Assessing ChIP-seq sample quality with *ChIPQC*

Thomas Carroll and Rory Stark

Edited: March 24, 2014; Compiled: August 25, 2023

Contents

```
> sample = ChIPQCsample("chip.bam")
> sample
> ChIPQCreport(sample)
```

The following examples explore this in more depth.

2 Example 1: ENCODE data set with problematic samples

The first example, derived using ENCODE data for three transcription factors (`CTCF`, `cMYC`, and `E2F1`) [**?**]. Each ChIP has been performed in replicate, for a total of six samples (there are no controls in this example, see the following example for working with controls).

The actual reads and peak for this experiment are not included with this vignette as they are too large. The sample sheet (see below) include the SRA SRR numbers for each of the experiments, so you can download reads and peaks to fully run the vignette. If you do so, you should put the reads in a sub directory called `reads` and the peaks in a sub directory called `peaks`. We intend to make all the vignette data more easily obtainable; email the package authors if you are interested.

2.1 Experiment sample sheet

The first step is to construct a sample sheet describing the ChIP-seq experiment. This can be passed as a data frame, or saved in a csv file. The experiment can also be represented by a `DBA` object constructed using the *DiffBind* package, which accepts the same sample sheets as *ChIPQC*. A csv sample sheet has been included for this data set, and can be accessed as follow:

```
> library(ChIPQC)
> samples = read.csv(file.path(system.file("extdata", package="ChIPQC"),
+                                          "example_QCexperiment.csv"))
> samples
```

```
  SampleID  Tissue Factor Replicate            bamReads
1   CTCF_1    A549   CTCF         1 reads/SRR568129.bam
2   CTCF_2    A549   CTCF         2 reads/SRR568130.bam
3   cMYC_1    A549   cMYC         1 reads/SRR568131.bam
4   cMYC_2    A549   cMYC         2 reads/SRR568132.bam
5   E2F1_1 HeLa-S3   E2F1         1 reads/SRR502355.bam
6   E2F1_2 HeLa-S3   E2F1         2 reads/SRR502356.bam
                           Peaks
1 peaks/SRR568129_chr22_peaks.bed
2 peaks/SRR568130_chr22_peaks.bed
3 peaks/SRR568131_chr22_peaks.bed
4 peaks/SRR568132_chr22_peaks.bed
5 peaks/SRR502355_chr22_peaks.bed
6 peaks/SRR502356_chr22_peaks.bed
```

This sample sheet details the metadata for the experiment. It also contains file paths to the aligned reads and previously called peaks. Note that if you have the sample sheet in a .csv file, you do not have to first load it into a data frame – the filename can be passed directly to `ChIPQC`.

2.2 Constructing a `ChIPQCexperiment` object

The main entry point for assessing experiments is `ChIPQC`. This takes a sample sheet and some other optional parameters and computes quality metrics for each of the individual samples. It does this using the *BiocParallel* package, which by default will run in parallel, using all available cores on your machine. For each unique sample (including controls), a `ChIPQCsample` object us constructed and added to a list. This list of samples forms the core of the `ChIPQCexperiment` object, which is constructed and returned by `ChIPQC`.

The parameters for `ChIPQC` include an `annotation` indicating what assembly of what genome is being analyzed. 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 18, the only chromosome for which data is included.

Other parameters include `mapQCth`, which represents a threshold for filtering on mapping (alignment) quality, which is set to `15` by default. An optional `blacklist` is a file or `GRanges` object with genomic intervals whereby reads in those regions will also be filtered out. Use of blacklists can be very important in ChIP-seq data processing ([**?**]), and should be applied for peak calling as well.

For this example, the main experiment is analyzed as follows:

```
> exampleExp = ChIPQC(samples,annotaiton="hg19")
> exampleExp
```

By default, the "hg19" annotation will use a default blacklist, and only the first chromosome (in this case "chr22") will be examined.

For the purposes of this vignette, we can load a pre-computed version of this object:

```
> data(example_QCexperiment)
> exampleExp

Samples: 6 : CTCF_1 CTCF_2 ... E2F1_1 E2F1_2
        Tissue Factor Replicate Peaks
CTCF_1    A549   CTCF         1  1118
CTCF_2    A549   CTCF         2   648
cMYC_1    A549   cMYC         1   253
cMYC_2    A549   cMYC         2   159
E2F1_1 HeLa-S3   E2F1         1   325
E2F1_2 HeLa-S3   E2F1         2   249
        Reads Map% Filt%   Dup% ReadL FragL RelCC  SSD  RiP% RiBL%
CTCF_1 341055  100  26.3 16.600    28   131 2.740 2.53 31.20  1.33
CTCF_2 303856  100  28.4  7.320    28   131 2.690 1.43 12.80  0.00
cMYC_1 287462  100  31.0 13.600    28    97 0.347 1.47  6.59  1.78
cMYC_2 317537  100  29.9  4.540    28   129 0.386 1.09  2.79  0.00
E2F1_1 223580  100  31.7  1.010    28   101 0.701 1.29  7.80  2.00
```

```
        E2F1_2 194919  100  31.8  0.663    28   107 0.303 1.40  5.36  2.79
```

2.3 Quality metrics summary

Looking at the output, the first line indicates how many samples there are. The next section shows the metadata for the six ChIP samples (cell line, transcription factor ChIP'ed, replicate number, and number of peaks called).

The final section (also retrievable as a *matrix* by calling `QCmetrics(exampleExp)`) shows a summary of the main QC metrics. These include the total number of reads in the bam file for each sample, the percentage of these that were successfully mapped (aligned), and the percentage of reads having a mapping quality score less than or equal to the `mapQCth` (in this case, 15). The percentage of reads that map to the exact position in the genome as at least one other read is then reported. In this example, we can see substantial variance in the duplication rates, although none are extremely high. Indeed, good quality ChIPs for narrowly binding transcriptions factors are expected to have regions of very high enrichment, which will correctly include fragments that originate at the same location. Sequencing of such "duplicate" fragments is expected and biologically meaningful where the factor binds with high affinity. The extremely low duplication rates for the `E2F1` samples are flags of potential problems with these ChIPs.

Next the read length, derived from the data in the bam files, is reported, along with the estimated mean fragment length. 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. 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. In this example, the `RelativeCC` score for the all CTCF samplesis greater than 1, indicating these samples are of high quality.

The SSD score, as implemented in *htSeqTools*, 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. Here the first CTCF replicate shows the highest SSD score and so greatest enrichment for depth of signal.

The final two metrics report the percentage of reads in different regions of interest. The first reports the percentage of reads that overlap called peaks (also known as FRIP). 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.

The final measure reports the percentage of reads that overlapped blacklisted regions (RiBL). 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 ([**?**]). The RiBL score then may act as a guide for the level of background signal in a ChIP or input and is found to be correlated with SSD in input samples and the read length cross coverage score in both input and ChIP samples ([**?**]).

2.4 Generating a summary QC report for experimental sample groups

The easiest way to get a visual overview of the QC metrics for the experiment is to generate a summary HTML report. This is accomplished by calling the `ChIPQCreport` method:

```
> ChIPQCreport(exampleExp)
```

By default, a sub directory will be created named `ChIPQCreport` (this is configurable using the `reportFolder` parameter), which will contain each of the plots. The main interactive HTML page (built using the R package *Nozzle*) is named `ChIPQC.html` by default. This can be loaded into a browser. This report is available at http://ChIPQC.starkhome.com/Reports/exampleExp/ChIPQC.html.

The report is interactive in several ways. The individual sections can be expanded or compressed to focus in on specific areas of the assessment. Table can be sorted by clicking on the column labels. Individual plots can be enlarged or shrunk by clicking on them.

2.4.1 Report: Overview

The first section include a textual overview of the report contents.

2.4.2 Report: QC Summary table

The second section, *QC Summary*, contains a table with the key QC metrics for each sample, similar to the table output by `QCmetrics`.

2.4.3 Report: QC Results plots

The third section contains the plots themselves, divided into three sections. Samples are grouped together using the experimental metadata; by default, samples sharing the same `Tissue` and `Factor` values are considered sample groups (this is controllable by the user using the `facetBy` parameter with `ChIPQCreport()` function).

The first section, *Mapping, Filtering, and Duplication Rate*, focuses on read mapping. begins with a table showing how the sequencing reads can be filtered: how many are mappable (align to a position in the genome) and how many reads pass the mapping quality filter and are not duplicates. Also included are the overall duplication rate for each sample, the percentage that pass the mapping quality filter, and the percentage of total reads that pass both the mapping quality filter and are not duplicates. Next is a plot showing the effect of blacklisting, with the proportion of reads that do and do not overlap with blacklisted regions. The final plot in this section uses the genomic annotation to show where reads map in terms of genomic features. This is represented as a heatmap showing the enrichment of reads compared to the background levels of the feature.

The second section, *ChIP Signal Distribution and Structure*, looks at the inherent "peakiness" of the samples. The first plot is a coverage histogram. The X-axis represent the read pileup height at a basepair position, and the Y-axis represents how many positions have this pileup height. This is on a log scale. A ChIP sample with good enrichment should have a reasonable "tail". This can be captured using the *SSD* metric, which is the standard deviation of this histogram (normalized to library depth so as to allow for sample-to-sample comparisons.) Samples with low enrichment, consisting of mostly background reads and genome wide low pile-up (such as in controls), should have lower SSD values than efficient ChIPs. Equivalence

between ChIP and control samples may reflect either low enrichment for ChIP or, when SSD is high for controls, the presense of large regions of high depth, aberrant signal in control samples and hence a flag for further blacklisting of genomic regions.

The other plot in this section show the cross-coverage scores (*CC Score*) of reads along the two strands at a range of successive shifts. There is usually a distinctive peak around the read length and a second peak should be at the expected fragment length. The read length peak is excluded from identification of the the shift with maximum cross coverage score, region of exclusion shown in red, to allow for the prediction of sensible fragment length scores. Failure to show a clear peak at the fragment length is indicative of a non-enriched sample. This can be seen in the *Relative CC* metric, which is the maximum CC Score (estimated fragment size) divided by the read length CC Score; low values tend to indicate a lack of enrichment.

In this example, both these plots suggest that the CTCF sample group is more enriched than the other two sets of samples, with the first replicate showing particularly high enrichment. The coverage histogram stays higher above the X-axis for the CTCF samples. Likewise, the CTCF samples show a clear peak at the fragment length in the cross-coverage plot, while the other samples do not show a clear peak other than at the read length.

The final set of plots, *Peak Profile and ChIP Enrichment*, are based on metric computed using the supplied peaks if available. The first plot shows average peak profiles, centered on the summit (point of highest pileup) for each peak. These profiles can vary depending on what type of mark is being studied – transcription factor, histone mark, or other DNA-binding protein such as a polymerase – but similar marks usually have a distinctive profile in successful ChIPs.

Next is a bar chart of the Reads in Peaks. ChIP samples with good enrichment will have a higher proportion of their reads overlapping called peaks.The final plot shows these data in a different way, offering a box plot of the distribution of how many reads are in each peak.

Finally, plots showing how the sample cluster are presented. The correlation heatmap is based on correlation values for all the peak scores for each sample. The other plot shows the first two principal component values for each sample. In this example, the replicates do cluster by replicate, which is a positive sign. the `CTCF` replicates are highly correlated and form their own cluster. The `E2F1` replicates are also highly correlated with each other, and anti-correlated with the `CTCF` samples, probably due to their being from different cell lines.

2.4.4 Report: Files and Versions

The final section of the report gives some version details as to the software run.

3 Example 2: ER binding in tamoxifen resistant data set

The next example demonstrates the use of *ChIPQC* with a larger experiment with good overall quality. This includes 11 ChIPs for the transcription factor ER in five cell lines, with at least two replicates for each ChIP, as well as 5 associated input DNA controls (one for each cell line). This data set is derived from [**?**]. The example only includes data from chromosome 18 (chr18) for time and space reasons. The actual aligned reads for the samples, in the form of BAM files, are not included with the package as they are approximately 200MB. The peaks (called using MACS [**?**]) are also not included. To run all aspects of the vignette, you will need to download these separately – inquire of the package maintainers if you are

interested in how to do this. Note that this data set is the same at that used for the vignette and example for the *DiffBind* package, which provides tools for further analyzing ChIP-seq experiments.

3.1 Experimental sample sheet

The first step is to construct a sample sheet describing the ChIP-seq experiment. This can be passed as a data frame, or saved in a csv file. The experiment can also be represented by a `DBA` object constructed using the *DiffBind* package, which accepts the same sample sheets as *ChIPQC*. A csv sample sheet has been included for the tamoxifen data set, and can be accessed as follow:

```
> library(ChIPQC)
> samples = read.csv(file.path(system.file("extdata", package="ChIPQC"),
+                                          "tamoxifenQC.csv"))
> samples
```

```
   SampleID Tissue Factor  Condition  Treatment Replicate
1    BT4741  BT474     ER  Resistant Full-Media         1
2    BT4742  BT474     ER  Resistant Full-Media         2
3     MCF71   MCF7     ER Responsive Full-Media         1
4     MCF72   MCF7     ER Responsive Full-Media         2
5     MCF73   MCF7     ER Responsive Full-Media         3
6     T47D1   T47D     ER Responsive Full-Media         1
7     T47D2   T47D     ER Responsive Full-Media         2
8     TAMR1   MCF7     ER  Resistant Full-Media         1
9     TAMR2   MCF7     ER  Resistant Full-Media         2
10    ZR751   ZR75     ER Responsive Full-Media         1
11    ZR752   ZR75     ER Responsive Full-Media         2
                     bamReads ControlID                  bamControl
1  reads/Chr18_BT474_ER_1.bam    BT474c reads/Chr18_BT474_input.bam
2  reads/Chr18_BT474_ER_2.bam    BT474c reads/Chr18_BT474_input.bam
3   reads/Chr18_MCF7_ER_1.bam     MCF7c  reads/Chr18_MCF7_input.bam
4   reads/Chr18_MCF7_ER_2.bam     MCF7c  reads/Chr18_MCF7_input.bam
5   reads/Chr18_MCF7_ER_3.bam     MCF7c  reads/Chr18_MCF7_input.bam
6   reads/Chr18_T47D_ER_1.bam     T47Dc        reads/T47D_input.bam
7   reads/Chr18_T47D_ER_2.bam     T47Dc        reads/T47D_input.bam
8   reads/Chr18_TAMR_ER_1.bam     TAMRc        reads/TAMR_input.bam
9         reads/TAMR_ER_2.bam     TAMRc        reads/TAMR_input.bam
10  reads/Chr18_ZR75_ER_1.bam     ZR75c        reads/ZR75_input.bam
11  reads/Chr18_ZR75_ER_2.bam     ZR75c        reads/ZR75_input.bam
                     Peaks PeakCaller
1  peaks/BT474_ER_1.bed.gz        bed
2  peaks/BT474_ER_2.bed.gz        bed
3   peaks/MCF7_ER_1.bed.gz        bed
4   peaks/MCF7_ER_2.bed.gz        bed
5   peaks/MCF7_ER_3.bed.gz        bed
6   peaks/T47D_ER_1.bed.gz        bed
7   peaks/T47D_ER_2.bed.gz        bed
8   peaks/TAMR_ER_1.bed.gz        bed
9   peaks/TAMR_ER_2.bed.gz        bed
10  peaks/ZR75_ER_1.bed.gz        bed
```

```
11  peaks/ZR75_ER_2.bed.gz        bed
```

This sample sheet details the metadata for the experiment, including how controls are matches with ChIP samples. It also contains file paths to the aligned reads and previously called peaks. Note that in this case the paths are relative to the current working directory. If you download the associated data to run all step of the vignette, you will need to unzip the "reads" and "peaks" directories as sub directories of where the sample sheet is.

If you have the sample sheet in a .csv file, you do not have to first load it into a data frame – the filename can be passed directly to `ChIPQC`.

3.2 Constructing a `ChIPQCexperiment` object

The main entry point for assessing experiments is `ChIPQC`. This takes a sample sheet and some other optional parameters and computes quality metrics for each of the individual samples. It does this using the *BiocParallel* package, which by default will run in parallel, using all available cores on your machine. For each unique sample (including controls), a `ChIPQCsample` object us constructed and added to a list. This list of samples forms the core of the `ChIPQCexperiment` object, which is constructed and returned by `ChIPQC`.

The parameters for `ChIPQC` include an `annotation` indicating what assembly of what genome is being analyzed. 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 18, the only chromosome for which data is included.

Other parameters include `mapQCth`, which represents a threshold for filtering on mapping (alignment) quality, which is set to `15` by default. An optional `blacklist` is a file or `GRanges` object with genomic intervals whereby reads in those regions will also be filtered out. Use of blacklists can be very important in ChIP-seq data processing ([**?**]), and should be applied for peak calling as well.

For this example, some other features of `ChIPQC` are used as well. By setting `consensus=TRUE`, a consensus peak set will be derived,so that even the control samples can be compared to determine their relative enrichment in these regions. With `bCounts=TRUE`, the peak scores will be computed by counting how many reads overlap these regions for every sample (include controls). This feature relies on the `dba.count` of the *DiffBind* package. By passing an extra parameter, `summits=250`, through to `dba.count`, the peaks will include 250 basepairs upstream and downstream of the peak summit.

```
> data(blacklist_hg19)
> tamoxifen = ChIPQC(samples, consensus=TRUE, bCount=TRUE, summits=250,
+                    annotation="hg19", chromosomes="chr18",
+                    blacklist = blacklist.hg19)
> tamoxifen
```

Note that only the sample sheet actually needed to be specified; by default, if the `annotation` parameter is missing, it will default to "hg19", while a missing `chromosomes` parameter will result in the first chromosome that has peak being examined (in this case there re only peaks for "chr18", so that would be the default), and if the "hg19" is used, and the blacklist is missing, it will default to the one supplied with the package.

For the purposes of this vignette, we can load a pre-computed version of this object:

```
> data(tamoxifen_QC)
> tamoxifen
```

```
Samples: 16 : BT4741 BT4742 ... TAMRc ZR75c
       Tissue  Factor  Condition  Treatment Replicate Peaks
BT4741  BT474      ER  Resistant Full-Media         1  2844
BT4742  BT474      ER  Resistant Full-Media         2  2844
MCF71    MCF7      ER Responsive Full-Media         1  2844
MCF72    MCF7      ER Responsive Full-Media         2  2844
MCF73    MCF7      ER Responsive Full-Media         3  2844
T47D1    T47D      ER Responsive Full-Media         1  2844
T47D2    T47D      ER Responsive Full-Media         2  2844
TAMR1    MCF7      ER  Resistant Full-Media         1  2844
TAMR2    MCF7      ER  Resistant Full-Media         2  2844
ZR751    ZR75      ER Responsive Full-Media         1  2844
ZR752    ZR75      ER Responsive Full-Media         2  2844
BT474c  BT474 Control  Resistant Full-Media        c1  2844
MCF7c    MCF7 Control Responsive Full-Media        c2  2844
T47Dc    T47D Control Responsive Full-Media        c3  2844
TAMRc    MCF7 Control  Resistant Full-Media        c4  2844
ZR75c    ZR75 Control Responsive Full-Media        c5  2844
         Reads Map% Filt%  Dup% ReadL FragL RelCC  SSD  RiP% RiBL%
BT4741  776353  100  15.8  8.42    28   153 2.110 1.88 14.90  1.75
BT4742  782419  100  15.1 10.30    28   147 2.010 1.63 13.80  1.68
MCF71   438994  100  20.9 21.30    28   134 2.300 2.52 26.50  1.70
MCF72   465700  100  20.8  4.84    28   155 2.090 1.65 16.10  2.17
MCF73   577273  100  19.1 10.30    28   153 2.460 2.18 21.80  1.83
T47D1   507492  100  21.1  7.86    28   153 1.520 1.59  9.62  2.24
T47D2  1831766  100  19.3  9.56    28   162 1.530 2.30  5.70  2.26
TAMR1   747610  100  17.4 15.50    28   151 2.490 2.15 19.30  2.05
TAMR2   728601  100  18.4  5.84    28   149 1.850 1.45 12.10  1.56
ZR751   804427  100  12.0 15.80    28   158 2.950 3.52 30.70  1.39
ZR752  2918549  100  11.6 23.80    28   160 3.140 4.45 20.70  1.22
BT474c  598010  100  18.1  3.30    28   105 0.202 1.14  2.98  1.72
MCF7c   485192  100  26.3  1.70    28   109 0.108 1.40  2.65  2.47
T47Dc   400396  100  44.2 31.60    36   196 0.268 6.02  1.18  8.56
TAMRc   779102  100  19.3  6.16    28   110 0.267 1.38  2.21  2.06
ZR75c  1023987  100  26.2 20.10    36   220 0.573 5.36  1.50  5.35
```

3.3 Quality metrics summary

Looking at the output, the first line indicates how many samples there are (16, including 11 ChIPs and 5 controls). The next section shows the metadata for the 11 ChIP samples. All samples have the same number of peaks as a consensus set (derived from merging overlapping peaks, and keeping peaks that were identified in at least two samples) was used.

The final section (also retrievable as a *matrix* by calling `QCmetrics(tamoxifen)`) shows a summary of the main QC metrics. These include the total number of reads in the bam file for each sample, the percentage of these that were successfully mapped (aligned), and the percentage of reads that were filtered out due to having a mapping quality score less than or equal to the `mapQCth` (in this case, 15). The percentage of reads that map to the exact

position in the genome as at least one other read is then reported. In this example, we can see that three of the controls have very low duplication rates, with the `T47D` and `ZR75` having duplication rates above 20%. This suggest it owuld be reasonable to filter duplciates from those two control. The ChIPs show substantial variance in their duplication rates, several above 10%. Higher duplication rates are expected in the ChIPs, as they should have regions of very high enrichment, which will correctly include fragments that originate at the same location. This is why we expect to see the controls samples exhibiting lower duplication rates ($<$5%) with the ChIP samples having higher rates (15%-20%), but not too high ($>$50%). A very high duplication rate ($>$50%) can result from a sample that has been biased by uneven PCR duplication of some fragments. This example shows some variance in duplication rates amongst the samples, including between replicates of the same condition. For example the third `MCF7` replicate has a duplication rate double that of the first replicat (although neither rate is so high as to indicate a massive PCR duplication issue). The second replicate has a very low rate, which should draw our attention to other metrics that indicate peak enrichment, such as SSD, Retlative CC, and the peak profile plot.

Next the read length, derived from the data in the bam files, is reported, along with the estimated mean fragment length. The fragment length is estimated from the data by systematically shifting the reads on each strand towards each other until the highest degree of cross-coverage is obtained. The `RelativeCC` metric is calculated by comparing the maximal cross coverage peak (at the shift size corresponding to he fragment length) to the cross coverage at a shift size corresponding to the read length, with higher scores (generally 2 or greater) indicating good enrichment. In this example, the `RelativeCC` scores for the controls are very low, as expected, while the scores for the ChIPs are all greater than 1.5, with most above 2, indicating good enrichment and reliable fragment length estimation.

The SSD value is another indication of evidence of enrichment. It is computed by looking at the density of positions with different pileup values. An enriched sample typically has a range of pileup values, with a high standard deviation, so a higher SSD is more indicative of better enrichment. SSD values close to 1 are generally correlated of poorly enriched samples, while successful ChIPs can expect values around 1.5, with highly enriched samples having SSD values of 2 or higher. In this example, the `RelativeCC` scores for the ChIPs are greater than 1.5, with several above 2, while most of the controls are closer to 1, indicating background sampling without clear peaks where reads along the two strands converge in peaks. The two controls with high duplication rates also have high SSD scores, probably driven by false peaks driven by PCR duplication. Several of the plots below, such as the coverage histogram, cross-coverage,a nd peak profiles show that the signals for these controls do not show meaningful enrichment when duplicates are removed.

The final two metrics report the percentage of reads in different regions of interest. The first reports the percentage of reads that overlap called peaks (also known as FRIP). 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 15% or higher generally reflect successful enrichment. In this case,where a consensus peak set was used (instead of peak sets unique to each sample), samples that originally had fewer peaks called can be expected to have lower RiP values (for example the `T47D` samples). Use of the consensus set does allow enrichment in the controls to be directly compared; in this example, the controls have much lower RiP% values, indicating that most of the the fragment sequenced in them were not located at peak sites.

The final measure report the percentage of reads that overlapped blacklisted regions (RiBL).

3.4 Generating a summary HTML report

As in the previous example, a summary HTML report can be generated for this example experiment by invoking `ChIPQCreport`. The only difference is that, in this case, the sample groups should be derived using the `Tissue` and `Condition` metadata values, as follows:

```
> ChIPQCreport(tamoxifen,facetBy=c("Tissue","Condition"))
```

The resulting report is not included in this vignette due to space limitations, but you are invited to generate it yourself and have a look. It is also available for examination at http://ChIPQC.starkhome.com/Reports/tamoxifen/ChIPQC.html.

3.5 Plotting QC metrics for experimental sample groups

As an alternative to generating a complete summary report, plotting methods are provided to generate specific plots, as follows:

3.5.1 Plotting Coverage Histograms

The coverage histogram plot is generated as follows:

```
> plotCoverageHist(tamoxifen,facetBy=c("Tissue","Condition"))
```

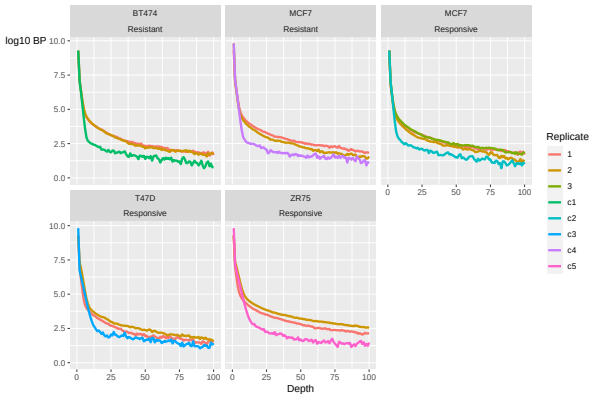

Figure 1: Example Coverage Histogram plot. Generated by `plotCoverageHist(tamoxifen,facetBy=c("Tissue","Condition"))`

As shown in Figure **??**, this plots the distribution of pileup values at each basepair. The first thing to looks for is that the controls are "below" the ChIPs in the plots. Enriched ChIP samples will tail off less quickly than input controls consisting of background reads, indicating positions with high pileup values (ultimately corresponding to peaks).

3.5.2 Plotting Cross-Coverage

A cross-coverage plot can be generated as follows:

```
> plotCC(tamoxifen,facetBy=c("Tissue","Condition"))
```

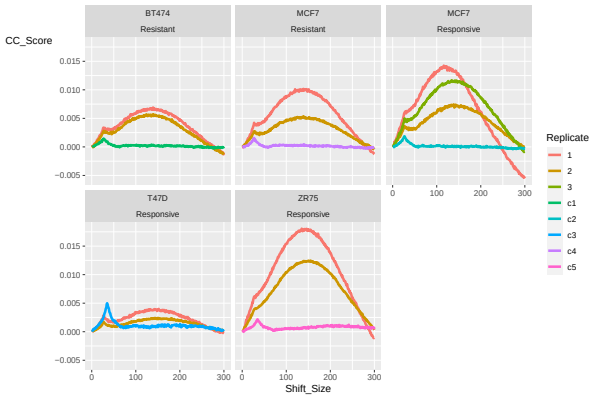

Figure 2: Example Cross-coverage plot. Generated by `plotCC(tamoxifen,facetBy=c("Tissue","Condition"))`

As shown in Figure **??**, this plots the cross-coverage values over a range of shift sizes. The region up to the read length is highlighted, and a small peak in cross-coverage is expected to be found here. Notice that the controls do not have another higher peak, while ChIP samples with good enrichment have another peak in cross-coverage value at the fragment length, as shifting the reads on both strand should increase coverage at peak sites.

3.5.3 Plotting Relative Enrichment of reads in Genomic Intervals

A heatmap plot showing relative enrichment of reads around annotated genomic features can be generated as follows:

```
> plotRegi(tamoxifen,facetBy=c("Tissue","Condition"))
```

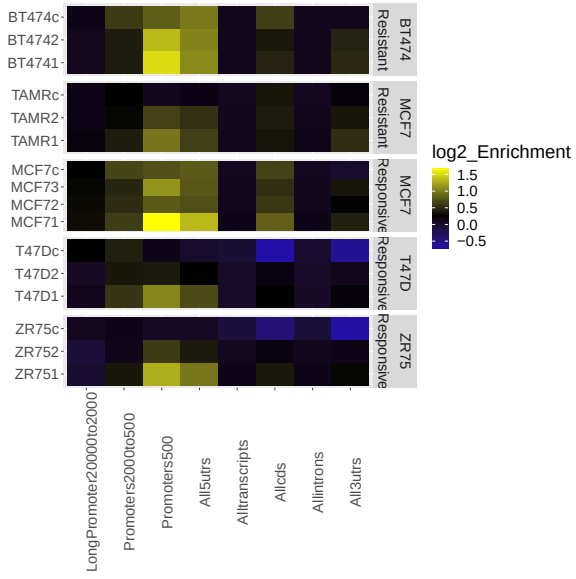

Figure 3: Example Relative Enrichment of Genomic features heatmap plot. Generated by `plotRegi(tamoxifen,facetBy=c("Tissue","Condition"))`

As shown in Figure **??**, samples with more reads than expected around certain features can be seen. Note that these genomic features tend to be enriched (and background genomic DNA depleted) in ChIP and control samples, as it is more likely that the chromatin is open near coding genes, and hence fragmentation can occur there more frequently.

3.5.4 Plotting Peak Profiles

Plots of composite peak profile can be generated as follows:

```
> plotPeakProfile(tamoxifen,facetBy=c("Tissue","Condition"))
```

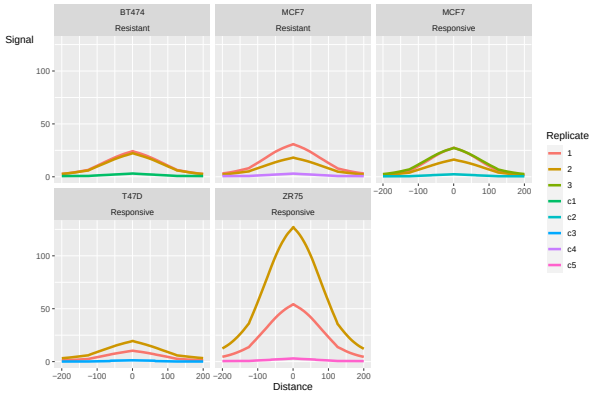

Figure 4: Example Peak Profile plot. Generated by `plotPeakProfile(tamoxifen,facetBy=c("Tissue","Condition"))`

As shown in Figure **??**, each peak is centered on its summit (point of highest pileup after extending the reads to the calculated fragment length), and the pileup values at bases in a window upstream and downstream of the summits is computed and averaged for all peaks in the sample. Good ChIPs will show distinctive patterns of enrichment in these peaks, while associated controls will be relatively flat. The exact shape depends on the specific protein being ChIPed and the biological conditions. This example shows a profile for a transcription factor binding directly to the DNA.

3.5.5 Plotting Reads overlapping Peaks and the Blacklist

Barplots of the relative number of reads that overlap peaks vs. background reads, as well and the proportion of reads overlapping blacklisted regions, can be generated as follows:

```
> plotRap(tamoxifen,facetBy=c("Tissue","Condition"))
```

```
> plotFribl(tamoxifen,facetBy=c("Tissue","Condition"))
```

These plots are shown for the example data set in Figures **??**. and **??**.

3.5.6 Plotting Sample Clustering

It can be useful when assessing the overall success of an experiment to see how the samples cluster. Two plots to aid in this can be generated as follows:

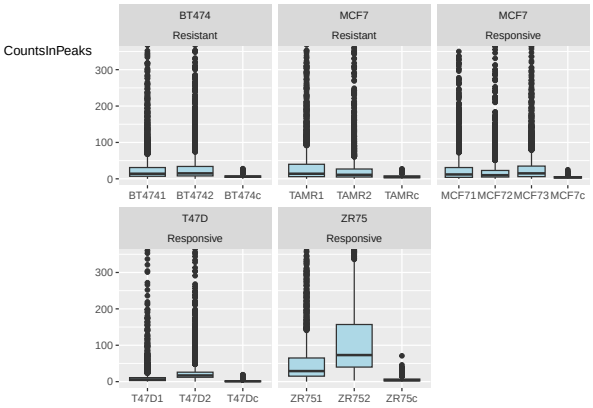

Figure 5: Example Reads overlapping Peaks plots. Generated by `plotRap(tamoxifen,facetBy=c("Tissue","Condition"))`

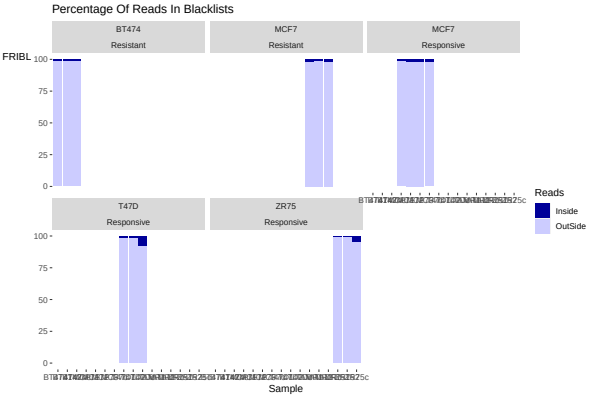

Figure 6: Example Reads overlapping Blacklist plots. Generated by `plotFribl(tamoxifen,facetBy=c("Tissue","Condition"))`

```
> plotCorHeatmap(tamoxifen,attributes=c("Tissue","Factor","Condition","Replicate"))
```

```
> plotPrincomp(tamoxifen,attributes=c("Tissue","Condition"))
```

Figure **??** shows the clustering in a correlation heatmap, where the scores for every peak are computed for each sample, and then scores for each pair of samples are used to calculate a correlation value. In this example, the controls form a distinct cluster, while the replicates each cluster together.

Additional insight can be gained from viewing the results of a principal components analysis using the peak scores. Figure **??** plots each sample along the first two components. The replicates, each plotted with the same color, mostly plot close to each other. The two controls, with very low peak scores (as evidenced by their low RiP% values), end up in almost the exact same point (with the yellow dot superimposed over a red dot) along the first component to the right, while the `T47D` samples, which are the least enriched of the ChIPs (lower RiP%), separate in the second component, occupying the top portion (green dots).

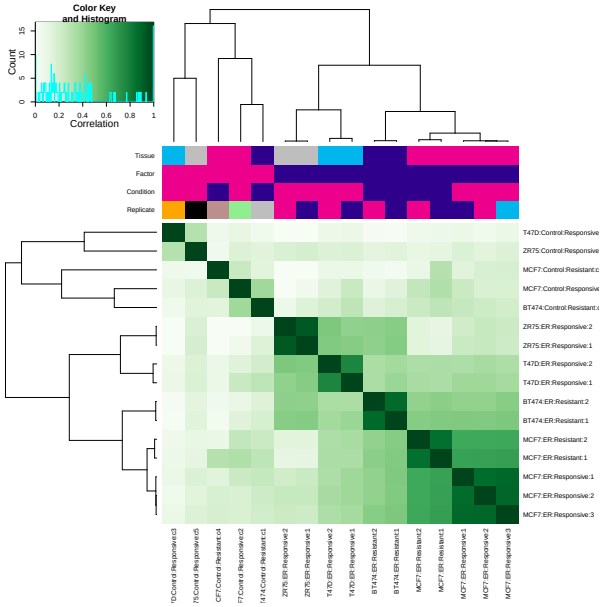

Figure 7: Example Correlation Heatmap with Clustering. Generated by `plotCorHeatmap(tamoxifen,attributes=c("Tissue","Condition"))`

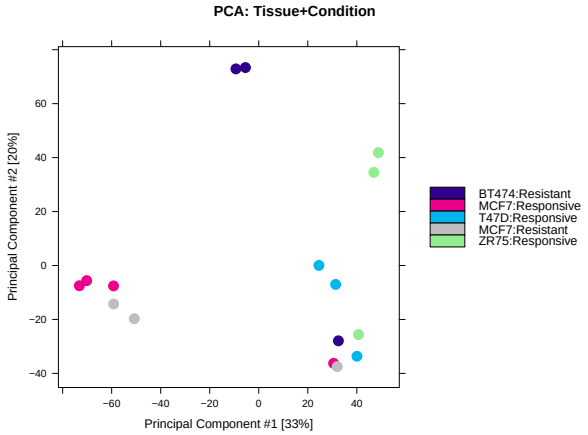

Figure 8: Example Principal Components plot. Generated by `plotPrincomp(tamoxifen,attributes=c("Tissue","Condition"))`

4 Example 3: Single sample assessment

While the vignette has focused on using *ChIPQC* to assess experiment consisting of multiple ChIP and control samples, it can also be used to examine samples individually without requiring the full experimental structure. This section briefly describes how to work with individual samples.

4.1 Constructing a `ChIPQsample` object and retrieving QC metrics

Given only a bam file (or a set of aligned reads in a *GappedAlignemnt* object), the QC metrics can be easily calculated (note the peak file is optional, but if missing, no peak-related metrics can be calculated):

```
> CTCF1 = ChIPQCsample("reads/SRR568129.bam", peaks="peaks/SRR568129_chr22_peaks.bed")
```

As the source data is not included with this vignette, the same effect can be simulated by retrieving the sample object we would have generated from the example experiment:

```
> CTCF1 = QCsample(exampleExp,1)
```

This returns a *ChIPQCsample* object, that can be examined as follows:

```
> CTCF1

                        ProportionOfCounts
Peaks                              0.42345591
BlackList                          0.01807147
LongPromoter20000to2000            0.39820917
Promoters2000to500                 0.05128542
Promoters500                       0.02985514
All5utrs                           0.01987424
Alltranscripts                     0.77778972
Allcds                             0.03430038
Allintrons                         0.71927332
All3utrs                           0.03180118
GRanges object with 1118 ranges and 2 metadata columns:
         seqnames            ranges strand |    Counts bedRangesSummitsTemp
            <Rle>         <IRanges>  <Rle> | <integer>            <numeric>
     [1]    chr22 16084142-16084556      * |        29             16084371
     [2]    chr22 16156879-16157272      * |        29             16157010
     [3]    chr22 16166244-16166805      * |       464             16166577
     [4]    chr22 16185962-16186461      * |        36             16186189
     [5]    chr22 16192141-16193098      * |        65             16192792
     ...      ...               ...    ... .       ...                  ...
  [1114]    chr22 51129810-51130449      * |       105             51130147
  [1115]    chr22 51165140-51165330      * |        14             51165225
  [1116]    chr22 51170467-51171033      * |       133             51170738
  [1117]    chr22 51195483-51195886      * |        24             51195608
  [1118]    chr22 51213448-51214119      * |       164             51213729
  -------
  seqinfo: 1 sequence from an unspecified genome; no seqlengths
```

This shows some summary statistics and the *GRanges* object containing the examined peaks.

A simpler way to see the key QC metrics is as follows:

```
> QCmetrics(CTCF1)

    Reads      Map%     Filt%      Dup%     ReadL     FragL     RelCC       SSD
341055.00    100.00     26.30     16.60     28.00    131.00      2.74      2.53
     RiP%     RiBL%
```

```
        31.20      1.33
```

tamThis returns a *vector* containing 10 metrics: the total number of reads in the bam file, the percentage that are aligned to the genome, the percentage that fall below the default mapping quality filter score of 15, the percentage of reads that align to exactly the same starting position as another read, the length of reads, the calculated fragment length, the relative cross-coverage score (based on the cross-coverage score when shifting by the fragment length vs. unshifted), the SSD (based on the standard deviation of the distribution of coverage pileup scores), and the percentage of reads that overlap peaks and blacklisted regions.

4.2 Plotting QC metrics for a sample

Plots can be generated for individual samples. For example, here is the cross-coverage plot:

```
> plotCC(CTCF1)
```

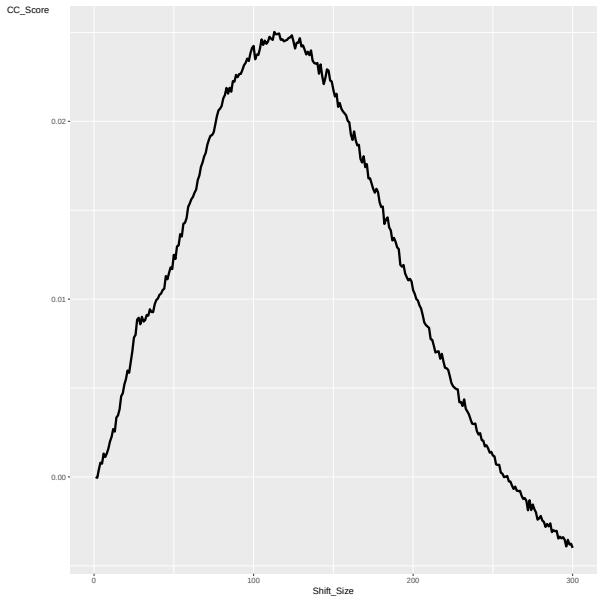

Figure 9: Sample-level cross-coverage plot. Generated by `figSampCC(CTCF)`

shown as Figure **??**

4.3 Combining multiple samples into an Experiment

Samples that have been analyzed separately as *ChIPQCsample* objects may be combined into an *ChIPQCexperiment* object, without requiring the QC metrics to be re-computed, by combining them with a list and supplying a sample sheet with the experimental metadata.

For example, suppose the samples in the tamoxifen resistance example had been assessed separately and been combined in a list. We can get equivalent of this using the `QCsample` method on the experiment:

```
> sampleList = QCsample(tamoxifen)
> class(sampleList)
```

```
[1] "list"

> length(sampleList)

[1] 16

> names(sampleList)

 [1] "BT4741" "BT4742" "MCF71"  "MCF72"  "MCF73"  "T47D1"  "T47D2"  "TAMR1"
 [9] "TAMR2"  "ZR751"  "ZR752"  "BT474c" "MCF7c"  "T47Dc"  "TAMRc"  "ZR75c"

> class(sampleList[[1]])

[1] "ChIPQCsample"
attr(,"package")
[1] "ChIPQC"
```

Suppose we also have constructed a matching samplesheet as a *data.frame*:

```
> sampleSheet = read.csv(file.path(system.file("extdata", package="ChIPQC"),
+                                          "tamoxifenQC.csv"))
> sampleSheet

   SampleID Tissue Factor  Condition  Treatment Replicate
1    BT4741  BT474     ER  Resistant Full-Media         1
2    BT4742  BT474     ER  Resistant Full-Media         2
3     MCF71   MCF7     ER Responsive Full-Media         1
4     MCF72   MCF7     ER Responsive Full-Media         2
5     MCF73   MCF7     ER Responsive Full-Media         3
6     T47D1   T47D     ER Responsive Full-Media         1
7     T47D2   T47D     ER Responsive Full-Media         2
8     TAMR1   MCF7     ER  Resistant Full-Media         1
9     TAMR2   MCF7     ER  Resistant Full-Media         2
10    ZR751   ZR75     ER Responsive Full-Media         1
11    ZR752   ZR75     ER Responsive Full-Media         2
                     bamReads ControlID                  bamControl
1  reads/Chr18_BT474_ER_1.bam    BT474c reads/Chr18_BT474_input.bam
2  reads/Chr18_BT474_ER_2.bam    BT474c reads/Chr18_BT474_input.bam
3   reads/Chr18_MCF7_ER_1.bam     MCF7c  reads/Chr18_MCF7_input.bam
4   reads/Chr18_MCF7_ER_2.bam     MCF7c  reads/Chr18_MCF7_input.bam
5   reads/Chr18_MCF7_ER_3.bam     MCF7c  reads/Chr18_MCF7_input.bam
6   reads/Chr18_T47D_ER_1.bam     T47Dc        reads/T47D_input.bam
7   reads/Chr18_T47D_ER_2.bam     T47Dc        reads/T47D_input.bam
8   reads/Chr18_TAMR_ER_1.bam     TAMRc        reads/TAMR_input.bam
9         reads/TAMR_ER_2.bam     TAMRc        reads/TAMR_input.bam
10  reads/Chr18_ZR75_ER_1.bam     ZR75c        reads/ZR75_input.bam
11  reads/Chr18_ZR75_ER_2.bam     ZR75c        reads/ZR75_input.bam
                     Peaks PeakCaller
1  peaks/BT474_ER_1.bed.gz        bed
2  peaks/BT474_ER_2.bed.gz        bed
3   peaks/MCF7_ER_1.bed.gz        bed
4   peaks/MCF7_ER_2.bed.gz        bed
5   peaks/MCF7_ER_3.bed.gz        bed
6   peaks/T47D_ER_1.bed.gz        bed
7   peaks/T47D_ER_2.bed.gz        bed
```

```
8   peaks/TAMR_ER_1.bed.gz        bed
9   peaks/TAMR_ER_2.bed.gz        bed
10  peaks/ZR75_ER_1.bed.gz        bed
11  peaks/ZR75_ER_2.bed.gz        bed
```

we can then re-construct the original *ChIPQCexperiment* object without having to re-compute the QC metrics:

```
> tamoxifen = ChIPQC(sampleSheet, samples=sampleList)
> tamoxifen
```

We could also supply a sample sheet that refers to only a subset of the *ChIPQCsample* objects to form a smaller experiment:

```
> BT474s = ChIPQC(sampleSheet[1:2,], samples=sampleList)
```

5 Customising QC plots

ChIPQC allows for some customisation of plots produced by the use of the additional parameters of `facetBy`, `colourBy` and `lineBy`. `facetBy` accepts a character vector corresponding to metadata column names and will group plots according to those supplied. In this example we choose to group by sample names to produce individual plots per sample (Figure **??**).

```
> plotCC(exampleExp,facetBy="Sample")
```

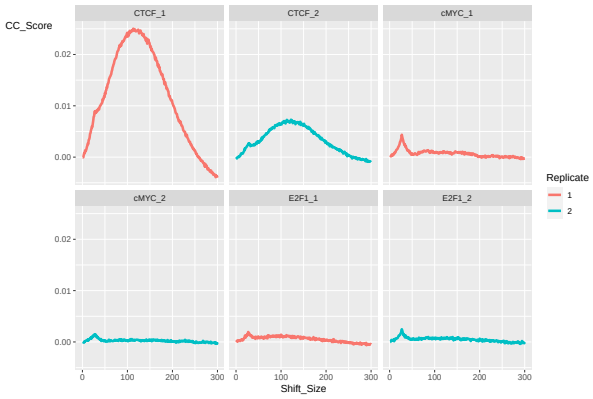

Figure 10: Cross-coverage plot grouped by Sample ID. Generated by `plotCC(exampleExp,facetBy="Sample")`

We can also use the `colourBy` and `lineBy` parameters for the `plotCoverageHist()`, `plotPeakProfile()` and `plotCC()` functions to alter line colours and types used. In Figure **??**) we colour by factor and set the line type to be by Tissue.

```
> plotCC(exampleExp,facetBy="Sample",colourBy="Factor",lineBy="Tissue")
```

In addition to these options, extra metadata information can be passed to the plotting functions and this used by `colourBy`, `lineBy` and `facetBy`. We illustrate this by showing thr relationship between SSD and cross-coverage scores by supplying the `QCmetrics` as additional metadata using the `addMetaData` parameter.

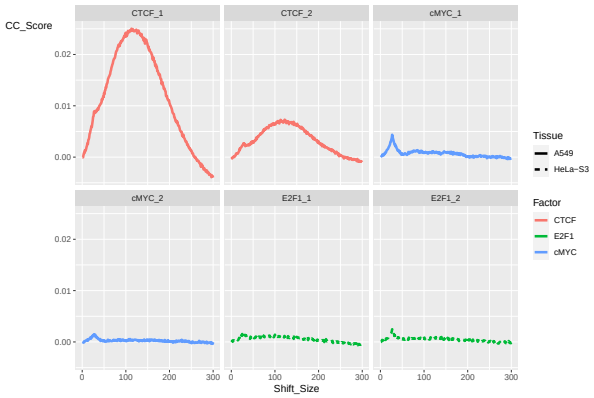

Figure 11: Cross-coverage plot grouped by Sample ID. Generated by `plotCC(exampleExp,facetBy="Sample",colourBy="Factor",lineBy="Tissue")`

```
> extraMetadata <- data.frame(Sample = rownames(QCmetrics(exampleExp)),
+ FRiBL = QCmetrics(exampleExp)[,("RiBL%")],
+ SSD =QCmetrics(exampleExp)[,("SSD")]
+ )
```

```
> plotCC(exampleExp,facetBy="Sample",lineBy="Tissue",addMetaData=extraMetadata,colourBy="SSD")
>
```

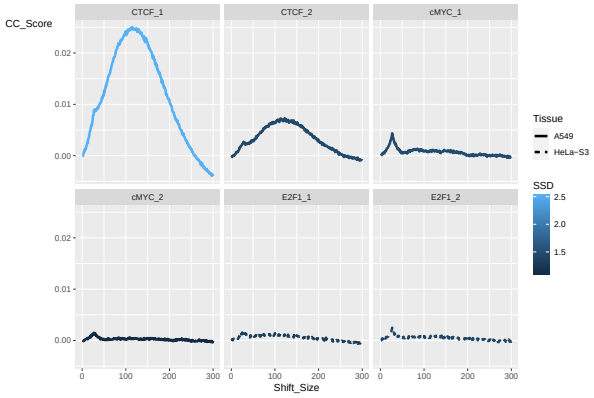

Figure 12: Cross-coverage plot grouped by Sample ID. Generated by `plotCC(exampleExp,facetBy="Sample",lineBy="Tissue",addMetaData=extraMetadata,colourBy="SSD")`

ChIPQC uses the *ggplot2* framework to make plots and the resulting `gg` object can be further customised after the production of the plot. Additional *ggplot2* aesthetics can be added and this instance we change the colour scale used.

```
> plotCC(exampleExp,facetBy="Sample",lineBy="Tissue",addMetaData=extraMetadata,colourBy="SSD")
> + scale_color_gradient2(high="red",mid="black",low="white",midpoint=1.5)
```

6 Using *ChIPQC* and *DiffBind* together

ChIPQC and *DiffBind* are both packages that help manage and analyze ChIP-seq experiments, and are designed to be used together.

If you already have a project in *DiffBind*, the main constructor `ChIPQC` can accept a *DBA* object in place of the sample sheet.

Once a *ChIPQCexperiment* object has been constructed, it can be used in place of a *DBA* object in most calls to *DiffBind*. All plotting, counting, and analysis functions are available from *DiffBind*.

It is also possible to extract a *DBA* object from a *ChIPQCexperiment* object using the `QCdba` method. The resulting *DBA* object can be used in *DiffBind* without restriction, although neither it nor *DBA* objects based on it can be re-attached to the original *ChIPQCexperiment* object (although they can be used in lieu of a sample sheet when creating a new one.)

In a typical workflow, the first step would be to run a *ChIPQC* analysis before peak calling to assess library quality and establish what filtering should be done at the read level (mapping quality, duplicates, and blacklists). Next peaks would be called externally, and read into a new *ChIPQCexperiment* object to assess peak-based metrics, such as FRIP, peak profiles, and clustering.

At this point, *DiffBind* could be used to perform occupancy analysis, derive consensus peak sets, re-count reads to form a binding matrix, and set up contrasts to carry out full differential binding analyses using the *edgeR* and *DESeq2* packages, along with plotting and reporting functions.

7 Acknowledgements

We would like to acknowledge our co-authors, Wei Liu and Ines de Santiago, as well as Gordon Brown and everyone in the Bioinformatics Core at Cancer Research UK's Cambridge Institute at the University of Cambridge.

8 Session Info

```
> toLatex(sessionInfo())
```

- R version 4.3.1 (2023-06-16 ucrt), `x86_64-w64-mingw32`
- Locale: `LC_COLLATE=C`, `LC_CTYPE=English_United States.utf8`, `LC_MONETARY=English_United States.utf8`, `LC_NUMERIC=C`, `LC_TIME=English_United States.utf8`
- Time zone: `America/New_York`
- TZcode source: `internal`
- Running under: `Windows Server 2022 x64 (build 20348)`
- Matrix products: default
- Base packages: base, datasets, grDevices, graphics, methods, stats, stats4, utils

- Other packages: Biobase 2.60.0, BiocGenerics 0.46.0, BiocParallel 1.34.2, ChIPQC 1.36.1, DiffBind 3.10.1, GenomeInfoDb 1.36.2, GenomicRanges 1.52.0, IRanges 2.34.1, MatrixGenerics 1.12.3, S4Vectors 0.38.1, SummarizedExperiment 1.30.2, ggplot2 3.4.3, matrixStats 1.0.0
- Loaded via a namespace (and not attached): AnnotationDbi 1.62.2, BSgenome 1.68.0, BiocFileCache 2.8.0, BiocIO 1.10.0, BiocManager 1.30.22, BiocStyle 2.28.0, Biostrings 2.68.1, DBI 1.1.3, DelayedArray 0.26.7, GenomeInfoDbData 1.2.10, GenomicAlignments 1.36.0, GenomicFeatures 1.52.2, GreyListChIP 1.32.0, KEGGREST 1.40.0, KernSmooth 2.23-22, MASS 7.3-60, Matrix 1.6-1, Nozzle.R1 1.1-1.1, R6 2.5.1, RColorBrewer 1.1-3, RCurl 1.98-1.12, RSQLite 2.3.1, Rcpp 1.0.11, Rsamtools 2.16.0, S4Arrays 1.0.5, SQUAREM 2021.1, ShortRead 1.58.0, TxDb.Celegans.UCSC.ce6.ensGene 3.2.2, TxDb.Dmelanogaster.UCSC.dm3.ensGene 3.2.2, TxDb.Hsapiens.UCSC.hg18.knownGene 3.2.2, TxDb.Hsapiens.UCSC.hg19.knownGene 3.2.2, TxDb.Mmusculus.UCSC.mm10.knownGene 3.10.0, TxDb.Mmusculus.UCSC.mm9.knownGene 3.2.2, TxDb.Rnorvegicus.UCSC.rn4.ensGene 3.2.2, XML 3.99-0.14, XVector 0.40.0, abind 1.4-5, amap 0.8-19, apeglm 1.22.1, ashr 2.2-63, bbmle 1.0.25, bdsmatrix 1.3-6, biomaRt 2.56.1, bit 4.0.5, bit64 4.0.5, bitops 1.0-7, blob 1.2.4, caTools 1.18.2, cachem 1.0.8, chipseq 1.50.0, cli 3.6.1, coda 0.19-4, codetools 0.2-19, colorspace 2.1-0, compiler 4.3.1, crayon 1.5.2, curl 5.0.2, dbplyr 2.3.3, deldir 1.0-9, digest 0.6.33, dplyr 1.1.2, emdbook 1.3.13, evaluate 0.21, fansi 1.0.4, farver 2.1.1, fastmap 1.1.1, filelock 1.0.2, generics 0.1.3, ggrepel 0.9.3, glue 1.6.2, gplots 3.1.3, grid 4.3.1, gtable 0.3.4, gtools 3.9.4, hms 1.1.3, htmltools 0.5.6, htmlwidgets 1.6.2, httr 1.4.7, hwriter 1.3.2.1, interp 1.1-4, invgamma 1.1, irlba 2.3.5.1, jpeg 0.1-10, knitr 1.43, labeling 0.4.2, lattice 0.21-8, latticeExtra 0.6-30, lifecycle 1.0.3, limma 3.56.2, locfit 1.5-9.8, magrittr 2.0.3, memoise 2.0.1, mixsqp 0.3-48, munsell 0.5.0, mvtnorm 1.2-3, numDeriv 2016.8-1.1, parallel 4.3.1, pillar 1.9.0, pkgconfig 2.0.3, plyr 1.8.8, png 0.1-8, prettyunits 1.1.1, progress 1.2.2, rappdirs 0.3.3, reshape2 1.4.4, restfulr 0.0.15, rjson 0.2.21, rlang 1.1.1, rmarkdown 2.24, rtracklayer 1.60.1, scales 1.2.1, stringi 1.7.12, stringr 1.5.0, systemPipeR 2.6.3, tibble 3.2.1, tidyselect 1.2.0, tools 4.3.1, truncnorm 1.0-9, utf8 1.2.3, vctrs 0.6.3, withr 2.5.0, xfun 0.40, xml2 1.3.5, yaml 2.3.7, zlibbioc 1.46.0