

Normalization of BAM files

deepTools contains 3 tools for the normalization of [BAM](#) files:

1. **correctGCBias**: in case, you would like to normalize your read distributions to fit the expected GC values, you can use the output from [computeGCBias](#) and produce a GC-corrected [BAM](#)-file.
2. **bamCoverage**: this tool converts a single [BAM](#) file into a [bigWig](#) file, enabling you to normalize for sequencing depth.
3. **bamCompare**: like bamCoverage, this tool produces a normalized [bigWig](#) file, but it takes 2 [BAM](#) files, normalizes them for sequencing depth and subsequently performs a mathematical operation of your choice, i.e. it can output the ratio of the read coverages in both files or the like.

[Here](#) you can download slides that we used for teaching. They contain additional details about how the coverage files are generated and normalized.

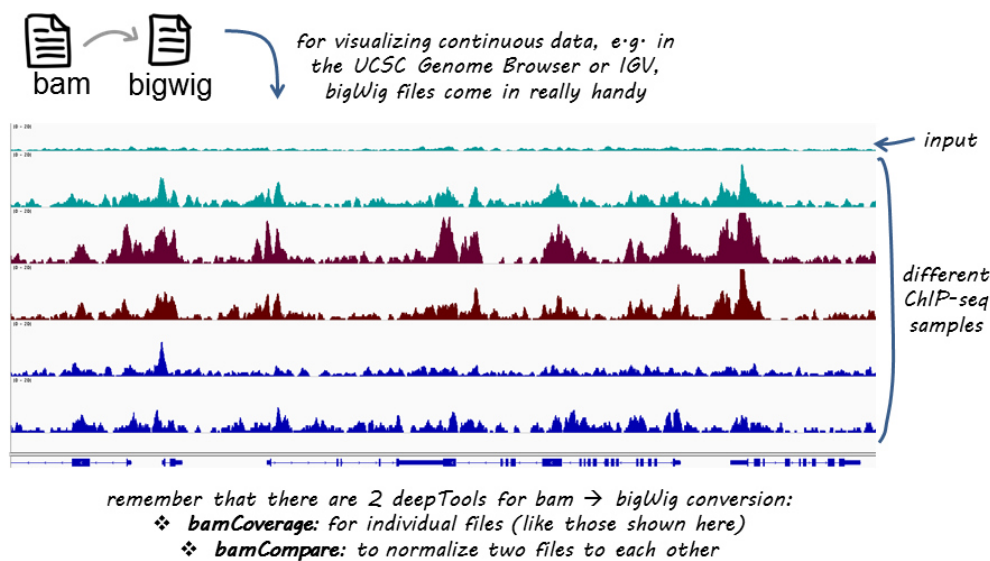


Table of Content

- [correctGCBias](#)
- [bamCoverage](#)
- [bamCompare](#)

CorrectGCbias

What it does (uses output from computeGCbias)

This tool requires the output from [computeGCbias](#) to correct the given [BAM](#) files according to the method proposed by [Benjamini and Speed](#). The resulting [BAM](#) files can be used in any downstream analyses, but **be aware that you should not filter out duplicates from here on**.

output

- GC-normalized BAM file

bamCoverage

What it does

Given a BAM file, this tool generates a [bigWig](#) or [bedGraph](#) file of fragment or read coverages. The way the method works is by first calculating all the number of reads (either extended to match the fragment length or not) that overlap each bin in the genome. Bins with zero counts are skipped, i.e. not added to the output file. The resulting read counts can be normalized using either a given scaling factor, the RPKM formula or to get a 1x depth of coverage (RPGC).

output

- **coverage file** either in [bigWig](#) or [bedGraph](#) format

Usage

Here's an exemplary command to generate a single [bigWig](#) file out of a single [BAM](#) file via the command line:

```
$ /deepTools-1.5/bin/bamCoverage --bam corrected_counts.bam --binSize 10 --normalizeTo1x 2150570000 --fragmentLength 200 -o Coverage.GCcorrected.SeqDepthNorm.bw --ignoreForNormalization chrX
```

- The bin size (**-bs**) can be chosen completely to your liking. The smaller it is, the bigger your file will be.
- This was a mouse sample, therefore the effective genome size for mouse had to be indicated once it was decided that the file should be normalized to 1x coverage.
- Chromosome X was excluded from sampling the regions for normalization as the sample was from a male mouse that therefore contained pairs of autosomes, but only a single X chromosome.
- The fragment length of 200 bp is only the fall-back option of bamCoverage as the sample provided here was done with paired-end sequencing. Only in case of singletons will bamCoverage resort to the user-specified fragment length.
- **--ignoreDuplicates** - important! in case where you normalized for GC bias using correctGCbias, you should absolutely **NOT** set this parameter

Using [deepTools Galaxy](#), this is what you would have done:

bamCoverage (version 1.0.2)

BAM file:

The BAM file must be sorted.

Length of the average fragment size:

Reads will be extended to match this length unless they are paired-end, in which case extended. *Warning* the fragment length affects the normalization to 1x (see "normalization length"). *NOTE*: If the BAM files contain mated and unmated paired-end reads, unmated reads will be ignored.

Bin size in bp:

The genome will be divided in bins (also called tiles) of the specified length. For each bin, the coverage is calculated as the sum of reads overlapping the bin.

Scaling/Normalization method:

Genome size:

Enter the genome size to normalize the reads counts. Sequencing depth is defined as the number of reads per base. Common values are: mm9: 2150570000, hg19:2451960000, dm3:1214000000.

Coverage file format:

Show advanced options:

bamCompare

What it does

This tool compares two BAM files based on the number of mapped reads. To compare the BAM files, the genome is partitioned into bins of equal size, the reads are counted for each bin and each BAM file and finally, a summarizing value is reported. This value can be the ratio of the number of reads per bin, the log2 of the ratio or the difference. This tool can normalize the number of reads on each BAM file using the SES method proposed by [Diaz et al.](#) Normalization based on read counts is also available. If paired-end reads are present, the fragment length reported in the BAM file is used by default.

output file

- same as for bamCoverage, except that you now obtain **1** coverage file that is based on **2** BAM files.

Usage

Here's an example command that generated the log2(ChIP/Input) values via the command line.

```
$ /deepTools-1.5/bin/bamCompare --bamfile1 ChIP.bam -bamfile2 Input.bam --binSize 25 --fragmentLength 200 --missingDataAsZero no --ratio log2 --scaleFactorsMethod SES -o log2ratio_ChIP_vs_Input.bw
```

The Galaxy equivalent:

bamCompare (version 1.0.2)

Treatment BAM file:

1: IMR90_H3K27ac_SRX012496.bam

The BAM file must be sorted.

BAM file:

3: IMR90_Input_SRX017548.bam

The BAM file must be sorted.

Length of the average fragment size:

200

Reads will be extended to match this length unless they are paired-end, in which case they will be extended to match the fragment length. *Warning* the fragment length affects the normalization to 1x (see "normalize coverage to 1x"). The formula to normalize coverage is: $\text{coverage} \times \text{fragment length} = 1$. *NOTE*: If the BAM files contain mated and unmated paired-end reads, unmated reads will be extended to match the fragment length.

Bin size in bp:

25

The genome will be divided in bins (also called tiles) of the specified length. For each bin the overlapping number of fragments (or reads) is counted.

Method to use for scaling the largest sample to the smallest:

signal extraction scaling (SES)

Length in base pairs used to sample the genome and compute the size or scaling factors to compare the two BAM files :

1000

The default is fine. Only change it if you know what you are doing

How to compare the two files:

compute log2 of the number of reads ratio

Coverage file format:

bigwig

Show advanced options:

no

Execute

Note that the option "missing Data As Zero" can be found within the "advanced options" (default: no).

- like for bamCoverage, the bin size is completely up to the user
- the fragment size (-f) will only be taken into consideration for reads without mates

- the SES method was used for normalization as the ChIP sample was done for a histone mark with highly localized enrichments (similar to the left-most plot of the [fingerprint-examples](#))

Some (more) parameters to pay special attention to

- `--scaleFactorsMethod` (in Galaxy: "Method to use for scaling the largest sample to the smallest") - here you can choose how you would like to normalize to account for variation in sequencing depths. We provide the simple normalization total read count or the more sophisticated signal extraction (SES) method proposed by [Diaz et al.](#) **We recommend to use SES only for those cases where the distinction between input and ChIP is very clear in the [bamFingerprint plots](#).** This is usually the case for transcription factors and sharply defined histone marks such as H3K4me3.
 - `--ratio` (in Galaxy: "How to compare the two files") - here you get to choose how you want the two input files to be compared, e.g. by taking the ratio or by subtracting the second BAM file from the first BAM file etc. In case you do want to subtract one sample from the other, you will have to choose whether you want to normalize to 1x coverage (`--normalizeTo1x`) or to reads per kilobase (`--normalizeUsingRPKM`; similar to RNA-seq normalization schemes)

many options = many choices to make



<http://romebusinessschool.it/en/decision-making-efficace/>

This tool is developed by the [Bioinformatics Facility](#) at the [Max Planck Institute for Immunobiology and Epigenetics](#), Freiburg.