Name of target chromosome

_getChromTitleBedgraph_parseWig(*line*)

To parse chromosome title from the format line of bedGraph format Wig file.

> **Warning:** **Private method**. Use it at your own risk. It is used internally in *WigHandler._parseWig()*.

Parameters **line** (*str*) – The line containing chromosome information from Wig file

_getChromTitle_parseWig(*line*)

To parse chromosome title from the format line of fixedStep and variableStep format Wig file.

> **Warning:** **Private method**. Use it at your own risk. It is used internally in *WigHandler._parseWig()*.

Parameters **line** (*str*) – The line containing chromosome information from Wig file

_getSpan_parseWig(*line*)
To parse span value the format line of fixedStep format Wig file.

> **Warning:** **Private method**. Use it at your own risk. It is used internally in *WigHandler._parseWig()*.

Parameters **line** (*str*) – The line containing chromosome information from Wig file

_getStartStepFixedStep_parseWig(*line*)
To parse start and step values the format line of fixedStep format Wig file.

> **Warning:** **Private method**. Use it at your own risk. It is used internally in *WigHandler._parseWig()*.

Parameters **line** (*str*) – The line containing chromosome information from Wig file

_loadChromSizeAndIndex()
Load chromosome sizes and indices from a json file

_parseWig(*wigFileName*)
Base method to parse a Wig file.

This method parses a Wig file and extracted data are copied in temporary numpy array files.

> **Warning:** **Private method**. Use it at your own risk. It is used internally in *WigHandler.parseWig()*.

Parameters **wigFileName** (*str*) – Name of Wig File

_saveChromSizeAndIndex()
Save chromosomes sizes and indices dictionary to a json file

getRawWigDataAsDictionary(*dicOut=None*)
To get a entire dictionary of data from Wig file

It generates a dictionary of numpy arrays for each chromosome. These arrays are stored in temporary numpy array files of *TempNumpyArrayFiles*.

Parameters **dicOut** (*dict*) – The output dictionary to which data will be added or replaced.

Returns **dicOut** – The output dictionary.

Return type dict

parseWig()
To parse Wig files

This method parses all Wig files listed in *WigHandler.WigFileNames*. The extracted data is further stored in temporary numpy array files of respective chromosome. These numpy array files can be used either for data coarsening or for further analysis.

- **To save as h5:** Use *WigHandler.saveAsH5()*.
- **To perform analysis:** Use *WigHandler.getRawWigDataAsDictionary()* to get a dictionary of numpy arrays.

saveAsH5(*hdf5Out*, *title=None*, *resolutions=None*, *coarsening_methods=None*, *compression='lzf'*, *keep_original=False*)

To convert Wig files to hdf5 file

Parameters

- **hdf5Out** (*HDF5Handler* or str) – Output hdf5 file name or *HDF5Handler* instance
- **title** (*str*) – Title of the data
- **resolutions** (*list of str*) – Additional input resolutions other than these default resolutions: 1kb', '2kb', '4kb', '5kb', '8kb', '10kb', '20kb', '40kb', '80kb', '100kb', '160kb','200kb', '320kb', '500kb', '640kb', and '1mb'.

  For Example: use `resolutions=['25kb', '50kb', '75kb']` to add additional 25kb, 50kb and 75kb resolution data.
- **coarsening_methods** (*list of str*) – Methods to coarse or downsample the data for converting from 1-base to coarser resolutions. Presently, five methods are implemented.
  - `'min'` -> Minimum value
  - `'max'` -> Maximum value
  - `'amean'` -> Arithmetic mean or average
  - `'hmean'` -> Harmonic mean
  - `'gmean'` -> Geometric mean
  - `'median'` -> Median

  In case of `None`, all five methods will be considered. User may use only subset of these methods. For example: `coarse_method=['max', 'amean']` can be used for downsampling by only these two methods.
- **compression** (*str*) – data compression method in HDF5 file : `lzf` or `gzip` method.
- **keep_original** (*bool*) – Whether original data present in bigwig file should be incorporated in HDF5 file. This will significantly increase size of HDF5 file.

setChromosome(*chromName*)

Set the target chromosome for reading and extracting from wig file

To read and convert data of another chromsome from a wig file, it can be set here. After this, directly use *WigHandler.saveAsH5()* to save data in H5 file.

Parameters **chromName** (*str*) – Name of new target chromosome

class BEDHandler

BEDHandler(filenames[, column, chromName, ...])	To convert BED files to hdf5/h5 file
BEDHandler.parseBed()	To parse bed files
BEDHandler.setChromosome(chromName)	Set the target chromosome for reading and extracting from bed file
BEDHandler.saveAsH5(hdf5Out[, title, ...])	To convert bed files to hdf5 file

class BEDHandler(*filenames*, *column=7*, *chromName=None*, *indexFile=None*, *tmpNumpyArrayFiles=None*, *methodToCombine='mean'*, *workDir=None*, *maxEntryWrite=10000000*)

To convert BED files to hdf5/h5 file

It parses bed files and save all data to a hdf5/h5 file for given resolutions.

BedFileNames
list[str] – List of input bed files.

Note: In case if BEDHandler.chromName is provided, only one wig file is accepted.

column
int – The column number, which is considered as data column. Column number could vary and depends on BED format. For example:

- ENCODE broadPeak format (BED 6+3): 7th column
- ENCODE gappedPeak format (BED 12+3): 13th column
- ENCODE narrowPeak format (BED 6+4): 7th column
- ENCODE RNA elements format (BED 6+3): 7th column

chromName
str – Name of target chromosome name need to be extracted from bed file.

chromSizeInfo
dict – A dictionary containing chromosome size information.

_chromPointerInFile
dict – A dictionary containing position index of each chromosome in bed file.

indexFile
str – A file in json format containing indices (position in bed file) and sizes of chromosomes. If this file is not present and given as input, a new file will be generated. If this file is present, indices and sizes will be taken from this file. If index and size of input chromosome is not present in json file, these will be determined from bed file and stored in same json file. This file could be very helpful in case when same wig file has to be read many times because step to determine index and size of chromosome is skipped.

methodToCombine
str – method to combine bed files, Presently, accepted keywords are: `mean`, `min` and `max`

tmpNumpyArrayFiles
TempNumpyArrayFiles – This *TempNumpyArrayFiles* instance stores the temporary numpy array files information.

isBedParsed
bool – Whether bed files are already parsed.

maxEntryWrite
int – Number of lines read from bed file at an instant, after this, data is dumped in temporary numpy array file

Parameters

- **filenames** (*str or list(str)*) – List of input bed files.

  Note: In case if *BEDHandler.chromName* is provided, only one bed file is accepted.

- **column** (*int*) – The column number, which is considered as data column. Column number could vary and depends on BED format. For example:
  - ENCODE broadPeak format (BED 6+3): 7th column
  - ENCODE gappedPeak format (BED 12+3): 13th column
  - ENCODE narrowPeak format (BED 6+4): 7th column
  - ENCODE RNA elements format (BED 6+3): 7th column
- **chromName** (*str*) – Name of target chromosome name need to be extracted from bed file.
- **indexFile** (*str*) – A file in json format containing indices (position in bed file) and sizes of chromosomes. If this file is not present and given as input, a new file will be generated. If this file is present, indices and sizes will be taken from this file. If index and size of input chromosome is not present in json file, these will be determined from bed file and stored in same json file. This file could be very helpful in case when same bed file has to be read many times because step to determine index and size of chromosome is skipped.
- **tmpNumpyArrayFiles** (*TempNumpyArrayFiles*) – This *TempNumpyArrayFiles* instance stores the temporary numpy array files information.
- **methodToCombine** (*str*) – method to combine bed files, Presently, accepted keywords are: `mean`, `min` and `max`
- **maxEntryWrite** (*int*) – Number of lines read from bed file at an instant, after this, data is dumped in temporary numpy array file. To reduce memory (RAM) occupancy, reduce this number because large numbers need large RAM.

_FillDataInNumpyArrayFile(*ChromTitle*, *location_list*, *value_list*)

Fill the extracted data from bed file to temporary numpy array file

Warning: **Private method**. Use it at your own risk. It is used internally in *BEDHandler._parseBed()*.

Parameters

- **ChromTitle** (*str*) – Name of chromosome
- **location_list** (*list of int*) – List of locations for given chromosome
- **value_list** (*list of float*) – List of values for respective chromosome location

_PerformDataCoarsening(*Chrom*, *resolution*, *coarse_method*)

Base method to perform Data coarsening.

This method read temporary Numpy array files and perform data coarsening using the given input method.

> **Warning:** **Private method**. Use it at your own risk. It is used internally in *BEDHandler._StoreInHdf5File()*.

Parameters

- **Chrom** (*str*) – Chromosome name
- **resolution** (*str*) – resolution in word.
- **coarse_method** (*str*) – Name of method to use for data coarsening. Accepted keywords: min, max, median, amean, gmean and hmean.

_StoreInHdf5File(*hdf5Out*, *title*, *resolutions=None*, *coarsening_methods=None*, *compression='lzf'*, *keep_original=False*)

Base method to store coarsened data in hdf5/h5 file.

At first data is coarsened and subsequently stored in h5 file.

> **Warning:** **Private method**. Use it at your own risk. It is used internally in *BEDHandler.saveAsH5()*.

Parameters

- **hdf5Out** (str or *HDF5Handler*) – Name of output hdf5 file or instance of *HDF5Handler*
- **title** (*str*) – Title of data
- **resolutions** (*list of str*) – Additional input resolutions other than these default resolutions: 1kb', '2kb', '4kb', '5kb', '8kb', '10kb', '20kb', '40kb', '80kb', '100kb', '160kb','200kb', '320kb', '500kb', '640kb', and '1mb'.

  For Example: use `resolutions=['25kb', '50kb', '75kb']` to add additional 25kb, 50kb and 75kb resolution data.
- **coarsening_methods** (*list of str*) – Methods to coarse or downsample the data for converting from 1-base to coarser resolutions. Presently, five methods are implemented.
  - `'min'` -> Minimum value
  - `'max'` -> Maximum value
  - `'amean'` -> Arithmetic mean or average
  - `'hmean'` -> Harmonic mean
  - `'gmean'` -> Geometric mean
  - `'median'` -> Median

  In case of `None`, all five methods will be considered. User may use only subset of these methods. For example: `coarse_method=['max', 'amean']` can be used for downsampling by only these two methods.
- **compression** (*str*) – data compression method in HDF5 file : `lzf` or `gzip` method.
- **keep_original** (*bool*) – Whether original data present in wig file should be incorporated in HDF5 file. This will significantly increase size of HDF5 file.

_getChromSizeInfo(*bedFileName*, *inputChrom=None*)
Get chromosome size and index bed file

This method parses a bed file, extracts chromosome size and index it for each chromosome.

It sets *BEDHandler._chromPointerInFile* and *BEDHandler.chromSizeInfo*.

Warning: **Private method**. Use it at your own risk. It is used internally during initialization and in *BEDHandler.setChromosome()*.

Parameters

- **bedFileName** (*str*) – Name of Wig File
- **inputChrom** (*str*) – Name of target chromosome

_loadChromSizeAndIndex()
Load chromosome sizes and indices from a json file

_parseBed(*bedFileName*)
Base method to parse a bed file.

This method parses a bed file and extracted data are copied in temporary numpy array files.

Warning: **Private method**. Use it at your own risk. It is used internally in *BEDHandler.parseBed()*.

Parameters **bedFileName** (*str*) – Name of bed File

_saveChromSizeAndIndex()
Save chromosomes sizes and indices dictionary to a json file

getRawWigDataAsDictionary(*dicOut=None*)
To get a entire dictionary of data from bed file

It generates a dictionary of numpy arrays for each chromosome. These arrays are stored in temporary numpy array files of *TempNumpyArrayFiles*.

Parameters **dicOut** (*dict*) – The output dictionary to which data will be added or replaced.

Returns **dicOut** – The output dictionary.

Return type dict

parseBed()
To parse bed files

This method parses all bed files listed in `BEDHandler.bedFileNames`. The extracted data is further stored in temporary numpy array files of respective chromosome. These numpy array files can be used either for data coarsening or for further analysis.

- **To save as h5:** Use *BEDHandler.saveAsH5()*.
- **To perform analysis:** Use *BEDHandler.getRawWigDataAsDictionary()* to get a dictionary of numpy arrays.

saveAsH5(*hdf5Out*, *title=None*, *resolutions=None*, *coarsening_methods=None*, *compression='lzf'*, *keep_original=False*)
To convert bed files to hdf5 file

It parses bed files, coarsened the data and store in an input hdf5/h5 file.

Parameters

- **hdf5Out** (*HDF5Handler* or str) – Output hdf5 file name or *HDF5Handler* instance
- **title** (*str*) – Title of the data
- **resolutions** (*list of str*) – Additional input resolutions other than these default resolutions: 1kb', '2kb', '4kb', '5kb', '8kb', '10kb', '20kb', '40kb', '80kb', '100kb', '160kb','200kb', '320kb', '500kb', '640kb', and '1mb'.

  For Example: use `resolutions=['25kb', '50kb', '75kb']` to add additional 25kb, 50kb and 75kb resolution data.
- **coarsening_methods** (*list of str*) – Methods to coarse or downsample the data for converting from 1-base to coarser resolutions. Presently, five methods are implemented.
  - `'min'` -> Minimum value
  - `'max'` -> Maximum value
  - `'amean'` -> Arithmetic mean or average
  - `'hmean'` -> Harmonic mean
  - `'gmean'` -> Geometric mean
  - `'median'` -> Median

  In case of `None`, all five methods will be considered. User may use only subset of these methods. For example: `coarse_method=['max', 'amean']` can be used for downsampling by only these two methods.
- **compression** (*str*) – data compression method in HDF5 file : `lzf` or `gzip` method.
- **keep_original** (*bool*) – Whether original data present in bigwig file should be incorporated in HDF5 file. This will significantly increase size of HDF5 file.

setChromosome(*chromName*)

Set the target chromosome for reading and extracting from bed file

To read and convert data of another chromsome from a bed file, it can be set here. After this, directly use *BEDHandler.saveAsH5()* to save data in H5 file.

Parameters **chromName** (*str*) – Name of new target chromosome

class EncodeDatasetsConverter

EncodeDatasetsConverter(inputFile, assembly)	Download and convert datasets from ENCODE Experiments matrix
EncodeDatasetsConverter.saveAsH5(outDir[, ...])	Download the files and convert to gcMapExplorer compatible hdf5 file.

class EncodeDatasetsConverter(*inputFile*, *assembly*, *methodToCombine='mean'*, *pathTobigWigToWig=None*, *pathTobigWigInfo=None*, *workDir=None*)

Download and convert datasets from ENCODE Experiments matrix

It can be used to download and convert multiple datasets from ENCODE Experiment matrix (https://www.encodeproject.org/matrix/?type=Experiment). Presently, only bigWig files are downloaded and then converted.

At first search the datasets on https://www.encodeproject.org/matrix/?type=Experiment . Then click on download button on top of the page. A text file will be downloaded. This text file can be used as input in this program. All bigWig files will be downloaded and converted to gcMapExplorer compatible hdf5 format.

Note: At first a metafile is automatically downloaded and then files are filtered according to bigWig format and Assembly. Subsequently, if several replicates are present, only datasets with combined replicates are considered. In case if two replicates are present and combined replicates are not present, first replicate will be considered. Combining replicates are not yet implemented

---

Warning: Presently `bigWigToWig` and `bigWigInfo` is not available for Windows OS. Therefore, this class will fail in this OS.

inputFile
str – Name of input file downloaded from ENCODE Experiments matrix website.

assembly
str – Name of reference genome. Example: hg19, GRCh38 etc.

pathTobigWigToWig
str – Path to `bigWigToWig` program. It can be downloaded from http://hgdownload.cse.ucsc.edu/admin/exe/ for MacOSX and Linux. If path to program is already present in configuration file, it will be taken from the configuration.

If it is not present in configuration file, the input path **should** be provided. It will be stored in configuration file for later use.

pathTobigWigInfo
str – Path to `bigWigInfo` program. It can be downloaded from http://hgdownload.cse.ucsc.edu/admin/exe/ for MacOSX and Linux. If path to program is already present in configuration file, it will be taken from the configuration.

If it is not present in configuration file, the input path **should** be provided. It will be stored in configuration file for later use.

metafile
str – Name of metafile downloaded from ENCODE website. It is automatically downloaded from input file. It contains all the meta-data required for processing.

metaData = list of dictionary
A list of dictionary read from metafile. It is already filtered according to the different criteria such as file-format, assembly, replicates.

Parameters

- **inputFile** (*str*) – Name of input file downloaded from ENCODE Experiments matrix website.
- **assembly** (*str*) – Name of reference genome. Example: hg19, GRCh38 etc.
- **pathTobigWigToWig** (*str*) – Path to `bigWigToWig` program. It can be downloaded from http://hgdownload.cse.ucsc.edu/admin/exe/ for MacOSX and Linux. If path to program is already present in configuration file, it will be taken from the configuration.

  If it is not present in configuration file, the input path **should** be provided. It will be stored in configuration file for later use.

- **pathTobigWigInfo** (*str*) – Path to `bigWigInfo` program. It can be downloaded from http://hgdownload.cse.ucsc.edu/admin/exe/ for MacOSX and Linux. If path to program is already present in configuration file, it will be taken from the configuration.

  If it is not present in configuration file, the input path **should** be provided. It will be stored in configuration file for later use.

_checkBigWigInfoProgram(*pathTobigWigInfo*)
Check if bigWigInfo program is available or accessible.

If program is not available in configuration file, the given path will be stored in the file after checking its accessibility.

The path is stored in *gcMapExplorer.lib.genomicsDataHandler.EncodeDatasetsConverter.pathTobigWigInfo*

Parameters **pathTobigWigInfo** (*str*) – Path to bigWigInfo program

_checkBigWigToWigProgram(*pathTobigWigToWig*)
Check if bigWigToWig program is available or accessible.

If program is not available in configuration file, the given path will be stored in the file after checking its accessibility.

The path is stored in *gcMapExplorer.lib.genomicsDataHandler.EncodeDatasetsConverter.pathTobigWigToWig*

Parameters **pathTobigWigToWig** (*str*) – Path to bigWigToWig program

_readFromCheckPoint()
Read the titles from checkpoint file.

_removeBigWigFiles()
Remove downloaded bigwig file

_writeToCheckPoint(*idx*)
Write to done titles to checkpoint file

downloadMetaData()
Download the metadata file

It downloads the metadata file and stored at *gcMapExplorer.lib.genomicsDataHandler.EncodeDatasetsConverter.metafile*

filterReplicates()
It filters the metaData according to the replicates. If several replicates for a dataset are present, it only reads the dataset with combined replicates.

In case if two replicates are present and combined replicates are not present, first replicate will be considered. Combining replicates are not yet implemented

readMetaData()
Read the metafile and extract the information

It reads the metafile, filter the datasets according to assembly and file-format and make a list as dictionary.

Each dictionary contains following field:

- title : `Experiment target-Experiment accession-File accession-[fold/signal]`
- type : `fold` for fold change over control or `signal` for signal p-value.
- url : URL to file

- `Experiment accession`
- `File accession`
- `Biological replicate(s)`

saveAsH5(*outDir*, *resolutions=None*, *retryDownload=5*, *coarsening_methods=None*, *compression='lzf'*, *keep_original=False*)

Download the files and convert to gcMapExplorer compatible hdf5 file.

Name of output files:

1. For ChIP-seq assay: a. signal-<Experiment target>-<Experiment accession>-<File accessions.h b. fold-<Experiment target>-<Experiment accession>-<File accessions>.h5
2. For RNA-seq: a. uniq-reads-<date>-<Experiment accession>-<File accessions>.h b. plus-uniq-reads-<date>-<Experiment accession>-<File accessions>.h c. minus-uniq-reads-<date>-<Experiment accession>-<File accessions>.h d. all-reads-<date>-<Experiment accession>-<File accessions>.h5 e. plus-all-reads-<date>-<Experiment accession>-<File accessions>.h5 f. minus-all-reads-<date>-<Experiment accession>-<File accessions>.h5 g. signal-<date>-<Experiment accession>-<File accession>.h5
3. For DNase-seq: a. uniq-reads-signal-<date>-<Experiment accession>-<File accessions>.h b. raw-signal-<date>-<Experiment accession>-<File accessions>.h c. all-reads-signal-<date>-<Experiment accession>-<File accessions>.h d. signal-<date>-<Experiment accession>-<File accessions>.h5
4. For siRNA + RNA-seq: a. uniq-reads-signal-<Experiment target>-<Experiment accession>-<File accessions>.h b. all-reads-signal-<Experiment target>-<Experiment accession>-<File accessions>.h c. signal-<Experiment target>-<Experiment accession>-<File accessions>.h5

Note: Because downloading and conversion might take very long time, it also generates a checkpoint file in the output directory. Therefore, in case of crash or abrupt exit, the process can be continued from the last file.

Parameters

- **outDir** (*str*) – Output directory where all files will be saved. Checkpoint file will be stored in same directory.
- **resolutions** (*list of str*) – Additional input resolutions other than these default resolutions: 1kb', '2kb', '4kb', '5kb', '8kb', '10kb', '20kb', '40kb', '80kb', '100kb', '160kb','200kb', '320kb', '500kb', '640kb', and '1mb'.

  For Example: use `resolutions=['25kb', '50kb', '75kb']` to add additional 25kb, 50kb and 75kb resolution data.
- **retryDownload** (*int*) – Try to download the bigWig files with this many attempt. The time gap between each attempt is two seconds.
- **coarsening_methods** (*list of str*) – Methods to coarse or downsample the data for converting from 1-base to coarser resolutions. Presently, five methods are implemented.
  - `'min'` -> Minimum value
  - `'max'` -> Maximum value
  - `'amean'` -> Arithmetic mean or average
  - `'hmean'` -> Harmonic mean
  - `'gmean'` -> Geometric mean

- `'median'` -> Median

In case of `None`, all five methods will be considered. User may use only subset of these methods. For example: `coarse_method=['max', 'amean']` can be used for downsampling by only these two methods.

- **compression** (*str*) – data compression method in HDF5 file : `lzf` or `gzip` method.
- **keep_original** (*bool*) – Whether original data present in bigwig file should be incorporated in HDF5 file. This will significantly increase size of HDF5 file.

class TextFileHandler

TextFileHandler(filename, shape[, binsize, ...])	To import a genomic data from column text file format
TextFileHandler.readData()	Read data from input file

class TextFileHandler (*filename*, *shape*, *binsize=None*, *title=None*, *workDir=None*)

To import a genomic data from column text file format

It reads text file, make an full array for given shape and fills the missing place with zeros. These zeros could be later masked to perform any analysis.

Example file format:

```
15670000    0.2917373776435852
15680000    0.2292359322309494
15690000    0.023434270173311234
15700000    0.06813383102416992
15710000    0.13660947978496552
15720000    0.17478400468826294
15730000    0.20540907979011536
.
.
.
```

This class is used in browser to visualize the genomic data, which are directly imported from text file.

filename
str – Input text file

shape
int – Size of array required to built

binsize
int – Size of bins expected in input file. If `binsize = None`, binsize will be determined from the files, however, it is good to give expected binsize to check whether expected binsize match with binsize present in input text file.

title
str – Title of the input data

workDir
str – Directory where temporary files will be generated. If `None`, defualt temporary directory of the respective OS will be used.

data
numpy.ndarray or numpy.memmap – One-dimensional array containing the data

tmpNumpyFileName
str – Name of temporary numpy memory-mapped file

Parameters

- **filename** (*str*) – Input text file
- **shape** (*int*) – Size of array required to built
- **binsize** (*int*) – Size of bins expected in input file. If `binsize = None`, binsize will be determined from the files, however, it is good to give expected binsize to check whether expected binsize match with binsize present in input text file.
- **title** (*str*) – Title of the input data
- **workDir** (*str*) – Directory where temporary files will be generated. If `None`, default temporary directory of the respective OS will be used.

_generateTempNumpyFile()
Generate temporary numpy memory-mapped file

_getBinSize()
Determine binsize from input file

_removeTempNumpyFile()
Remove temporary numpy memory-mapped file

readData()
Read data from input file

Read data from input file and store in *TextFileHandler.data* as one-dimensional array

class TempNumpyArrayFiles

TempNumpyArrayFiles([workDir])	To handle temporary numpy array files
TempNumpyArrayFiles.updateArraysByBigWig(...)	Update/resize all array files using given bigWig file
TempNumpyArrayFiles.updateArraysByChromSize(...)	Update/resize an array file using given chromosome and its size
TempNumpyArrayFiles.addChromSizeInfo(...)	Update chromosome sizes using new bigWig file
TempNumpyArrayFiles.generateAllTempNumpyFiles()	Generate all memory mapped numpy array files
TempNumpyArrayFiles.generateTempNumpyFile(key)	Generate a memory mapped numpy array file
TempNumpyArrayFiles.fillAllArraysWithZeros()	Fill all arrays with zeros.

class TempNumpyArrayFiles(*workDir=None*)
To handle temporary numpy array files

To convert a Wig file to hdf5 file, data are parsed, and further stored temporarily in these memory-mapped numpy array files. Use of numpy arrays avoid dependency from storing chromosome location/coordinates because array index is used as the location/coordinates. Additionally, chromosome could be very large and to store these arrays could be memory expensive, these arrays are stored as binary files on the disk.

Note: These generated files are either automatically deleted after execution of script or by deleting [del] *TempNumpyArrayFiles* instance.

chromSizeInfo
dict – Dictionary contains chromosome size. Numpy array files will be generated on the basis of these sizes.

files
dict – Dictionary for names of temporary numpy array files, where keys are chromosomes and values are respective file names.

arrays
dict – Dictionary of memory mapped numpy array (numpy.memmap), where keys are chromosomes and values are respective numpy.memmap arrays.

workDir
str – Working directory where temporary numpy array files will be generated.

_generateTempNumpyFile(*key*, *regenerate=False*)
enerate a memory mapped numpy array file

It is used to generate a memory mapped numpy array file for given chromosome for which size is already in the *TempNumpyArrayFiles.chromSizeInfo* dictionary.

Warning: **Private method.** Use it at your own risk. It is used internally in *TempNumpyArrayFiles.generateTempNumpyFile()*.

Parameters

- **key** (*str*) – chromosome name. Should be present as key in *TempNumpyArrayFiles.chromSizeInfo*.
- **regenerate** (*bool*) – Replace or regenerate new memory mapped array for given chromosome

_getBigWigInfo(*filename*)
Base method to Retrieve chromosome names and their sizes

- Use *TempNumpyArrayFiles.addChromSizeInfo()* to automatically retrieve chromosome size information from bigWig file and to store in *TempNumpyArrayFiles.chromSizeInfo*.

Warning: **Private method.** Use it at your own risk. It is used internally in *TempNumpyArrayFiles.addChromSizeInfo()*

Parameters **filename** (*str*) – Input bigWig file

addChromSizeInfo(*bigWigFileName*)
Update chromosome sizes using new bigWig file

To update *TempNumpyArrayFiles.chromSizeInfo* for new bigWig files, this method can be used.

Note: This method only updates the *TempNumpyArrayFiles.chromSizeInfo* dictionary. It does not resize numpy array files.

Parameters **bigWigFileName** (*str*) – Name of input bigWig file

fillAllArraysWithZeros()
Fill all arrays with zeros.

To fill all memory mapped array with zero. It is used in *WigHandler* so that new data extracted from Wig files can be stored in these array files.

generateAllTempNumpyFiles()
Generate all memory mapped numpy array files

It is used to generate all memory mapped numpy array files using the *TempNumpyArrayFiles.chromSizeInfo* dictionary.

generateTempNumpyFile(*key*, *regenerate=False*)
Generate a memory mapped numpy array file

It is used to generate a memory mapped numpy array for given chromosome for which size is already the *TempNumpyArrayFiles.chromSizeInfo* dictionary.

Parameters

- **key** (*str*) – chromosome name. Should be present as key in *TempNumpyArrayFiles.chromSizeInfo*.
- **regenerate** (*bool*) – Replace or regenerate new memory mapped array for given chromosome

updateArraysByBigWig(*bigWigFileName*)
Update/resize all array files using given bigWig file

Parameters **bigWigFileName** (*str*) – Name of input bigWig file

updateArraysByChromSize(*chrom*, *size*)
Update/resize an array file using given chromosome and its size

Parameters

- **chrom** (*str*) – Chromosome name
- **size** (*int*) – Total size of chromosome

config module

It contains functions that handle configuration of gcMapExplorer. gcMapExplorer uses some default settings and options. This can be read and changed through these modules.

Configuration file structure

```
Configuration
    ├──────────── Dirs
    │               └──────── WorkingDirectory
    │
    └──────────── Programs
                    ├──────── bigWigInfo
                    └──────── bigWigToWig
```

Examples

```
import gcMapExplorer

gcMapExplorer.config.cleanScratch()    # Clean default scratch directory

# Change scratch directory
gcMapExplorer.config.updateConfig('Dirs', 'WorkingDirectory',␣
↪'Path/to/new/scratch/directory')

# Set path to bigWigInfo program
gcMapExplorer.config.updateConfig('Programs', 'bigWigInfo', 'Path/to/bigWigInfo')

# Set path to bigWigToWig program
gcMapExplorer.config.updateConfig('Programs', 'bigWigToWig', 'Path/to/bigWigToWig')

# Print current configuration file content
gcMapExplorer.config.printConfig()

# Get configuration
config = gcMapExplorer.config.getConfig()

# Get scratch directory
print(config['Dirs']['WorkingDirectory'])
```

Summary

updateConfig(section, option, value)	Update configuration file
getConfig()	To get the present configuration.
printConfig()	Print configuration file
cleanScratch()	Clean scratch directory.

updateConfig (*section*, *option*, *value*)

Update configuration file

Parameters

- **section** (*str*) – Section of the configuration files. It could be `Dirs` or `Programs`.
- **option** (*str*) – Input option, for which value is to be changed.
- **value** (*str or int*) – New value of the input option.

getConfig ()

To get the present configuration.

Configuration file has the following organization.

```
Configuration
    |------------ Dirs
    |               |---------- WorkingDirectory
    |
    |------------ Programs
                    |---------- bigWigInfo
                    |---------- bigWigToWig
```

In case no configuration file is found, a new file is generated and default value is assigned to each option.

Returns config – Dictionary of Dictionaries with option name and value pair. For example, config['Dirs']['WorkingDirectory'] contains path to scratch directory. Similarly, config['Programs']['bigWigInfo'] contains path to bigWigInfo program.

Return type dict

printConfig()
Print configuration file

It can be used to print the configuration file. It shows the current configuration of gcMapExplorer.

cleanScratch()
Clean scratch directory.

It checks whether any other gcMapExplorer process is running. In case, when only one process (i.e. current) is running, all files with "gcx" prefix will be deleted from default scratch directory.

3.2 Indices

- genindex
- modindex

Python Module Index

g

Index

Symbols

A

B

C

D

E

F

G

H

I

J

K

L

M

U

W

X

Y