BioC for HTS - PDCB topic
ChIP-seq 01

LCG Leonardo Collado Torres
lcollado@wintergenomics.com – lcollado@ibt.unam.mx

December 1st, 2010

ChIP-seq overview

BayesPeak

General workflow

From the BayesPeak vignette:

- Chromatin ImmunoPrecipitation (ChIP) is an experiment designed to study protein-DNA interactions, particularly to identify the genomic sites where proteins, such as transcription factors, bind to the DNA, or sites where histone modifications occur. The experiment produces samples that are enriched for the sites of interest compared to the rest of the genome. The use of this method combined with high-throughput sequencing of the samples is referred to as ChIP-seq.
- Given our protein of interest, the ChIP-seq protocol usually consists of the following steps.
  1. Cross-linking the proteins to the DNA - The protein is permanently bound to the DNA, usually with formaldehyde.

General workflow

2. Shearing - The cells are lysed, and the DNA is randomly cut into small fragments by sonication.
3. Immunoprecipitation - An antibody specific to the protein of interest is used to isolate the protein and the attached DNA fragments. The resulting sample is enriched for those genomic regions. If we are instead interested in locating histones with a certain epigenetic modification, then we use an antibody specific to histones with that modification.
4. Reverse crosslinking and purification - The bonds between the protein and DNA are broken. The DNA is subsequently purified.

General workflow

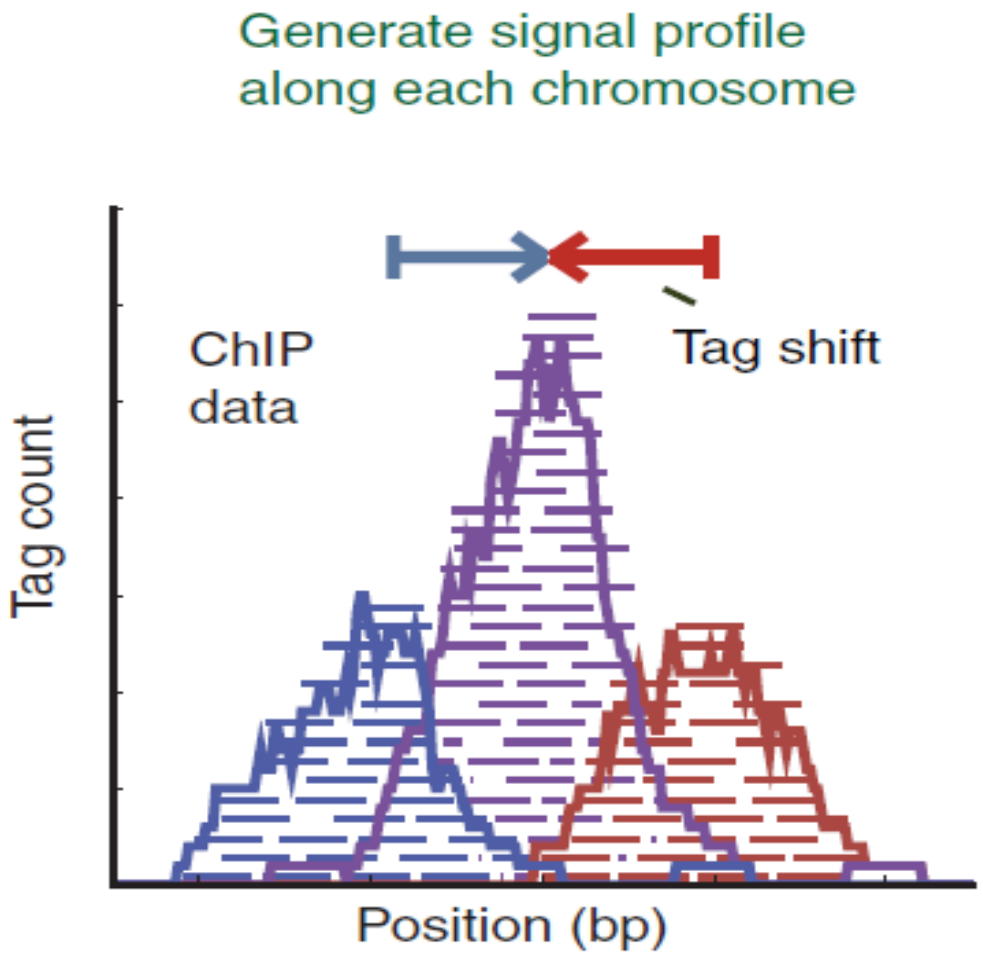
5. Sequencing - The contents of the samples are size selected such that the fragments' length lies in the region of 200-300 bp. This step is required by the sequencing protocol. Adaptors are attached to the fragments and amplification usually takes place. Subsequently, the sample undergoes "high-throughput sequencing" during which the sequences of the ends of the present fragments are identified. For ChIP-seq applications usually one end of each fragment is sequenced to produce "single-end" reads.
6. Alignment - The DNA is aligned back to reference genome, taking the quality of the reads into account. Usually only the reads that map to unique locations in the genome are included in the downstream analysis.

General workflow

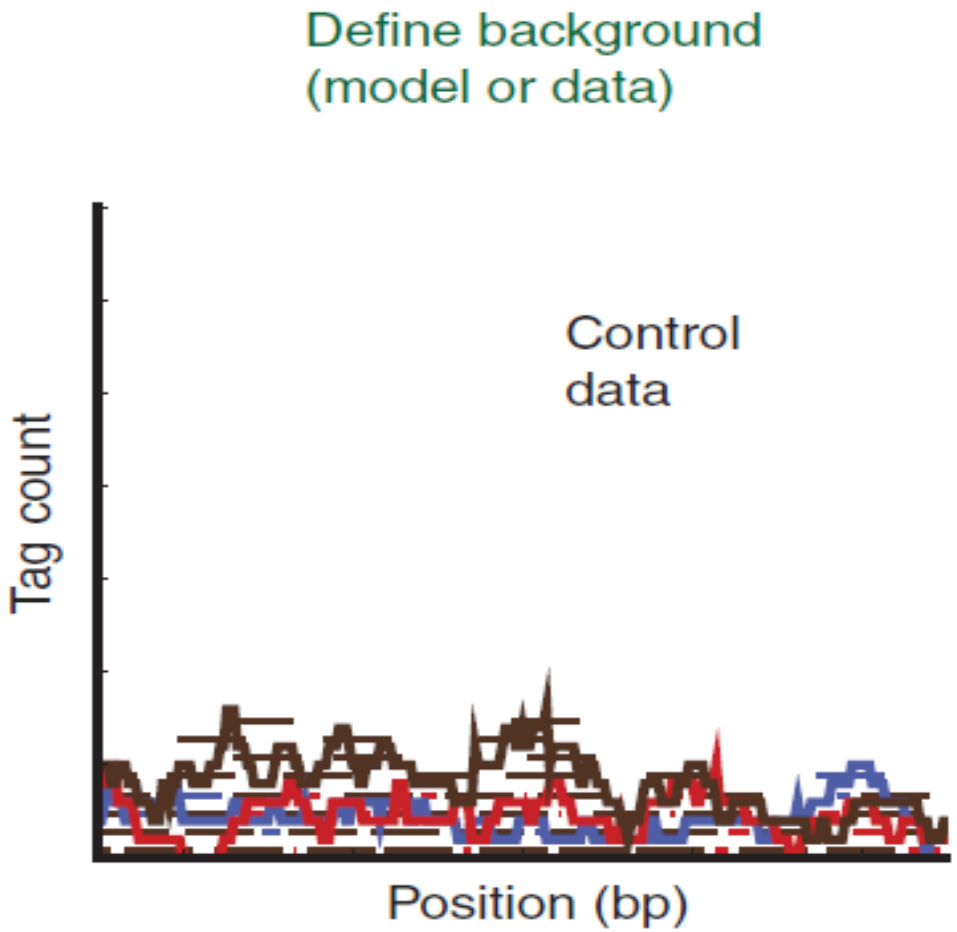
7. Analysis - The sites of interest correspond to the genomic regions where there is high abundance of reads compared to the background or the control sample. Peak-callers carry out this step.

- Usually this process is repeated omitting the immunoprecipitation step to produce a sample with no preferential enrichment. This control sample has the same characteristics as ChIP-seq data and its inclusion is important to identify experimental biases.
- There are many sources of error - for example, misalignment, impurities, or DNA that simply has a high affinity for being sequenced - which can result in noise across the genome, or even false peaks.

Peak finding

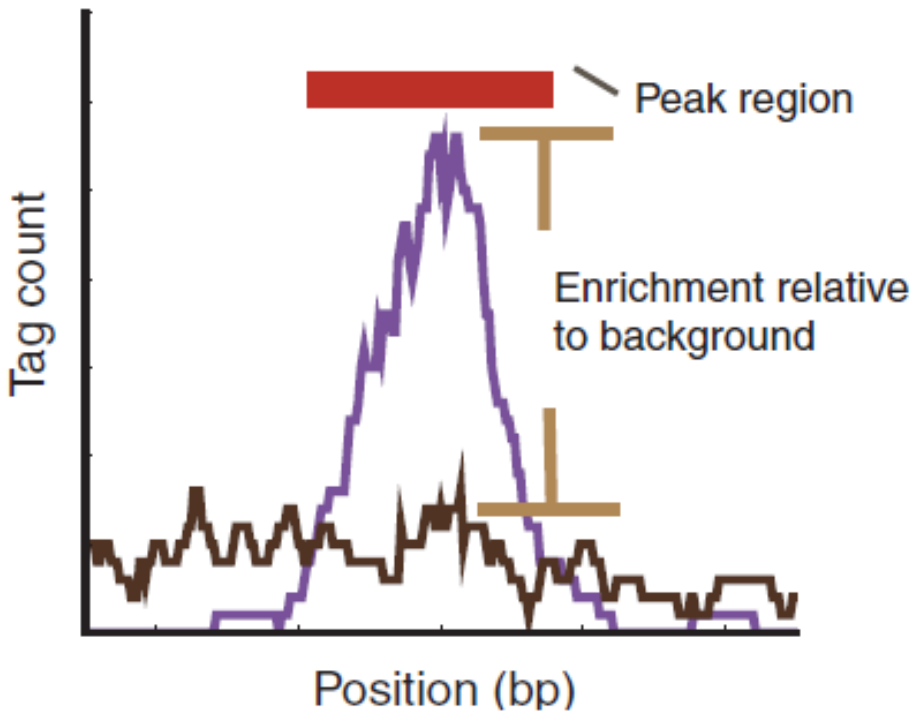


Peak finding

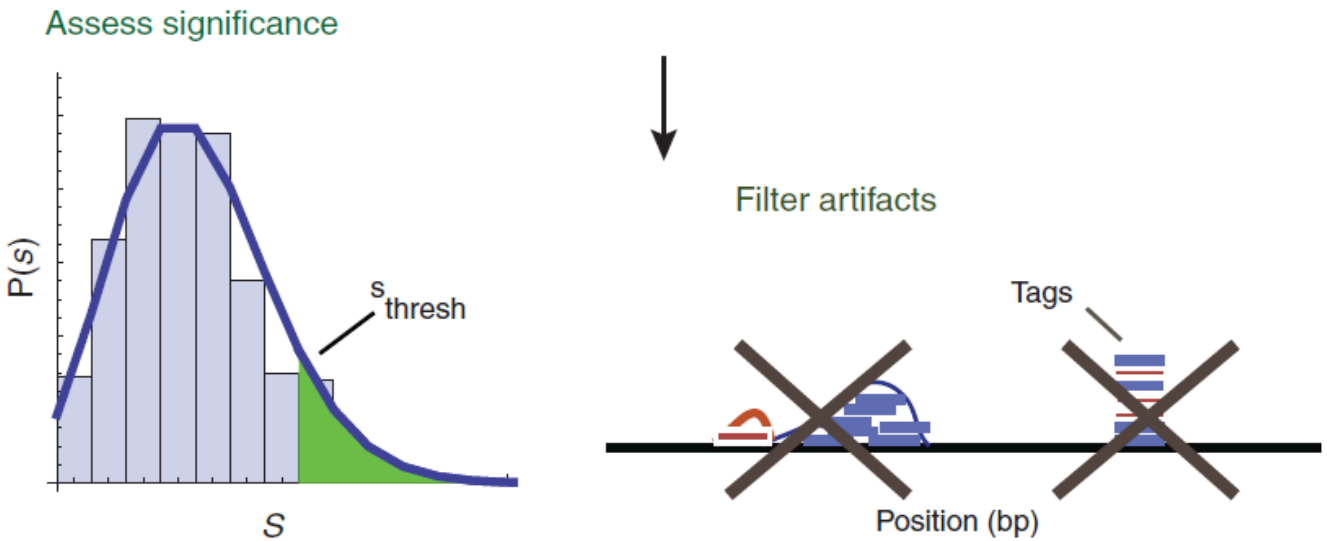


Peak finding

Identify peaks in ChIP signal



Peak finding



Lots of peak-callers

- ChipSeq Peak Finder clusters the sequenced reads (i.e., the beginnings of the fragments), and uses the ratio of the counts from the immunoprecipitated sample to the control in order to identify (or call) regions where large numbers of fragments overlap as "peaks".
- eRange, also allows the use of reads that map to multiple locations in the genome, which results in an increase in the amount of data applied to peak-calling.
- XSET uses the full estimated length of the DNA fragments to identify the regions with the highest numbers of overlapping fragments.

Lots of peak-callers

- the method in Mikkelsen et al. takes into account the "mappability" of the underlying sequence, by excluding regions from the reference genome that correspond to multiple occurrences of the same short sequences, and computes p-values to find significant differences between the observed and expected numbers of fragments.
- Peak-Seq, another algorithm that allows for this mappability effect, starts with a normalization step comparing the control to the background component of the ChIP sample and then, using the Binomial distribution, identifies significantly different concentrations of reads between the two samples.
- by examining only the start of protein-bound fragments, we can identify peaks offset on the forward and reverse strands of the DNA, the true binding site lying somewhere in between

Lots of peak-callers

- Model-based Analysis for ChIP-seq (MACS) shifts the reads on the forward and reverse strands together, and uses the Poisson distribution to identify the density of reads in enriched and non-enriched regions in order to call peaks. In addition, the method identifies multiple identical reads to avoid biases during amplification and sequencing library preparation.
- Quantitative enrichment of sequence tags (QuEST) also shifts the peaks from opposite strands together and derives a kernel density estimation score to call the enriched regions.
- FindPeaks calls peaks according to some minimum height criteria without including a control sample in the analysis.

Lots of peak-callers

- Short Sequence Reads (SISSR), estimates Poisson probabilities of high read counts, and calls regions where the peaks shift from the forward to the reverse strand.
- In Kharchenko et al. three similar peak calling methods are proposed that score read counts upstream and downstream of each region to match read patterns in the forward and reverse strands.
- Nix et al. have simulated spike-in data, combined them with control reads from real experiments and used different metrics to score the peaks while controlling for false discoveries.

More tools

Pepke et al revised more public ChIP-seq tools:

- Useq
- ssp
- SICER
- GLITR
- F-Seq
- ERANGE
- CisGenome

Differences

The methods differ in

- whether they require a control sample (control) or whether they only use the ChIP sample (IP only)
- whether they take into account the (average) length of the reads/fragments and their orientation
- whether they take into account the different DNA strands or if they merge the reads, and whether the reads are shifted towards the 3' end
- whether an externally estimated mappability file is used
- how the scores, on which the classifications are based, are estimated
- whether/how any FDR or sensitivity/specificity estimates are calculated

Differences

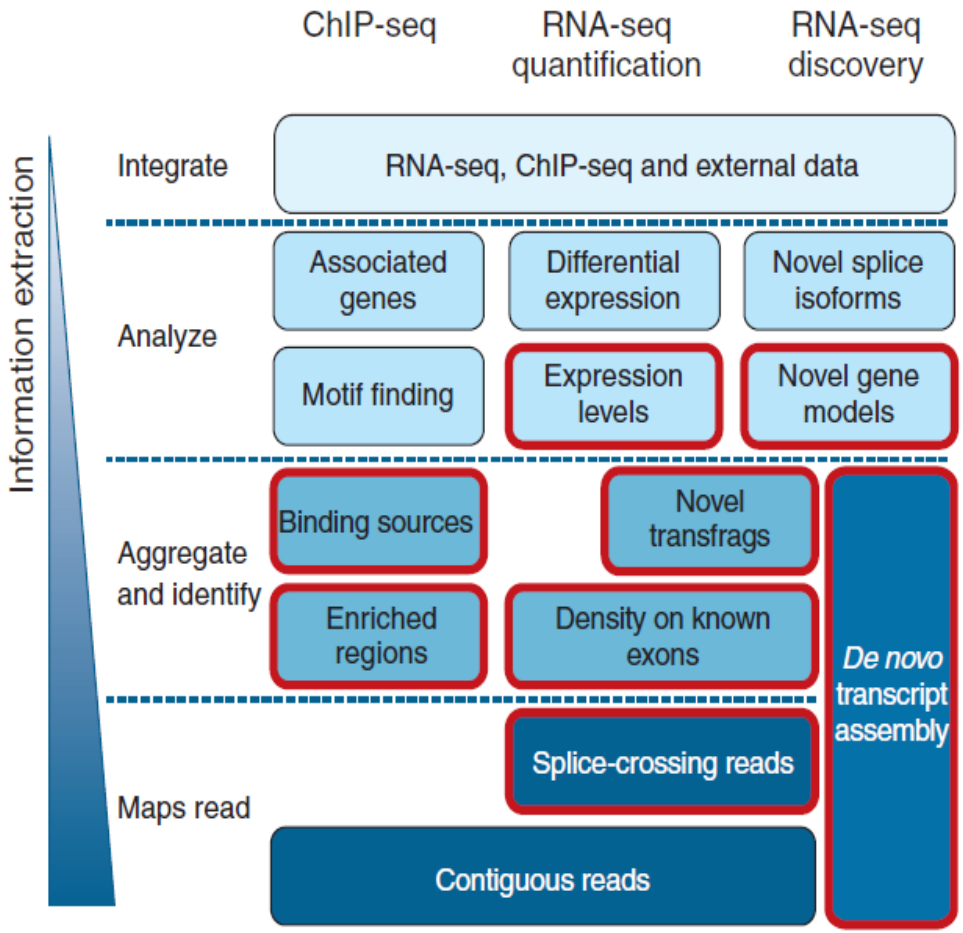
- whether or not the method is applicable to both transcription factor (TF) and histone mark data.

In Bioconductor

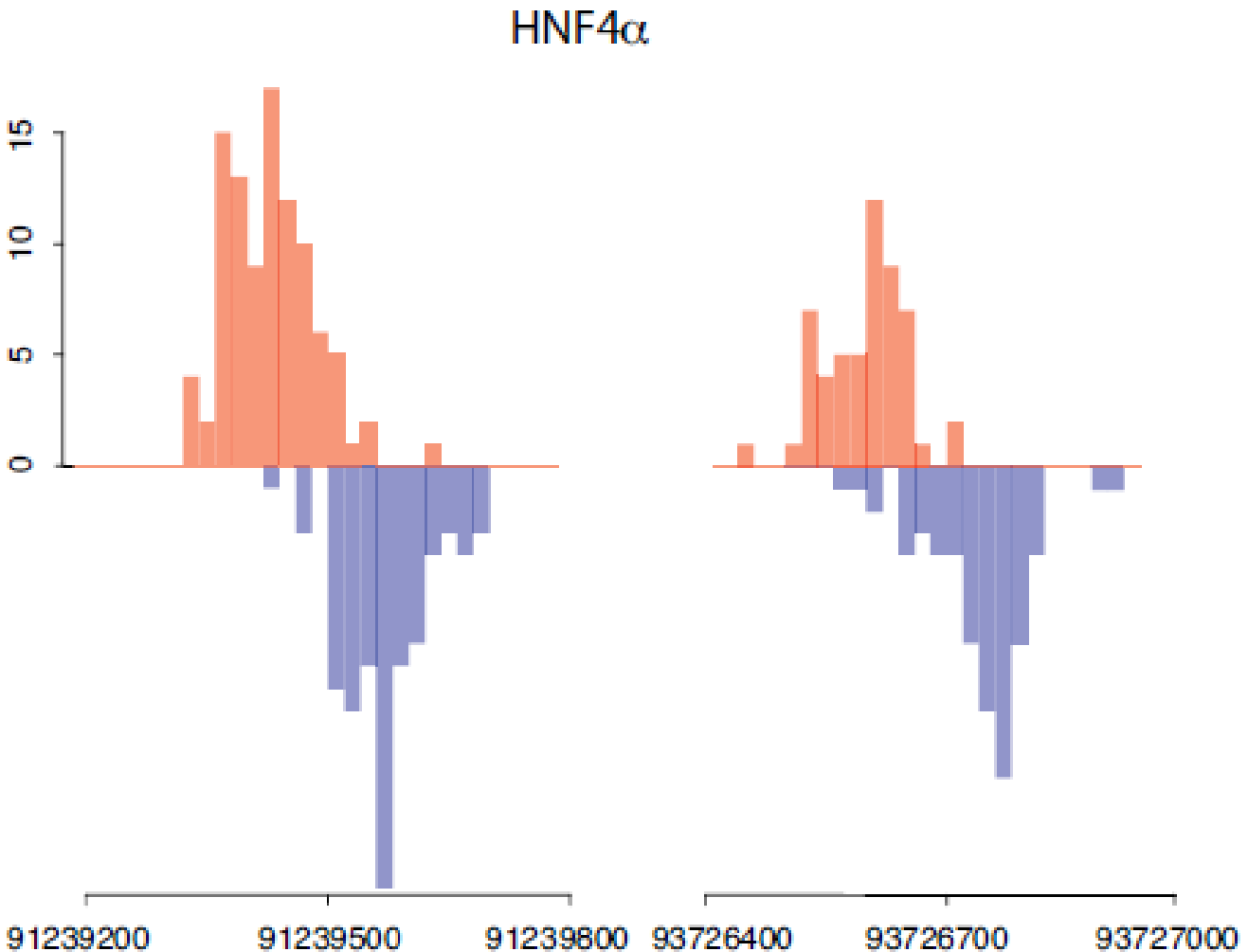
`http://bioconductor.org/packages/2.7/ChIPseq.html`

- BayesPeak: Bayesian posterior probabilities and negative binomial
- ChIPseqR: focuses on nucleosomes and the distance between peaks is important
- ChIPsim: same author as ChIPseqR, simulates ChIP-seq data
- CSAR: poisson test, controls FDR with a simple threshold
- PICS: is quite complete, also uses Bayes, uses the mappability of the genome
- Note that peak-calling is computationally intensive and you will most likely need to run them in Linux to use `multicore` or `snow`

The story continues with motif finding



Peaks

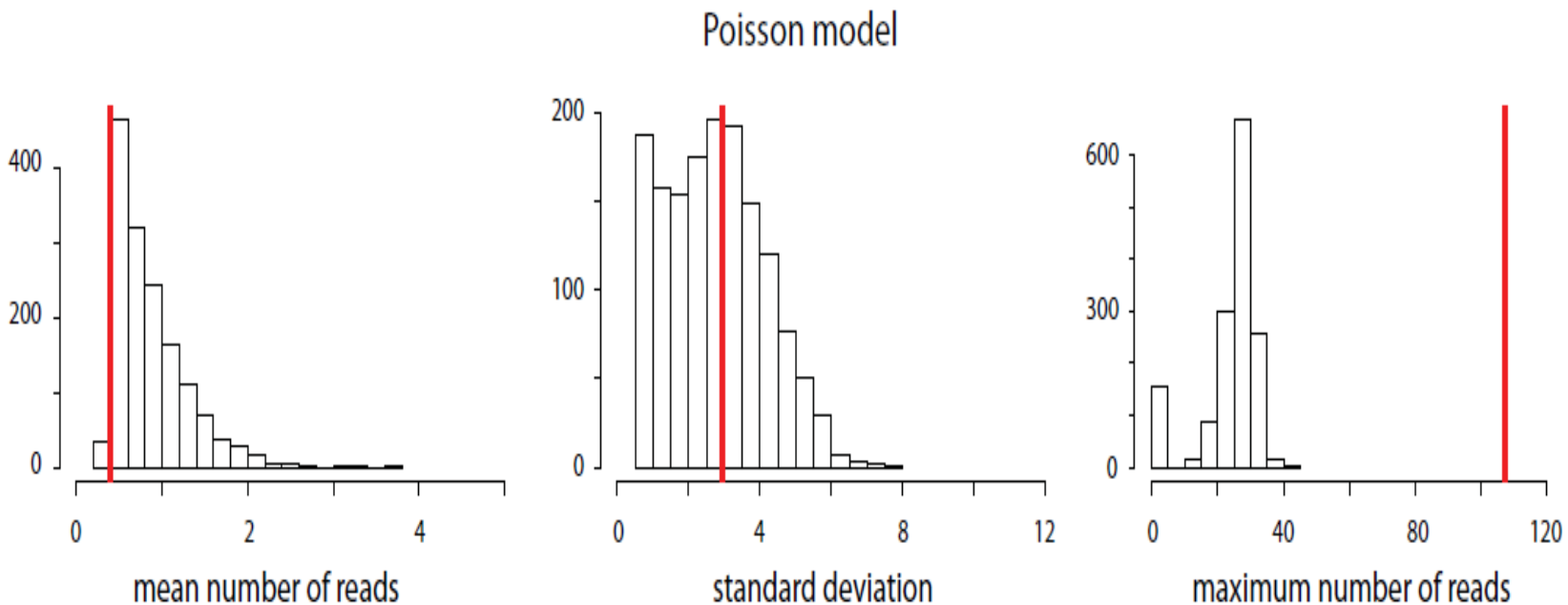

HNF4α
0
5
10
15
91239200
91239500
91239800
93726400
93726700
93727000

NB vs Poisson

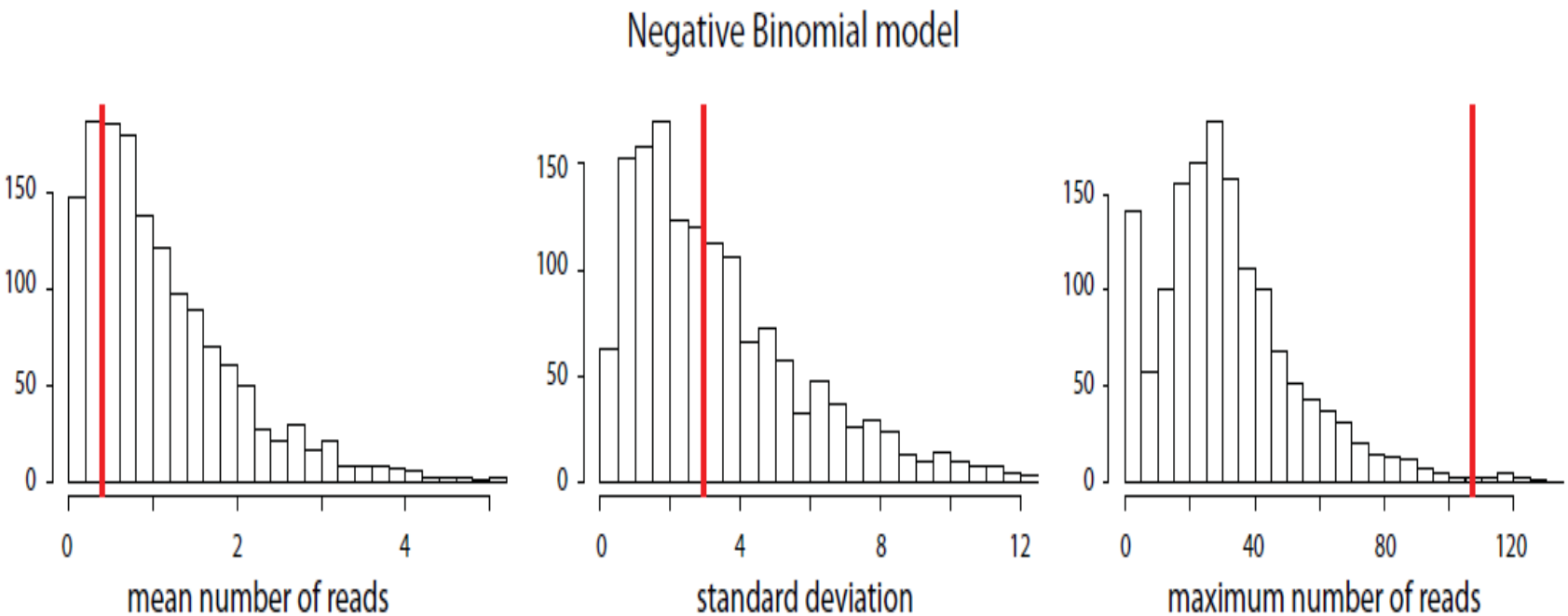
Figure 4

Checking the model fit: Histograms of the discrepancy statistics for the HNF4α data. The red lines represent the value we observe from the data for the mean number of peaks per window, their standard deviation, and the maximum possible read score. The histograms are plots of the same variables estimated using the 3,000 simulated values of the parameters during the run of the algorithm. The closer the real value is to the simulated distribution, the better the model explains those aspects of the data. The first row shows the three histograms of the values generated using the Poisson model and the second row shows the corresponding values for the negative binomial model. The latter shows a notable improvement in explaining these features of the data.

NB vs Poisson



NB vs Poisson



BayesPeak vs 3 other peak callers

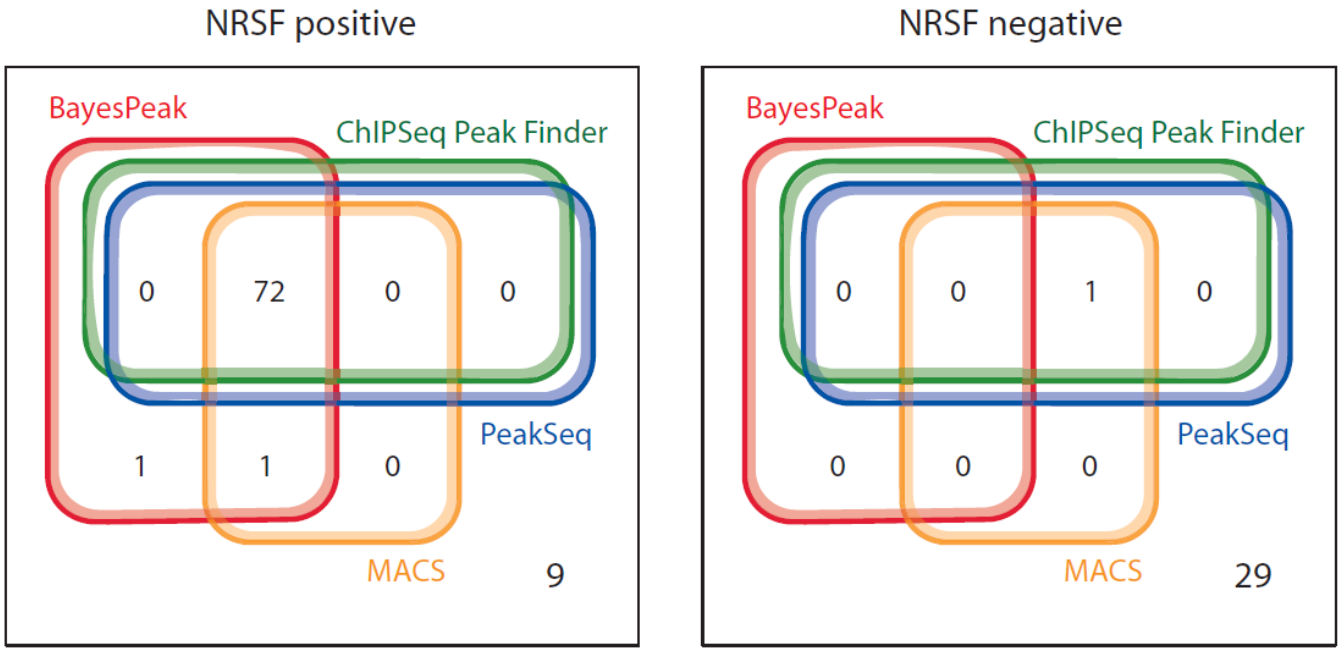


Figure 8
Algorithm comparison for NRSF data. This figure shows how the known-positive and known-negative regions called by the same four methods compare. We merged those results for CSPF and PeakSeq that were identical.

Algorithm

- `BayesPeak` fits a Markov model to the data (the aligned reads) via Markov Chain Monte Carlo (MCMC) techniques.
- The genome is firstly divided up into "jobs", i.e. short regions, by default of size 6 Mb, on which the algorithm is run independently. This allows us to account for the variation in read abundance across each chromosome.
- Within a job, we divide the region into small bins (by default, 100 bases each), and we consider the number of reads whose starts lie within each bin, for each strand.

Algorithm

- A hidden Markov model is fitted to the bins, thereby classifying them as enriched or unenriched for sites of interest. However, since the parameters of the model are unknown (for example, the mean number of counts within enriched or unenriched bins), we estimate them by sampling from their posterior distributions using MCMC methods.
- The output of the algorithm is the Posterior Probability (often abbreviated to *PP*) of each bin being enriched. The *PP* value is useful not only for calling the peaks, but could also be used in downstream analyses - for example, to weight observations when searching for a novel transcription factor motif. The *PP* value is not to be confused with the *p* value from hypothesis testing.

Example from vignette

- The example data set used below, consisting of the files "H3K4me3-chr16.bed" and "Input-chr16.bed", can be downloaded from `http://www.compbio.group.cam.ac.uk/Resources/BayesPeak/csbayespeak.html`. These data were generated from a histone mark ChIP-seq experiment, performed on mouse chromosome 16.
- The following code is a very simple example of a `BayesPeak` workflow, where we analyse the region 92,000,000 - 95,000,000 bp on chromosome 16. It should take a couple of minutes on a relatively modern machine.

```
> library(BayesPeak)
```

Example from vignette

```
> raw.output <- bayespeak("H3K4me3-chr16.bed",
+     "Input-chr16.bed", chr = "chr16",
+     start = 9.2e+07, end = 9.5e+07,
+     job.size = 6e+06)
> output <- summarize.peaks(raw.output,
+     method = "lowerbound")

> class(raw.output)

[1] "list"

> names(raw.output)

[1] "peaks"     "QC"          "p.samples"
[4] "call"

> head(output)
```

Example from vignette

```
RangedData with 6 rows and 1 value column across 1 spac
        space                 ranges |
  <character>              <IRanges> |
1       chr16 [92057850, 92059600] |
2       chr16 [92158500, 92159150] |
3       chr16 [92300850, 92302350] |
4       chr16 [92341500, 92341650] |
5       chr16 [92398900, 92399150] |
6       chr16 [92399500, 92400700] |
         PP
  <numeric>
1 1.0000000
2 1.0000000
3 1.0000000
```

Example from vignette

```
4 0.9999000
5 0.9999994
6 1.0000000
```

bayespeak

- Within each job, read abundance is modelled using non-overlapping bins. To avoid missing any peaks that straddle a boundary between two bins, we run the algorithm a second time. This second job is described as "offset" and shifts the boundaries of the bins by half a bin's length.
- The output of the function bayespeak is a list of four things:
  1. `raw.output$peaks`: Locations of "potentially enriched" bins, with their associated posterior probabilities. (A "potentially enriched" bin is defined as any bin with $PP > 0.01$.) Note that this is output is preliminary and does not correspond to the final result of the analysis.
  2. `raw.output$QC`: Information about the individual jobs.
  3. `raw.output$call`: A record of the arguments used when the function was called.

bayespeak

4. `raw.output$p.samples`: A list of parameter samples from the MCMC runs. This output can be used to assess convergence.

summarize.peaks

- The raw output of `bayespeak()` consists of the details of all of the individual bins in each individual job. The function `summarize.peaks()` combines the output of the individual jobs and joins adjacent bins into contiguous regions to give the final peak calls.
- In more detail, `summarize.peaks()` does the following:
  1. Filtering of unenriched jobs:
     The model naturally tries to identify enriched states in the background even for jobs with consistently low read abundance. This can result in many unreliable calls and, to avoid this, all calls associated with such jobs can be removed.
  2. Filtering of unenriched bins:
     We remove all bins whose *PP* values are below a certain value.

summarize.peaks

3. Assembly of enriched bins:
   The bins across all remaining jobs are collected together. If two jobs call exactly the same bin, which could happen in regions where jobs overlap, then the one with the larger *PP* value is used.
4. Conversion of bins to peaks:
   Where a number of bins form a contiguous region, we combine them into one large peak. The large peak is assigned a *PP* value, based on the *PP* values of its component bins. Combining the *PP* values from multiple bins may be done in several ways. By default, the "lowerbound" method is used, which calculates a lower bound for the overall *PP* by a dynamic programming technique. (See the help file for more information.) The alternative method that the package provides is the "max" method, which simply takes the maximum *PP* value in the component bins and is therefore

summarize.peaks

ill-equipped to provide appropriate credit for sustained regions of moderate enrichment.

Overfitting

- `BayesPeak` can run into an overfitting problem when a job does not contain any peaks, or when peaks are weak compared to the background.
- The model assumes that there are both unenriched and enriched regions present. When the data contains no enriched regions, the model still tries to identify peaks in the data. Since some bins in the background will have higher counts than others, purely by chance, these will be marked as enriched.

Overfitting

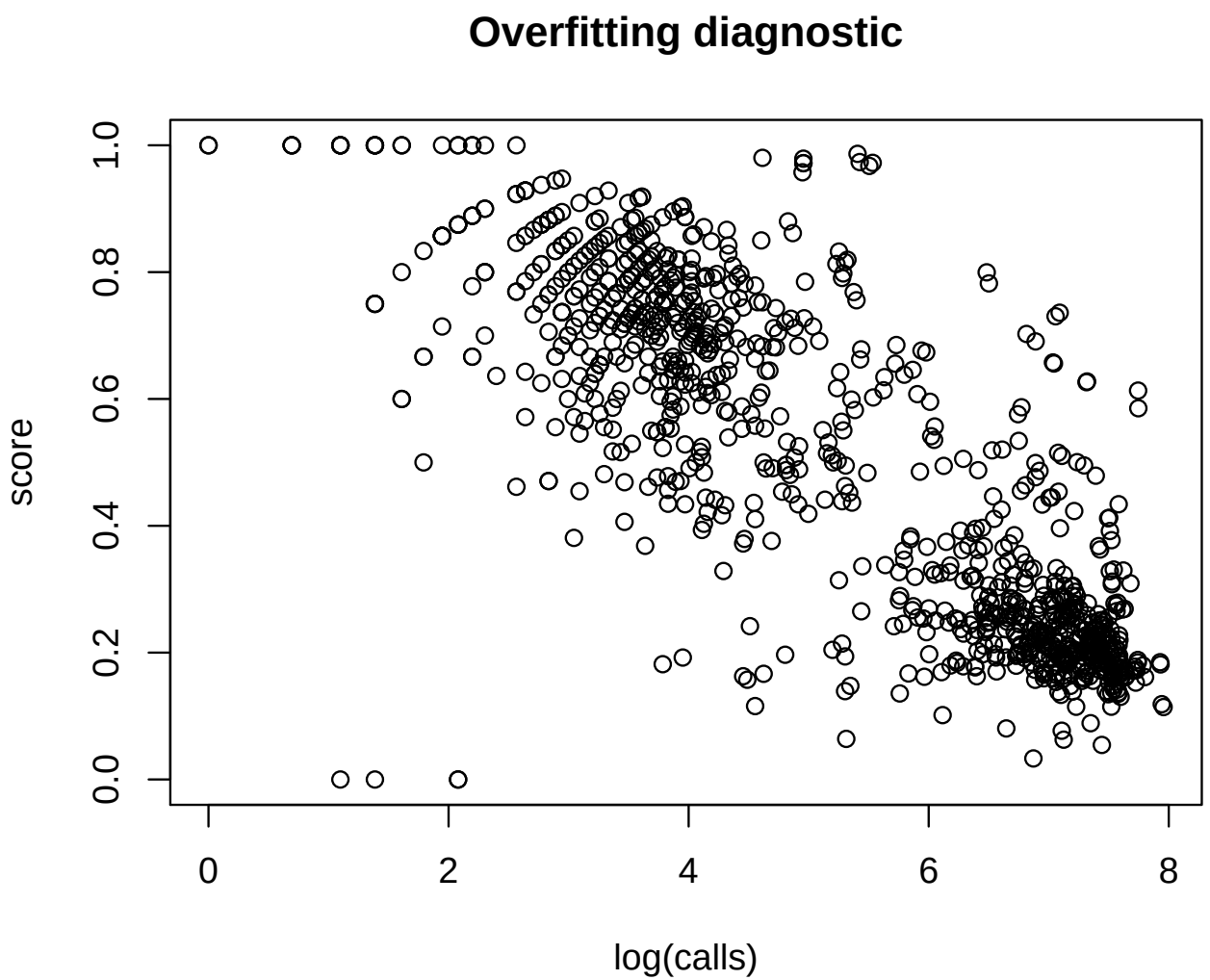
- This effect will be reflected in the parameters of the model, since the expected number of reads allocated at "enriched" regions will be much lower for the jobs where no peaks are present compared to the other jobs. This is one purpose of the QC component of the output - we can diagnose which peaks were called simply because they are in an unenriched job rather than because they are actual peaks.
- We'll explore this effect with the example data:

```
> data(raw.output)
```

OF diag 1

```
> plot.overfitdiag(raw.output, whatX = "calls",
+     whatY = "score", logX = TRUE,
+     logY = FALSE)
```

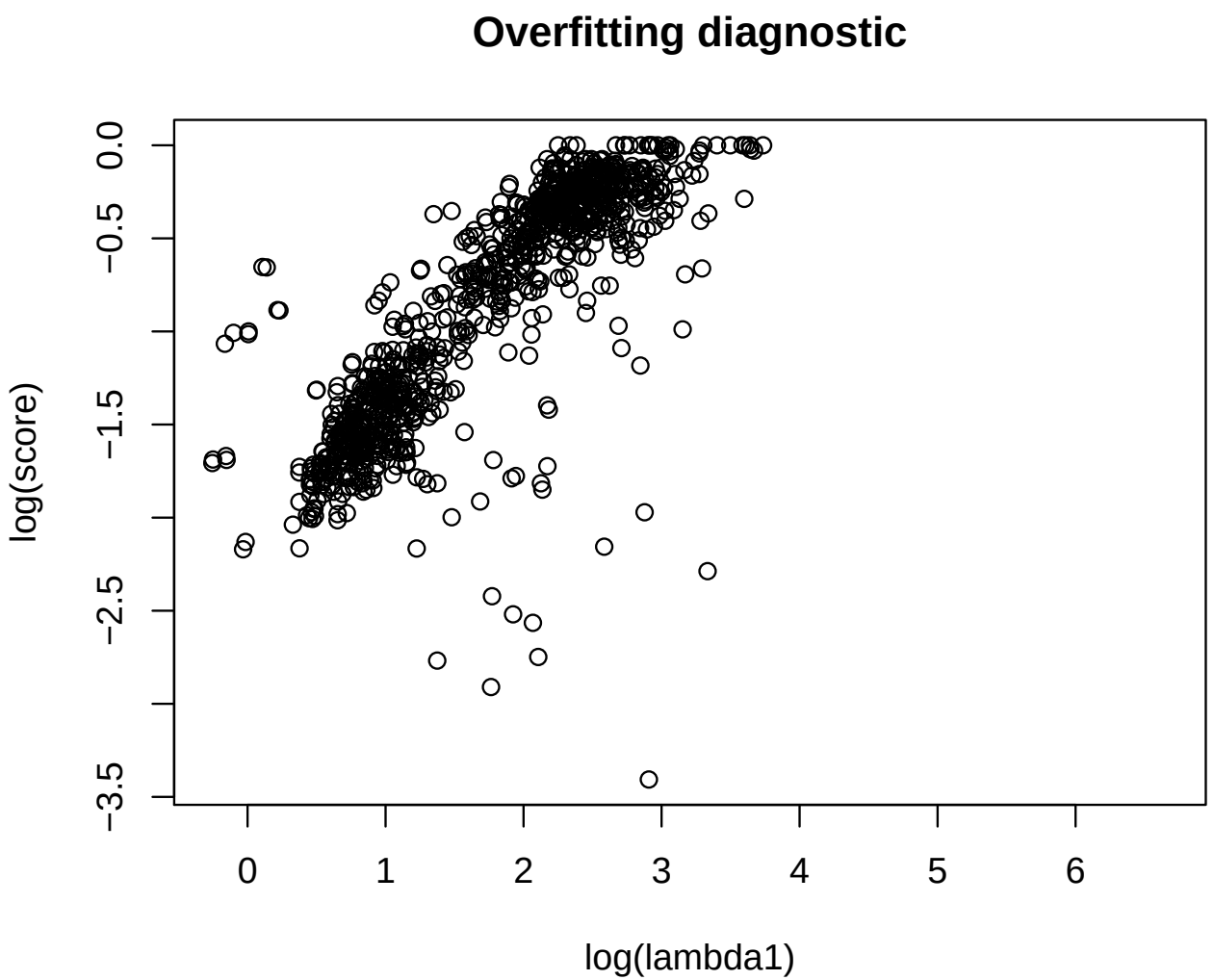
OF diag 1


Overfitting diagnostic
score
0.0
0.2
0.4
0.6
0.8
1.0
0
2
4
6
8
log(calls)

OF diag 2

```
> plot.overfitdiag(raw.output, whatX = "lambda1",
+     whatY = "score", logX = TRUE,
+     logY = TRUE)
```

OF diag 2


Overfitting diagnostic
log(score)
0.0
−0.5
−1.5
−2.5
−3.5
0
1
2
3
4
5
6
log(lambda1)

Filtering

- We can choose to simply remove all calls from the jobs that we believe to be overfit. For example, we can specify the overfit cluster by removing all jobs with low counts in their enriched bins - for example, $\log(\lambda_1) < 1.5$.
- Alternatively, we could try to specify the overfit cluster better by adding jobs with excessive numbers of calls e.g. $\log(\mathit{calls}) > 5$. This gives us two selection criteria as follows:

Filtering

```
> unreliable.jobs <- log(raw.output$QC$lambda1) <
+     1.5
> output <- summarize.peaks(raw.output,
+     method = "lowerbound", exclude.jobs = unreliable.
> unreliable.jobs2 <- log(raw.output$QC$lambda1) <
+     1.5 | log(raw.output$QC$calls) >
+     5
> output.2 <- summarize.peaks(raw.output,
+     method = "lowerbound", exclude.jobs = unreliable.
> nrow(output)

[1] 750

> nrow(output.2)

[1] 372
```

A manual method

- This method allows us to filter out peaks by manually selecting the area we consider to be overfitted.

```
> unreliable.jobs3 <- region.overfitdiag(raw.output,
+      whatX = "lambda1", whatY = "score",
+      logX = TRUE, logY = TRUE)
> output.3 <- summarize.peaks(raw.output,
+      method = "lowerbound", exclude.jobs = unreliable.
```

OF diag with polygon

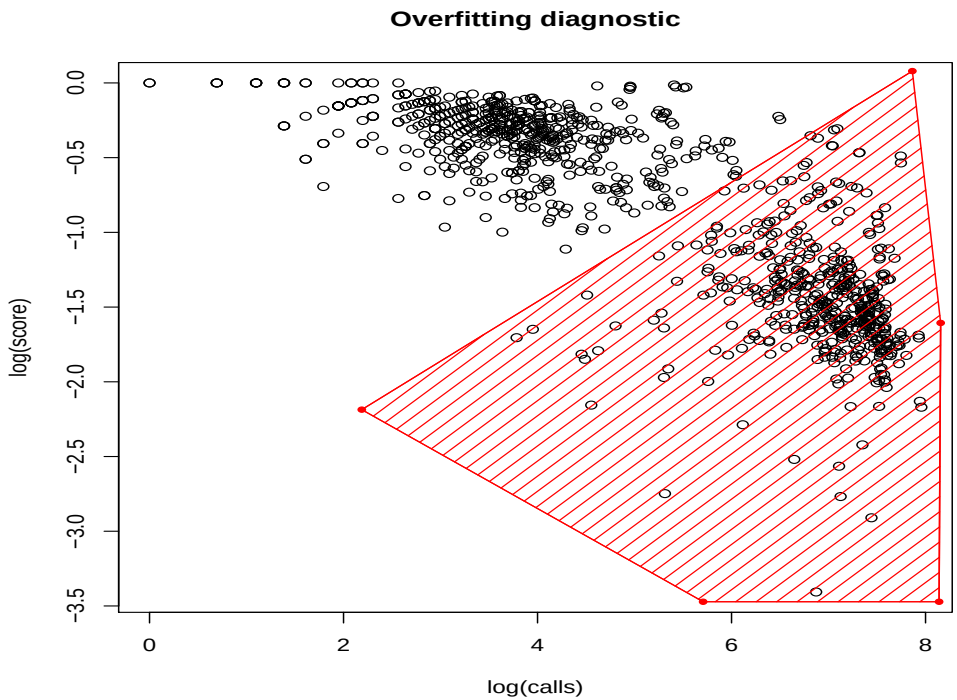


Figure 1: Application of `region.overfitdiag()` to the `raw.output` data. Jobs in the red hatched region will be excluded from the summary.

Session Information

```
> sessionInfo()

R version 2.12.0 (2010-10-15)
Platform: i386-pc-mingw32/i386 (32-bit)

locale:
[1] LC_COLLATE=English_United States.1252
[2] LC_CTYPE=English_United States.1252
[3] LC_MONETARY=English_United States.1252
[4] LC_NUMERIC=C
[5] LC_TIME=English_United States.1252

attached base packages:
[1] stats     graphics  grDevices
[4] utils     datasets  methods
[7] base

other attached packages:
[1] BayesPeak_1.2.2 IRanges_1.8.0
```

Session Information

```
loaded via a namespace (and not attached):
[1] tools_2.12.0
```