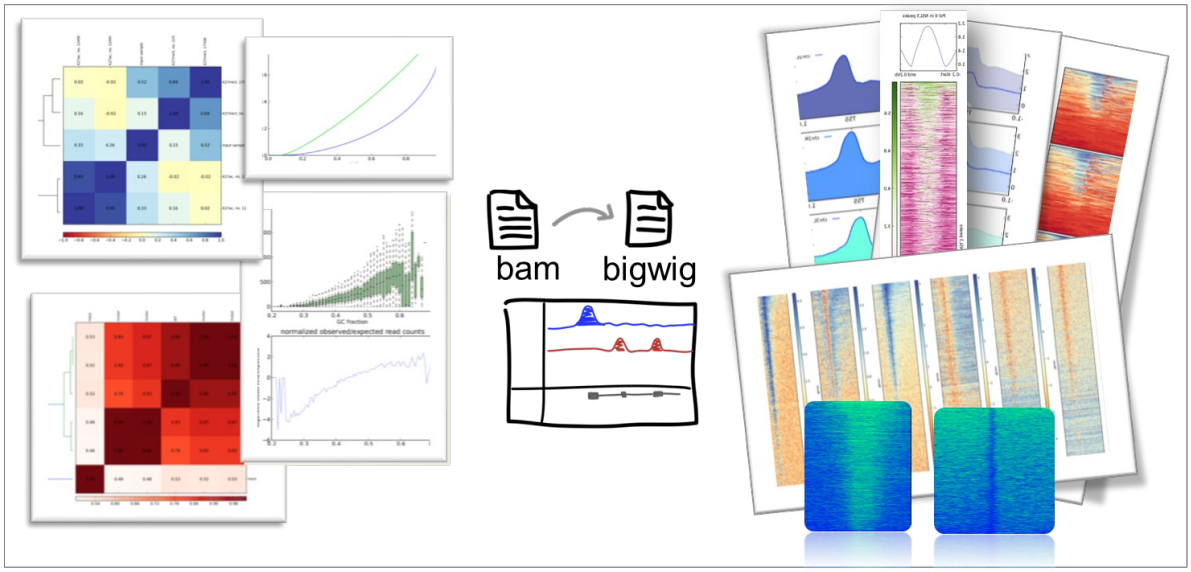
deepTools: a flexible platform for exploring deep-sequencing data

MANUAL



Fidel Ramírez, Friederike Dündar, Sarah Diehl, Björn A. Grüning, Thomas Manke

Bioinformatics Group, Max-Planck-Institute of Immunobiology and Epigenetics & Department of Computer Science, University of Freiburg

Web server (incl. sample data): deepTools.ie-freiburg.mpg.de
Code: github.com/fidelram/deepTools

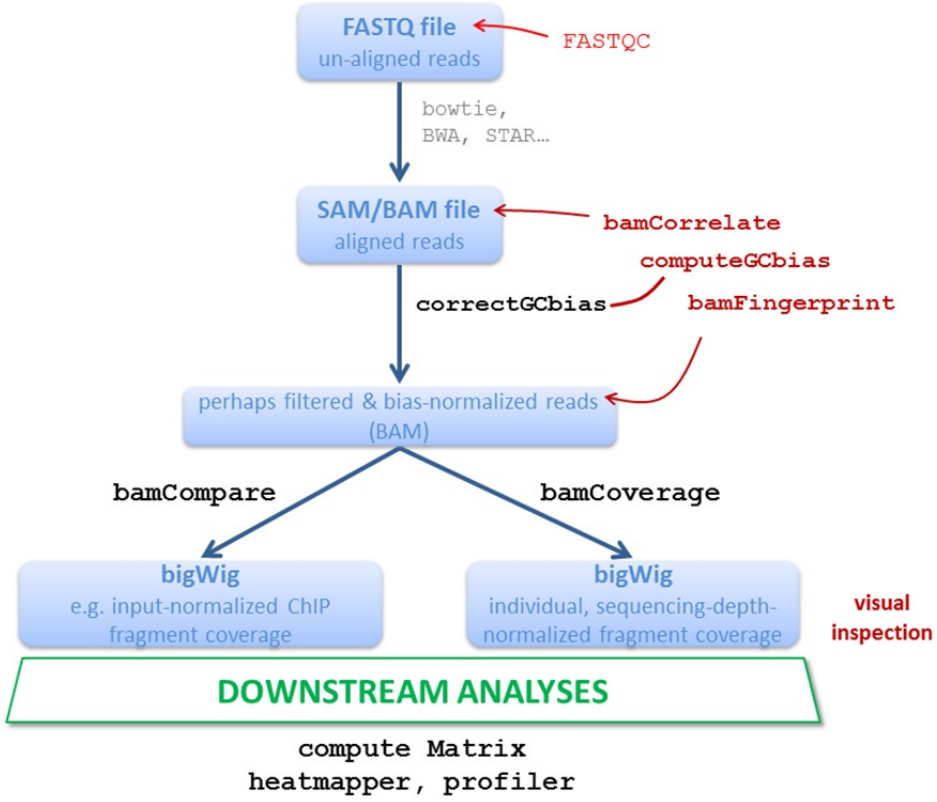
This document contains some chapters of our wiki on deepTools usage for NGS data analysis. For the most updated version of our help site and for **more information about deepTools, a brief introduction into Galaxy** as well as **step-by-step protocols**, please visit: https://github.com/fidelram/deepTools/wiki

Why we built deepTools

The main reason why deepTools was started is the simple fact that in 2011 we could not find tools that met all our needs for NGS data analysis. While there were individual tools for separate tasks, we wanted software that would fulfill *all* of the following criteria:

- **efficiently extract reads from BAM files** and perform various computations on them
- **turn BAM files of aligned reads into bigWig files** using different normalization strategies
- make use of **multiple processors** (speed!)
- generation of **highly customizable images** (change colors, size, labels, file format etc.)
- enable **customized down-stream analyses** which requires that every data set that is being produced can be stored by the user
- **modular approach** - compatibility, flexibility, scalability (i.e. we can add more and more modules making use of established methods)

The flow chart below depicts the different tool modules that are currently available within deepTools (deepTools modules are written in bold red and black font). If you are not familiar with the file names shown here, please see our Glossary at the end of the document for more information.



How we use deepTools

You will find many examples from ChIP-seq analyses in this tutorial, but this does not mean that deepTools is restricted to ChIP-seq data analysis. However, some tools, such as *bamFingerprint* specifically address ChIP-seq-issues. (That being said, we do process quite a bit of RNA-seq, other -seq and genomic sequencing data using deepTools, too, but many normalization issues arose during handling of ChIP-seq data).

As shown in the flow chart above, our work usually begins with one or more FASTQ file(s) of deeply-sequenced samples. After a first quality control using FASTQC, we align the reads to the reference genome, e.g. using bowtie2][].
We then use deepTools to assess the quality of the aligned reads:

1. **Correlation between BAM files** (*bamCorrelate*). This is a very basic test to see whether the sequenced and aligned reads meet your expectations. We use this check to assess the reproducibility - either between replicates and/or between different experiments that might have used the same antibody/the same cell type etc. For instance, replicates should correlate better than differently treated samples.
2. **GC bias check** (*computeGCbias*). Many sequencing protocols require several rounds of PCR-based amplification of the DNA to be sequenced. Unfortunately, most DNA polymerases used for PCR introduce significant GC biases as they prefer to amplify GC-rich templates. Depending on the sample (preparation), the GC bias can vary significantly and we routinely check its extent. In case we need to compare files with different GC biases, we use the *correctGCbias* module to match the GC bias. See the paper by Benjamini and Speed for many insights into this problem.
3. **Assessing the ChIP strength**. We do this quality control do to get a feeling for the signal-to-noise ratio in samples from ChIP-seq experiments. It is based on the insights published by Diaz et al..

Once we are satisfied by the basic quality checks, we normally **convert the large BAM files into a leaner data format, typically bigWig**. bigWig files have several advantages over BAM files that mainly stem from their significantly decreased size:

- useful for data sharing & storage
- intuitive visualization in Genome Browsers (e.g. IGV)
- more efficient downstream analyses are possible

The deepTools modules *bamCompare* and *bamCoverage* do not only allow the simple conversion from BAM to bigWig (or bedGraph for that matter), **the main reason why we developed those tools was that we wanted to be able to *normalize* the read coverages** so that we could compare different samples despite differences in sequencing depth, GC biases and so on.

Finally, once all the files have passed our visual inspections, the fun of downstream analyses with *heatmapper* and *profiler* can begin!

deepTools overview

deepTools consists of a set of modules that can be used independently to work with mapped reads. We have subdivided such tasks into *quality controls* (QC), *normalizations* and *visualizations*.

Here's a concise summary of the tools. In the following pages, you can find more details about the individual tools. We have included many screenshots of our Galaxy deepTools web server to explain the usage of our tools. In addition, we show the commands for the stand-alone usage, as they often indicate the options that one should pay attention to more succinctly.

tool	type	input files	main output file(s)	application
bamCorrelate	QC	2 or more BAM	clustered heatmap	Pearson or Spearman correlation between read distributions
bamFingerprint	QC	2 BAM	1 diagnostic plot	assess enrichment strength of a ChIP sample
computeGCbias	QC	1 BAM	2 diagnostic plots	calculate the expected and observed GC distribution of reads
correctGCbias	QC	1 BAM, output from computeGCbias	1 GC-corrected BAM	obtain a BAM file with reads distributed according to the genome's GC content
bamCoverage	normalization	BAM	bedGraph or bigWig	obtain the normalized read coverage of a single BAM file
bamCompare	normalization	2 BAM	bedGraph or bigWig	normalize 2 BAM files to each other using a mathematical operation of your choice (e.g. log2ratio, difference)
computeMatrix	visualization	1 bigWig, 1 BED	gzipped table, to be used with heatmapper or profiler	compute the values needed for heatmaps and summary plots
heatmapper	visualization	computeMatrix output	heatmap of read coverages	visualize the read coverages for genomic regions
profiler	visualization	computeMatrix output	summary plot	visualize the average read coverages over a group of genomic regions

QC of aligned reads

These tools work on BAM files that contain read-related information (e.g. read DNA sequence, sequencing quality, mapping quality etc.). They are typically generated by read alignment programs such as bowtie2.

The following tools will allow you to inspect your BAM files more closely.

bamCorrelate

This tool is useful to assess the overall similarity of different BAM files. A typical application is to check the correlation between replicates or published data sets, but really, you can apply it to any inquiry that boils down to the question: "How (dis)similar are these BAM files?".

What it does

bamCorrelate computes the overall similarity between **two or more** BAM files based on read coverage (number of reads) within genomic regions, i.e. for each *pair* of BAM files reads overlapping with the same genomic intervals are counted and the counts are correlated. The result is a table of correlation coefficients that will be visualized as a heatmap. The correlation coefficient indicates how "strong" the relationship between the two samples is and it will consist of numbers between -1 and 1. (-1 indicates perfect anticorrelation, 1 perfect correlation.)

We offer two different functions for the correlation computation: Pearson or Spearman. In short, Pearson is an appropriate measure for data that follows a normal distribution, while Spearman does not make this assumption and is generally less driven by outliers, but with the caveat of also being less sensitive.

NOTE: bamCorrelate usually takes a long time to finish, thus it is advisable to first run the tool for a tiny region (using the --region option) to adjust plotting parameters like colors and labels before running the
whole computation.

Important parameters

bamCorrelate can be run in 2 modes: *bins* and *bed*.

In the *bins* mode, the correlation is based on read coverage over consecutive bins of equal size (10k bp by default). This mode is useful to assess the overall similarity of BAM files. The bin size and the distance between bins can be adjusted.

Note that by default, a filtering of extremes is done, when *bins* mode is selected.

In the *BED-file option*, the user supplies a list of genomic regions in BED format in addition to the BAM files. bamCorrelate subsequently uses this list to compare the read coverages for these regions only. This can be used, for example, to compare the ChIP-seq coverages of two different samples for a set of peak regions.

In addition to specifying the regions for which the read numbers should be compared (random regions in the bins mode, selected regions in the BED-file mode), you can also specify what kind of correlation measure you would like to compute: Pearson or Spearman. In short, Pearson is an appropriate measure for data that follows a normal distribution while Spearman does not make this assumption and is generally less driven by outliers. As genome-wide sequencing data very rarely follows a normal distribution and we often encounter few regions that capture extremely high read counts (= outlier), we tend to prefer the Spearman correlation coefficient.

Output files:

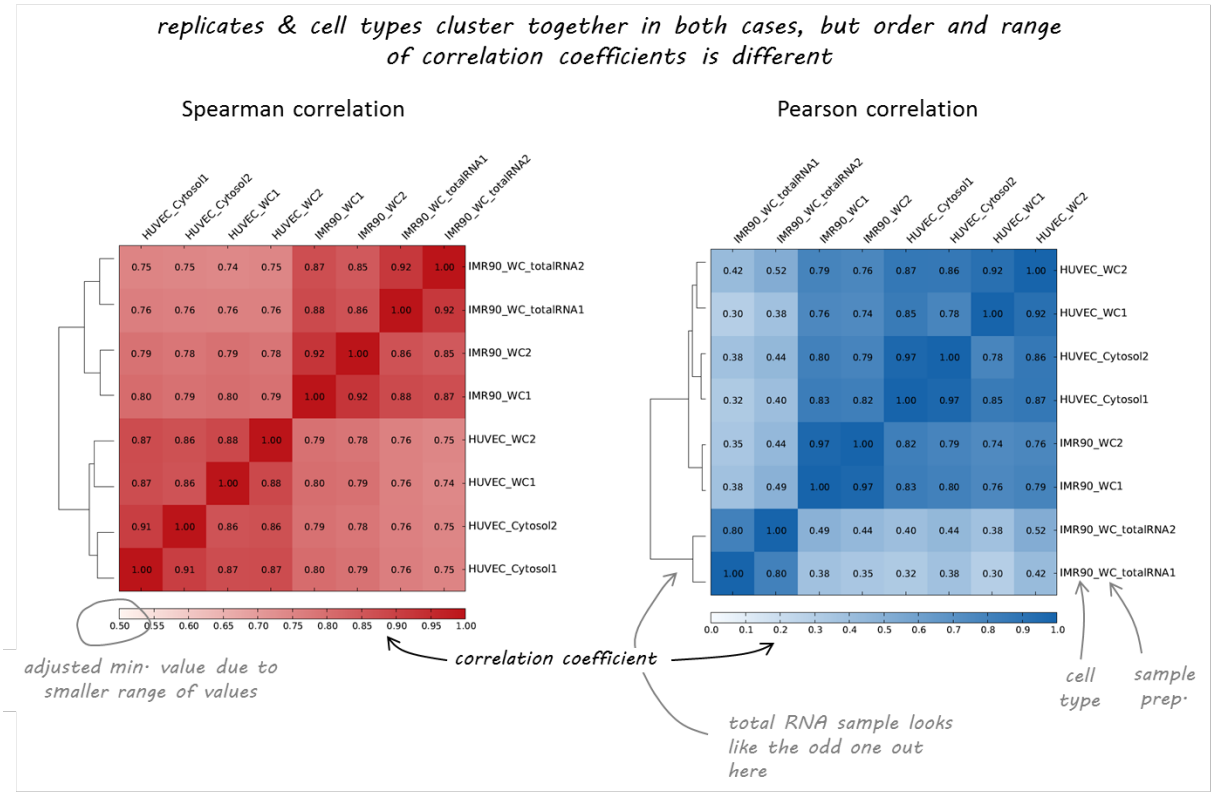
- **diagnostic plot** the plot produced by bamCorrelate is a clustered heatmap displaying the values for each pair-wise correlation, see below for an example
- **data matrix** (optional) in case you want to plot the correlation values using a different program, e.g. R, this matrix can be used

Example Figures

Here is a result of running bamCorrelate: heatmaps where the pairwise correlation coefficients are depicted by varying color intensities and are clustered using hierarchical clustering.

For the two example plots below, we supplied BAM files of RNA-seq data from different human cell lines that we had downloaded from the ENCODE project and a list of genes from RefSeq (Note that you can supply any number of BAM files that you would like to compare. In Galaxy, you just click "Add BAM file", in the command line you simply list all files one after the other, giving meaningful names via the --label option). We then calculated the pair-wise correlations of read numbers for the different genes, once with Spearman correlation, once with Pearson correlation.

You can find the original file at http://hgdownload.cse.ucsc.edu/goldenpath/hg19/database/ (just add the file names you see in the command at the end).



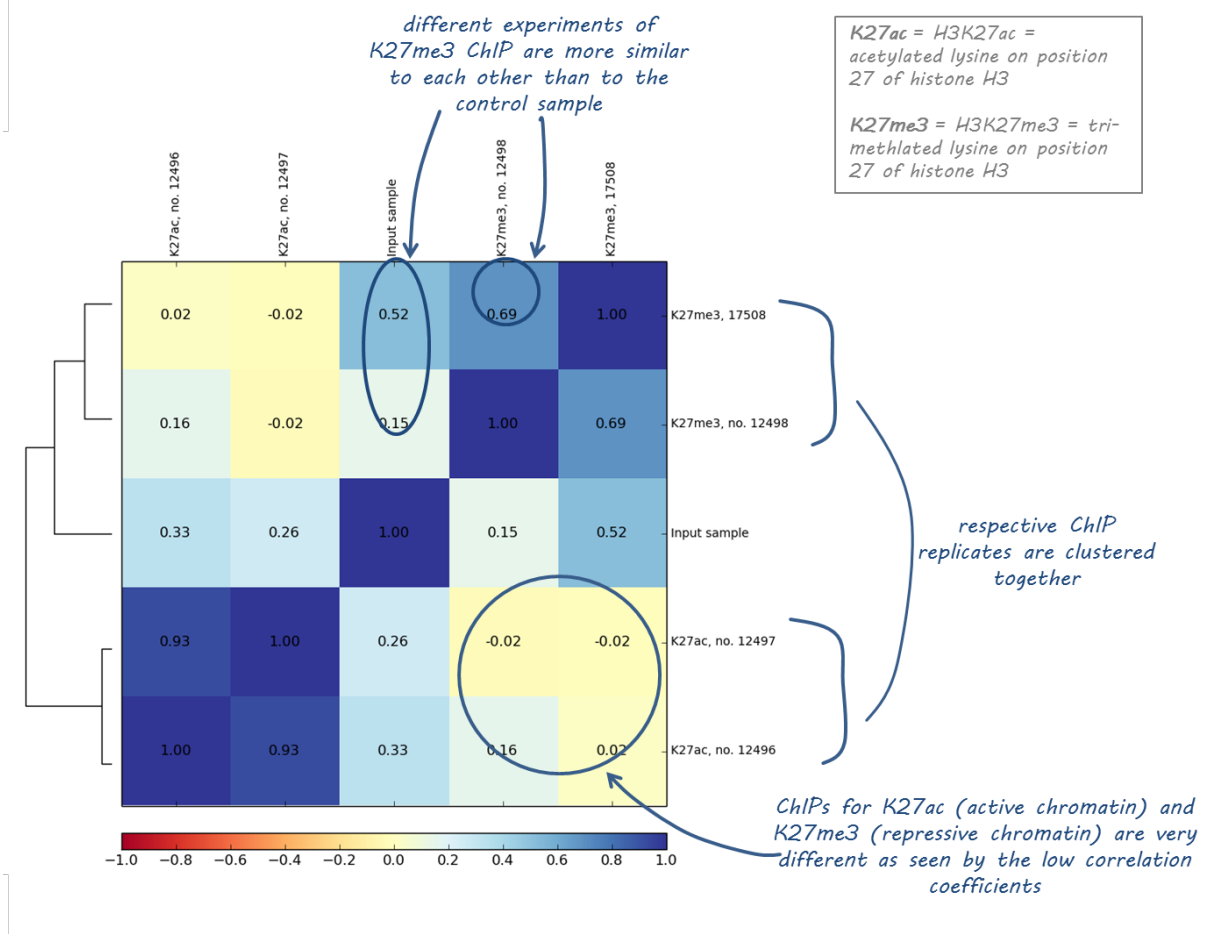
As you can see, both correlation calculations more or less agree which samples are nearly identical (the replicates, indicated by 1 or 2 at the end of the label). The Spearman correlation, however, seems to be more robust and meets our expectations more closely as the two different cell types (HUVEC and IMR90) are clearly separated.

This is the command that was used to generate the plot on the left-hand side:

```
$ deepTools-1.5.7/bin/bamCorrelate BED-file \
--BED RefSeq_Genes.bed \
--bamfiles wgEncodeCshlLongRnaSeqImr90CellPapAlnRep1.bam \
  wgEncodeCshlLongRnaSeqImr90CellPapAlnRep2.bam  \
  wgEncodeCshlLongRnaSeqImr90CellTotalAlnRep1.bam   \
  wgEncodeCshlLongRnaSeqImr90CellTotalAlnRep2.bam \
  wgEncodeCshlLongRnaSeqHuvecCellPapAlnRep1.bam \
  wgEncodeCshlLongRnaSeqHuvecCellPapAlnRep2.bam \
  wgEncodeCshlLongRnaSeqHuvecCytosolPapAlnRep3.bam \
  wgEncodeCshlLongRnaSeqHuvecCytosolPapAlnRep4.bam \
--labels IMR90_WC1 IMR90_WC2 IMR90_WC_totalRNA1 \
IMR90_WC_totalRNA2 HUVEC_WC1 HUVEC_WC2 HUVEC_Cytosol1 HUVEC_Cytosol2 \
--binSize 1000 --corMethod spearman -f 200 \
--colorMap Reds --zMin 0.5 --zMax 1 -o correlation_spearman.pdf
```

Here is another example of ChIP samples for two different histone marks (the histone marks are abbreviated H3K27me3 and H3K27ac and have been shown to mark inactive and active chromatin, respectively. For our example, H3K27ac was ChIPed by the same experimentator for different cell populations while H3K27me3 was performed with the same antibody, but at different times. You can see that the correlation between the H3K27ac replicates is much higher than for the H3K27me3 samples, however, for both histone marks, the ChIP-seq experiments are more similar to each other than to the other ChIP or to the input. In fact, the signals of H3K27ac and H3K27me3 are almost not correlated at all which supports the notion that their

biological function is also quite opposing.



computeGCbias

This tool computes the GC bias using the method proposed by Benjamini and Speed.

What it does

The basic assumption of the GC bias diagnosis is that an ideal sample should show a uniform distribution of sequenced reads across the genome, i.e. all regions of the genome should have similar numbers of reads, regardless of their base-pair composition. In reality, the DNA polymerases used for PCR-based amplifications during the library preparation of the sequencing protocols prefer GC-rich regions. This will influence the outcome of the sequencing as there will be more reads for GC-rich regions just because of the DNA polymerase's preference.

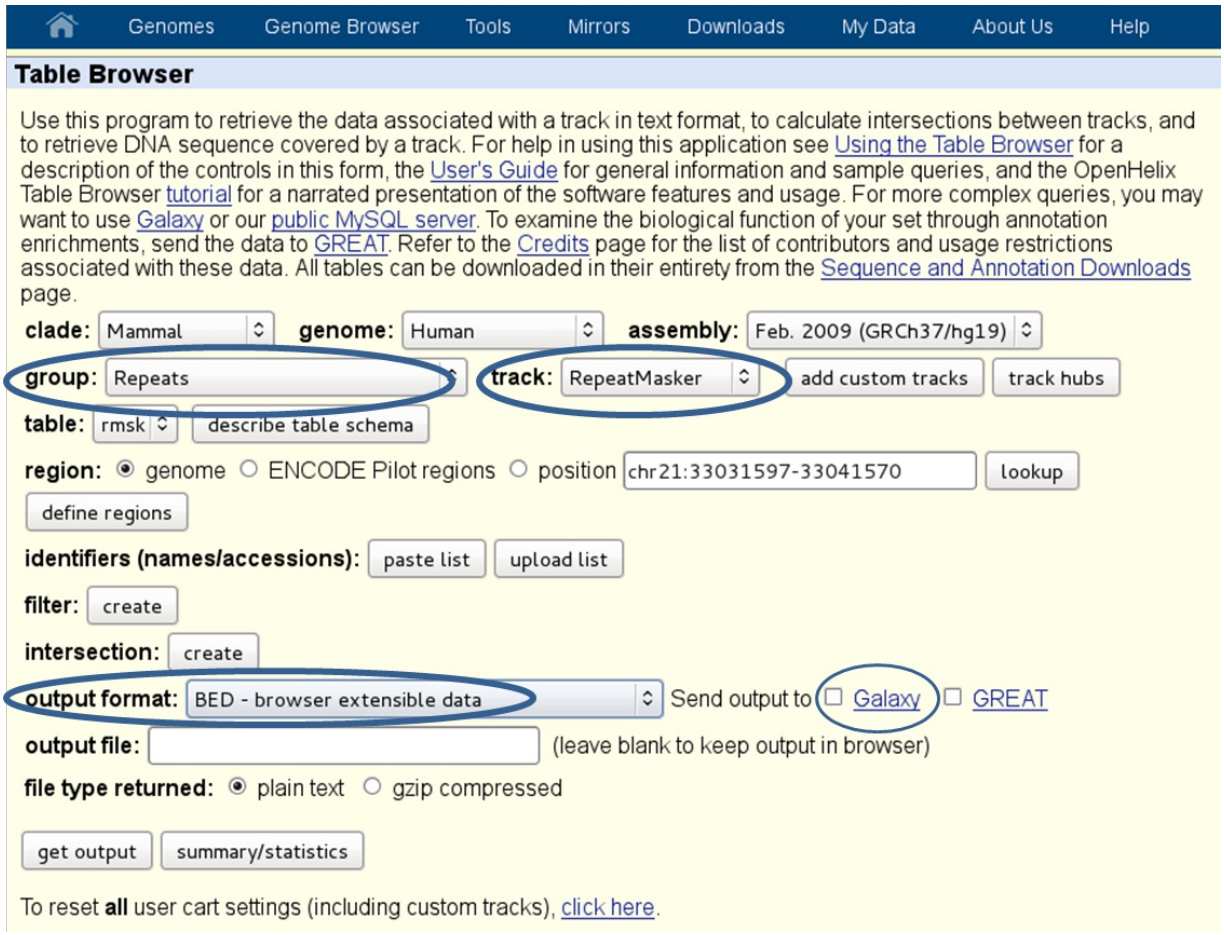
computeGCbias will **first calculate the *expected* GC profile** by counting the number of DNA fragments of a fixed size per GC fraction (GC fraction is defined as the number of G's or C's in a genome region of a given length)(a). This profile is then **compared to the *observed* GC profile** by counting the number of sequenced reads per GC fraction.

(a) The expected GC profile depends on the reference genome as different organisms have very different GC contents. For example, one would expect more fragments with GC fractions between 30% to 60% in mouse samples (GC content of the mouse genome: 45 %) than for genome fragments from Plasmodium falciparum (genome GC content P. falciparum: 20%).

Excluding regions from the read distribution calculation

In some cases, it will make sense to exclude certain regions from the calculation of the read distributions to increase the accuracy of the computation. There are several kinds of regions that are either not expected to show a background read distribution or where the uncertainty of the reference genome might be too big. Please consider the following points:

- **repetitive regions**: if multi-reads (reads that map to more than one genomic position) were excluded from the BAM file, it will help to exclude known repetitive regions. You can get BED files of known repetitive regions from UCSC Table Browser (see the screenshot below for an example of human repetitive elements).



- **regions of low mappability**: these are regions where the mapping of the reads notoriously fails and we recommend to exclude known regions with mappability issues from the GC computation. You can download the mappability tracks for different read lengths from UCSC, e.g. for mouse and human. In the github deepTools folder "scripts", you can find a shell script called *mappabilityBigWig_to_unmappableBed.sh* which will turn the bigWig mappability file from UCSC into a BED file.

- **ChIP-seq peaks**: in ChIP-seq samples it is *expected* that certain regions *should* show more reads than expected based on the background distribution, therefore it makes absolute sense to exclude those regions from the GC bias calculation. We recommend to run a simple, non-conservative peak calling on the uncorrected BAM file first to obtain a BED file of peak regions that should then subsequently be supplied to computeGCbias.

Output files

- **Diagnostic plot**
  - box plot of *absolute* read numbers per genomic GC fraction
  - x-y plot of *observed/expected* read ratios per genomic GC fraction (ideally, ratio should always be 1 (log2(1) = 0))
- **Data matrix**
  - tabular matrix file
  - to be used for GC correction with *correctGCbias*

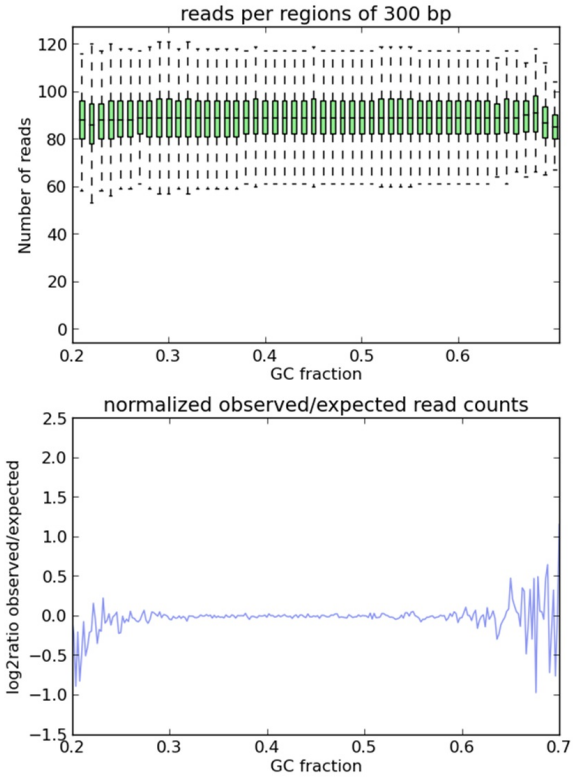
What the plots tell you

In an ideal sample without GC bias, the ratio of observed/expected values should be close to 1 for all GC content bins.

However, due to PCR (over)amplifications, the majority of ChIP samples usually shows a significant bias towards reads with high GC content (>50%) and a depletion of reads from GC-poor regions.

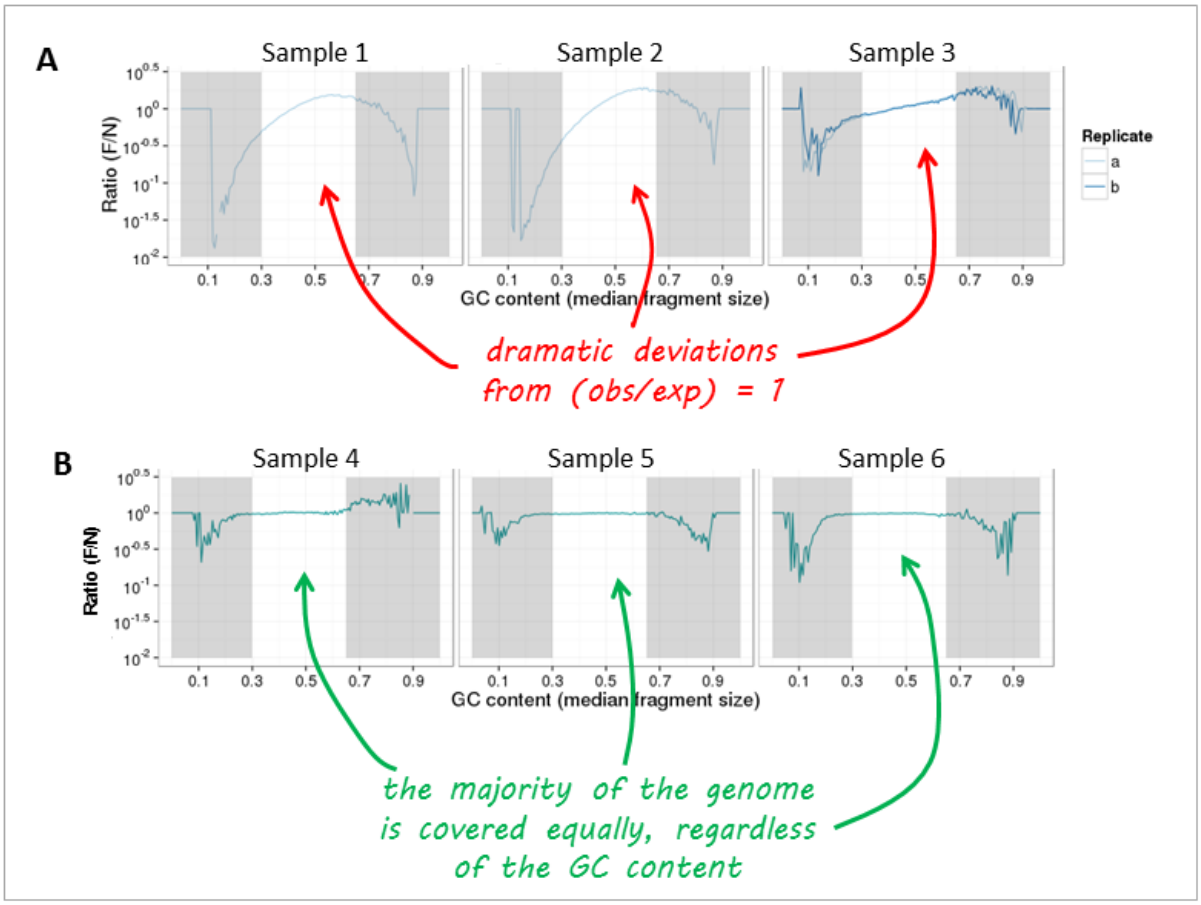
Example figures

Let's start with an ideal case. The following plots were generated with computeGCbias using simulated reads from the *Drosophila* genome.



As you can see, both plots based on **simulated reads** do not show enrichments or depletions for specific GC content bins, there is an almost flat line at log2ratio of 0 (= ratio of 1). The fluctuations on the ends of the x axis are due to the fact that only very, very few regions in the genome have such extreme GC fractions so that the number of fragments that are picked up in the random sampling can vary.

Now, let's have a look at **real-life data** from genomic DNA sequencing. Panels A and B can be clearly distinguished and the major change that took place between the experiments underlying the plots was that the samples in panel A were prepared with too many PCR cycles and a standard polymerase whereas the samples of panel B were subjected to very few rounds of amplification using a high fidelity DNA polymerase.



bamFingerprint

This quality control will most likely be of interest for you if you are dealing with ChIP-seq samples as a pressing question in ChIP-seq experiments is "Did my ChIP work?", i.e. did the antibody-treatment enrich sufficiently so that the ChIP signal can be separated from the background signal? (After all, around 90 % of all DNA fragments in a ChIP experiment will represent the genomic background). We use bamFingerprint routinely to monitor the outcome of ChIP-seq experiments.

What it does

This tool is based on a method developed by Diaz et al. and it determines how well the signal in the ChIP-seq sample can be differentiated from the background distribution of reads in the control sample. For factors that will enrich well-defined, rather narrow regions (e.g. transcription factors such as p300), the resulting plot can be used to assess the strength of a ChIP, but the broader the enrichments are to be expected, the less clear the plot will be. Vice versa, if you do not know what kind of signal to expect, the bamFingerprint plot will give you a straight-forward indication of how careful you will have to be during your downstream analyses to separate biological noise from meaningful signal.

The tool first samples indexed BAM files and counts all reads overlapping a window (bin) of specified length. These counts are then sorted according to their rank and the cumulative sum of read counts is plotted.

Output files:

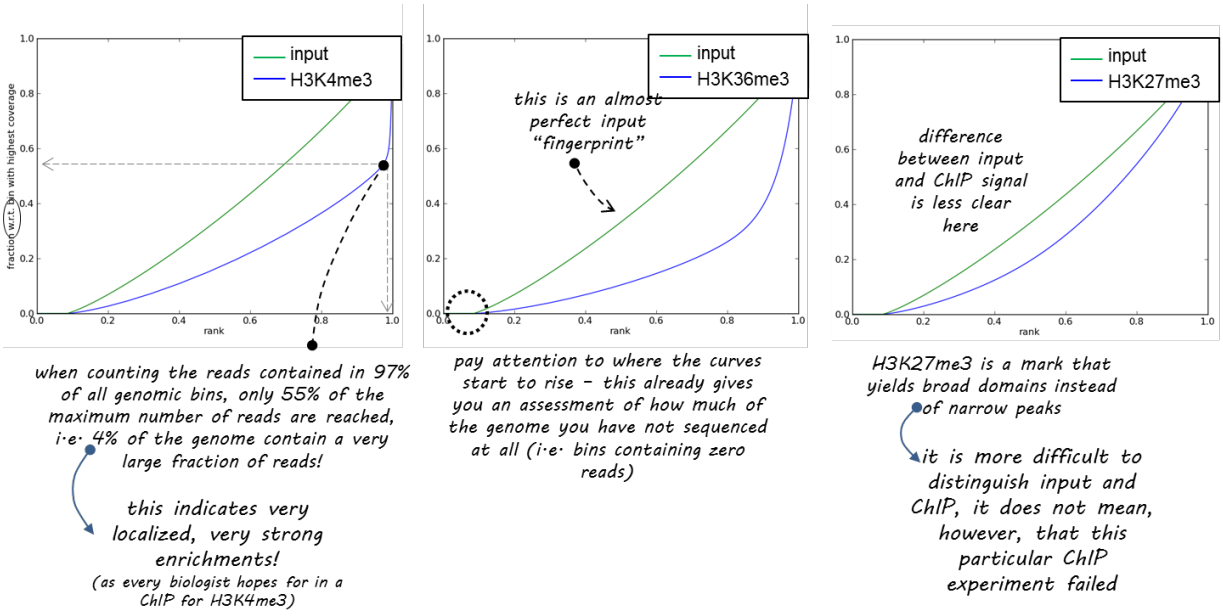
- **Diagnostic plot**
- **Data matrix** of raw counts (optional)

What the plots tell you

An ideal input with perfect uniform distribution of reads along the genome (i.e. without enrichments in open chromatin etc.) should generate a straight diagonal line. A very specific and strong ChIP enrichment will be indicated by a prominent and steep rise of the cumulative sum towards the highest rank. This means that a big chunk
of reads from the ChIP sample is located in few bins which corresponds to high, narrow enrichments seen for transcription factors.

Example figures

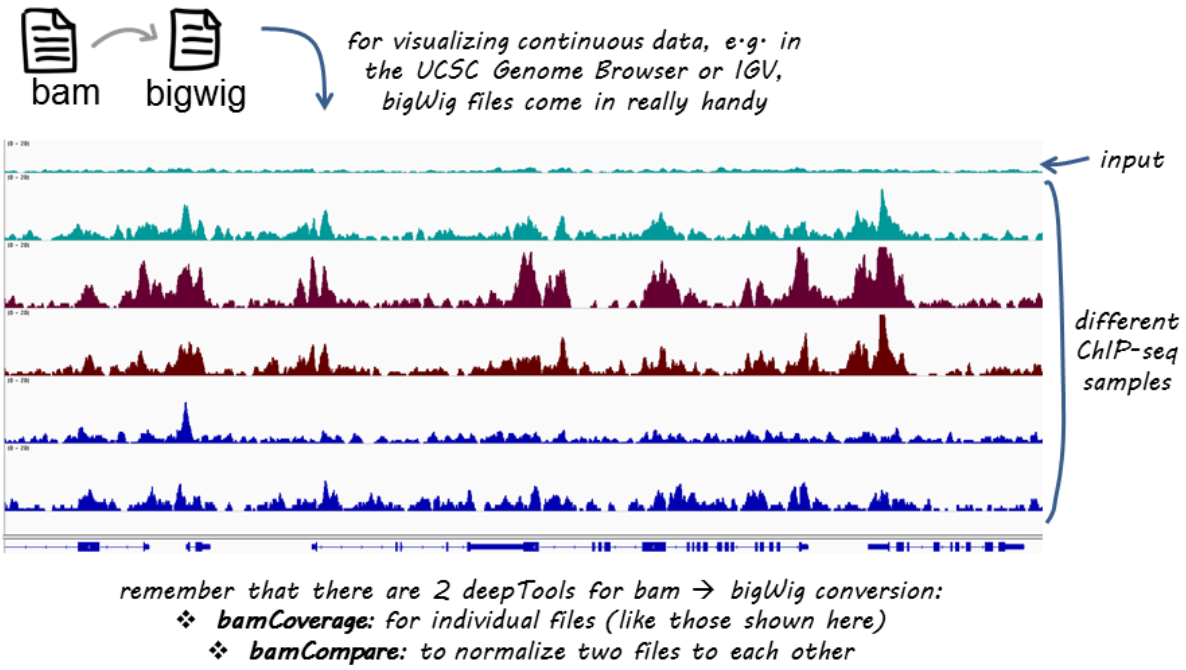
Here you see 3 different fingerprint plots.
We chose these examples to show you how the nature of the ChIP signal (narrow and high vs. wide and not extremely high) is reflected in the "fingerprint" plots. Please note that these plots go by the name of "fingerprints" in our facility because we feel that they help us tremendously in judging individual files, but the idea underlying these plots came from Diaz et al.



Normalization of BAM files

deepTools contains 3 tools for the normalization of BAM files:

1. **correctGCbias**: if 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.



correctGCbias

What it does

This tool requires the **output from computeGCBias** to correct a given BAM file according to the method proposed by Benjamini and Speed.

correctGCbias will remove reads from regions with too high coverage compared to the expected values (typically GC-rich regions) and will add reads to regions where too few reads are seen (typically AT-rich regions).

The resulting BAM file can be used in any downstream analyses, but **be aware that you should not filter out duplicates from here on** (duplicate removal would eliminate those reads that were added to reach the expected number of reads for GC-depleted regions).

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).

- RPKM:
  - reads per kilobase per million reads
  - The formula is: RPKM (per bin) = number of reads per bin / (number of mapped reads (in millions) * bin length (kp))
- RPGC:
  - reads per genomic content
  - used to normalize reads to 1x depth of coverage
  - sequencing depth is defined as: (total number of mapped reads * fragment length) / effective genome size

output

- **coverage file** either in bigWig or bedGraph format

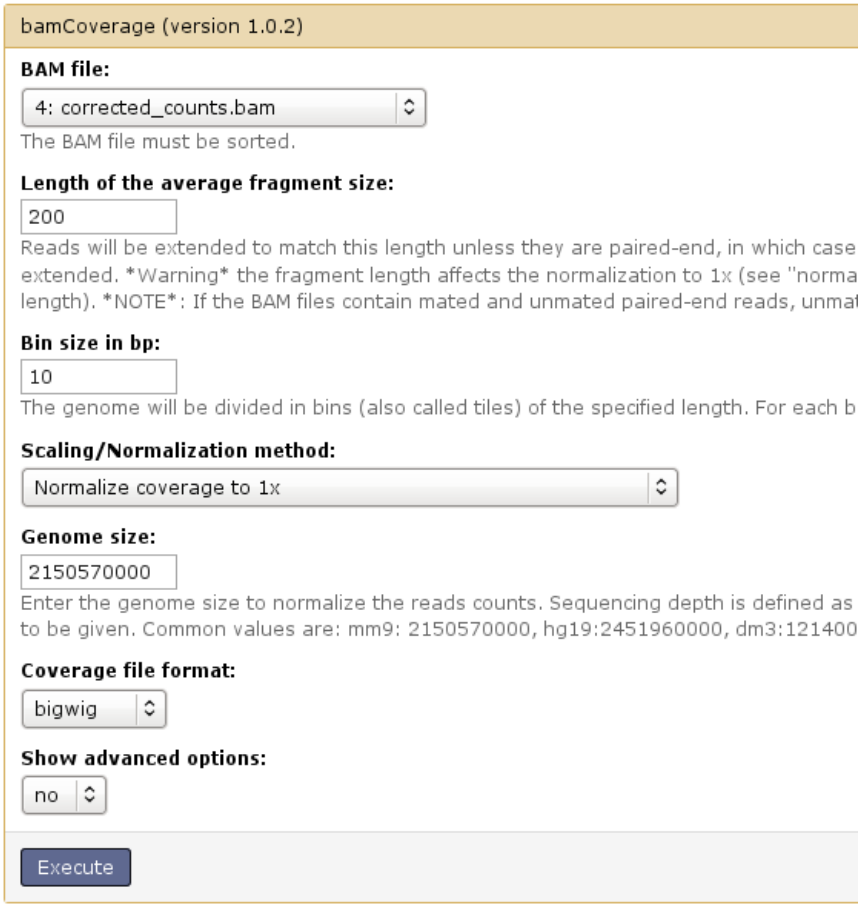
Usage

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 normalize 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 autosome, 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 (pay attention to the hints on the command line as well!):



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

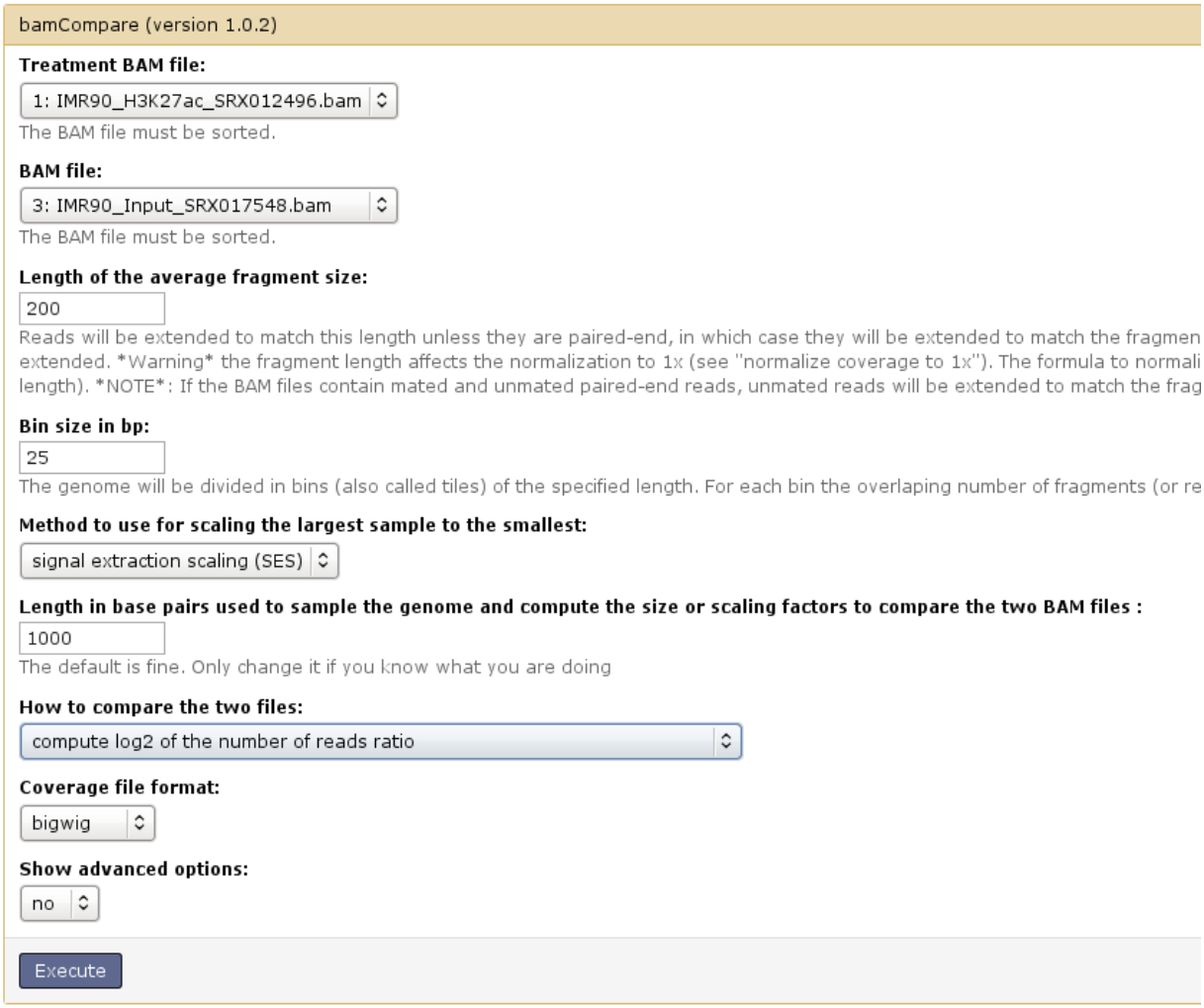
- same as for bamCoverage, except that you now obtain **1** coverage file that is based on **2** BAM files.

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:



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 (see below) 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 bamFingerprint-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:

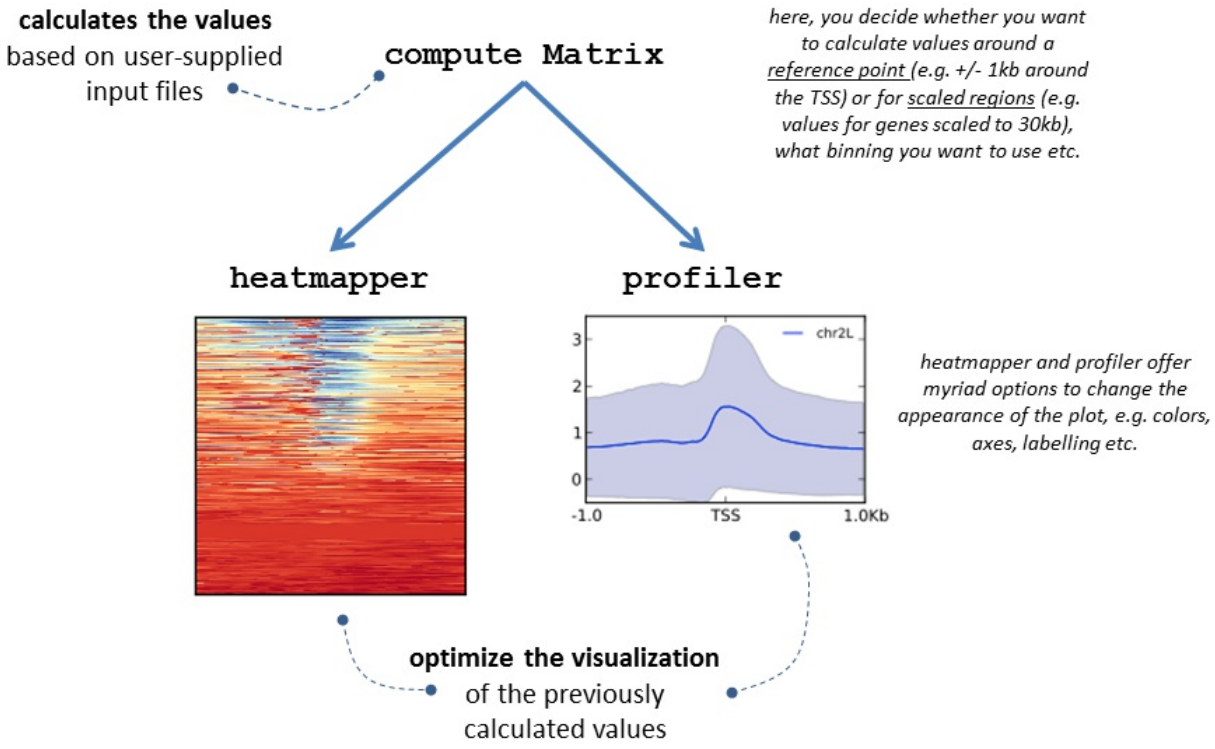
- the simple normalization **total read count**
- the more sophisticated signal extraction (SES) method proposed by Diaz et al. for the normalization of ChIP-seq samples. **We recommend to use SES only for those cases where the distinction between input and ChIP is very clear in the bamFingerprint plots**. 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 **r**eads **p**er **k**ilobase (--normalizeUsingRPKM; similar to RNA-seq normalization schemes).

Visualization

The modules for visualizing scores contained in bigWig files are separated into 1 tool that calculates the values (*computeMatrix*) and 2 tools that contain many, many options to fine-tune the plots (*heatmapper* and *profiler*). In other words: computeMatrix generates the values that are the basis for heatmapper and profiler.



computeMatrix

This tool summarizes and prepares an intermediary file containing scores associated with genomic regions that can be used afterwards to plot a heatmap or a profile.

Genomic regions can really be anything - genes, parts of genes, ChIP-seq peaks, favorite genome regions... as long as you provide a proper file in BED or INTERVAL format. This tool can also be used to filter and sort regions according to their score.

As indicated in the plot above, computeMatrix can be run with either one of the two modes: **scaled regions** or **reference point**.

Please see the example figures down below for explanations of parameters and options.

Output files

- **obligatory**: zipped matrix of values to be used with *heatmapper* and/or *profiler*
- **optional** (can also be generated with heatmapper or profiler in case you forgot to produce them in the beginning):
  - BED-file of the regions sorted according to the calculated values
  - list of average values per genomic bin
  - matrix of values per genomic bin per genomic interval

heatmapper

The heatmapper depicts values extracted from the bigWig file for each genomic region individually.
It requires the output from computeMatrix and most of its options are related to tweaking the visualization only. The values calculated by computeMatrix are not changed.

Definitely check the example at the bottom of the page to get a feeling for how many things you can tune.

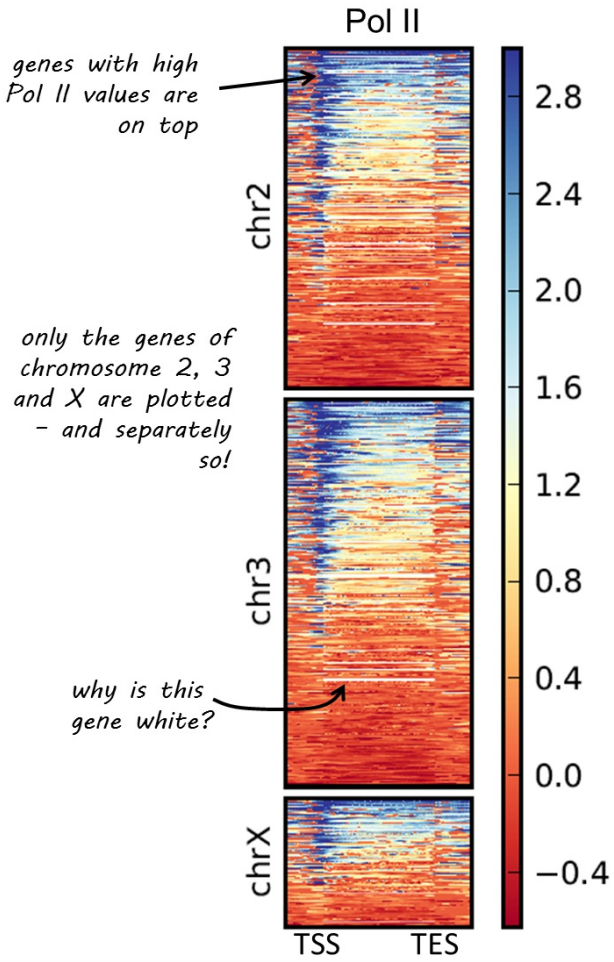
profiler

This tool plots the average enrichments over all genomic regions supplied to computeMarix. It is a very useful complement to the heatmapper, especially in cases when you want to compare the scores for many different groups. Like heatmapper, profiler does not change the values that were compute by computeMatrix, but you can choose between many different ways to color and display the plots.

Example figures

Here you see a typical, not too pretty example of a heatmap. We will use this example to explain several features of computeMatrix and heatmapper, so do take a closer look.

1st example: Heatmap with all genes scaled to the one size and user-specified groups of genes



As you can see, all genes have been scaled to the same size and the (mean) values per bin size (10 bp) are colored accordingly. In addition to the gene bodies, we added 500 bp up- and down-stream of the genes.

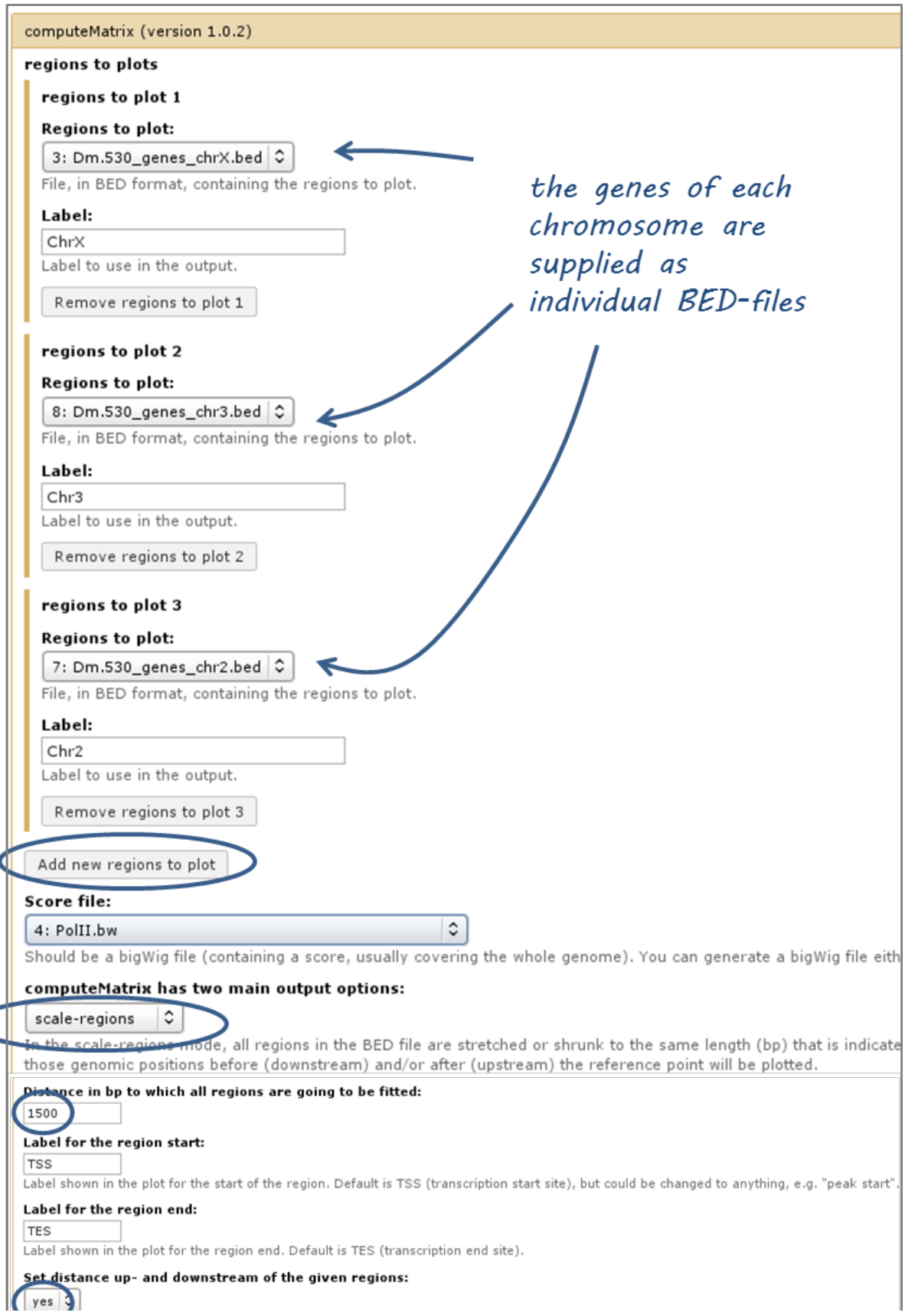
The plot was produced with the following commands:

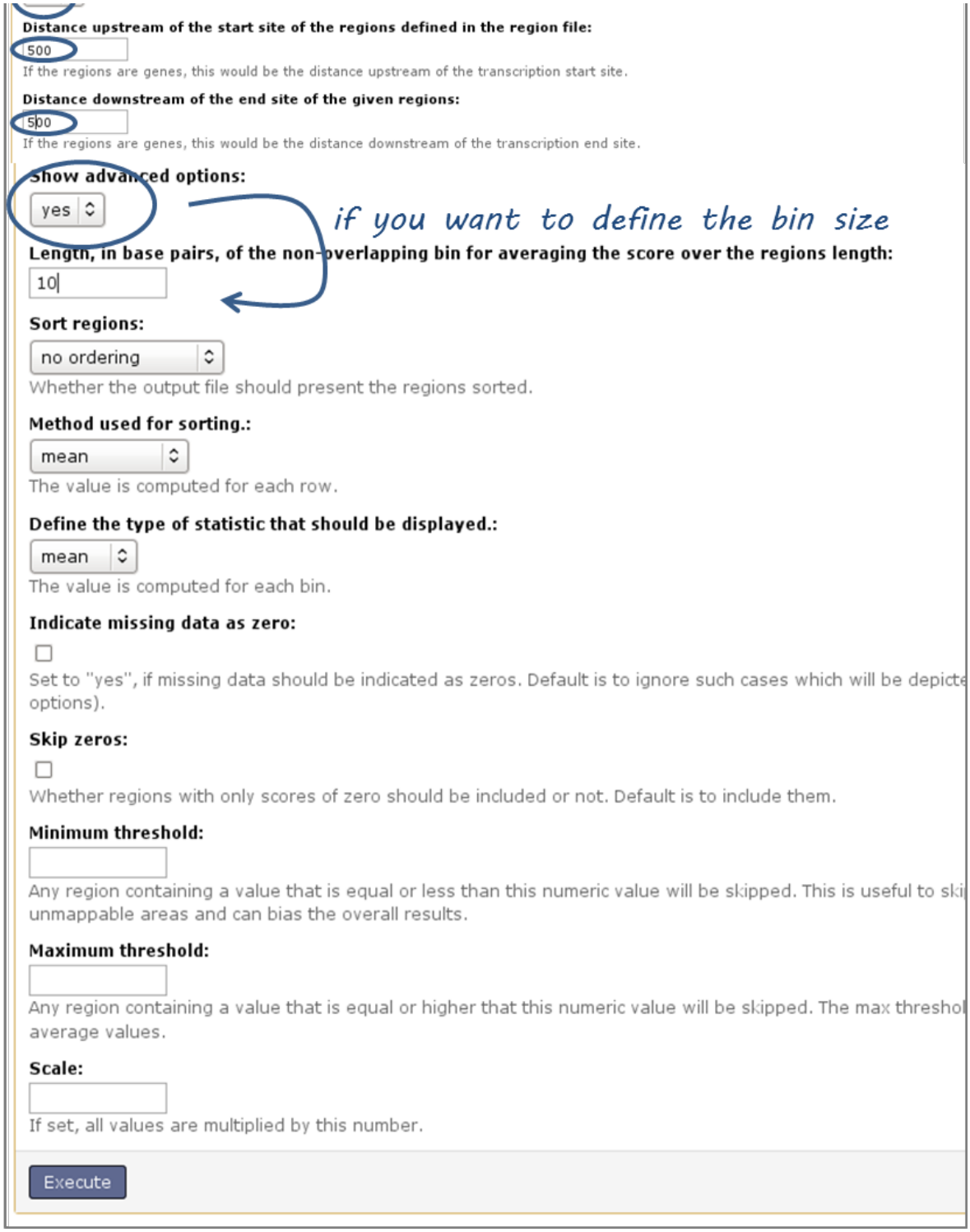
```
$ /deepTools-1.5.2/bin/computeMatrix scale-regions --regionsFileName Dm.genes.indChromLabeled.bed \
--scoreFileName PolII.bw --beforeRegionStartLength 500 --afterRegionStartLength 500 \
--regionBodyLength 1500 --binSize 10 \
--outFileName PolII_matrix_scaledGenes --sortRegions no

$ /deepTools-1.5.2/bin/heatmapper --matrixFile PolII_matrix_scaledGenes \
```

```
--outFileName PolII_indChr_scaledGenes.pdf \
--plotTitle "Pol II" --whatToShow "heatmap and colorbar"
```

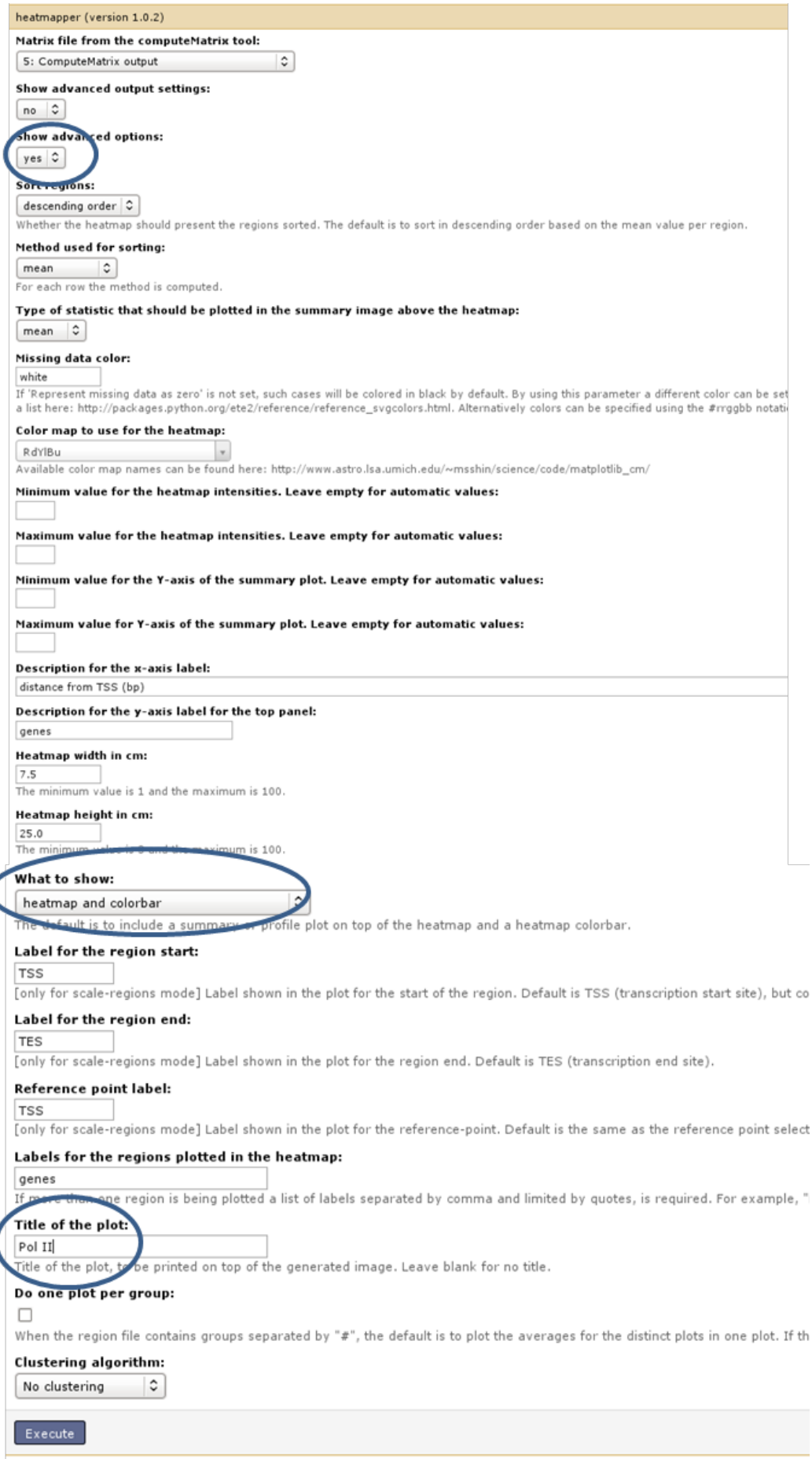
This is what you would have to select to achieve the same result within Galaxy (pay attention to the fact that you will have to use two tools, computeMatrix and heatmapper):

computeMatrix




Distance upstream of the start site of the regions defined in the region file:
500
If the regions are genes, this would be the distance upstream of the transcription start site.
Distance downstream of the end site of the given regions:
500
If the regions are genes, this would be the distance downstream of the transcription end site.
Show advanced options:
yes
if you want to define the bin size
Length, in base pairs, of the non-overlapping bin for averaging the score over the regions length:
10
Sort regions:
no ordering
Whether the output file should present the regions sorted.
Method used for sorting.:
mean
The value is computed for each row.
Define the type of statistic that should be displayed.:
mean
The value is computed for each bin.
Indicate missing data as zero:
Set to "yes", if missing data should be indicated as zeros. Default is to ignore such cases which will be depicte
options).
Skip zeros:
Whether regions with only scores of zero should be included or not. Default is to include them.
Minimum threshold:
Any region containing a value that is equal or less than this numeric value will be skipped. This is useful to ski
unmappable areas and can bias the overall results.
Maximum threshold:
Any region containing a value that is equal or higher that this numeric value will be skipped. The max thresho
average values.
Scale:
If set, all values are multiplied by this number.
Execute

heatmapper



The main difference between computeMatrix usage on the command line and Galaxy: the input of the regions file (BED)

Note that we supplied just *one* BED-file via the command line whereas in Galaxy we indicated three different files (one per chromosome).

On the command line, the program expects a BED file where different groups of genomic regions are concatenated into one file, where the beginning of each group should be indicated by "#group name".
The BED-file that was used here, contained 3 such lines and could be prepared as follows:

```
$ grep ^chr2 AllGenes.bed > Dm.genes.indChromLabeled.bed
$ echo "#chr2" >> Dm.genes.indChromLabeled.bed
$ grep ^chr3 AllGenes.bed >> Dm.genes.indChromLabeled.bed
$ echo "#chr3" >> Dm.genes.indChromLabeled.bed
$ grep ^chrX AllGenes.bed >> Dm.genes.indChromLabeled.bed
$ echo "#chrX" >> Dm.genes.indChromLabeled.bed
```

In Galaxy, you can simply generate three different data sets starting from a whole genome list of *Drosophila melanogaster* genes by using the "Filter" tool ("Filter and Sort" --> "Filter") on the entries in the first column three times:

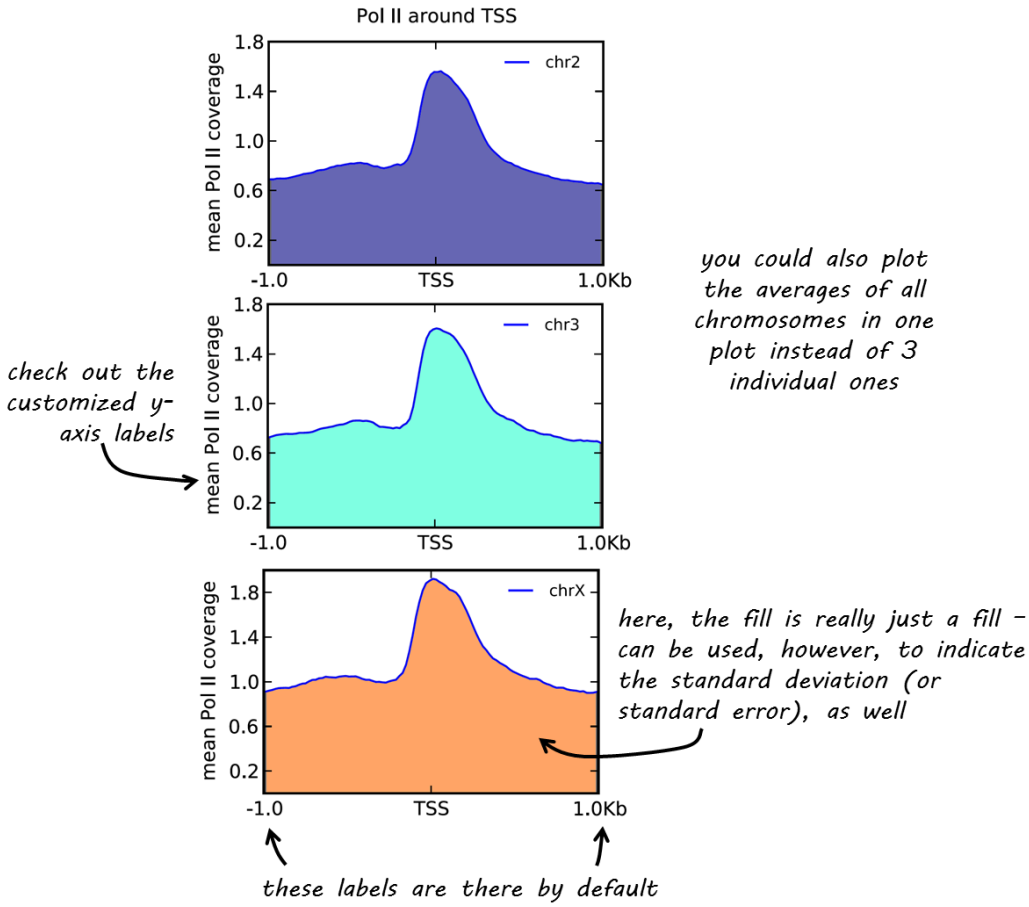
1. c1=="chr2" --> Dm.genes.chr2.bed
2. c1=="chr3" --> Dm.genes.chr3.bed
3. c1=="chrX" --> Dm.genes.chrX.bed

Important parameters for optimizing the visualization

1. **sorting of the regions**: The default of heatmapper is to sort the values in descending order. You can change that to ascending, no sorting at all or according to the size of the region (Using the `--sort` option on the command line or advanced options in Galaxy). We strongly recommend to leave the sorting option at "no sorting" for the initial computeMatrix step.
2. **coloring**: The default coloring by heatmapper is done using the python color map "RdYlBu", but this can be changed (--colorMap on the command line, advanced options within Galaxy).
3. **dealing with missing data**: You have certainly noticed that some gene bodies are depicted as white lines within the otherwise colorful mass of genes. Those regions are due to genes that, for whatever reason, did not have any read coverage in the bigWig file. There are several ways to handle these cases:
    - **--skipZeros** this is useful when your data actually has a quite nice coverage, but there are 2 or 3 regions where you deliberately filtered out reads or you don't expect any coverage (e.g. hardly mapable regions). This will only work if the entire region does not contain a single value.
    - **--missingDataAsZero** this option allows computeMatrix do interpret missing data points as zeroes. Be aware of the changes to the average values that this might cause.
    - **--missingDataColor** this is in case you have very sparse data or were missing values make sense (e.g. when plotting methylated CpGs - half the genome should have no value). This option then allows you to pick out your favorite color for those regions. The default is black (was white when the above shown image was produced).

2nd example: Summary plots with all genes scaled to the one size and user-specified groups of genes

Here's the **profiler** plot corresponding to the heatmap above. There's one major difference though - do you spot it?



We used the same BED file(s) as for the heatmap, hence the 3 different groups (1 per chromosome). However, this time we used computeMatrix not with *scale-regions* but with *reference-point* mode.

```
$ /deepTools-1.5.2/bin/computeMatrix reference-point --referencePoint TSS \
--regionsFileName Dm.genes.indChromLabeled.bed --scoreFileName PolII.bw \
--beforeRegionStartLength 1000 --afterRegionStartLength 1000 \
--binSize 10 --outFileName PolII_matrix_indChr_refPoint \
--missingDataAsZero --sortRegions no

$ /deepTools-1.5.2/bin/profiler --matrixFile PolII_matrix_indChr_refPoint \
--outFileName profile_PolII_indChr_refPoint.pdf
--plotType fill --startLabel "TSS" \
--plotTitle "Pol II around TSS" --yAxisLabel "mean Pol II coverage" \
--onePlotPerGroup
```

When you compare the profiler commands with the heatmapper commands, you also notice that we made use of many more labeling options here, e.g. `--yAxisLabel` and a more specific title via `-T`

This is how you would have obtained this plot in Galaxy (only the part that's *different* from the above shown command for the scale-regions version is shown):

computeMatrix

The reference point for the plotting:

beginning of region (e.g. TSS)

Discard any values after the region end:

☐

This is useful to visualize the region end when not using the scale-regions mode and when the reference-point is set to the TSS.

Distance upstream of the start site of the regions defined in the region file:

1000

If the regions are genes, this would be the distance upstream of the transcription start site.

Distance downstream of the end site of the given regions:

1000

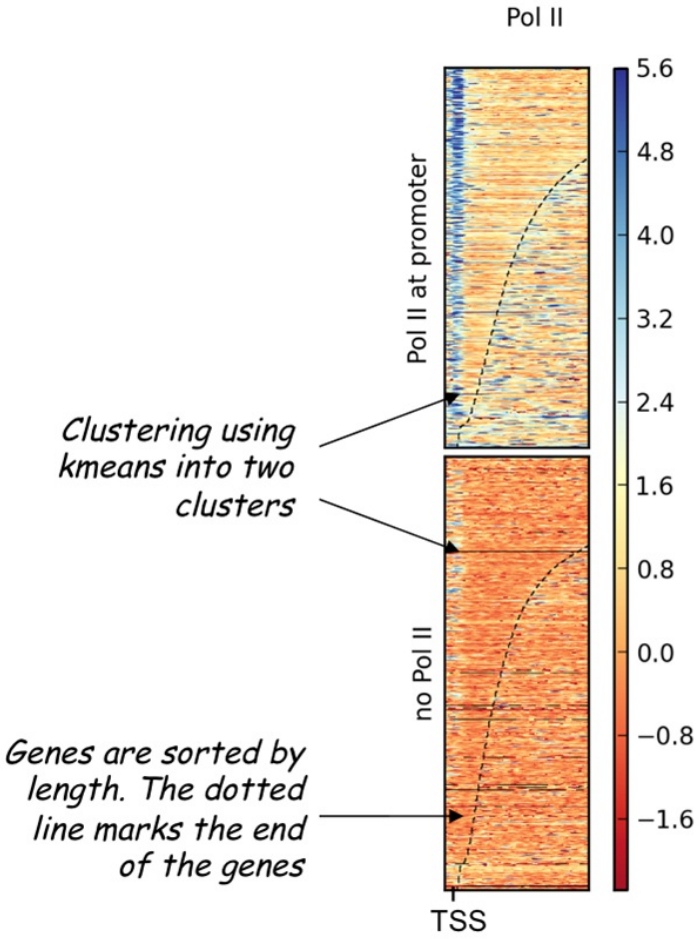
If the regions are genes, this would be the distance downstream of the transcription end site.

profiler

3rd example: Heatmap with all genes scaled to the one size and kmeans clustering

Instead of supplying groups of regions on your own, you can use the clustering function of heatmapper to get a first impression whether the signal of your experiment can be easily clustered into two or more groups of similar signal distribution.

Have a look at this example with two clusters. The values correspond to log2ratios(ChIP/input) from a ChIP-seq experiment for RNA Polymerase II in *Drosophila melanogaster*:



The plot was produced with the following commands:

```
$ /deepTools-1.5.2/bin/computeMatrix reference-point \
--regionsFilenName Dm.genes.indChromLabeled.bed \
--scoreFileName PolII.bw \
--beforeRegionStartLength 500 --afterRegionStartLength 5000 \
--binSize 50 \
--outFileName PolII_matrix_TSS

$ /deepTools-1.5.2/bin/heatmapper --matrixFile PolII_matrix_TSS \
--kmeans 2 \
--sortUsing region_length \
--outFileName PolII_two_clusters.pdf \
--plotTitle "Pol II" --whatToShow "heatmap and colorbar"
```

In Galaxy, these are the screenshots from the commands for computeMatrix and heatmapper:

computeMatrix

computeMatrix (version 1.0.3)

regions to plots

regions to plot 1

Regions to plot:

1: UCSC Main on D. melanogaster: flyBaseGene (genome)

File, in BED format, containing the regions to plot.

Label:

genes

Label to use in the output.

Remove regions to plot 1

regions to plot 2

Regions to plot:

82: (as bed) bamCompare on data 62 and data 63: ratio

File, in BED format, containing the regions to plot.

Label:

Label to use in the output.

Remove regions to plot 2

Add new regions to plot

Score file:

53: log2ratio_PolII_Input.bigwig (SES)

Should be a bigWig file (containing a score, usually covering the whole genome). You can generate a bigWig file either from a bedGraph or WIG file using UCSC tools or from a BAM file using the deepTool bamCoverage.

computeMatrix has two main output options:

reference-point

In the scale-regions mode, all regions in the BED file are stretched or shrunk to the same length (bp) that is indicated by the user. Reference-point refers to a position within the BED regions (e.g start of region). In the reference-point mode only those genomic positions before (downstream) and/or after (upstream) the reference point will be plotted.

The reference point for the plotting:

beginning of region (e.g. TSS)

Discard any values after the region end:

☐

This is useful to visualize the region end when not using the scale-regions mode and when the reference-point is set to the TSS.

Distance upstream of the start site of the regions defined in the region file:

500

If the regions are genes, this would be the distance upstream of the transcription start site.

Distance downstream of the end site of the given regions:

5000

If the regions are genes, this would be the distance downstream of the transcription end site.

Show advanced output settings:

yes

Save the matrix of values underlying the heatmap:

☐

Save the data underlying the average profile:

☐

Save the regions after skipping zeros or min/max threshold values:

☐

The order of the regions in the file follows the sorting order selected. This is useful, for example, to generate other heatmaps keeping the sorting of the first heatmap.

Show advanced options:

yes

Length, in base pairs, of the non-overlapping bin for averaging the score over the regions length:

50

Sort regions:

no ordering

Whether the output file should present the regions sorted.

Method used for sorting.:

mean

The value is computed for each row.

Define the type of statistic that should be displayed.:

mean

The value is computed for each bin.

Indicate missing data as zero:

☑

Set to "yes", if missing data should be indicated as zeros. Default is to ignore such cases which will be depicted as black areas in the heatmap. (see "Missing data color" options of the heatmapper for additional options).

Skip zeros:

☑

Whether regions with only scores of zero should be included or not. Default is to include them.

Minimum threshold:

Skip zeros:

☑

Whether regions with only scores of zero should be included or not. Default is to include them.

Minimum threshold:

Any region containing a value that is equal or less than this numeric value will be skipped. This is useful to skip, for example, genes where the read count is zero for any of the bins. This could be the result of unmappable areas and can bias the overall results.

Maximum threshold:

Any region containing a value that is equal or higher that this numeric value will be skipped. The max threshold is useful to skip those few regions with very high read counts (e.g. major satellites) that may bias the average values.

Scale:

If set, all values are multiplied by this number.

Execute

heatmapper


heatmapper (version 1.0.3)
Matrix file from the computeMatrix tool:
85: computeMatrix on data 5, data 82, and data 53: Matrix
Show advanced output settings:
no
Show advanced options:
yes
Sort regions:
descending order
Whether the heatmap should present the regions sorted. The default is to sort in descending order based on the mean value per region.
Method used for sorting:
region length
instead of mean, we choose to sort according to the regions' length
(just to show you an alternative sorting)
For each row the method is computed.
Type of statistic that should be plotted in the summary image above the heatmap:
mean
What to show:
heatmap and colorbar
The default is to include a summary or profile plot on top of the heatmap and a heatmap colorbar.
Label for the region start:
TSS
[only for scale-regions mode] Label shown in the plot for the start of the region. Default is TSS (transcription start site), but could be changed to anything, e.g. "peak start".
Label for the region end:
TES
[only for scale-regions mode] Label shown in the plot for the region end. Default is TES (transcription end site).
Reference point label:
TSS
[only for scale-regions mode] Label shown in the plot for the reference-point. Default is the same as the reference point selected (e.g. TSS), but could be anything, e.g. "peak start" etc.
Labels for the regions plotted in the heatmap:
no Pol II, Pol II at promoter
2 clusters <-> 2 names
If more than one region is being plotted a list of labels separated by comma and limited by quotes, is required. For example, label1, label2.
Title of the plot:
Pol II
Title of the plot, to be printed on top of the generated image. Leave blank for no title.
Do one plot per group:
When computeMatrix was used on more than one group of genes, the average plots for all the groups will be drawn in one panel by default. If this option is set, each group will get its own plot, stacked on top of each other.
Did you used multiple regions in ComputeMatrix?:
No, I used only one region.
clustering will overwrite any user-specified groups of regions which
is why we recommend to use it only for cases where you supplied
just one BED file to computeMatrix
Would you like to cluster the regions according to the similarity of the signal distribution? This is only possible if you used computeMatrix on only one group of regions.
Clustering algorithm:
Kmeans clustering
Number of clusters to compute:
2
When this option is set, then the matrix is split into clusters using the kmeans algorithm. Only works for data that is not grouped, otherwise only the first group will be clustered. If more specific clustering methods are required it is advisable to save the underlying matrix and run the clustering using other software. The plotting of the clustering may fail (Error: Segmentation fault) if a cluster has very few members compared to the total number or regions. (default: None).
Execute

When the `--kmeans` option is chosen and more than 0 clusters are specified, heatmapper will run the k-means clustering algorithm. In this example, we wanted to divide *Drosophila melanogaster* genes into two clusters. As you can see above, the algorithm nicely identified two groups - one with mostly those genes with lots of Pol II at the promoter region (top) from those genes without Poll II at the promoter (bottom).
Please note that the clustering will only work if the initial BED-file used with computeMatrix contained only *one* group of genes.

The genes belonging to each cluster can be obtained by via `--outFileSortedRegions` on the command line and "advanced output options in Galaxy". On the command line, this will result in a BED file where the groups are separated by a hash tag. In Galaxy, you will obtain individual data sets per cluster.

To have a better control on the clustering it is recommended to load the matrix raw data into **specialized software like cluster3 or R**. You can obtain the matrix via the option `--outFileNameMatrix` on the command line and by the "advanced output options" in Galaxy. The order of the rows is the same as in the output of the `--outFileSortedRegions` BED file.

Glossary

Like most specialized fields, next-generation sequencing has inspired many an acronym. We are trying to keep track of those abbreviations that we heavily use. Do make us aware if something is unclear: deeptools@googlegroups.com

If you are unfamiliar with the file formats of next-generation sequencing data, do have a look on the next page.

Abbreviations

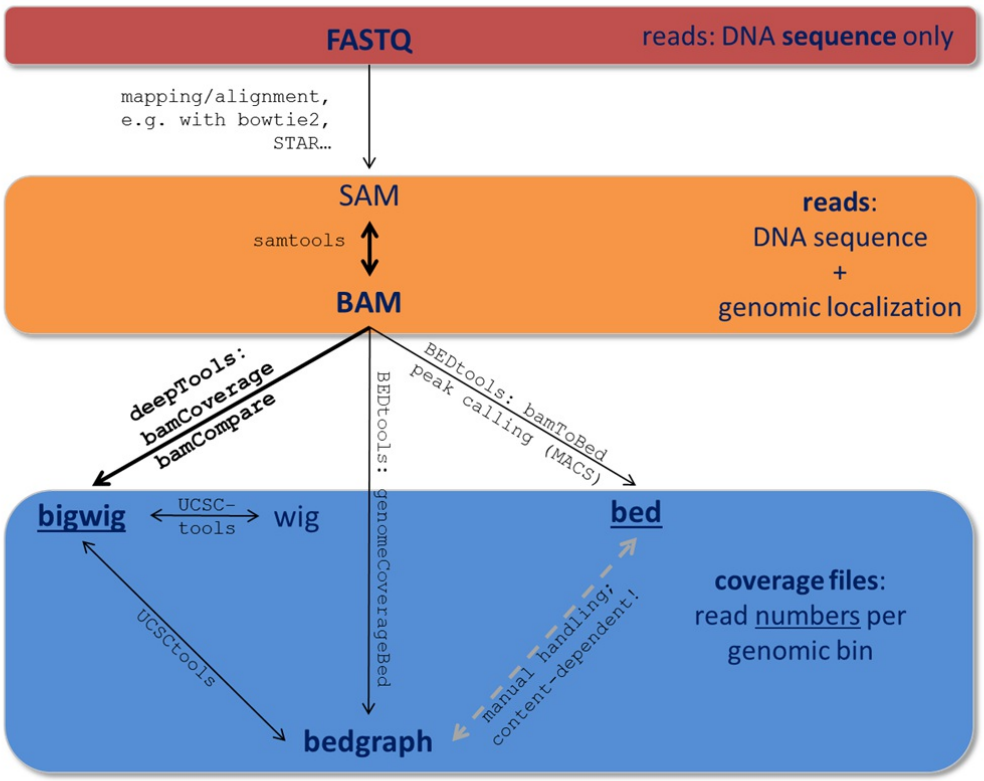
Acronym	full phrase	Synonyms/Explanation
-seq	-sequencing	indicates that an experiment was completed by DNA sequencing using NGS
ChIP-seq	chromatin immunoprecipitation sequencing	NGS technique for detecting transcription factor binding sites and histone modifications (see entry "Input" for more information)
DNase	deoxyribonuclease	micrococcal nuclease
HTS	high-throughput sequencing	next-generation sequencing, massive parallel short read sequencing, deep sequencing
Input	--	control experiment typically done for ChIP-seq experiments (see above) - while ChIP-seq relies on antibodies to enrich for DNA fragments bound to a certain protein, the input sample should be processed exactly the same way, excluding the antibody. This way, one hopes to account for biases introduced by the sample handling and the general chromatin structure of the cells
MNase	micrococcal nuclease	DNase
NGS	next-generation sequencing	high-throughput (DNA) sequencing, massive parallel short read sequencing, deep sequencing
RPGC	reads per genomic content	used to normalize read numbers (also: normalize to 1x sequencing depth), sequencing depth is defined as: (total number of mapped reads * fragment length) / effective genome size.
RPKM	reads per kilobase per million reads	used to normalize read numbers, the following formula is used by bamCoverage: RPKM (per bin) = number of reads per bin / (number of mapped reads (in millions) * bin length (kb))

File Formats

Data obtained from next-generation sequencing data must be processed several times. Most of the processing steps are aimed at extracting only those information that are truly needed for a specific down-stream analysis and to discard all the redundant entries. Therefore, **specific data formats are often associated with different steps of a data processing pipeline**. These associations, however, are by no means binding, but you should understand what kind of data is represented in which data format - this will help you to select the correct tools further down the road.

Here, we just want to give very brief key descriptions of the file, for elaborate information we will link to external websites (links to be found in our online wiki). Be aware, that the file name sorting here is purely alphabetically, not according to their usage within an analysis pipeline that is depicted here.
For more information on the different tool collections mentioned in the figure, please check the following links:

- deepTools wiki: http://github.com/fidelram/deepTools/wiki
- samtools: http://samtools.sourceforge.net/http://samtools.sourceforge.net/
- UCSCtools download: http://hgdownload.cse.ucsc.edu/admin/exe/
- BEDtools: http://bedtools.readthedocs.org/en/latest/



2bit

- compressed, binary version of genome sequences that are often stored in [FASTA][]
- most genomes in 2bit format can be found at UCSC]
- [FASTA][] files can be converted to 2bit using the UCSC programm **faToTwoBit** available for different platforms at UCSC
- more information can be found here or from UCSC

BAM

- typical file extension: .bam

- *binary* file format (complement to SAM)
- contains information about sequenced reads *after alignment* to a reference genome
- each line = 1 mapped read, with information about:
  - its mapping quality (how certain is the read alignment to this particular genome locus?)
  - its sequencing quality (how well was each base pair detected during sequencing?)
  - its DNA sequence
  - its location in the genome
  - etc.
- highly recommended format for storing data
- to make a BAM file human-readable, one can, for example, use the program samtools view (from UCSC tools)
- for more information, see below for the definition of SAM files

bed

- typical file extension: .bed
- text file
- used for genomic intervals, e.g. genes, peak regions etc.
- actually, there is a rather strict definition of the format that can be found at UCSC
- for deepTools, the first 3 columns are important: chromosome, start position of the region, end position of the genome
- do not confuse it with the bedGraph format (eventhough they are quite similar)
- example lines from a BED file of mouse genes (note that the start position is 0-based, the end-position 1-based, following UCSC conventions for BED files):

```
chr1    3204562 3661579 NM_001011874    Xkr4    -
chr1    4481008 4486494 NM_011441   Sox17   -
chr1    4763278 4775807 NM_001177658    Mrpl15  -
chr1    4797973 4836816 NM_008866   Lypla1  +
```

bedGraph

- typical file extension: .bg, .bedgraph
- text file
- similar to BED file (not the same!), it can ONLY contain 4 columns and 4th column MUST be a score
- again, read the UCSC description for more details
- 4 exemplary lines from a bedGraph file (like BED files following the UCSC convention, the start position is 0-based, the end-position 1-based in bedGraph files):

```
chr1 10 20 1.5
chr1 20 30 1.7
chr1 30 40 2.0
chr1 40 50 1.8
```

bigWig

- typical file extension: .bw, .bigwig
- *binary* version of a bedGraph file
- usually contains 4 columns: chromosome, start of genomic bin, end of genomic bin, score
- the score can be anything, e.g. an average read coverage
- UCSC description for more details

FASTA

- typical file extension: .fasta
- text file, often gzipped (--> .fasta.gz)
- very simple format for **DNA/RNA** or **protein** sequences, this can be anything from small pieces of DNA or proteins to entire genome information (most likely, you will get the genome sequence of your organism of interest in fasta format)
- see the 2bit file format entry for a compressed alternative of the fasta format
- example from wikipedia showing exactly one sequence:

```
>gi|5524211|gb|AAD44166.1| cytochrome b [Elephas maximus maximus]
LCLYTHIGRNIYYGSYLYSETWNTGIMLLLITMATAFMGYVLPWGQMSFWGATVITNLFSAIPYIGTNLV
```

```
EWIWGGFSVDKATLNRFFAFHFILPFTMVALAGVHLTFLHETGSNNPLGLTSDSDKIPFHPYYTIKDFLG
LLILILLLLLLALLSPDMLGDPDNHMPADPLNTPLHIKPEWYFLFAYAILRSVPNKLGGVLALFLSIVIL
GLMPFLHTSKHRSMMLRPLSQALFWTLTMDLLTLTWIGSQPVEYPYTIIGQMASILYFSIILAFLPIAGX
IENY
```

FASTQ

- typical file extension: .fastq, fq
- text file, often gzipped (--> .fastq.gz)
- contains raw read information (e.g. base calls, sequencing quality measures etc.), but not information about where in the genome the read originated from
- example from the wikipedia page

```
A FASTQ file containing a single sequence might look like this:
@SEQ_ID
GATTTGGGGTTCAAAGCAGTATCGATCAAATAGTAAATCCATTTGTTCAACTCACAGTTT
+
!''*((((***+))%%%++)(%%%%).1***-+*''))**55CCF>>>>>>CCCCCCC65

The character '!' represents the lowest quality while '~' is the highest.
```

SAM

- typical file extension: .sam
- should be the result of an alignment of sequenced reads to a reference genome
- each line = 1 mapped read, with information about its mapping quality, its sequence, its location in the genome etc.
- it is recommended to generate the binary (compressed) version of this file format: BAM
- for more information, see the SAM specification
- two exemplary lines
  - each one corresponds to one read (named r001 and r002 here)
  - the different columns contain various information about each read, e.g. which chromosome they were mapped to (here: chr1) and the left-most mapping position in the genome (here: 7 and 9 on chr1); the *flag* in the second column summarizes multiple information about each single read at once (in hexadecimal encoding) (see below for more information on the flag)

```
r001 163 chr1 7 30 8M2I4M1D3M = 37 39 TTAGATAAAGGATACTG *
r002 0 chr1 9 30 3S6M1P1I4M * 0 0 AAAAGATAAGGATA *
```

- the flag contains the answer to several yes/no assessments that are encoded in a single number. The questions are the following ones:
  - template having multiple segments in sequencing = Is the read part of a read pair?
  - each segment properly aligned according to the aligner = Was the read properly paired?
  - segment unmapped = Is the read unmapped?
  - next segment in the template unmapped = Is the mate unmapped?
  - reverse complemented = Did the read map to the reverse strand?
  - next segment in the template is reversed = Did the mate map to the reverse strand?
  - the first segment in the template = Is this read the first one in the pair?
  - the last segment in the template = Is this read the second one in the pair?
  - secondary alignment = Is this not the primary (i.e. unique optimal) alignment for the read?
  - not passing quality controls = Did the read not pass the quality control?
  - PCR or optical duplicate = Was this read a PCR or optical duplicate?
- for more details on the flag, see this thorough explanation or this more technical explanation

Links and references

Literature

Benjamini and Speed, Nucleic Acids Research (2012): http://nar.oxfordjournals.org/content/40/10/e72
Diaz et al., Stat. Appl. Gen. Mol. Biol. (2012): http://www.degruyter.com/view/j/sagmb.2012.11.issue-3/1544-6115.1750/1544-6115.1750.xml

For more NGS-related literature, see our collection at the deepTools web server: http://deeptools.ie-freiburg.mpg.de/u/fduendar/p/useful-bioinfo-literature

Additional bioinformatic tools

bowtie2: http://bowtie-bio.sourceforge.net/bowtie2/manual.shtml
cluster3: http://bonsai.hgc.jp/~mdehoon/software/cluster/
IGV (Integrative Genome Browser): http://www.broadinstitute.org/igv/
k-means clustering: http://en.wikipedia.org/wiki/K-means_clustering
R: http://www.r-project.org/

File format information

SAM file specification: http://samtools.sourceforge.net/SAMv1.pdf
File formats explained at UCSC: http://genome.ucsc.edu/FAQ/FAQformat.html#format1

Fidel Ramírez, Friederike Dündar, Sarah Diehl, Björn A. Grüning, Thomas Manke

Bioinformatics Group, Max-Planck-Institute of Immunobiology and Epigenetics, Freiburg & Department of Computer Science, University of Freiburg

Web server (incl. sample data): http://deepTools.ie-freiburg.mpg.de
Code: https://github.com/fidelram/deepTools
Wiki & Tutorials: https://github.com/fidelram/deepTools/wiki