deepTools

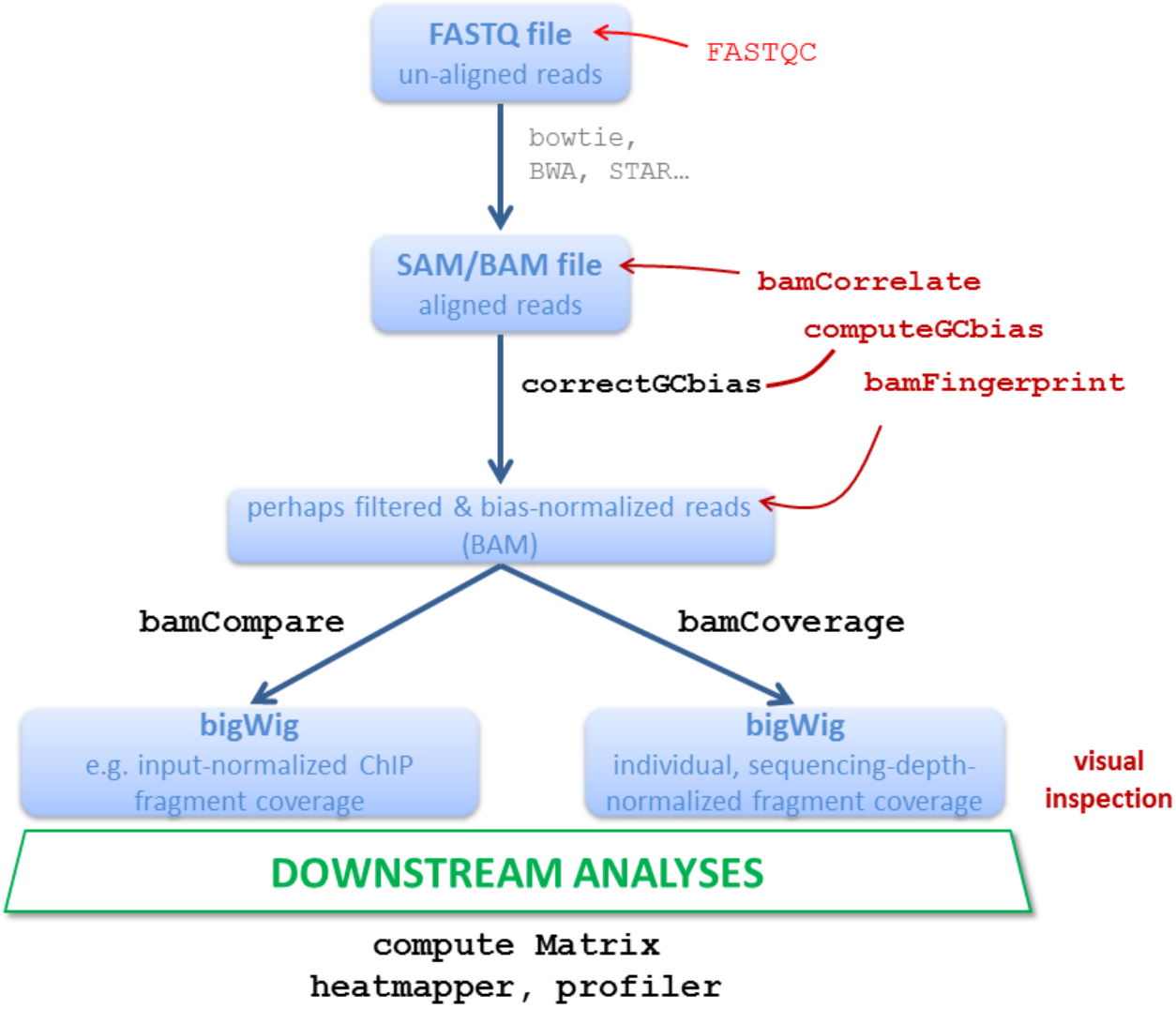
Table of content

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 of 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 colours, 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). For more information on a typical analysis pipeline, read the text below and What deepTools can do.



How we use deepTools

The majority of samples that we handle within our facility come from ChIP-seq experiments, therefore you will find many examples from ChIP-seq analyses. This does not mean that deepTools is restricted to ChIP-seq data analysis, but 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.)

Here are slides that we used for teaching at the University of Freiburg that contain more details on the deepTools usage and aims.

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**. This is a QC we 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're 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 details

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 - for more information, please read below.

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 exp. and obs. 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	zipped file, 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 ("meta-profile")	visualize the average read coverages over a group of genomic regions

QC of aligned reads

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.

What it does

The tool splits the genomes into bins of a given length. For each bin, the number of reads found in each BAM file is counted and a correlation of the read coverages is computed for all pairs of BAM files.

Important parameters

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

In the bins mode, the correlation is computed based on **randomly sampled bins of equal length**. The user has to specify the *number* of bins. This is useful to assess the overall similarity of BAM files,
but outliers, such as heavily biased regions have the potential to skew the correlation values.

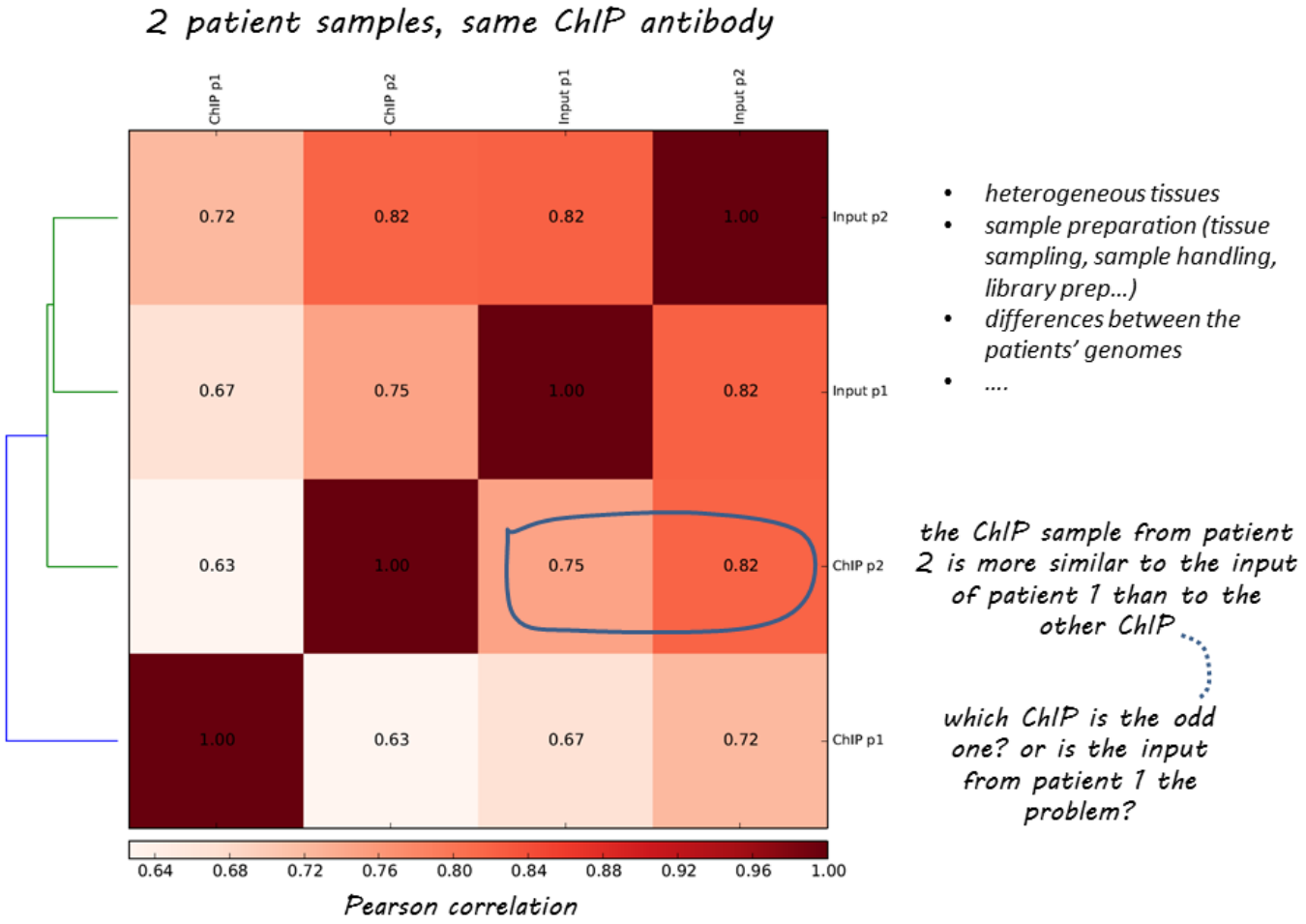
In the BED-file options, the user supplies a list of genomic regions in BED format in addition to (a) BAM file(s). bamCorrelate subsequently uses this list to compare the read coverages for these regions only. This can be used, for example, to compare the ChIP-seq coverages of two different samples for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

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 the result of running bamCorrelate. We supplied four BAM files that were generated from 2 patients - for each patient, there is an input and a ChIP-seq sample (from GSE32222).

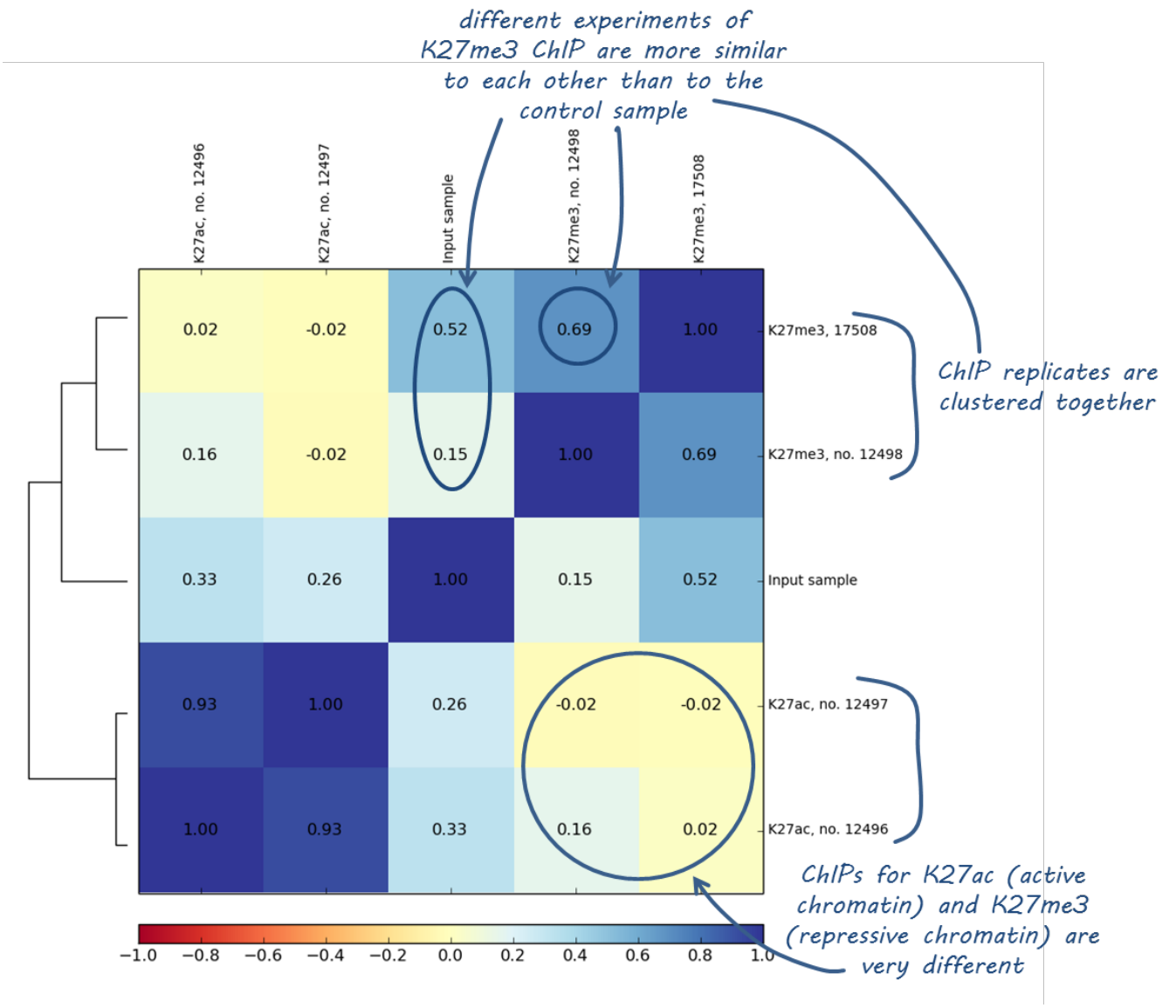


You can supply any number of BAM files that you would like to compare. In Galaxy, you simply have to click "Add BAM file", in the command line you simply list all files one after the other (you can give meaningful name via the --label option).

Here's the command that was used with the command line version:

```
$ deepTools-1.5.2/bin/bamCorrelate bins --fragmentLength 200 --bamfiles \
GSM798383_SLX-1201.250.s_4.bwa.homo_sapiens_f.bam \
GSM798384_SLX-1881.334.s_1.bwa.homo_sapiens_f.bam \
GSM798406_SLX-1202.250.s_1.bwa.homo_sapiens_f.bam \
GSM798407_SLX-1880.337.s_8.bwa.homo_sapiens_f.bam \
--labels "ChIP p1" "ChIP p2" "Input p1" "Input p2" \
--plotFile /eva_data/deeptools_manual/bamCorrelate_bad2.pdf \
--corMethod pearson
```

Here is another example of ChIP samples where H3K27ac was ChIPed by the same experimentator for different cell populations while H3K27me was performed with the same antibody, but at different times. You can see that the correlation between the K27ac 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 K27ac and K27me3 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 (see below for more explanations).

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 be similarly well sequenced.

computeGCbias estimates how many reads with what kind of GC content one should have sequenced given an organism's genome GC content. The calculations are based on the methods published by Benjamini and Speed. The tool first determines how many regions the specific reference genome contains for each amount of GC content, i.e. how many regions in the genome have 50% GC (or 10% GC or 90% GC or...). For this, it samples a large number of equally sized genome bins and counts how many times we see a bin with 50% GC (or 10% GC or 90% or...). These **expected values are independent of any sequencing, but they do depend on the respective reference genome**. This means, that the expected values will, of course, differ between mouse and fruit fly due to their genome's different GC contents, but it also means that strong biases in the reference genome assembly might lead to a false positive diagnosis of GC bias.

After the expected values, the tool samples the BAM file of sequenced reads. Instead of noting how many genomic regions there are per GC content, we now count the **reads per GC content**.

Output files

- **Diagnostic plot**
  - box plot of absolute read numbers per genomic GC bin
  - x-y plot of observed/expected read ratios per genomic GC content bin

- **Data matrix**
  - to be used for GC correction with *correctGCbias*

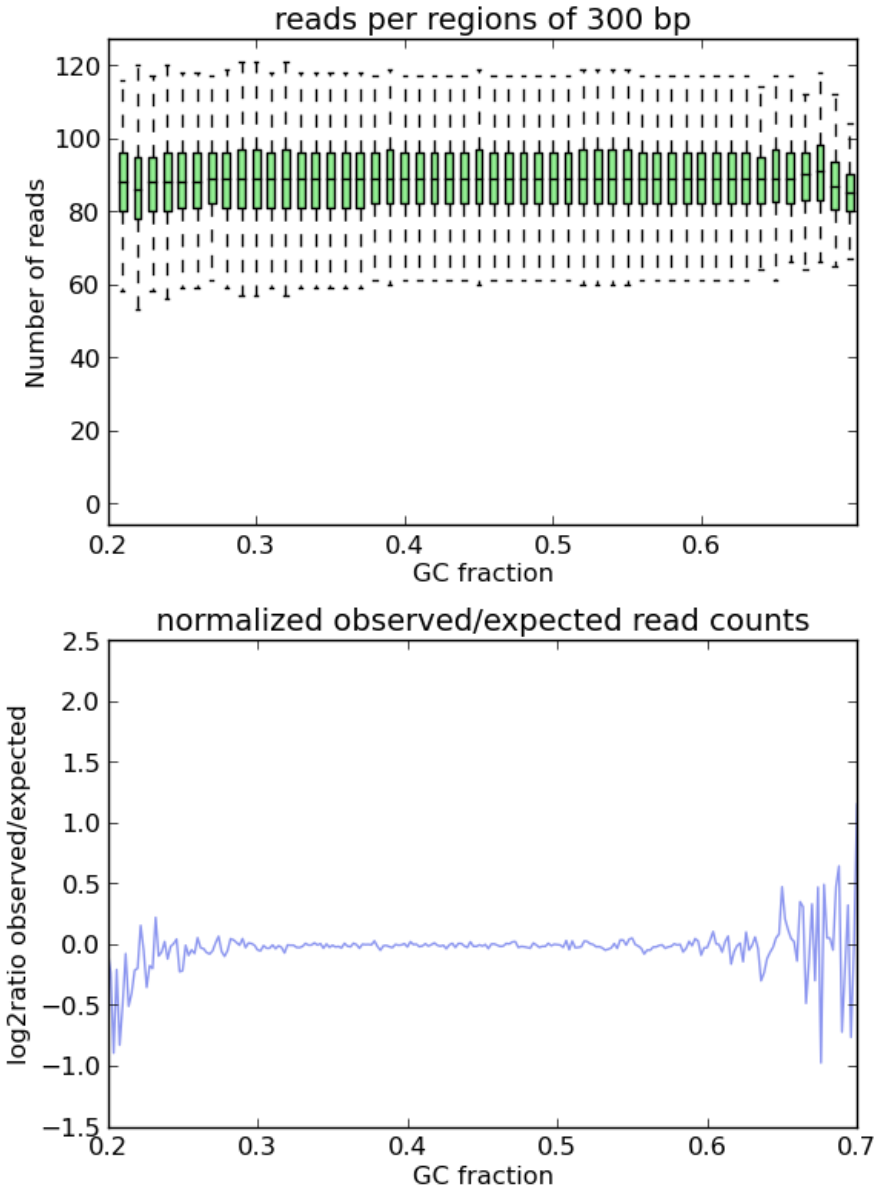
What the plots tell you

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.

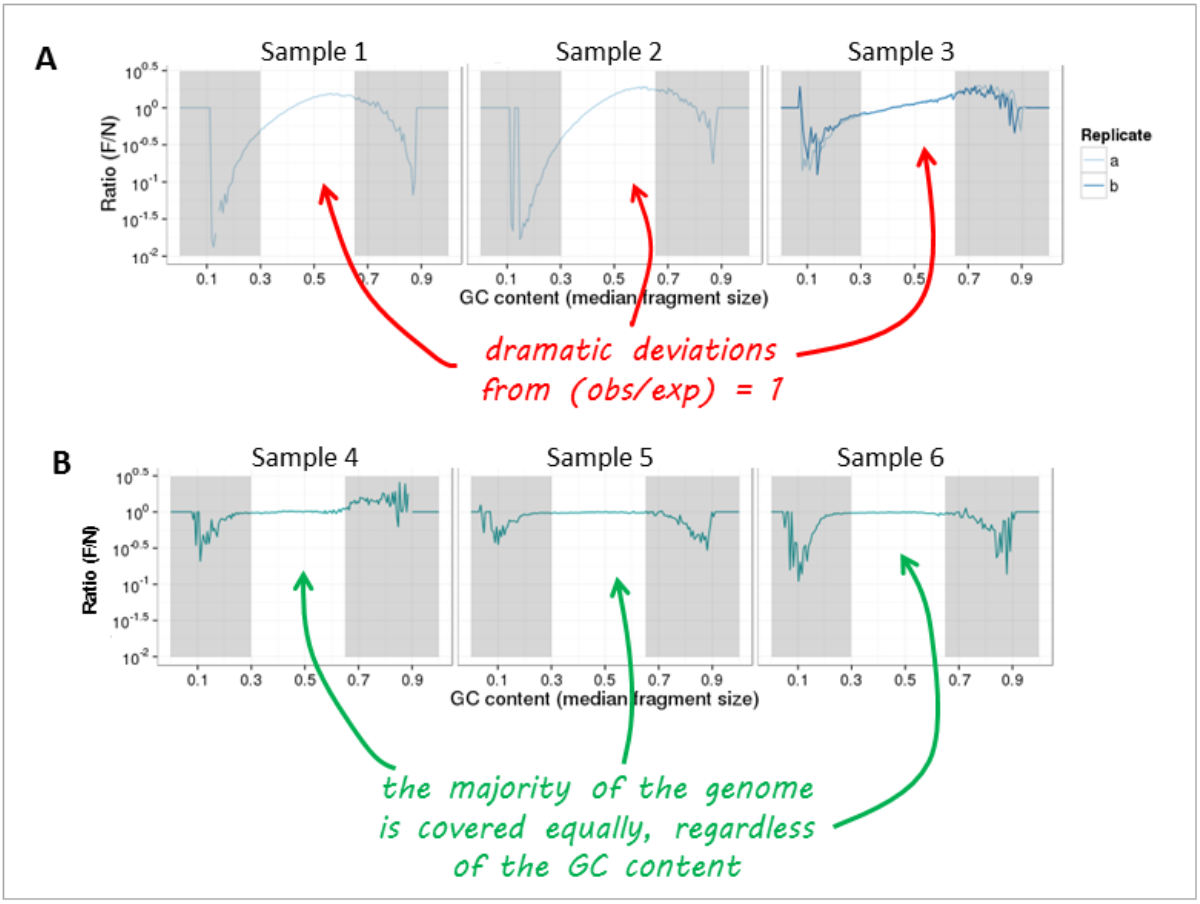
Example figures

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



As you can see, both plots do not show enrichments or depletions for specific GC content bins.

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

What it does

This tool is based on a method developed by Diaz et al..
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, i.e. whether the signal of the enrichment can be clearly distinguished from the background.

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:

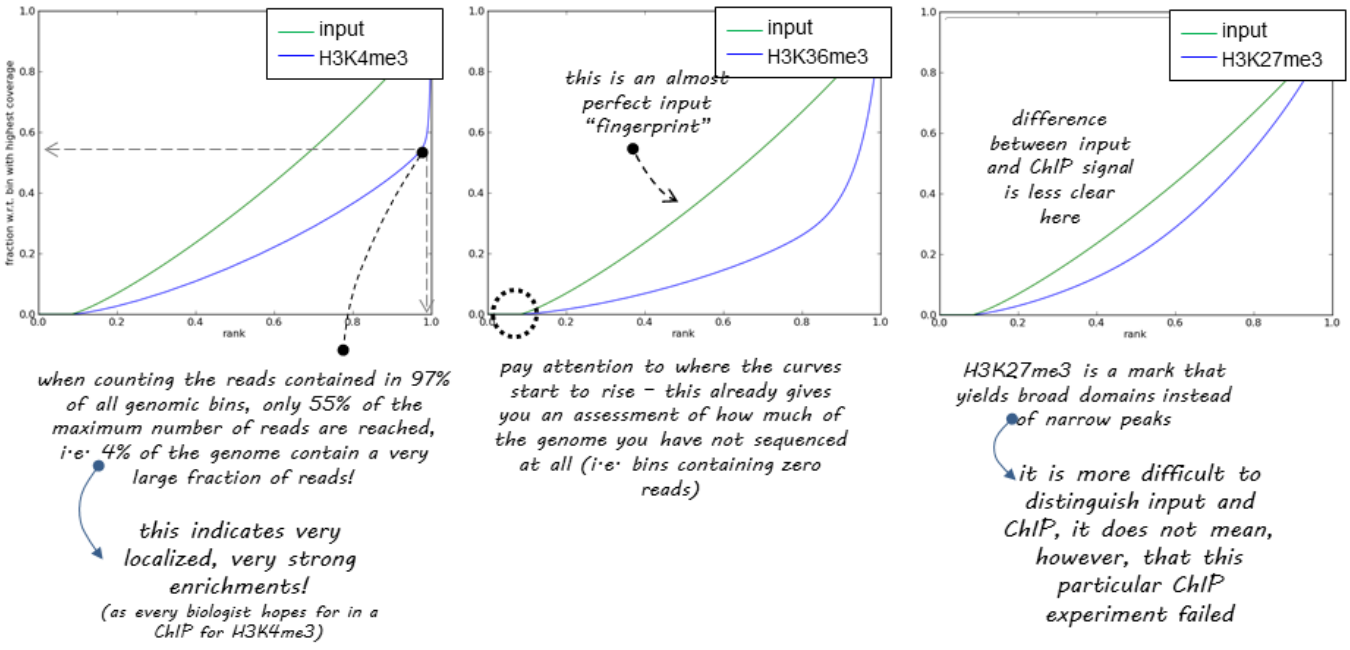
- **Diagnostic plot**
- **Data matrix** of raw counts (optional)

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 three different fingerprint plots that we routinely generate to check the outcome of ChIP-seq experiments.
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 fill 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.

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

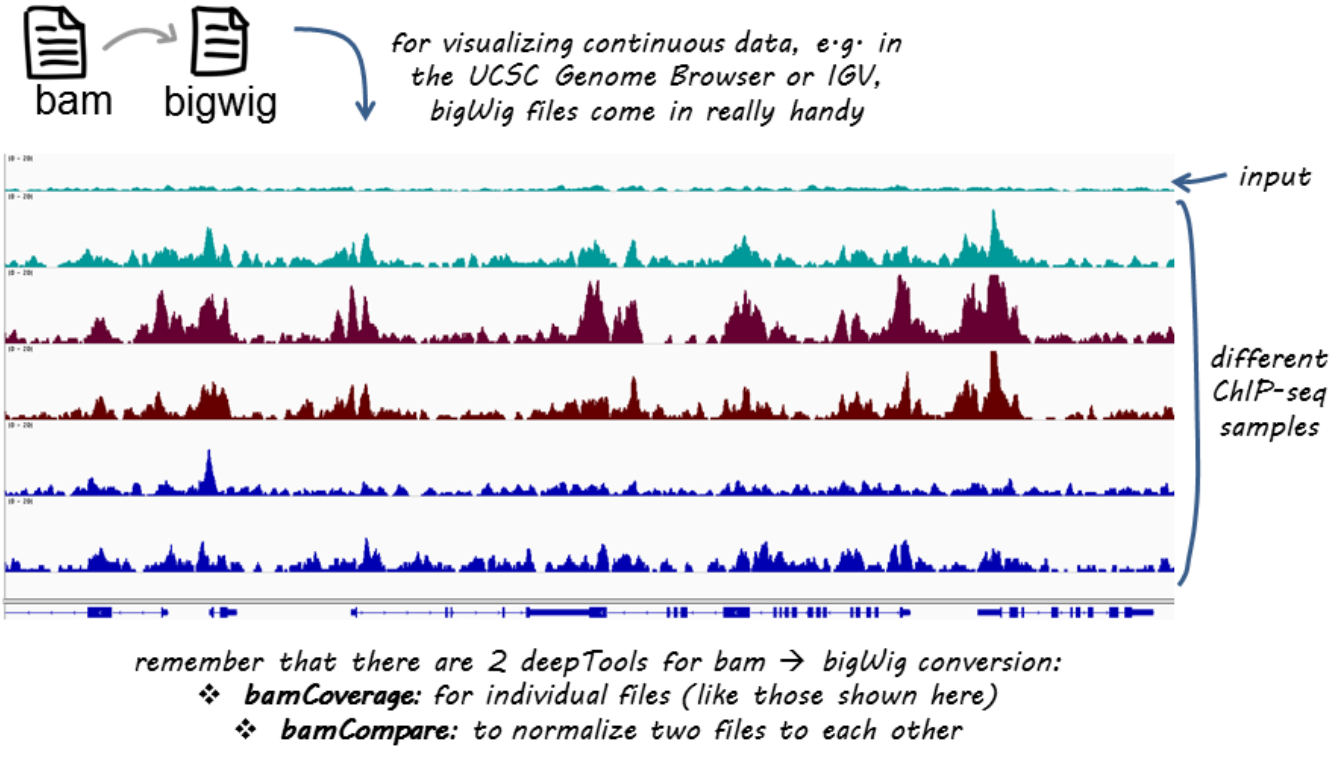


Table of Content

- 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

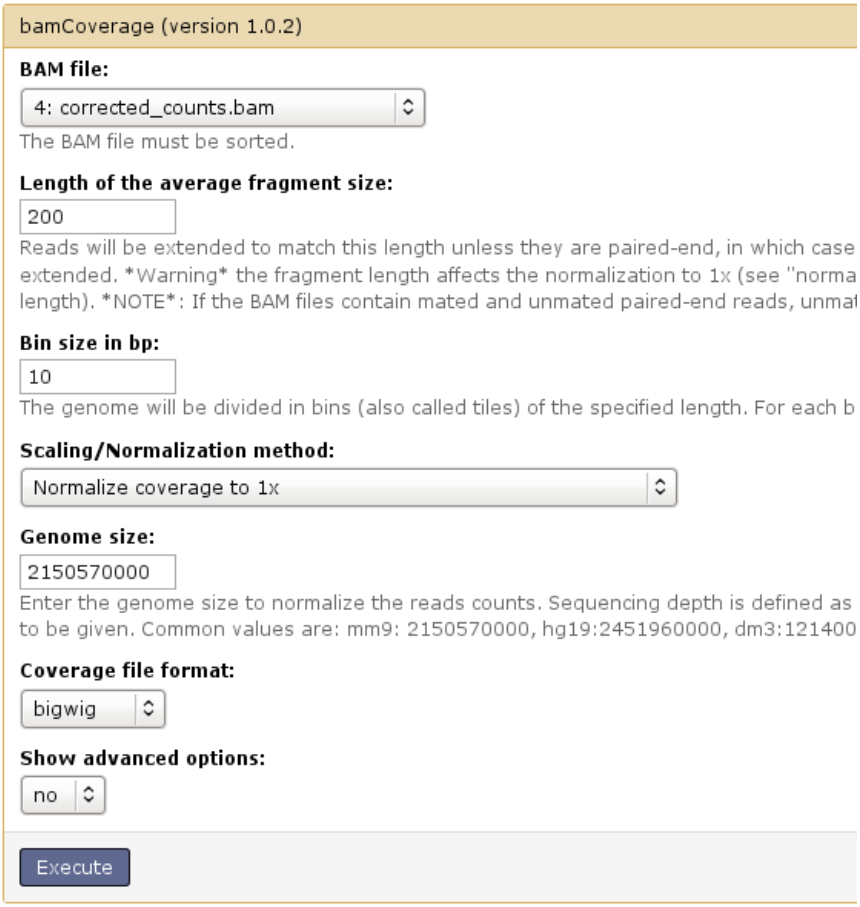
Usage

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 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:



bamCompare

What it does

This tool compares two BAM files based on the number of mappedreads. 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

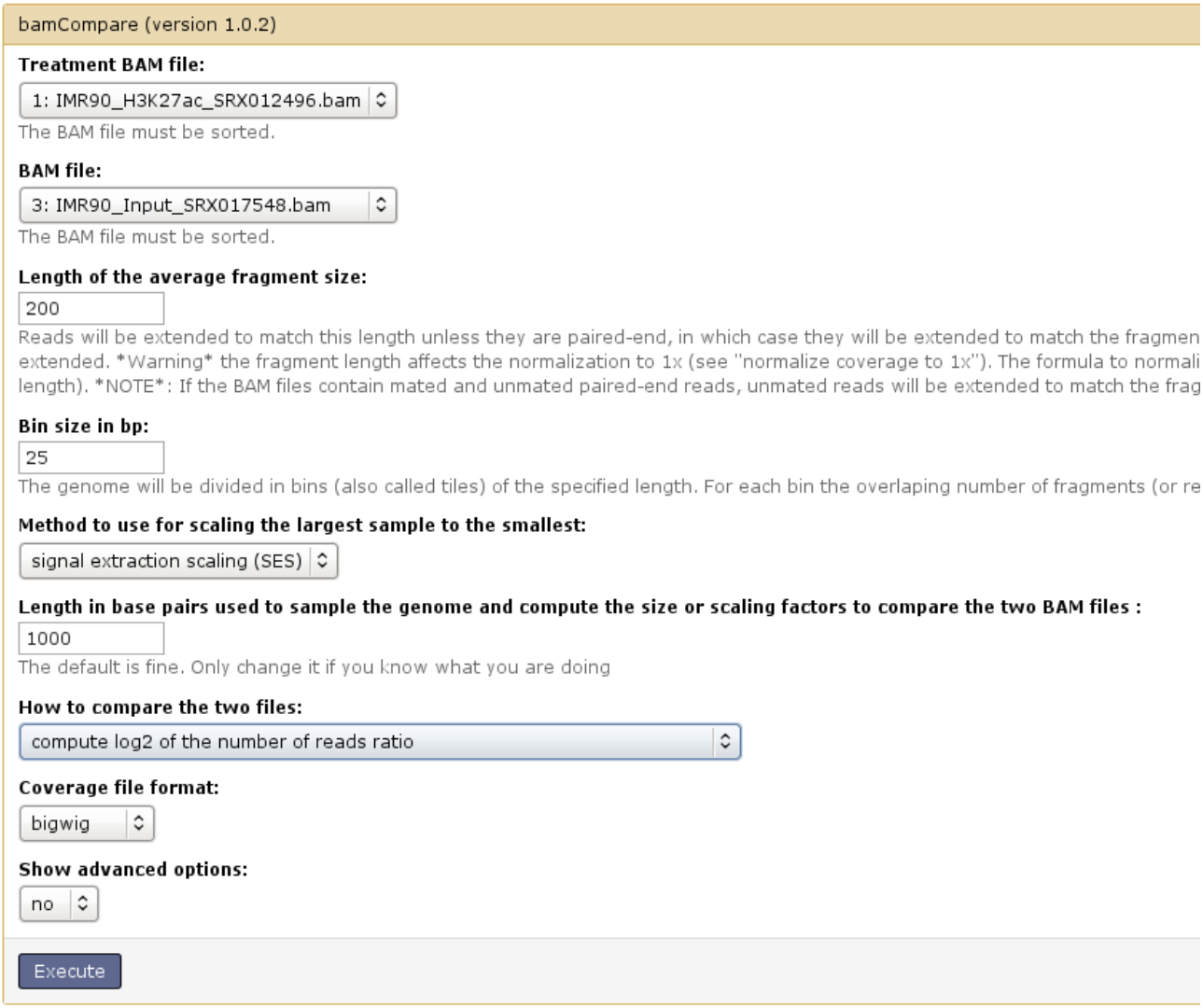
- same as for bamCoverage, except that you now obtain **1** coverage file that is based on **2** BAM files.

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
```

The Galaxy equivalent:

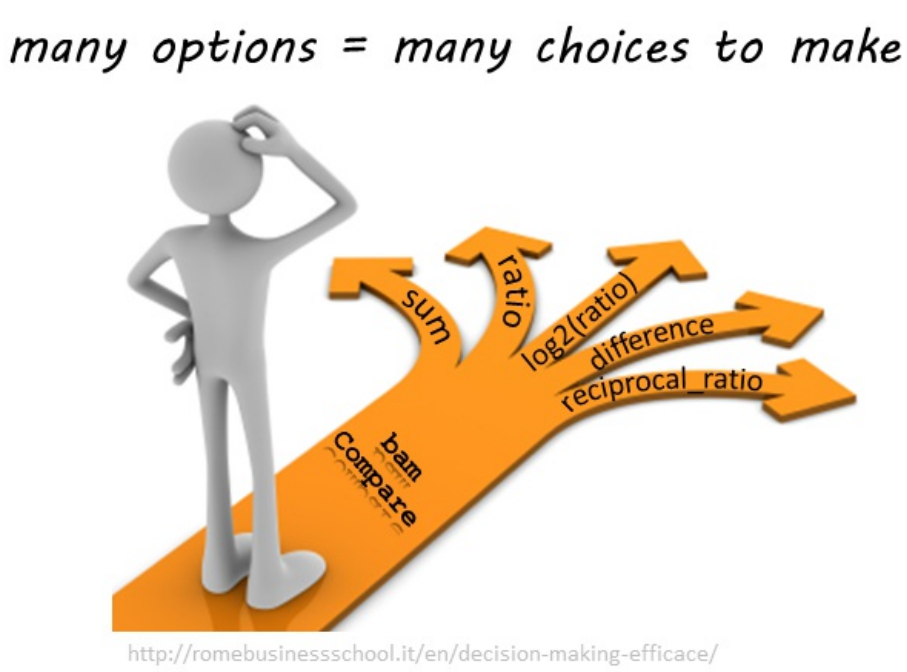


Note that the option "missing Data As Zero" can be found within the "advanced options" (default: no).

- 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 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

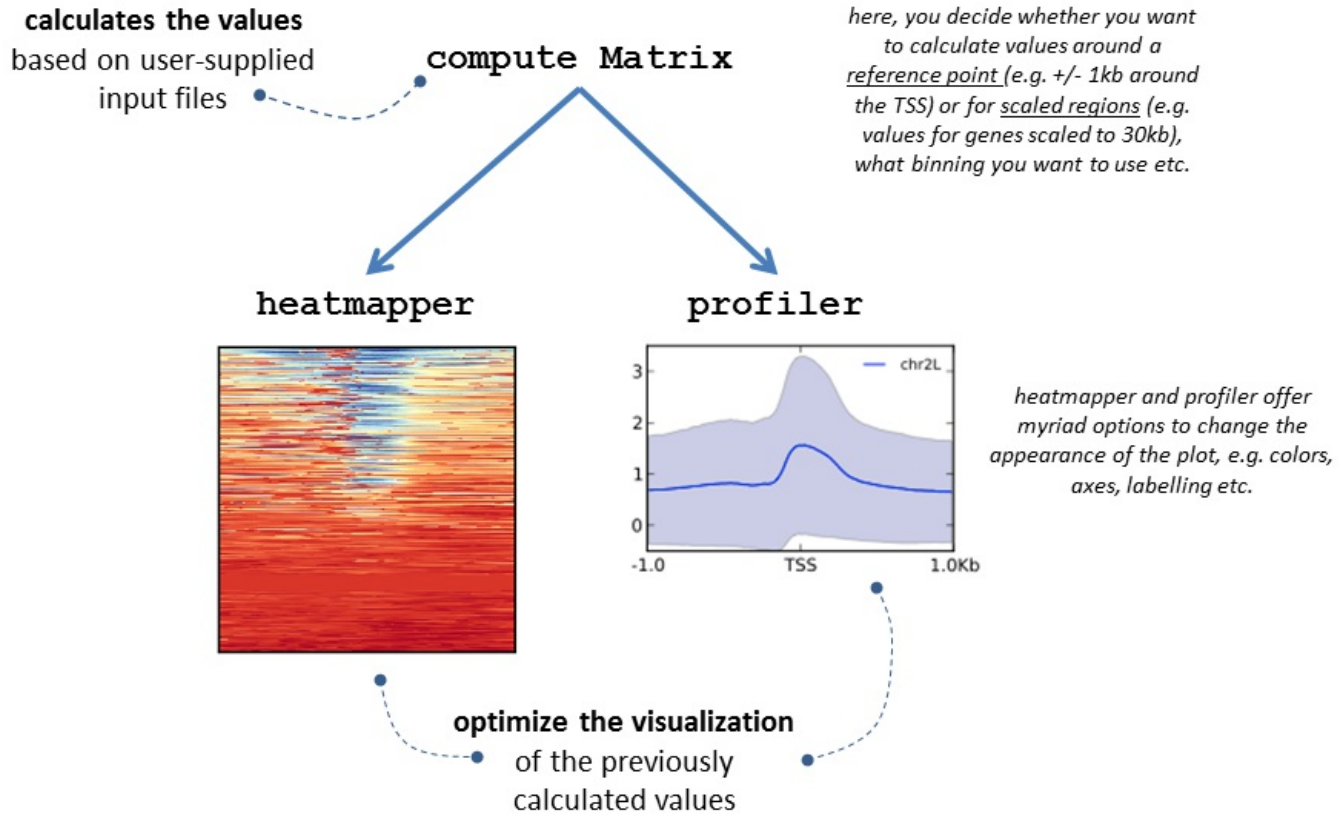
Some (more) parameters to pay special attention to

- --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 **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 a tool that calculates the values (*computeMatrix*) and two 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 tweeking 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.

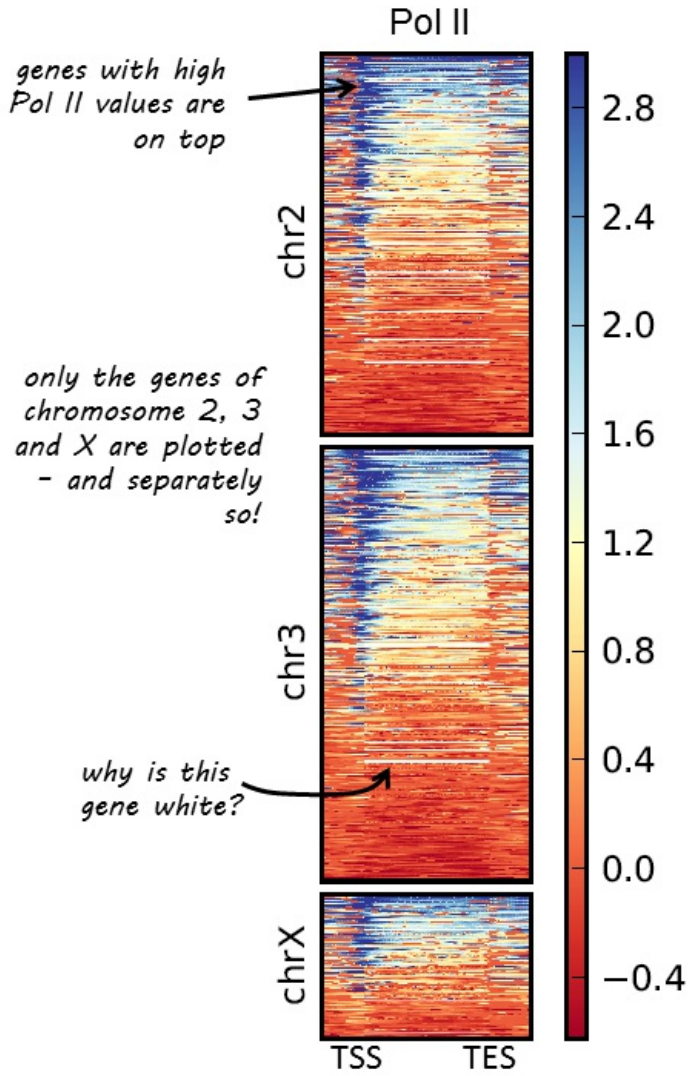
profiler

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.

Heatmap with all genes scaled to the one size and user-specified groups of 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 only"
```

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.

This is what you would have to select to achieve the same result within Galaxy:

computeMatrix

computeMatrix (version 1.0.2)

regions to plots

regions to plot 1

Regions to plot:

3: Dm.530_genes_chrX.bed

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

Label:

ChrX

Label to use in the output.

Remove regions to plot 1

regions to plot 2

Regions to plot:

8: Dm.530_genes_chr3.bed

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

Label:

Chr3

Label to use in the output.

Remove regions to plot 2

regions to plot 3

Regions to plot:

7: Dm.530_genes_chr2.bed

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

Label:

Chr2

Label to use in the output.

Remove regions to plot 3

the genes of each chromosome are supplied as individual BED-files

Add new regions to plot

Score file:

4: PolII.bw

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

computeMatrix has two main output options:

scale-regions

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 those genomic positions before (downstream) and/or after (upstream) the reference point will be plotted.

Distance in bp to which all regions are going to be fitted:

1500

Label for the region start:

TSS

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

Label shown in the plot for the region end. Default is TES (transcription end site).

Set distance up- and downstream of the given regions:

yes

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.

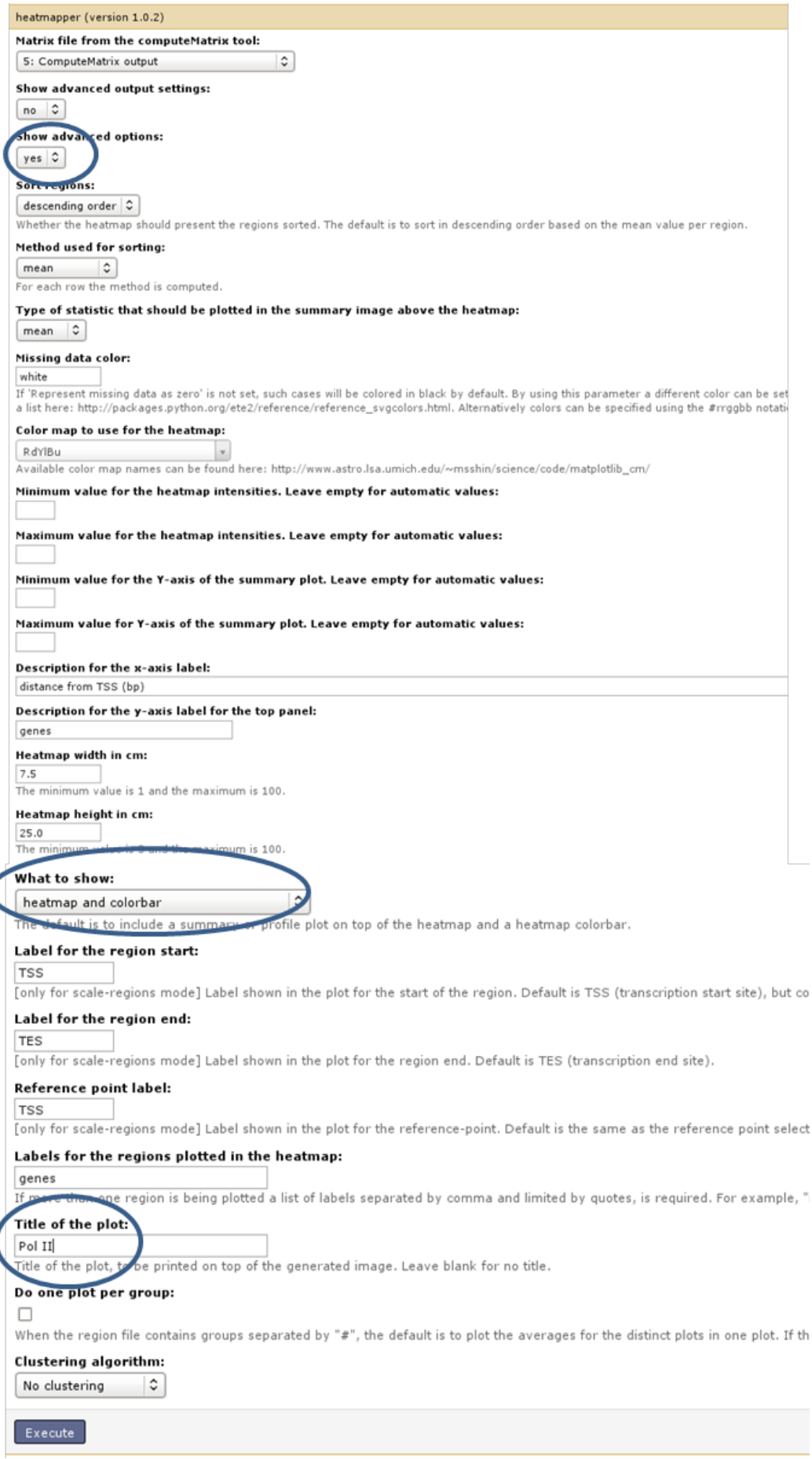
Maximum threshold:

Any region containing a value that is equal or higher that this numeric value will be skipped. The max threshol
average values.

Scale:

If set, all values are multiplied by this number.

Execute



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 by using the "Filter" tool three times:

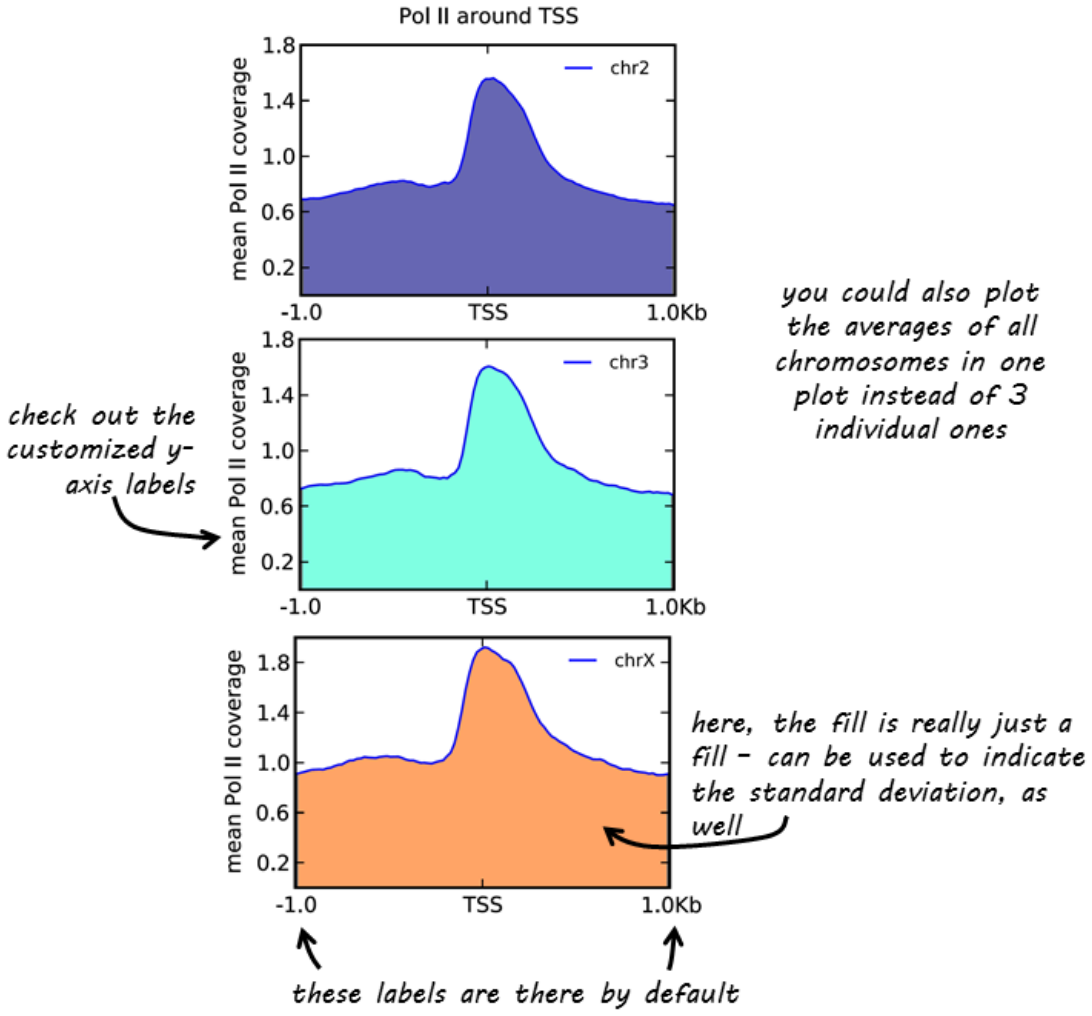
1. c1=="chr2" --> Dm.genes.chr2.bed
2. c1=="chr3" --> Dm.genes.chr3.bed
3. c1=="chrX" --> Dm.genes.chrX.bed

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).

Summary plots

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

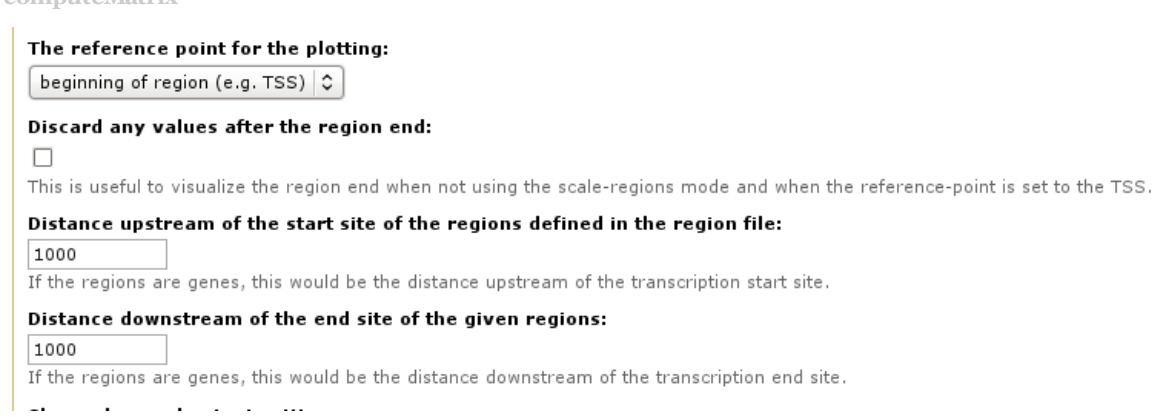


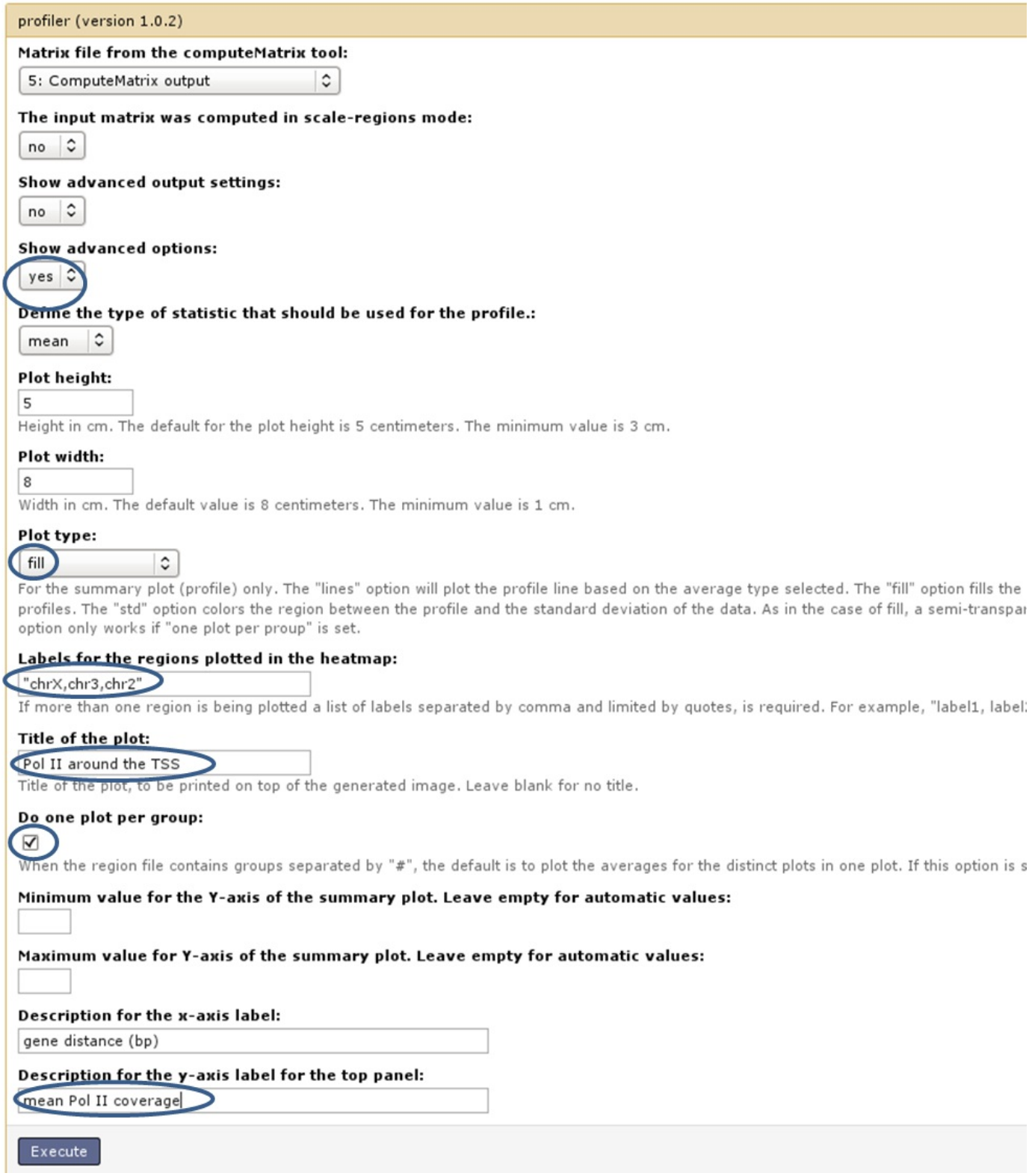
We used the same [BED][] file(s) as for the heatmap, hence the 3 different groups (1 per chromosome). However, this time we used computeMatrix not with *scale-regions* but with *reference-point* mode.

```
$ /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):

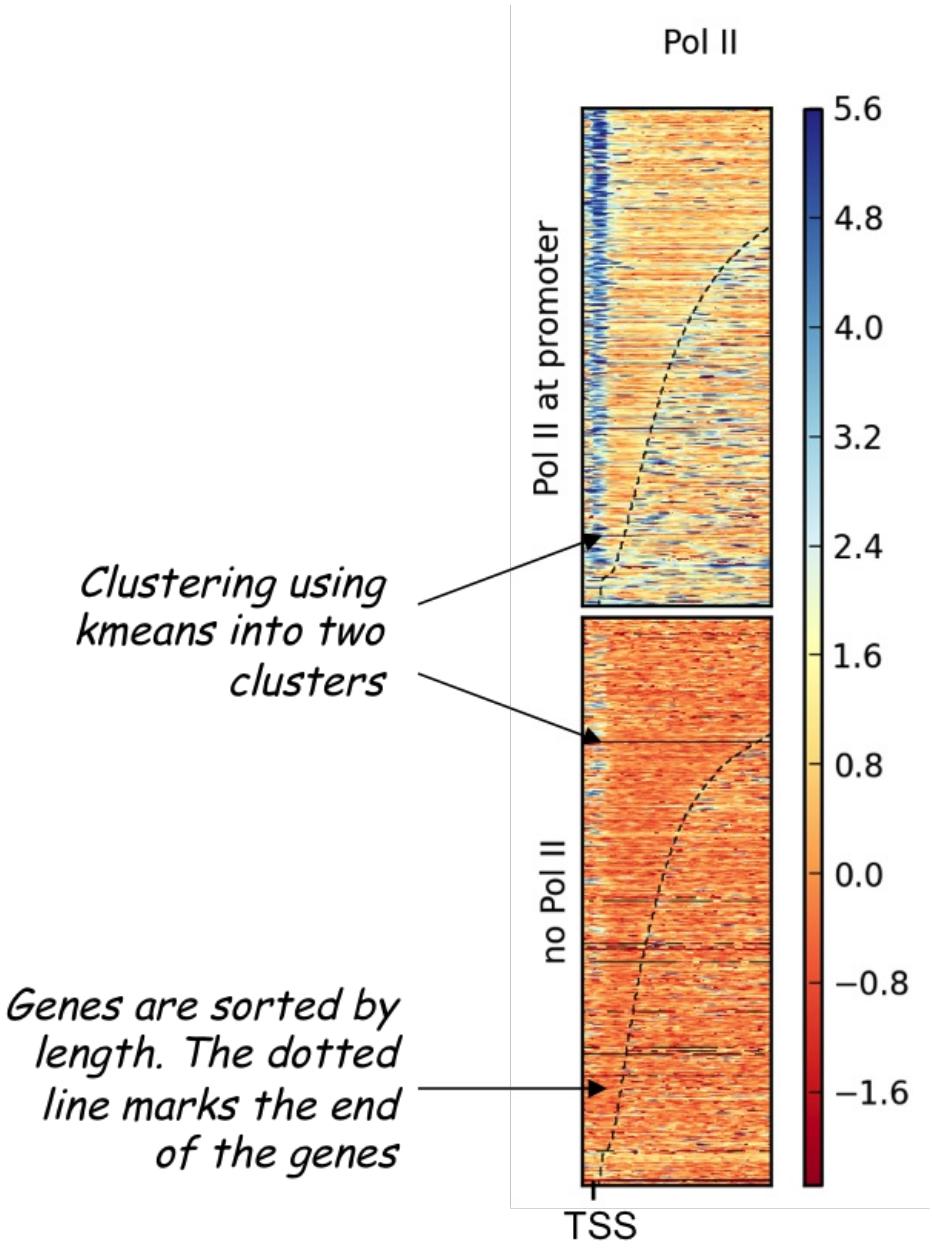




Heatmap with all genes scaled to the one size and kmeans clustering

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 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 500 --binSize 50 \
--outFileName PolII_matrix_TSS
$ /deepTools-1.5.2/bin/heatmapper --matrixFile PolII_matrix_TSS --kmeans 2 \
--outFileName PolII_two_clusters.pdf --plotTitle "Pol II" \
--sortUsing region_length --whatToShow "heatmap only"
```

When the `--kmeans` option is chosen and more than 0 clusters are specified, heatmapper will run the [k-means][] clustering algorithm. In this example *Drosophila m.* genes were divided into two clusters separating those genes with 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 (i.e. all genes, without any hash tags separating them)

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 down below.

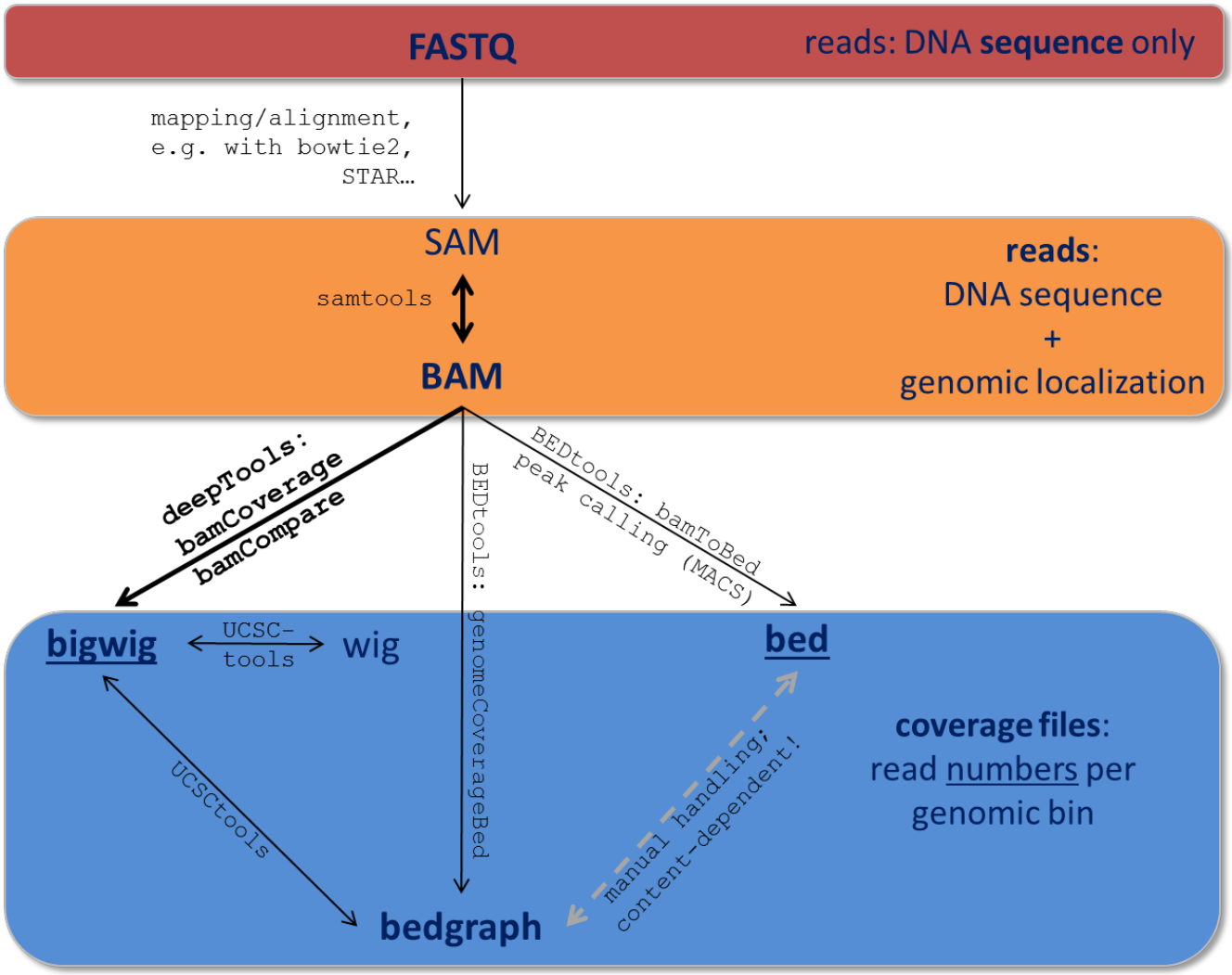
Abbreviations

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
DNase	deoxyribonuclease	micrococcal nuclease
HTS	high-throughput sequencing	next-generation sequencing, massive parallel short read sequencing, deep sequencing
MNase	micrococcal nuclease	DNase
NGS	next-generation sequencing	high-throughput (DNA) sequencing, massive parallel short read sequencing, deep sequencing

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. Be aware, that the sorting here is purely alphabetically, not according to their usage within an analysis pipeline that is depicted here:



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

- 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

bedGraph

- text file
- similar to bed file, except that it can ONLY contain 4 columns and 4th column MUST be a score
- again, read the UCSC description for more details

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

FASTQ

- text file
- contains raw read information (e.g. base calls, sequencing quality measures etc.), but not information about where in the genome the read originated from

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
