Cseq-Simulator manual

by Wanja Kassuhn
wanja.kassuhn@mdc-berlin.de

and

Philipp Drewe
philipp.drewe@mdc-berlin.de

Contents

1 Getting started

1.1 Requirements

The following tools and packages are used by this pipeline and need to be installed first:

- flux-simulator: `http://sammeth.net/confluence/display/SIM/2+-+Download`
- FIMO: `http://meme-suite.org/doc/download.html`

Please make sure that flux-simulator and FIMO are both exported to your PATH variable.

```
$ export PATH=$PATH:/path/to/flux-simulator
$ export PATH=$PATH:/path/to/FIMO
```

No further installation required.

1.2 Basic workflow

If all required software is installed and the input files are in the same directory (including the par.par file), the pipeline can be started with the following command:

bash Cseq.sh mode option1 value option2 value ...

You can also use:

bash Cseq.sh --help

For each RBP you first need to generate a motif index. This can be done with mode 1 like so:

```
$ bash Cseq.sh generateIndex -o /output -i /input -m FIMOinputFile.txt
```

Copy the motif index file to your input directory. Then you can generate an expression profile. This step is optional and if no expression profile is supplied in mode 3, a new one will be generated.

```
$ bash Cseq.sh generateExpression -o /output -i /input -g /genome -G annotation.gtf
    -mi index.txt
```

Now copy the expression profile to your input directory. Finally, perform mode 3 like so:

```
$ bash Cseq.sh simulateReads -o /output -i /input -g /genome -G annotation.gtf
    -mi index.txt -m FIMOinputFile.txt -r INT -rn STRING -ml INT -c mutRates.txt
    -e expression.pro -x STRING -ml INT
```

2 Input files

2.1 Genome

The genome needs to be separated into individual chromosomes (names: chr*.fasta or chr*.fa). You can also provide a genome as a gziped file (genome.tar.gz). A genome can be downloaded from: `http://www.gencodegenes.org/releases/current.html`

Fasta files

Content	Regions	Description	Download
Protein-coding transcript sequences	CHR	• Nucleotide sequences of coding transcripts on the reference chromosomes • Transcript biotypes: protein_coding, nonsense_mediated_decay, non_stop_decay, IG_*_gene, TR_*_gene, polymorphic_pseudogene	Fasta
Protein-coding transcript translation sequences	CHR	• Amino acid sequences of coding transcript translations on the reference chromosomes • Transcript biotypes: protein_coding, nonsense_mediated_decay, non_stop_decay, IG_*_gene, TR_*_gene, polymorphic_pseudogene	Fasta
Long non-coding RNA transcript sequences	CHR	• Nucleotide sequences of long non-coding RNA transcripts on the reference chromosomes	Fasta
Genome sequence (GRCh38.p2)	ALL	• Nucleotide sequence of the GRCh38.p2 genome assembly version on all regions, including reference chromosomes, scaffolds, assembly patches and haplotypes • The sequence region names are the same as in the GTF/GFF3 files	Fasta

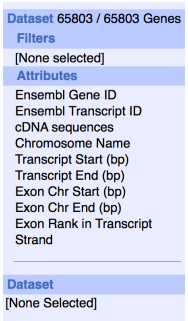
To prepare the genome for use in this pipeline, use the included script "splitGenome.py" like this :

```
$ cd /Downloads
$ gzcat GRCh38.p2.genome.fa.gz | python /Path/to/scripts/splitGenome.py
$ tar -zcf genome.tar.gz -C /Downloads chr*.fa*
$ rm /Downloads/chr*.fa*
```

```
Wanjas-MBP:genome Pro$ ls
chr1.fasta      chr15.fasta     chr20.fasta     chr6.fasta      genome.genome
chr10.fasta     chr16.fasta     chr21.fasta     chr7.fasta      genome.tar.gz
chr11.fasta     chr17.fasta     chr22.fasta     chr8.fasta
chr12.fasta     chr18.fasta     chr3.fasta      chr9.fasta
chr13.fasta     chr19.fasta     chr4.fasta      chrX.fasta
chr14.fasta     chr2.fasta      chr5.fasta      chrY.fasta
```

2.2 Transcriptome

The order of information in the identifiers of the transcripts is important. You can download a transcriptome from `http://www.ensembl.org/biomart/martview/` and select the options (in that order!) from the following image:



Chromosome
Gene ID | Transcript ID | Transcript start/end | Exon start coordinates | Exon end coordinate | Strand | Exon rank

```
>ENSG00000003137|ENST00000001146|2|72129238|72148038|72129238;72133023;72143989;72147631;72134761;72135144|72132619;72133307;72144213;72148038;72134916;72135419|6;5;2;1;4;3|-1
AGGCAATTTTTTTCCTCCCTCTCTCCGCTCCCCTCGCAGCCTCCACTCCCTTTCCCTTGG
CCCCTTCCTCCTTCTCTGTTTCGGCTGGAGGTGCCAGGACCCCCGGCCGCAGCCTCCCCT
```

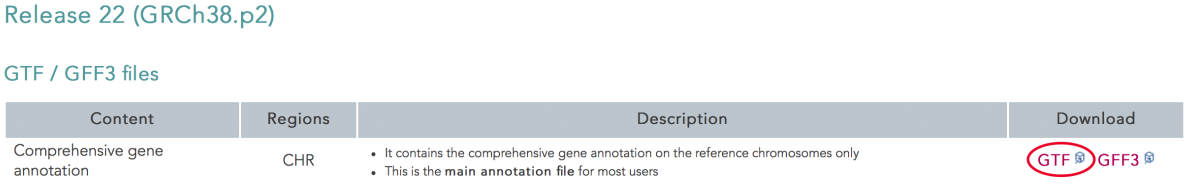
When you have successfully downloaded the transcriptome, use the included script "transOrder.py" to reorder the exon coordinates of the identifiers. It can be either be compressed ("transcripts.tar.gz") or as a fasta file ("transcripts.fasta"). However, it must be located in the input

directory.

```
$ cd /Downloads
$ gzcat mart_export.txt.gz | python /Path/to/script/transOrder.py
```

2.3 Annotation

The pipeline needs an annotation in gtf format. This annotation has to correspond to the used genome and transcriptome versions. Annotations can be found at `http://www.gencodegenes.org/releases/current.html` and `http://www.ensembl.org/info/data/ftp/index.html?redirect=no`.



2.4 Fimo motif file

The motif search requires a motif file. This consists of an alphabet, a background distribution and a PWM of the RBPs interaction site (`http://meme-suite.org/doc/meme-format.html`). An example is included in the example directory.

2.5 Mutation rates

The conversions are introduced with help of an array of floats separated by newlines ("\n"). The entries in the array (lines from top to bottom) correspond to positions upstream of the motif, followed by the positions for the motif and finally the downstream positions. This means the positions corresponding to the motif are in the center of the file (default: 31). When using longer reads, the number of lines might need to be extended to avoid index errors. Changes can be made in line 18 of the "mutReadsUniversal.py" script. An example is included in the examples directory.
Exception: The default value for the truncation rate file is set to 61, since the RNA-seq read length is doubled to ensure the that the final reads have the correct length.

2.6 Other

3 Output files

Mutated reads The simulated reads are output in fasta format. The first pair of coordinates in the header represent the originating locus and the second pair the exact coordinates (keep in mind that coordinates can be 0- or 1-based). The last integer in each reads header corresponds to the amount of conversions in each of the reads (mutated_Reads*.tar.gz).

Unmutated reads The simulated reads are also output without conversions. These can be used to exactly locate the position of the T to C conversions in each of the reads (by identical read header) (unmutated_Reads*.tar.gz).

Data set statistics The pipeline generates a statistical output of the reads. It includes the amount of conversions, spliced and unspliced reads, as well as an overall read count. Currently no deletions or truncations are introduced (read_log.txt).

RNA-Seq output The output of the RNA-Seq simulation is stored as a library archive (LIB.tar.gz) and a two gziped files (bed.tar.gz which stores a bed file and Reads*.tar.gz which stores the reads in fasta). The fasta file can be used to simulate additional data with the same read base. Both files can either be deleted or used for further runs.

Index file The indexing.txt file contains all occurrences of the motif in genomic coordinates. It can be saved for other purposes or be deleted. It will be generated in every run, as it increases the run time only marginally.

Logs The flux_log.txt is a log file of the RNA-Seq simulation. It gives detailed information about the fragment counts and the run time. It is very important that if the flux-simulator is interrupted all intermediate files are deleted or the pipeline will be stuck and wait for an input. The log.txt is an additional run time tracker for the whole pipeline.

Other All other intermediate files are used as checkpoints for the pipeline. This allows to skip already performed actions..

4 Description of all options

4.1 Modes

generateIndex Generates a motif index. The motif index is used to filter the fragments of the RNA-Seq simulation for overlaps. This mode requires the following parameters: -i, -o and -m.

generateExpression Generates an expression profile. The expression profile allows to either reproduce a run with the same library of fragments (different mutations), or lets the user customize the expression of individual transcripts. This mode requires the following parameters: -i, -o, -g, -G and -mi.

simulateReads Simulates PAR-CLIP reads. This mode generates reads from an expression profile, creates and filters the library, sequences the fragments, and finally introduces T to C conversions and truncations. This mode requires the following parameters: -i, -o, -g, -G, -c and -ml.

4.2 Required

-g, --genomeDir Sets the path to genome file in fasta format and *.genome file with the same base name, containing the sizes of the individual chromosomes. (Optional as gziped file: genome.tar.gz if no fasta files are present in the directory)

-o, --outDir Sets the output directory which is also used as a working directory.

-G, --gtf Sets the name of the annotation file (located in the input directory).

-i, --inputDir Sets the path to the input directory (annotation, transcriptome, flux parameter template, pattern and mutation rate files).

-m, --motifFile Sets the name of the FIMO motif file (located in the input directory).

-ml, --motifLen Sets the length of the motif in nt.

-c, --mutRates Sets the name of the mutation rate file (located in the input directory).

4.3 Optional

-r, --readsN Sets the starting amount of molecules for the RNA-Seq simulation (the resulting read amount is strongly dependent on expression profile). This is not the number of PAR-CLIP reads (default: 2,000,000,000). It is recommended to use the default value or if enough memory is available a higher one.

-mi, --motifIdx Sets the name of the motif index that needs to be located in the input directory (format of fimo output file).

-y, --libName Sets the archive name the library file is stored in (default: LIB).

-s, --motifDef Sets the motif to unspecific. This changes the threshold for the motif search with FIMO. For example, ELAVL1 is not detected in a motif search with the specific option (default: specific).

-l, --readLength Sets the read length (default: 30). Minimum is 11. When -x T is set, a tuple can be specified (i.e. 25,45).

-rn, --runName Sets the name of the output files of the PAR-CLIP simulation (default: Runx).

-e, --express Sets the name of a expression profile. If specified, it must be located in the input directory. Otherwise, a new expression profile will be generated.

-u, --memory Sets the the available memory (default: 16). Higher memory lets you sequence more reads without crashing.

-a, --bsaffin Sets the the binding site affinity for all binding sites. This means a binding site affinity of 1.0 will include all fragments with motif into the filtered library (default: 1.0).

-x,--mutMode Sets the diagnostic events that are introduced (default: C). Any combination of D, C or T seperated by a ",". A "T" will double the user-specified read length and truncate the generated reads to the user-specified read length.

5 Examples

- PAR-CLIP with conversions and deletions

  First start by creating a motif index:

  ```
  $ bash Cseq.sh generateIndex -o /output -i /input -m FIMOinputFile.txt
  ```

  Now copy the generated file (/output/index.txt) to your input directory for the simulation. Optional: Generate an expression profile:

  ```
  $ bash Cseq.sh generateExpression -o /output -i /input -g /genome -G annotation.gtf
  -mi index.txt
  ```

  Finally, copy the expression profile (/output/expression.pro) to your input directory for the simulation. Make sure all files are located in the input directory (if step 2 is skipped also include -m FIMOinputFile.txt)and generate the reads like so:

  ```
  $ bash /Users/Pro/Desktop/Cseq-Simulator-v1.02/bin/Cseq.sh simulateReads -g
  /Users/Pro/Documents/workspace/PARCLIP/input/genome/ -o /Users/Pro/Desktop/test2/
  -G gencode.v22.annotation.gtf -i /Users/Pro/Documents/workspace/PARCLIP/input/ -ml 8 -mi index.txt
  -c PUM2_mutRates.txt -d PUM2_delRates.txt -r 20000000 -u 12 -l 28 -e Runx.pro -x C,D
  ```

- iCLIP

  First start by creating a motif index:

  ```
  $ bash Cseq.sh generateIndex -o /output -i /input -m FIMOinputFile.txt
  ```

  Now copy the generated file (/output/index.txt) to your input directory for the simulation. Optional: Generate an expression profile:

  ```
  $ bash Cseq.sh generateExpression -o /output -i /input -g /genome -G annotation.gtf
  -mi index.txt
  ```

  Finally, copy the expression profile (/output/expression.pro) to your input directory for the simulation. Make sure all files are located in the input directory (if step 2 is skipped also include -m FIMOinputFile.txt)and generate the reads like so:

  ```
  $ bash /Users/Pro/Desktop/Cseq-Simulator-v1.02/bin/Cseq.sh simulateReads -g
  /Users/Pro/Documents/workspace/PARCLIP/input/genome/ -o /Users/Pro/Desktop/test2/
  -G gencode.v22.annotation.gtf -i /Users/Pro/Documents/workspace/PARCLIP/input/ -ml 8
  -mi index.txt -t PUM2_truncRates.txt -r 20000000 -u 12 -l 28 -e Runx.pro -x T
  ```

  Alternatively, bounds for read lengths can be set:

  ```
  $ bash /Users/Pro/Desktop/Cseq-Simulator-v1.02/bin/Cseq.sh simulateReads -g
  /Users/Pro/Documents/workspace/PARCLIP/input/genome/ -o /Users/Pro/Desktop/test2/
  -G gencode.v22.annotation.gtf -i /Users/Pro/Documents/workspace/PARCLIP/input/ -ml 8
  -mi index.txt -t PUM2_truncRates.txt -r 20000000 -u 12 -l 25,45 -e Runx.pro -x T
  ```