

Working with ChIP-Seq Data in R/Bioconductor

Tom Carroll and Ines de Santiago

July 23, 2015

Contents

1	Introduction	1
2	Working with aligned data	1
2.1	Reading in data	1
3	Calling peaks with MACS2 in the command line	2
4	A simple example using ChIPQsample()	3
4.1	QCmetrics summary	3
4.2	Cross-coverage and the FragmentLength/Relative cross-coverage scores (FragCC/RelCC)	4
4.3	Distribution of Signal: Within peaks, blacklists and known annotation	5
4.4	Distribution of Signal: Distribution of coverage depth across the genome	8
4.5	Conclusion	9
5	Assessing a ChIP-seq experiment using ChIPQC()	9
5.1	An example ChIP-seq experiment using ChIPQC()	9
5.2	Examining Cross-coverage and FragCC/RelCC scores across an experiment	11
5.3	Distribution of Signal across a ChIPQC experiment	13
5.4	Assessing sample similarity with Diffbind	20
5.5	Generating a summary HTML report	21
5.6	Conclusion	21
6	Advanced Topics	21
6.1	Providing additional data to ChIPQC plotting and reporting	21
6.2	Some plotting tips	24
7	References	27
8	Session Info	28

1 Introduction

This practical aims to introduce to analysing ChIP-seq data in R. This will include loading aligned reads and peak call data into the R environment, performing various data analyses and visualisations and assessing ChIP-seq data quality using the ChIPQC package with real world datasets. R provides support for various sequencing data formats. Here we will work with aligned reads in indexed BAM files and BED files containing peaks called by MACS.

First we would need to load the libraries required:

```
library(GenomicAlignments)
library(ChIPQC)
```

2 Working with aligned data

2.1 Reading in data

All files needed within this library are contained within the directory `"/data_for_practical/"` including BAM files (`.bam`) and BAM index files (`.bam.bai`).

Firstly we create a *BamFileList* object for use within the R environment. Many packages assume the naming convention for BAM indices to be `.bam.bai` although different names may be explicitly passed to these functions.

Use Case: Create a *BamFileList* object containing details of the BAM files we wish to analyse

```
dataDir <- "/data_for_practical/"
bamFiles <- dir(file.path(getwd(), dataDir), pattern="*.bam$", full.name=T)
names(bamFiles) <- gsub(".bam", "", basename(bamFiles))
bamFiles
bfList <- BamFileList(bamFiles)
bfList
```

Now that we have all the BAM files in a convenient list we can look at the information within the headers using `scanBamHeader()` function. This requires a path to the BAM file which we can get using the `path()` function.

Use Case: Extract the sam header for a single bam file and examine the structure of the list using the `str()` function

```
path(bfList)
samHeader <- scanBamHeader(path(bfList["TF_1"]))
str(samHeader, max.level=2)
```

We get a list containing two components: The first is 'targets' which contains a list of chromosomes used in the alignment. The second is 'text' which contains information including the species and alignment method used.

Use Case: Explore the information provided in the sam header. Find how the data has been sorted, the aligner used and what species the data has been aligned to.

```
samHeader[[1]]$targets
samHeader[[1]]$text
samHeader[[1]]$text["@HD"]
samHeader[[1]]$text["@PG"]
samHeader[[1]]$text["@CO"]
```

BAM files usually contain a lot of information and it is often more feasible to deal with a single chromosome at a time for processing and analysis.

Here we will set up parameters in order to select only reads aligning to Chromosome 1

Use Case: Extract information about chromosome 1 coordinates to set up filtering parameters using the `ScanBamParam()` function.

```
chr1dat <- samHeader[[1]]$targets["chr1"]
chr1dat
chr1range <- GRanges(seqnames=names(chr1dat), ranges=IRanges(1, chr1dat))
param <- ScanBamParam(which=chr1range)
param
```

Note that in this example, we use the default `ScanBamParam` 'flag' argument. However we could use this to build further filters (e.g. to remove duplicates or other potential artifacts) using the `scanBamFlag()` function. Look at the options for this function for more details using `?scanBamFlag`

Use Case: Having selected an area of interest, read in read data from a single BAM file using the parameters set using the `readGAlignments()` function. Convert this to a *GenomicRanges* object and inspect the data contained within. Calculate the average read length of the selected reads (in case some reads were trimmed).

```
alignDat <- readGAlignments(path(bfList["TF_1"]),param=param)
alignGR <- granges(alignDat)
seqlevels(alignGR) <- "chr1"
median(width(alignGR))
```

3 Calling peaks with MACS2 in the command line

MACS takes mapped BAM files of ChIP-seq and control samples and calls peaks. To call peaks, we will use the main module in MACS2 called 'callpeak'. It can be invoked by 'macs2 callpeak' command. If you type this command without parameters, you will see a full description of commandline options. Here we only list commonly used ones, for more detail read MACS2 documentation <https://github.com/taoliu/MACS/>.

Use Case: Run MACS2 to call peaks. Use TF_1.bam and the Input.bam files located in the '/data_for_practical/' directory.

```
macs2 callpeak -t TF_1.bam -c Input.bam
```

We used the following options:

```
-t: This is the only required parameter for MACS, refers to the name of the file with the ChIP-seq data
-c: The control or mock data file
```

MAC2 creates 3 files (peaks, summits, excel) and a directory containing wig or bedGraph. The peak file is named TF_1.peaks.bed

4 A simple example using ChIPQCsample()

ChIPQC package allows for the rapid generation of ChIP-seq quality metrics from aligned data in BAM format. The main function `ChIPQCsample()` can be run with simply a BAM file location and will return a `ChIPQCsample` object.

Use Case: Run `ChIPQCsample` on a single Bam file. As an example, we will use the TF_1.bam file located in the '/data_for_practical/' directory.

The `chromosomes` parameter specifies which chromosomes to examine when computing quality metrics. Restricting the chromosomes analysed can greatly increase the speed of the computations, as well as reducing the size of the resulting `ChIPQCexperiment` object. For this example we will limit the analysis to chromosome 1.

```
library(ChIPQC)
bamFile <- file.path(getwd(), "/data_for_practical/TF_1.bam")
exampleExp = ChIPQCsample(bamFile,peaks=NULL,annotation=NULL,chromosomes="chr1")

QCmetrics(exampleExp)
##      Reads      Map%      Filt%      Dup%      ReadL      FragL      RelCC      SSD
## 4.98e+05 1.00e+02 1.56e+01 4.35e+00 3.60e+01 1.55e+02 2.80e+00 1.59e+00
##      RiP%
##      NA
```

QCmetrics shows a summary of the main QCmetrics. These include the total number of reads in the bam file for each sample.

Use Case: To take full advantage of ChIPQC features, run ChIPQCsample on a single Bam file including additional information on blacklisted regions, genome annotation and any peaks

```
peaksFile <- file.path(getwd(), "/data_for_practical/TF_1_peaks.bed")
data(blacklist_hg19)
exampleExp = ChIPQCsample(bamFile, peaks=peaksFile, blacklist=blacklist_hg19,
                          annotation="hg19", chromosomes="chr1")
```

```
QCmetrics(exampleExp)
```

```
##      Reads      Map%      Filt%      Dup%      ReadL      FragL      RelCC      SSD
## 4.98e+05 1.00e+02 1.56e+01 4.35e+00 3.60e+01 1.55e+02 2.80e+00 1.59e+00
##      RiP%      RiBL%
## 1.30e+01 2.92e+00
```

Annotation is provided for human (hg19, hg18), mouse (mm10, mm9), rat (rn4), C Elegans (ce6) and D Melanogaster (dm3) by way of the Bioconductor TranscriptDb annotations packages. Additional or custom annotation can also be provided to the ChIPQCsample function and we go into this in detail later on.

Blacklist regions are included for hg19. You can access these regions by typing data(blacklist_hg19). Blacklists can also be provided as a GRanges object, or a character string that specifies complete file path to blacklists in bed format. Blacklists for a range of species is available from Anshul Kundaje's google site (<https://sites.google.com/site/anshulkundaje/projects/blacklists>).

Peaks can also be provided to the ChIPQCsample function as a GRanges object or as complete file path to peaks file in bed format.

4.1 QCmetrics summary

Now the result of QCmetrics contains full metrics for ChIP-seq and the additional information on RiP% and RiBL%. This is what they mean:

Reads total number of reads in the bam file.

Map% Percentage of total reads that were successfully mapped (aligned).

Filt% Percentage of mapped reads passing MapQ filter, in this case having a mapping quality score less than or equal to 15 (mapQCth=15 by default)

Dup% Percentage of mapped reads marked as duplicates.

ReadL Mean read length (as integer) derived from the data.

FragL Predicted fragment length by cross-coverage method. The fragment length is estimated by methods implemented in the chipseq package by systematically shifting the reads on each strand towards each other until the minimum genome coverage is achieved.

RelCC The relative cross-coverage score. The RelativeCC metric is calculated by comparing the maximum cross coverage peak (at the shift size corresponding to the fragment length) to the cross coverage at a shift size corresponding to the read length, with higher scores (generally 1 or greater) indicating good enrichment.

SSD Standardised standard deviation, or SSD score, as implemented in htSeqTools. This is another indication of evidence of enrichment. It is computed by looking at the standard deviation of signal pile-up along the genome normalised to the total number of reads. An enriched sample typically has regions of significant pile-up so a higher SSD is more indicative of better enrichment. SSD scores are dependent on the degree of total genome wide signal pile-up and so are sensitive to regions of high signal found with Blacklisted regions as well as genuine ChIP enrichment.

RiP% Reads mapped to peaks. This is another good indication of how "enriched" the sample is, and can be considered a "signal-to-noise" measure of what proportion of the library consists of fragments from binding sites vs. background reads. RiP% values for ChIPs around 5% or higher generally reflect successful enrichment.

RiBL% Reads mapped to blacklists. The signal from blacklisted has been shown to contribute to confound peak callers and fragment length estimation as well as contribute to the read length peak in cross coverage and cross coverage read length peak [1].

Now we have our full ChIPQCsample object we can start to review and visualise the metrics generated.

4.2 Cross-coverage and the FragmentLength/Relative cross-coverage scores (FragCC/RelCC)

For transcription factors and narrow epigenetic marks, an accumulation of Watson and Crick reads around the binding site/mark may often be seen. The degree to which your ChIP-seq signal is arranged into the Watson and Crick read clusters around such sites has been previously exploited as a metric of ChIP efficiency. In ChIPQC we assess the reduction in total genome covered which occurs from shifting the Watson reads along the genome (from 5' to 3' of chromosome). This is performed by measuring measure total coverage after each every successive shift of 1bp. As the Watson reads overlap the Crick reads around peaks the total genome covered will be reduced. The total coverage after each successive shift is then converted to cross-coverage scores after each shift.

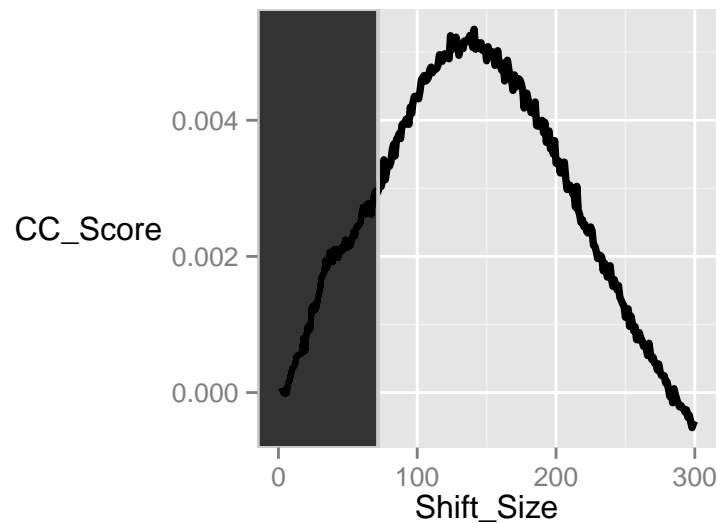
$$CrossCoverageScore_n = (TotalGenomeCoverage_0 - TotalGenomeCoverage_n) / TotalGenomeCoverage_0$$

Where n is the bp shift of Watson reads and 0 is after no shift of Watson reads.

The cross-coverage scores after successive shifts can then be visualised and reviewed to identify the expected increase in cross-coverage scores around the fragment length as well as any evidence of artefacts by a peak in the cross-coverage score at the read length.

Use Case: Use the `plotCC` function to calculate cross-coverage scores and plot those after successive shifts.

```
plotCC(exampleExp)
```



The area shaded in black (from 0 to $1.5 \times \text{readlength}$) is excluded when identifying the fragment length peak. This is due to the presence of the artefact peak which, in less enriched samples, may have greater cross-coverage score than observed at the fragment length. To avoid obscuring the selection of the fragment length peak for fragment length prediction and ascertainment of FragCC score and RelCC score this region is removed prior to calculation of shift with maximum cross-coverage score.

We can see that this ChIP has a peak in cross-coverage scores at 155bp, corresponding to the fragment size. This indicates that we have successfully enriched for signal around binding sites.

```
fragmentlength(exampleExp)
## [1] 155
```

Further to the visual inspection of the cross-coverage scores, we can extract RelCC and FragCC scores. These metrics can be considered to relate to efficiency of ChIP (FragCC) and efficiency of ChIP compared to artefact signal (RelCC).

$$FragCC = CrossCoverageScore_{max}$$

$$RelCC = CrossCoverageScore_{max} / CrossCoverageScore_{readlength}$$

Where *max* is shift with maximum cross-coverage score (excluding area 0 to 1.5*readlength) and *readlength* is the shift corresponding to the read length.

Use Case: Extract the RelCC and FragCC scores for this sample using `RelativeCrossCoverage` and `FragmentLengthCrossCoverage` functions respectively.

```
FragmentLengthCrossCoverage(exampleExp)
## [1] 0.00534

RelativeCrossCoverage(exampleExp)
## [1] 2.8
```

In this example we find, as expected from cross-coverage scores graph, that the FragCC is high and RelCC score is above 1 indicating a successful ChIP.

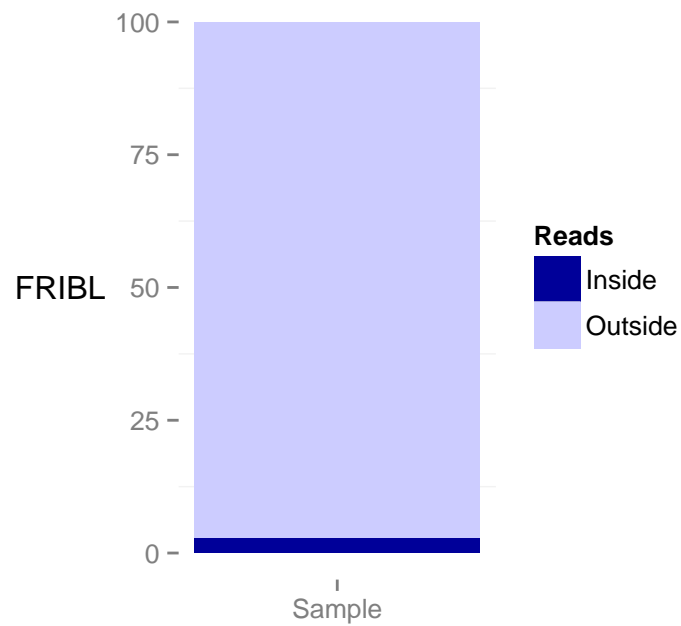
4.3 Distribution of Signal: Within peaks, blacklists and known annotation

Another useful set of characteristics of your ChIP-seq data is where in the genome the signal is distributed. This can be done by looking for the proportion or percentage of signal in peaks, blacklists or even in known annotation.

Use Case: To get the percentage of reads landing in peaks and blacklists we can use the `rip` and `ribl` functions as well as the `plotFrip` and `plotFribl` functions for visualisation.

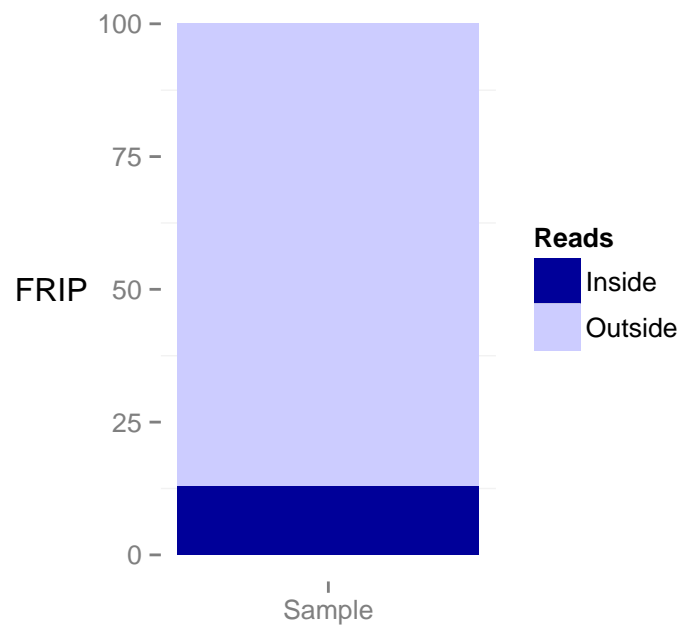
```
plotFribl(exampleExp)
```

Percentage Of Reads In Blacklists



```
plotFrip(exampleExp)
```

Percentage of Reads In Peaks



```
frip(exampleExp)
```

```
## [1] 0.13
```

The frip and fribl plot, show that we have a percentage of signal in peaks greater than 5% indicating a ChIP of acceptable quality. Reassuringly, we have a higher signal in peaks than blacklisted regions (2%) and so an enrichment over artefact signal.

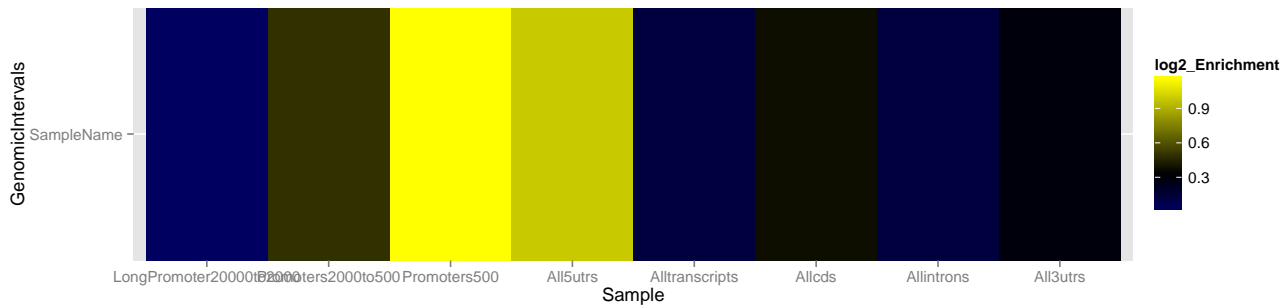
When provided with annotation in the form of genomic regions, ChIPQC can measure the enrichment of signal within them. The regi function provides a simple enrichment statistic which illustrates the distribution of signal within genomic interval annotation over that expected given their size.

regi = ProportionOfReadsInInterval / ProportionOfGenomeInInterval

Use Case: To review regi statistics, use the regi function or plot enrichment using plotRegi.

```
regi(exampleExp)
## LongPromoter20000to2000      Promoters2000to500      Promoters500
##                0.0274                0.4918                1.1518
##                All5utrs                Alltranscripts      Allcds
##                0.9837                0.1327                0.3780
##                Allintrons              All3utrs
##                0.1378                0.2939

plotRegi(exampleExp)
```



The regi scores and heatmap show an enrichment for regions around the TSS including 5'UTRs, and 500bp upstream regions. This suggests that the TF in question is a promoter associated transcription factor.

4.4 Distribution of Signal: Distribution of coverage depth across the genome

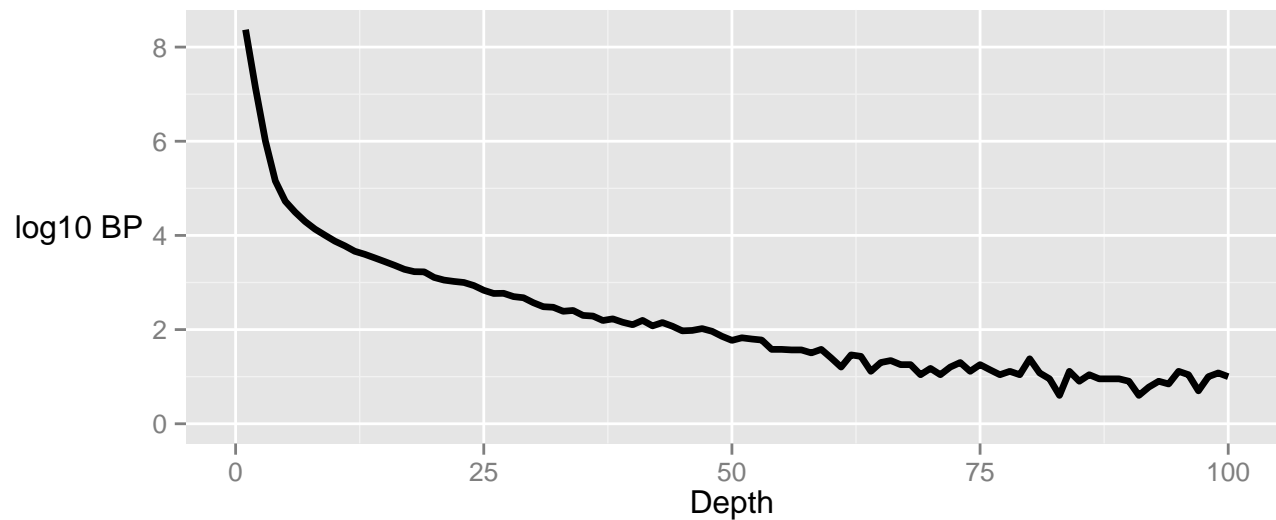
The final metric to review are those of the distribution of global pile-up across the genome. We can access this in two ways within ChIPQC, first by visualising the histogram of coverage depths and secondly by applying the SSD metric before and after removal of blacklisted regions.

First we draw the coverage histogram using plotCoverageHistogram. Note the cut-off at 100bp for visualisation purposes. If you want to replot the whole histogram you can extract the data using coveragehistogram function or extend x-axis as shown in Advanced Topics section.

```
coveragehistogram(exampleExp)[1:10]
##      0      1      2      3      4      5      6      7
## 2.34e+08 1.36e+07 1.02e+06 1.44e+05 5.32e+04 3.11e+04 1.97e+04 1.36e+04
##      8      9
## 1.01e+04 7.55e+03

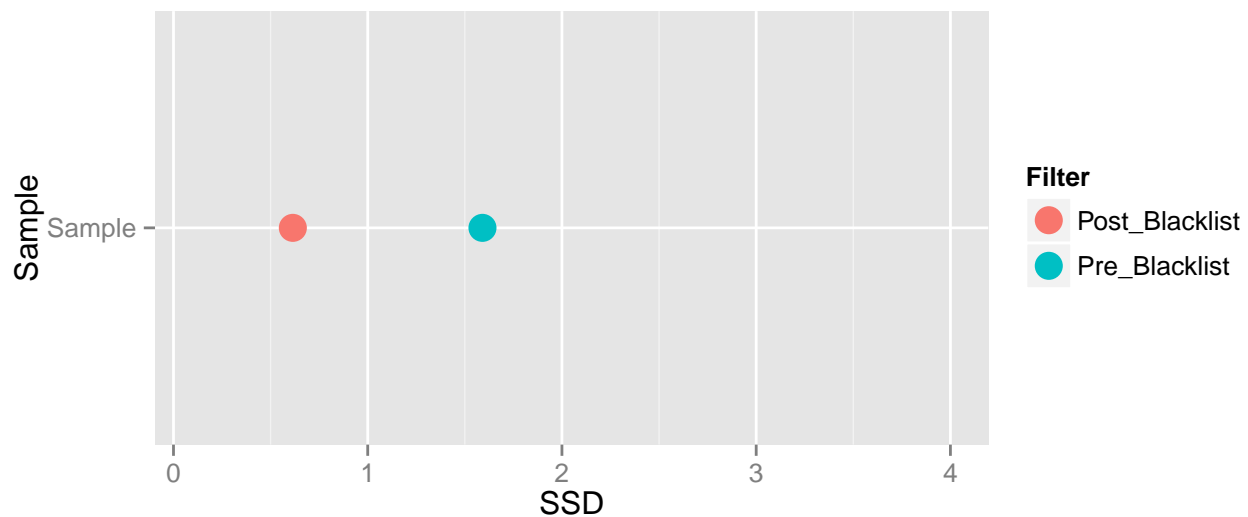
sum(coveragehistogram(exampleExp)[-c(1:19)])
## [1] 13693

plotCoverageHist(exampleExp)
```

The coverage histogram shows that there is a significant stretch of high signal pile-up. This may indicate significant signal associated to binding events but could also be from signal seen within blacklisted regions. To assess the contribution of artefact signal to global distribution of signal pileup we measure SSD before and after exclusion of signal from known blacklisted regions.

```
plotSSD(exampleExp)
```



Here we find that the SSD signal is not greatly affected by blacklisted regions and so, taken together with the coverage histogram, indicates the TF ChIP has a clear ChIP-signal above background.

4.5 Conclusion

From the combination of metrics used here we can establish that this ChIP showed an enrichment for structured ChIP-signal, had a signal pile-up above that seen within artefact regions and that this ChIP is positively associated with TSS regions.

5 Assessing a ChIP-seq experiment using ChIPQC()

When assessing ChIP-seq quality it is most useful to consider your sample quality alongside other ChIP and input samples. Taken together, a full experiment of ChIP and input samples allows for the identification of expected enrichment of sample metrics above input but also the variation of sample quality between biological replicates and the identification of bias within input/control samples.

The ChIPQC function wraps the functionality of ChIPQCsample to allow for the assessment of within experiment ChIP sample/input quality alongside user supplied experimental metadata.

5.1 An example ChIP-seq experiment using ChIPQC()

The ChIPQC function accepts a samplesheet of metadata and file locations alongside the same set of arguments as ChIPQCsample. An example of the layout for a sample sheet can be seen in R object below. The result of a call to ChIPQC is the ChIPQCexperiment object which contains the list of ChIPQCsample objects.

```
#resExperiment = ChIPQC(SampleSheet,annotation="mm9")
Rdata <- file.path(getwd(), "/data_for_practical/BCell_Examples.RData")
load(Rdata)
```

```
resExperiment
```

```
## Samples: 19 : DNase Ebf1 ... RNAPol2 RNAPol2ser2
##
##          Tissue      Factor Replicate Peaks
## DNase      Ch12      DNase           1 72437
## Ebf1       ProB      Ebf1           1  9841
## H3K4me3_IkNeg ProB    H3K4me3        1 18462
## H3K4me3_IkPos ProB    H3K4me3        2 20052
## H3K9ac_IkNeg  ProB    H3K9ac         1 27486
## H3K9ac_IkPos  ProB    H3K9ac         2 25902
## Ikaros_1_DPT  DPT     Ikaros         1 22253
## Ikaros_1_preproB preProB Ikaros         1 16240
## Ikaros_1_proB  ProB     Ikaros         1 15377
## Ikaros_2_preproB preProB Ikaros         2 16038
## Input_2_proB   ProB     Input          1    0
## Input_Ch12     Ch12     Input          1    0
## Irf            ProB     Irf            1 39628
## Mi2b_IKneg     ProB     Mi2b           1    0
## Mi2b_IKpos     ProB     Mi2b           2    0
## Myc            Ch12     Myc            1 44982
## Pu1            ProB     Pu1            1 48472
## RNAPol2        Ch12     RNAPol2         1 59266
## RNAPol2ser2    Ch12     RNAPol2_Ser2     1 30787
##
##          Reads Map% Filt% Dup% ReadL FragL RelCC SSD RiP% RiBL%
## DNase      1.10e+08 100 15.5 27.60 36 73 0.9280 3.22 45.40 7.94
## Ebf1       2.91e+07 100 24.3 6.60 36 177 2.6400 2.67 3.07 11.00
## H3K4me3_IkNeg 7.61e+07 100 15.1 27.00 36 228 3.9300 2.98 45.90 7.94
## H3K4me3_IkPos 1.29e+08 100 14.0 50.00 36 243 3.8200 3.61 50.70 7.40
## H3K9ac_IkNeg  6.46e+07 100 14.8 13.00 36 215 3.9300 2.28 36.30 6.73
## H3K9ac_IkPos  6.91e+07 100 13.9 14.70 36 213 4.1700 2.01 37.90 5.84
## Ikaros_1_DPT  4.83e+07 100 23.0 15.30 36 173 1.8000 3.19 5.31 10.50
## Ikaros_1_preproB 3.59e+07 100 24.7 15.30 36 175 0.2780 3.43 3.50 12.50
## Ikaros_1_proB  2.48e+07 100 23.2 13.10 36 165 0.9210 2.28 3.49 10.50
## Ikaros_2_preproB 3.55e+07 100 24.7 15.20 36 181 0.3110 3.42 3.43 12.60
```

```
## Input_2_proB      2.35e+07 100 22.1  9.73   36  120 1.6500 1.93   NA  9.02
## Input_Ch12       1.91e+07 100 25.5 21.50   36  157 0.0556 2.06   NA  9.81
## Irf              2.79e+07 100 21.4 20.80   36  153 2.3200 2.24 10.50  9.91
## Mi2b_IKneg       3.85e+07 100 23.5 29.30   36  151 0.4990 3.09   NA 11.40
## Mi2b_IKpos       3.48e+07 100 23.8  6.08   36  152 0.6380 3.02   NA 11.60
## Myc              3.93e+07 100 19.4 11.60   36  150 1.5700 2.29 16.80  8.74
## Pu1              3.41e+07 100 20.8 20.30   36  204 3.3300 2.52 22.20  8.89
## RNAPol2          3.35e+07 100 12.3 17.90   36  152 1.9200 1.11 45.70  4.98
## RNAPol2ser2      4.82e+07 100 15.0  6.63   36  125 0.8050 1.61 30.30  5.86
```

The QCmetric function now displays a table of metrics as seen for ChIPQCsample and similarly all accessors and plotting functions used for ChIPQCsample objects can be used with the ChIPQCexperiment object.

```
QCmetrics(resExperiment)
```

```
##           Reads Map% Filt% Dup% ReadL FragL RelCC SSD RiP% RiBL%
## DNase      1.10e+08 100 15.5 27.60   36   73 0.9280 3.22 45.40  7.94
## Ebf1       2.91e+07 100 24.3  6.60   36  177 2.6400 2.67  3.07 11.00
## H3K4me3_IkNeg 7.61e+07 100 15.1 27.00   36  228 3.9300 2.98 45.90  7.94
## H3K4me3_IkPos 1.29e+08 100 14.0 50.00   36  243 3.8200 3.61 50.70  7.40
## H3K9ac_IkNeg  6.46e+07 100 14.8 13.00   36  215 3.9300 2.28 36.30  6.73
## H3K9ac_IkPos  6.91e+07 100 13.9 14.70   36  213 4.1700 2.01 37.90  5.84
## Ikaros_1_DPT  4.83e+07 100 23.0 15.30   36  173 1.8000 3.19  5.31 10.50
## Ikaros_1_preproB 3.59e+07 100 24.7 15.30   36  175 0.2780 3.43  3.50 12.50
## Ikaros_1_proB  2.48e+07 100 23.2 13.10   36  165 0.9210 2.28  3.49 10.50
## Ikaros_2_preproB 3.55e+07 100 24.7 15.20   36  181 0.3110 3.42  3.43 12.60
## Input_2_proB  2.35e+07 100 22.1  9.73   36  120 1.6500 1.93   NA  9.02
## Input_Ch12    1.91e+07 100 25.5 21.50   36  157 0.0556 2.06   NA  9.81
## Irf          2.79e+07 100 21.4 20.80   36  153 2.3200 2.24 10.50  9.91
## Mi2b_IKneg    3.85e+07 100 23.5 29.30   36  151 0.4990 3.09   NA 11.40
## Mi2b_IKpos    3.48e+07 100 23.8  6.08   36  152 0.6380 3.02   NA 11.60
## Myc          3.93e+07 100 19.4 11.60   36  150 1.5700 2.29 16.80  8.74
## Pu1          3.41e+07 100 20.8 20.30   36  204 3.3300 2.52 22.20  8.89
## RNAPol2       3.35e+07 100 12.3 17.90   36  152 1.9200 1.11 45.70  4.98
## RNAPol2ser2   4.82e+07 100 15.0  6.63   36  125 0.8050 1.61 30.30  5.86
```

In addition to standard plotting routine, ChIPQCexperiment plots can be grouped by the metadata provided using the argument facetBy and for plotCoverageHistogram and plotCC methods the colours and line types controlled by colourBy and lineBy respectively. To group/colour/line type by metadata, a character vector of the metadata column title/s to use may be provided.

```
facetBy = c("Tissue","Factor","Condition")
colourBy = c("Treatment","Replicate")
lineBy = c("Replicate")
```

5.2 Examining Cross-coverage and FragCC/RelCC scores across an experiment

As with the ChIPQCsample object, the cross coverage scores for a group of samples can be plotted using plotCC(). By default samples will be groups by their Tissue and Factor combinations and coloured by their Replicate number. Here we group by Factor, colour by Tissue and set the line type by the Replicate number.

```
FragmentLengthCrossCoverage(resExperiment)
```

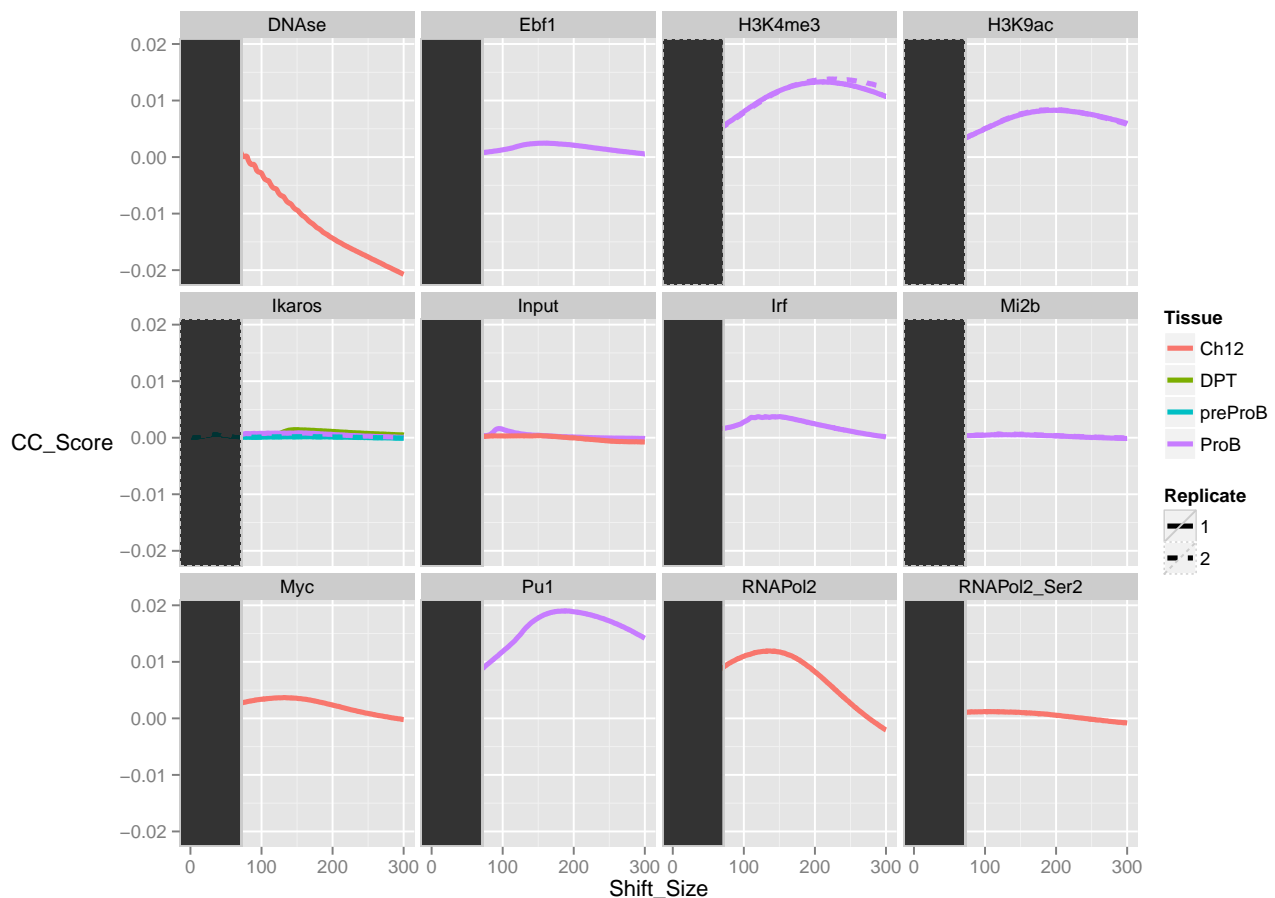
```
##           DNase           Ebf1      H3K4me3_IkNeg      H3K4me3_IkPos
##           0.000149       0.002461       0.013315       0.013781
##      H3K9ac_IkNeg      H3K9ac_IkPos      Ikaros_1_DPT Ikaros_1_preproB
```

```
##      0.008278      0.008374      0.001399      0.000167
##      Ikaros_1_proB Ikaros_2_preproB      Input_2_proB      Input_Ch12
##      0.000861      0.000191      0.001500      0.000331
##      Irf      Mi2b_IKneg      Mi2b_IKpos      Myc
##      0.003734      0.000533      0.000599      0.003622
##      Pu1      RNAPol2      RNAPol2ser2
##      0.018980      0.011852      0.001195
```

```
RelativeCrossCoverage(resExperiment)
```

```
##      DNase      Ebf1      H3K4me3_IkNeg      H3K4me3_IkPos
##      0.9277      2.6369      3.9340      3.8202
##      H3K9ac_IkNeg      H3K9ac_IkPos      Ikaros_1_DPT Ikaros_1_preproB
##      3.9301      4.1730      1.7986      0.2776
##      Ikaros_1_proB Ikaros_2_preproB      Input_2_proB      Input_Ch12
##      0.9207      0.3106      1.6458      0.0556
##      Irf      Mi2b_IKneg      Mi2b_IKpos      Myc
##      2.3151      0.4991      0.6378      1.5726
##      Pu1      RNAPol2      RNAPol2ser2
##      3.3293      1.9199      0.8046
```

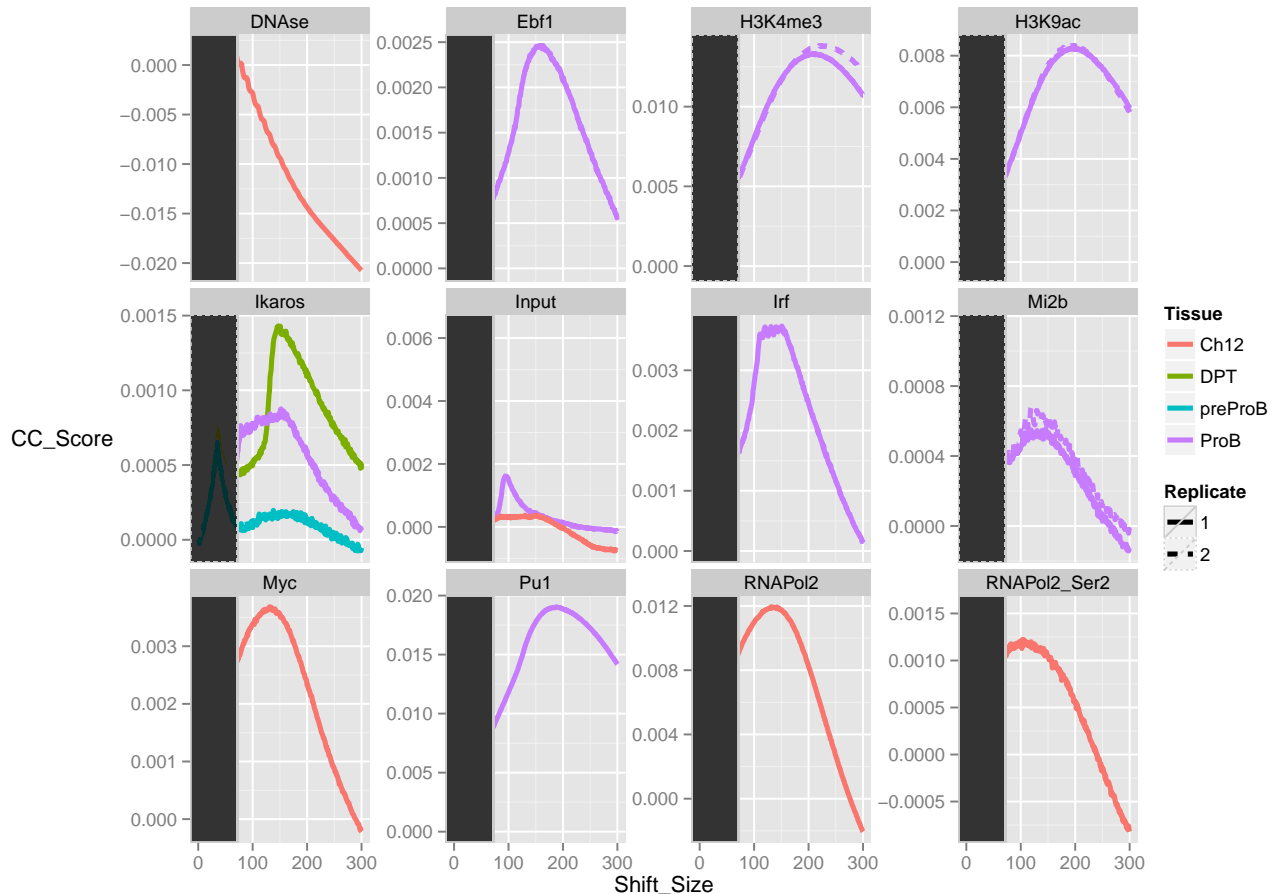
```
plotCC(resExperiment, colourBy="Tissue", facetBy="Factor", lineBy="Replicate")
```



From this, it is immediately apparent that some samples not only have much higher scores, and hence efficiency, than others but that their fragment lengths appear to be very different from each other. The DNase sample for example has a fragment length less than half of the Pu1 sample. Now we have established the difference in total efficiency, we can

look at the overall shape of cross-coverage scores and the relationship between signal peaks and artefact peaks in the cross-coverage scores. To help better visualise, we will apply a further facet wrap to the ggplot2 object returned by `plotCC` in order to compare within factors.

```
ccplot <- plotCC(resExperiment, colourBy="Tissue", facetBy="Factor", lineBy="Replicate")
ccplot + facet_wrap(~Factor, scales="free_y")
```



The free scaled cross-coverage score plots now reveal more about the distribution of signal within the samples. The Ebf1, Ikaros, Myc, Irf and RNA Pol2 all show tight peaks within their cross-coverage score profiles illustrating their sharp binding profiles as expected for a transcription factor (and RNA Pol2 around TSS/Enhancers). The histone marks and Pu1 however show longer more diffused peaks reflecting the wider breadth of signal seen for these epigenetic marks.

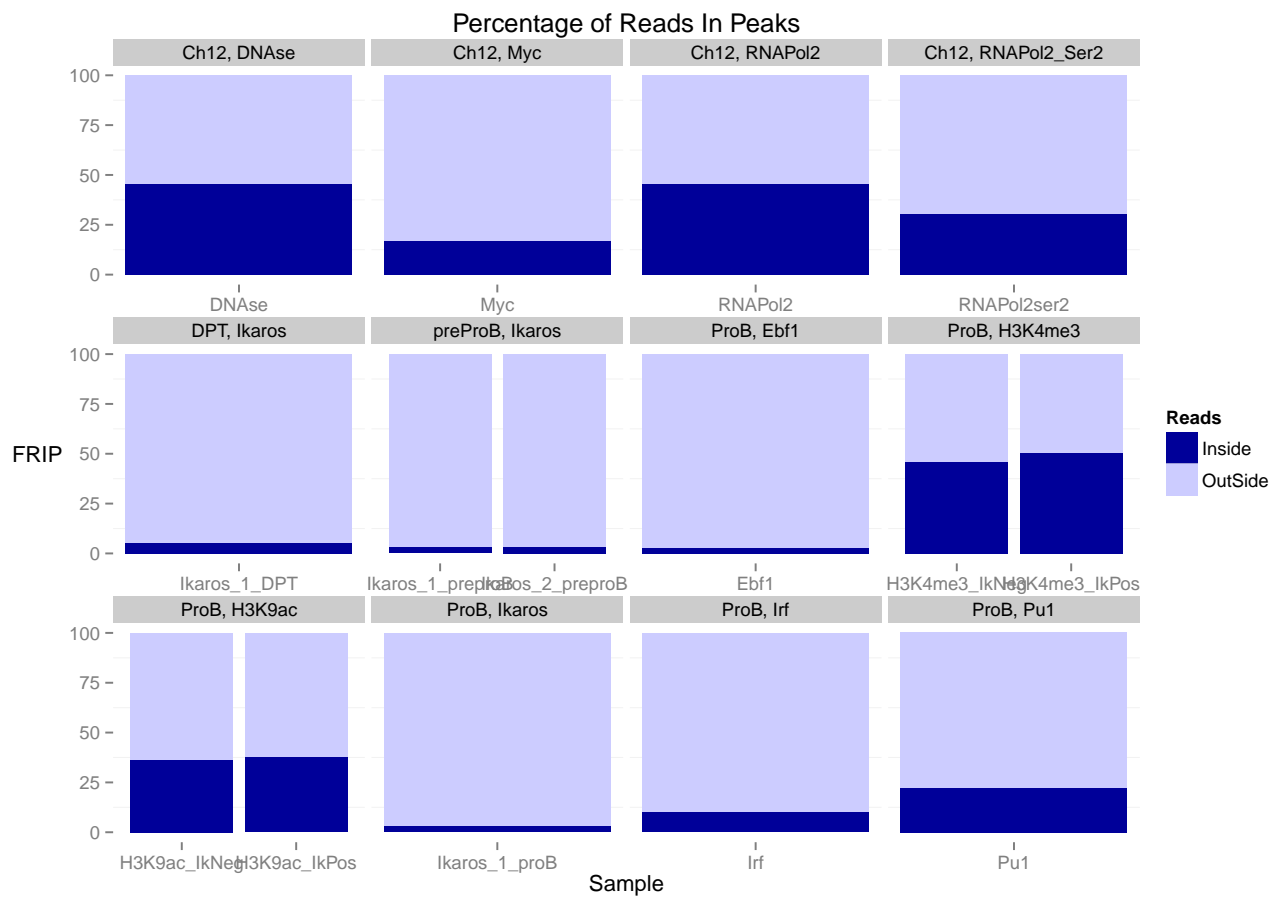
The signal of the Ikaros ChIP between cell lines can also be seen to be highly variable with DP thymocytes containing highest RelCC scores, ProB lower and preProB the lowest. This reflects the increased concentrations of Ikaros along haemopoietic differentiation with DP thymocytes having the highest Ikaros levels.

Finally, an enrichment for fragment length signal can be seen in the ProB input. The sharpness of this enrichment suggests a highly duplicated peak like signal within this track which would be cause for further investigation. This may be from the sono-seq effect commonly observed in input, where gel-selection of fragment lengths for input causes a small peak in cross coverage scores close to selected fragment length.

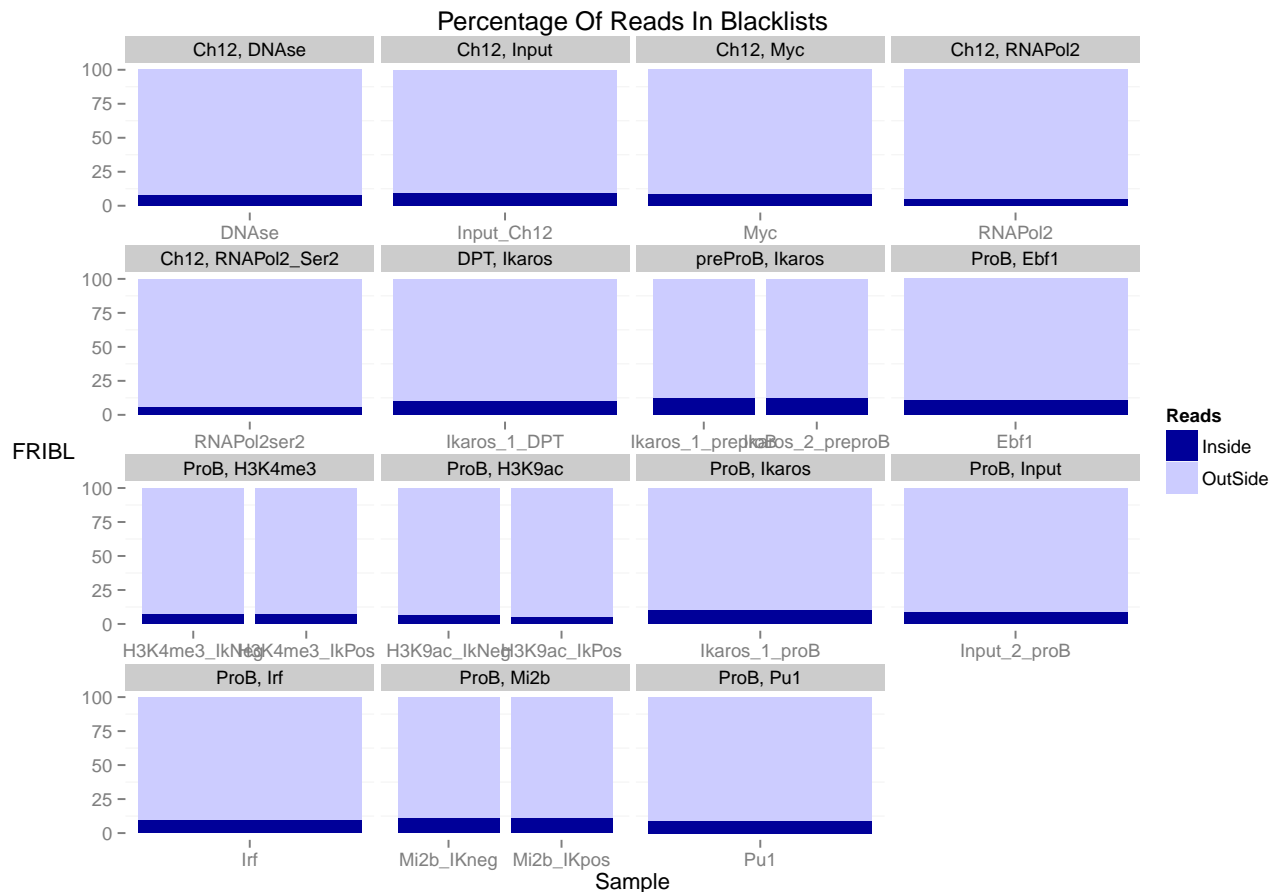
5.3 Distribution of Signal across a ChIPQC experiment

As with the `ChIPQCsample` object, the fraction of signal in peaks, blacklists and annotated genomic intervals can provide an understanding of the ChIPs' efficiency and pattern of enrichment.

```
plotFriP(resExperiment)
## Using Sample as id variables
```



```
plotFribl(resExperiment)
## Using Sample as id variables
```

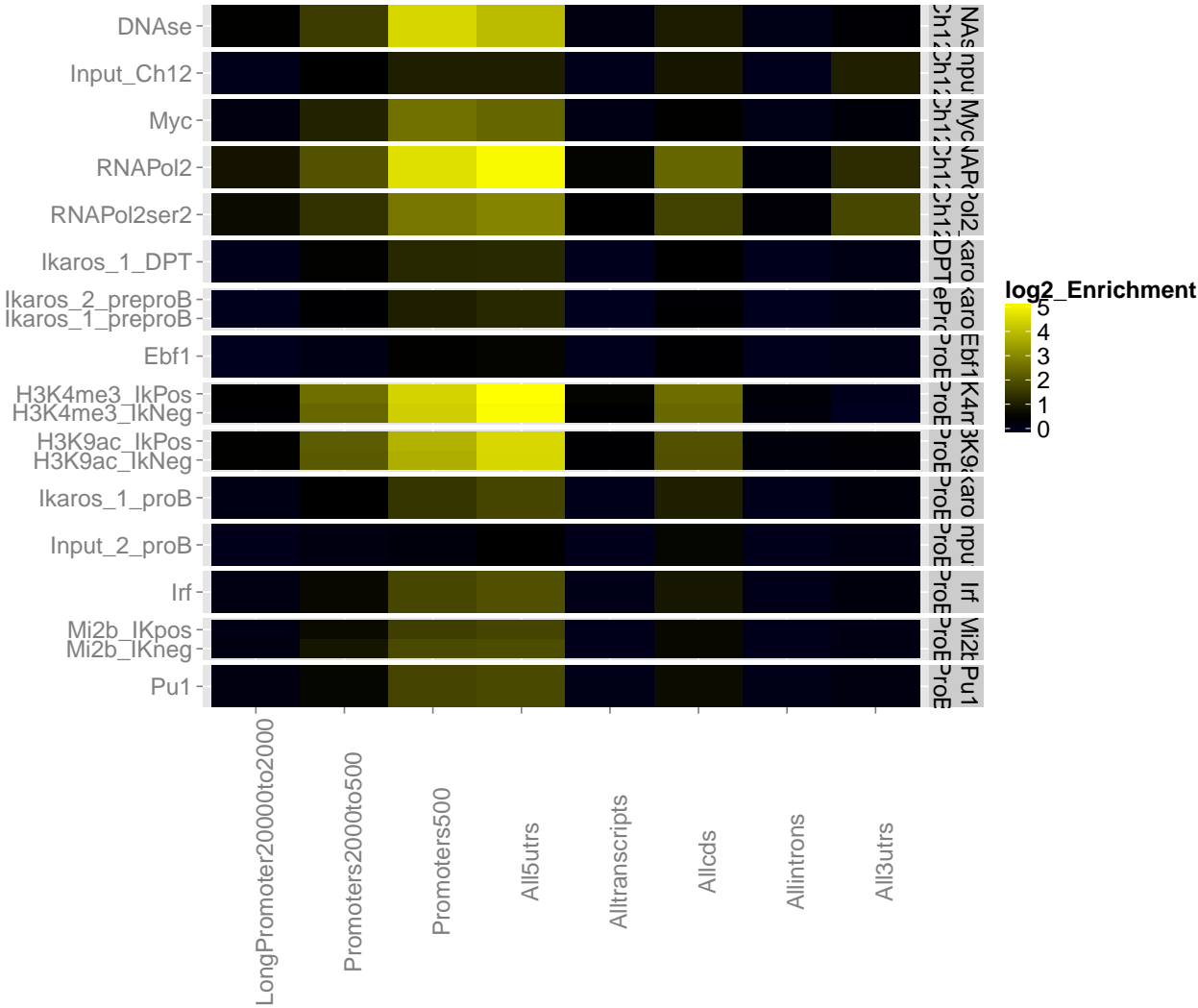


The output from `plotFrip` immediately identifies the Histone, RNA Pol2 and DNase ChIP as having the highest enrichment (25% to 50%) for reads in peaks as expected for such epigenetic marks. Also apparent are the significant enrichment seen within Pu1, Myc and Irf whereas the Ikaros ChIPs all show considerably lower enrichment for signal within peaks.

The Fribl plot here shows that all samples have equivalent levels of signal so no outlier or need to investigate signal within known blacklisted regions.

Samples such as RNA Pol2, DNase and Histone marks will have an expected enrichment for genomic locations. Here RNA Pol2 should be expected to have a stronger enrichment at the TSS than RNA Pol2ser2 whereas RNA Pol2ser2 should show enrichment within the 3'UTRs regions.

```
plotRegi(resExperiment)
```



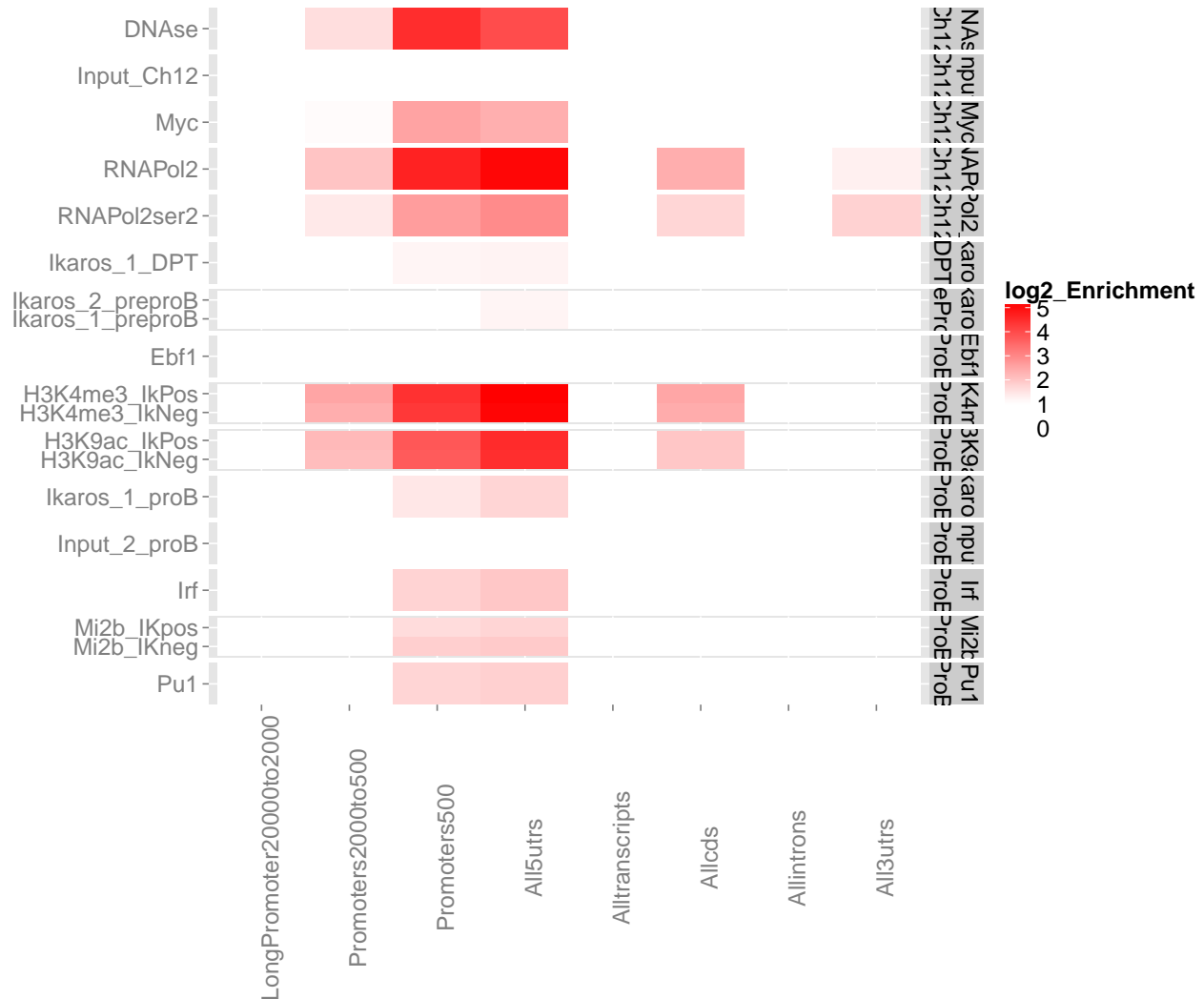
```
regi(resExperiment) ["All3utrs",]  
##          DNase          Ebf1      H3K4me3_IkNeg      H3K4me3_IkPos  
##          0.4215          0.0855          -0.0823          0.0188  
##      H3K9ac_IkNeg      H3K9ac_IkPos      Ikaros_1_DPT      Ikaros_1_preproB  
##          0.3931          0.4044          0.1095          0.1080  
##      Ikaros_1_proB      Ikaros_2_preproB      Input_2_proB      Input_Ch12  
##          0.2967          0.1083          0.1527          1.0896  
##          Irf      Mi2b_IKneg      Mi2b_IKpos      Myc  
##          0.2840          0.1622          0.1627          0.3285  
##          Pu1      RNAPol2      RNAPol2ser2  
##          0.2329          1.3167          1.8052
```

From the Regi plot it can be seen that all histone marks, DNA and RNAPII show the expected enrichment across gene regions. Combined with the output from regi function, the RNA Pol2Ser2 has the greatest enrichment at 3'UTRs and so the expected pattern of enrichment.

To better visualise this enrichment we can adjust the scale to the enrichment seen in Ch12 input for 3'UTRs.

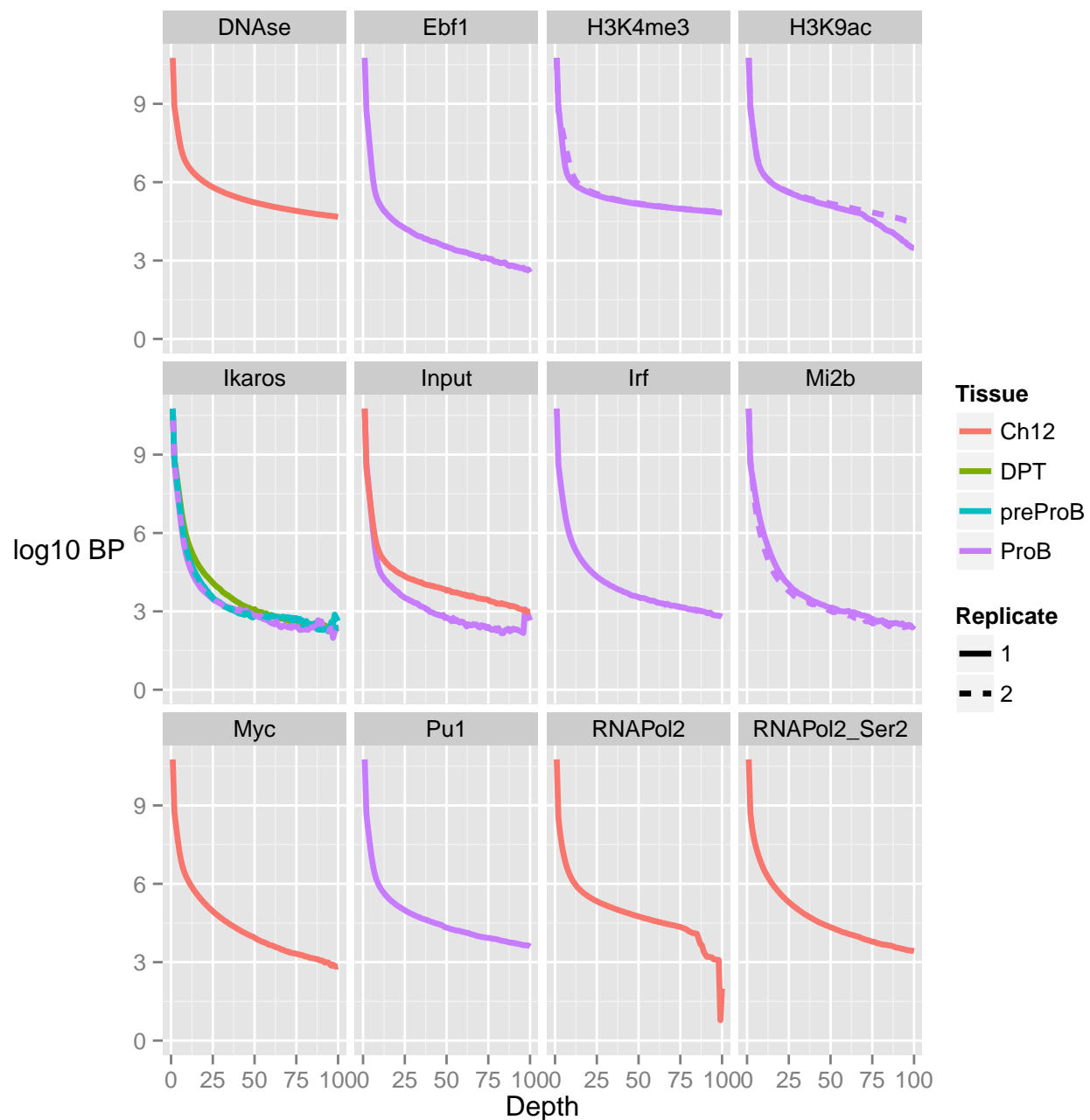

```
plotRegi(resExperiment)+scale_fill_gradient2(low="white",high="red",
  mid="white",midpoint=regi(resExperiment)["All3utrs","Input_Ch12"])
```

Scale for 'fill' is already present. Adding another scale for 'fill', which will replace the existing scale.



Both inputs showed a small but comparatively low enrichment for reads in genic regions. Such enrichment around gene regions can typically be seen for input samples due to the increased accessibility of chromatin to fragmentation around TSSs. As with the ChIPQCsample object we can plot the coverage histogram and SSD before and after blacklisting by using the plotCoverageHist and plotSSD functions.

```
plotCoverageHist(resExperiment,facetBy="Factor",colourBy="Tissue",lineBy="Replicate")
```



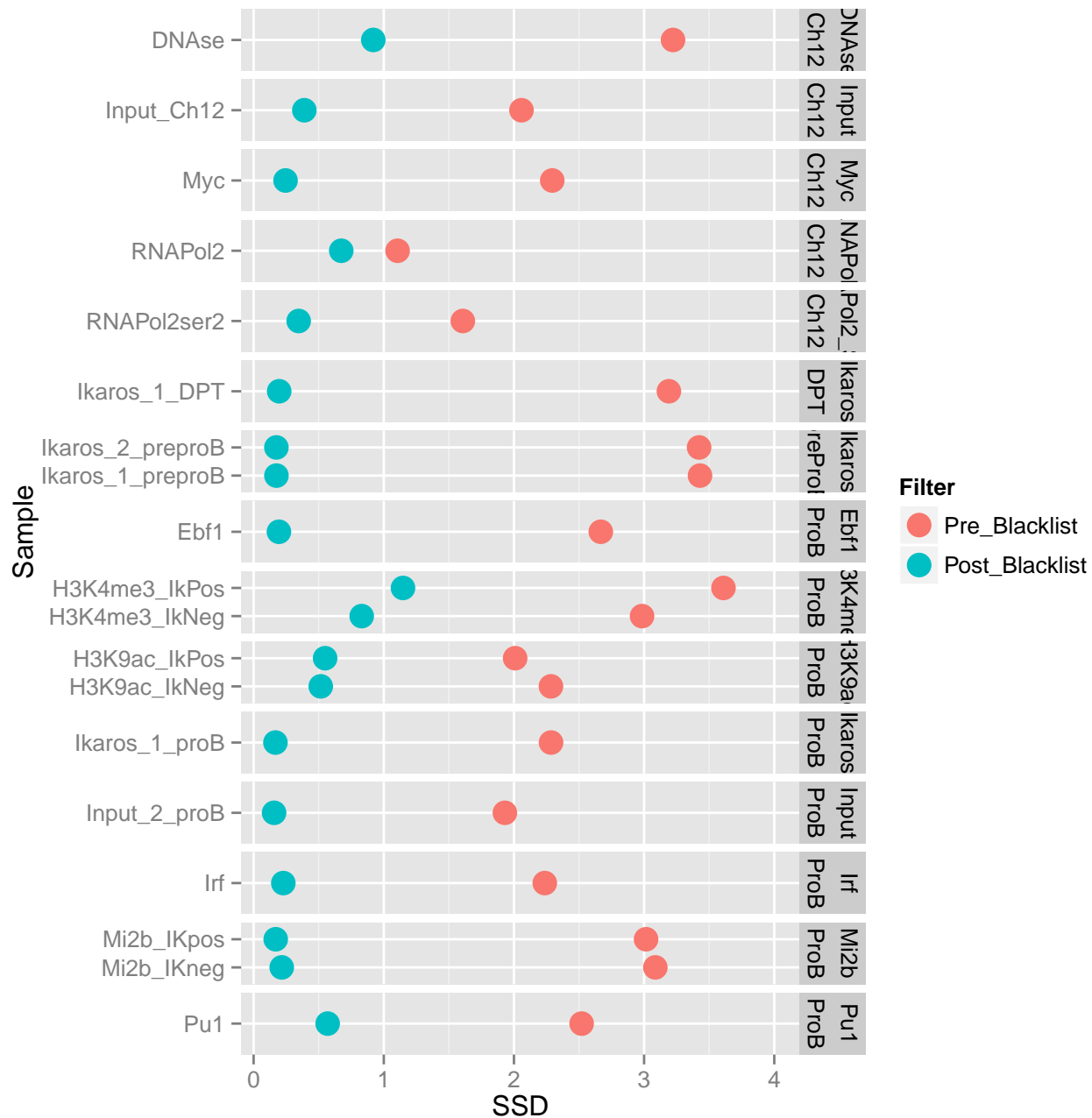
The coverage histogram shows the expected greater spans of high signal in Histone, RNA Pol2 and DNase ChIPs as well as for transcription factors with both high Rip% and RelCC scores. Although most ProB transcription factors have greater spans of high depth than their input, enrichment for these transcription factors can be seen to be much smaller than observed for Histone, RNA Pol2 and DNase ChIPs.

The Ikaros ChIPs universally had low enrichment and as seen with other metrics DP thymocytes had the greatest signal.

The two inputs samples show very different patterns of signal depth. The Ch12 input shows considerable span of high signal whereas the ProB shows a spike in signal at above 98 reads high, in keeping with observed high duplication rate for this sample and fragment length peak in cross-coverage scores.

```
plotSSD(resExperiment)
```

```
## Using Sample as id variables
```



The plot of SSD scores before and after blacklisting show that the effect of blacklisting on SSD scores is dramatic.

As with the coverage histogram plots, the histone, polymerase, DNase marks as well as high scoring TFs have greatest SSDs after blacklisting.

The SSD for Ikaros after blacklisting can be seen to be just above background with again DP thymocytes having the greatest score.

For the inputs, the ProB input can be seen to have its SSD score reduced to a background level of around 0.14 indicating the successful removal of artefact signal. The Ch12 input however can be seen to not drop to the background level after blacklisting suggesting remaining regions of artefact signal. The failure to reduce SSD after blacklisting suggests

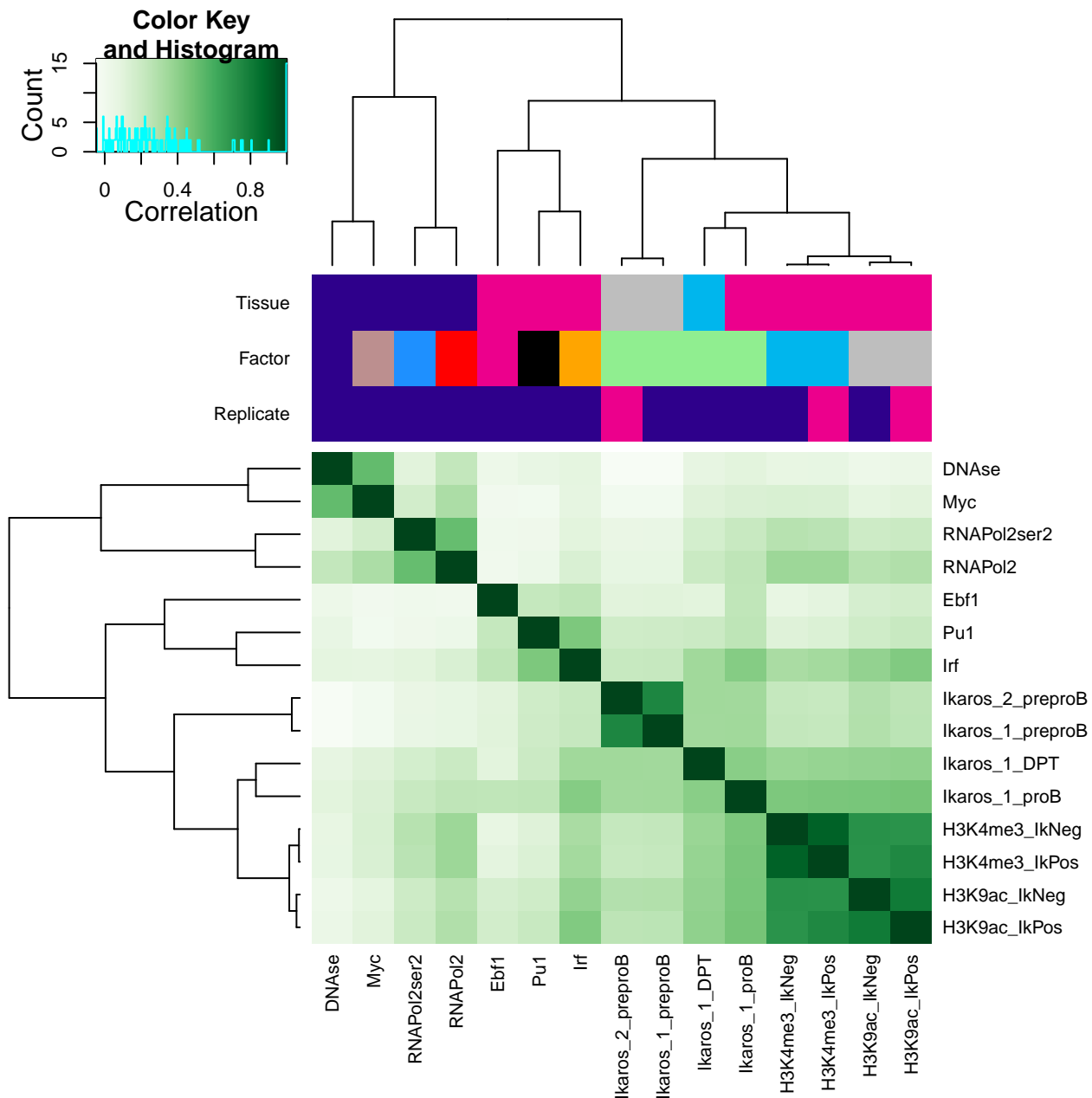
the persistent presence of artefacts within the Ch12 input and flags this sample for further blacklisting.

5.4 Assessing sample similarity with Diffbind

A final set of metrics useful for ChIP-quality relate to the correlation between binding events across samples within a ChIPQCexperiment.

The Diffbind package allows for the clustering of samples based on the co-occurrence of peaks within samples. When analysing an experiment ChIPQC will perform a sample clustering by default as well as construct the Diffbind DBA object. To produce the a sample heatmap, the plotCorHeatmap function can be used.

```
plotCorHeatmap(resExperiment)
```



The clustering and heatmaps generated from `plotCorHeatmap` allow us to identify which samples are most closely related and so both the reproducibility of replicates and the similarity in binding profiles of different epigenetic marks. Here we see that all replicates cluster tightly together, illustrating the relative reproducibility of our replicates. Further to this we see that the samples broadly group into their respective tissues with DNase, RNA Pol2 and Myc forming the Ch12 cluster. The ProB transcription factor (Pu1, Irf and Ebf) are seen to group together as expected and are found to cluster away from ProB histone marks. This suggests that these transcription factors may be less associated to these histone marks than the Ikaros ChIPs.

5.5 Generating a summary HTML report

A summary HTML report can be generated for this example experiment by invoking `ChIPQCReport`

```
ChIPQCReport(resExperiment, colourBy="Tissue", facetBy="Factor", lineBy="Replicate")
```

ChIPQC will create a folder called 'ChIPQCReport' in your working directory that contains the html report. You can see the report in your browser by opening file '`ChIPQCReport/ChIPQC.html`'.

tip: to check your working directory type:

```
getwd()
```

5.6 Conclusion

The analysis of the quality of this experiment's ChIP quality using ChIPQC has identified several informatics characteristics of the data as well as highlighted variability in quality across ChIPs and cell-lines.

The Histone, RNA Pol2 and DNase ChIPs all have high RIP% and SSD after blacklisting as expected from broader epigenetic marks. Cross-coverage score profiles illustrate the broader regions of enrichment for the Histone marks, tighter profiles for the RNA Pol2 and RNA Pol2-ser2 marks, due to their narrow enrichment in TSSs and a sharp narrow profile for the DNase ChIP. These epigenetic marks showed the characteristic enrichment within TSSs with RNA Pol2-ser2 most enriched for 3'UTRs.

The transcription factor ChIPs showed a much wider variability in RIP%, SSD and RelCC than seen for histone, pol2 and DNase ChIPs. Pu1 and Irf were found to be highly efficient ChIPs and Pu1 was seen in its cross coverage scores to have a broader enrichment pattern than seen for other TFs. The Ikaros ChIPs were found to have acceptable but low enrichment for signal by all metrics, and the enrichment for Ikaros ChIP signal was seen to fit known concentrations of Ikaros within these cell-lines. The ProB Pu1, Ebf and Irf transcription factors were found to cluster away from Ikaros ChIPs indicating a greater co-occurrence of binding among them than with Ikaros ChIPs.

The inputs used in this study showed different sources of artefact contamination. The Ch12 input showed a strong artefact, read length, peak in cross-coverage scores and significant pile-up of signal. Following blacklisting its SSD score didn't drop to that of a background level and so flagged this as a control for further blacklisting. The ProB input however showed peak like signal in its cross-coverage scores profile, a high level of duplication and a spike in its coverage histogram but removal of known blacklisted regions removed much of artefact signal as measured by SSD. Taken together, this suggests that the ProB input contains highly duplicated peak shaped spikes in signal which were contained within blacklisted regions.

6 Advanced Topics

6.1 Providing additional data to ChIPQC plotting and reporting

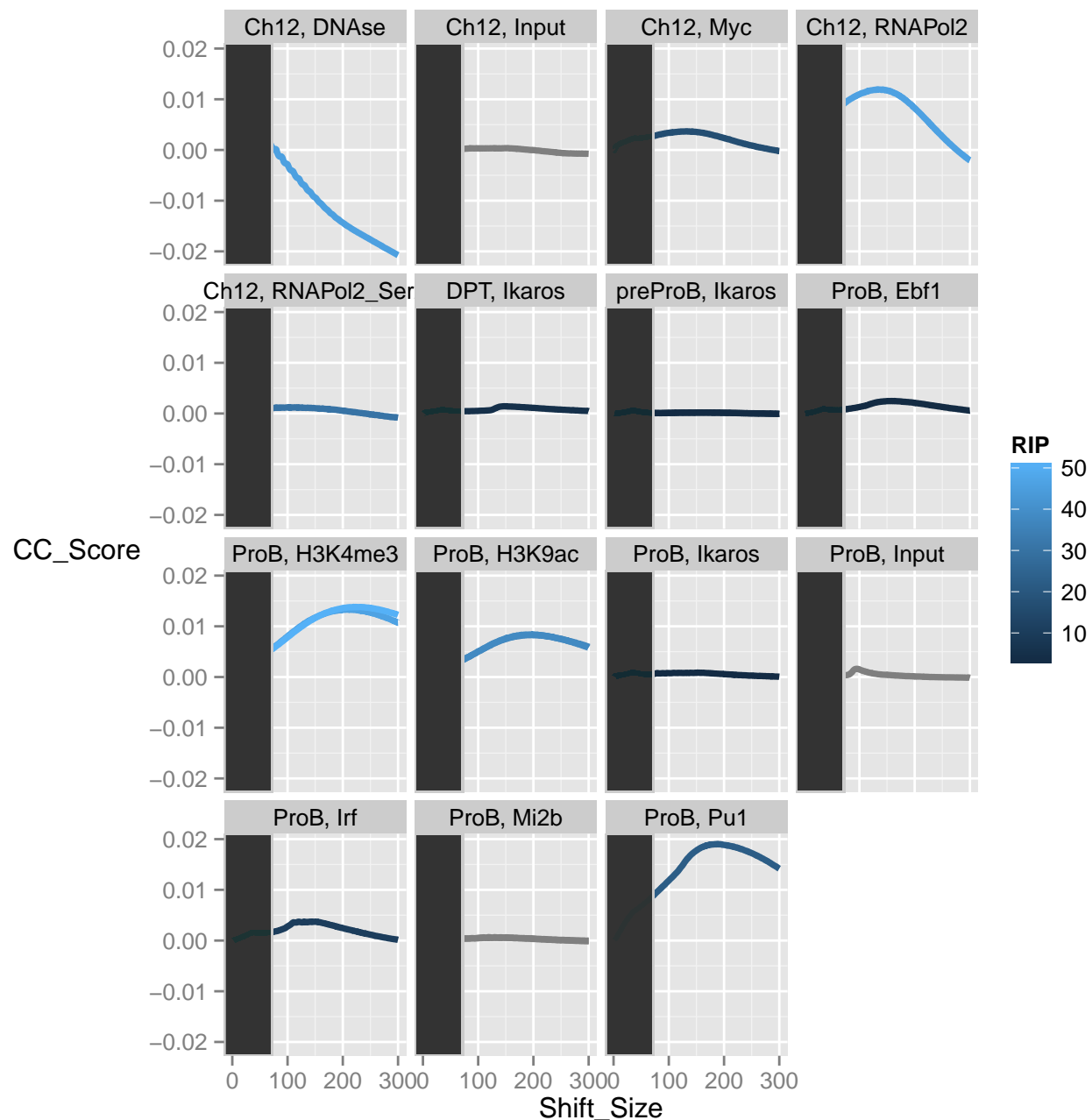
The sample sheets for ChIPQC may contain the optional metadata columns `Tissue`, `Factor`, `Condition` and `Treatment`. In order to allow the user to specify custom metadata for their plotting and reporting, the additional `addMetadata` argument

can be supplied with a data frame of sample IDs and associated additional metadata. The first column for addMetadata data frame must be SampleID and remaining columns may be categorical or discrete data.

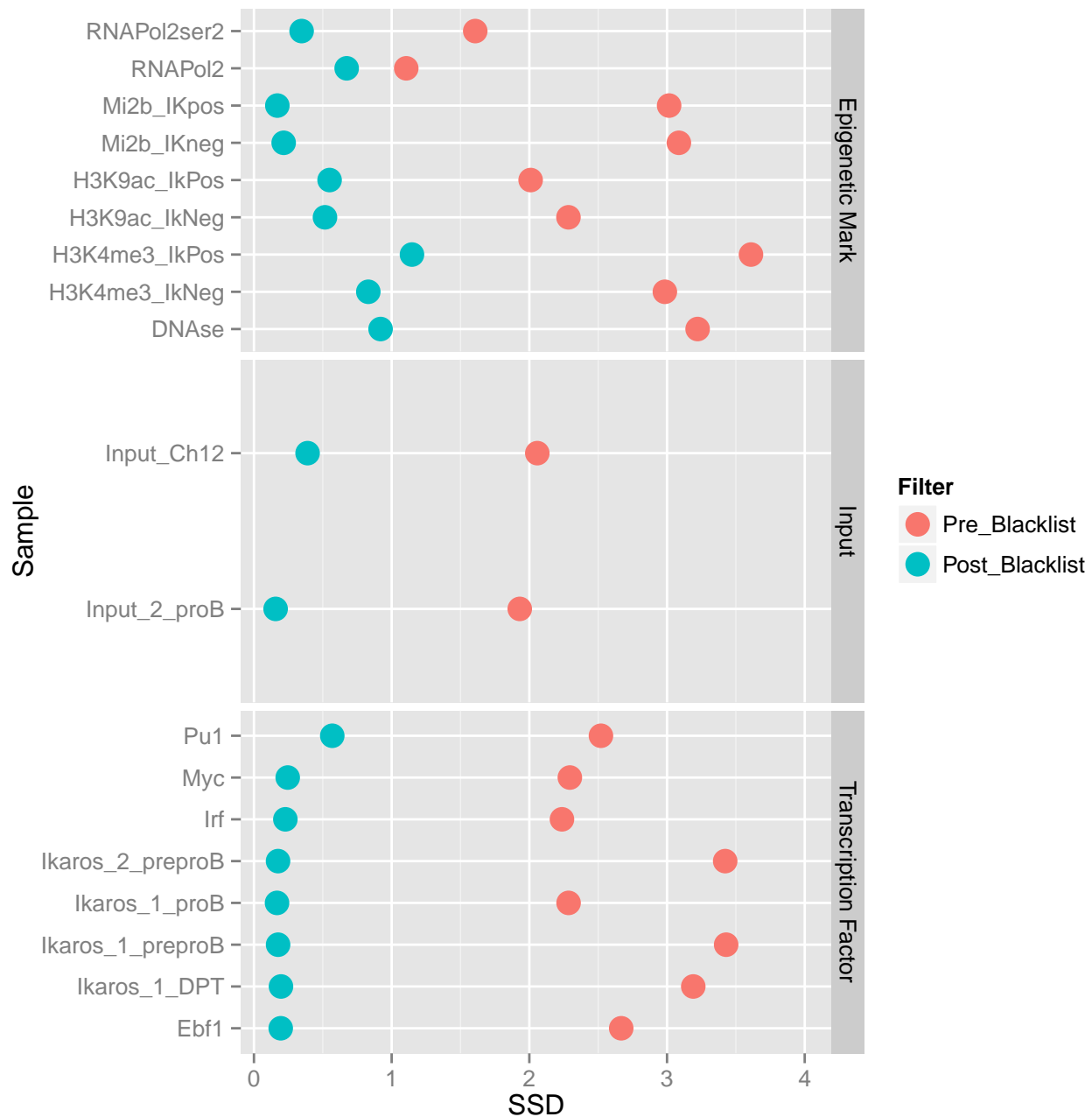
Here, first we illustrate the relationship between RIP and cross coverage scores by including RIP metrics as metadata and setting the column as colourBy argument.

Then we can use addMetadata argument to group SSD by their ChIP type to highlight higher SSD scores typically seen to Epigenetic marks.

```
metrics <- QCmetrics(resExperiment)
metricsMetadata <- data.frame(SampleID=rownames(metrics),RIP =metrics[, "RiP%",drop=T])
plotCC(resExperiment,addMetaData = metricsMetadata,colourBy="RIP")
```



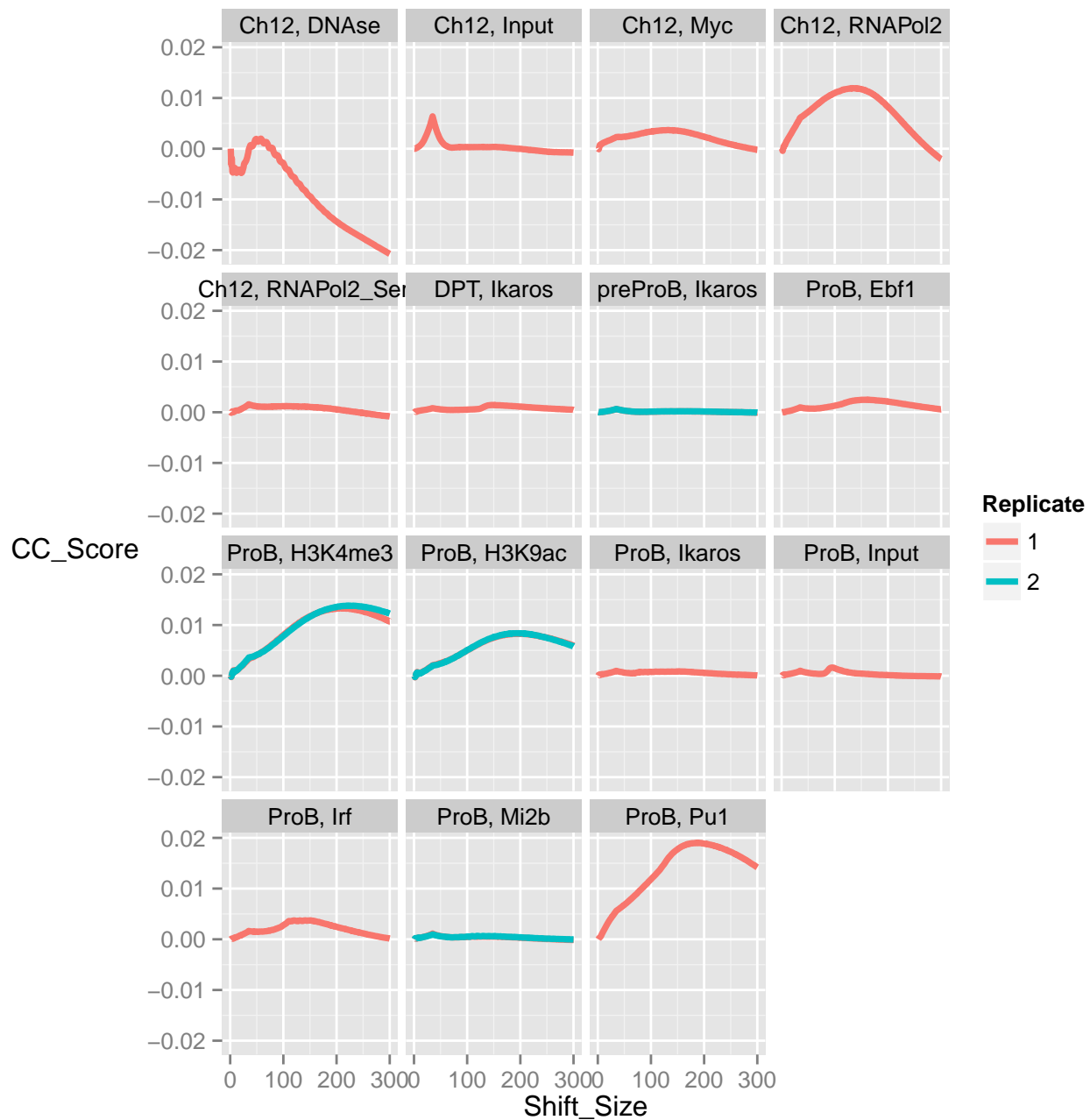
```
metricsMetadata <- data.frame(SampleID=rownames(metrics),ChIPType=
  c(rep("Epigenetic Mark",1),
    rep("Transcription Factor",1),
    rep("Epigenetic Mark",4),
    rep("Transcription Factor",4),
    rep("Input",2),
    rep("Transcription Factor",1),
    rep("Epigenetic Mark",2),
    rep("Transcription Factor",2),
    rep("Epigenetic Mark",2)))
plotSSD(resExperiment,addMetaData = metricsMetadata,facetBy="ChIPType")
## Using Sample as id variables
```



6.2 Some plotting tips

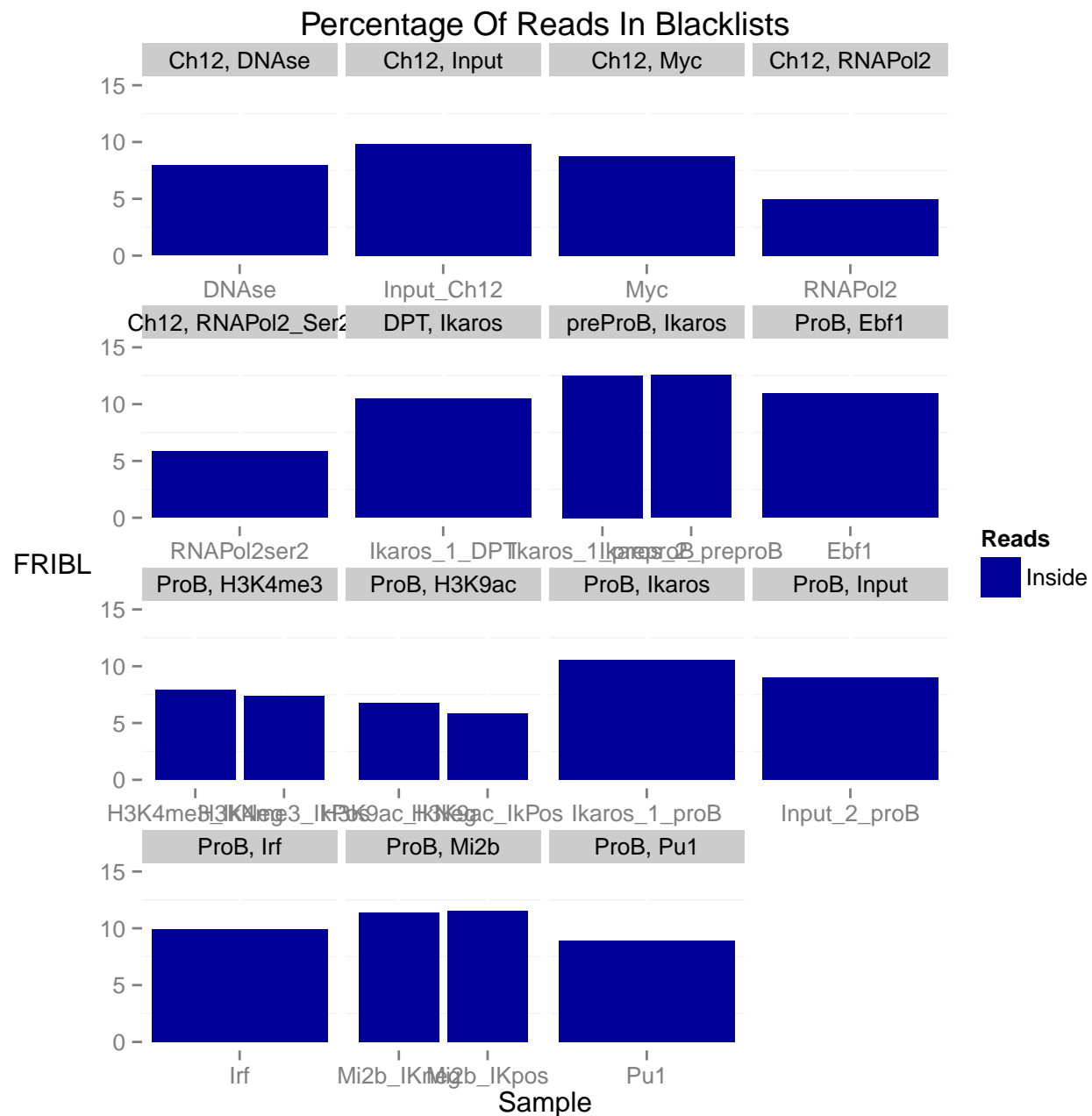
The `plotCC` function produces a line plot with the section excluded from identification of fragment length cross-coverage score shaded in grey. To remove this the `plotCC` ggplot object maybe altered and replotted.

```
ccplot <- plotCC(resExperiment)
ccplot$layers <- ccplot$layers[1]
ccplot
```

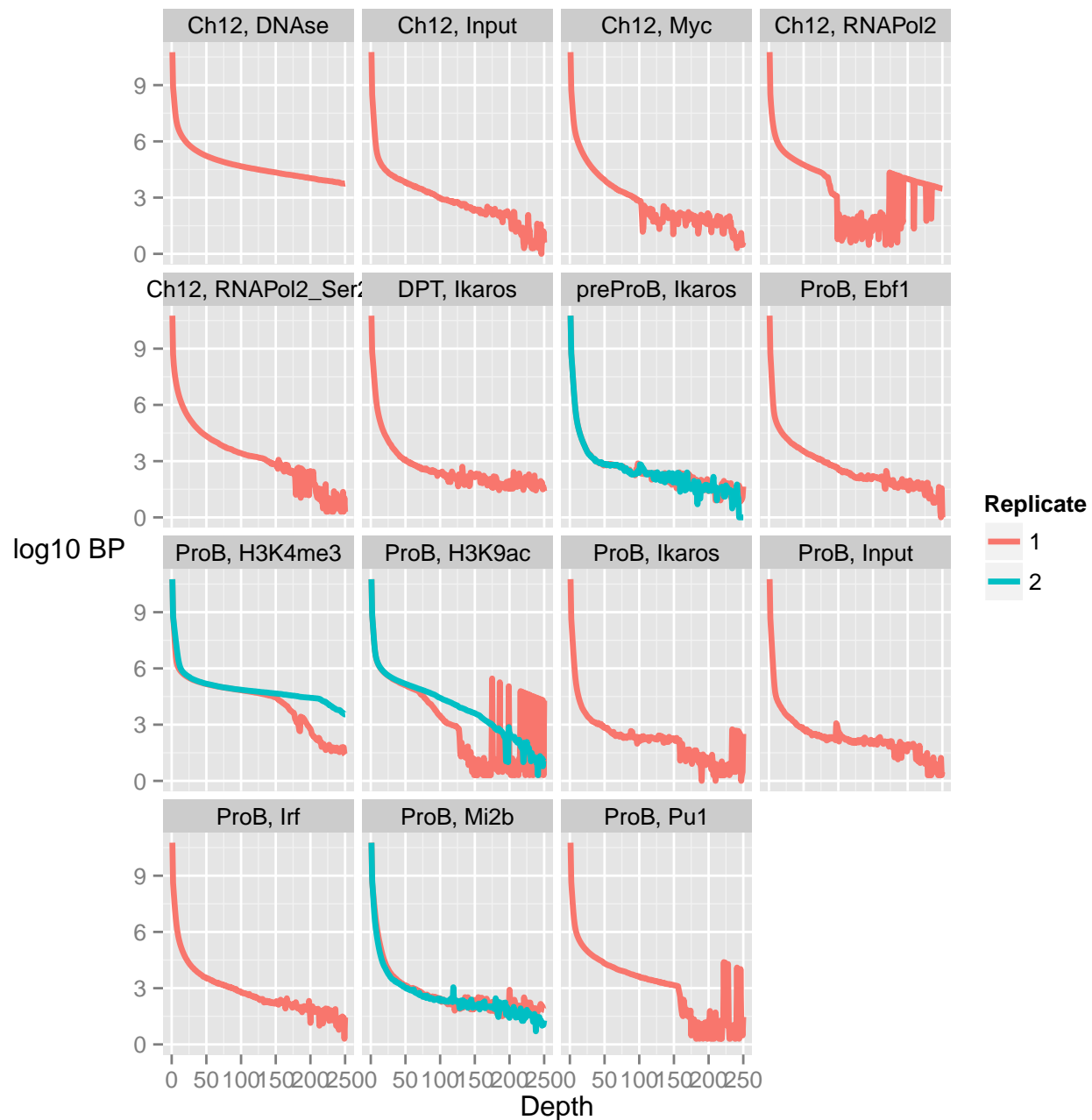
Another useful tip is expanding or narrowing the Y and X axis limits post plotting. This can be done by simply adding the `xlim()` and `ylim()` arguments

```
plotFribl(resExperiment) + ylim(0,15)
## Using Sample as id variables
```



```
plotCoverageHist(resExperiment) + xlim(0,250)
```

```
## Scale for 'x' is already present. Adding another scale for 'x', which will replace the existing scale.
```



7 References

- [1] Carroll T Liang Z , Salama R, Stark R and Santiago I Impact of artifact removal on ChIP quality metrics in ChIP-seq and ChIP-exo data, *Frontiers in Genetics*, April 2014.
- [2] Carroll T. Assessing ChIP-seq sample quality with CHIPQC. BioC 2014, Boston. 2014 http://www.bioconductor.org/help/course-materials/2014/BioC2014/Bioc2014_CHIPQC_practical.pdf

8 Session Info

```
sessionInfo();

## R version 3.2.1 (2015-06-18)
## Platform: x86_64-apple-darwin10.8.0 (64-bit)
## Running under: OS X 10.7.5 (Lion)
##
## locale:
## [1] en_GB.UTF-8/en_GB.UTF-8/en_GB.UTF-8/C/en_GB.UTF-8/en_GB.UTF-8
##
## attached base packages:
## [1] parallel stats graphics grDevices utils datasets methods
## [8] base
##
## other attached packages:
## [1] ChIPQC_1.0.9 DiffBind_1.10.2 limma_3.20.9
## [4] ggplot2_1.0.1 GenomicAlignments_1.0.6 BSgenome_1.32.0
## [7] Rsamtools_1.16.1 Biostrings_2.32.1 XVector_0.4.0
## [10] GenomicRanges_1.16.4 GenomeInfoDb_1.0.2 IRanges_1.22.10
## [13] BiocGenerics_0.10.0 knitr_1.10.5
##
## loaded via a namespace (and not attached):
## [1] gtools_3.5.0 reshape2_1.4.1 lattice_0.20-33
## [4] colorspace_1.2-6 amap_0.8-14 testthat_0.10.0
## [7] stats4_3.2.1 base64enc_0.1-2 DBI_0.3.1
## [10] BiocParallel_0.6.1 RColorBrewer_1.1-2 foreach_1.4.2
## [13] plyr_1.8.3 stringr_1.0.0 zlibbioc_1.10.0
## [16] munsell_0.4.2 gtable_0.1.2 hwriter_1.3.2
## [19] caTools_1.17.1 codetools_0.2-14 memoise_0.2.1
## [22] evaluate_0.7 labeling_0.3 chipseq_1.14.0
## [25] latticeExtra_0.6-26 Biobase_2.24.0 highr_0.5
## [28] proto_0.3-10 Rcpp_0.11.6 KernSmooth_2.23-15
## [31] edgeR_3.6.8 scales_0.2.5 checkmate_1.6.1
## [34] formatR_1.2 gdata_2.17.0 ShortRead_1.22.0
## [37] sendmailR_1.2-1 gplots_2.17.0 brew_1.0-6
## [40] BatchJobs_1.6 BiocStyle_1.2.0 fail_1.2
## [43] digest_0.6.8 stringi_0.5-5 BBmisc_1.9
## [46] grid_3.2.1 tools_3.2.1 bitops_1.0-6
## [49] magrittr_1.5 RSQLite_1.0.0 crayon_1.3.1
## [52] MASS_7.3-43 Nozzle.R1_1.1-1 iterators_1.0.7
```