epic Documentation

Endre Bakken Stovner

Nov 21, 2018

Getting Started

epic is a modern reimplementation of the extremely popular SICER algorithm for finding broad, diffusely enriched regions in ChIP-Seq data. epic focuses on speed, innovation and ease of use.

In addtion to the classic SICER algorithm, epic contains an array of tools to help you do differential analysis of ChIP-Seq data from multiple conditions.

CHAPTER 1

Novel features

- epic creates output files that allow you to leverage state-of-the-art statistical software to do rigorous differential analyses of experimental conditions
- epic creates bigwigs and bed files that allow you to visualize and explore the results in genome browsers

CHAPTER 2

Improvements

epic contains a slew of improvements

- Python 2.7 and 3+ compatible
- much faster and multicore
- easy to install and use; exists both in Bioconda and PyPI
- accepts both single- and paired-end reads
- can analyse multiple ChIP and input files at the same time (and even mix paired and single end files in the same analysis)
- sensible defaults - only needs the files to analyse as command line args
- effective genome size autoselected based on read length (autodetected) and genome
- epic can compute the effective genome fraction for you, allowing you to analyze any genome for which you have a fasta file
- (soon) metadata for all genomes in UCSC is available so you only need to give the genome name for the species you wish to analyse
- accepts custom genomes and assemblies
- automatically tested, which safely and easily allows contributions
- continually used, updated and maintained by the author

2.1 Installation

The preferred way to install epic is to use the conda package manager.

First install conda according to the instructions, then you can install epic with

```
conda install -c bioconda epic
```

You can also install epic from PyPI with

```
pip install bioepic
```

2.2 Quick-start

To use epic, you need at least one ChIP and one input bed file. Let's download some test data from github:

```
wget https://raw.githubusercontent.com/biocore-ntnu/epic/master/examples/control.bedpe
wget https://raw.githubusercontent.com/biocore-ntnu/epic/master/examples/test.bed
```

Now you can run epic on these files with

```
epic -t test.bed -c control.bedpe -o enriched_regions.csv
```

The -t means treatment (ChIP), while the -c means control (input). The -o is the path where you want to store the results. If no output path is given, the results are written to stdout.

2.3 Basic intro

epic is used to find differentially enriched regions in ChIP-Seq data. It is especially well suited for proteins that bind diffusely over longer regions (such as many histone modifications).

The epic domain caller has three features.

- finding enriched regions in ChIP-Seq data
- creating files that allow you to visualize the ChIP-Seq signal in genome browsers
- creating a matrix of read counts for differential analysis between experimental conditions

Let's look at these in turn.

2.3.1 Finding enriched regions in ChIP-Seq data

An enriched region is a region where the amount of ChIP signal is significantly higher than the amount of input (background) signal.

The list of enriched regions is the main output-file from epic and is always included (matrix and visualization files are optional).

In the quick-start example we found enriched regions and stored them in.

Now let us see what such a file looks like. Here the first five lines are shown.

```
# epic -c control.bedpe -t examples/test.bed -o results.csv
Chromosome Start End ChIP Input Score Fold_change P FDR
chr1 23568400 23568599 2 0 13.54464274405703 2534.5007148630584 8.184731894908448e-11␣
↪6.319374393278151e-10
chr1 26401200 26401399 2 0 13.54464274405703 2534.5007148630584 8.184731894908448e-11␣
↪6.319374393278151e-10
chr1 33054800 33055399 2 0 14.288252844518933 844.8335716210196 2.207263610333191e-09␣
↪3.85690272963484e-09
```

(continues on next page)

(continued from previous page)

The first line merely contains the command used to produce the output. The second line contains the column names. From the third line out, the data we are actually interested in is shown.

The three first columns give the location of the enriched region. The ChIP and Input columns contain the number of ChIP and Input reads in those regions, respectively.

The last four columns contain statistical information about how enriched the region is.

2.3.2 Visualization

To allow for visualization of the data in genome browsers, epic can output both bed and bigwig files that allow you to explore and analyze the data.

The -bw flag takes a folder to store the bigwigs in (if the folder does not exist, it is created for you).

```
epic -t test.bed -c control.bedpe -o enriched_regions.csv -bw bigwigs
```

Now one bigwig for each analysed file is stored in the folder bigwigs

```
ls bigwigs/
# control.bw test.bw
```

2.3.3 Count-matrix

For downstream statistical analyses, epic can output a matrix of read-counts.

You can ask for one with the -sm flag. (Remember to give it a .gz extension as the results are stored as a gzipped file!)

```
epic -t test.bed -c control.bedpe -o enriched_regions.csv -sm matrix.gz
```

2.4 Options

epic allows for many flags to denote (optional) output files or to change the execution of epic.

- **-t, –treatment**

  One or more ChIP files (bed or bedpe format)

- **-c, –control**

  One or more input files (bed or bedpe format)

- **-o, –outfile**

  File to write results to. By default sent to stdout.

- **-l, –log**

  File to write log messages to. Also written to stderr by default.

- **-cpu, –number-cores**

  The number of cores epic should use. Can at most take advantage of 1 core per strand per chromosome (i.e. 46 for humans). Default: 1

- **-gn, –genome**

  Which genome to analyze. By default hg19.

- **-k, –keep-duplicates**

  Keep reads mapping to the same position on the same strand within a library. The default is to remove all but the first duplicate (this is done once per file, not for all files collectively.)

- **-w, –window-size**

  Size of the windows (bins) used to scan the genome. This is also the smallest possible enriched region you can get. Default 200.

- **-g, –gaps-allowed**

  How many non-enriched windows in a row can be part of the same enriched region. If the number of gaps between two enriched windows is higher than this number, they are considered separate regions. Default: 3

- **-fs, –fragment-size**

  (Only used for single-end files) Size of the sequenced fragment. The center of the fragment will be used to calculate which window a read ended up in. So reads are shifted by fragment-size/2. Default 150.

- **-fdr, –false-discovery-rate-cutoff**

  Remove all regions with an FDR below cutoff. Note: this also affects which windows are considered enriched in the optional matrix output and which regions are included in the optional bed output.

- **-egf, –effective-genome-fraction**

  Use a different effective genome fraction than the one included in epic. Or include an egf for custom genomes that are not a part of epic. Should be a number between 0 and 1. Autoinferred by sampled read-length and genome by default.

- **-cs, –chromsizes**

  Set the chromosome lengths yourself in a file with two columns: chromosome names and sizes. Useful to analyze custom genomes, assemblies or simulated data. Only chromosomes included in the file will be analyzed.

- **-sm, –store-matrix**

  The path in which to store a gzipped matrix of read counts per window. One column for each ChIP and Input file.

- **-b, –bed**

  A summary bed file of all regions, for display in the UCSC genome browser or for use in downstream analyses with e.g. bedtools. The score field is log2(#ChIP/#Input) * 100 capped at a 1000.

- **-bw, –bigwig**

  The folder in which to store a RPKM-normalized bigwig for each file in the dataset. The bigwig shows how epic sees the data. It shows all bins, not just those in enriched regions.

- **-i2bw, –individual-log2fc-bigwigs**

  The folder in which to store a log2FC bigwig for each ChIP bed/bedpe. Each file is divided against a normalized sum of the Input.

- **-cbw, –chip-bigwig**

  Store the sum of the RPKM-normalized ChIP-data to a bigwig file.

- **-ibw, –input-bigwig**

  Store the sum of the RPKM-normalized Input-data to a bigwig file.

- **-v, –version**

  Show version.

2.5 Output files

epic can produce many output files. Here we explain what they contain.

2.5.1 -o/–outfile

The main output is the file of regions with an FDR score lower than the cutoff (0.05 by default).

```
# epic -cpu 25 -t examples/test.bed -c examples/control.bed -o enriched_regions.csv
Chromosome Start End ChIP Input Score Log2FC P FDR
chr1 23568400 23568599 2 0 13.54464274405703 11.30748585550573 8.184731894908448e-11␣
↪6.319374393278151e-10
chr1 26401200 26401399 2 0 13.54464274405703 11.30748585550573 8.184731894908448e-11␣
↪6.319374393278151e-10
chr1 33054800 33055399 2 0 14.288252844518933 9.722523354784574 2.207263610333191e-09␣
↪3.85690272963484e-09
chr1 33365200 33365399 2 0 13.54464274405703 11.30748585550573 8.184731894908448e-11␣
↪6.319374393278151e-10
chr1 39422200 39422799 2 0 14.288252844518933 9.722523354784574 2.207263610333191e-09␣
↪3.85690272963484e-09
chr1 51473600 51474399 2 0 14.288252844518933 9.30748585550573 5.228937118152232e-09␣
↪6.861688234097e-09
chr1 58785200 58786199 2 0 14.288252844518933 8.985557760618368 1.0206726421638307e-
↪08 1.1002055753194539e-08
chr1 59430000 59430199 2 0 13.54464274405703 11.30748585550573 8.184731894908448e-11␣
↪6.319374393278151e-10
```

The first two lines are special and do not contain any data:

- The first line contains the command invoked
- The second line contains the column names

The column names are:

- **Chromosome, Start, End**

This is the location of the region.

- **ChIP, Input**

These two columns contain the number of ChIP-reads and Input-reads within the region.

- **Score**

(Mostly uninteresting to end users). This is the region Poisson score.

- **Fold_change**

The log2_fold change is the number of ChIP reads divided by the number of Input reads in the region (where a pseudocount is computed for regions with no input-reads.)

- **P**

This is the p-value, based on the Poisson-distribution.

- **FDR**

FDR is the p-value adjusted for multiple testing with Benjamini-Hochberg.

2.5.2 -b/–bed (optional)

This is a bed file of the regions. This file is intended for downstream analyses with tools like bedtools or for display in the UCSC genome browser. (As opposed to the *–output* file which contains more statistical info and the read counts for a region and is not usable as-is by most tools.)

The three first columns are the region, the fourth is the FDR score (same as in the –outfile), and the fifth contains the log2 fold change * 100 capped at 1000. The sixth contains the strand, but ChIP-Seq data is not stranded.

```
chr1        23568400        23568599        6.319374393278151e-10   1000.0  .
chr1        26401200        26401399        6.319374393278151e-10   1000.0  .
chr1        33054800        33055399        3.85690272963484e-09    1000.0  .
chr1        33365200        33365399        6.319374393278151e-10   1000.0  .
chr1        39422200        39422799        3.85690272963484e-09    1000.0  .
chr1        51473600        51474399        6.861688234097e-09      1000.0  .
chr1        58785200        58786199        1.1002055753194539e-08  1000.0  .
chr1        59430000        59430199        6.319374393278151e-10   1000.0  .
chr1        65065600        65066199        3.85690272963484e-09    1000.0  .
chr1        91625400        91625799        1.8569048582126885e-09  1000.0  .
```

2.5.3 -sm/–store-matrix

This option gives a matrix of counts for each genomic bin (tile/window) for all the bed/bedpes given to epic. It is intended for downstream statistical analyses. It is gzipped since it can be enormously big.

- **Chromosome**, **Bin**

  The two first columns give the location of the bin (window).

- **Enriched**

  This column tells whether the bin was part of an enriched region (according to the FDR cutoff given to epic). 0 means no and 1 means yes.

- **chip1.bed**, **input1.bed**, . . . , **chipn.bed**, **inputn.bed**

  The next columns are the files used in the epic analysis and contain the read-count for each bin. In the example below the files used are examples/test.bed and examples/control.bed so this is what the columns are called too.

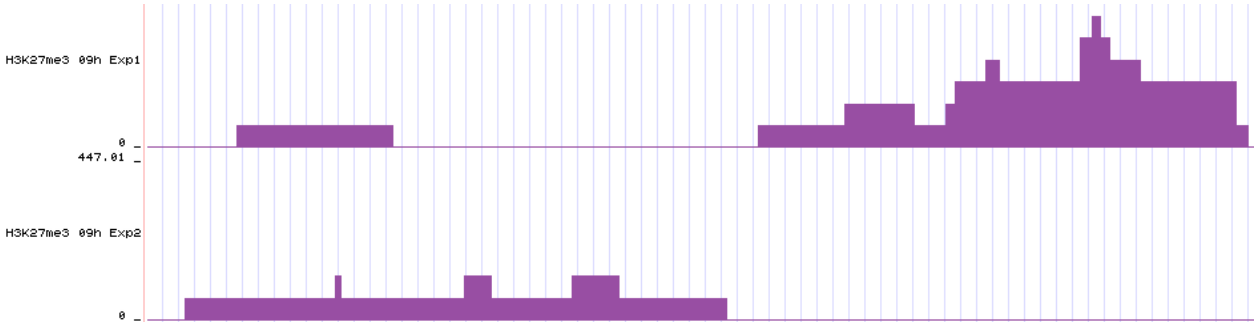
```
epic -t examples/test.bed -c examples/control.bed -sm matrix.gz > /dev/null
zcat matrix.gz  | head # gzcat on some mac OSes
# Chromosome Bin Enriched examples/test.bed examples/control.bed
# chr1 887600 0 0 1
# chr1 994600 0 0 1
# chr1 1041000 0 0 1
# chr1 1325200 0 1 0
# chr1 1541600 0 1 0
# chr1 1599000 0 1 0
# chr1 1770200 0 0 1
# chr1 1820200 0 1 0
# chr1 1995000 0 0 1
```

2.5.4 -bw/–bigwig (optional)

This flag takes a folder to store bigwigs in. One bigwig file is created per bed/bedpe file given for epic to analyze. The scores are RPKM-normalized.

```
epic -bw bigwigs/ -t examples/test.bed -c examples/control.bed \
            -o examples/expected_results_log2fc.csv
ls bigwigs/
# control.bw test.bw
```

These bigwigs show how epic saw the data. So the data will look like a histogram where the bars are bins and the counts within a bin gives the height of the bar. The results are RPKM-normalized. Here are two bigwigs displayed in an arbitrary genomic region in the UCSC genome browser:



2.5.5 -i2bw/–individual-log2fc-bigwigs

This flag takes a folder to store bigwigs in. One bigwig file is created per ChIP bed/bedpe file given for epic to analyze. The scores are RPKM-normalized and divided by the mean of the summed input RPKM. A pseudocount of one is given to bins with no input. I is the number of Input-files in the equation below:

$$log_2 \frac{RPKM_{ChIP}}{\frac{\sum_{i=1}^{I} RPKM_{Input_i}}{I}}$$

2.5.6 -cbw/–chip-bigwig

The ChIP-bigwig creates a common bigwig for all the ChIP-Seq files. First the RPKM is computed for each bed/bedpe file, then these are added together and the *–chip-bigwig* is produced.

The value in each bin is (where C is the number of ChIP-files):

$$\frac{\sum_{i=1}^{C} RPKM_{ChIP_i}}{C}$$

2.5.7 -ibw/–input-bigwig

The Input-bigwig creates a common bigwig for all the input files. First the RPKM is computed for each bed/bedpe file, then these are added together and the –input-bigwig is produced.

The value in each bin is (where I is the number of Input-files):

$$\frac{\sum_{i=1}^{I} RPKM_{Input_i}}{I}$$

2.5.8 -2bw/–log2fc-bigwig

Sums of the RPKM-scores for each library is computed like described in *-cbw* and *-ibw*. Then a pseudocount of one is added to each bin with a count of zero in the input. Finally the summed ChIP and Input vectors are divided and then the log2 is computed.

$$log_2 \frac{\frac{\sum_{i=1}^{C} RPKM_{ChIP_i}}{C}}{\frac{\sum_{i=1}^{I} RPKM_{Input_i}}{I}}$$

2.5.9 -l/–log

Write all the logging messages to a file (in addition to stderr).

2.6 epic-merge

epic-merge is used to merge two or more output-matrixes from epic. The result is a count-matrix containing several experiments, so that producing an input-file for differential analysis with state-of-the-art tools such as limma is simplified.

By default, epic-merge only keeps the bins within a region epic considered enriched, but you can tell it to keep all bins - although this obviously is much more memory and time-consuming. epic-merge can also take one or more bed-files with regions to consider enriched instead of those found by epic. This allows you to use the epic toolsuite for differential analysis of experiments, even if you found the enriched regions with other region callers such as macs2.

The problem epic-merge solves (relatively) quickly and efficiently is that of horizontal concatenation.

2.6.1 Options

epic-merge has the following flags:

- **-m, –matrixes**

  The epic count-matrixes to be merged.

- **-r, –regions**

  One or more bed-files with regions to consider enriched. For each bin, epic-merge can keep a column with how many times (i.e. in how many files) it was considered enriched, so you should keep the peaks/regions in separate files if they indeed were from different experiments.

- **-e, –enriched-per-file**

  Whether to keep a column with info about in how many experiments a bin was considered enriched or not. The downstream tool epic-cluster can use this info.

- **-o, –output**

  Path to write the (gzipped) merged matrix to.

- **-cpu, –number-cores**

  The number of CPUs to use. Can at most use one per chromosome.

2.6.2 Example1: merging two output-matrixes from epic.

Here we show how to merge two output-matrixes from epic; one from timepoint 0 and one from timepoint 12 of a time-series experiment.

```
zcat H3K27_me3_12h.gz | head -5
# Chromosome Bin Enriched H3K27me3/Exp1_12h_H3K27me3.bed H3K27me3/Exp2_12h_H3K27me3.
↪bed Input/Exp1_12h_Input.bed Input/Exp2_12h_Input.bed
# chr1 10000 0 0 0 2 5
# chr1 10200 0 2 0 0 1
# chr1 10400 0 0 0 1 0
# chr1 13200 0 0 0 1 0
zcat H3K27_me3_0h.gz | head -5
# Chromosome Bin Enriched H3K27me3/Exp1_0h_H3K27me3.bed H3K27me3/Exp2_0h_H3K27me3.bed␣
↪Input/Exp1_0h_Input.bed Input/Exp2_0h_Input.bed
# chr1 10000 0 0 1 2 5
# chr1 10200 0 0 2 1 1
# chr1 10400 0 1 0 0 0
# chr1 13200 0 0 0 0 1

epic-merge -cpu 25 -m H3K27_me3_0h.gz H3K27_me3_12h.gz -o 0_12.gz

Chromosome Bin TotalEnriched H3K27me3/Exp1_0h_H3K27me3.bed H3K27me3/Exp1_12h_H3K27me3.
↪bed H3K27me3/Exp2_0h_H3K27me3.bed H3K27me3/Exp2_12h_H3K27me3.bed Input/Exp1_0h_
↪Input.bed Input/Exp1_12h_Input.bed Input/Exp2_0h_Input.bed Input/Exp2_12h_Input.bed
chr1 264400 1.0 1.0 1.0 5.0 3.0 0.0 0.0 0.0 0.0
chr1 264600 1.0 3.0 3.0 3.0 5.0 1.0 2.0 0.0 1.0
chr1 265200 1.0 3.0 6.0 2.0 2.0 4.0 3.0 2.0 0.0
chr1 265400 1.0 2.0 1.0 2.0 0.0 1.0 2.0 1.0 3.0
```

2.6.3 Example2: Using custom region files to extract bins.

head test1.bed # chr1 10050 10100

2.7 epic-cluster

When doing differential analysis of multiple ChIP-Seq experiments, one problem is to choose the regions that should be tested for differential enrichment. This is the problem epic-cluster tries to solve.

epic-cluster takes one merged-matrix created by the epic-merge tool and clusters the genomic bins within it to produce regions that should be tested for differential enrichment.

Several algorithms might be added to the tool; for now epic-cluster contains one, which we have called trunks, flanks and valleys.

2.7.1 Algorithm: trunks, flanks and valleys

Each matrix from epic-merge contains a column with the number of experiments in which the bin was considered part of an enriched region. In this example the matrix contains four experiments. The illustration below shows which regions were considered enriched in which experiments:

In the illustration below you see how this translates into clusters:

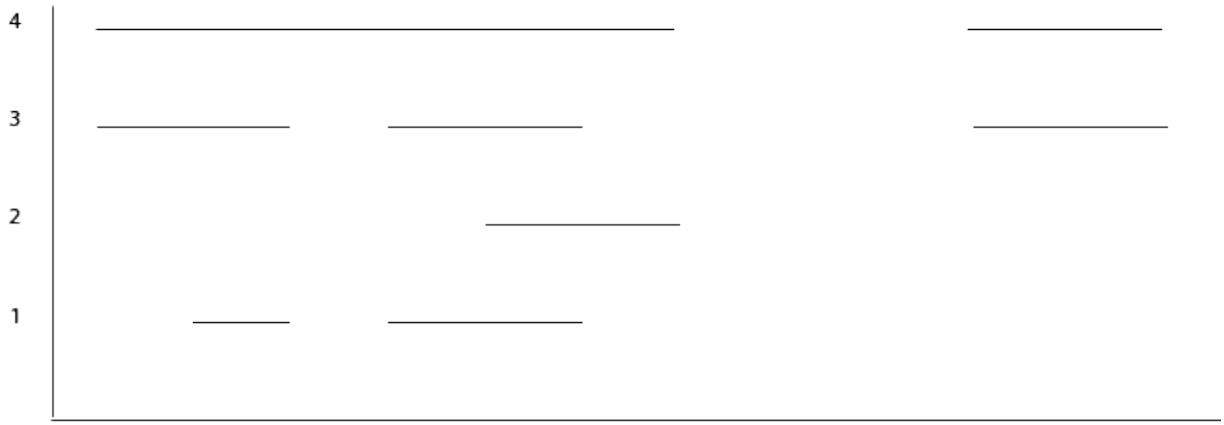


Fig. 2.1: Which regions are enriched in which experiments.

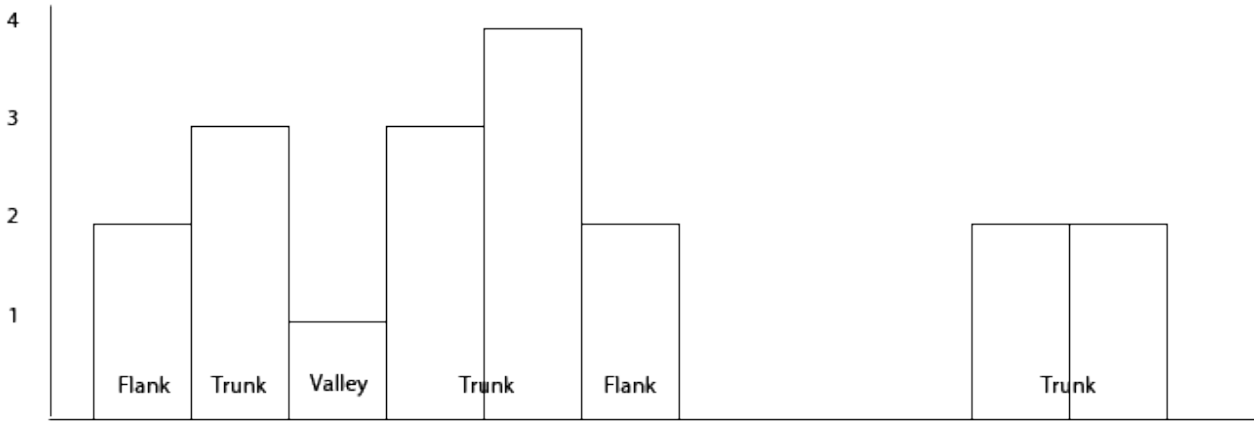


Fig. 2.2: The trunks, flanks and valleys algorithm.

In Fig. 2.2 there are two clusters of bins. The first cluster is subdivided into five regions, the second is not subdivided into any.

This algorithm first looks for genomic bins that are at most *–distance-allowed* apart. So if the bin-size is 200 and the *–distance-allowed* is 400, the data is subdivided into two clusters as seen above (since the two clusters are separated by 3 bins or 600 nucleotides).

In the first cluster the max number of enriched regions of any bin is 4 and the *–trunk-diff* has the default value, namely 1. So anything with 4 - *–trunk-diff* number of enriched regions is considered a trunk, while anything else is a flank or a valley. Flanks and valleys are regions where the total number of enriched is lower than *max_enriched - trunk-diff*. Flanks lie at the edges of a cluster, while valleys lie in-between trunks.

2.7.2 Options

- **-m, –matrix**

  A single epic-merge output matrix.

- **-o, –outfile

  Where to store the cluster matrix

- **-B, –bedfile

  Where to store the bed file with info about each cluster.

- **-t, –trunk-diff**

  The difference in number of enriched regions that decides when to consider a subregion a flank or valley, instead of a trunk. If you do not want to split clusters into flanks and valleys, you can set this to a high number.

- **-d, –distance-allowed**

  The number of nucleotides before two bins are considered to be in separate clusters.

- **-b, –bin-size**

  The bin-size used in the input matrix file. Auto-inferred by default.

- **-cpu, –number-cores**

  The number of cores to use. Can at most use one per chromosome

2.7.3 Example

We start with an epic-merge matrix that only includes two experiments:

```
zcat 0_12.gz | head
# Chromosome Bin TotalEnriched Exp1_0h.bed Exp1_12h.bed Exp2_0h.bed Exp2_12h.bed Exp1_
↪0h_Input.bed Exp1_12h_Input.bed Exp2_0h_Input.bed Exp2_12h_Input.bed
# chr1 264400 1.0 1.0 1.0 5.0 3.0 0.0 0.0 0.0 0.0
# chr1 264600 1.0 3.0 3.0 3.0 5.0 1.0 2.0 0.0 1.0
# chr1 265200 1.0 3.0 6.0 2.0 2.0 4.0 3.0 2.0 0.0
# chr1 265400 1.0 2.0 1.0 2.0 0.0 1.0 2.0 1.0 3.0
# chr1 265800 1.0 7.0 12.0 6.0 7.0 0.0 2.0 4.0 3.0
# chr1 268400 1.0 5.0 5.0 6.0 5.0 6.0 3.0 3.0 3.0
# chr1 268600 1.0 2.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
# chr1 269000 1.0 4.0 1.0 0.0 2.0 1.0 3.0 1.0 0.0
# chr1 269200 1.0 0.0 1.0 3.0 2.0 0.0 0.0 2.0 1.0
```

Since it only includes two experiments, the TotalEnriched column can only include 1 or 2 (which means the bin was enriched in 1 or 2 experiments)[1].

Since the TotalEnriched-column is either 1 or 2, we set the *-trunk-diff* option to 0, which means that for a region where the max TotalEnriched is 2, a count of 2 will be considered a trunk, while 1 will be a flank or a valley. For a region where the max TotalEnriched is 1, everything is a trunk.

```
epic-cluster -cpu 25 -t 0 -m 0_12.gz -o 0_12_cluster.gz
zcat 0_12_cluster.gz
```

These are what the results look like:

```
zcat 0_12_cluster.gz | head
Index Exp1_0h.bed Exp1_12h.bed Exp2_0h.bed Exp2_12h.bed Exp1_0h_Input.bed Exp1_12h_
↪Input.bed Exp2_0h_Input.bed Exp2_12h_Input.bed
# chr1_0_264400:264799_trunk 4.0 4.0 8.0 8.0 1.0 2.0 0.0 1.0
# chr1_1_265200:265599_trunk 5.0 7.0 4.0 2.0 5.0 5.0 3.0 3.0
# chr1_2_265800:265999_trunk 7.0 12.0 6.0 7.0 0.0 2.0 4.0 3.0
# chr1_3_268400:268799_trunk 7.0 5.0 6.0 5.0 6.0 3.0 3.0 3.0
# chr1_4_269000:269599_trunk 8.0 6.0 6.0 5.0 2.0 7.0 4.0 1.0
# chr1_5_862200:865199_trunk 19.0 27.0 20.0 35.0 11.0 12.0 10.0 8.0
# chr1_6_882000:884599_flank 25.0 22.0 16.0 27.0 17.0 15.0 8.0 13.0
# chr1_6_884600:887399_trunk 23.0 35.0 30.0 43.0 19.0 12.0 15.0 15.0
# chr1_6_887400:887599_flank 1.0 2.0 2.0 4.0 0.0 2.0 1.0 1.0
```

Since most tools for differential analysis are written in R, which only accepts one index column, all the data about the region is squashed into the first column.

2.8 epic-count

epic-count creates a simple matrix of counts from the .bed/.bedpe files it is given. These matrixes do not contain information about enrichment.

- **-i, –infiles**

  One or more bed/bedpe files to count reads in.

- **-o, –outfile**

  File to write results to. By default sent to stdout.

- **-cpu, –number-cores**

  The number of cores epic should use. Can at most take advantage of 1 core per strand per chromosome (i.e. 46 for humans). Default: 1

- **-gn, –genome**

  Which genome to analyze. By default hg19.

- **-k, –keep-duplicates**

  Keep reads mapping to the same position on the same strand within a library. The default is to remove all but the first duplicate (this is done once per file, not for all files collectively.)

- **-fs, –fragment-size**

[1] If epic-merge was used with the option *–keep-nonenriched*, the TotalEnriched column can also include 0.

(Only used for single-end files) Size of the sequenced fragment. The center of the fragment will be used to calculate which window a read ended up in. So reads are shifted by fragment-size/2. Default 150.

- **-cs, –chromsizes**

  Set the chromosome lengths yourself in a file with two columns: chromosome names and sizes. Useful to analyze custom genomes, assemblies or simulated data. Only chromosomes included in the file will be analyzed.

2.9 epic-blacklist

epic-blacklist takes one or more ChIP-files that contain data unrelated to the experiment (ChIP from another species for example) and finds bins where a lot of reads still bind. Bins with a statistically significant number of reads are computed according to a Poisson model. These bins are written as a bed-file to stdout.

- **-i, –infiles**

  One or more bed/bedpe files to count reads in.

- **-o, –outfile**

  File to write results to. By default sent to stdout.

- **-cpu, –number-cores**

  The number of cores epic should use. Can at most take advantage of 1 core per strand per chromosome (i.e. 46 for humans). Default: 1

- **-gn, –genome**

  Which genome to analyze. By default hg19.

- **-k, –keep-duplicates**

  Keep reads mapping to the same position on the same strand within a library. The default is to remove all but the first duplicate (this is done once per file, not for all files collectively.)

- **-fs, –fragment-size**

  (Only used for single-end files) Size of the sequenced fragment. The center of the fragment will be used to calculate which window a read ended up in. So reads are shifted by fragment-size/2. Default 150.

- **-cs, –chromsizes**

  Set the chromosome lengths yourself in a file with two columns: chromosome names and sizes. Useful to analyze custom genomes, assemblies or simulated data. Only chromosomes included in the file will be analyzed.

- **-f, –fdr**

  Cut-off to consider a bin enriched (Default: 0.05)

- **-egf, –effective-genome-fraction**

  Use a different effective genome fraction than the one included in epic. Or include an egf for custom genomes that are not a part of epic. Should be a number between 0 and 1. Autoinferred by sampled read-length and genome by default.

CHAPTER 3

Indices and tables

- genindex
- modindex
- search