DiffBind: Differential binding analysis of ChIP-Seq peak data

Rory Stark
rory.stark@cruk.cam.ac.uk

Gord Brown
gdbzork@gmail.com

University of Cambridge
Cancer Research UK - Cambridge Institute

Edited: May 17, 2016; Compiled: September 26, 2016

Contents

1 Introduction

This document offers an introduction and overview of the *R* Bioconductor package *DiffBind* , which provides functions for processing ChIP-Seq data enriched for genomic loci where specific protein/DNA binding occurs, including peak sets identified by ChIP-Seq peak callers and aligned sequence read datasets. It is designed to work with multiple peak sets simultaneously, representing different ChIP experiments (antibodies, transcription factor and/or histone marks, experimental conditions, replicates) as well as managing the results of multiple peak callers.

The primary emphasis of the package is on identifying sites that are differentially bound between two sample groups. It includes functions to support the processing of peak sets, including overlapping and merging peak sets, counting sequencing reads overlapping intervals in peak sets, and identifying statistically significantly differentially bound sites based on evidence of binding affinity (measured by differences in read densities). To this end it uses statistical routines developed in an RNA-Seq context (primarily the Bioconductor packages *edgeR* and *DESeq2*). Additionally, the package builds on *R*graphics routines to provide a set of standardized plots to aid in binding analysis.

This guide includes a brief overview of the processing flow, followed by four sections of examples: the first focusing on the core task of obtaining differentially bound sites based on affinity data, the second working through the main plotting routines, the third discussing the use of a blocking factor, and the fourth revisiting occupancy data (peak calls) in more detail, as well as comparing the results of an occupancy-based analysis with an affinity-based one. Finally, certain technical aspects of the how these analyses are accomplished are detailed.

2 Processing overview

DiffBind works primarily with peaksets, which are sets of genomic intervals representing candidate protein binding sites. Each interval consists of a chromosome, a start and end position, and usually a score of some type indicating confidence in, or strength of, the peak. Associated with each peakset are metadata relating to the experiment from which the peakset was derived. Additionally, files containing mapped sequencing reads (generally .bam files) can be associated with each peakset (one for the ChIP data, and optionally another representing a control sample).

Generally, processing data with *DiffBind* involves five phases:

1. `Reading in peaksets`: The first step is to read in a set of peaksets and associated metadata. Peaksets are derived either from ChIP-Seq peak callers, such as MACS ([**?**]), or using some other criterion (e.g. genomic windows, or all the promoter regions in a genome). The easiest way to read in peaksets is using a comma-separated value (csv)

sample sheet with one line for each peakset. (Spreadsheets in Excel® format, with a `.xls` or `.xlsx` suffix, are also accepted.) A single experiment can have more than one associated peakset; e.g. if multiple peak callers are used for comparison purposes each sample would have more than one line in the sample sheet. Once the peaksets are read in, a merging function finds all overlapping peaks and derives a single set of unique genomic intervals covering all the supplied peaks (a *consensus peakset* for the experiment).

2. `Occupancy analysis`: Peaksets, especially those generated by peak callers, provide an insight into the potential *occupancy* of the protein being ChIPed for at specific genomic loci. After the peaksets have been loaded, it can be useful to perform some exploratory plotting to determine how these occupancy maps agree with each other, e.g. between experimental replicates (re-doing the ChIP under the same conditions), between different peak callers on the same experiment, and within groups of samples representing a common experimental condition. *DiffBind* provides functions to enable overlaps to be examined, as well as functions to determine how well similar samples cluster together. Beyond quality control, the product of an occupancy analysis may be a *consensus peakset*, representing an overall set of candidate binding sites to be used in further analysis.
3. `Counting reads`: Once a consensus peakset has been derived, *DiffBind* can use the supplied sequence read files to count how many reads overlap each interval for each unique sample. The peaks in the consensus peakset may be re-centered and trimmed based on calculating their summits (point of greatest read overlap) in order to provide more standardized peak intervals. The final result of counting is a *binding affinity matrix* containing a (normalized) read count for each sample at every potential binding site. With this matrix, the samples can be re-clustered using affinity, rather than occupancy, data. The binding affinity matrix is used for QC plotting as well as for subsequent differential analysis.
4. `Differential binding affinity analysis`: The core functionality of *DiffBind* is the differential binding affinity analysis, which enables binding sites to be identified that are statistically significantly differentially bound between sample groups. To accomplish this, first a contrast (or contrasts) is established, dividing the samples into groups to be compared. Next the core analysis routines are executed, by default using *DESeq2* . This will assign a p-value and FDR to each candidate binding site indicating confidence that they are differentially bound.
5. `Plotting and reporting`: Once one or more contrasts have been run, *DiffBind* provides a number of functions for reporting and plotting the results. MA plots give an overview of the results of the analysis, while correlation heatmaps and PCA plots show how the groups cluster based on differentially bound sites. Boxplots show the distribution of reads within differentially bound sites corresponding to whether they gain or lose affinity between the two sample groups. A reporting mechanism enables differentially bound sites to be extracted for further processing, such as annotation, motif, and pathway analyses.

3 Example: Obtaining differentially bound sites

This section offers a quick example of how to use *DiffBind* to identify significantly differentially bound sites using affinity (read count) data.

The dataset for this example consists of ChIPs against the transcription factor ERa using five breast cancer cell lines ([**?**]). Three of these cell lines are responsive to tamoxifen treatment, while two others are resistant to tamoxifen. There are at least two replicates for each of the cell lines, with one cell line having three replicates, for a total of eleven sequenced libraries. Of the five cell lines, two are based on MCF7 cells: the regular tamoxifen responsive line, as well as MCF7 cells specially treated with tamoxifen until a tamoxifen resistant cell line is obtained. For each sample, we have one peakset originally derived using the MACS peak caller ([**?**]), for a total of eleven peaksets. Note that to save space in the package, only data for chromosome 18 is used for the vignette. The metadata and peak data are available in the `extra` subdirectory of the *DiffBind* package directory; you can make this your working directory by entering:

```
> library(DiffBind)
> setwd(system.file("extra", package="DiffBind"))
```

Obtaining the sites significantly differentially bound (DB) between the samples that respond to tamoxifen and those that are resistant can be done in a five-step script:

```
> print(savewd)
> tamoxifen <- dba(sampleSheet="tamoxifen.csv")
```

```
> tamoxifen <- dba.count(tamoxifen)
> tamoxifen <- dba.contrast(tamoxifen)
> tamoxifen <- dba.analyze(tamoxifen)
> tamoxifen.DB <- dba.report(tamoxifen)
```

The following subsections describe these steps in more detail

3.1 Reading in the peaksets

The easiest way to set up an experiment to analyze is with a sample sheet. The sample sheet can be a `dataframe`, or it can be read directly from a `csv` file. Here is the example sample sheet read into a `dataframe` form a `csv` file:

```
> samples <- read.csv(file.path(system.file("extra", package="DiffBind"),
+                                          "tamoxifen.csv"))
> names(samples)
```

```
 [1] "SampleID"   "Tissue"     "Factor"     "Condition"  "Treatment"  "Replicate"
 [7] "bamReads"   "ControlID"  "bamControl" "Peaks"      "PeakCaller"
```

```
> samples
```

```
   SampleID Tissue Factor  Condition  Treatment Replicate                  bamReads
1    BT4741  BT474     ER  Resistant Full-Media         1 reads/Chr18_BT474_ER_1.bam
2    BT4742  BT474     ER  Resistant Full-Media         2 reads/Chr18_BT474_ER_2.bam
3     MCF71   MCF7     ER Responsive Full-Media         1  reads/Chr18_MCF7_ER_1.bam
4     MCF72   MCF7     ER Responsive Full-Media         2  reads/Chr18_MCF7_ER_2.bam
5     MCF73   MCF7     ER Responsive Full-Media         3  reads/Chr18_MCF7_ER_3.bam
6     T47D1   T47D     ER Responsive Full-Media         1  reads/Chr18_T47D_ER_1.bam
7     T47D2   T47D     ER Responsive Full-Media         2  reads/Chr18_T47D_ER_2.bam
8    MCF7r1   MCF7     ER  Resistant Full-Media         1  reads/Chr18_TAMR_ER_1.bam
9    MCF7r2   MCF7     ER  Resistant Full-Media         2        reads/TAMR_ER_2.bam
10    ZR751   ZR75     ER Responsive Full-Media         1  reads/Chr18_ZR75_ER_1.bam
11    ZR752   ZR75     ER Responsive Full-Media         2  reads/Chr18_ZR75_ER_2.bam
   ControlID                 bamControl                   Peaks PeakCaller
1     BT474c reads/Chr18_BT474_input.bam peaks/BT474_ER_1.bed.gz        bed
2     BT474c reads/Chr18_BT474_input.bam peaks/BT474_ER_2.bed.gz        bed
3      MCF7c  reads/Chr18_MCF7_input.bam  peaks/MCF7_ER_1.bed.gz        bed
4      MCF7c  reads/Chr18_MCF7_input.bam  peaks/MCF7_ER_2.bed.gz        bed
5      MCF7c  reads/Chr18_MCF7_input.bam  peaks/MCF7_ER_3.bed.gz        bed
6      T47Dc        reads/T47D_input.bam  peaks/T47D_ER_1.bed.gz        bed
7      T47Dc        reads/T47D_input.bam  peaks/T47D_ER_2.bed.gz        bed
8      TAMRc        reads/TAMR_input.bam  peaks/TAMR_ER_1.bed.gz        bed
9      TAMRc        reads/TAMR_input.bam  peaks/TAMR_ER_2.bed.gz        bed
10     ZR75c        reads/ZR75_input.bam  peaks/ZR75_ER_1.bed.gz        bed
11     ZR75c        reads/ZR75_input.bam  peaks/ZR75_ER_2.bed.gz        bed
```

The peaksets are read in using the following *DiffBind* function:

```
> tamoxifen <- dba(sampleSheet="tamoxifen.csv")
```

The result is a *DBA object*; the metadata associated with this object can be displayed simply as follows:

```
> tamoxifen
```

```
11 Samples, 2845 sites in matrix (3795 total):
      ID Tissue Factor  Condition  Treatment Replicate Caller Intervals
1  BT4741  BT474     ER  Resistant Full-Media         1    bed      1080
2  BT4742  BT474     ER  Resistant Full-Media         2    bed      1122
```

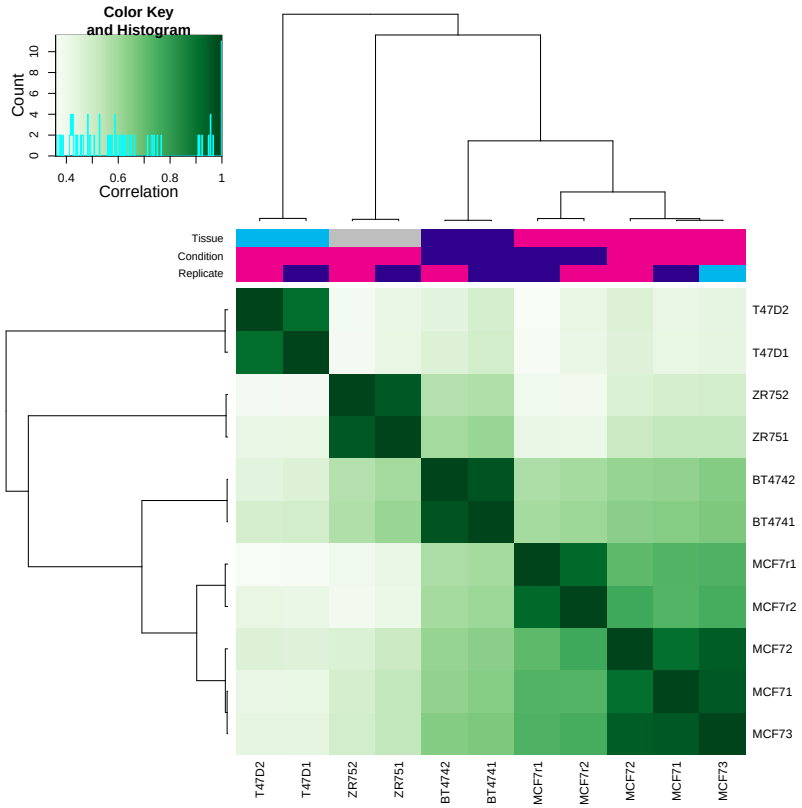

Figure 1: **Correlation heatmap, using occupancy (peak caller score) data.** Generated by: `plot(tamoxifen)`; can also be generated by: `dba.plotHeatmap(tamoxifen)`.

```
3    MCF71    MCF7     ER Responsive Full-Media          1     bed       1556
4    MCF72    MCF7     ER Responsive Full-Media          2     bed       1046
5    MCF73    MCF7     ER Responsive Full-Media          3     bed       1339
6    T47D1    T47D     ER Responsive Full-Media          1     bed        527
7    T47D2    T47D     ER Responsive Full-Media          2     bed        373
8   MCF7r1    MCF7     ER  Resistant Full-Media          1     bed       1438
9   MCF7r2    MCF7     ER  Resistant Full-Media          2     bed        930
10   ZR751    ZR75     ER Responsive Full-Media          1     bed       2346
11   ZR752    ZR75     ER Responsive Full-Media          2     bed       2345
```

This shows how many peaks are in each peakset, as well as (in the first line) the total number of unique peaks after merging overlapping ones (3795), and the dimensions of dba.plotPCA(default binding matrix of 11 samples by the 2845 sites that overlap in at least two of the samples. This object is available for loading using `data(tamoxifen_peaks)`.

Using only this peak caller data, a correlation heatmap can be generated which gives an initial clustering of the samples using the cross-correlations of each row of the binding matrix:

```
> plot(tamoxifen)
```

The resulting plot (Figure **??**) shows that while the replicates for each cell line cluster together appropriately, the cell lines do not cluster into groups corresponding to those that are responsive (MCF7, T47D, and ZR75) vs. those resistant (BT474 and MCF7r) to tamoxifen treatment. It also shows that the two most highly correlated cell lines are the two MCF7-based ones, even though they respond differently to tamoxifen treatment.

3.2 Counting reads

The next step is to calculate a binding matrix with scores based on read counts for every sample (affinity scores), rather than confidence scores for only those peaks called in a specific sample (occupancy scores). These reads are obtained using the `dba.count` function.[1] As this example is based on a transcription factor that binds to the DNA, resulting in "punctate", narrow peaks, it is advisable to use the "summits" option to re-center each peak around the point of greatest enrichment. This keeps the peaks at a consistent width (in this case, with `summits=250`, the peaks will be 500bp, extending 250bp up- and down- stream of the summit):

```
> tamoxifen <- dba.count(tamoxifen, summits=250)
```

If you do not have the raw reads available to you, this object is available loading using `data(tamoxifen_counts)`. After the `dba.count` call, the *DBA object* can be examined:

```
> tamoxifen
```

```
11 Samples, 2845 sites in matrix:
       ID Tissue Factor  Condition  Treatment Replicate Caller Intervals FRiP
1  BT4741  BT474     ER  Resistant Full-Media         1 counts      2845 0.16
2  BT4742  BT474     ER  Resistant Full-Media         2 counts      2845 0.15
3   MCF71   MCF7     ER Responsive Full-Media         1 counts      2845 0.27
4   MCF72   MCF7     ER Responsive Full-Media         2 counts      2845 0.17
5   MCF73   MCF7     ER Responsive Full-Media         3 counts      2845 0.23
6   T47D1   T47D     ER Responsive Full-Media         1 counts      2845 0.10
7   T47D2   T47D     ER Responsive Full-Media         2 counts      2845 0.06
8  MCF7r1   MCF7     ER  Resistant Full-Media         1 counts      2845 0.20
9  MCF7r2   MCF7     ER  Resistant Full-Media         2 counts      2845 0.13
10  ZR751   ZR75     ER Responsive Full-Media         1 counts      2845 0.32
11  ZR752   ZR75     ER Responsive Full-Media         2 counts      2845 0.22
```

This shows that all the samples are using the same, 2845 length consensus peakset. Also, a new column has been added, called `FRiP`, which stands for Fraction of Reads in Peaks. This is the proportion of reads for that sample that overlap a peak in the consensus peakset, and can be used to indicate which samples show more enrichment overall.

```
> plot(tamoxifen)
```

We can also plot a new correlation heatmap based on the affinity scores, seen in Figure **??**. While this shows a slightly different clustering, responsiveness to tamoxifen treatment does not appear to form a basis for clustering when using all of the affinity scores. (Note that the clustering can change based on what scoring metric is used; see Section 4.4 for more details).

3.3 Establishing a contrast

Before running the differential analysis, we need to tell *DiffBind* which cell lines fall in which groups. This is done using the `dba.contrast` function, as follows:

```
> tamoxifen <- dba.contrast(tamoxifen, categories=DBA_CONDITION)
```

The uses the *Condition* metadata (Responsive vs. Resistant) to set up a a contrast with 4 samples in the Resistant group and 7 samples in the Responsive group.[2]

[1] Note that due to space limitations the reads are not shipped with the package. See Section 8 for options to obtain the full dataset. Alternatively, you can get the result of the `dba.count` call by loading the supplied *R*object by invoking `data(tamoxifen_counts)`

[2] This step is actually optional: if the main analysis function `dba.analyze` is invoked with no contrasts established, *DiffBind* will set up a default set of contrasts automatically, which in this case will be the one we are interested in.

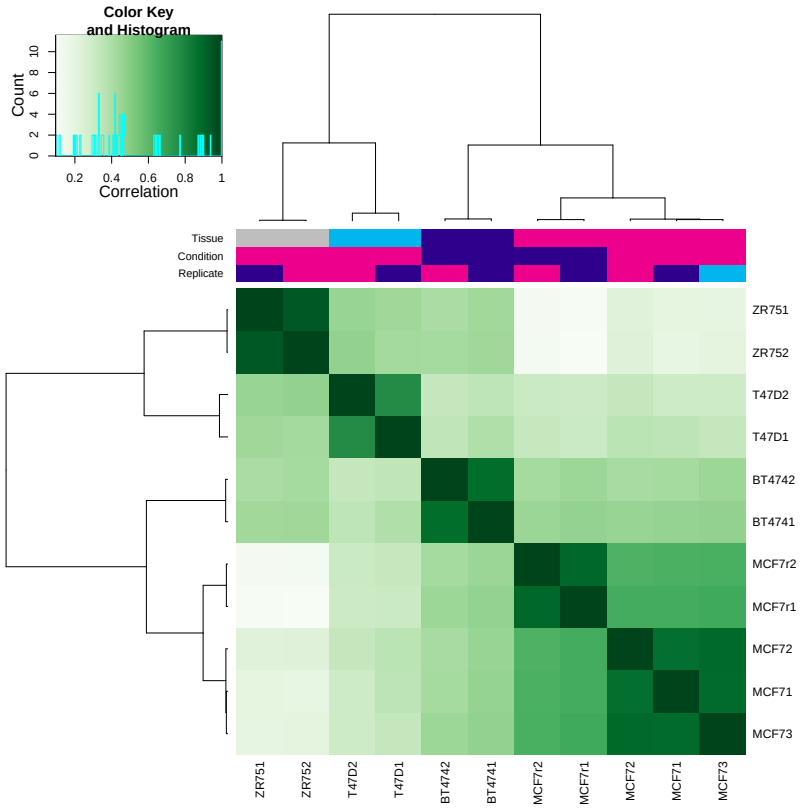

Figure 2: **Correlation heatmap, using affinity (read count) data.** Generated by: `plot(tamoxifen)`; can also be generated by: `dba.plotHeatmap(tamoxifen)`

3.4 Performing the differential analysis

The main differential analysis function is invoked as follows:

```
> tamoxifen <- dba.analyze(tamoxifen)
> tamoxifen
```

```
11 Samples, 2845 sites in matrix:
       ID Tissue Factor  Condition  Treatment Replicate Caller Intervals FRiP
1  BT4741  BT474     ER  Resistant Full-Media         1 counts      2845 0.16
2  BT4742  BT474     ER  Resistant Full-Media         2 counts      2845 0.15
3   MCF71   MCF7     ER Responsive Full-Media         1 counts      2845 0.27
4   MCF72   MCF7     ER Responsive Full-Media         2 counts      2845 0.17
5   MCF73   MCF7     ER Responsive Full-Media         3 counts      2845 0.23
6   T47D1   T47D     ER Responsive Full-Media         1 counts      2845 0.10
7   T47D2   T47D     ER Responsive Full-Media         2 counts      2845 0.06
8  MCF7r1   MCF7     ER  Resistant Full-Media         1 counts      2845 0.20
9  MCF7r2   MCF7     ER  Resistant Full-Media         2 counts      2845 0.13
10  ZR751   ZR75     ER Responsive Full-Media         1 counts      2845 0.32
11  ZR752   ZR75     ER Responsive Full-Media         2 counts      2845 0.22

1 Contrast:
     Group1 Members1     Group2 Members2 DB.DESeq2
```

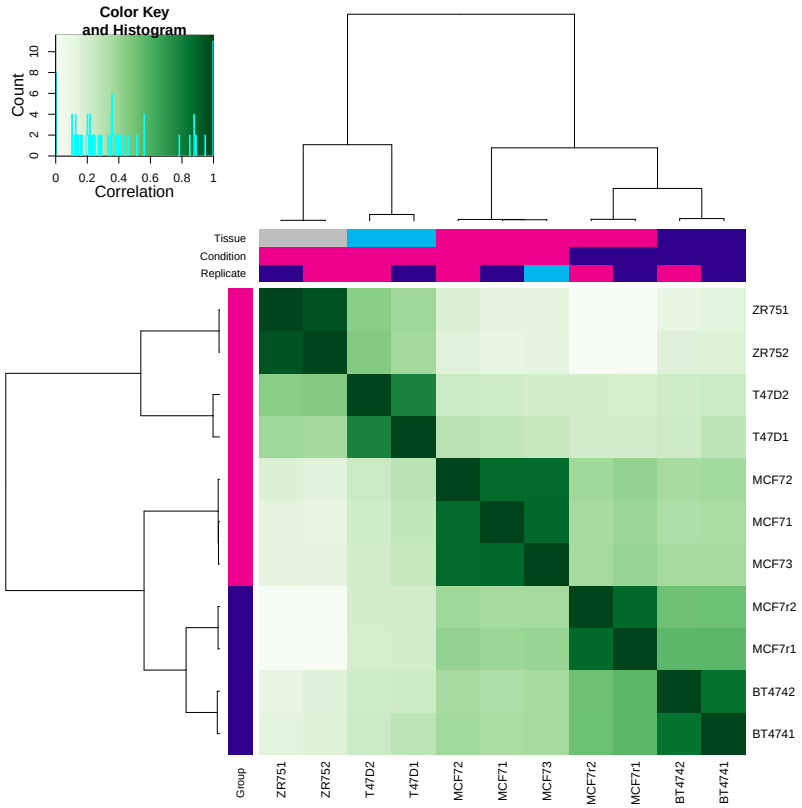

Figure 3: **Correlation heatmap, using only significantly differentially bound sites.** Generated by: `plot(tamoxifen, contrast=1)`; can also be generated by: `dba.plotHeatmap(tamoxifen, contrast=1)`

```
1 Resistant         4 Responsive         7        677
```

This will run an *DESeq2* analysis (see subsequent section discussing the technical details of the analysis) using the default binding matrix. Displaying the resultant `DBA object` shows that 677 of the 2845 sites are identified as being significantly differentially bound (DB) using the default threshold of FDR <= 0.05

A correlation heatmap can be plotted, based on the result of the analysis, as shown in Figure **??**.

```
> plot(tamoxifen, contrast=1)
```

Using only the differentially bound sites, we now see that the four tamoxifen resistant samples (representing two cell lines) cluster together, although the tamoxifen-responsive MCF7 replicates cluster closer to them than to the other tamoxifen responsive samples. Comparing Figure **??**, which uses all 2845 consensus binding sites, with Figure **??**, which uses only the 677 differentially bound sites, demonstrates how the differential binding analysis isolates sites that help distinguish between the Resistant and Responsive sample groups. Note this is plot is not a "result" in the sense that the analysis is selecting for sites that differ between the two conditions, and hence are expected to form clusters representing the conditions. Indeed, the fact that these site do not enable a perfect clustering show the importance of checking the results at this stage.

3.5 Retrieving the differentially bound sites

The final step is to retrieve the differentially bound sites as follows:

```
> tamoxifen.DB <- dba.report(tamoxifen)
```

These are returned as a *GRanges* object, appropriate for downstream processing:

```
> tamoxifen.DB
```

```
GRanges object with 677 ranges and 6 metadata columns:
       seqnames              ranges strand |      Conc Conc_Resistant Conc_Responsive
          <Rle>           <IRanges>  <Rle> | <numeric>      <numeric>       <numeric>
  1291    chr18 [34597700, 34598200]      * |      5.33           0.02            5.97
  2452    chr18 [64490684, 64491184]      * |      6.36           1.39               7
  2571    chr18 [69433116, 69433616]      * |      4.57          -0.79            5.21
  2771    chr18 [74536113, 74536613]      * |      3.93          -0.79            4.57
   976    chr18 [26860992, 26861492]      * |       7.3            3.1            7.92
   ...      ...                 ...    ... .       ...            ...             ...
  1405    chr18 [38482733, 38483233]      * |      3.23           0.99            3.76
  1695    chr18 [45053220, 45053720]      * |      2.77           0.81            3.28
  1650    chr18 [43648626, 43649126]      * |      3.88           2.31            4.34
  1702    chr18 [45489315, 45489815]      * |      1.54          -0.22            2.03
  1506    chr18 [41736699, 41737199]      * |      1.84              0            2.34
           Fold   p-value       FDR
      <numeric> <numeric> <numeric>
  1291     -5.95  1.24e-10  3.21e-07
  2452     -5.61  2.26e-10  3.21e-07
  2571        -6  3.59e-09  3.41e-06
  2771     -5.35  6.56e-09  4.67e-06
   976     -4.82  8.74e-09  4.97e-06
   ...       ...       ...       ...
  1405     -2.77    0.0116    0.0489
  1695     -2.47    0.0117    0.0492
  1650     -2.03    0.0117    0.0492
  1702     -2.25    0.0118    0.0494
  1506     -2.34    0.0118    0.0494
  -------
  seqinfo: 1 sequence from an unspecified genome; no seqlengths
```

The value columns show the mean read concentration over all the samples (the default calculation uses log2 normalized ChIP read counts with control read counts subtracted) and the mean concentration over the first (Resistant) group and second (Responsive) group. The Fold column shows the difference in mean concentrations between the two groups (Conc_Resistant - Conc_Responsive), with a positive value indicating increased binding affinity in the Resistant group and a negative value indicating increased binding affinity in the Responsive group. The final two columns give confidence measures for identifying these sites as differentially bound, with a raw p-value and a multiple testing corrected FDR in the final column.

4 Example: Plotting

Besides the correlation heatmaps automatically generated by the core functions, a number of other plots are available using the affinity data. This sections covers Venn diagrams, MA plots, PCA plots, Boxplots, and Heatmaps.

4.1 Venn diagrams

Venn diagrams are useful for examining overlaps between peaksets, particularly when determining how best to derive consensus peaksets for further analysis. Section 6.2, which discusses consensus peaksets, shows a number of Venn plots

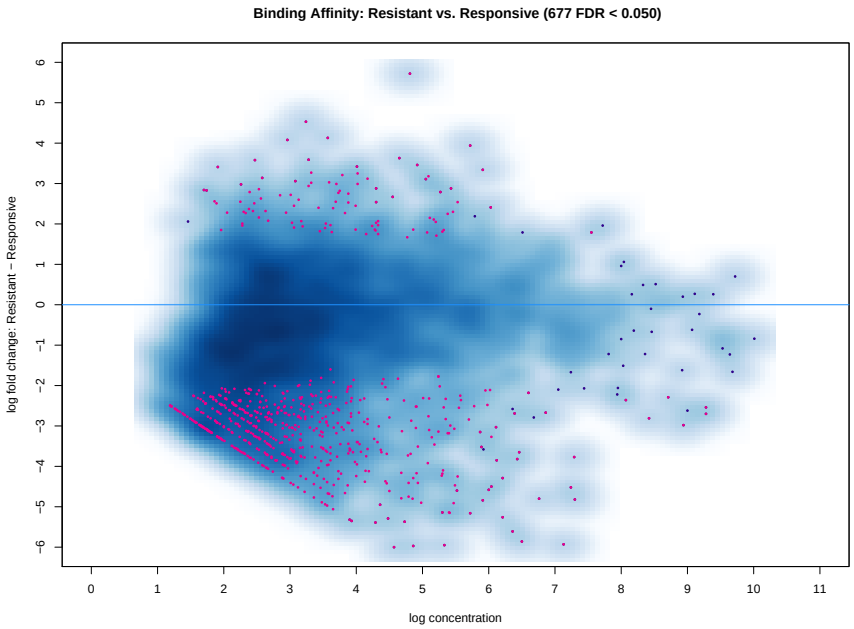

Figure 4: **MA plot of Resistant-Responsive contrast.** Sites identified as significantly differentially bound shown in red. Generated by: `dba.plotMA(tamoxifen)`

in context, and the help page for `dba.plotVenn` has a number of additional examples.

4.2 MA plots

MA plots are a useful way to visualize the effect of normalization on data, as well as seeing which of the datapoints are being identified as differentially bound. An MA plot can be obtained for the resistant-responsive contrast as follows:

```
> data(tamoxifen_analysis)
> dba.plotMA(tamoxifen)
```

The plot is shown in Figure **??**. Each point represents a binding site, with points in red representing sites identified as differentially bound. The plot shows how the differentially bound sites appear to have an absolute log fold difference of at least 2. It also suggests that more ERa binding sites lose binding affinity in the tamoxifen resistant condition than gain binding affinity, as evidenced by more red dots below the center line than are above it. This same data can also be shown with the concentrations of each sample groups plotted against each other plot using `dba.plotMA(tamoxifen, bXY=TRUE)`.

4.3 PCA plots

While the correlation heatmaps already seen are good for showing clustering, plots based on principal components analysis can be used to give a deeper insight into how samples are associated. A PCA plot corresponding to Figure **??**, which includes normalized read counts for all the binding sites, can be obtained as follows:

```
> dba.plotPCA(tamoxifen,DBA_TISSUE,label=DBA_CONDITION)
```

The resulting plot (Figure **??**) shows all the MCF7-derived samples (red) clustering together, with the Responsive and Resistance samples not separable from the Responsive sample in either the first (horizontal) or the second (vertical) components when looking at all the binding sites.

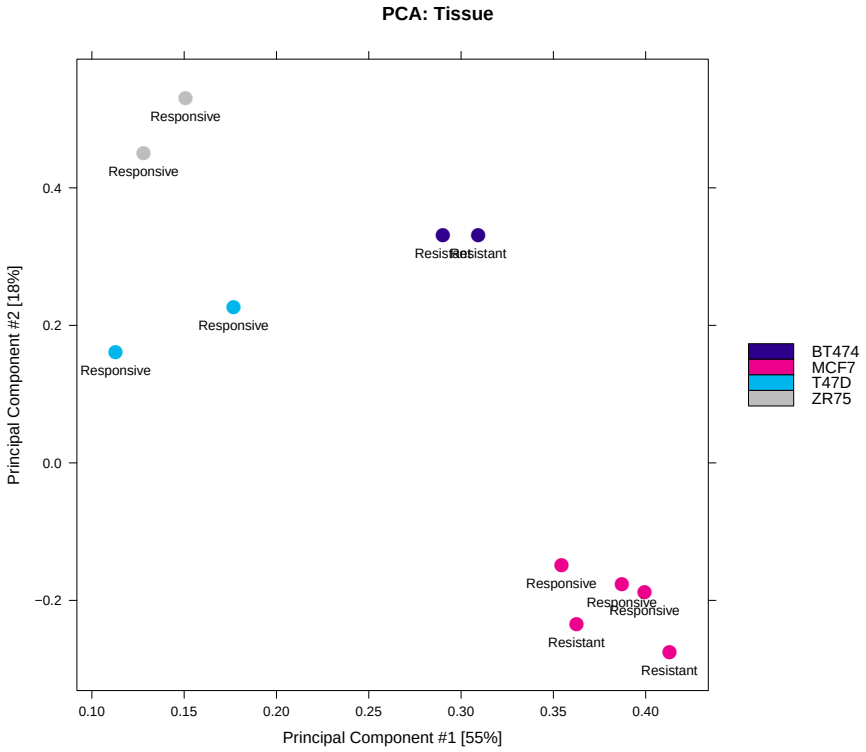

Figure 5: **PCA plot using affinity data for all sites.** Generated by: `dba.plotPCA(tamoxifen,DBA_TISSUE,label=DBA_CONDITION)`

A PCA plot using only the differentially bound sites (corresponding to Figure **??**), using an FDR threshold of 0.05, can be drawn as follows:

```
> dba.plotPCA(tamoxifen, contrast=1,label=DBA_TISSUE)
```

This plot (Figure **??**) shows that the differential analysis identifies sites than can be used to separate the sample groups along the second component, indicating why the hierarchical clustering in shown in Figure **??** is imperfect.

The `dba.plotPCA` function is customizable. For example, if you want to see where the replicates for each of the unique cell lines lies, type `dba.plotPCA(tamoxifen, attributes=c(DBA_TISSUE,DBA_CONDITION),label=DBA_REPLICATE`. If your installation of *R*supports 3D graphics using the *rgl* package, try `dba.plotPCA(tamoxifen, b3D=T)`. Seeing the first three principal components can be a useful exploratory exercise.

4.4 Boxplots

Boxplots provide a way to view how read distributions differ between classes of binding sites. Consider the example, where the 677 differentially bound sites are identified. The MA plot (Figure **??**) shows that these are not distributed evenly between those that increase binding affinity in the Responsive group vs. those that increase binding affinity in the Resistant groups. This can be seen quantitatively using the sites returned in the report:

```
> sum(tamoxifen.DB$Fold<0)
```

```
[1] 580
```

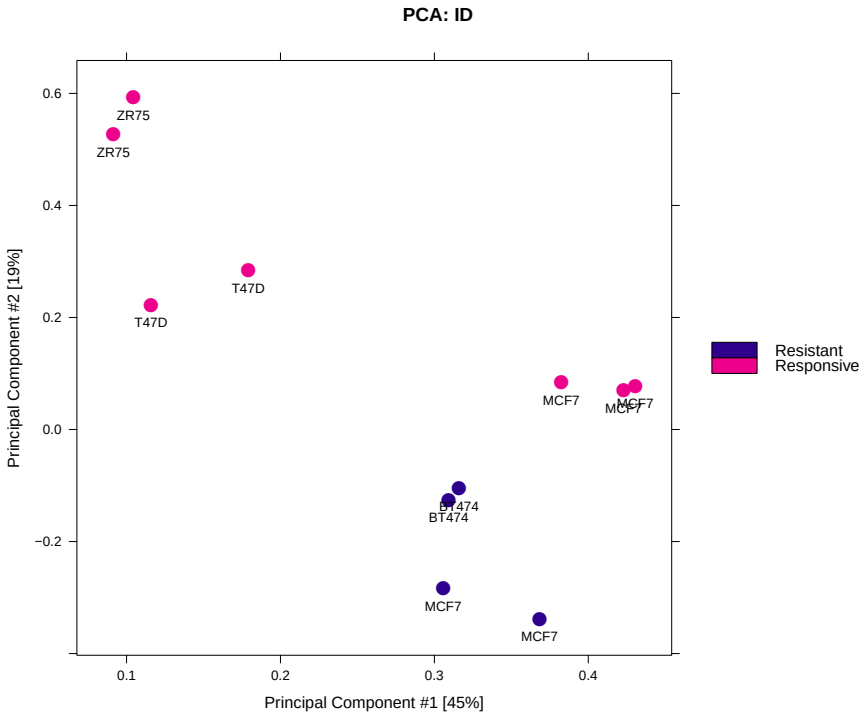

Figure 6: **PCA plot using affinity data for only differentially bound sites.** Generated by: `dba.plotPCA(tamoxifen,contrast=1,label=DBA_TISSUE)`

```
> sum(tamoxifen.DB$Fold>0)
```

```
[1] 97
```

But how are reads distributed amongst the different classes of differentially bound sites and sample groups? These data can be more clearly seen using a boxplot:

```
> pvals <- dba.plotBox(tamoxifen)
```

The default plot (Figure **??**) shows in the first two boxes that amongst differentially bound sites overall, the Responsive samples have a somewhat higher mean read concentration. The next two boxes show the distribution of reads in differentially bound sites that exhibit increased affinity in the Responsive samples, while the final two boxes show the distribution of reads in differentially bound sites that exhibit increased affinity in the Resistant samples.

`dba.plotBox` returns a matrix of p-values (computed using a two-sided Wilcoxon 'Mann-Whitney' test, paired where appropriate) indicating which of these distributions are significantly different from another distribution.

```
> pvals
```

```
               Resistant.DB Responsive.DB Resistant.DB+ Responsive.DB+ Resistant.DB-
Resistant.DB       1.00e+00      8.11e-77      2.50e-05      3.83e-103      2.53e-39
Responsive.DB      8.11e-77      1.00e+00     1.38e-155       9.57e-03      6.45e-18
Resistant.DB+      2.50e-05     1.38e-155      1.00e+00       1.10e-96      6.29e-52
Responsive.DB+    3.83e-103      9.57e-03       1.10e-96       1.00e+00      1.40e-14
Resistant.DB-      2.53e-39      6.45e-18       6.29e-52       1.40e-14      1.00e+00
```

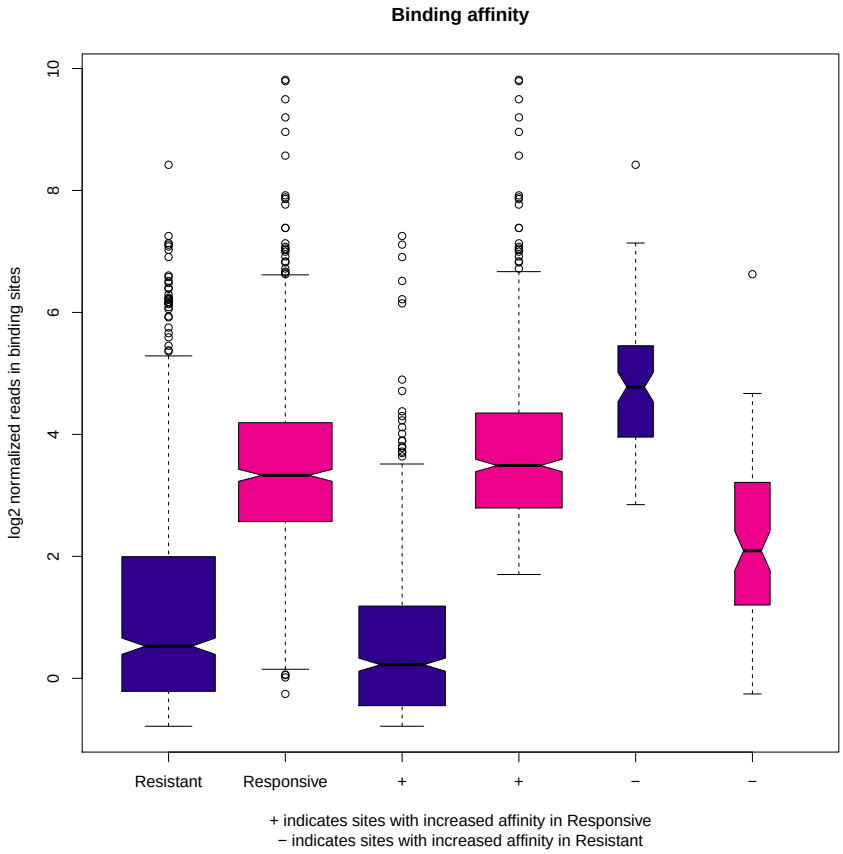

Figure 7: **Box plots of read distributions for significantly differentially bound (DB) sites.** Tamoxifen resistant samples are shown in red, and responsive samples are shown in blue. Left two boxes show distribution of reads over all DB sites in the Resistant and Responsive groups; middle two boxes show distributions of reads in DB sites that increase in affinity in the Responsive group; last two boxes show distributions of reads in DB sites that increase in affinity in the Resistant group. Generated by: `dba.plotBox(tamoxifen)`

```
Responsive.DB-     2.92e-12      6.94e-16      4.77e-25      1.13e-20      1.24e-17
               Responsive.DB-
Resistant.DB         2.92e-12
Responsive.DB        6.94e-16
Resistant.DB+        4.77e-25
Responsive.DB+       1.13e-20
Resistant.DB-        1.24e-17
Responsive.DB-       1.00e+00
```

The significance of the overall difference in distribution of concentrations amongst the differentially bound sites in the two groups is shown to be p-value=8.11e-77, while those between the Resistant and Responsive groups in the individual cases (increased in Responsive or increased in Resistant) have p-values computed as 1.10e-96 and 1.24e-17.

4.5 Heatmaps

DiffBind provides two types of heatmaps. This first, correlation heatmaps, we have already seen. For example, the heatmap shown in Figure **??** can be generated as follows:

```
> corvals <- dba.plotHeatmap(tamoxifen)
```

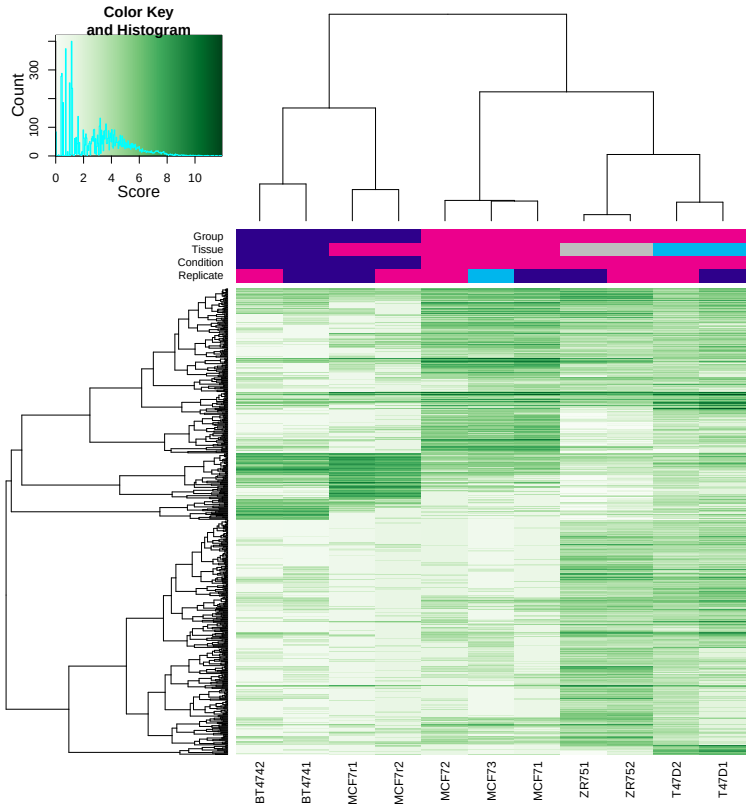

Figure 8: **Binding affinity heatmap showing affinities for differentially bound sites.** Samples cluster first by whether they are responsive to tamoxifen treatment, then by cell line. Clusters of binding sites show distinct patterns of affinity levels. Generated by: `dba.plotHeatmap(tamoxifen, contrast=1, correlations=FALSE)`

The effect of different scoring methods (normalization) can be examined in these plots by setting the `score` parameter to a different value. The default value, `DBA_SCORE_TMM_MINUS_FULL`, uses the TMM normalization procedure from *edgeR* , with control reads subtracted first and using the full library size (total reads in library). Another scoring method is to use RPKM fold (RPKM of the ChIP reads divided by RPKM of the control reads; a correlation heatmap for all the data using this scoring method can be obtained by typing `dba.plotHeatmap(tamoxifen, score=DBA_SCORE_RPKM_FOLD)`.

Another way to view the patterns of binding affinity directly in the differentially bound sites is via a binding affinity heatmap. This can be plotted for the example case as follows:

```
> corvals <- dba.plotHeatmap(tamoxifen, contrast=1, correlations=FALSE)
```

Figure **??** shows the affinities and clustering of the differentially bound sites (rows), as well as the sample clustering (columns). This plot can be tweaked to get more contrast, for example by using row-scaling `dba.plotHeatmap(tamoxifen, contrast=1, correlations=FALSE, scale="row")`.

5 Example: Differential binding analysis using a blocking factor

The previous example showed how to perform a differential binding analysis using a single factor with two values; that is, finding the significantly differentially bound sites between two sets of samples. This section extends the example by including a second factor, potentially with multiple values, that represents a confounding condition. Examples of experiments where it is appropriate to use a blocking factor include ones where there are potential batch effects, with

samples from the two conditions prepared together, or a matched design (e.g. matched normal and tumor pairs, where the primary factor of interest is to discover sites consistently differentially bound between normal and tumor samples). In the current example, the confounding effect we want to control for is the presence of two sets of samples, one tamoxifen responsive and one tamoxifen resistant, that are both derived from the same MCF7 cell line.

In the previous analysis, the two MCF7-derived cell lines tended to cluster together. While the differential binding analysis was able to identify sites that could be used to separate the resistant from the responsive samples, the confounding effect of the common ancestry could still be seen even when considering only the significantly differentially bound sites (Figure **??**).

Using the generalized linear modelling (GLM) functionality included in *edgeR* and *DESeq2* , the confounding factor can be explicitly modeled. This is done by specifying a blocking factor to `dba.contrast`. There are a number of ways to specify this factor. If it is encapsulated in a class of metadata (eg. DBA_REPLICATE, or DBA_TREATMENT etc.), simply specifying the metadata field is sufficient. In the current case, there is no specific metadata field that captures the factor we want to block (although an unused metadata field, such as DBA_TREATMENT, could be used to specify this factor). An alternate way of specifying the confounded samples is to use a mask:

```
> data(tamoxifen_counts)
> tamoxifen <- dba.contrast(tamoxifen,categories=DBA_CONDITION,
+                           block=tamoxifen$masks$MCF7)
```

Now when the analysis is run, it will be run using both the single-factor comparison as well as fitting a linear model with the second, blocking factor, for comparison:

```
> tamoxifen <- dba.analyze(tamoxifen)
> tamoxifen
```

```
11 Samples, 2845 sites in matrix:
       ID Tissue Factor  Condition  Treatment Replicate Caller Intervals FRiP
1  BT4741  BT474     ER  Resistant Full-Media         1 counts      2845 0.16
2  BT4742  BT474     ER  Resistant Full-Media         2 counts      2845 0.15
3   MCF71   MCF7     ER Responsive Full-Media         1 counts      2845 0.27
4   MCF72   MCF7     ER Responsive Full-Media         2 counts      2845 0.17
5   MCF73   MCF7     ER Responsive Full-Media         3 counts      2845 0.23
6   T47D1   T47D     ER Responsive Full-Media         1 counts      2845 0.10
7   T47D2   T47D     ER Responsive Full-Media         2 counts      2845 0.06
8  MCF7r1   MCF7     ER  Resistant Full-Media         1 counts      2845 0.20
9  MCF7r2   MCF7     ER  Resistant Full-Media         2 counts      2845 0.13
10  ZR751   ZR75     ER Responsive Full-Media         1 counts      2845 0.32
11  ZR752   ZR75     ER Responsive Full-Media         2 counts      2845 0.22

1 Contrast:
     Group1 Members1     Group2 Members2 Block1Val InBlock1 Block2Val InBlock2 DB.DESeq2
1 Resistant        4 Responsive        7      true        5     false        6       677
  DB.DESeq2.block
1             767
```

This indicates that where the standard, single-factor *DESeq2* analysis identifies 677 differentially bound sites, the analysis using the blocking factor finds 767 such sites. An MA plot shows how the analysis has changed:

```
> dba.plotMA(tamoxifen,method=DBA_DESEQ2_BLOCK)
```

The resulting plot is shown in Figure **??**. Comparing this to Figure **??**, at least two differences can be observed. The analysis has become more sensitive, with sites being identified as significantly differentially bound with lower magnitude fold changes (as low as twofold, as this plot is on a log2 scale). But it is not merely lowering a fold threshold: some sites with higher fold changes are no longer found to be significant. These were identified as significantly differentially bound in the earlier analysis because the confounding factor was not being modeled.

It is also interesting to compare the performance of *edgeR* with that of *DESeq2* on this dataset:

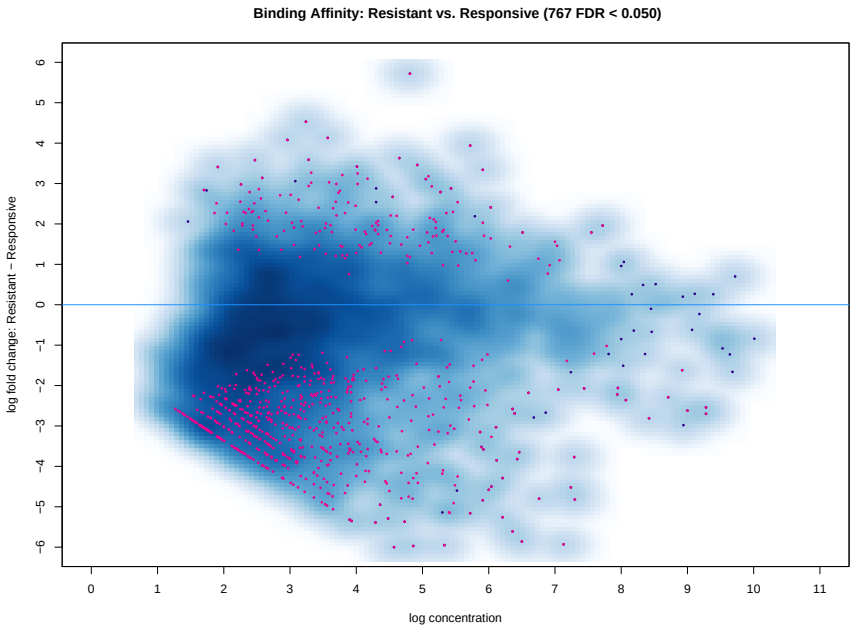

Figure 9: **MA plot of Resistant-Responsive contrast, using MCF7 origin as a blocking factor.** Sites identified as significantly differentially bound shown in red. Generated by: `dba.plotMA(tamoxifen, method=DBA_DESEQ2_BLOCK)`

```
> tamoxifen <- dba.analyze(tamoxifen,method=DBA_ALL_METHODS)
> dba.show(tamoxifen,bContrasts=T)[9:12]
  DB.edgeR DB.edgeR.block DB.DESeq2 DB.DESeq2.block
1      339            599       677             767
```

The main difference between the two methods is how the normalize the raw read counts. We see that while *edgeR* identifies a lower number of sites than *DESeq2*, when modelling the confounding factor the greater sensitivity results in more sites being identified using both packages. You can check this by looking at the identified sites using `dba.report`, and performing MA, heatmap, and PCA plots.

We can also compare the sites identified using *edgeR* and *DESeq2*. An easy way to do this is to use a special feature of the `dba.report` function that constructs a "results-bdba.plotVenn(ased" *DBA object*:

```
> tam.block <- dba.report(tamoxifen,method=DBA_ALL_METHODS_BLOCK,bDB=TRUE,bAll=TRUE)
> tam.block
4 Samples, 1018 sites in matrix:
                      ID Tissue Factor Condition Treatment Intervals
1 Resistant_vs_Responsive    All     DB     edgeR                339
2 Resistant_vs_Responsive    All     DB    DESeq2                677
3 Resistant_vs_Responsive    All     DB     edgeR     block       599
4 Resistant_vs_Responsive    All     DB    DESeq2     block       767
> dba.plotVenn(tam.block,1:4,label1="edgeR",label2="DESeq2",
+                            label3="edgeR Blocked", label4="DESeq2 Blocked")
```

The overlap is shown in Figure **??**. The largest group of sites are identified by both *edgeR* and *DESeq2* using a multi-factor (blocked) analysis.

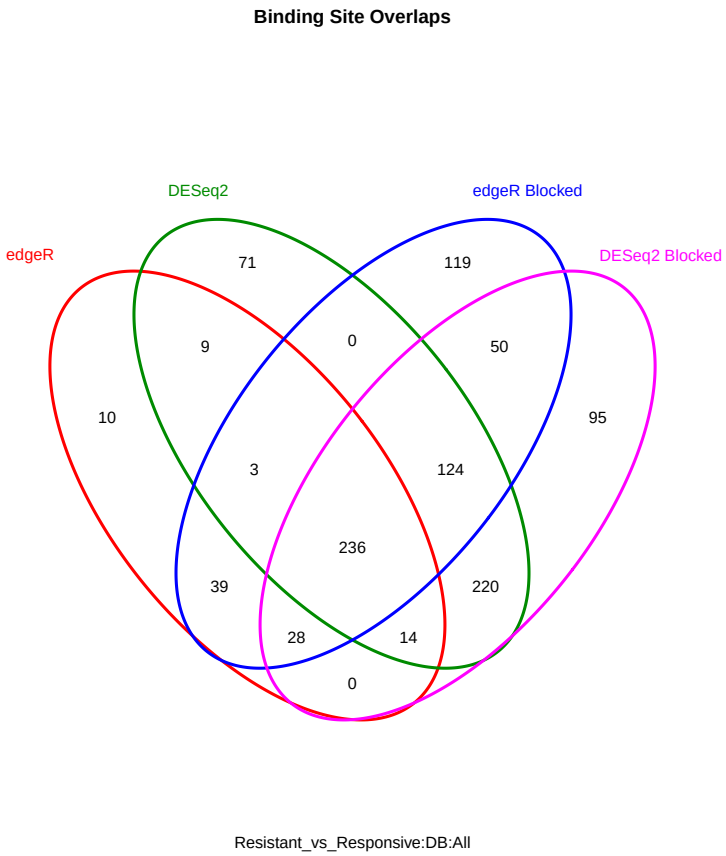

Figure 10: **Venn diagram showing overlap of differentially bound peaks identified using edgeR, and DESeq2 to do a simple and a multi-factor (blocking) analysis.** Generated by plotting the result of: `dba.plotVenn(tam.block,1:4, label1="edgeR",label2="DESeq2", label3="edgeR Blocked", label4="DESeq2 Blocked")`

6 Example: Occupancy analysis and overlaps

In this section, we look at the tamoxifen resistance ER-binding dataset in some more detail, showing what a pure occupancy-based analysis would look like, and comparing it to the results obtained using the affinity data. For this we will start by re-loading the peaksets:

```
> data(tamoxifen_peaks)
```

6.1 Overlap rates

One reason to do an occupancy-based analysis is to determine what candidate sites should be used in a subsequent affinity-based analysis. In the example so far, we took all sites that were identified in peaks in at least two of the eleven peaksets, reducing the number of sites from 3795 overall to the 2845 sites used in the differential analysis. We could have used a more stringent criterion, such as only taking sites identified in five or six of the peaksets, or a less stringent one, such as including all 3795 sites. In making the decision of what criteria to use many factors come into play, but it helps to get an idea of the rates at which the peaksets overlap (for more details on how overlaps are determined, see Section 7.2 on peak merging). A global overview can be obtained using the `RATE` mode of the `dba.overlap` function as follows:

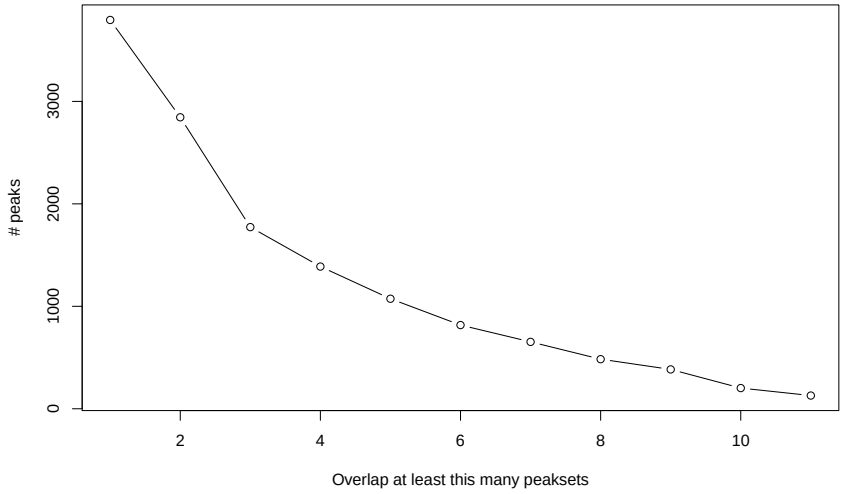

Figure 11: **Overlap rate plot.** Shows how the number of overlapping peaks decreases as the overlap criteria becomes more stringent. X axis shows the number of peaksets in which the site is identified, while the Y axis shows the number of overlapping sites. Generated by plotting the result of: `dba.overlap(tamoxifen,mode=DBA_OLAP_RATE)`

```
> olap.rate <- dba.overlap(tamoxifen,mode=DBA_OLAP_RATE)
> olap.rate
 [1] 3795 2845 1773 1388 1074  817  653  484  384  202  129
```

The returned data in `olap.rate` is a vector containing the number of peaks that appear in at least one, two, three, and so on up to all eleven peaksets.

These values can be plotted to show the overlap rate drop-off curve:

```
> plot(olap.rate,type='b',ylab='# peaks', xlab='Overlap at least this many peaksets')
```

The rate plot is shown in Figure **??**. These curves typically exhibit a roughly geometric drop-off, with the number of overlapping sites halving as the overlap criterion become stricter by one site. When the drop-off is extremely steep, this is an indication that the peaksets do not agree very well. For example, if there are replicates you expect to agree, there may be a problem with the experiment. In the current example, peak agreement is high and the curve exhibits a better than geometric drop-off.

6.2 Deriving consensus peaksets

When performing an overlap analysis, it is often the case that the overlap criteria are set stringently in order to lower noise and drive down false positives.[3] The presence of a peak in multiple peaksets is an indication that it is a "real" binding site, in the sense of being identifiable in a repeatable manner. The use of biological replicates (performing the ChIP multiple times), as in the tamoxifen dataset, can be used to guide derivation of a consensus peakset. Alternatively, an inexpensive but less powerful way to help accomplish this is to use multiple peak callers for each ChIP dataset and look for agreement between peak callers ([**?**]).

Consider for example the standard (tamoxifen responsive) MCF7 cell line, represented by three replicates in this dataset. How well do the replicates agree on their peak calls? The overlap rate for just the positive MCF7 samples can be isolated

[3]It is less clear that limiting the potential binding sites in this way is appropriate when focusing on affinity data, as the differential binding analysis method will identify only sites that are significantly differentially bound, even if operating on peaksets that include incorrectly identified sites.

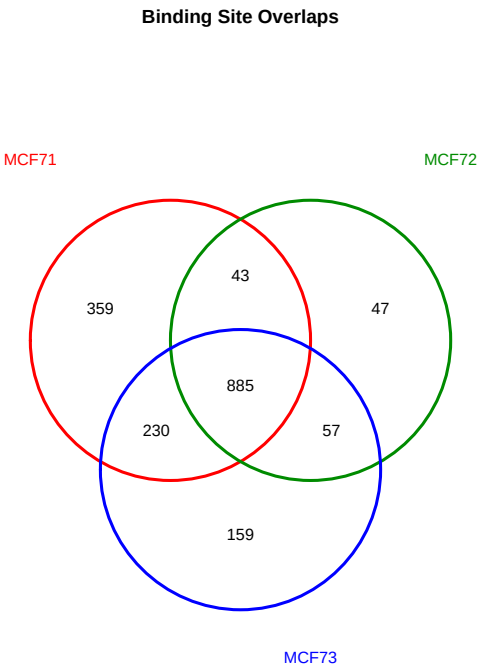

Figure 12: **Venn diagram showing how the ER peak calls for three replicates of responsive MCF7 cell line overlap.** Generated by plotting the result of: `dba.venn(tamoxifen,tamoxifen$masks$MCF7 & tamoxifen$masks$Responsive)`

using a *sample mask*. A set of sample masks are automatically associated with a `DBA object` in the `$masks` field:

```
> names(tamoxifen$masks)
```

```
 [1] "BT474"       "MCF7"        "T47D"        "ZR75"        "ER"          "Resistant"
 [7] "Responsive"  "Full-Media"  "bed"         "Replicate.1" "Replicate.2" "Replicate.3"
[13] "All"         "None"
```

Arbitrary masks can be generated using the `dba.mask` function, or simply by specifying a vector of peakset numbers. In this case, a mask that isolates the MCF7 samples can be generated by combining to pre-defined masks (MCF7 and Responsive) and passed into the `dba.overlap` function:

```
> dba.overlap(tamoxifen,tamoxifen$masks$MCF7 & tamoxifen$masks$Responsive,
+             mode=DBA_OLAP_RATE)
```

```
[1] 1780 1215  885
```

There are 885 peaks (out of 1780) identified in all three replicates. A finer grained view of the overlaps can be obtained with the `dba.plotVenn` function:

```
> dba.plotVenn(tamoxifen, tamoxifen$masks$MCF7 & tamoxifen$masks$Responsive)
```

The resultant plot is shown as Figure **??**. This plot shows the 885 consensus peaks identified as common to all replicates, but further breaks down how the replicates relate to each other. The same can be done for each of the replicated cell line experiments, and rather than applying a global cutoff (3 of 11), each cell line could be dealt with individually in deriving

a final peakset. A separate consensus peakset for each of the replicated sample types can be added to the DBA object using `dba.peakset`:

```
> tamoxifen <- dba.peakset(tamoxifen, consensus=c(DBA_TISSUE,DBA_CONDITION),
+             minOverlap=0.66)
> tamoxifen
```

```
16 Samples, 2845 sites in matrix (3795 total):
                ID Tissue Factor  Condition  Treatment Replicate Caller Intervals
1           BT4741  BT474     ER  Resistant Full-Media         1    bed      1080
2           BT4742  BT474     ER  Resistant Full-Media         2    bed      1122
3            MCF71   MCF7     ER Responsive Full-Media         1    bed      1556
4            MCF72   MCF7     ER Responsive Full-Media         2    bed      1046
5            MCF73   MCF7     ER Responsive Full-Media         3    bed      1339
6            T47D1   T47D     ER Responsive Full-Media         1    bed       527
7            T47D2   T47D     ER Responsive Full-Media         2    bed       373
8           MCF7r1   MCF7     ER  Resistant Full-Media         1    bed      1438
9           MCF7r2   MCF7     ER  Resistant Full-Media         2    bed       930
10           ZR751   ZR75     ER Responsive Full-Media         1    bed      2346
11           ZR752   ZR75     ER Responsive Full-Media         2    bed      2345
12 BT474:Resistant  BT474     ER  Resistant Full-Media       1-2    bed       896
13 MCF7:Responsive   MCF7     ER Responsive Full-Media     1-2-3    bed      1215
14 T47D:Responsive   T47D     ER Responsive Full-Media       1-2    bed       318
15  MCF7:Resistant   MCF7     ER  Resistant Full-Media       1-2    bed       879
16 ZR75:Responsive   ZR75     ER Responsive Full-Media       1-2    bed      1933
```

This adds a new consensus peakset for each set of samples that share the same Tissue and Condition values. The exact effect could be obtained by calling `tamoxifen <- dba.peakset(tamoxifen, consensus=-DBA_REPLICATE)` on the original set of peaks; this tells *DiffBind* to generate a consensus peakset for every set of samples that have identical metadata values *except* the Replicate number.

From this, a new *DBA* object can be generated consisting of only the five consensus peaksets (the $Consensus mask filters peaksets previously formed using `dba.peakset`) :

```
> tamoxifen_consensus <- dba(tamoxifen, mask=tamoxifen$masks$Consensus)
> tamoxifen_consensus
```

```
5 Samples, 1247 sites in matrix (2666 total):
              ID Tissue Factor  Condition  Treatment Replicate Caller Intervals
1 BT474:Resistant  BT474     ER  Resistant Full-Media       1-2    bed       896
2 MCF7:Responsive   MCF7     ER Responsive Full-Media     1-2-3    bed      1215
3 T47D:Responsive   T47D     ER Responsive Full-Media       1-2    bed       318
4  MCF7:Resistant   MCF7     ER  Resistant Full-Media       1-2    bed       879
5 ZR75:Responsive   ZR75     ER Responsive Full-Media       1-2    bed      1933
```

Alternatively, a master consensus peakset could be generated, and reads counted, directly using `dba.count`: `tamoxifen <- dba.count(tamoxifen, peaks=tamoxifen$masks$Consensus)`

Finally, consider an analysis where we wished to treat all five MCF7 samples together to look for binding sites specific to that cell line irrespective of tamoxifen resistant/responsive status. We can create consensus peaksets for each cell type, and look at how the resultant peaks overlap (shown in Figure **??**):

```
> data(tamoxifen_peaks)
> tamoxifen <- dba.peakset(tamoxifen, consensus=DBA_TISSUE, minOverlap=0.66)
> dba.plotVenn(tamoxifen, tamoxifen$masks$Consensus)
```

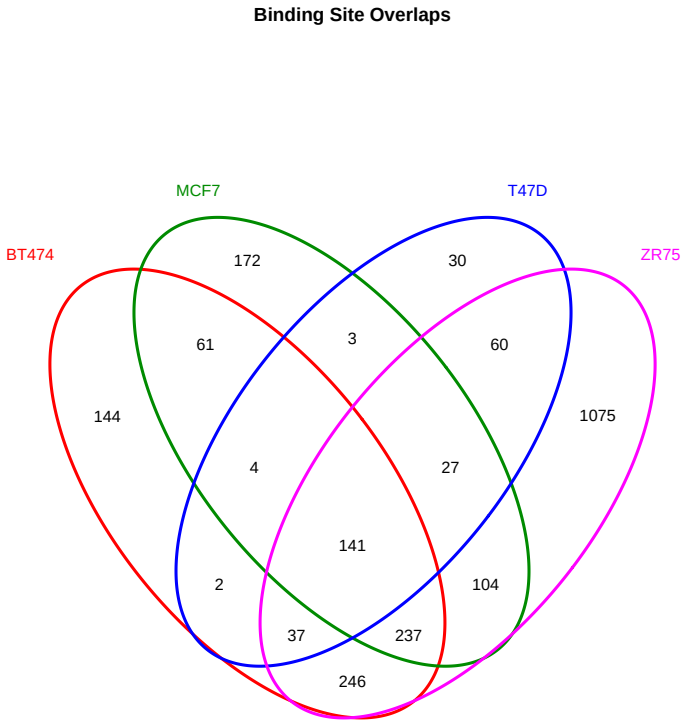

Figure 13: **Venn diagram showing how the consensus peaks for each cell type overlap.** Generated by plotting the result of: `dba.venn(tamoxifen,tamoxifen$masks$Consensus)`

6.3 A complete occupancy analysis: identifying sites unique to a sample group

Occupancy-based analysis, in addition to offering many ways of deriving consensus peaksets, can also be used to identify sites unique to a group of samples. This is analogous to, but not the same as, finding differentially bound sites. In these subsections, the two approaches are directly compared.

Returning to the original tamoxifen dataset:

```
> data(tamoxifen_peaks)
```

We can derive consensus peaksets for the Resistant and Responsive groups. First we examine the overlap rates:

```
> dba.overlap(tamoxifen,tamoxifen$masks$Resistant,mode=DBA_OLAP_RATE)
[1] 2029 1375  637  456
> dba.overlap(tamoxifen,tamoxifen$masks$Responsive,mode=DBA_OLAP_RATE)
[1] 3416 2503 1284  865  660  284  180
```

Requiring that consensus peaks overlap in at least one third of the samples in each group results in 1375 sites for the Resistant group and 1284 sites for the Responsive group:

```
> tamoxifen <- dba.peakset(tamoxifen, consensus=DBA_CONDITION, minOverlap=0.33)
> dba.plotVenn(tamoxifen,tamoxifen$masks$Consensus)
```

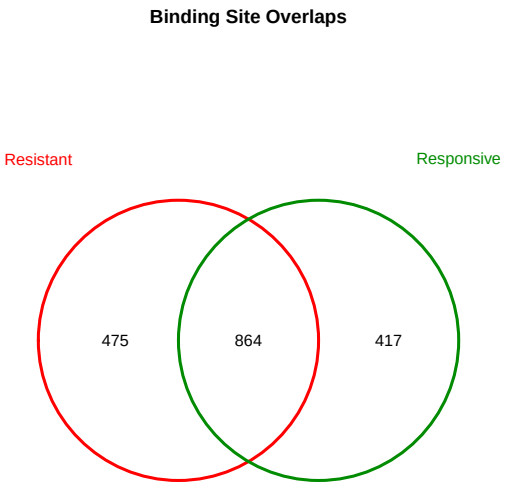

Figure 14: **Venn diagram showing how the ER peak calls for two response groups overlap.** Generated by plotting the result of: `dba.plotVenn(tamoxifen, tamoxifen$masks$Consensus)`

Figure **??** shows that 475 sites are unique to the Resistant group, and 417 sites are unique to the Responsive group, with 864 sites being identified in both groups (meaning in at least half the Resistant samples and at least three of the seven Responsive samples). If our primary interest is in finding binding sites that are different between the two groups, it may seem reasonable to consider the 864 common sites to be uninteresting, and focus on the 892 sites that are unique to a specific group. These unique sites can be obtained using `dba.overlap`:

```
> tamoxifen.OL <- dba.overlap(tamoxifen, tamoxifen$masks$Consensus)
```

The sites unique to the Resistant group are accessible in `tamoxifen.OL$onlyA`, with the Responsive-unique sites in `tamoxifen.OL$onlyB`:

```
> tamoxifen.OL$onlyA
```

```
GRanges object with 475 ranges and 1 metadata column:
       seqnames              ranges strand |              score
          <Rle>           <IRanges>  <Rle> |          <numeric>
     2    chr18     [150764, 151269]      * | 0.0216970450072851
     3    chr18     [188982, 189652]      * | 0.0829603544003535
     5    chr18     [311530, 312172]      * | 0.0647360163743927
     7    chr18     [356560, 357362]      * | 0.0264811285562276
     8    chr18     [371110, 372102]      * | 0.0327929633632912
   ...      ...                  ...    ... .                 ...
  1731    chr18 [76528540, 76529618]      * | 0.0367730545440824
  1744    chr18 [77056886, 77057516]      * | 0.0242664497643353
```

```
  1745    chr18 [77062037, 77062828]      * | 0.0233993536647987
  1747    chr18 [77300430, 77301170]      * | 0.0386594849737823
  1750    chr18 [77424530, 77425198]      * | 0.0280821172783238
  -------
  seqinfo: 1 sequence from an unspecified genome; no seqlengths
```

```
> tamoxifen.OL$onlyB
```

```
GRanges object with 417 ranges and 1 metadata column:
       seqnames              ranges strand |              score
          <Rle>           <IRanges>  <Rle> |          <numeric>
     1    chr18   [ 111564,  112005]      * | 0.0465361539385305
     6    chr18   [ 346464,  347342]      * | 0.0589574123703341
    24    chr18   [ 812595,  813462]      * | 0.0526326822538518
    32    chr18   [1075317, 1076051]      * | 0.0670931275898503
    37    chr18   [1241658, 1242455]      * | 0.0414763618020171
   ...      ...                 ...    ... .                ...
  1742    chr18 [76805366, 76806312]      * | 0.0398786180425705
  1748    chr18 [77318446, 77319078]      * | 0.0358981080016437
  1749    chr18 [77389690, 77390304]      * | 0.0237104056623095
  1752    chr18 [77541035, 77541645]      * | 0.0499611446024885
  1756    chr18 [77987044, 77988289]      * |  0.300099483685293
  -------
  seqinfo: 1 sequence from an unspecified genome; no seqlengths
```

The scores associated with each site are derived from the peak caller confidence score, and are a measure of confidence in the peak call (occupancy), not a measure of how strong or distinct the peak is.

6.4 Comparison of occupancy and affinity based analyses

So how does this occupancy-based analysis compare to the previous affinity-based analysis?

First, different criteria were used to select the overall consensus peakset. We can compare them to see how well they agree:

```
> tamoxifen <- dba.peakset(tamoxifen,tamoxifen$masks$Consensus,
+             minOverlap=1,sampID="OL Consensus")
> tamoxifen <- dba.peakset(tamoxifen,!tamoxifen$masks$Consensus,
+             minOverlap=3,sampID="Consensus_3")
> dba.plotVenn(tamoxifen,14:15)
```

Figure **??** shows that the two sets agree on about 85% of their sites, so the results should be directly comparable between the differing parameters used to establish the consensus peaksets. [4]

Next re-load the affinity analysis:

```
> data(tamoxifen_analysis)
```

To compare the sites unique to each sample group identified from the occupancy analysis with those sites identified as differentially bound based on affinity (read count) data, we use a feature of `dba.report` that facilitates evaluating the occupancy status of sites. Here we obtain a report of all the sites (`th=1`) with occupancy statistics (`bCalled=T`):

```
> tamoxifen.rep <- dba.report(tamoxifen,bCalled=TRUE,th=1)
```

The `bCalled` option adds two columns to the report (`Called1` and `Called2`), one for each group, giving the number of samples within the group in which the site was identified as a peak in the original peaksets generated by the peak caller. We can use these to recreate the overlap criteria used in the occupancy analysis:

[4] Alternatively, we could re-run the analysis using the newly derived consensus peakset by passing it into the counting function: `> tamoxifen <- dba.count(tamoxifen, peaks=tamoxifen$masks$Consensus)`

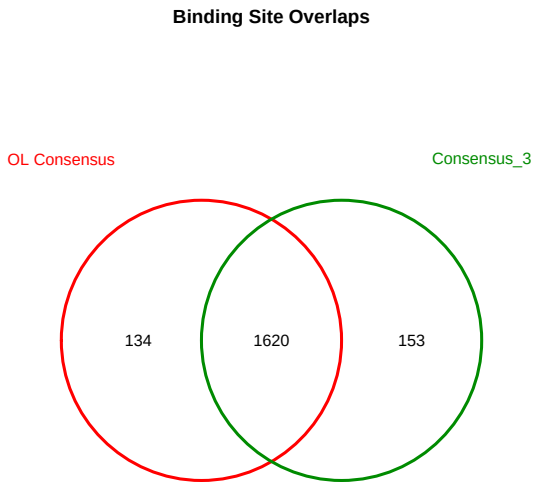

Figure 15: **Venn diagram showing how the ER peak calls for two different ways of deriving consensus peaksets.** Generated by plotting the result of: `dba.plotVenn(tamoxifen,14:15)`

```
> onlyResistant <- tamoxifen.rep$Called1>=2 & tamoxifen.rep$Called2<3
> sum(onlyResistant )
```

```
[1] 473
```

```
> onlyResponsive <- tamoxifen.rep$Called2>=3 &  tamoxifen.rep$Called1<2
> sum(onlyResponsive)
```

```
[1] 417
```

```
> bothGroups <- tamoxifen.rep$Called1>= 2 & tamoxifen.rep$Called2>=3
> sum(bothGroups)
```

```
[1] 864
```

Comparing these numbers verifies the similarity with those seen in Figure **??**, showing again how the basic analysis is not oversensitive to differences in how the consensus peaksets are formed. This overlap analysis suggests that 890 of the sites are uniquely bound in either the Responsive or Resistant groups, while 864 sites are common to both.

Completing a full differential analysis and focusing on only those sites identified as significantly differentially bound (FDR $<= 0.1$), however, shows a different story than that obtainable using only occupancy data:

```
> tamoxifen.DB <- dba.report(tamoxifen,bCalled=T)
> onlyResistant.DB <- tamoxifen.DB$Called1>=2 & tamoxifen.DB$Called2<3
> sum(onlyResistant.DB)
```

```
[1] 71
```

```
> onlyResponsive.DB <- tamoxifen.DB$Called2>=3 & tamoxifen.DB$Called1<2
> sum(onlyResponsive.DB)
```

```
[1] 207
```

```
> bothGroups.DB <- tamoxifen.DB$Called1>=2 & tamoxifen.DB$Called2>=3
> sum(bothGroups.DB)
```

```
[1] 67
```

There are a number of notable differences in the results. First, overall there are many fewer sites identified as differentially bound (71+207+67 <- 677) than are unique to one condition (473+417 = 890). Indeed, most of the sites identified in the occupancy analysis as unique to a sample group are not found to be significantly differentially bound using the affinity data. While partly this is a result of the stringency of the statistical tests, it shows how the affinity analysis can discriminate between sites where peak callers are making occupancy decisions that do not reflect significant differences in read densities at these sites. Note that only about 31% of sites unique to one condition are identifiable as significantly differentially bound (71+207 = 278 out of 890). Secondly, differentially bound sites are as likely to be called in the consensus of both response groups as they are to be unique to one group, as about 20% of the total sites identified as significantly differentially bound (345) are called as peaks in *both* response groups (67).

A final advantage of a quantitative analysis is that the differentially bound peaks identified using the affinity analysis are associated with significance statistics (p-value and FDR) that can be used to rank them for further examination, while the occupancy analysis yields a relatively unordered list of peaks, as the peak caller statistics refer only to the significance of occupancy, and not of differential binding.

7 Technical notes

This section includes some technical notes explaining some of the technical details of *DiffBind* processing.

7.1 Loading peaksets

There are a number of ways to get peaksets loaded into a DBA object. Peaksets can be read in from files or loaded from interval sets already stored in an R object. Samples can be specified either in a sample sheet (using `dba`) or loaded one at a time (using `dba.peakset`).

When loading in peaksets from files, specifying what peak caller generated the file enables peaks from supported peak callers to be read in. See the help page for `dba.peakset` for a list of supported peak callers. Any string can be used to indicate the peak caller; if it is not one of the supported callers, a default "raw" format is assumed, consisting of a text file with three or four columns (indicating the chromosome, start position, and end position, with a score for each interval found in the fourth column, if present). You can further control how peaks are read using the `PeakFormat`, `ScoreCol`, and `bLowerBetter` fields if you want to override the defaults for the specified peak caller identifier. For example, with the tamoxifen dataset used in this tutorial, the peaks were called using the MACS peak caller, but the data are supplied as text files in BED format, not the expected MACS "xls" format. To maintain the peak caller in the metadata, we could specify the `PeakCaller` as "macs" but the `PeakFormat` as "bed". If we wanted to use peak scores from a column other than the fifth, the `scorecol` parameter could be set to indicate the appropriate column number. When handling scoring, *DiffBind* by default assumes that a higher score indicates a "better" peak. If this is not the case, for example if the score is a p-value or FDR, we could set `bLowerScoreBetter` to `TRUE`.

When using a sample sheet, values for fields missing in the sample sheet can be supplied when calling `dba`. In addition to the minimal sample sheet used for the tutorial, an equivalent sample sheet with all the metadata fields is included, called "tamoxifen_allfields.csv". See the help page for `dba` for an example using this sample sheet.

7.2 Merging peaks

When forming the global binding matrix consensus peaksets, *DiffBind* first identifies all unique peaks amongst the relevant peaksets. As part of this process, it merges overlapping peaks, replacing them with a single peak representing the narrowest region that covers all peaks that overlap by at least one base. There are at least two consequences of this that are worth noting.

First, as more peaksets are included in analysis, the average peak width tends to become longer as more overlapping peaks are detected and the start/end points are adjusted outward to account for them. Secondly, peak counts may not appear to add up as you may expect due to merging. For example, if one peakset contains two small peaks near to each other, while a second peakset includes a single peak that overlaps both of these by at least one base, these will all be replaced in the merged matrix with a single peak. A s more peaksets are added, multiple peaks from multiple peaksets may be merged together to form a single, wider peak. Use of the "summits" parameter is recommended to control for this widening effect.

7.3 DESeq2 analysis

When `dba.analyze` is invoked using `method=DBA_DESEQ2`, a standardized differential analysis is performed using the *DESeq2*package ([**?**]). This section details the precise steps in that analysis.

For each contrast, a separate analysis is performed. First, a matrix of counts is constructed for the contrast, with columns for all the samples in the first group, followed by columns for all the samples in the second group. The raw read count is used for this matrix; if the `bSubControl` parameter is set to `TRUE` (as it is by default), the raw number of reads in the control sample (if available) will be subtracted. Next the library size is computed for each sample for use in subsequent normalization. By default, this is the total number of reads in peaks (the sum of each column). Alternatively, if the `bFullLibrarySize` parameter is set to TRUE, the total number of reads in the library (calculated from the source BAM/BED file) is used. The first step concludes with a call to *DESeq2*'s `DESeqDataSetFromMatrix` function, which returns a *DESeqDataSet* object.

If `bFullLibrarySize` is set to TRUE, then `sizeFactors` is called with the number of reads in the BAM/BED files for each ChIP sample, divided by the minimum of these; otherwise, `estimateSizeFactors` is invoked.

`estimateDispersions` is then called with the *DESeqDataSet* object and `fitType` set to `local`. Next the model is fitted and tested using `nbinomWaldTest`. The final results (as a *DESeqDataSet*) are accessible within the *DBA* object as

```
DBA$contrasts[[n]]$DESeq2$DEdata
```

and may be examined and manipulated directly for further customization. Note however that if you wish to use this object directly with *DESeq2* functions, then the `bReduceObjects` parameter should be set to FALSE, otherwise the default value of TRUE will result in essential object fields being stripped.

If a blocking factor has been added to the contrast, an additional *DESeq2* analysis is carried out by setting the `design` to include all the unique values for the blocking factor. This occurs before the dispersion values are calculated. The resultant *DESeqDataSet* object is accessible as

`DBA$contrasts[[n]]$DESeq2$block$DEdata`.

7.4 edgeR analysis

When `dba.analyze` is invoked using the default `method=DBA_EDGER`[5], a standardized differential analysis is performed using the *edgeR* package ([**?**]). This section details the precise steps in that analysis.

For each contrast, a separate analysis is performed. First, a matrix of counts is constructed for the contrast, with columns for all the samples in the first group, followed by columns for all the samples in the second group. The raw read count is

[5]Note that *edgeR* can be made the default analysis method for a DBA object by setting `DBA$config$AnalysisMethod=DBA_EDGER`.

used for this matrix; if the `bSubControl` parameter is set to `TRUE` (as it is by default), the raw number of reads in the control sample (if available) will be subtracted (with a minimum final read count of 1). Next the library size is computed for each sample for use in subsequent normalization. By default, this is the total number of reads in the library (calculated from the source BAM//BED file). Alternatively, if the `bFullLibrarySize` parameter is set to FALSE,the total number of reads in peaks (the sum of each column) is used. Note that "effective" library size (`bFullLibrarySize=FALSE`) may be more appropriate for situations when the overall signal (binding rate) is expected to be directly comparable between the samples. Next comes a call to *edgeR* 's `DGEList` function. The *DGEList* object that results is next passed to `calcNormFactors` with `method="TMM"` and `doWeighting=FALSE`, returning an updated `DGEList` object. This is passed to `estimateCommonDisp` with default parameters.

If the method is `DBA_EDGER_CLASSIC`, then if `bTagwise` is TRUE (most useful when there are at least three members in each group of a contrast), the resulting *DGEList* object is then passed to `estimateTagwiseDisp`, with the prior set to 50 divided by two less than the total number of samples in the contrast, and `trend="none"`. The final steps are to perform testing to determine the significance measure of the differences between the sample groups by calling `exactTest` ([**?**]) using the *DGEList* with the `dispersion` set based on the `bTagwise` parameter.

If the method is `DBA_EDGER_GLM` (the default), then a a design matrix is generated with two coefficients (the Intercept and one of the groups). Next `estimateGLMCommonDisp` is called; if `bTagwise=TRUE`, `estimateGLMTagwiseDisp` is called as well. The model is fitted by calling `glmFit`, and the specific contrast fitted by calling `glmLRT`, specifying that the second coefficient be dropped. Finally, an `exactTest` ([**?**]) is performed, using either common or tagwise dispersion depending on the value specified for `bTagwise`.

This final *DGEList* for contrast n is stored in the *DBA* object as

```
DBA$contrasts[[n]]$edgeR
```

and may be examined and manipulated directly for further customization. Note however that if you wish to use this object directly with *edgeR* functions, then the `bReduceObjects` parameter should be set to FALSE, otherwise the default value of TRUE will result in essential object fields being stripped.

If a blocking factor has been added to the contrast, an additional *edgeR* analysis is carried out. This follows the `DBA_EDGER_GLM` case detailed above, except a more complex design matrix is generated that includes all the unique values for the blocking factor. These coefficients are all included in the `glmLRT` call. The resultant object is accessible as

`DBA$contrasts[[n]]$edgeR$block`.

7.5 DESeq analysis

This section is included for backward compatibility.

When `dba.analyze` is invoked using `method=DBA_DESEQ`[6], a standardized differential analysis is performed using the *DESeq* package ([**?**]). This section details the steps in that analysis.

For each contrast, a separate analysis is performed. First, a matrix of counts is constructed for the contrast, with columns for all the samples in the first group, followed by columns for all the samples in the second group. The raw read count is used for this matrix; if the `bSubControl` parameter is set to `TRUE` (as it is by default), the raw number of reads in the control sample (if available) will be subtracted. Next the library size is computed for each sample for use in subsequent normalization. By default, this is the total number of reads in the library (calculated from the source BAM//BED file). Alternatively, if the `bFullLibrarySize` parameter is set to FALSE,the total number of reads in peaks (the sum of each column) is used. Note that "effective" library size (`bFullLibrarySize=FALSE`) may be more appropriate for situations when the overall signal (binding rate) is expected to be directly comparable between the samples. The first step concludes with a call to *DESeq*'s `newCountDataSet` function, which returns a *CountDataSet* object. If `bFullLibrarySize` is set to TRUE, then `sizeFactors` is called with the number of reads in the BAM/BED files for each ChIP sample, divided by the minimum of these; otherwise, `estimateSizeFactors` is invoked. Next, `estimateDispersions` is called with the *CountDataSet* object and `fitType` set to `local`. If there are no replicates, (only one sample in each group), `method`

[6]Note that *DESeq* can be made the default analysis method for a DBA object by setting `DBA$config$AnalysisMethod=DBA_DESEQ`.

is set to `blind`. Otherwise, if `bTagwise` is TRUE, `method` is set to `per-condition`; if it is FALSE, `method` is set to `pooled` (or `pooled-CR` for a blocking analysis).

If the method is `DBA_DESEQ_CLASSIC`, `nbinomTest` is called, and the result (reordered by adjusted p-value) saved for reporting.

If the method is `DBA_DESEQ_GLM` (the default), two models are fitted using `fitNbinomGLMs`: a full model is fitted with all the coefficients, and a second model is fitted with the second coefficient dropped. These are tested against each other using `nbinomGLMTest`, with the resulting p values adjusted using `p.adjust` (with `method="BH"`).

The final results are accessible within the *DBA* object as

```
DBA$contrasts[[n]]$DESeq1$DEdata
```

and may be examined and manipulated directly for further customization. Note however that if you wish to use this object directly with *DESeq* functions, then the `bReduceObjects` parameter should be set to FALSE, otherwise the default value of TRUE will result in essential object fields being stripped.

If a blocking factor has been added to the contrast, an additional *DESeq* analysis is carried out. This follows the `DBA_DESEQ_GLM` case detailed above, except a more complex design is generated when `newCountDataSet` is called that includes all the unique values for the blocking factor. These coefficients are all included in the `fitNbinomGLMs` calls. The resultant object is accessible as

`DBA$contrasts[[n]]$DESeq1$block$DEdata`.

8 Vignette Data

Due to space limitations, the aligned reads associated with the cell line data used in this vignette are not included as part of the *DiffBind* package.

Data for the vignette are available for download at http://DiffBind.starkhome.com.

The full data for all chromosomes are also available in the Short Read Archive (GEO scession number GSE32222). Email for detailed instructions on how to retrieve them in the appropriate form.

9 Using DiffBind and ChIPQC together

DiffBind and *ChIPQC* 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*, then *ChIPQC* can accept a *DBA object* in place of the sample sheet when creating a *ChIPQCexperiment* object.

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.

10 Acknowledgements

This package was developed at Cancer Research UK's Cambridge Research Institute with the help and support of many people there. We wish to acknowledge everyone the Bioinformatics Core under the leadership of Matthew Eldridge, as well as the Nuclear Receptor Transcription Laboratory under the leadership of Jason Carroll. Researchers who contributed ideas and/or pushed us in the right direction include Caryn-Ross Innes, Vasiliki Theodorou, and Tamir Chandra among many others. We also thank members of the Gordon Smyth laboratory at the WEHI, Melbourne, particularly Mark Robinson and Davis McCarthy, for helpful discussions.

11 Session Info

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

- R version 3.3.1 (2016-06-21), `x86_64-apple-darwin13.4.0`
- Locale: `C/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8`
- Base packages: base, datasets, grDevices, graphics, methods, parallel, stats, stats4, utils
- Other packages: Biobase 2.32.0, BiocGenerics 0.18.0, DiffBind 2.0.9, GenomeInfoDb 1.8.7, GenomicRanges 1.24.3, IRanges 2.6.1, S4Vectors 0.10.3, SummarizedExperiment 1.2.3
- Loaded via a namespace (and not attached): AnnotationDbi 1.34.4, AnnotationForge 1.14.2, BBmisc 1.10, BatchJobs 1.6, BiocParallel 1.6.6, BiocStyle 2.0.3, Biostrings 2.40.2, Category 2.38.0, DBI 0.5-1, DESeq2 1.12.4, Formula 1.2-1, GO.db 3.3.0, GOstats 2.38.1, GSEABase 1.34.1, GenomicAlignments 1.8.4, GenomicFeatures 1.24.5, Hmisc 3.17-4, KernSmooth 2.23-15, Matrix 1.2-7.1, R6 2.1.3, RBGL 1.48.1, RColorBrewer 1.1-2, RCurl 1.95-4.8, RSQLite 1.0.0, Rcpp 0.12.7, Rsamtools 1.24.0, ShortRead 1.30.0, XML 3.98-1.4, XVector 0.12.1, acepack 1.3-3.3, amap 0.8-14, annotate 1.50.0, assertthat 0.1, backports 1.0.3, base64enc 0.1-3, biomaRt 2.28.0, bitops 1.0-6, brew 1.0-6, caTools 1.17.1, checkmate 1.8.1, chron 2.3-47, cluster 2.0.4, colorspace 1.2-6, data.table 1.9.6, digest 0.6.10, dplyr 0.5.0, edgeR 3.14.0, fail 1.3, foreign 0.8-67, gdata 2.17.0, genefilter 1.54.2, geneplotter 1.50.0, ggplot2 2.1.0, gplots 3.0.1, graph 1.50.0, grid 3.3.1, gridExtra 2.2.1, gtable 0.2.0, gtools 3.5.0, hwriter 1.3.2, lattice 0.20-34, latticeExtra 0.6-28, limma 3.28.21, locfit 1.5-9.1, magrittr 1.5, munsell 0.4.3, nnet 7.3-12, pheatmap 1.0.8, plyr 1.8.4, rjson 0.2.15, rpart 4.1-10, rtracklayer 1.32.2, scales 0.4.0, sendmailR 1.2-1, splines 3.3.1, stringi 1.1.1, stringr 1.1.0, survival 2.39-5, systemPipeR 1.6.4, tibble 1.2, tools 3.3.1, xtable 1.8-2, zlibbioc 1.18.0