Manual for gCluster

Last updated: 05/04/2017

Developed and maintained by:

Computational Epigenomics and Bioinformatics Group
Dept. of Genetics & Inst. of Biotechnology,
University of Granada, Spain

Questions and feedback: hackenberg@ugr.es

Content

1. Preparation of the protocol

You can follow the protocol in two ways:

a) Install the executable files.
b) Use gClusterVM, a virtual machine with all that you need.

Both options work on most of operating system (Windows, Mac OS X, Linux and Solaris).

a. Using gClusterVM

- Download and install VirtualBox (Windows, Mac OS X, Linux and Solaris are supported). Install also the VirtualBox Extension Pack.
- Download gClusterVM OVA file and import it on VirtualBox by double-clicking.
- Configures the machine virtual: allocated memory (minimum: X GB), CPUs assigned (minimum: X) and folder shared for exchange files with the machine virtual (highly recommended).
- Run the virtual machine.

b. Using standalone executables

To use standalone executables, first you must follow the following steps:

i) Download and install Java 8 or higher.
ii) Download and install Python 3.4 or higher. Important: check the option 'Add to the PATH' during the installation.
iii) Download and extract the standalone executables.
iv) Open a terminal to execute the commands of the protocol or in this manual. If you are using windows, you can use CMD or PowerShell.

In the following sections we will describe in detail the features of all programs with usage examples.

2. Prepare the sequences: prepareAssembly.py, makeSeqObj.jar and randomizer.jar

a. prepareAssembly.py

It can obtains an assembly (from UCSC) and extract canonical sequences from it. It also works with local assembly files.

Input parameters

- **-i <path or URL>:** Path or URL to multi-FASTA file (.fa, .fa.gz,.fa.tar.gz, .bz2)
- **-u <UCSC id>:** UCSC assembly ID (e.g. hg38)
- **-l <label>:** Label for output files. (Default: 'assembly')
- **-o <outdir>:** Path to output directory, default current directory

- **-r <regex>:** REGEX to filter canonical or non-canonical chromosomes. REGEX to filter canonical chromosomes must be precede$

Output files

In the output folder two files will be generated:

- **'label'_canonical.fa:** multi-FASTA file that contains all canonical chromosomes sequence.
- **'label'_noncanonical.fa:** multi-FASTA file that contains all non-canonical chromosomes sequence.

Extract canonical sequences from assembly (from local file):

```
python prepareAssembly.py -i /home/meth/sequences/hg38.fa -o /home/meth/sequences
```

Obtain assembly and extract canonical sequences (from UCSC):

```
python prepareAssembly.py -u hg38 -o /home/meth/sequences
```

b. makeSeqObj.jar

It index the 'FASTA' input files in order to make them faster accessible. With an 'assembly.fa' input file it generates an 'assembly.zip' file which, in turn, is the input file for the programs described below.

Input parameters

- **'assembly'_canonical.fa:** Multi-FASTA file of the genome assembly.

Output files

- **'assembly'.zip** file, that should be used for all the subsequent analyses.
- **'assembly'_canonical.N:** file with the coordinates of the N-runs in the assembly.
- **'assembly'_canonical.chromSize:** chromosome size file.

All of them will be created in the same folder as input

```
java –jar makeSeqObj.jar /home/meth/sequences/hg38_canonical.fa
```

c. randomizer.jar

Randomizes DNA sequences preserving the dinucleotide frequencies.

Input parameters

- **'assembly'.zip** file generated by makeSeqObj.jar

Output files

- **'assembly'_random.zip** file. It would be created in the same folder as input.

```
java –jar randomizer.jar /home/meth/sequences/hg38_canonical.zip
```

3. Determine local clusters of DNA words and its global clustering properties: gCluster

gCluster could determine the local clusters of a given DNA word (CpG islands if the DNA word is 'CG') and its global clustering properties. It can work on both strands (for non-palindromic words) and accepts any combination of DNA words (like CAG:CTG:CCG for the methylation context CHG).

The following parameters are all possible options of gCluster. In the above sections it would be detailed which parameters must be set for each analysis.

Input parameters:

- **genome=<path>:** the indexed multi-FASTA file (see 'Prepare the sequences' section above).
- **output=<output folder>:** the output folder where the results will be written.
- **pattern=<the k-mers>:** the k-mers (DNA words) that should be analysed, eg. Pattern=CG to obtain CpG islands. For detection of CWG cluster, pattern=CAG:CTG should be used.
- **writedistribution=<boolean>:** write out the observed and expected distance distribution (default writedistribution=false)
- **chromStat=<boolean>:** if true, the program writes out additional information of the chromosome sequences like G+C content, CpG frequencies (observed/expected ratios), lengths, etc. (default: chromStat=false)

Output files:

- **cluster.txt:** The clusters identified by its chromosomal coordinates and p-value adding basic compositional statistics like G+C content and O/E ratios of the pattern (the number of observed patterns divided by the number of expected pattern). In case of a compound pattern like CAG:CTG, the mean O/E ratio is calculated.
- **CVnor.txt:** The file holds the normalized coefficient of variation for each chromosome.
- ***.distr files:** The distance distribution, the observed and expected frequencies as a function of next-neighbour distance.
- **log.txt:** A log file
- **stat.txt:** A basic statistic as a function of chromosome, i.e pattern frequency, G+C content, lengths and contig length (the sequence length minus the sum of all N's)

a. Calculate clustering properties

mkdir /home/meth/results (generates the 'results' output directory)

java –jar gCluster.jar genome=/home/meth/sequences/hg38_canonical.zip pattern=CG output=/home/meth/results/hg38_CG writedistribution=true chromStat=true

IMPORTANT: writedistribution and chromStat parameters must be set to 'True' to obtain clustering properties.

b. Detect the clusters of DNA words

CpG islands:

With the following command (same as above), the clusters are automatically determined and written to the cluster.txt file. The distance threshold is determined by the genome intersection.

java –jar gCluster.jar genome=/home/meth/sequences/hg38_canonical.zip pattern=CG output=/home/meth/results/hg38_CG writedistribution=true chromStat=true

Clusters of other, biologically relevant k-mers:

Like mentioned before, in plants cytosine can be methylated as well in other contexts like CWG (CAG, CTG), CCG or CHH.

The clustering of these contexts can be calculated with the following command.

java –jar gCluster.jar genome=/home/meth/sequences/ hg38_canonical.zip pattern=CAG:CTG output=/home/meth/results/hg38_CWG writedistribution=true chromStat=true

4. Clusters of clusters: GenomeCluster.pl

GenomeCluster determines the local clusters of genome elements identified by its chromosome coordinates. It could be used to cluster the clusters predicted by gCluster, in order to make clusters of cluster.

Input parameters (the arguments must be given in this order):

- **Argument 1:** the distance model <element; start; middle; end>.
- **Argument 2:** the BED file within the chromosome coordinates of whatever genome element (e.g. CpG islands).
- **Argument 3**: the distance threshold model <gi: genome intersection; ci: chromosome intersection; N (integer): percentile>.
- **Argument 4:** the p-value threshold (by default 1E-5 as in gCluster).
- **Argument 5:** the chromosome size file created by makeSeqObj.jar (see 'Prepare the sequences' section above).
- **Argument 6:** the file with the N-runs created by makeSeqObj.jar (see 'Prepare the sequences' section above).

- **Argument 7:** the maximum number of allowed Ns between two elements. It must be an integer greater or equal to 0 (default: 0).

Output files:

- ***_genomeIntersec_start_GenomeCluster.txt:** there are as many of these files as chromosomes in the assembly. They are tabular files with the columns described below.

Columns in outfile	Description
Chrom	Chromosome where the local cluster belongs.
From	Start chromosome coordinate of the local cluster.
To	End chromosome coordinate of the local cluster.
Length	Length of the local cluster.
Count	Number of genome elements within the local cluster.
PValue	P-value of the local cluster.
logPValue	Decimal logarithm of the p-value.

perl GenomeCluster.pl start /home/meth/results/hg38_CG/cluster.txt gi 1E-5 /home/meth/sequences/hg38_canonical.chromSize /home/meth/sequences/hg38_canonical.N 0

5. Determine the methylation of CpG Islands: NGSmethDB_API_client.py

In order to calculate differentially methylation CGIs, we will have to obtain first the methylation values for the CpG islands from our database and second, compare the methylation values between two samples. To obtain the methylomes, we will use NGSmethDB_API_client.py executable.

Input parameters:

- **-i <input bed file>:** in general, BED3 files will be accepted (only the first 3 columns will be considered), including the 'cluster.txt' output files generated by gCluster
- **-o <output folder>:** output folder

Output files:

- **<sample>.CG.meth.tsv:** Contains the CG methylation data for all regions defined in the bed data input file. There is one <sample>.CG.meth.tsv file for each selected sample. Each line corresponds to one region.

Columns in outfile	Description
ID	chrom_start_end
refPatt	Number of CpGs within the region in the reference genome
indPatt	Number of CpGs in the region for the genotype of the sample
meanMR	Mean of CpG methylation ratio distribution
sdMR	Standard Deviation of CpG methylation ratio distribution

p10MR	Percentile 10 of CpG methylation ratio distribution
q1MR	Quartile 1 of CpG methylation ratio distribution
q2MR	Median of CpG methylation ratio distribution
q3MR	Quartile 3 of CpG methylation ratio distribution
p90MR	Percentile 90 of CpG methylation ratio distribution
totalMC	Total number of methylcytosines in the region
totalC	Total number of mapped reads within the region

- **<sample>.CHG.meth.tsv (plants only**): Same as above, but for CHG.

```
python NGSmethDB_API_client.py -i /home/meth/results/hg38_CG/cluster.txt -o /home/meth/results/methylationData
```

6. Calculate differential methylation: calcDMIs.py

Takes the output files from NGSmethDB_API_client.py for two samples and calculates statistically significant differentially methylated CpG islands between them. The statistical significance is assessed by means of a Fisher exact test.

Input parameters:

- **-a <sample1 file> and –b <sample2 file>:** output methylation files from NGSmethDB_API_client.py
- **-o <outfile>:** output file. Default is 'outputDMIs.txt'
- **-p <pvalue>:** p-value threshold for DMIs. Default is 1E-5.

Output files:

- **<outfile>:** a file containing the differentially methylated CpG islands.

Columns in outfile	**Description**
ID	chrom_start_end
mC_Sample1	Number of methylcytosines for each cluster in sample 1
reads_Sample1	Number of reads for each cluster in sample 1
mRatio_Sample1	methRatio (mC_Sample1/reads_Sample1) for each cluster in sample 1
mC_Sample2	Number of methylcytosines for each cluster in sample 2
reads_Sample2	Number of reads for each cluster in sample 2
mRatio_Sample2	methRatio (mC_Sample1/reads_Sample2) for each cluster in sample 2
diffMeth	Absolute difference between methRatio_Sample1 and methRatio_Sample2
pvalue	Statistical significance assessed by means of the Fisher exact test

```
python calc_DMIs.py –a home/meth/results/methylationData/STL003.gastric.CG.meth.tsv -b /home/meth/results/methylationData/STL003.pancreas.CG.meth.tsv -o /home/meth/results/methylationData/STL003.gastric.STL003.pancreas.DMIs.txt -p 1E-5
```