ChIPQC practical

Dóra Bihary

13 September, 2017

Contents

1 Introduction **1**

2 An example of running the ChIPQC package **1**
2.1 Reading in the data . . . 2
2.2 Perform quality control using ChIPQC . . . 2
2.3 Summary of quality metrics . . . 3
2.4 Plotting reads in peaks . . . 4
2.5 Plotting coverage heatmaps . . . 4
2.6 Principal component analysis . . . 5
2.7 Plotting cross-coverage . . . 6
2.8 Distribution of signal within peaks . . . 7
2.9 Distribution of signal in annotated regions . . . 8
2.10 Plotting peak profiles . . . 9
2.11 Saving the results . . . 10

This tutorial was created based on its previous version presented by Ines de Santiago at the Analysing Next Generation Sequencing data using Bioconductor course in 2015 and the online ChIPQC Vignette.

1 Introduction

ChIPQC is a Bioconductor tool to analyse ChIP-seq data. It includes functions to visualise and assess the data quality of already aligned or peak called reads.

There are two main functions in the ChIPQC package:

- ChIPQCsample() - generates QC report for a single ChIP-seq sample,
- ChIPQC() - generates QC report for a ChIP-seq experiment by wrapping the functionality of ChIPQCsample to multiple datasets.

For both functions you need to provide the aligned reads (single or multiple .bam files), but giving a peak file is optional. If this is missing, obviously no peak-related metrics will be calculated.

2 An example of running the ChIPQC package

For this tutorial we are using the preprocessed dataset that you can find in the *~/Course_Materials/ChIPSeq/Preprocessed/* folder. This dataset contains samples from a ChIP-seq experiment against two different transcription factors, tp53 and tp73 (TAp73beta isoform) in the human osteosarcoma cell lines Saos-2. Reads were mapped to grch38 genome using BWA and peaks were called by MACS2. The aim of this exercise is to get an idea about the quality of the sequence alignment and peak calling.

2.1 Reading in the data

First, let's load the ChIPQC library and set the working directory to the folder where you can find the sample sheet describing the data we will process:

```
library("ChIPQC")
library("TxDb.Hsapiens.UCSC.hg38.knownGene")
setwd("~/Course_Materials/ChIPSeq/ChIPQC_DiffBind/")
```

You can read in the sample sheet to have a closer look at it:

```
samples <- read.csv("sampleSheet.csv")
samples
##       SampleID Tissue    Factor Condition Treatment Replicate
## 1 TAp73beta_r1  SAOS2 TAp73beta TAp73beta TAp73beta         1
## 2 TAp73beta_r2  SAOS2 TAp73beta TAp73beta TAp73beta         2
## 3       p53_r1  SAOS2       p53       p53       p53         1
## 4       p53_r2  SAOS2       p53       p53       p53         2
##
## 1 /home/participant/Course_Materials/ChIPSeq/Preprocessed/Alignment_BWA/TAp73beta_r1.fastq_trimmed.faste
## 2 /home/participant/Course_Materials/ChIPSeq/Preprocessed/Alignment_BWA/TAp73beta_r2.fastq_trimmed.faste
## 3      /home/participant/Course_Materials/ChIPSeq/Preprocessed/Alignment_BWA/tp53_r1.fastq_trimmed.faste
## 4      /home/participant/Course_Materials/ChIPSeq/Preprocessed/Alignment_BWA/tp53_r2.fastq_trimmed.faste
##   ControlID
## 1     input
## 2     input
## 3     input
## 4     input
##                                                                                                    bamCo
## 1 /home/participant/Course_Materials/ChIPSeq/Preprocessed/Alignment_BWA/input.fastq_trimmed.fastq_sorte
## 2 /home/participant/Course_Materials/ChIPSeq/Preprocessed/Alignment_BWA/input.fastq_trimmed.fastq_sorte
## 3 /home/participant/Course_Materials/ChIPSeq/Preprocessed/Alignment_BWA/input.fastq_trimmed.fastq_sorte
## 4 /home/participant/Course_Materials/ChIPSeq/Preprocessed/Alignment_BWA/input.fastq_trimmed.fastq_sorte
##
## 1 /home/participant/Course_Materials/ChIPSeq/Preprocessed/Peaks/TAp73beta_r1.fastq_trimmed.fastq_sorted_
## 2 /home/participant/Course_Materials/ChIPSeq/Preprocessed/Peaks/TAp73beta_r2.fastq_trimmed.fastq_sorted_
## 3      /home/participant/Course_Materials/ChIPSeq/Preprocessed/Peaks/tp53_r1.fastq_trimmed.fastq_sorted_
## 4      /home/participant/Course_Materials/ChIPSeq/Preprocessed/Peaks/tp53_r2.fastq_trimmed.fastq_sorted_
##   PeakCaller
## 1        bed
## 2        bed
## 3        bed
## 4        bed
```

This is a summary of our dataset. Each line in the sample sheet represents a single sample. You define which sample belongs to which tissue/condition/factor/treatment. You provide the path where the .bam files with the mapped reads and the peak files (.bed/.narrowPeak etc) with the peak regions are accessible. You can also define the control file you were using for peak calling.

2.2 Perform quality control using ChIPQC

We can perform the ChIPQC experiment with the following command:

```
exampleExp <- ChIPQC(samples, annotation="hg38", consensus=TRUE)
## [1] "chr3"
```

```
exampleExp
## Samples: 5 : TAp73beta_r1 TAp73beta_r2 p53_r1 p53_r2 input
##              Tissue    Factor     Condition     Treatment Replicate Peaks
## TAp73beta_r1  SAOS2 TAp73beta     TAp73beta     TAp73beta         1  6185
## TAp73beta_r2  SAOS2 TAp73beta     TAp73beta     TAp73beta         2  2962
## p53_r1        SAOS2       p53           p53           p53         1  1069
## p53_r2        SAOS2       p53           p53           p53         2  3450
## input         SAOS2   Control TAp73beta:p53 TAp73beta:p53        c1  2282
##                 Reads Map% Filt% Dup% ReadL FragL   RelCC  SSD  RiP%
## TAp73beta_r1 3252639  100  62.9    0    35   117 0.17000 3.63 5.590
## TAp73beta_r2 2887535  100  58.6    0    35   155 0.23300 5.56 5.200
## p53_r1       1097382  100  69.1    0    32   115 0.42800 8.91 3.140
## p53_r2       3258590  100  60.8    0    35    72 0.02940 7.11 3.570
## input        3273180  100  61.0    0    35    71 0.00306 6.61 0.516
```

The *annotation* parameter can be a built in annotation (default is "hg19", but now we use "hg38", since this is the genome build we used for sequence alignment), or you can define a prebuilt annotation as well. The "consensus=TRUE" parameter will create a consensus peak set so that even the control samples can be compared to determine their enrichment. You can also define the chromosome/chromosomes based on which you want to perform the analysis, however the default value here is the first chromosome that has peaks (in a real example usually chr1), so in this example there is no need to set this parameter.

2.3 Summary of quality metrics

Using the *QCmetrics()* function we can examine the quality metrics calculated by ChIPQC.

```
QCmetrics(exampleExp)
##                 Reads Map% Filt% Dup% ReadL FragL   RelCC  SSD  RiP%
## TAp73beta_r1 3252639  100  62.9    0    35   117 0.17000 3.63 5.590
## TAp73beta_r2 2887535  100  58.6    0    35   155 0.23300 5.56 5.200
## p53_r1       1097382  100  69.1    0    32   115 0.42800 8.91 3.140
## p53_r2       3258590  100  60.8    0    35    72 0.02940 7.11 3.570
## input        3273180  100  61.0    0    35    71 0.00306 6.61 0.516
```

This table shows us the total number of reads in each sample the percentage of these that were successfully mapped, the percentage of those passing the mapping quality threshold (15 by default), the duplication rates, the read length and the estimated fragment length. You can find a bit more detailed description about some of the columns below:

FragL

This is the estimated mean fragment length that is calculated from the data by systematically shifting the reads on each strand towards each other until the highest degree of cross-coverage is obtained.

RelCC

RelCC is a metric of ChIP-enrichment. It is calculated by comparing the maximal 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. This measure is quite low in our datasets, indicating that the quality of the data might not be that great.

SSD

SSD is another metric of ChIP-enrichment. It is looking at the density of positions with different pileup values. The higher the value the more successful the ChIP was.

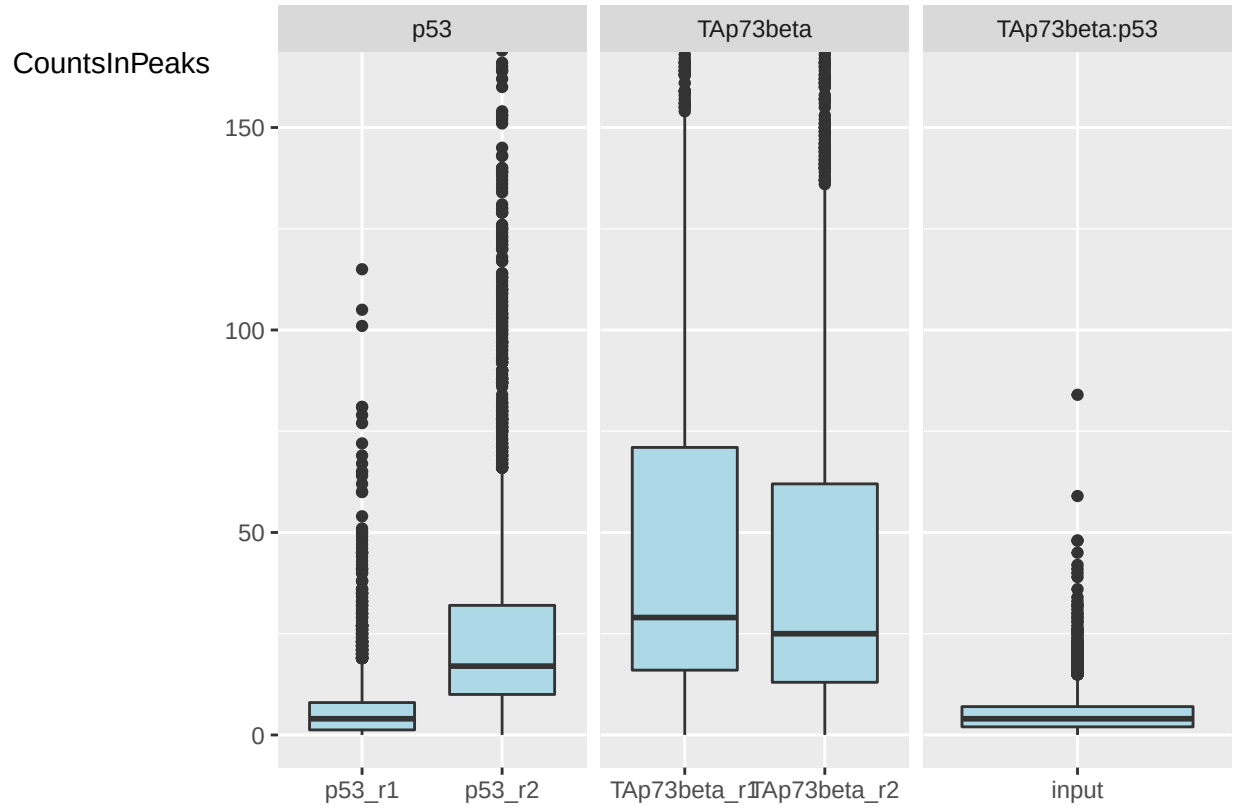
RiP%

RiP stands for reads in peaks. This is again an indicator of good enrichments. ChIPs around 5% or higher usually indicate successful enrichment.

2.4 Plotting reads in peaks

With the help of the *plotRap()* function we can visualise barplots of the number of reads that overlap with peaks vs background reads.

```
plotRap(exampleExp, facetBy="Condition")
```

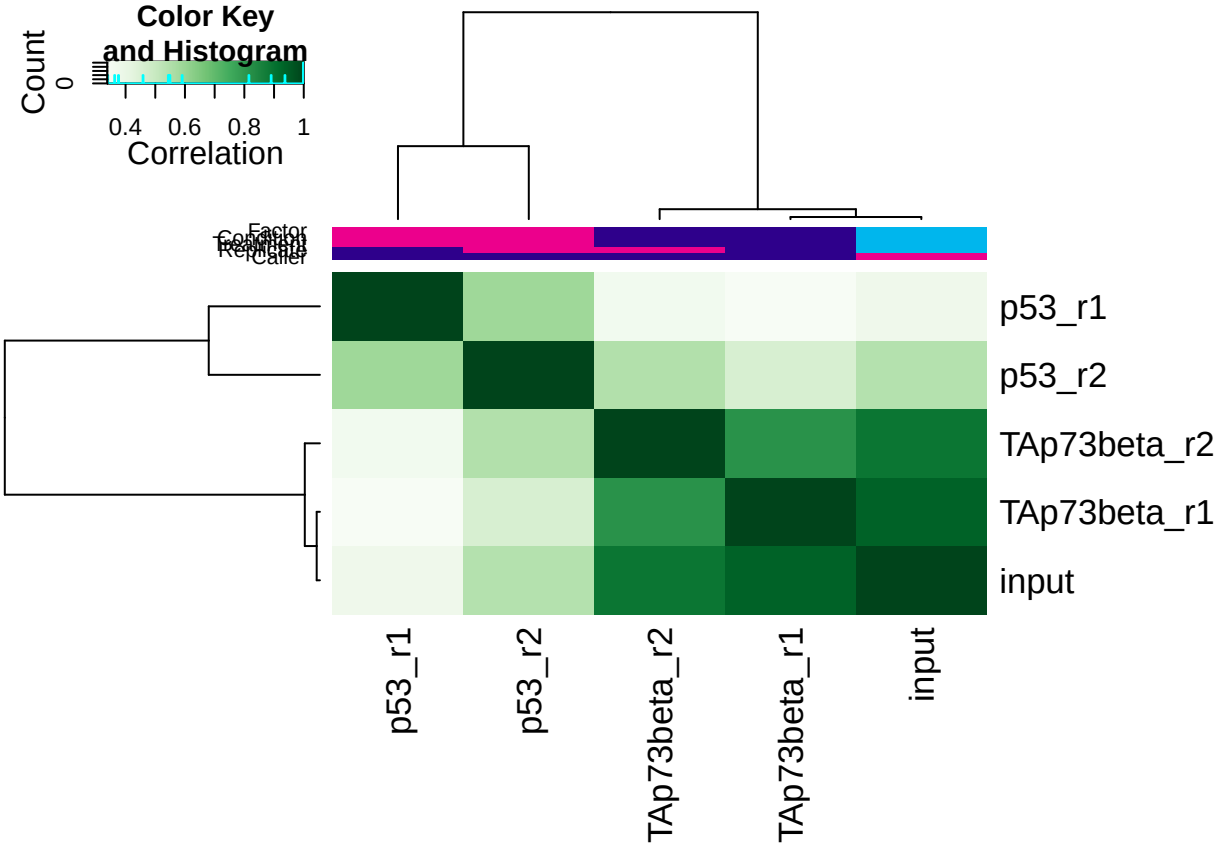


2.5 Plotting coverage heatmaps

We can also have a look at the correlation between samples using binding information. This function is based on the co-occurrence of peaks using methods from the DiffBind package.

We can see that the p53 datasets clusters separately, however the p73 samples seem to be really similar to the input.

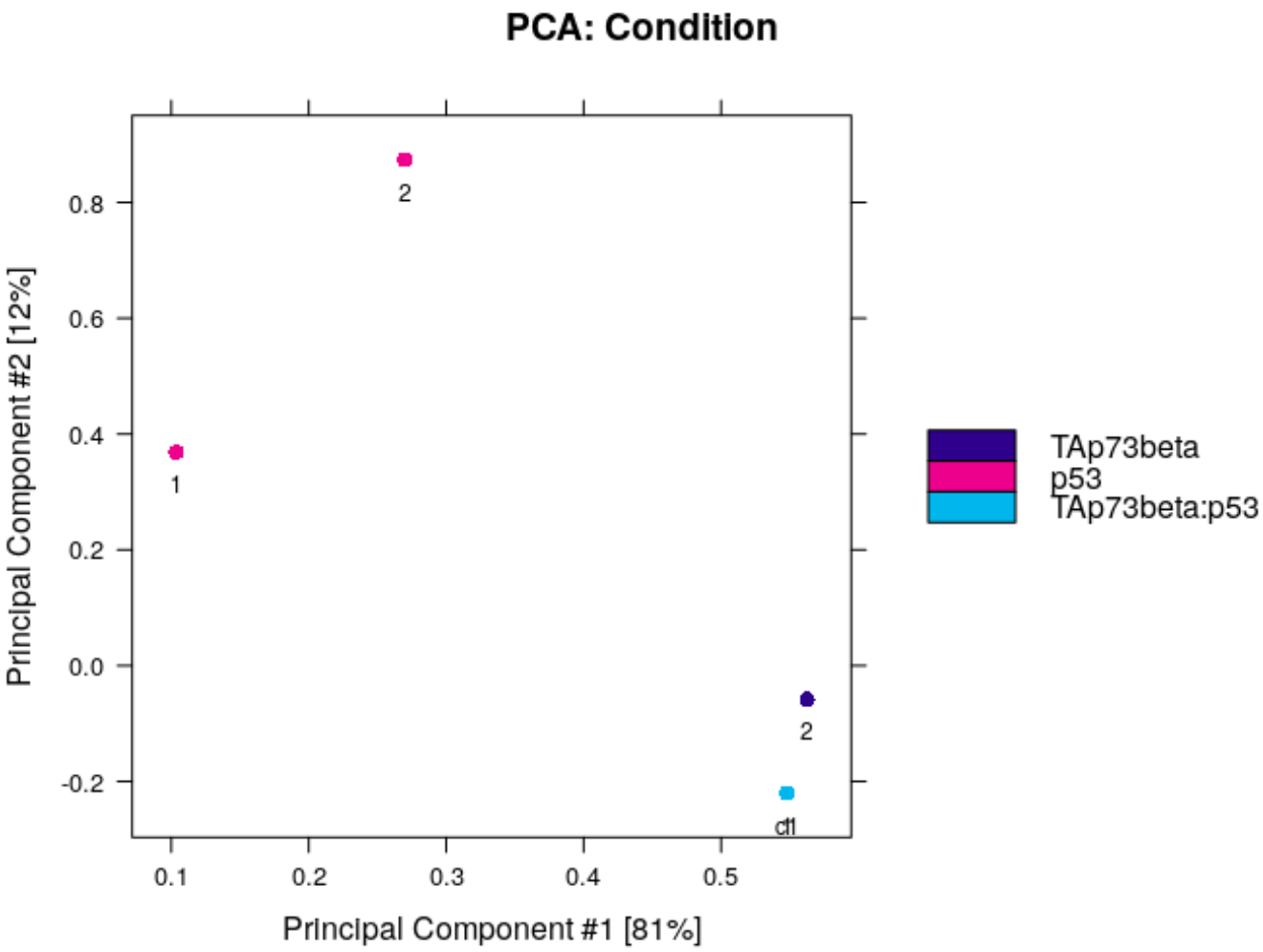
```
plotCorHeatmap(exampleExp,  attributes = "Condition:Replicate", lineBy = "Replicate")
```



2.6 Principal component analysis

PCA (Principal Component Analysis) can give us an idea about how the samples are associated. We see the same type of separation of the sampels as on the previous plot.

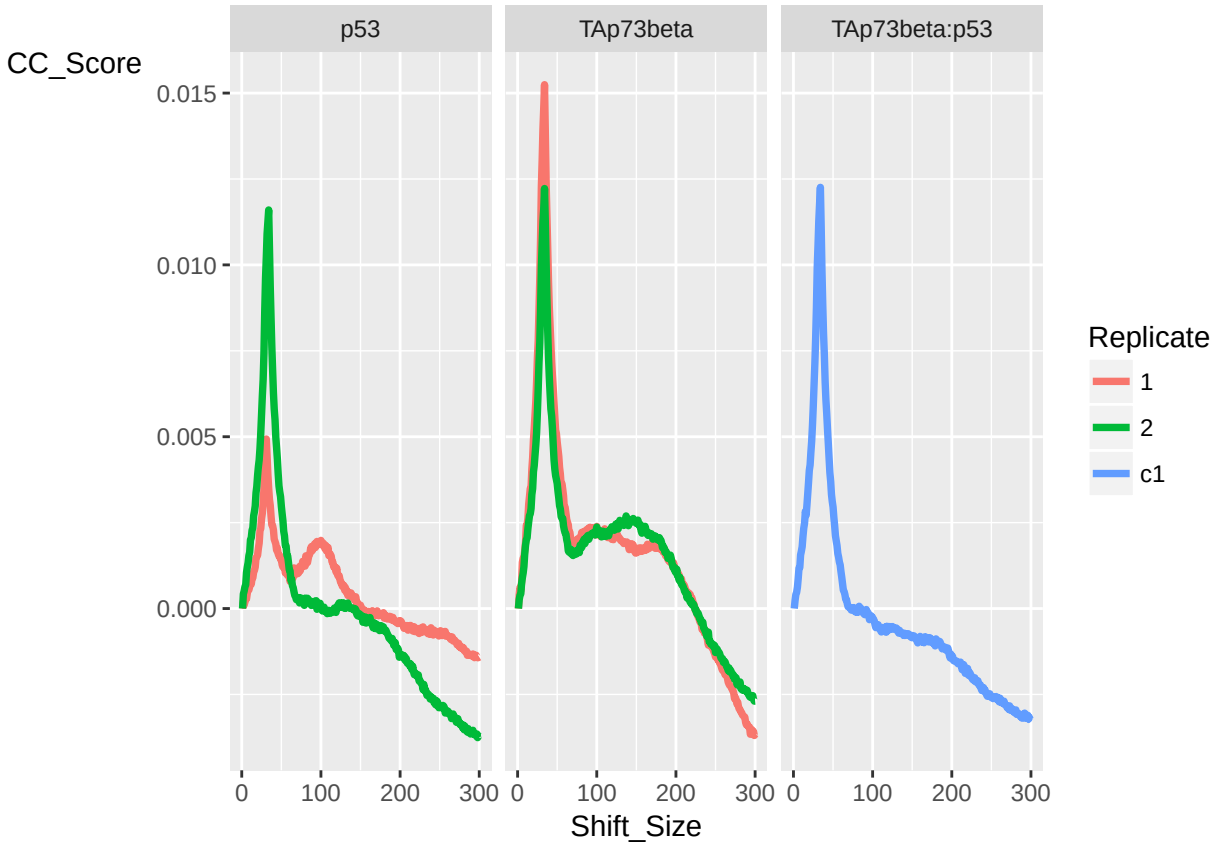
```
plotPrincomp(exampleExp, attributes = "Condition", label="Replicate", dotSize=1, labelSize=0.75)
```



2.7 Plotting cross-coverage

A cross-coverage plot can be generated as follows:

```
plotCC(exampleExp, facetBy="Condition")
```



This plot shows the cross-coverage values over a range of shift sizes. Usually there is a smaller or bigger (in our case quite high) peak on these plots at the read length, this is why the area from 0 to 1,5*readLength is excluded when identifying the fragment length. Usually ChIP peaks have a higher peak at fragment length (that you can't see for inputs), as shifting the reads on both strands should increase coverage at peak sites.

This metric is generated using the following values:

FragCC = *CrossCoverageScore_max*

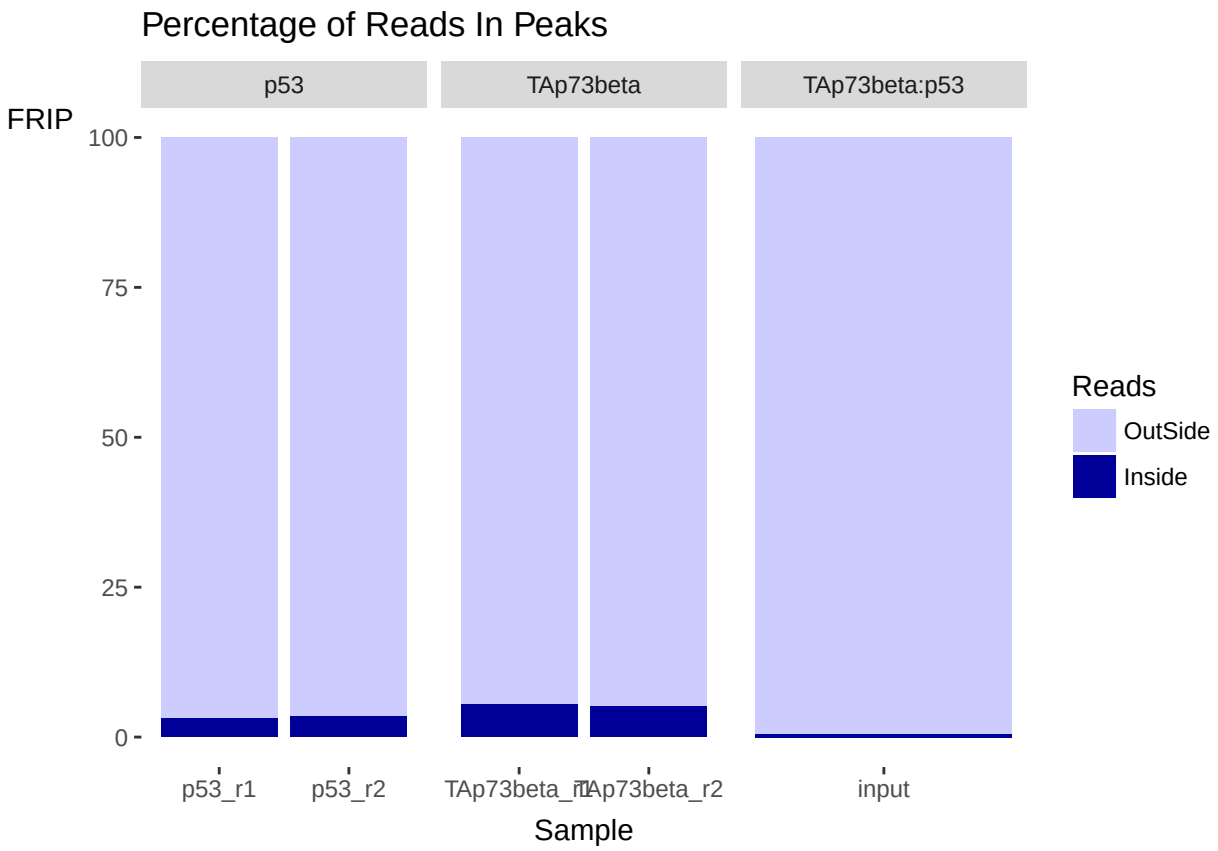
RelCC = *CrossCoverageScore_max*/*CrossCoverageScore_readLength*

```
FragmentLengthCrossCoverage(exampleExp)
## TAp73beta_r1 TAp73beta_r2       p53_r1       p53_r2        input
## 0.0023485073 0.0026208383 0.0019022762 0.0003092794 0.0000340492
ReadLengthCrossCoverage(exampleExp)
## TAp73beta_r1 TAp73beta_r2       p53_r1       p53_r2        input
##  0.013849645  0.011228478  0.004442816  0.010504589  0.011139430
RelativeCrossCoverage(exampleExp)
## TAp73beta_r1 TAp73beta_r2       p53_r1       p53_r2        input
##  0.169571655  0.233409925  0.428169014  0.029442316  0.003056638
```

2.8 Distribution of signal within peaks

The fraction of signal in peaks can also give us an understanding of the ChIP's efficiency. As we would expect, this count is significantly lower for the input than for the ChIP samples.:

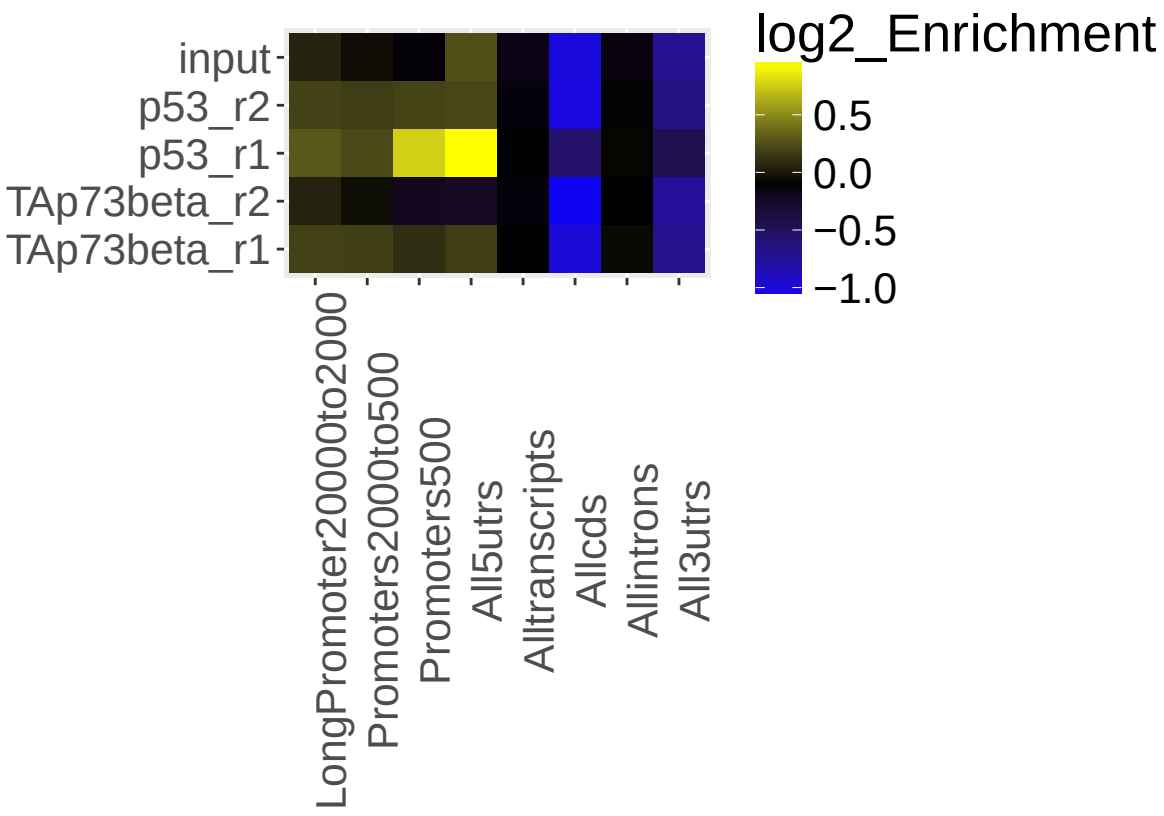
```
plotFrip(exampleExp, facetBy="Condition")
## Using Sample as id variables
```



2.9 Distribution of signal in annotated regions

Different samples can have different expected enrichment if certain genomic locations like promoters, 5'UTRs, introns etc. Plotting the distribution of the signal in these annotated regions as you see below can help us identify whether our samples behave in an expected way resulting in higher or lower enrichment at certain regions or it can help to reveal interesting, previously unknown behaviours. We can also have a closer look at the enrichment values of specific regions of interest.

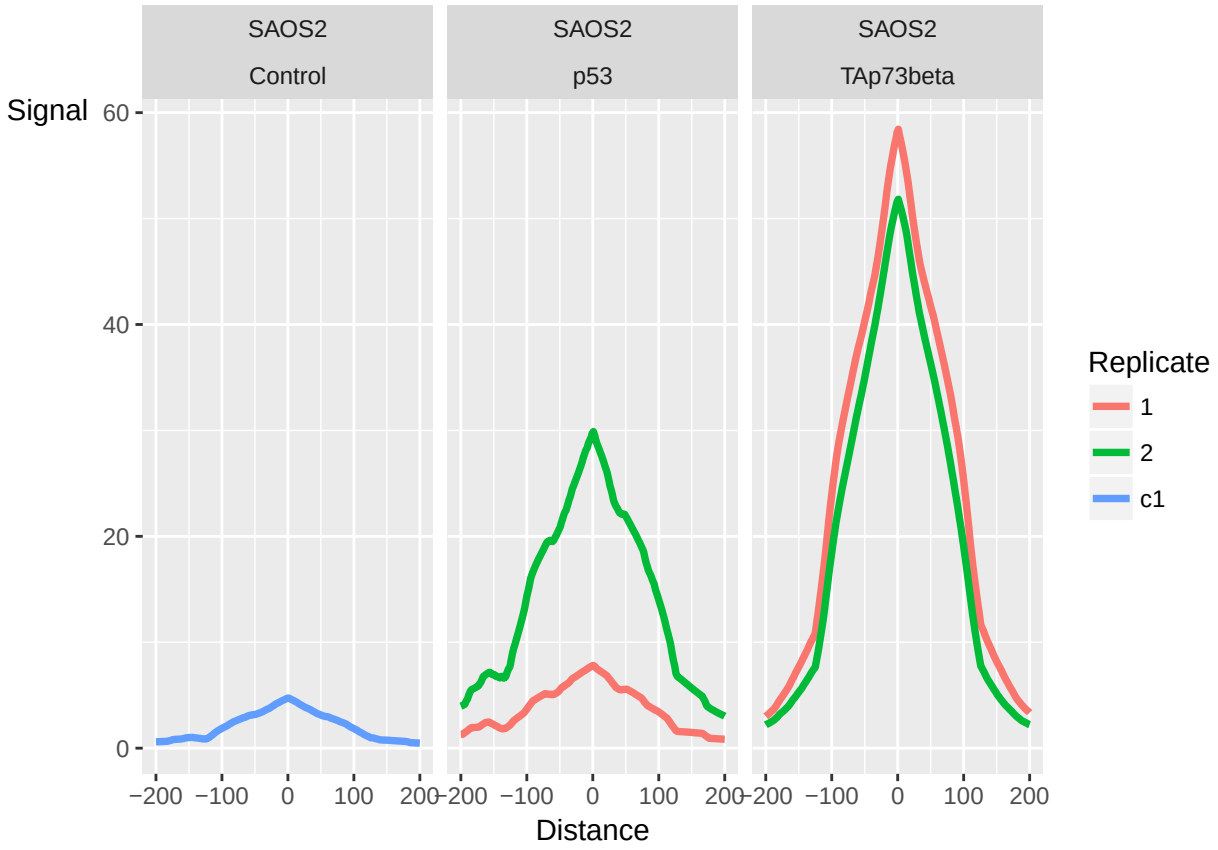
```
plotRegi(exampleExp, facet=F)
```




```
regi(exampleExp)["All5utrs",]
## TAp73beta_r1 TAp73beta_r2       p53_r1       p53_r2        input
##        0.165       -0.238        0.923        0.208        0.248
```

2.10 Plotting peak profiles

```
plotPeakProfile(exampleExp)
```



Each peak is centered on its summits. The shape of good experiments will have distinctive peaks, while the conrtols will be relatively flat. The exact shape can depend on the specific protein that is bound by the experiment and other biological conditions.

2.11 Saving the results

With the help of *ChIPQCreport()* function we can save all the above mentioned plots easily.

```
ChIPQCreport(exampleExp, facetBy = "Condition", lineBy = "Replicate")
```