DiffBind: Differential binding analysis of ChIP-Seq peak data

Rory Stark and Gord Brown*

Cancer Research UK - Cambridge Institute
University of Cambridge

*bioconductor@starkhome.com

Edited: 5 August 2026; Compiled: November 24, 2026

Contents

1 Introduction . . . 3

2 Processing overview . . . 3

3 Example: Obtaining differentially bound sites . . . 5

3.1 Reading in the peaksets . . . 6
3.2 Blacklists and greylists . . . 8
3.3 Counting reads . . . 8
3.4 Normalizing the data . . . 10
3.5 Establishing a model design and contrast . . . 11
3.6 Performing the differential analysis . . . 12
3.7 Retrieving the differentially bound sites . . . 13

4 Plotting in *DiffBind* . . . 14

4.1 Venn diagrams . . . 15
4.2 PCA plots . . . 15
4.3 MA plots . . . 16
4.4 Volcano plots . . . 17
4.5 Boxplots . . . 19
4.6 Heatmaps . . . 20
4.7 Profiling and Profile Heatmaps . . . 21
4.7.1 Default profile plot . . . 22
4.7.2 Merging all samples in a contrast condition . . . 22
4.7.3 Avoiding merging to show all sample replicates . . . 22

5 Example: Multi-factor designs . . . 26

6 Blacklists and Greylists . . . 29
6.1 What are blacklists and greylists? . . . 29
6.2 Why apply blacklists and greylists? . . . 29
6.3 When should blacklists and greylists be applied? . . . 30
6.4 Example: How to apply a blacklist . . . 30
6.5 Example: How to apply a greylist . . . 32
6.6 Example: How to compute a greylist with *GreyListChIP* . . . 33

7 Normalization . . . 35
7.1 Core normalization methods . . . 35
7.2 Library size calculations . . . 40
7.3 Background normalization . . . 43
7.4 Offsets and loess normalization . . . 45
7.5 Comparing the impact of normalization methods on analysis results 48
7.6 Spike-in normalization . . . 52
7.7 Parallel factor normalization . . . 57
7.8 Normalization summary . . . 58

8 Example: Occupancy analysis and overlaps . . . 60
8.1 Overlap rates . . . 60
8.2 Deriving consensus peaksets . . . 61
8.3 A complete occupancy analysis: identifying sites unique to a sample group . . . 63
8.4 Comparison of occupancy and affinity based analyses . . . 66

9 Backward compatibility with pre-version 3.0 analyses . . . 68
9.1 Running with saved DBA objects . . . 68
9.2 Re-running *DiffBind* scripts . . . 69

10 Technical notes . . . 69
10.1 Loading peaksets . . . 70
10.2 Merging peaks . . . 70
10.3 Details of *DESeq2* analysis . . . 70
10.4 Details of *edgeR* analysis . . . 71

11 Technical notes for versions prior to *DiffBind* 3.0 (without an explicit model design) . . . 72
11.1 *DESeq2* analysis . . . 72
11.2 *edgeR* analysis . . . 73

12 Vignette Data . . . 74

13 Using *DiffBind* and *ChIPQC* together 75

14 Acknowledgements. 75

15 Session Info 75

1 Introduction

This document offers an introduction and overview of the *R* Bioconductor package *DiffBind* , which provides functions for processing DNA data enriched for genomic loci, including ChIP-seq data enriched for sites where specific protein/DNA binding occurs, or histone marks are enriched, as well as open-chromatin assays such as ATAC-seq.

It is designed to work with aligned sequence reads as well as lists of enriched loci identified by a peak caller. The tool is optimized 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 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 several sections containing examples and discussion of more advanced analytic options. The first example focuses on the core task of obtaining differentially bound sites based on affinity data, while the second demonstrates the main plotting routines.

This is followed by discussions of multi-factor designs, blacklists/greylists, and normalization.

The final example revisits occupancy data (peak calls) in more detail, comparing the results of an occupancy-based analysis with an affinity-based one.

The last portions of this document include 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 ([1]), 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.) An individual sample 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.

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. In addition, peaks may be filtered based on published **blacklists** of region known to be problematic, as well as custom **greylists** derived from control track specific to the experiment (see Section 6). 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. By default, the peaks in the consensus peakset are 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 read count for each sample at every consensus binding site, whether or not it was identified as a peak in that sample. 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 significantly differentially bound between sample groups. This step includes normalizing the experimental data and establishing a model design and a contrast (or contrasts). Next the underlying 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 and volcano 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[2]. 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 standard MCF7 tamoxifen responsive line, and MCF7 cells specially treated with tamoxifen until a tamoxifen resistant version of the cell line is obtained. For each sample, there is an associated peakset derived using the MACS peak caller[1], for a total of eleven peaksets.

To save time and space in the package, only data for chromosome 18 is used for the vignette. The metadata and peak data[1] 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'))
```

If you have downloaded the vignette data, you can set the current working directory to where it is located. Alternatively, the following code will download the data into a temporary directory in which you can run the vignette:

```
> tmpdir <- tempdir()
> url <- 'https://content.cruk.cam.ac.uk/bioinformatics/software/DiffBind/DiffBind_vignette_data.tar.gz'
> file <- basename(url)
> options(timeout=600)
> download.file(url, file.path(tmpdir,file))
> untar(file.path(tmpdir,file), exdir = tmpdir )
> setwd(file.path(tmpdir,"DiffBind_Vignette"))
```

Performing a full differential binding analysis can be accomplished in a single step based on the sample sheet:

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

Obtaining the sites significantly differentially bound (DB) between the samples that respond to tamoxifen and those that are resistant is equally straightforward:

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

The `dba.analyze` function simplifies processing if you want to perform an analysis using only defaults. However this may not be the optimal (or even correct) analysis, so it is often necessary to perform each step separately in order to have greater control of the analysis. The default analysis involves six such steps, as follows:

[1]Note that due to space limitations the reads are not shipped with the package. See Section 12 for options to obtain the full dataset. It is highly recommended that your obtain the 500M dataset to work through this vignette.

```
> tamoxifen <- dba(sampleSheet="tamoxifen.csv") %>%
+   dba.blacklist() %>%
+   dba.count()     %>%
+   dba.normalize() %>%
+   dba.contrast()  %>%
+   dba.analyze()
```

Along the way, there are a number of useful plots and reports that can illuminate characteristics of the data set and guide subsequent steps.

The following subsections describe the primary analysis 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` from a `csv` file:

```
> samples <- read.csv(file.path(system.file("extra", package="DiffBind"),
+                               "tamoxifen.csv"))
> names(samples)
 [1] "SampleID"   "Tissue"     "Factor"     "Condition"  "Treatment"
 [6] "Replicate"  "bamReads"   "ControlID"  "bamControl" "Peaks"
[11] "PeakCaller"

> 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    MCF7r1   MCF7     ER  Resistant Full-Media         1
9    MCF7r2   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/Chr18_T47D_input.bam
7   reads/Chr18_T47D_ER_2.bam     T47Dc  reads/Chr18_T47D_input.bam
8   reads/Chr18_TAMR_ER_1.bam     TAMRc  reads/Chr18_TAMR_input.bam
9   reads/Chr18_TAMR_ER_2.bam     TAMRc  reads/Chr18_TAMR_input.bam
10  reads/Chr18_ZR75_ER_1.bam     ZR75c  reads/Chr18_ZR75_input.bam
11  reads/Chr18_ZR75_ER_2.bam     ZR75c  reads/Chr18_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
```

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

```
> tamoxifen <- dba(sampleSheet="tamoxifen.csv",
+                  dir=system.file("extra", package="DiffBind"))
```

Alternatively, the previously read-in sample sheet could be used directly to create the *DBA object*:

```
> tamoxifen <- dba(sampleSheet=samples)
```

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 Intervals
1  BT4741  BT474     ER  Resistant Full-Media         1      1080
2  BT4742  BT474     ER  Resistant Full-Media         2      1122
3   MCF71   MCF7     ER Responsive Full-Media         1      1556
4   MCF72   MCF7     ER Responsive Full-Media         2      1046
5   MCF73   MCF7     ER Responsive Full-Media         3      1339
6   T47D1   T47D     ER Responsive Full-Media         1       527
7   T47D2   T47D     ER Responsive Full-Media         2       373
8  MCF7r1   MCF7     ER  Resistant Full-Media         1      1438
9  MCF7r2   MCF7     ER  Resistant Full-Media         2       930
10  ZR751   ZR75     ER Responsive Full-Media         1      2346
11  ZR752   ZR75     ER Responsive Full-Media         2      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 the default binding matrix of 11 samples by the 2845 sites that overlap in at least two of the samples.

Note: *This* `DBA object`, `tamoxifen`, *is available for loading using* `data(tamoxifen_peaks)`.

Using the data from the peak calls, 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)
```

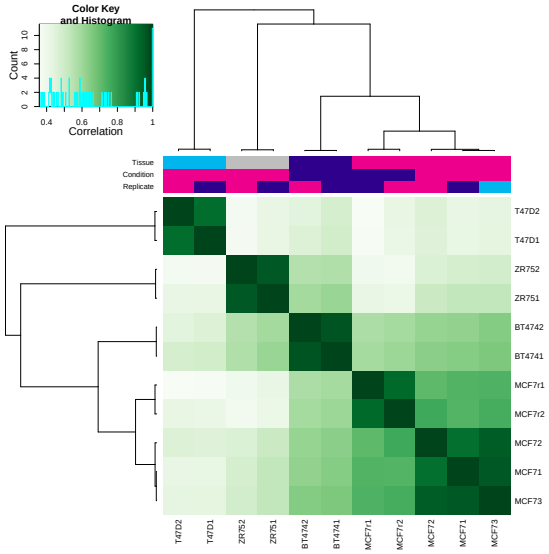


Figure 1: Correlation heatmap, using occupancy (peak caller score) data. Generated by: `plot(tamoxifen)`; can also be generated by: `dba.plotHeatmap(tamoxifen)`.

The resulting plot (Figure 1) 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 Blacklists and greylists

Blacklists and greylists are discussed in a subsequent section. See Section 6 for more details.

3.3 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: [2]

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

If you do not have the raw reads available to you, this object is available for 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   Reads FRiP
```

[2]Note that due to space limitations the reads are not shipped with the package. See Section 12 for options to obtain the full dataset. You can get the end result of the `dba.count` call by loading the supplied *R*object by invoking `data(tamoxifen_counts)`

```
1  BT4741  BT474     ER  Resistant Full-Media         1  652697 0.16
2  BT4742  BT474     ER  Resistant Full-Media         2  663370 0.15
3   MCF71   MCF7     ER Responsive Full-Media         1  346429 0.31
4   MCF72   MCF7     ER Responsive Full-Media         2  368052 0.19
5   MCF73   MCF7     ER Responsive Full-Media         3  466273 0.25
6   T47D1   T47D     ER Responsive Full-Media         1  399879 0.11
7   T47D2   T47D     ER Responsive Full-Media         2 1475415 0.06
8  MCF7r1   MCF7     ER  Resistant Full-Media         1  616630 0.22
9  MCF7r2   MCF7     ER  Resistant Full-Media         2  593224 0.14
10  ZR751   ZR75     ER Responsive Full-Media         1  706836 0.33
11  ZR752   ZR75     ER Responsive Full-Media         2 2575408 0.22
```

This shows that all the samples are using the same, 2845 length consensus peakset. Also, two new columns have been added. The first shows the total number of aligned reads for each sample (the "Full" library sizes). The second is labeled `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. For each sample, multiplying the value in the `Reads` column by the corresponding `FRiP` value will yield the number of reads that overlap a consensus peak. This can be done using the `dba.show` function:

```
> info <- dba.show(tamoxifen)
> libsizes <- cbind(LibReads=info$Reads, FRiP=info$FRiP,
+                   PeakReads=round(info$Reads * info$FRiP))
> rownames(libsizes) <- info$ID
> libsizes
       LibReads FRiP PeakReads
BT4741   652697 0.16    104432
BT4742   663370 0.15     99506
MCF71    346429 0.31    107393
MCF72    368052 0.19     69930
MCF73    466273 0.25    116568
T47D1    399879 0.11     43987
T47D2   1475415 0.06     88525
MCF7r1   616630 0.22    135659
MCF7r2   593224 0.14     83051
ZR751    706836 0.33    233256
ZR752   2575408 0.22    566590
```

As this example is based on a transcription factor that binds to the DNA, resulting in "punctate", relatively narrow peaks, the default option to re-center each peak around the point of greatest enrichment is appropriate. This keeps the peaks at a consistent width (in this case, the default `summits=200` results in 401bp-wide intervals, extending 200bp up- and downstream of the summit)

We can also plot a new correlation heatmap based on the count scores, seen in Figure 2 (compare to Figure 1). While this shows a slightly different clustering, with overall higher levels of correlation (due to using normalized read counts instead of whether or not a peak was called), responsiveness to tamoxifen treatment does not appear to form a basis for clustering when using all of the affinity scores. (Note that at this point the count scores are computed using default normalization parameters. Note that the clustering can change based on what normalization scoring metric is used; see Section 7 for more details).

```
> plot(tamoxifen)
```

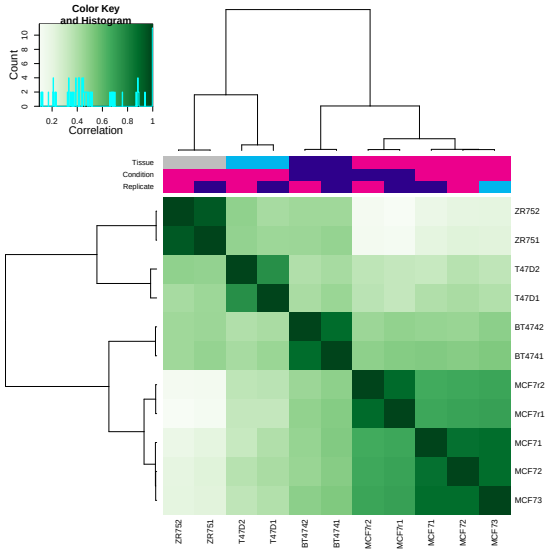


Figure 2: Correlation heatmap, using affinity (read count) data. Generated by: `plot(tamoxifen)`; can also be generated by: `dba.plotHeatmap(tamoxifen)`

3.4 Normalizing the data

The next step is to tell *DiffBind* how the data are to be normalized. Normalization is discussed in detail in Section 7; here we consider the default normalization for our example, obtained using the `dba.normalize` function:

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

By default, the data are normalized based on sequencing depth.

The details of the normalization can be examined:

```
> norm <- dba.normalize(tamoxifen, bRetrieve=TRUE)
> norm

$norm.method
[1] "lib"

$norm.factors
 [1] 0.8099610 0.8232056 0.4298993 0.4567323 0.5786191 0.4962278 1.8309087
 [8] 0.7652039 0.7361583 0.8771445 3.1959394

$lib.method
[1] "full"

$lib.sizes
 [1]  652697  663370  346429  368052  466273  399879 1475415  616630  593224
```

```
[10]  706836 2575408

$filter.value
[1] 1
```

This shows the normalization method used (lib), the calculated normalization factors for each sample, and the full library sizes (which include the total number of reads in the associated .bam files).

The default library-size based methods results in all the library sizes being normalized to be the same (the `mean` library size):

```
> normlibs <- cbind(FullLibSize=norm$lib.sizes, NormFacs=norm$norm.factors,
+                   NormLibSize=round(norm$lib.sizes/norm$norm.factors))
> rownames(normlibs) <- info$ID
> normlibs

       FullLibSize  NormFacs NormLibSize
BT4741      652697 0.8099610      805838
BT4742      663370 0.8232056      805838
MCF71       346429 0.4298993      805838
MCF72       368052 0.4567323      805838
MCF73       466273 0.5786191      805838
T47D1       399879 0.4962278      805838
T47D2      1475415 1.8309087      805838
MCF7r1      616630 0.7652039      805838
MCF7r2      593224 0.7361583      805838
ZR751       706836 0.8771445      805838
ZR752      2575408 3.1959394      805838
```

Other values show that the control reads were subtracted from the ChIP reads (this is done by default because no blacklists/greylists were applied, see Section 6 for more details).

Normalization of ChIP (and related assays such as ATAC) data is a crucial, if somewhat complex, area. Please see Section 7 for a more in-depth discussion of normalization in *DiffBind* .

3.5 Establishing a model design and contrast

Before running the differential analysis, we need to tell *DiffBind* how to model the data, including which comparison(s) we are interested in. This is done using the `dba.contrast` function, as follows:

```
> tamoxifen <- dba.contrast(tamoxifen,
+                           reorderMeta=list(Condition="Responsive"))
> tamoxifen

11 Samples, 2845 sites in matrix:
       ID Tissue Factor  Condition  Treatment Replicate   Reads FRiP
1  BT4741  BT474     ER  Resistant Full-Media         1  652697 0.16
2  BT4742  BT474     ER  Resistant Full-Media         2  663370 0.15
3   MCF71   MCF7     ER Responsive Full-Media         1  346429 0.31
4   MCF72   MCF7     ER Responsive Full-Media         2  368052 0.19
```

```
5   MCF73   MCF7     ER Responsive Full-Media         3  466273 0.25
6   T47D1   T47D     ER Responsive Full-Media         1  399879 0.11
7   T47D2   T47D     ER Responsive Full-Media         2 1475415 0.06
8  MCF7r1   MCF7     ER  Resistant Full-Media         1  616630 0.22
9  MCF7r2   MCF7     ER  Resistant Full-Media         2  593224 0.14
10  ZR751   ZR75     ER Responsive Full-Media         1  706836 0.33
11  ZR752   ZR75     ER Responsive Full-Media         2 2575408 0.22

Design: [~Condition] | 1 Contrast:
     Factor     Group Samples     Group2 Samples2
1 Condition Resistant       4 Responsive        7
```

This call will set up any "default" contrasts by examining the project metadata factors and assuming we want to look at the differences between any two sample groups with at least three replicates in each side of the comparison (that is, any factor that has two different values where there are at least three samples that share each value.) It also establishes the Responsive condition as the baseline, so it will be in the denominator of the default contrast. In the current case, there is only one such comparison that qualifies: The `Condition` metadata factor has two values, `Resistant` and `Responsive`, that have at least three replicates each (we see that there are four `Resistant` sample replicates and seven `Responsive` sample replicates.)

This function also establishes the default design, which includes only the metadata factor directly involved in the contrast (*Condition*).

While in this example we are using `dba.contrast` in the default mode, it does allow for fine-grained control over the design and contrasts one wishes to model. See Section 5 for a more detailed discussion of how including the `Tissue` factor in the design provides for better modeling of the example experiment.

3.6 Performing the differential analysis

The main differential analysis function is invoked as follows:

```
> tamoxifen <- dba.analyze(tamoxifen)
> dba.show(tamoxifen, bContrasts=TRUE)

     Factor     Group Samples     Group2 Samples2 DB.DESeq2
1 Condition Resistant       4 Responsive        7       246
```

This will run the default *DESeq2* analysis (see Section 10.3 discussing the technical details of the analysis). Displaying the results from the `DBA object` shows that 246 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, correlating only the 246 differentially bound sites identified by the analysis, as shown in Figure 3.

```
> 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, while the seven responsive form a separate cluster. Comparing Figure 2, which uses all 2845 consensus binding sites, with Figure 3,

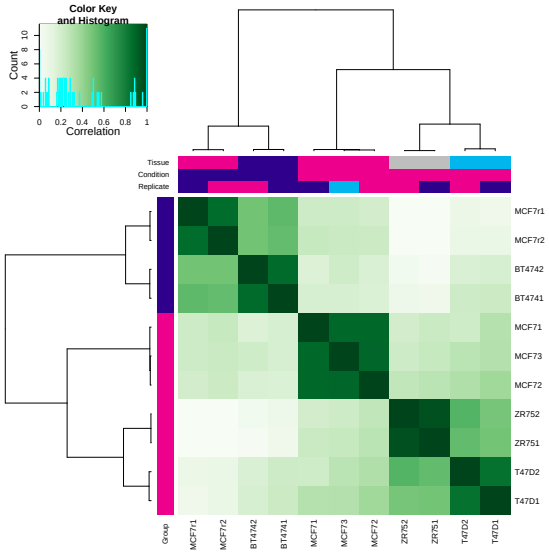


Figure 3: Correlation heatmap, using only significantly differentially bound sites. Generated by: `plot(tamoxifen,contrast=1)`; can also be generated by: `dba.plotHeatmap(tamoxifen,contrast=1)`

which uses only the 246 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.

See Section 5, where a multi-factor design is applied to this analysis, for a more sophisticated way to model these data.

3.7 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 246 ranges and 6 metadata columns:
       seqnames            ranges strand |      Conc Conc_Resistant
          <Rle>         <IRanges>  <Rle> | <numeric>      <numeric>
   976    chr18 26861047-26861447      * |   8.37281        4.44855
  2470    chr18 65030123-65030523      * |   6.02482        2.65613
  1484    chr18 41369550-41369950      * |   8.25529        5.12414
  2452    chr18 64490736-64491136      * |   7.42827        3.23712
  2338    chr18 60892950-60893350      * |   8.09133        3.44567
   ...      ...               ...    ... .       ...            ...
```

```
  2524    chr18 67565747-67566147      * |   3.80325        2.43917
  1221    chr18 33021825-33022225      * |   5.04267        2.81227
   433    chr18 11320959-11321359      * |   4.29417        5.15749
   235    chr18   7757228-7757628      * |   5.04773        6.13374
  1405    chr18 38482793-38483193      * |   4.23057        2.02778
       Conc_Responsive      Fold     p-value         FDR
             <numeric> <numeric>   <numeric>   <numeric>
   976         8.98991  -3.09374 1.81584e-08 3.36377e-05
  2470         6.62520  -2.86498 2.84202e-08 3.36377e-05
  1484         8.84621  -2.74884 3.82692e-08 3.36377e-05
  2452         8.05134  -3.08707 4.82434e-08 3.36377e-05
  2338         8.72229  -3.12556 7.73225e-08 4.31305e-05
   ...             ...       ...         ...         ...
  2524         4.23561  -1.30051  0.00423292   0.0486294
  1221         5.57838  -1.46474  0.00423698   0.0486294
   433         3.38332   1.31165  0.00429149   0.0490532
   235         3.56723   1.45105  0.00436302   0.0496672
  1405         4.76394  -1.45924  0.00440771   0.0499719
  -------
  seqinfo: 1 sequence from an unspecified genome; no seqlengths
```

The metadata columns show the mean read concentration over all the samples (the default calculation uses log2 normalized read counts) and the mean concentration over the samples in each of the first (Resistant) group and second (Responsive) group. The Fold column shows the log fold changes (LFCs) between the two groups, as calculated by the *DESeq2* analysis. A positive value indicates increased binding affinity in the Resistant group, and a negative value indicates 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 (also calculated by the *DESeq2* analysis).

We can compare the number of differentially bound sites that have enriched ER binding in the tamoxifen Resistant samples and those with enriched binding in the tamoxifen Responsive samples:

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

[1] 64

> sum(tamoxifen.DB$Fold<0)

[1] 182
```

The bias towards enriched binding in the Responsive case (or loss of binding affinity in the Resistant case) can be visualized using MA and Volcano plots, as shown in the following Section.

4 Plotting in *DiffBind*

Besides the correlation heatmaps we have been looking at, a number of other plots are available using the affinity data. This section covers Venn diagrams, MA plots, Volcano plots, PCA plots, Boxplots, and Heatmaps.

4.1 Venn diagrams

Venn diagrams illustrate overlaps between different sets of peaks. For example, amongst the differentially bound sites, we can see the differences between the "Gain" sites (those that increase binding enrichment in the Resistant condition) and the "Loss" sites (those with lower enrichment) as follows:

```
> dba.plotVenn(tamoxifen, contrast=1, bDB=TRUE,
+              bGain=TRUE, bLoss=TRUE, bAll=FALSE)
```

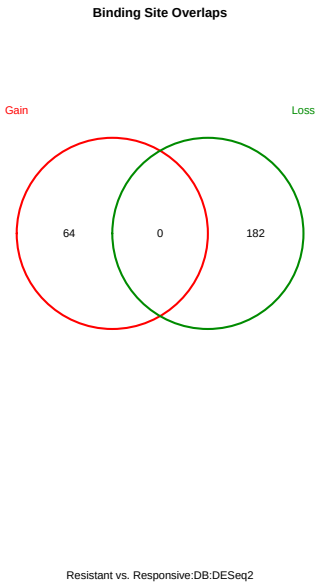


Figure 4: Venn diagram of Gain vs Loss differentially bound sites. Generated by: `dba.plotVenn(tamoxifen, contrast=1,bDB=TRUE,bGain=TRUE,bLoss=TRUE,bAll=FALSE)`

Figure 4 shows the result.

Venn diagrams are also useful for examining overlaps between peaksets, particularly when determining how best to derive consensus peaksets for read counting and further analysis. Section 8, which discusses consensus peaksets, shows a number of Venn plots in context, and the help page for `dba.plotVenn` has a number of additional examples.

4.2 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 2, which includes normalized read counts for all 2845 binding sites, can be obtained as follows:

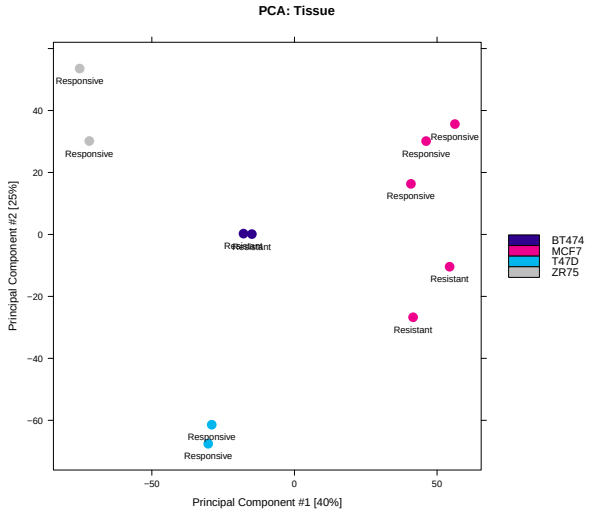


Figure 5: PCA plot using affinity data for all sites. Generated by: `dba.plotPCA(tamoxifen,DBA_TISSUE,label=DBA_CONDITION)`

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

The resulting plot (Figure 5) shows all the MCF7-derived samples (red) clustering on one side of the first (horizontal) component, with the Responsive and Resistant samples not separable either in the first nor in the second (vertical) component.[3]

A PCA plot using only the 246 differentially bound sites (corresponding to Figure 3), using an FDR threshold of 0.05, can be drawn as follows:

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

This plot (Figure 6) shows how the differential analysis identifies sites that can be used to separate the Resistant and Responsive sample groups along the first component.

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,contrast=1, b3D=TRUE)`. Seeing the first three principal components can be a useful exploratory exercise.

4.3 MA plots

MA plots are a useful way to visualize the relationship between the overall binding level at each site and the magnitude of the change in binding enrichment between conditions, as well as the effect of normalization on data. An MA plot can be obtained for the Resistant vs Responsive contrast as follows:

[3]Note that they are separable in the second and third components; try `dba.plotPCA(tamoxifen, DBA_CONDITION,label=DBA_TISSUE,components=2:3)`

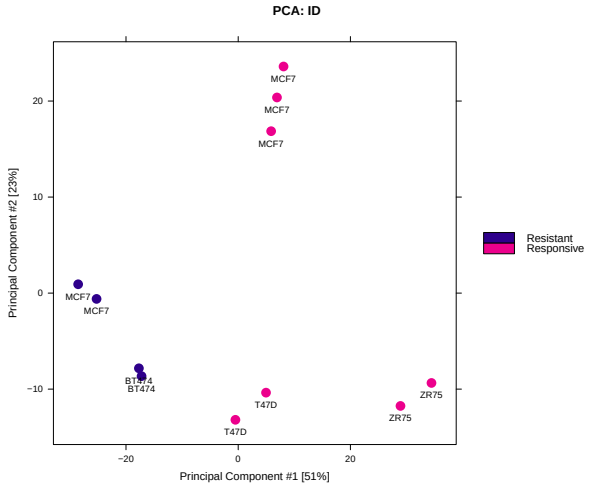


Figure 6: PCA plot using affinity data for only differentially bound sites. Generated by: `dba.plotPCA(tamoxifen,contrast=1,label=DBA_TISSUE)`

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

The plot is shown in Figure 7. Each point represents a binding site, with the 246 points in magenta representing sites identified as differentially bound. There is a blue horizontal line through the origin (0 LFC), as well as a horizontal red curve representing a non-linear loess fit showing the underlying relationship between coverage levels and fold changes. The plot shows how the differentially bound sites appear to have a minimum absolute log fold difference of somewhere between one and two. As we have already seen, it also shows 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)`.

Section 7 contains several examples of MA plots, including showing non-normalized data, and the ability to plot any subset of samples against any other set of sample.

4.4 Volcano plots

Similar to MA plots, Volcano plots also highlight significantly differentially bound sites and show their fold changes. Here, however, the confidence statistic (FDR or p-value) is shown on a negative log scale, helping visualize the relationship between the magnitude of fold changes and the confidence that sites are differentially bound.

For example, the same data as plotted in Figure 7 can be visualized as a volcano plot:

```
> dba.plotVolcano(tamoxifen)
```

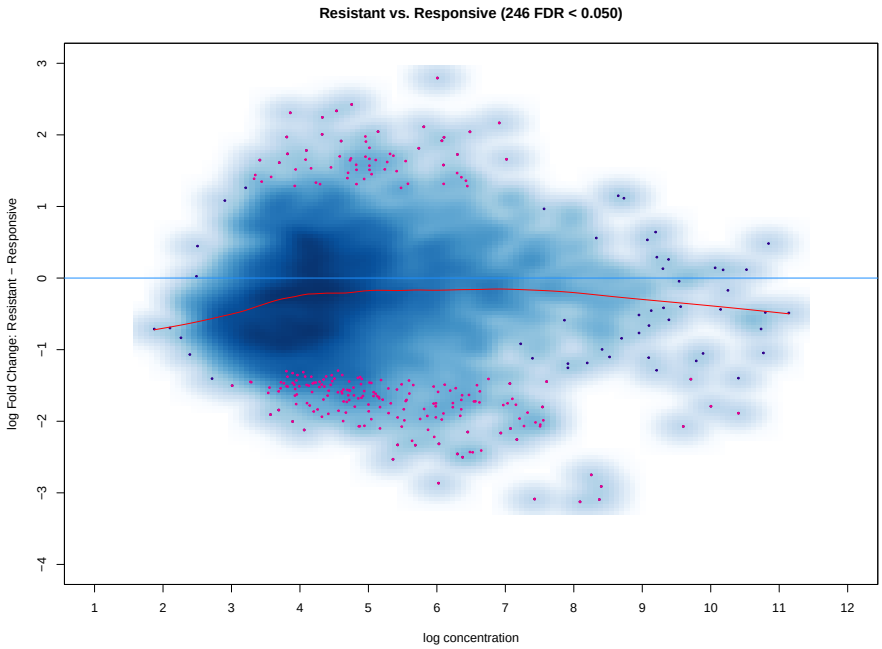


Figure 7: MA plot of Resistant-Responsive contrast. Sites identified as significantly differentially bound shown in red. Generated by: `dba.plotMA(tamoxifen)`

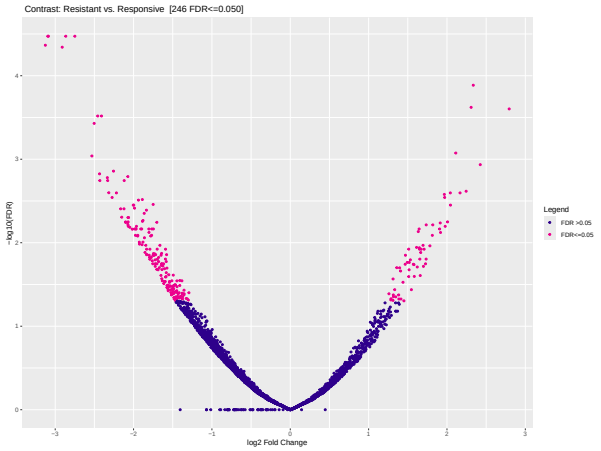


Figure 8: Volcano plot of Resistant-Responsive contrast. Sites identified as significantly differentially bound shown in red. Generated by: `dba.plotVolcano(tamoxifen)`

The plot is shown in Figure 8, with the predominance of lower binding in the Resistant case evidenced by the greater number of significant sites on the negative side of the Fold Change (X) axis.

4.5 Boxplots

Boxplots provide a way to view how read distributions differ between classes of binding sites. Consider the example, where 246 differentially bound sites are identified. The MA plot (Figure 7) 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$Fold<0)
[1] 182
> sum(tamoxifen.DB$Fold>0)
[1] 64
```

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)
```

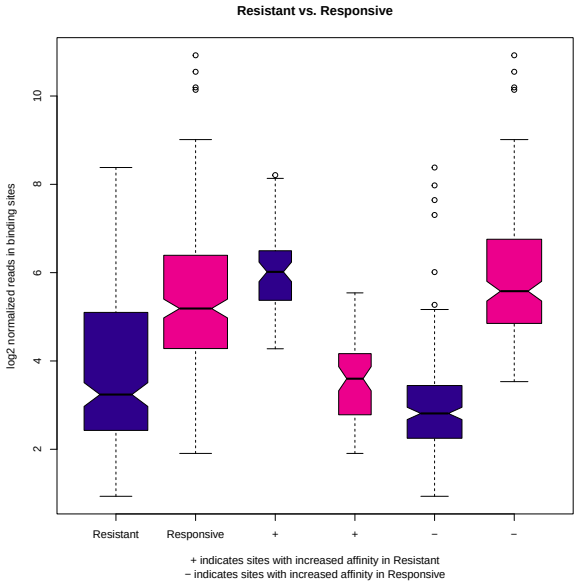


Figure 9: Box plots of read distributions for significantly differentially bound (DB) sites. Tamoxifen Resistant samples are shown in blue, and Responsive samples are shown in red. Left two boxes show distribution of reads over all DB sites in the Resistant and Responsive groups; middle two boxes show distributions of reads in DB sites that increase in affinity in the Resistant group; last two boxes show distributions of reads in DB sites that increase in affinity in the Responsive group. Generated by: `dba.plotBox(tamoxifen)`

The default plot (Figure 9) 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 Resistant samples, while the final two boxes show the distribution of reads in differentially bound sites that exhibit increased affinity in the Responsive 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       1.00e+00      2.69e-17      2.08e-17       9.01e-01
Responsive.DB      2.69e-17      1.00e+00      3.40e-05       2.44e-15
Resistant.DB+      2.08e-17      3.40e-05      1.00e+00       3.61e-12
Responsive.DB+     9.01e-01      2.44e-15      3.61e-12       1.00e+00
Resistant.DB-      1.82e-05      0.00e+00      0.00e+00       4.84e-05
Responsive.DB-     9.89e-30      6.34e-05      1.32e-01       4.89e-25
               Resistant.DB- Responsive.DB-
Resistant.DB        1.82e-05       9.89e-30
Responsive.DB       0.00e+00       6.34e-05
Resistant.DB+       0.00e+00       1.32e-01
Responsive.DB+      4.84e-05       4.89e-25
Resistant.DB-       1.00e+00       1.30e-31
Responsive.DB-      1.30e-31       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=2.69e-17, while those between the Resistant and Responsive groups in the individual cases (increased in Resistant or increased in Responsive) have p-values computed as 3.61e-12 and 1.30e-31.

4.6 Heatmaps

DiffBind provides two types of heatmaps. This first, correlation heatmaps, we have already seen. For example, the heatmap shown in Figure 2 can be generated as follows:

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

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_NORMALIZED`, uses the normalized read counts (see Section 7). 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*, showing the read scores for some or all of the binding sites. This can be plotted for the example case as follows:

```
> hmap <- colorRampPalette(c("red", "black", "green"))(n = 13)
> readscores <- dba.plotHeatmap(tamoxifen, contrast=1, correlations=FALSE,
+                               scale="row", colScheme = hmap)
```

Figure 10 shows the affinities and clustering of the differentially bound sites (rows), as well as the sample clustering (columns). In this case, the (normalized) counts have been row scaled, and a red/green heatmap color palette applied.

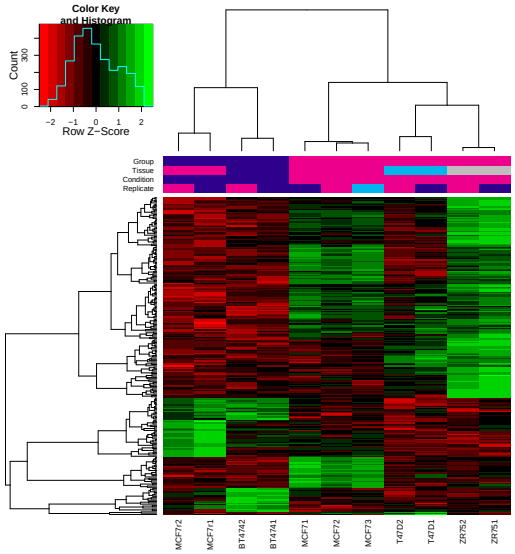


Figure 10: Binding affinity heatmap showing affinities for differentially bound sites. Samples cluster first by whether they are responsive to tamoxifen treatment, then by cell line, then by replicate. Clusters of binding sites show distinct patterns of affinity levels. Generated by: `dba.plotHeatmap(tamoxifen,contrast=1,correlations=FALSE)`

4.7 Profiling and Profile Heatmaps

The `dba.plotProfile()` function enables the computation of peakset profiles and the plotting of complex heatmaps. Profiles are computed using `EnrichedHeatmap::normalizeToMatrix()` and plotted using the *EnrichedHeatmap* and *ComplexHeatmap* packages, which must be installed in order to use this function.

Processing proceeds in two phases.

In the first phase, specific peaksets are extracted from a *DiffBind* `DBA object` and profiles are calculated for these peaks for set of samples in the *DiffBind* experiment. Profiles are calculated by counting the number of overlapping reads in a series of bins upstream and downstream of each peak center.

In the second phase, the derived profiles are plotted in a series of complex heatmaps showing the relative intensity of overlapping peaks in each bin for each peak in each sample, along with summary plots showing the average profile across the sites for each sample.

Due to the computational cost of this function, it is advised that the calculation of profiles and the plotting be separated into two calls, so that the profiles do not need to be re-generated if something goes wrong in the plotting. By default, when a `DBA object` is passed in to generate profiles, plotting is turned off and the computed profiles are returned. When `dba.plotProfile()` is called with a previously-computed set of profiles, a plot is generated by default.

The main aspects of the profile plot are which **samples** are plotted (the X-axis) and which **sites** are plotted (the Y-axis). These can be specified in a number of flexible ways. Other parameters to `dba.plotProfile()` determine how the data are treated, controlling aspects such as how many sites are included in the plot, data normalization, sample merging (computing mean profiles for groups of samples), and control over the appearance of the plot.

While few brief example and included here, a more complete document is available showing more of the various options. The markdown source for this can be accessed as `system.file('extra/plotProfileDemo.Rmd',package='DiffBind')`; an html version is available for browsing at https://content.cruk.cam.ac.uk/bioinformatics/software/DiffBind/plotProfileDemo.html, and a pdf document can be found at https://content.cruk.cam.ac.uk/bioinformatics/software/DiffBind/plotProfileDemo.pdf

4.7.1 Default profile plot

If an analysis has been completed, the default plot will be based on the results of the first contrast. If the contrast compares two conditions, all of the samples in each condition will be included, with the heatmaps colored separately for samples in each contrast condition.

Sample groups are merged based on the `DBA_REPLICATE` attribute, such that each sample class will have one heatmap based on the normalized mean read counts for all the samples in that class that have the same metadata except for the Replicate number.

In terms of sites, two groups of differentially bound sites are included: **Gain** sites (with positive fold change) and **Loss** sites (negative fold change). If there are more than 1,000 sites in either category, the 1,000 sites with the great absolute value fold change will be included (the maximum number of sites to be profiled can be altered).

```
> profiles <- dba.plotProfile(tamoxifen)
> dba.plotProfile(profiles)
```

This plot shows how the differentially bound sites are divided into Gain and Loss groups, and how sample groups belonging to each of the two contrast conditions (Resistant and Responsive) result in differently colored heatmaps.

4.7.2 Merging all samples in a contrast condition

In the sample experiment, there are multiple sample groups comprising each side of the contrast: the Resistant class has two sample groups based on the BT474 and MCF7 cell lines, while the Responsive class has three groups, based on the MCF7, T47D, and ZR75 cell lines. If we want to generate composite profiles for the Responsive and Resistant classes, the `DBA_TISSUE` attribute can be added to the merge specification. By specifying `merge=c(DBA_TISSUE, DBA_REPLICATE)`, the samples are divided into groups each with the same metadata values *except* for the Replicate and Tissue factors. Samples within each group are merged, so that the (normalized) mean counts for all of the Resistant samples will calculated, as well as for all of the Responsive samples:

```
> profiles <- dba.plotProfile(tamoxifen,merge=c(DBA_TISSUE, DBA_REPLICATE))
> dba.plotProfile(profiles)
```

4.7.3 Avoiding merging to show all sample replicates

Masks can be used to specify which samples we want to include in the plot. For example, to see each of the MCF7 samples separately, divided into contrast groups, specify the samples as a list of two sample masks, one combining `MCF7` with `Resistant`, and one combining `MCF7` with `Responsive`. Further specifying `merge=NULL` will prevent the replicates from being merged, so profiles for each replicate will be computed and plotted:

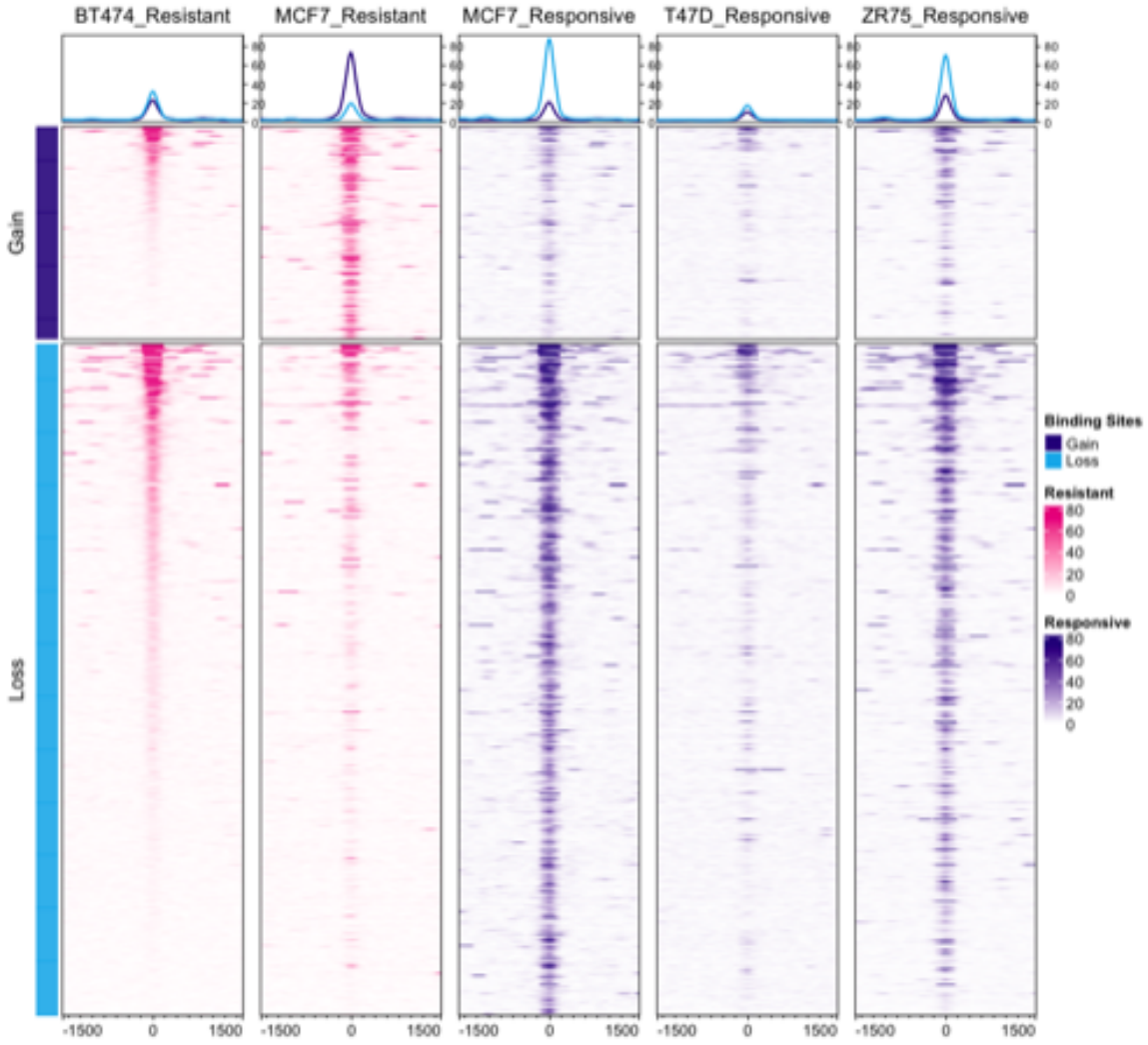


Figure 11: Default `dba.plotProfile()` plot, showing each sample group, colored by contrast condition (Resistant or Responsive).

```
> mask.MCF7 <- tamoxifen$masks$MCF7
> mask.Resistant  <- tamoxifen$masks$Resistant
> mask.Responsive <- tamoxifen$masks$Responsive
> profiles <- dba.plotProfile(tamoxifen,
+                             samples=list(MCF7_Resistant=
+                                            mask.MCF7 & mask.Resistant,
+                                          MCF7_Responsive=
+                                            mask.MCF7 & mask.Responsive),
+                             merge=NULL)
> dba.plotProfile(profiles)
```

Further examples can be found in the `plotProfileDemo` notebook.

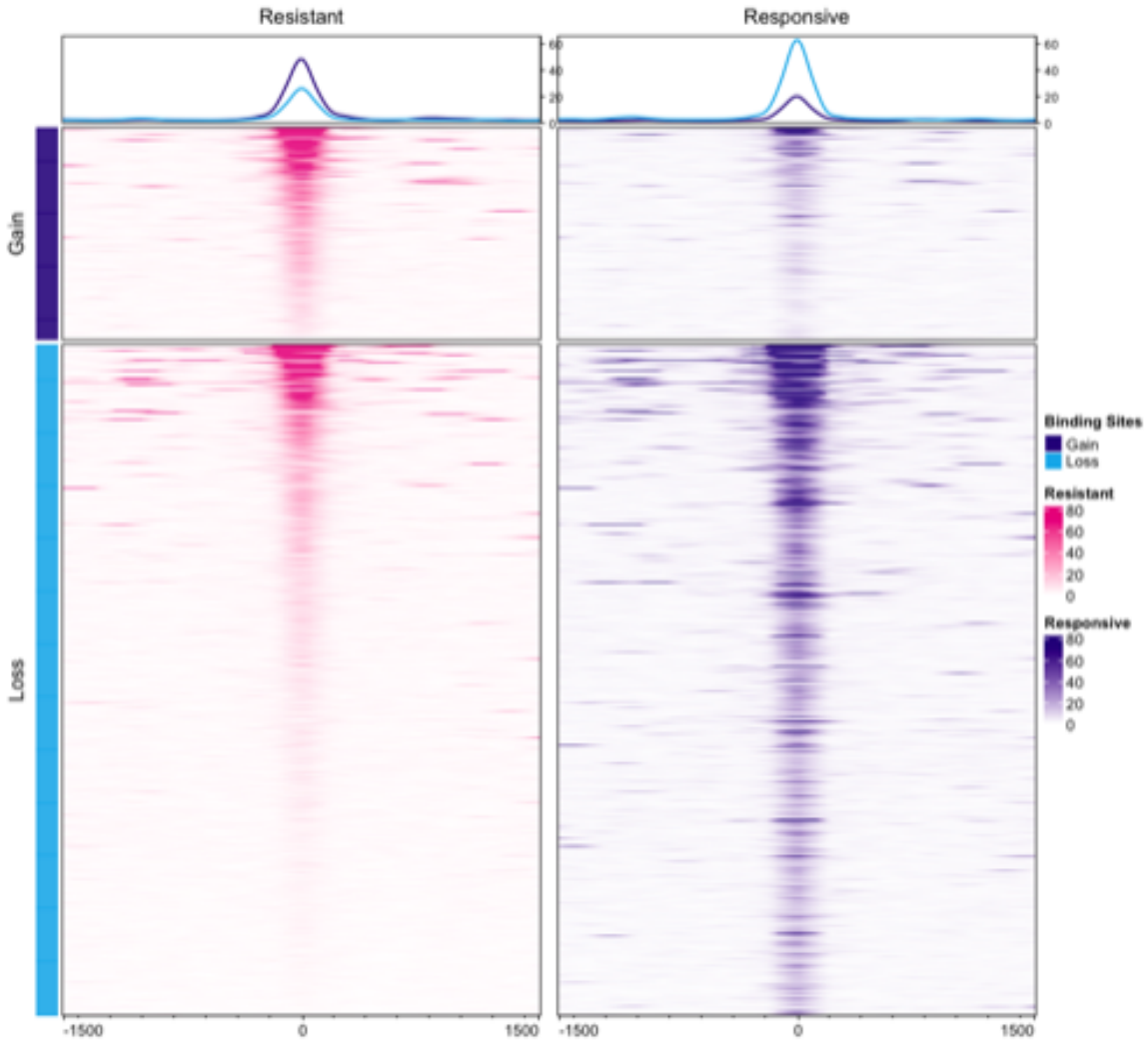


Figure 12: Default `dba.plotProfile()` plot with sample groups merged, showing mean signal for all Resistant and Responsive samples.

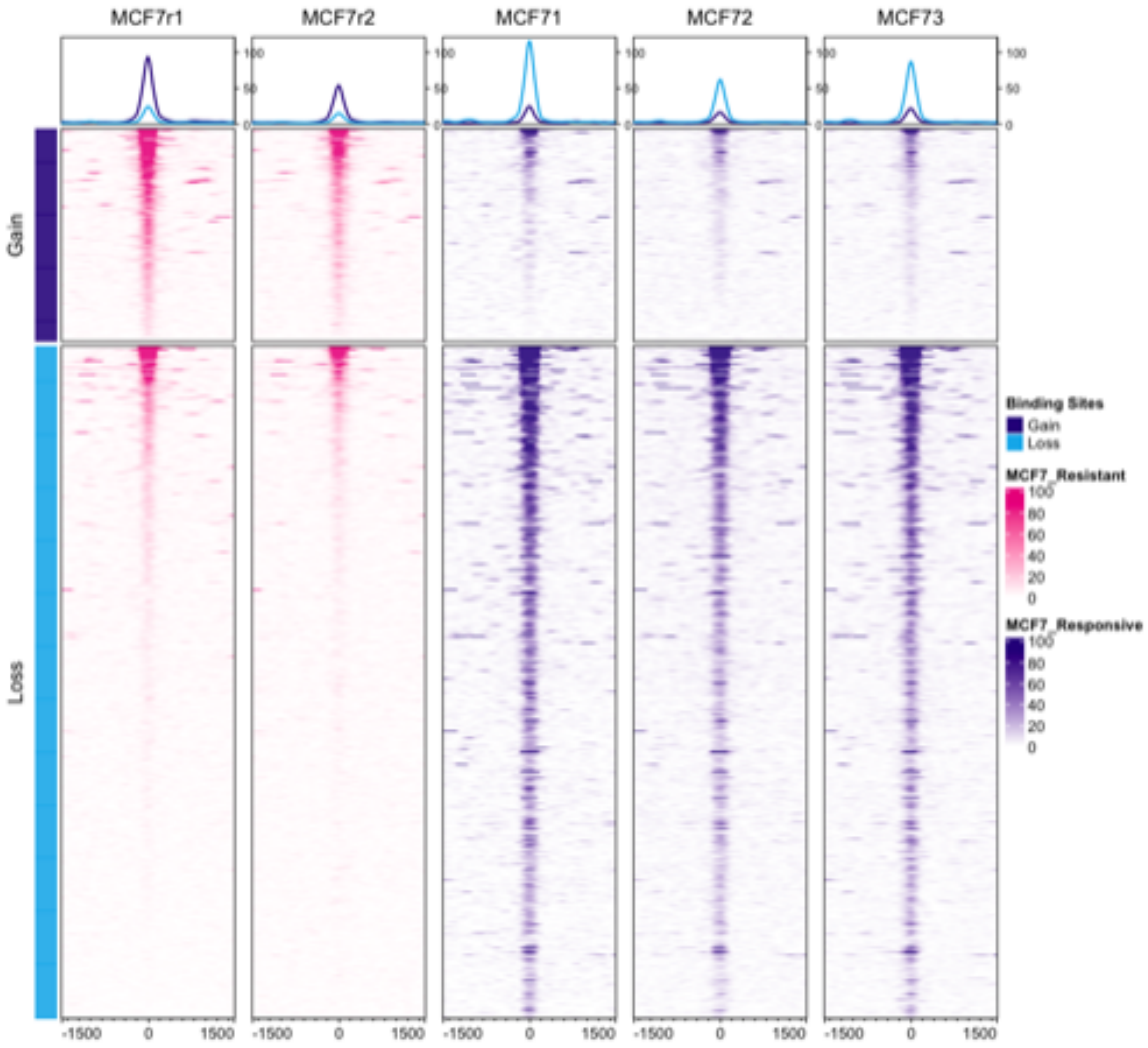


Figure 13: Default `dba.plotProfile()` plot showing Gain and Loss sites in all MCF7 sample replicates.

5 Example: Multi-factor designs

The previous discussion showed how to perform a differential binding analysis using a single factor (`Condition`) with two values (`Resistant` and `Responsive`); that is, finding the significantly differentially bound sites between two sample groups. This section extends the example by including other factors in the design.

One example of this is where there is a second factor, potentially with multiple values, that represents a confounding condition. Examples include cases 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). More complex designs can also include specific interactions between factors.

In the current example, the effect we want to control for is the different cell culture strains used, in particular the fact that some samples in both of the sample groups, based on different `Condition` values (one tamoxifen responsive and one tamoxifen resistant), are both derived from the same `Tissue` (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 2).

Using the generalized linear modeling (GLM) functionality included in *DESeq2* and *edgeR* , the confounding factor can be modeled enabling more sensitive results. This is done by specifying a design formula explicitly to `dba.contrast`.

```
> tamoxifen <- dba.contrast(tamoxifen,design="~Tissue + Condition")
```

Note that by changing the design formula, the previous results are cleared, requiring the analysis to be run anew:

```
> tamoxifen <- dba.analyze(tamoxifen)
> dba.show(tamoxifen, bContrasts=TRUE)

     Factor     Group Samples     Group2 Samples2 DB.DESeq2
1 Condition Resistant       4 Responsive        7       783
```

This shows that where the standard, single-factor *DESeq2* analysis identifies 246 differentially bound sites, the analysis using the two-factor design finds 783 such sites. MA and Volcano plots show how the analysis has changed:

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

```
> dba.plotVolcano(tamoxifen)
```

The resulting plots are shown in Figure 14 and Figure 15 Comparing these to Figure 7 and Figure 8, a number of differences can be observed. The analysis has become more sensitive, with sites being identified as significantly differentially bound with lower magnitude fold changes. Secondly, the distribution of differentially bound sites has shifted. In the single-factor analysis, they were mostly concentrated in the lower left of the MA plot, identifying sites with increased binding in the Responsive condition, but with relatively low concentrations. In the multi-factor analysis, sites are identified as being significantly differentially bound

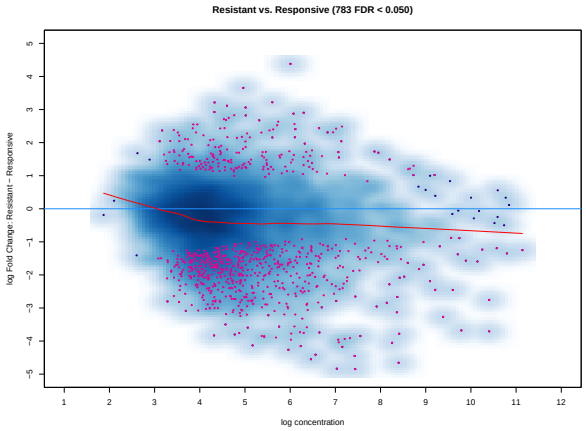


Figure 14: MA plot of Resistant-Responsive contrast, using a multi-factor design " Tissue + Condition". Sites identified as significantly differentially bound shown in magenta. Generated by: `dba.plotMA(tamoxifen)`

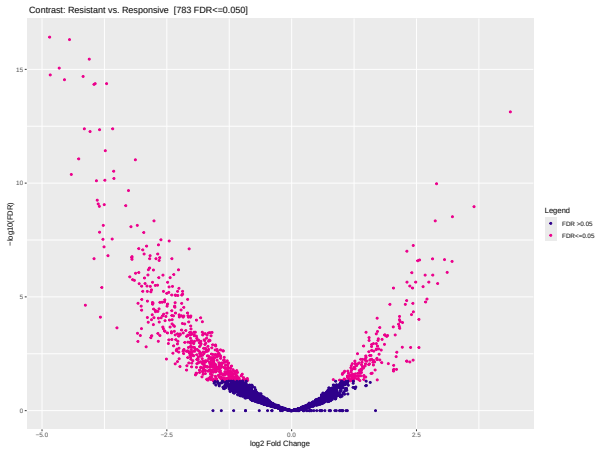


Figure 15: Volcano plot of Resistant-Responsive contrast, using a multi-factor design " Tissue + Condition". Sites identified as significantly differentially bound shown in magenta. Generated by: `dba.plotVolcano(tamoxifen)`

across a wider range of concentrations and more are identified that gain binding affinity in the Resistant condition. Finally, the horizontal red loess fit line appears to better fit the calculated fold changes while showing an overall slight loss of binding enrichment.

These observations can be quantified from the report data:

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

Looking at sensitivity, we can compare the distribution of fold changes of the differentially bound sites identified by the single- and multi-factor analyses:

```
> min(abs(tamoxifen.DB$Fold))

[1] 1.260171

> min(abs(multifactor.DB$Fold))
```

```
[1] 0.8344149
```

Likewise, we can compare the proportions of sites identified as being differentially bound between those that gain binding enrichment in the Resistant condition over those more enriched in the Responsive conditions, between the single- and multi-factor analyses:

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

[1] 0.3516484

> sum(multifactor.DB$Fold > 0) / sum(multifactor.DB$Fold < 0)

[1] 0.3159664
```

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

```
> tamoxifen <- dba.analyze(tamoxifen,method=DBA_ALL_METHODS)
> dba.show(tamoxifen,bContrasts=TRUE)

     Factor     Group Samples     Group2 Samples2 DB.edgeR DB.DESeq2
1 Condition Resistant       4 Responsive        7      819       783
```

We see that *edgeR* identifies a somewhat higher number of sites than *DESeq2* . You can check this by looking at the identified sites using `dba.report`, and performing MA, Volcano, 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.plotVenn` function that shows the overlaps of contrast results:

```
> tamoxifen.OL <- dba.plotVenn(tamoxifen,contrast=1,method=DBA_ALL_METHODS,
+                              bDB=TRUE)
```

The overlap is shown in Figure 16. The largest group of sites are identified by both *edgeR* and *DESeq2* . Note that the binding sites unique to *edgeR* , *DESeq2* , and common to both are returned in the variable `tamoxifen.OL`.

To further illustrate the ability to model the data by evaluating multiple contrasts against a single model, consider another comparison. Suppose we'd like to identify ER binding sites that a differentially bound in all of the MCF7 samples compared to the T47D samples[4]. We can do this by adding a contrast to our existing model. For illustrative purposes, we will also re-order the values for the `Tissue` factor to make MCF7 the reference group:

```
> tamoxifen <- dba.contrast(tamoxifen,contrast=c("Tissue","MCF7","T47D"),
+                           reorderMeta = list(Tissue="MCF7"))
> tamoxifen <- dba.analyze(tamoxifen,method=DBA_ALL_METHODS)
> dba.show(tamoxifen,bContrasts=TRUE)

     Factor     Group Samples     Group2 Samples2 DB.edgeR DB.DESeq2
1 Condition Resistant       4 Responsive        7      819       783
2    Tissue      MCF7       5       T47D        2     1518      1470
```

[4]We consider this example again in Section 7.4.

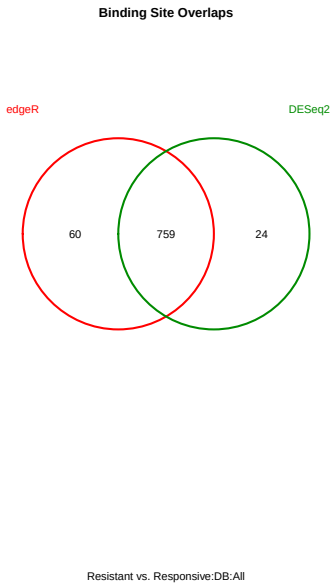


Figure 16: Venn diagram showing overlap of differentially bound peaks identified using *edgeR* and *DESeq2* to do a multi-factor analysis. Generated by plotting the result of: `dba.plotVenn(tamoxifen,contrast=1,method=DBA_ALL_METHODS,bDB=TRUE)`

6 Blacklists and Greylists

Good practice in analyzing ChIP-seq (and ATAC-seq) experimental data include the use of **blacklists** to remove certain regions in the reference genome from the analysis[3]. This section describes how to accomplish this for both publicly published blacklists as well as experiment-specific **greylists**.

6.1 What are blacklists and greylists?

Blacklists are pre-defined lists of regions specific to a reference genome that are known to be problematic. The best known lists have been identified as part of the ENCODE project[3] and are available for a variety of reference genomes and genome versions. The current ENCODE blacklists are available through the `dba.blacklist` function.

Greylists are specific to a ChIP-seq experiment, and are derived from the controls generated as part of the experiment[4]. The idea is to analyze libraries that are not meant to show systematic enrichment (such as Inputs, in which no anti-body is introduced), and identify anomalous regions where a disproportionate degree of signal is present. These regions can then be excluded from subsequent analysis.

6.2 Why apply blacklists and greylists?

Application of blacklists prevents identification of problematic regions in the reference genome as being differentially bound. The regions tend to be ones with a high degree of repeats or unusual base concentrations. Application of greylists prevents identification of problematic genomic regions in the materials used in the experiment as being differentially bound. For example, these could include areas of high copy-number alterations in a cell line.

It has been shown that problematic reads, such as duplicate reads, are disproportionately likely to overlap blacklisted regions, and the quality of experimental data can be increased reads in blacklisted regions are excluded[5].

Another way of thinking about greylists is that they are one way of using the information in the control tracks to improve the reliability of the analysis. Prior to version 3.0, the default in *DiffBind* has been to simply subtract control reads from ChIP reads in order to dampen the magnitude of enrichment in anomalous regions. Greylists represent a more principled way of accomplishing this. If a greylist has been applied, the current default in *DiffBind* is to *not* subtract control reads.

6.3 When should blacklists and greylists be applied?

Within *DiffBind*, blacklists and greylists are applied to candidate peak regions prior to performing a quantitative analysis. This should be done before calculating a consensus peakset by excluding blacklisted peaks from each individual peakset. It can also be done after counting overlapping reads by excluding consensus peaks that overlap a blacklisted or greylists region (see examples below).

It is worth noting that, ideally, blacklists and greylists would be applied earlier in the process, to the aligned reads (bam files) themselves, prior to any peak calling. Popular peak callers, such as MACS, use the control tracks to model the background noise levels which plays a critical role in identifying truly enriched "peak" regions. Excluding the blacklisted reads prior to peak calling should result in more accurate identification of enriched regions in the non-blacklisted areas of the genome. The `dba.blacklist` function offers a way to easily retrieve any blacklists and computed greylists, enabling the ability to go back and re-process the data with blacklisted reads removed prior to the peak-calling step.

6.4 Example: How to apply a blacklist

The primary function that controls blacklists and greylists is called `dba.blacklist`. In order to use this, you either need to have an existing blacklist, or know the reference genome used to align your sequencing data. In the example dataset, the reference is Hg19.

ENCODE blacklists can be applied straightforwardly as follows:[5]

```
> data(tamoxifen_peaks)
> tamoxifen

11 Samples, 2845 sites in matrix (3795 total):
       ID Tissue Factor  Condition  Treatment Replicate Intervals
1  BT4741  BT474     ER  Resistant Full-Media         1      1080
2  BT4742  BT474     ER  Resistant Full-Media         2      1122
3   MCF71   MCF7     ER Responsive Full-Media         1      1556
4   MCF72   MCF7     ER Responsive Full-Media         2      1046
5   MCF73   MCF7     ER Responsive Full-Media         3      1339
6   T47D1   T47D     ER Responsive Full-Media         1       527
7   T47D2   T47D     ER Responsive Full-Media         2       373
8  MCF7r1   MCF7     ER  Resistant Full-Media         1      1438
9  MCF7r2   MCF7     ER  Resistant Full-Media         2       930
```

[5]While in this example the blacklist used is specified explicitly, *DiffBind* can usually determine the correct blacklist to apply using the default `blacklist=TRUE`

```
10  ZR751   ZR75     ER Responsive Full-Media         1      2346
11  ZR752   ZR75     ER Responsive Full-Media         2      2345
```

```
> peakdata  <- dba.show(tamoxifen)$Intervals
> tamoxifen <- dba.blacklist(tamoxifen, blacklist=DBA_BLACKLIST_HG19,
+                            greylist=FALSE)
> tamoxifen
```

```
11 Samples, 2844 sites in matrix (3794 total):
       ID Tissue Factor  Condition  Treatment Replicate Intervals
1  BT4741  BT474     ER  Resistant Full-Media         1      1080
2  BT4742  BT474     ER  Resistant Full-Media         2      1122
3   MCF71   MCF7     ER Responsive Full-Media         1      1555
4   MCF72   MCF7     ER Responsive Full-Media         2      1045
5   MCF73   MCF7     ER Responsive Full-Media         3      1338
6   T47D1   T47D     ER Responsive Full-Media         1       527
7   T47D2   T47D     ER Responsive Full-Media         2       373
8  MCF7r1   MCF7     ER  Resistant Full-Media         1      1438
9  MCF7r2   MCF7     ER  Resistant Full-Media         2       930
10  ZR751   ZR75     ER Responsive Full-Media         1      2346
11  ZR752   ZR75     ER Responsive Full-Media         2      2345
```

```
> peakdata.BL <- dba.show(tamoxifen)$Intervals
> peakdata - peakdata.BL
```

```
 [1] 0 0 1 1 1 0 0 0 0 0 0
```

This shows that a single peak was excluded from each of the three MCF7 (Responsive) replicates.

Alternatively, the blacklist could be applied *after* composing a consensus peakset and counting reads. This is useful to see its impact on the analysis.

Remembering our earlier analysis, with the results stored in `multifactor.DB`:

```
> length(multifactor.DB)
```

```
[1] 783
```

```
> data(tamoxifen_counts)
> tamoxifen   <- dba.blacklist(tamoxifen, blacklist=DBA_BLACKLIST_HG19,
+                              greylist=FALSE)
> blacklisted <- dba.blacklist(tamoxifen, Retrieve=DBA_BLACKLISTED_PEAKS)
> tamoxifen   <- dba.contrast(tamoxifen, design="~Tissue + Condition")
> tamoxifen   <- dba.analyze(tamoxifen)
> blacklisted.DB <- dba.report(tamoxifen)
> length(blacklisted.DB)
```

```
[1] 773
```

This shows that the peak interval excluded by the blacklist was returned as a significantly differentially bound site in the original analysis, but is absent in the analysis performed after blacklisting:

```
> bl_site <- match(blacklisted[[1]], multifactor.DB)
> multifactor.DB[bl_site,]
```

```
GRanges object with 1 range and 6 metadata columns:
    seqnames        ranges strand |      Conc Conc_Resistant Conc_Responsive
       <Rle>     <IRanges>  <Rle> | <numeric>      <numeric>       <numeric>
  2    chr18 111395-111795      * |   6.47225         5.6949          6.7803
         Fold    p-value       FDR
    <numeric>  <numeric> <numeric>
  2  -1.18129 0.00518054 0.0229707
  -------
  seqinfo: 1 sequence from an unspecified genome; no seqlengths
```

```
> is.na(match(blacklisted[[1]], blacklisted.DB))
```

```
[1] TRUE
```

6.5 Example: How to apply a greylist

Using greylists is somewhat more complicated. If they have been pre-computed from control tracks, either using the *GreyListChIP* package or from a previous run of *DiffBind* , they can be supplied directly to `dba.blacklist`. A pre-computed greylist for the sample experiment has been included with the *DiffBind* package:

```
> data(tamoxifen_greylist)
> names(tamoxifen.greylist)
```

```
[1] "master"   "controls"
```

```
> tamoxifen.greylist$master
```

```
GRanges object with 69 ranges and 0 metadata columns:
       seqnames            ranges strand
          <Rle>         <IRanges>  <Rle>
   [1]    chr18        9217-11264      *
   [2]    chr18     105473-112128      *
   [3]    chr18     267265-268800      *
   [4]    chr18     402433-403456      *
   [5]    chr18     504832-505855      *
   ...      ...               ...    ...
  [65]    chr18 61089131-61090154      *
  [66]    chr18 71815505-71816528      *
  [67]    chr18 74060619-74061642      *
  [68]    chr18 76774213-76777284      *
  [69]    chr18 77377859-77380930      *
  -------
  seqinfo: 1 sequence from hg19 genome
```

The greylist object is a list with two elements. The first, `tamoxifen.greylist$master`, contains the full greylist to be applied. The other element, `tamoxifen.greylist$controls`, is a `GRangesList` containing the individual greylists computed for each of the five control tracks:

```
> names(tamoxifen.greylist$controls)
```

```
[1] "BT474c" "MCF7c"  "T47Dc"  "TAMRc"  "ZR75c"
```

```
> tamoxifen.greylist$controls

GRangesList object of length 5:
$BT474c
GRanges object with 58 ranges and 0 metadata columns:
       seqnames            ranges strand
          <Rle>         <IRanges>  <Rle>
   [1]    chr18     105985-111616      *
   [2]    chr18     267265-268800      *
   [3]    chr18     402433-403456      *
   [4]    chr18     504832-505855      *
   [5]    chr18     568320-569343      *
   ...      ...               ...    ...
  [54]    chr18 61089131-61090154      *
  [55]    chr18 71815505-71816528      *
  [56]    chr18 74060619-74061642      *
  [57]    chr18 76774213-76775748      *
  [58]    chr18 77378371-77380930      *
  -------
  seqinfo: 1 sequence from hg19 genome

...
<4 more elements>
```

These greylists can be re-combined if the controls are re-used in other experiments.

The greylist can be applied (along with the blacklist) as follows:

```
> data(tamoxifen_peaks)
> tamoxifen <- dba.blacklist(tamoxifen, blacklist=DBA_BLACKLIST_HG19,
+                            greylist=tamoxifen.greylist)
```

For this example, we can apply it to the binding matrix after computing a consensus peakset and counting overlapping reads:

```
> data(tamoxifen_counts)
> cons.peaks <- dba.show(tamoxifen)$Intervals[1]
> tamoxifen  <- dba.blacklist(tamoxifen, blacklist=DBA_BLACKLIST_HG19,
+                             greylist=tamoxifen.greylist)
> cons.peaks.grey <- dba.show(tamoxifen)$Intervals[1]
> cons.peaks - cons.peaks.grey

[1] 51
```

Some 51 peaks that overlap the greylist have been excluded from the consensus peakset (representing 1.8% of the total).

6.6 Example: How to compute a greylist with *GreyListChIP*

Most of the time, if your experiment has controls such as Input tracks, the greylist needs to be computed specifically for the analysis. This can be accomplished using `dba.blacklist` to invoke the *GreyListChIP* package automatically.

Here is how the sample greylist was computed for this example. You can execute this code if you have all the read data for the vignette (see Section 12).

Instead of specifying a greylist explicitly, if the `greylist` is set to `TRUE`, the genome will be determined (if possible) automatically from the control bam files associated with the experiment. Alternatively, a specific reference genome can be supplied, either using a constant provided by *DiffBind* (such as `DBA_BLACKLIST_HG19`), or the name of a *BSgenome* object (such as `"BSgenome.Hsapiens.UCSC.hg19"`). In these cases, the *GreyListChIP* package is invoked to compute a greylist for each control in the experiment, as well as a merged "master" greylist of all the overlapping regions of all the control greylists.

In the current example we can use the following code. (Note that even for the sample data, this can take a substantial amount of time to run, as long or longer than running `dba.count`):

```
> tamoxifen <- dba(sampleSheet="tamoxifen.csv")
> tamoxifen <- dba.blacklist(tamoxifen)
> tamoxifen.greylist <- dba.blacklist(tamoxifen, Retrieve=DBA_GREYLIST)
```

The output for this command appears as follows:

```
Genome detected: Hsapiens.UCSC.hg19
Applying blacklist...
Removed: 3 of 14102 intervals.
Building greylist: BT474c
coverage: 166912 bp (0.21%)
Building greylist: MCF7c
coverage: 106495 bp (0.14%)
Building greylist: T47Dc
coverage: 56832 bp (0.07%)
Building greylist: TAMRc
coverage: 122879 bp (0.16%)
Building greylist: ZR75c
coverage: 68608 bp (0.09%)
BT474c: 58 ranges, 166912 bases
MCF7c: 14 ranges, 106495 bases
T47Dc: 11 ranges, 56832 bases
TAMRc: 10 ranges, 122879 bases
ZR75c: 12 ranges, 68608 bases
Master greylist: 69 ranges, 251391 bases
Removed: 423 of 14102 intervals.
```

Note that the sensitivity of how much is excluded can be controlled by a configuration parameter. The default is `0.999`:

```
> tamoxifen$config$greylist.pval <- 0.999
```

Altering this can cause more or less of the genome to be excluded for an experiment. See the documentation for *GreyListChIP* for more details.

7 Normalization

Normalization of experimental data is particularly important in ChIP-seq (and ATAC-seq) analysis, and may require more careful consideration than needed for RNA-seq analysis. This is because the range of ChIP-seq experiments covers more cases than RNA-seq, which usually involve a similar set of possible expressed genes and/or transcripts, many of which are not expected to significantly change expression. ChIP, ATAC, and similar enrichment-based sequencing data may not follow the assumptions inherent in popular methods for normalizing RNA-seq data, as well as exhibiting different types of efficiency and other biases.

This section discusses options for normalization in *DiffBind* using the `dba.normalize` interface function, and shows how they affect the example experiment. It describes the core normalization methods, methods for calculating library sizes, options for using background binning for a more "neutral" normalization, and the potential to apply separate normalization factors for each read measurement (offsets), including loess fit normalization. Section 7.5 compares the impact of seven different normalization options in both *DESeq2* and *edgeR* , showing how normalizing against the background vs. enriched consensus reads has a greater impact on analysis results than which specific normalization method is chosen.

The final part of the discussion demonstrates how to exploit reads associated with spike-ins and parallel factors, as an alternative to background reads, to normalize these datasets.

7.1 Core normalization methods

DiffBind relies on three underlying core methods for normalization. These include the "native" normalization methods supplied with *DESeq2* and *edgeR* , as well as a simple library-based method. The normalization method is specified with the `normalize` parameter to `dba.normalize`

The native *DESeq2* normalization method is based on calculating the geometric mean for each gene across samples[6], and is referred to `"RLE"` or `DBA_NORM_RLE` in *DiffBind* .

The native *edgeR* normalization method is based on the trimmed mean of M-values approach[7], and is referred to as `"TMM"` or `DBA_NORM_TMM` in *DiffBind* .

A third method is also provided that avoids making any assumptions regarding the distribution of reads between binding sites in different samples that may be specific to RNA-seq analysis and inappropriate for ChIP-seq analysis. This method (`"lib"` or `DBA_NORM_LIB`) is based on the different library sizes for each sample, computing normalization factors to weight each sample so as to appear to have the same library size. For *DESeq2* , this is accomplished by dividing the number of reads in each library by the `mean` library size[6]. For *edgeR* , the normalization factors are all set to `1.0`, indicating that only library-size normalization should occur.

Note that any of these normalization methods can be used with either the *DESeq2* or *edgeR* analysis methods, as *DiffBind* converts normalization factors to work appropriately in either *DESeq2* or *edgeR* .

Consider some potential normalization options for the example dataset. We can start with non-normalized data:

[6] The `mean` function can be changed by the user by setting the `libFun` parameter

```
> data(tamoxifen_analysis)
> dba.plotMA(tamoxifen, contrast=list(Resistant=tamoxifen$masks$Resistant),
+            bNormalized=FALSE, sub="Non-Normalized")
```

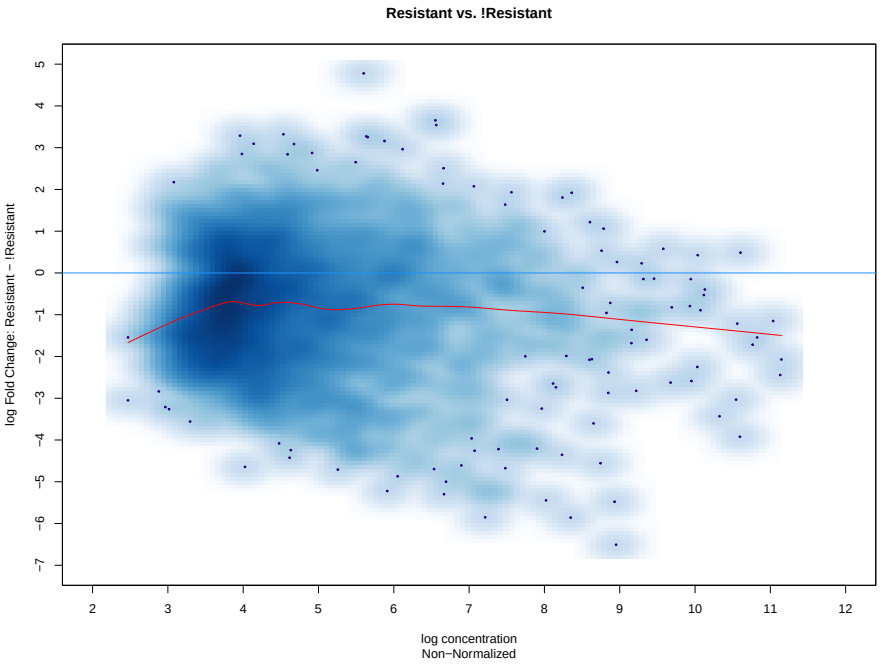


Figure 17: MA plot showing non-normalized data Generated by plotting the result of: `dba.plotMA(tamoxifen,contrast=list(Resistant=tamoxifen$masks$Resistant),bNormalized=FALSE,sub="Non-Normalized")`

Figure 17 shows the non-Resistant (Responsive) samples with a greater raw read density, with the darkest part of the blue cloud being located beneath the horizontal blue line centered zero fold change, as well as the entirety of the red loess fit curve likewise residing below that line.

A key question is if this is an artifact that needs to be corrected via normalization, or a biological signal that needs to be retained.

Compare the non-normalized data to an analysis based on a normalization that only takes the library size (total number of aligned reads, or sequencing depth) into account:

```
> tamoxifen <- dba.normalize(tamoxifen, normalize=DBA_NORM_LIB)
> tamoxifen <- dba.analyze(tamoxifen)
> dba.plotMA(tamoxifen, method=DBA_DESEQ2, sub="DESeq2:lib:full")
```

Figure 18 shows that normalizing for sequencing depth alters the bias in signal enrichment towards the Responsive samples only slightly, with the highest density closer to, but still below, the center line, and the bulk of the loess fit line in the lower half as well.

The following code gathers the differentially bound sites in a variable `dbs`, in which we will accumulate results in order to compare the results of different types of normalization:

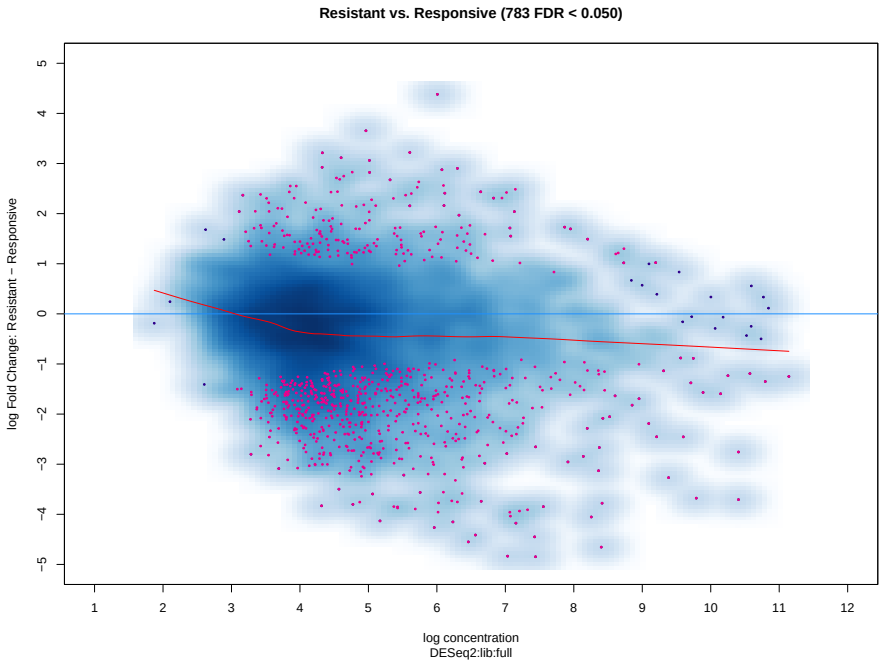


Figure 18: MA plot showing results of analysis using Library-size based normalization.

```
> dbs <- dba.report(tamoxifen, bDB=TRUE, bGain=TRUE, bLoss=TRUE)
> dbs$config$factor <- "normalize"
> dbs$class[DBA_ID,] <- colnames(dbs$class)[1] <-  "LIB_Full"
> dbs$class[DBA_FACTOR,] <- DBA_NORM_LIB
> dbs

3 Samples, 783 sites in matrix:
  Contrast Direction normalize Method Intervals
1 LIB_Full       All       lib DESeq2       783
2 LIB_Full      Gain       lib DESeq2       188
3 LIB_Full      Loss       lib DESeq2       595
```

The analysis results reflects the bias towards enrichment in the Responsive samples. Of 783 total sites identified as differentially bound, 595 (76%) exhibit greater binding affinity in the Responsive condition, while only 188 are enriched in the Resistant condition.

Compare this to the `RLE` normalization procedure native to *DESeq2* :

```
> tamoxifen <- dba.normalize(tamoxifen, normalize=DBA_NORM_NATIVE)
> tamoxifen <- dba.analyze(tamoxifen)
> dba.plotMA(tamoxifen, method=DBA_DESEQ2, sub="DESeq2:RLE:RiP")
```

Figure 19 shows quite a different picture. More sites are close to the zero-fold center line, and the loess fit sits largely on top, if not slightly above, that line. The results of the analysis have changed substantially as well:

```
> db <- dba.report(tamoxifen, bDB=TRUE, bGain=TRUE, bLoss=TRUE)
> db$class[DBA_ID,] <- "RLE_RiP"
```

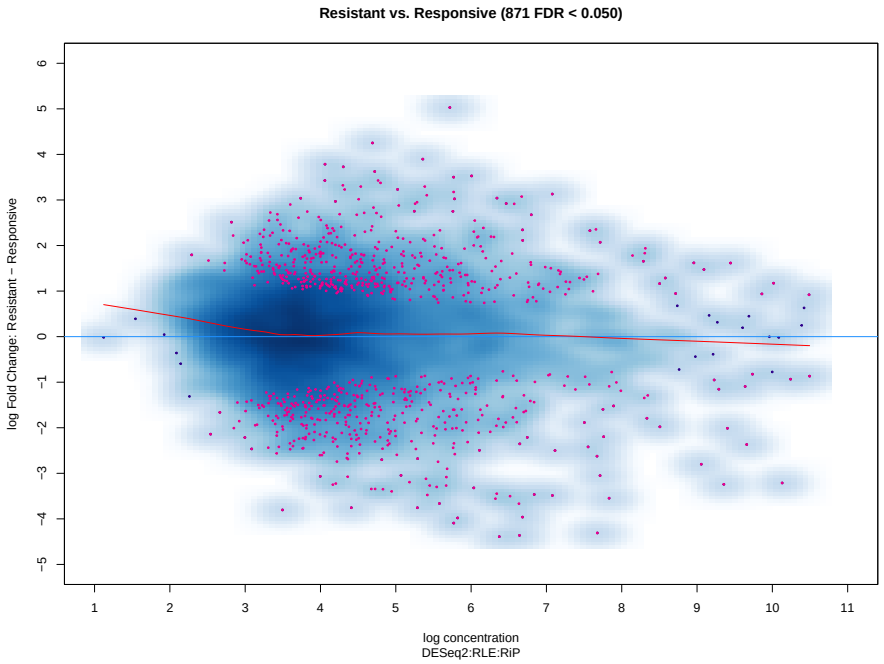


Figure 19: MA plot showing results of analysis using RLE based normalization.

```
> db$class[DBA_FACTOR,] <- DBA_NORM_RLE
> dbs <- dba.peakset(dbs, db)
> db

3 Samples, 871 sites in matrix:
  Contrast Direction  DB Method Intervals
1  RLE_RiP       All RLE DESeq2       871
2  RLE_RiP      Gain RLE DESeq2       469
3  RLE_RiP      Loss RLE DESeq2       402
```

The 871 sites as identified as differentially bound in this analysis are much more evenly divided, with a few *more* sites enriched in the Resistant condition (469) than in the Responsive condition 402. We can compare overlaps between the analysis based on library size normalization vs RLE normalization:

```
> par(mfrow=c(3,1))
> dba.plotVenn(dbs,c(1,4), main="Total DB Sites")
> dba.plotVenn(dbs,dbs$masks$Gain,main="Gain in Resistant")
> dba.plotVenn(dbs,dbs$masks$Loss,main="Gain in Responsive")
> par(mfrow=c(1,1))
```

The `RLE` normalization, developed for normalizing RNA-seq count matrices, has resulted in normalizing the data such that the binding changes are more evenly distributed between the two conditions. In Figure 20, the top plot shows there are 590 sites identified in both analyses, with another 474 unique to one analysis or the other. The remaining diagrams show how the

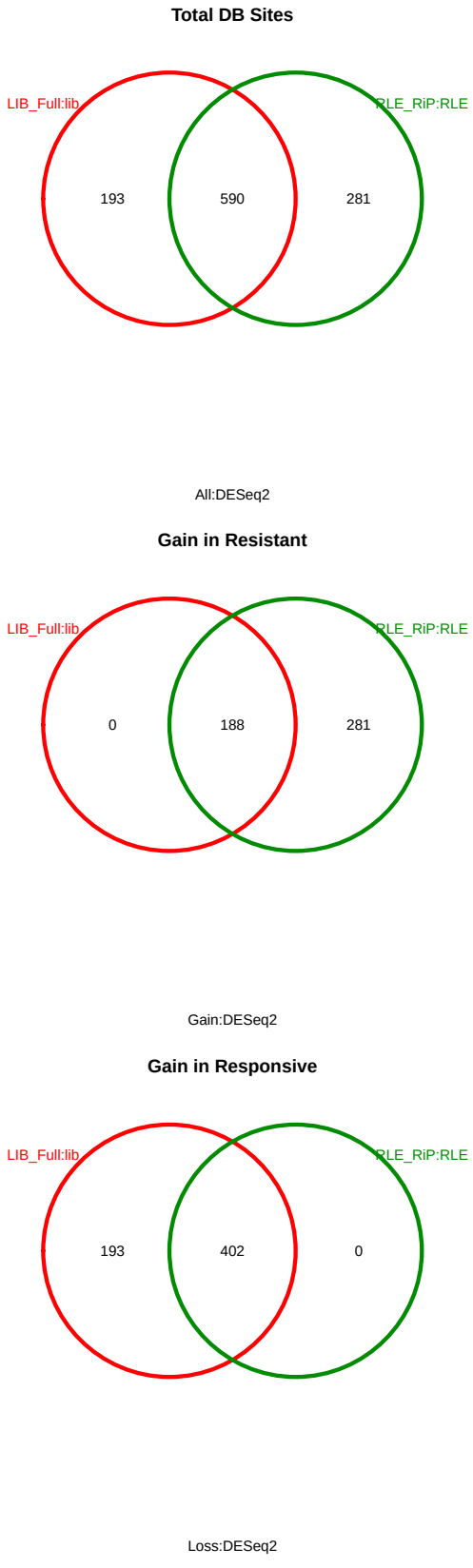


Figure 20: Venn diagrams showing overlaps of sites identified as differentially bound when using library size vs RLE normalization.

`RLE`-based analysis identifies many additional sites that gain binding affinity in the Resistant condition, while "missing" sites identified in the library-sized based analysis as being enriched in the Responsive condition.

These two analyses could lead to quite different biological conclusions regarding the dynamics of ER binding in response to tamoxifen. Which one should be favored? Given the nature of the experimental design for the example data, with both biological replication in the form of multiple cell lines as well as multiple experimental/technical replicates, and without having a prior reason to believe that changes in binding affinity should be balanced, it would be difficult to justify preferring the `RLE`-based analysis, as it alters the data distribution to a greater extent. It is largely for this reason that a library-size normalization is set as the default method within *DiffBind* .

7.2 Library size calculations

An important parameter involved in normalizing is the calculation of library sizes. In the previous sub-section, for the library-size based normalization, the library size for each sample was set to the total number of aligned reads in the bam file for that sample, representing the sequencing depth. This is the Full library size, represented in *DiffBind* as `DBA_LIBSIZE_FULL` or `"full"`.

Another popular way of calculating the library size is to sum the reads that overlap consensus peaks in each sample (Reads in Peaks). Within *DiffBind* , this is known as `DBA_LIBSIZE_PEAKREADS` or `"RiP"`. In this case, the matrix of read counts overlapping consensus sites functions similarly the count matrix in a standard RNA-seq analysis. Library sizes calculated in this manner take into account aspects of both the sequencing depth and the "efficiency" of the ChIP. An inefficient ChIP, where a high proportion of the reads are not in enriched peaks (high proportion of "background" reads), may not have a strong signal even if sequenced to a greater depth. When using the native normalization methods, developed originally for RNA-seq, the `"RiP"` library sizes are used.

We can compare the results of a library-size based normalization using the full library sizes, as in Figure 18, with one using the reads in peaks:

```
> tamoxifen <- dba.normalize(tamoxifen, normalize=DBA_NORM_LIB,
+                            library=DBA_LIBSIZE_PEAKREADS, background=FALSE)
> tamoxifen <- dba.analyze(tamoxifen)
> dba.plotMA(tamoxifen, method=DBA_DESEQ2, sub="DESeq2:lib:RiP")
```

Figure 21 shows how the library-size normalization based on peak reads differs from one based on sequencing depth alone. The normalization is more "even", in that most sites have similar read densities (and are closer to the blue, center zero-fold line), and the loess curve is no longer below the zero-fold line. The analysis itself show more balance between differentially bound sites that gain affinity in the two conditions:

```
> dbs$class[DBA_CONDITION,1:3] <- DBA_LIBSIZE_FULL
> dbs$class[DBA_CONDITION,4:6] <- DBA_LIBSIZE_PEAKREADS
> dbs$config$condition <- "lib.size"
> db <- dba.report(tamoxifen, bDB=TRUE, bGain=TRUE, bLoss=TRUE)
> db$class[DBA_ID,] <- "LIB_RiP"
> db$class[DBA_FACTOR,] <- DBA_NORM_LIB
> db$class[DBA_CONDITION,] <- DBA_LIBSIZE_PEAKREADS
> dbs <- dba.peakset(dbs, db)
```

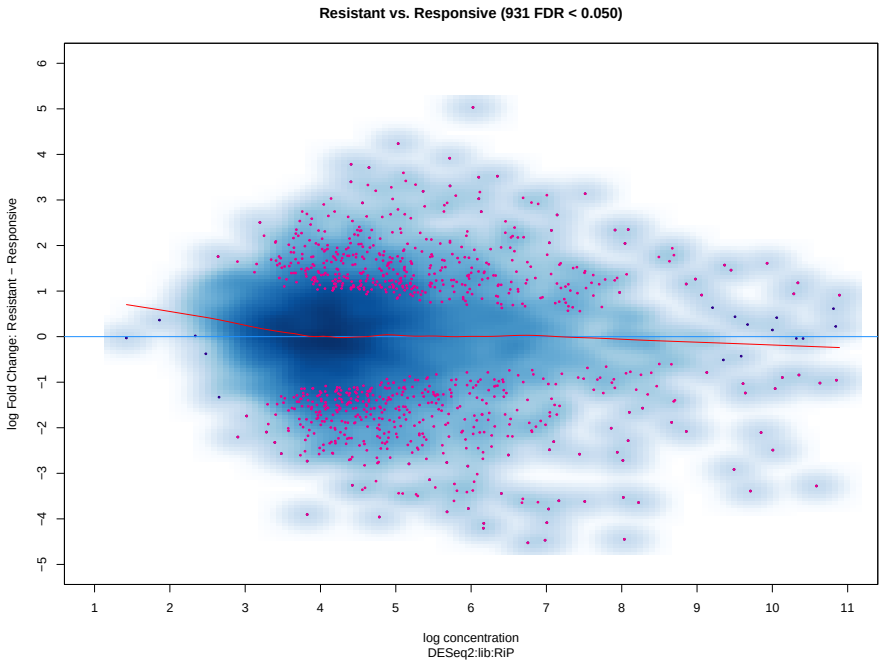


Figure 21: MA Plot showing results of analysis using "RiP" for library-size normalization

```
> db

3 Samples, 931 sites in matrix:
  Contrast Direction  DB Method Intervals
1  LIB_RiP       All lib    RiP       931
2  LIB_RiP      Gain lib    RiP       466
3  LIB_RiP      Loss lib    RiP       465
```

The numbers and ratios of the gain and loss sites look a lot more like the previous `RLE`-based analysis than the full library-size based analysis, as can be confirmed with a Venn diagram:

```
> dba.plotVenn(dbs,c(1,7,4),main="DB Sites")
```

Figure 22 shows the overlaps of the differentially bound sites from the two library-size based analyses and the prior `RLE` analysis. When using reads in peaks, the library-size analysis identifies many of the same sites enriched in the Resistant group that the `RLE` analysis detected, as well as 64 sites that are more enriched in the Responsive condition that the prior library-size based analysis identified but that the `RLE` on did not.

Since calculating library sizes using Reads in Peaks yields results that are similar to the `RLE` method, identifying more and more balanced differentially bound peaks, it may be that taking ChIP efficiency into consideration, not just sequencing depth, is advantageous. However, there is no unbiased way to determine if the underlying cause of differences in reads overlapping consensus peaks is due to a technical issue in ChIP, or to a true biological signal whereby there "really" are different degrees of overall binding.

In the current example, the fraction of reads in peaks (`FRiP`) is fairly consistent between replicates for each cell type, indicating that the differences are *not* due to ChIP efficiency:

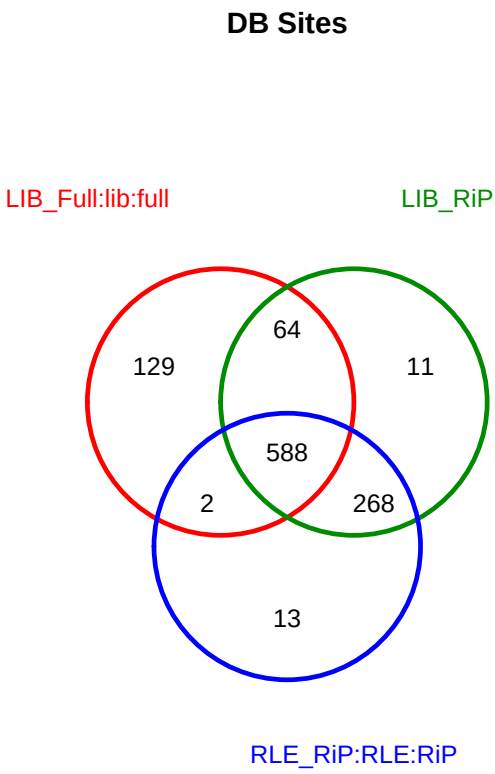


Figure 22: Venn diagram showing overlapping differentially bound sites identified in analyses using library-size normalization with full and reads-in-peaks library sizes, as well as `RLE` normalization using reads-in-peaks.

```
> dba.show(tamoxifen,attributes=c(DBA_ID,DBA_FRIP))

       ID FRiP
1  BT4741 0.16
2  BT4742 0.15
3   MCF71 0.31
4   MCF72 0.19
5   MCF73 0.25
6   T47D1 0.11
7   T47D2 0.06
8  MCF7r1 0.22
9  MCF7r2 0.14
10  ZR751 0.33
```

```
11  ZR752 0.22
```

It is largely for this reason that the default normalization in *DiffBind* uses library size normalization based on full library sizes.

7.3 Background normalization

In the previous section, we saw how a library-size normalization based on the full library size, rather than on the reads in the enriched areas designated by the consensus peaks set, enabled an analysis that avoided making assumption about the changes in binding patterns (specifically, assumptions that binding changes are roughly balanced). Another way to do this, while gaining some of the benefits of the native normalization methods, is to use *background normalization*. Given the difficulty in differentiating between technical biases and biological signal when attempting to normalize ChIP-seq data, the idea is to normalize based on a more neutral sample of reads.

The core background normalization technique is to divide the genome into large bins and count overlapping reads[7]. As the enrichment expected in ChIP-seq (and ATAC-seq) is expected to occur over relatively narrow intervals (roughly between 100bp and 600bp), it is expected that there should not be systematic differences in signals over much larger intervals (on the order of 10,000bp and greater). Any differences seen should be technical rather than biological, so it is safer to normalize based these differences.

Note also that this type of background normalization is one of the methods recommended for ATAC-seq analysis [9].

By specifying `background=TRUE` in `dba.normalize`, all chromosomes that contains peaks in the consensus set are divided into non-overlapping bins (default size 15,000bp) and overlapping reads counted. These can then be normalized using any of the normalization methods. While generally doing a library size based normalization will yield the same result as on the full library size[8], the native normalization methods can be applied to the background bins.

For the example, we can compare to a *DESeq2* analysis with `RLE` normalization, as well as a *edgeR* analyses (and its native `TMM` normalization). Note that computing background reads requires access to the full sequencing data (bam files); however the loaded example objects `tamoxifen_counts` and `tamoxifen_analysis` include the results of calculating background reads and can be used as-is.

```
> data(tamoxifen_analysis)
> tamoxifen <- dba.normalize(tamoxifen, method=DBA_ALL_METHODS,
+                            normalize=DBA_NORM_NATIVE,
+                            background=TRUE)
> tamoxifen <- dba.analyze(tamoxifen, method=DBA_ALL_METHODS)
> dba.show(tamoxifen,bContrasts=TRUE)

     Factor     Group Samples     Group2 Samples2 DB.edgeR DB.DESeq2
1 Condition Resistant       4 Responsive        7      732       652
```

[7] This method is used in the *THOR* differential analysis tool`allhoff2016thor`, and is discussed in the User Guide for the *csaw* package[8]. Internally, *DiffBind* uses the *csaw* methods to compute background reads

[8] The only difference is that the "full" library size includes the total number of reads in the supplied bam files, while the background normalization, by default, counts the reads on chromosomes which contain peaks in the peakset.

```
> par(mfrow=c(2,1))
> dba.plotMA(tamoxifen, method=DBA_EDGER, sub="edgeR:TMM:background")
> dba.plotMA(tamoxifen, method=DBA_DESEQ2, sub="DESeq2:RLE:background")
> par(mfrow=c(1,1))
```

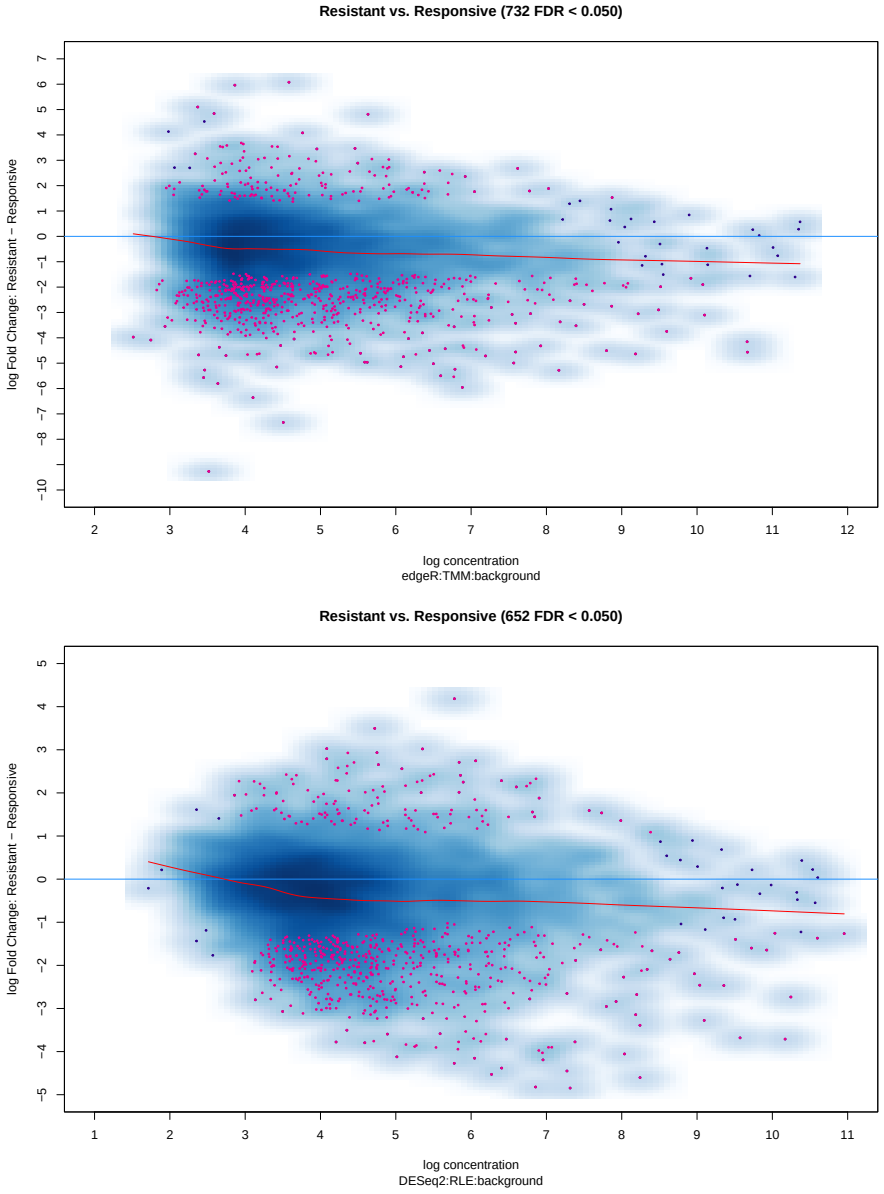


Figure 23: MA Plots showing results of analysis using background reads and (top) `TMM` normalization with *edgeR* and (bottom) `RLE` normalization with *DESeq2*

Figure 23 shows the results of these analyses. While the loess fit line is smoother and closer to the zero fold change line, the sites identified as being differentially bound remain biased towards those with a loss of binding affinity in the Resistant condition.

```
> db <- dba.report(tamoxifen, bDB=TRUE, bGain=TRUE, bLoss=TRUE)
> db$class[DBA_ID,] <- "RLE_BG"
> db$class[DBA_FACTOR,] <- DBA_NORM_RLE
> db$class[DBA_CONDITION,] <- DBA_LIBSIZE_BACKGROUND
> dbs <- dba.peakset(dbs, db)
> db

3 Samples, 652 sites in matrix:
  Contrast Direction  DB     Method Intervals
1   RLE_BG       All RLE background       652
2   RLE_BG      Gain RLE background       128
3   RLE_BG      Loss RLE background       524
```

Of the 652 differentially bound sites, only 128 gain affinity in the Resistant condition, while 524 show a loss. This analysis is quite similar to the one carried out using full library size based normalization:

```
> par(mfcol=c(3,2))
> dba.plotVenn(dbs,c(1,10),   main="All Differentially Bound Sites")
> dba.plotVenn(dbs,c(2,11),   main="Gain in Resistant cells")
> dba.plotVenn(dbs,c(3,12),   main="Loss in Resistant cells")
> dba.plotVenn(dbs,c(1,10,4), main="All Differentially Bound Sies")
> dba.plotVenn(dbs,c(2,11,5), main="Gain in Resistant cells")
> dba.plotVenn(dbs,c(3,12,6), main="Loss in Resistant cells")
```

Figure 24 illustrates the differences. The left column compares the impact of full library size normalization to that of background normalization, showing how the background normalization is even more conservative, identifying fewer sites that both gain and lose binding affinity. The right column includes the results of an analysis based on the native `RLE` normalization method for *DESeq2* , identifying many additional sites that gain binding affinity in the Resistant cells, while "missing" many sites that lose binding affinity in the Resistant cells.

7.4 Offsets and loess normalization

While the normalization discussed so far have relied on computing normalization coefficients for each sample, an alternative is to compute normalization factors for each read count in the consensus count matrix (that is, for each consensus peak for each sample). These are called *normalization factors* in *DESeq2* and *offsets* in *edgeR* .

A matrix of offsets can be supplied via the `dba.normalize` function, or an offset matrix can be calculated using a `loess` fit. As described in the user guide for the *csaw* package[8], this method can help correct certain kinds of biases in the data, particularly trended biases where there is a systematic difference in fold changes as mean concentration levels change. This normalization method was also identified in [9] as being advantageous for ATAC-seq data that show a trended bias.

In the tamoxifen example, we don't see this kind of bias. However we can approximate it by reducing the data to two sample groups:

```
> mcf7t47d <- dba(tamoxifen,mask=c(3:7))
> dba.plotMA(mcf7t47d,
+            contrast=list(MCF7=mcf7t47d$masks$MCF7,
```

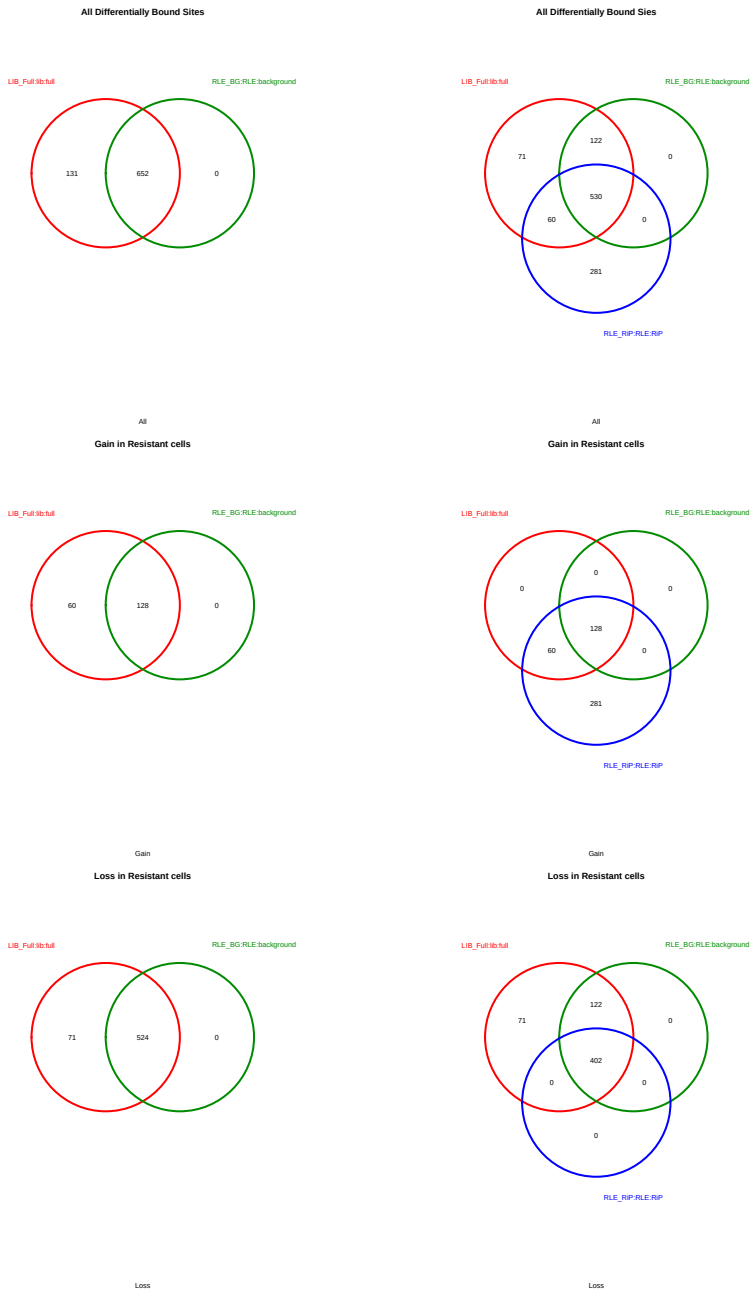


Figure 24: Venn diagrams of overlapping differentially bound sites identified using full library size based normalization compared to background RLE and Reads in Peaks RLE normalization

```
+                         T47D=mcf7t47d$masks$T47D),
+             bNormalized=FALSE)
```

Figure 25 shows the apparent trended bias between the Responsive MCF7 and T47D samples.

Next we will use a loess fit to generate offsets, followed by an *edgeR* analysis (to better approximate the *csaw* example):

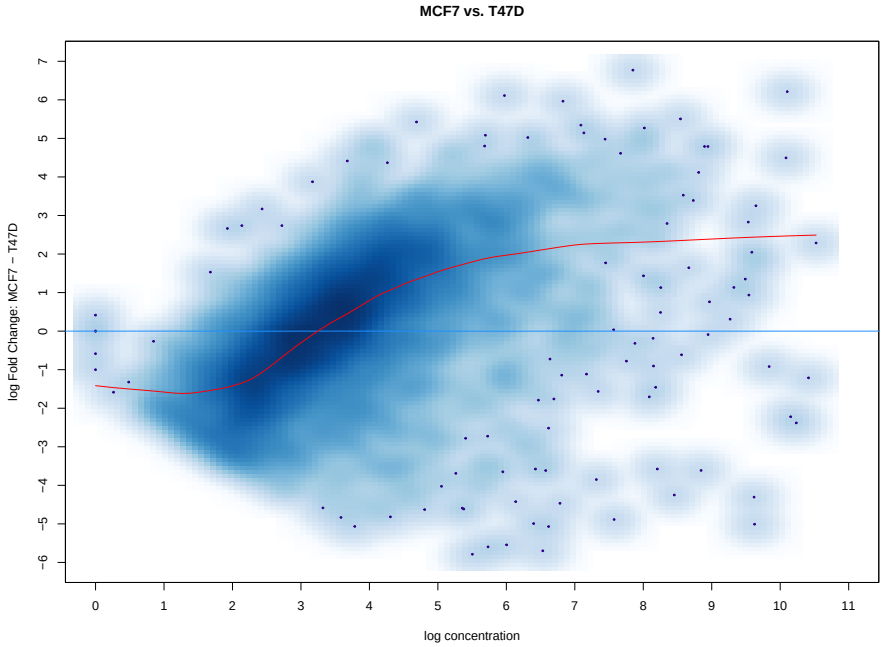


Figure 25: MA Plot showing evidence of a "trended bias" in reduced dataset

```
> mcf7t47d$config$AnalysisMethod <- DBA_EDGER
> mcf7t47d <- dba.normalize(mcf7t47d, offsets=TRUE)
> mcf7t47d <- dba.contrast(mcf7t47d, contrast=c("Tissue","MCF7","T47D"))
> mcf7t47d <- dba.analyze(mcf7t47d)
> dba.plotMA(mcf7t47d)
```

Figure 26 shows how the bias is "corrected" by this normalization. The result is a very close balance between sites identified and significantly gaining binding affinity in the MCF7 samples versus those gaining binding affinity in the T47D samples:

```
> mcf7t47d.DB <- dba.report(mcf7t47d)
> sum(mcf7t47d.DB$Fold > 0)

[1] 423

> sum(mcf7t47d.DB$Fold < 0)

[1] 513
```

If we are certain that the bias is technical, this could save the dataset. However, if we are not certain that the observed trend does not reflect a genuine biological signal, such as a case where the MCF7 cells have a set of sites with much higher binding affinity overall, this normalization could skew the data so as to remove identification of these sites.

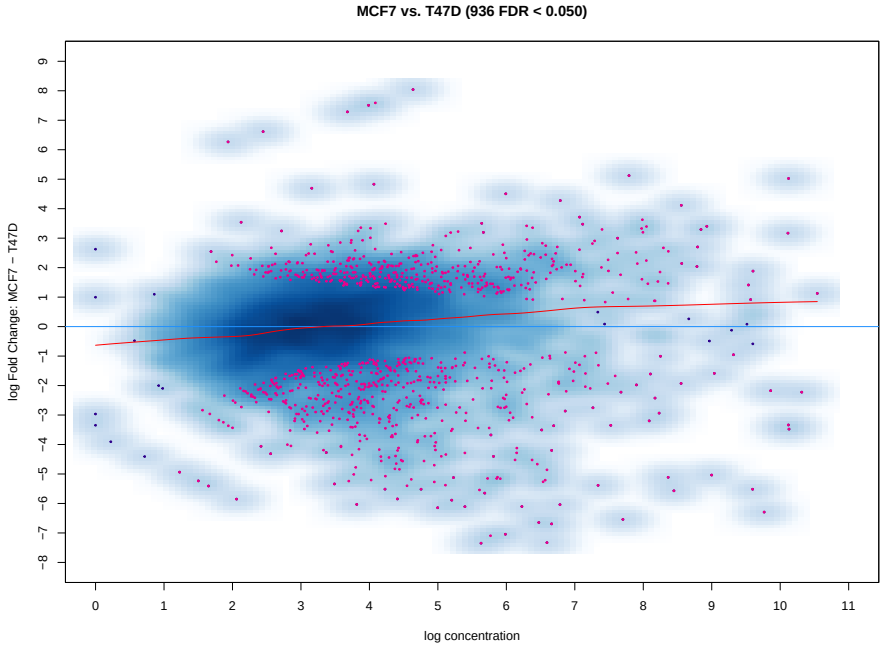


Figure 26: MA Plot showing results of normalizing with offsets generated using a loess fit.

7.5 Comparing the impact of normalization methods on analysis results

As can be seen, there are many possible ways of analyzing the same data when taking into account analysis methods (*DESeq2* and *edgeR*), library size calculations (relying on all sequencing reads or only those that overlap consensus peaks), normalization methods (`RLE`, `TMM`, `loess offsets`, or solely by library size), and finally whether to normalize based on enriched versus background regions.

There are seven primary ways to normalize the example dataset:

1. Library size normalization using full sequencing depth
2. Library size normalization using Reads in Peaks
3. `RLE` on Reads in Peaks
4. `TMM` on Reads in Peaks
5. `loess` fit on Reads in Peaks
6. `RLE` on Background bins
7. `TMM` on Background bins

We can cycle through the seven possibilities, repeated for *DESeq2* and for *edgeR* , and gather the results in a report-based `DBA` object called `dbs.all` for plotting and further analysis:

```
14 Samples, 1194 sites in matrix:
          Contrast Normalization.Method Reference.Reads Analysis.Method
1    lib_full_edgeR                 lib              BG           edgeR
```

```
2   lib_full_DESeq2                lib              BG          DESeq2
3     lib_RiP_edgeR                lib             RiP           edgeR
4    lib_RiP_DESeq2                lib             RiP          DESeq2
5     RLE_RiP_edgeR                RLE             RiP           edgeR
6    RLE_RiP_DESeq2                RLE             RiP          DESeq2
7      RLE_BG_edgeR                RLE              BG           edgeR
8     RLE_BG_DESeq2                RLE              BG          DESeq2
9     TMM_RiP_edgeR                TMM             RiP           edgeR
10   TMM_RiP_DESeq2                TMM             RiP          DESeq2
11    TMM_BG_edgeR                 TMM              BG           edgeR
12    TMM_BG_DESeq2                TMM              BG          DESeq2
13  loess_RiP_edgeR               loess             RiP           edgeR
14 loess_RiP_DESeq2               loess             RiP          DESeq2
   Intervals
1        819
2        783
3        998
4        931
5        991
6        871
7        727
8        652
9        979
10       911
11       732
12       689
13      1056
14       923
```

First considering the *DESeq2* results, we can plot a heatmap of the identified differentially bound peaks to see how the methods cluster:

```
> deseq <- dba(dbs.all,mask=dbs.all$masks$DESeq2, minOverlap = 1)
> binding <- dba.peakset(deseq, bRetrieve=TRUE)
> dba.plotHeatmap(deseq, maxSites=nrow(binding),  bLog=FALSE,
+                 correlations=FALSE,minval=-5, maxval=5, cexCol=1.3,
+                 colScheme = hmap, main = "DESeq2 Differentially Bound Sites",
+                 ColAttributes = c(DBA_CONDITION, DBA_FACTOR),
+                 key.title = "LFC")
```

In Figure 27, sites that gain affinity in the tamoxifen Resistant condition are shown in green (positive fold changes), those that gain affinity in the Responsive condition are shown in red (negative fold changes), and those that are not identified by a specific analysis are shown in black (zero fold change).

The analyses break into two main clusters, one containing the four analyses that relied on the main count matrix ("RiP") for normalizing, and the other encompassing the three analyses that did not (relying on either background read counts, or on the total number of reads in the sequencing libraries.) The RiP cluster shows a greater balance between sites that gain binding strength in each of the two conditions, while sites identified by analyses using background reads identify mostly sites with greater binding affinity in the Responsive sample group.

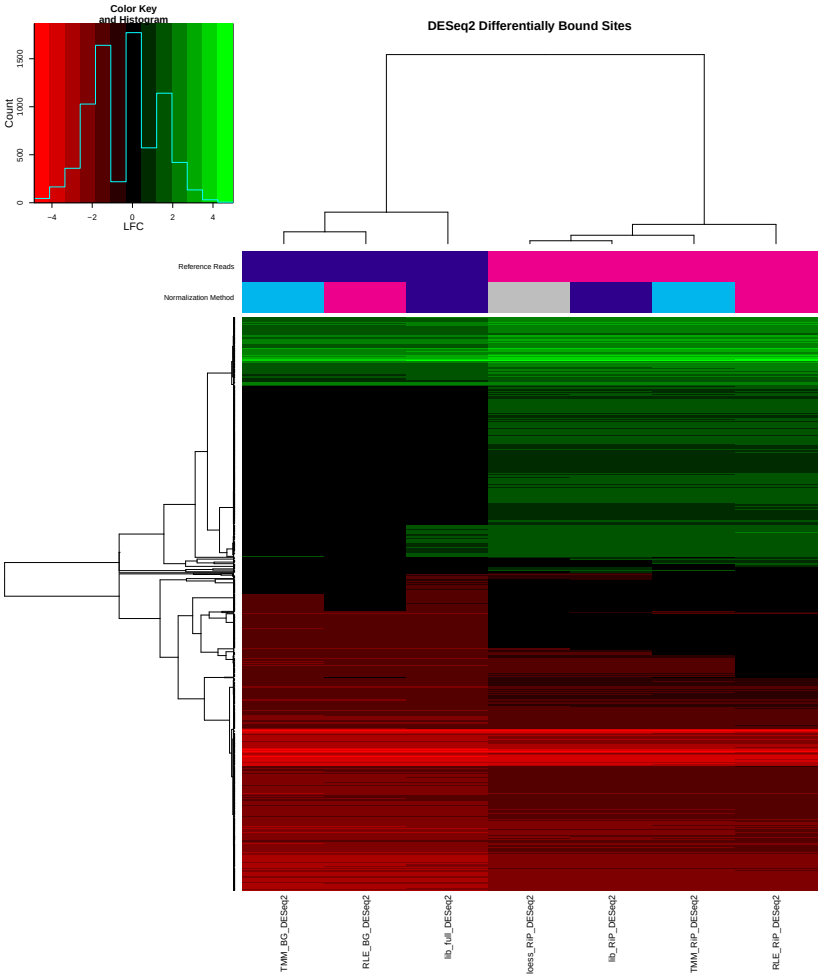


Figure 27: Clustering heatmap of differentially bound sites from *DESeq2* analyses using various normalization and reference read methods

Given that the choice of which sets of reads to use for normalizing (focusing on reads in peaks or on all the reads in the libraries) is more important in determining analysis results than the choice of specific normalization method, we can now look at the relative importance of the choice of analysis methods (*DESeq2* and *edgeR*):

```
> binding <- dba.peakset(dbs.all, bRetrieve=TRUE)
> dba.plotHeatmap(dbs.all, maxSites=nrow(binding), bLog=FALSE,
+                 correlations=FALSE, minval=-5, maxval=5, cexCol=1.3,
+                 colScheme = hmap, key.title="LFC",
+                 ColAttributes = c(DBA_CONDITION, DBA_TREATMENT, DBA_FACTOR),
+                 main="All Differentially Bound Sites")
```

```
> dba.plotHeatmap(dbs.all, cexCol=1.3, main="Correlations of DB Sites",
+                 ColAttributes = c(DBA_CONDITION, DBA_TREATMENT, DBA_FACTOR))
```

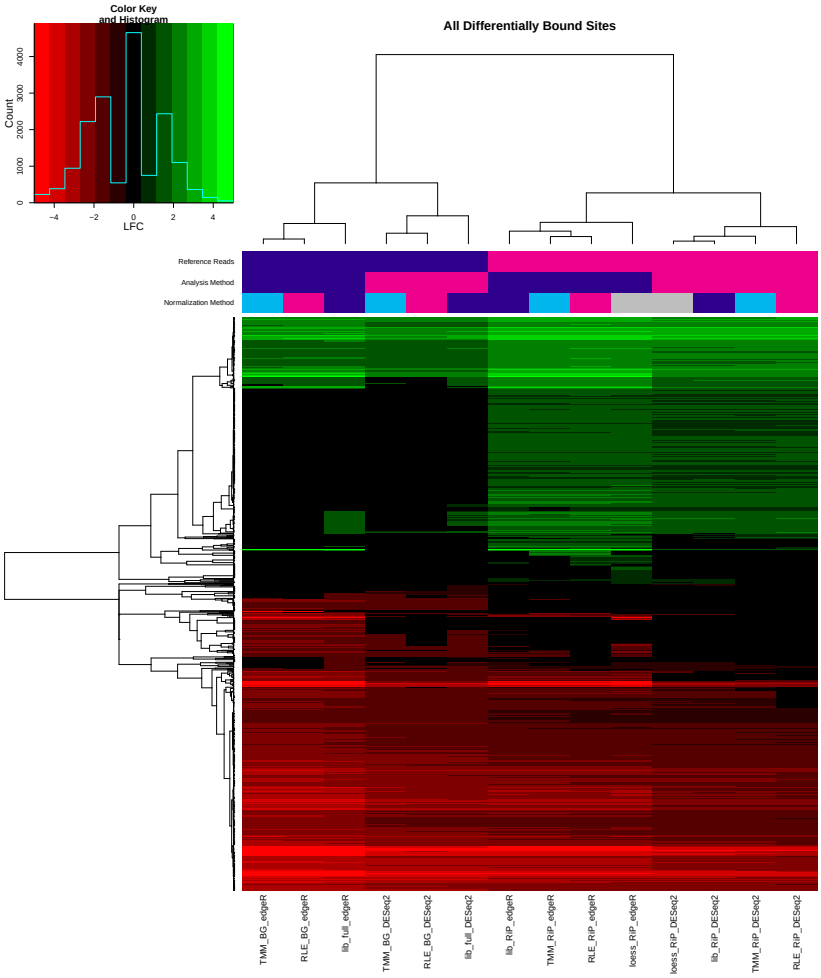


Figure 28: Heatmap of fold changes in differentially bound sites from *DESeq2* and *edgeR* analyses using various normalization and reference read methods

Figure 28 and Figure 29 show that main division into a cluster that normalizes using the consensus count matrix and another that uses background reads (or raw sequencing depth) is maintained. Within each of these two clusters, the analyses cluster by analysis method (*DESeq2* vs *edgeR*).

Again, the specific analysis method used appears to be far less important than which sets of reads are used as the basis for normalization. Indeed, the native RNA-seq methods, `RLE` and `TMM`, give very similar and highly correlated results when using the same count data.

An assumption in RNA-seq analysis, that the read count matrix reflects an unbiased representation of the experimental data, may be violated when using a narrow set of consensus peaks that are chosen specifically based on their rates of enrichment. It is not clear that using normalization methods developed for RNA-seq count matrices on the consensus reads will not alter biological signals; it is probably a good idea to avoid using the consensus count matrix (Reads in Peaks) for normalizing unless there is a good prior reason to expect balanced changes in binding.

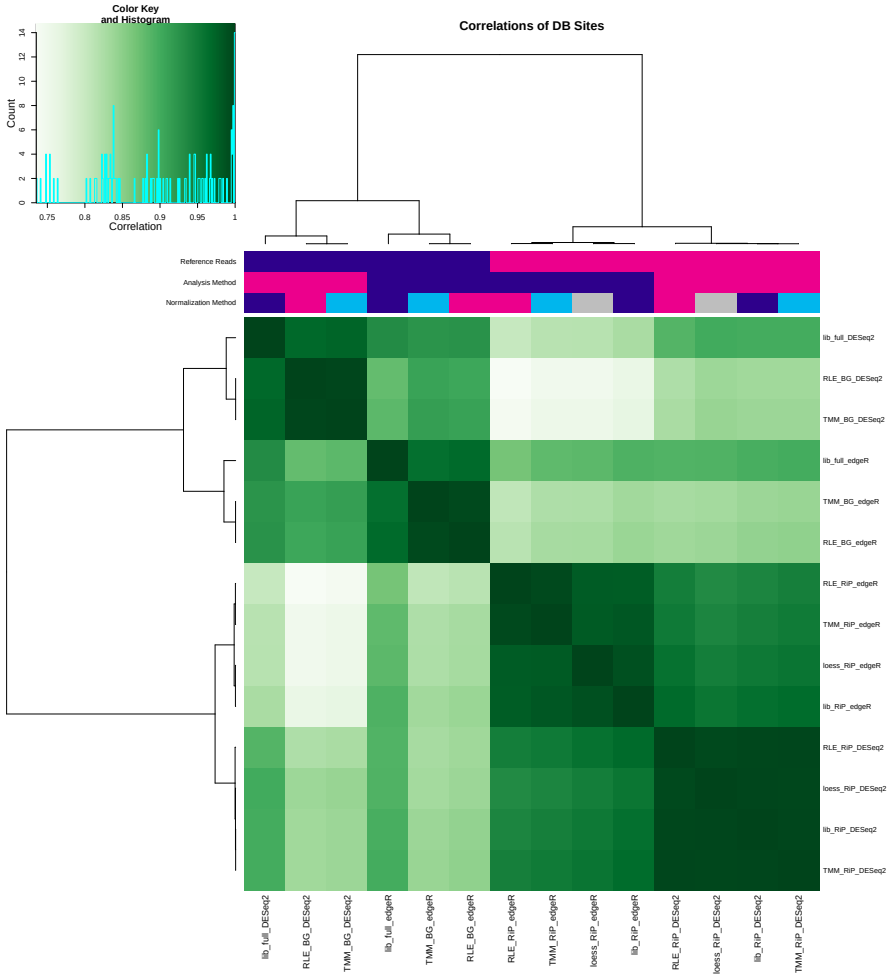


Figure 29: Correlation heatmap of fold changes in differentially bound sites from *DESeq2* and *edgeR* analyses using various normalization and reference read methods

7.6 Spike-in normalization

An alternative for avoiding the use of the consensus count matrix when normalizing ChIP-seq data is to use *spike-in* data, where exogenous chromatin (usually from *Drosophila melanogaster*) is "spiked in" to the ChIP. If the amount of spiked-in chromatin can be precisely controlled, then we can use the relative amounts of reads that map to the alternative reference genome for each sample. *DiffBind* allows for spike-in reads to be included in the experiment, either as an additional set of sequencing reads (bam) files, or included in the primary reads (assuming a hybrid reference genome, where the exogenous reads align to different chromosome names than do the consensus peaks). Counting these reads then forms a background we can use in the same manner as background normalization discussed previously (either a library size adjustment using the total number of exogenous reads for each sample, or a native RNA-seq method (`TMM` or `RLE`) taking into account how those reads are distributed).

To illustrate this, a example dataset has been included in the *DiffBind* package. This dataset is derived from that used in [10][9]. In this dataset, which also looks at ER binding, the signal is highly unbalanced between conditions. In this case we know that this is the result of a genuine biological phenomenon, as the condition treated with Fulvestrant is known to block ER binding. We are looking to normalize the data while preserving the bias toward binding in the non-Fulvestrant condition.

We can load the data as follows, observing the two conditions, each with four replicates:

```
> load(system.file('extra/spikes.rda',package='DiffBind'))
> spikes

8 Samples, 53 sites in matrix:
  ID Tissue Factor   Condition Replicate Reads FRiP
1 1a     S2     ER Fulvestrant         1 30294 0.02
2 1b     S2     ER        none         1 45985 0.05
3 2a     S2     ER Fulvestrant         2 72925 0.02
4 2b     S2     ER        none         2 76829 0.06
5 3a     S2     ER Fulvestrant         3 62127 0.02
6 3b     S2     ER        none         3 42439 0.06
7 4a     S2     ER Fulvestrant         4 23090 0.02
8 4b     S2     ER        none         4 60695 0.06
```

For this experiment, *Drosophila* reads are included in the samplesheet using the `Spikein` column in the samplesheet:

```
> spikes$samples$Spikein

[1] "bams/drosophila/SLX-8047.D705_D506.C81G5ANXX.s_1.r_1.fq.bam"
[2] "bams/drosophila/SLX-8047.D706_D508.C81G5ANXX.s_1.r_1.fq.bam"
[3] "bams/drosophila/SLX-8047.D704_D507.C81G5ANXX.s_1.r_1.fq.bam"
[4] "bams/drosophila/SLX-8047.D704_D506.C81G5ANXX.s_1.r_1.fq.bam"
[5] "bams/drosophila/SLX-8047.D706_D505.C81G5ANXX.s_1.r_1.fq.bam"
[6] "bams/drosophila/SLX-8047.D705_D508.C81G5ANXX.s_1.r_1.fq.bam"
[7] "bams/drosophila/SLX-8047.D706_D507.C81G5ANXX.s_1.r_1.fq.bam"
[8] "bams/drosophila/SLX-8047.D704_D505.C81G5ANXX.s_1.r_1.fq.bam"
```

The loss of binding affinity in the Fulvestrant condition can be seen in a MA plot:

```
> dba.plotMA(spikes, contrast=list(Fulvestrant=spikes$masks$Fulvestrant),
+            bNormalized=FALSE, sub="RAW", bSmooth=FALSE, dotSize=1.5)
```

Figure 30 shows the non-normalized data.

Next we use the non-spikein normalization methods already discussed: full library sizes, as well as `RLE` normalization using Reads-in-Peaks and background reads.

```
> par(mfrow=c(3,1))
> spikes <- dba.normalize(spikes, normalize=DBA_NORM_LIB,
+                         background=FALSE)
> spikes <- dba.analyze(spikes)
> dba.plotMA(spikes, sub="LIB full", bSmooth=FALSE, dotSize=1.5)
> spikes <- dba.normalize(spikes, normalize="RLE",
```

[9] the data, including the sequencing reads, are available to install from github at `andrewholding/Brundle"` and `andrewholding/BrundleData"`.

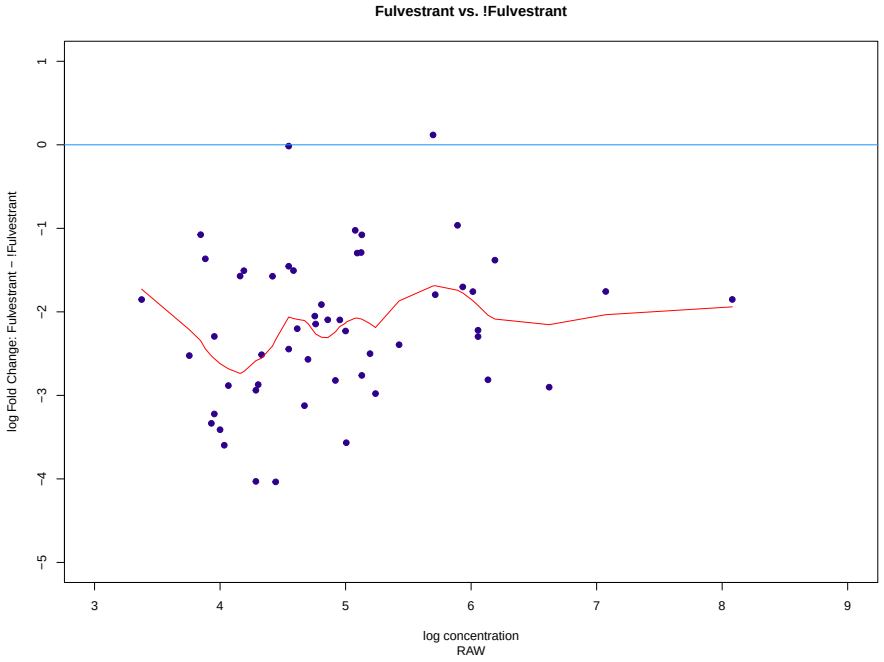


Figure 30: MA plot of ER dataset with and without Fulvestrant treatment, non-normalized

```
+                         background=FALSE)
> spikes <- dba.analyze(spikes)
> dba.plotMA(spikes, sub="RLE RiP", bSmooth=FALSE, dotSize=1.5)
> spikes <- dba.normalize(spikes, normalize="RLE",
+                         background=TRUE)
> spikes <- dba.analyze(spikes)
> dba.plotMA(spikes, sub="RLE BG", bSmooth=FALSE, dotSize=1.5)
> par(mfrow=c(1,1))
```

In Figure 31, the middle plot shows how a direct application of `RLE` using the consensus peaks over-normalizes, causing many sites, after normalization, to have positive fold changes and be identified as significantly gaining in binding affinity after Fulvestrant treatment. The analysis based on a full-depth library size adjustment does better, but it still somewhat shifts the read density upwards towards the Fulvestrant condition, and identifies at least one site as gaining binding affinity. Using a background normalization performs best.

Now we can compare to an alternative background derived from counting the *Drosophila* reads[10]

```
> par(mfrow=c(2,1))
> spikes <- dba.normalize(spikes, normalize=DBA_NORM_LIB,
+                         spikein=spikes.spikeins)
> spikes <- dba.analyze(spikes)
```

[10]Note that pre-calculated background reads are included for the example in an object named `spikes.spikeins`, so we do not need to re-count them for the vignette; we can pass the pre-calculated ones in instead. Normally, with access to the spike-in reads, setting `spikein=TRUE` will result in the spike-in reads being counted.

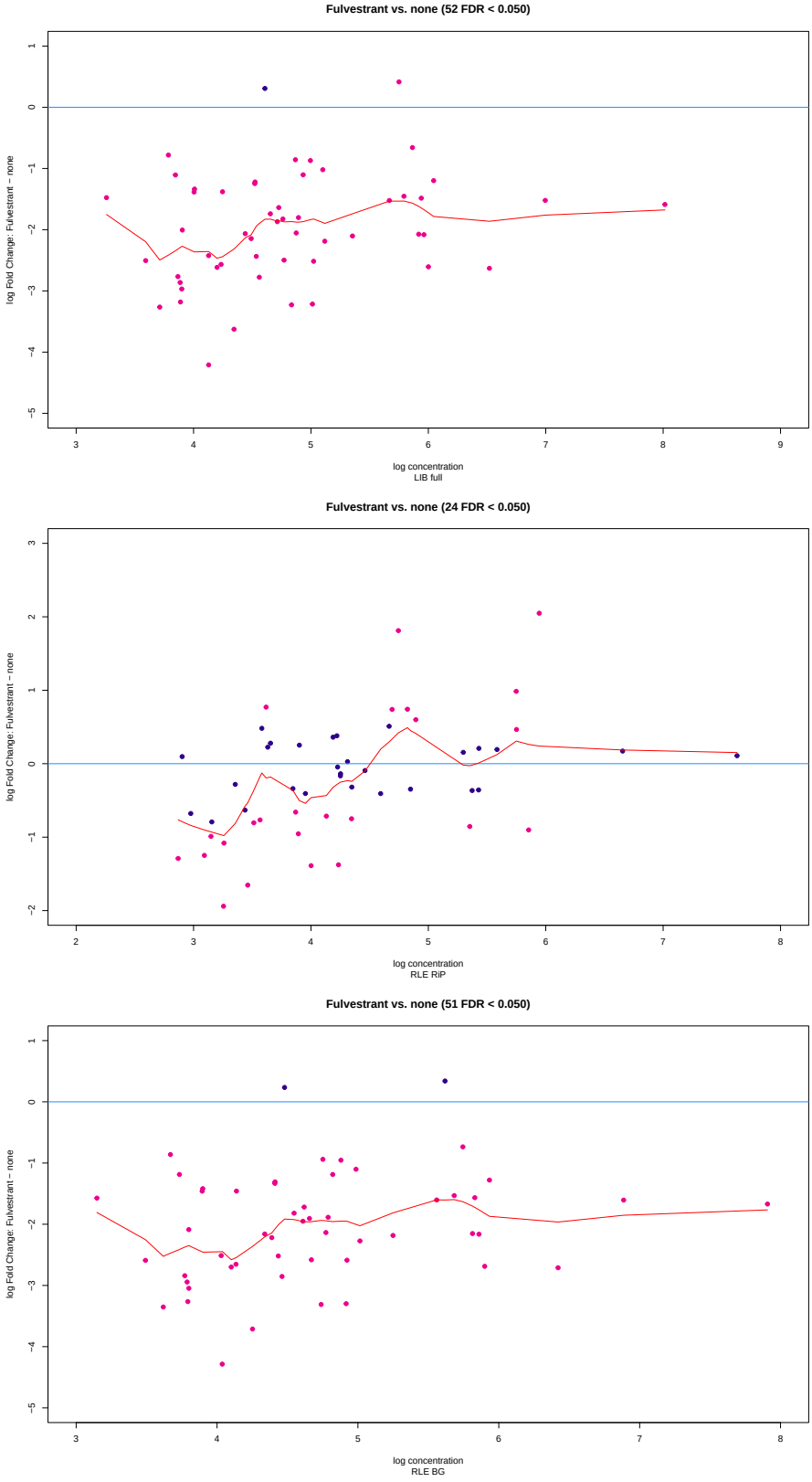


Figure 31: MA plots of ER dataset with and without Fulvestrant treatment, normalized by library size and `RLE` using Reads-in-Peaks and background reads.

```
> dba.plotMA(spikes, sub="LIB spikein", bSmooth=FALSE, dotSize=1.5)
> spikes <- dba.normalize(spikes, normalize=DBA_NORM_RLE, spikein = TRUE)
> spikes <- dba.analyze(spikes)
> dba.plotMA(spikes, sub="RLE spikein", bSmooth=FALSE, dotSize=1.5)
> par(mfrow=c(1,1))
```

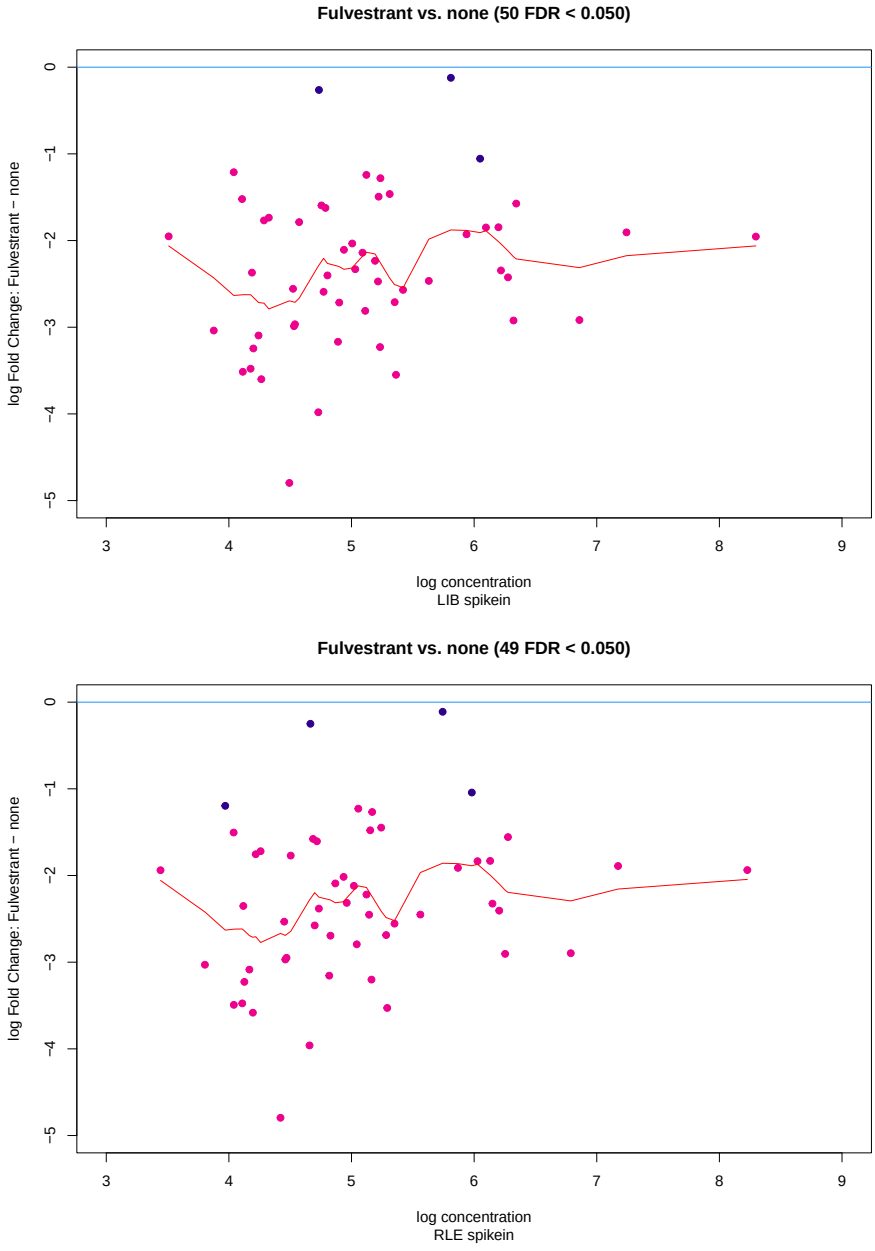


Figure 32: MA plots of ER dataset with and without Fulvestrant treatment, using *Drosophila* spike-in reads as a background, normalized using the number of *Drosophila* reads and `RLE`.

Figure 32 shows how using the spike-in reads enable normalization where all the apparent gains in binding affinity are eliminated, and the bulk of the sites are identified as significantly losing binding affinity in the Fulvestrant condition.

7.7 Parallel factor normalization

Another method related to spike-ins is supported in *DiffBind* , whereby an antibody for a "parallel factor" is also pulled down in the ChIP samples, and consensus peaks from this second factor are used instead of the consensus peaks of the primary factor. This idea behind this *parallel factor normalization* is that while we may not know the true biological properties of the "foreground" factor being studied, we can perform a ChIP on the same sample for an alternative factor that is known not to change its binding patterns under the conditions of our experiment. In [10], the transcription factor **CTCF** is identified as an appropriate such parallel factor[11].

Alternatively, if there are sites for the primary pull-down that are known to not change binding affinity, these can be used without performing a second pull down. This can be seen in the *THOR* tool, where the "housekeeping" normalization is based on focusing on histone marks such as H3K4me3 that should be consistently bound in the promoter regions associated with housekeeping genes[11].

The utility of having a parallel factor can be demonstrated using a sample dataset, again looking at ER binding before and after treatment with Fulvestrant:

```
> load(system.file('extra/parallelFactor.rda',package='DiffBind'))
> parallelFactor

6 Samples, 254 sites in matrix:
  ID Tissue Factor   Condition Replicate  Reads FRiP
1 1a   MCF7 ERCTCF        none         1 869195 0.02
2 1b   MCF7 ERCTCF Fulvestrant         1 281683 0.01
3 2a   MCF7 ERCTCF        none         2 751521 0.02
4 2b   MCF7 ERCTCF Fulvestrant         2 732522 0.01
5 3b   MCF7 ERCTCF Fulvestrant         3 371664 0.01
6 3a   MCF7 ERCTCF        none         3 483964 0.02
```

Again,the loss of binding affinity in the Fulvestrant condition can be seen in the MA plot:

```
> dba.plotMA(parallelFactor,
+            contrast=list(Fulvestrant=parallelFactor$masks$Fulvestrant),
+            bNormalized=FALSE, sub="RAW", bSmooth=FALSE, dotSize=1.5)
```

Figure 33 shows the non-normalized data.

By supplying a set of CTCF consensus peaks (loaded with the example data and called `parallelFactor.peaks`), we can direct *DiffBind* to count reads overlapping these peaks to form the background distributions and proceed to normalization:

```
> par(mfrow=c(2,1))
> parallelFactor <- dba.normalize(parallelFactor, norm=DBA_NORM_LIB,
+                                 spikein = parallelFactor.peaks)
> parallelFactor <- dba.analyze(parallelFactor)
```

[11] Note that the version of parallel factor normalization supported directly in *DiffBind* is not the same as that discussed in [10], where a more sophisticated modeling approach is used.

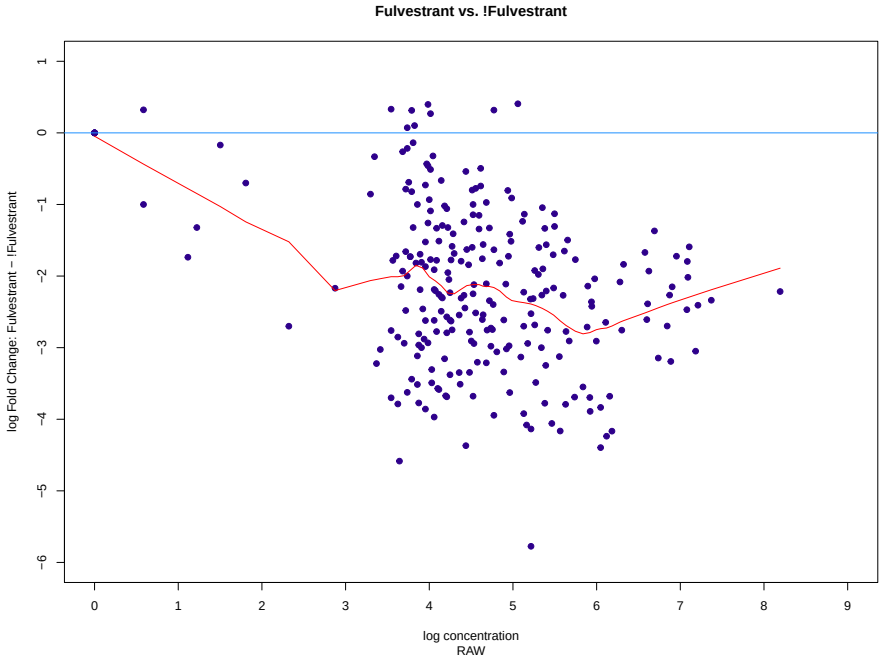


Figure 33: MA plot of second ER dataset with and without Fulvestrant treatment, non-normalized

```
> dba.plotMA(parallelFactor, sub="LIB CTCF", bSmooth=FALSE, dotSize=1.5)
> parallelFactor <- dba.normalize(parallelFactor, norm=DBA_NORM_RLE,
+                                 spikein = TRUE)
> parallelFactor <- dba.analyze(parallelFactor)
> dba.plotMA(parallelFactor, sub="RLE CTCF", bSmooth=FALSE, dotSize=1.5)
> par(mfrow=c(1,1))
```

Figure 34 shows how the parallel factor is able to avoid over-normalizing the data and results in analyses identifying the majority of ER binding sites as significantly losing binding affinity when treated with Fulvestrant.

7.8 Normalization summary

There are a myriad of normalization option available in *DiffBind* . Table 1 summarizes the allowable combinations of normalization methods (columns) and which sets of references reads they can be applied to (rows):

As we have seen, different normalization parameters can alter the experimental data, potentially altering the biological conclusions an analysis might suggest. How then should the normalization parameters be set?

There is no single answer to this, and establishing the correct normalization can be one of the most challenging aspects of a differential binding analysis. Unless we have prior knowledge about the expected signal, or if technical biases are particularly obvious, it may be wise to avoid over-normalizing the data.

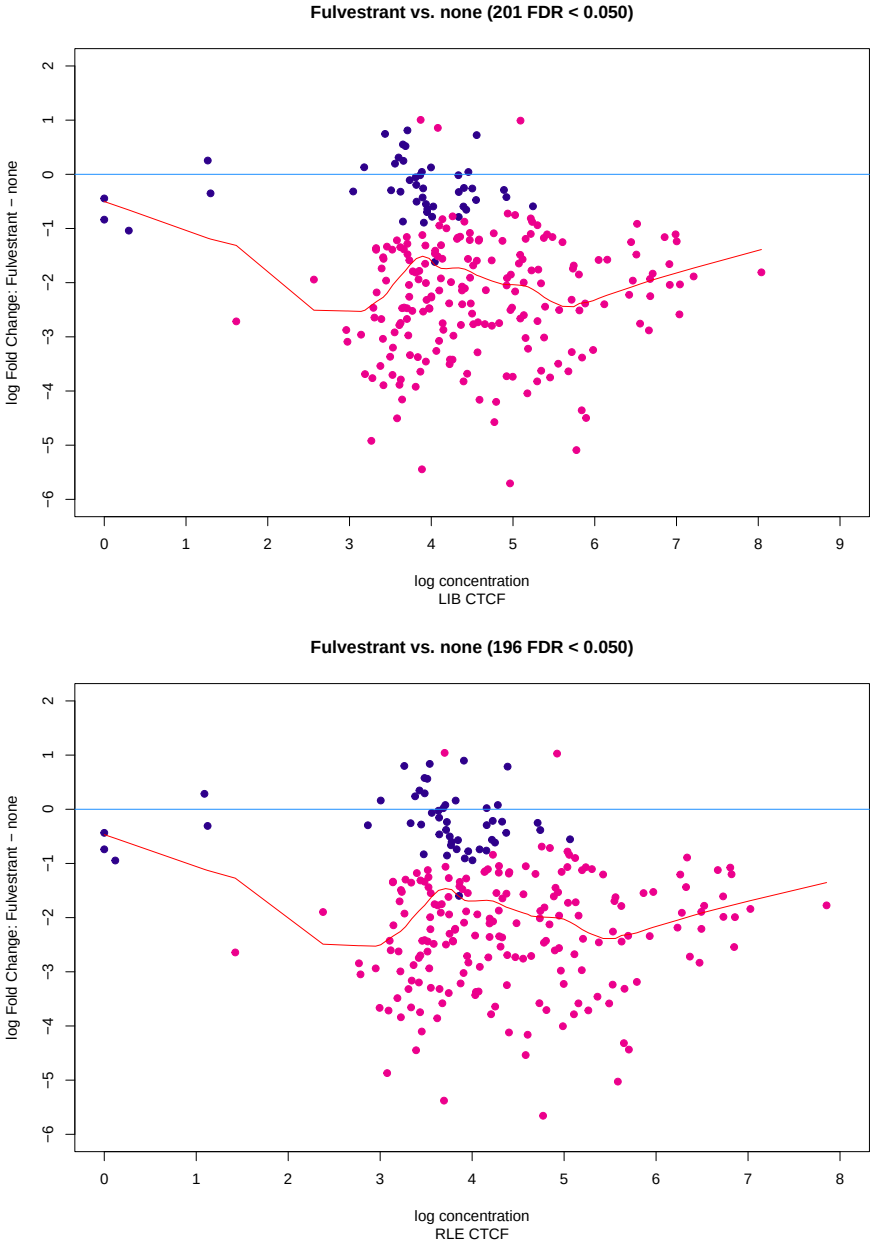


Figure 34: MA plots of ER dataset with and without Fulvestrant treatment, using reads overlapping CTCF sites as a background, normalized using the number of reads overlapping CTCF sites, and `RLE` over the CTCF sites.

In the absence of spike-ins or a parallel factors, the "safest" method is probably to set `background=TRUE` and `normalize=DBA_NORM_NATIVE`, resulting in the use of background reads and the native normalization method (`TMM` for *edgeR* , and `RLE` for *DESeq2*). This can be approximated at very low computational cost, with no extra reading of bam files, by the default settings of `library=DBA_LIBSIZE_FULL`, `normalize=DBA_NORM_LIBRARY`, and `background=FALSE`.

Reference	Normalization			
	lib	RLE	TMM	loess
full	X			
RiP	X	X	X	X
background	X	X	X	
spike-in	X	X	X	
parallel factor	X	X	X	

Table 1: Table of allowable normalization schemes. Columns are normalization methods set by `normalize` or `offsets`. Rows are reference reads, set by `library`, `background`, or `spikein`.

If a well-characterized parallel factor is available, or if one is available for "free" in the case of certain histone mark ChIPs, this is probably preferable to spike-ins, given the difficulties in initial spike-in quantification.

Only in certain cases where indicated by prior knowledge is the use of the main count matrix, based on consensus peaks, appropriate.

8 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)
```

8.1 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 10.2 on peak merging). A global overview can be obtained using the `RATE` mode of the `dba.overlap` function as follows:

```
> 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')
```

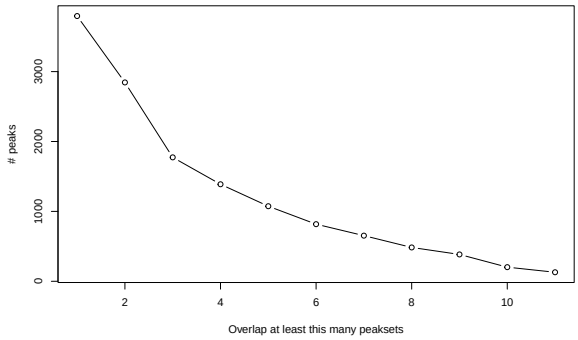


Figure 35: Overlap rate plot. Shows how the number of overlapping peaks decreases as the overlap criteria becomes more stringent. X axis shows the number of peaksets in which the site is identified, while the Y axis shows the number of overlapping sites. Generated by plotting the result of: `dba.overlap(tamoxifen,mode=DBA_OLAP_RATE)`

The rate plot is shown in Figure 35. 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.

8.2 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.[12] 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 ([12]).

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 the Responsive MCF7 samples can be isolated 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"
 [6] "Resistant"   "Responsive"  "Full-Media"  "bed"         "Replicate.1"
[11] "Replicate.2" "Replicate.3" "All"         "None"
```

[12] 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.

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)
```

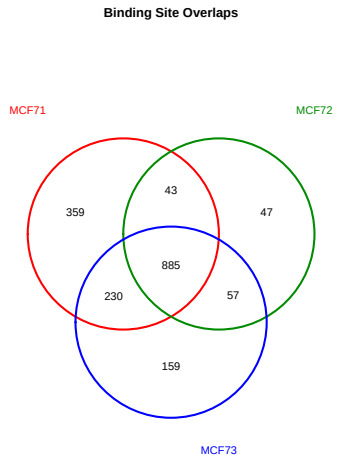


Figure 36: Venn diagram showing how the ER peak calls for three replicates of responsive MCF7 cell line overlap. Generated by plotting the result of: `dba.venn(tamoxifen,tamoxifen$masks$MCF7&tamoxifen$masks$Responsive)`

The resultant plot is shown as Figure 36. 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 (eg. 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_consensus <- dba.peakset(tamoxifen,
+                                    consensus=c(DBA_TISSUE,DBA_CONDITION),
+                                    minOverlap=0.66)
```

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_consensus <- 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_consensus,
+                            mask=tamoxifen_consensus$masks$Consensus,
+                            minOverlap=1)
> tamoxifen_consensus

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

and an overall consensus peakset, that includes peaks identified in at least two replicates of at least one sample group, can be identified:

```
> consensus_peaks <- dba.peakset(tamoxifen_consensus, bRetrieve=TRUE)
```

This consensus peakset could then be used as the basis for the binding matrix used in `dba.count`:

```
tamoxifen <- dba.count(tamoxifen, peaks=consensus_peaks)
```

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 37):

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

Figure 37 shows how consensus peaksets derived for each cultured cell type overlap. The ZR75 samples stand out for having 1075 peaks common to both replicates that are not identified in any other cell type.

8.3 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:

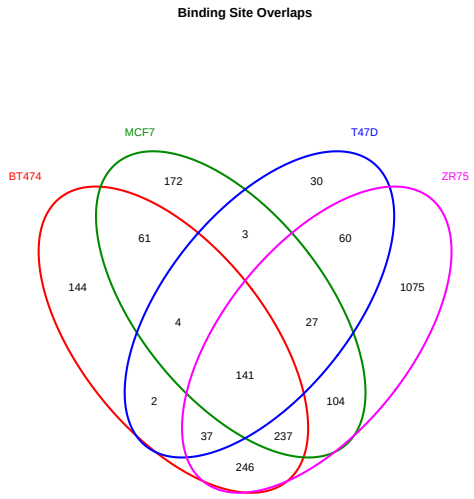


Figure 37: Venn diagram showing how the consensus peaks for each cell type overlap. Generated by plotting the result of: `dba.venn(tamoxifen,tamoxifen$masks$Consensus)`

```
> 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 38 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`:

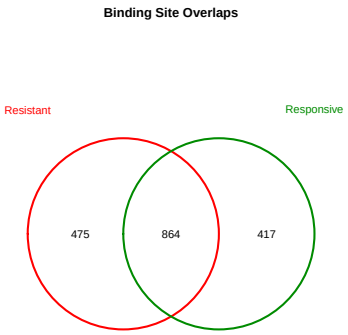


Figure 38: Venn diagram showing how the ER peak calls for two response groups overlap. Generated by plotting the result of: `dba.plotVenn(tamoxifen,tamoxifen$masks$Consensus)`

```
> tamoxifen.OL$onlyA

GRanges object with 475 ranges and 3 metadata columns:
       seqnames            ranges strand |     score    scoreA    scoreB
          <Rle>         <IRanges>  <Rle> | <numeric> <numeric> <numeric>
     2    chr18     150764-151269      * | 0.0216970        NA        NA
     3    chr18     188982-189652      * | 0.0829604        NA        NA
     5    chr18     311530-312172      * | 0.0647360        NA        NA
     7    chr18     356560-357362      * | 0.0264811        NA        NA
     8    chr18     371110-372102      * | 0.0327930        NA        NA
   ...      ...               ...    ... .       ...       ...       ...
  1731    chr18 76528540-76529618      * | 0.0367731        NA        NA
  1744    chr18 77056886-77057516      * | 0.0242664        NA        NA
  1745    chr18 77062037-77062828      * | 0.0233994        NA        NA
  1747    chr18 77300430-77301170      * | 0.0386595        NA        NA
  1750    chr18 77424530-77425198      * | 0.0280821        NA        NA
  -------
  seqinfo: 1 sequence from an unspecified genome; no seqlengths

> tamoxifen.OL$onlyB

GRanges object with 417 ranges and 3 metadata columns:
       seqnames            ranges strand |     score    scoreA    scoreB
          <Rle>         <IRanges>  <Rle> | <numeric> <numeric> <numeric>
     1    chr18     111564-112005      * | 0.0465362        NA        NA
     6    chr18     346464-347342      * | 0.0589574        NA        NA
    24    chr18     812595-813462      * | 0.0526327        NA        NA
    32    chr18   1075317-1076051      * | 0.0670931        NA        NA
    37    chr18   1241658-1242455      * | 0.0414764        NA        NA
```

```
  ...      ...                ...    ... .       ...       ...       ...
1742    chr18 76805366-76806312      * | 0.0398786        NA        NA
1748    chr18 77318446-77319078      * | 0.0358981        NA        NA
1749    chr18 77389690-77390304      * | 0.0237104        NA        NA
1752    chr18 77541035-77541645      * | 0.0499611        NA        NA
1756    chr18 77987044-77988289      * | 0.3000995        NA        NA
-------
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.

8.4 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)
```

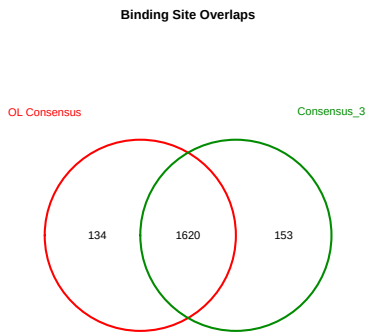


Figure 39: Venn diagram showing how the ER peak calls for two different ways of deriving consensus peaksets. Generated by plotting the result of: `dba.plotVenn(tamoxifen,14:15)`

Figure 39 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.[13]

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=TRUE`):

```
> 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:

```
> 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 38, 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 $<=$ 0.05), however, shows a different story than that obtainable using only occupancy data:

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

[1] 129

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

[1] 226
```

[13] 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)`

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

[1] 293

> neitherGroup.DB   <- (tamoxifen.DB$Called1  < 2) & (tamoxifen.DB$Called2 < 3)
> sum(neitherGroup.DB)

[1] 135
```

There are a number of notable differences in the results.

First, overall there are fewer sites identified as differentially bound (783) than are sites identified as being unique to one condition (473+417 == 890). Only about 40% of sites unique to one condition are identifiable as significantly differentially bound (129+226 = 355 out of 890). Focusing only on sites unique to one condition would result in many false positives; 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.

Second, 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; of the total sites identified as significantly differentially bound, (783), 37% are called as peaks in *both* response groups (293).

Third, the largest single group of differentially bound sites (135) were not identified as being consistently associated with *either* sample group (peaks called no more than no Resistant sample and no more than 2 Responsive samples), yet were still shown to have significantly different read densities. Any reasonable criteria for isolating peaks in an occupancy analysis would miss these completely, resulting in many false negatives.

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.

9 Backward compatibility with pre-version 3.0 analyses

It is recommended that existing analyses be re-run with the current software. Existing scripts should execute (with the exception of two normalization parameters which have been moved from `dba.analyze` to the new interface function `dba.normalize`.)

Most existing *DiffBind* scripts and saved objects will run correctly using version 3.0, but there may be differences in the results.

This section describes how to approximate earlier results for existing scripts and objects.

9.1 Running with saved DBA objects

If a `DBA` object was created with an earlier version of *DiffBind* , and saved using the `dba.save` function, and loaded using the `dba.load` function, all settings should be preserved, such that running the analysis anew will yield similar results.

In order to re-run the analysis using the post-version 3.0 settings, the original script should be used to re-create the `DBA` object.

9.2 Re-running *DiffBind* scripts

By default, if you re-run a *DiffBind* script, it will use the new defaults from version 3.0 and beyond. In order to re-analyze an experiment in the pre-version 3.0 mode, a number of defaults need to be changed.

When calling `dba.count`, the following defaults are changed:

1. `summits:` This parameter is now set by default. Setting `summits=FALSE` will preempt re-centering each peak interval around its point of highest pileup.
2. `filter:` The new default for this parameter is `1` and is based on `RPKM` values; previously it was set to `filter=0` and was based on read counts. previously it was set to `filter=0`.
3. `minCount:` This is a new parameter representing a minimum read count value. It now default to `0`; to get the previous behavior, set `minCount=1`.
4.

The easiest way to perform subsequent processing in a pre-version 3.0 manner is to set a configuration option:

```
> DBA$config$design <- FALSE
```

This will result in the appropriate defaults being set for the new interface function, `dba.normalize` (which does not need to be invoked explicitly.) The pre-version 3.0 settings for `dba.normalize` parameters are as follows:

1. `normalize:` `DBA_NORM_DEFAULT`
2. `library:` `DBA_LIBSIZE_FULL`
3. `background:` `FALSE`

Note that two parameters that used to be available when calling `dba.analyze` have been moved:

1. `bSubControl:` now integrated into `dba.count`. `TRUE` by default (unless a greylist has been added using `dba.blacklist`).
2. `bFullLibrarySize:` This is now part of the `library` parameter for `dba.normalize`. `library=DBA_LIBSIZE_FULL` is equivalent to `bFullLibrarySize=TRUE`, and `library=DBA_LIBSIZE_PEAKREADS` is equivalent to `bFullLibrarySize=FALSE`.

10 Technical notes

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

10.1 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.

10.2 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. As 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.

10.3 Details of *DESeq2* analysis

When `dba.analyze` is invoked using the default method `method=DBA_DESEQ2`, a standardized differential analysis is performed using the *DESeq2*package ([6]). This section details the steps in that analysis.

First, a matrix of counts is constructed using all the samples in the experiment, with rows for each consensus interval, and columns for each sample in the experiment. The raw read count is used for this matrix; if the `bSubControl` parameter is set to `TRUE`, the raw number of reads in the control sample (if available) will be subtracted. And read counts that are less than `minCount` are set to `minCount` (default is zero).

A `DESeqDataSet` object is then created using `DESeq2::DESeqDataSetFromMatrix` with the count matrix, metadata from the `DBA` object, and the design formula.

Next the normalization parameters are set for the analysis. If the normalization involves `offsets`, the *DESeq2* normalization factors are set to the offsets; if the `DBA_NORM_OFFSETs_ADJUST` is set, the offsets are assumed to be in *edgeR* format and are run through the `edgeR::scaleOffset` function, adjusted to be mean centered on 1, then normalized to library size. If offsets are not used, then `DESeq::sizeFactors` is called with the scaling factors computed in `dba.normalize`. If these were based on the native `RLE` method, then the result of a previous call to `DESeq2::estimateSizeFactors` is used.

`DESeq2::estimateDispersions` is then called with the *DESeqDataSet* object. By default the `fitType` will be set to `local`; this can be overridden by setting a configuration option `DBA$config$DESeq2$fitType` to the desired value.

Finally the model is fitted and tested using the `DESeq2::nbinomWaldTest`, with defaults.

The final results, as a *DESeqDataSet* object, are accessible by calling `dba.analyze()` with `bRetrieveAnalysis=DBA_DESeq2` (or `bRetrieveAnalysis=TRUE`).

When a contrast is evaluated, the results are obtained by calling `DESeq2::results`. If the contrast was specified to `dba.contrast()` using a single character string containing the name of a specific column coefficient, the call is made with `name` set to the character string. Otherwise, the call is made with `contrast` set appropriate (either as a vector of three character strings representing a design `Factor`, and two values for that Factor, or a list of one or two character strings representing coefficient names, or as a numeric vector of the same length as the number of coefficients in the design matrix. Note that coefficient names can be retrieved by calling `dba.contrast()` with `bRetrieveCoefficients=TRUE`; these are obtained from a call to `DESeq2::resultsNames`.

The fold changes used in subsequent reports and plots are the shrunk values estimated from the `results` using `DESeq2::lfcShrink`.

When retrieving or plotting results (e.g. calling `dba.report()`, `dba.plotMA()`,or `dba.plotVolcano()`), if a `fold` cutoff is specified other than `0.0`, the results are re-computed using `DESeq2::results` with the `lfcThreshold` supplied. Fold values are re-computed as well (using `DESeq2::lfcShrink`).

10.4 Details of *edgeR* analysis

When `dba.analyze` is invoked using the `method=DBA_EDGER`[14], a standardized differential analysis is performed using the *edgeR* package ([13]). This section details the steps in that analysis.

First, a matrix of counts is constructed using all the samples in the experiment, with rows for each consensus interval, and columns for each sample in the experiment. The raw read count is used for this matrix; if the `bSubControl` parameter is set to `TRUE`, the raw number of reads in the control sample (if available) will be subtracted. And read counts that are less than `minCount` are set to `minCount` (default is zero).

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

Next an *edgeR* `DGEList` is created by calling `edgeR::DGEList` with the count matrix, the pre-computed library sizes and normalization factors (from `dba.normalize`), and meta-data derived from the `DBA` object. If these were based on the `TMM` or `RLE` methods, then the result of a previous call to `edgeR::calcNormFactors` (with `doWeighting=FALSE`) is used as the normalization factors.

If the normalization involves `offsets`, the offsets are retrieved. If the `DBA_NORM_OFFSETs_ADJUST` is set, the offsets `edgeR::scaleOffset` function before being set as the offsets in the `DGEList`.

Next `edgeR::estimateGLMTrendedDisp` is called with the `DGEList` and a design matrix derived from the design formula. By default, `edgeR::estimateGLMTagwiseDisp` is called next; this can be bypassed by setting `DBA$config$edgeR$bTagwise=FALSE`.[15]

The model is fitted by calling `edgeR::glmQLFit` with the design matrix. The final results, as a *DGEGLM* object, are accessible by calling `dba.analyze()` with `bRetrieveAnalysis=DBA_EDGER`.

Each specified contrast is evaluated in two steps. First the main test is performed by a call to First is a call to `edgeR::glmQLFTest` with the fitted model and the values for each coefficient in the design matrix. If the contrast was specified to `dba.contrast()` using values for each of the coefficients in the design matrix, those values are used. Otherwise, if the contrast was specified using a single character string containing a) the name of a specific column coefficient; b) a vector of three character strings representing a design `Factor`, and two values for that Factor; or c) list of one or two character strings representing coefficient names, a numeric vector of the same length as the number of coefficients is derived. Note that allowable coefficient names (which differ from those used internally by *edgeR*) can be retrieved by calling `dba.contrast()` with `bRetrieveCoefficients=TRUE`.

Finally, `edgeR::topTags` is called to retrieve the results of the test. The fold changes used in subsequent reports and plots are those estimated from the `TopTags`.

When retrieving or plotting results (e.g. calling `dba.report()`, `dba.plotMA()`,or `dba.plotVolcano()`), if a `fold` cutoff is specified other than `0.0`, the results are re-computed using `edgeR::glmTreat` instead of `edgeR::glmQLFTest`.

11 Technical notes for versions prior to *DiffBind* 3.0 (without an explicit model design)

Prior to version `3.0`, *DiffBind* did not offer a way to explicitly specify a model design. The technical details of how it performed analyses are described here.

Note that these methods are still supported. In order to conduct analyses in the former style, the user must call `dba.contrast()` with `design=FALSE`. Contrasts can be added either automatically or explicitly (specifying at least `groups1`), and a `blocking` fact may also be specified.

11.1 *DESeq2* analysis

When `dba.analyze` is invoked using the default method `method=DBA_DESEQ2`, a standardized differential analysis is performed using the *DESeq2*package ([6]). This section details the precise steps in that analysis.

[15] In versions prior to 3.0, `bTagwise` was a parameter to `dba.analyze()`.

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. The first step concludes with a call to *DESeq2*'s `DESeqDataSetFromMatrix` function, which returns a *DESeqDataSet* object.

If `bFullLibrarySize` is set to TRUE (default), 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`.

11.2 *edgeR* analysis

When `dba.analyze` is invoked using the `method=DBA_EDGER`[16], a standardized differential analysis is performed using the *edgeR* package ([13]). 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 (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.

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

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` ([14]) 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` ([15]) 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`.

12 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 https://www.cruk.cam.ac.uk/core-facilities/bioinformatics-core/software/DiffBind.

The following code can be used to download and set up the vignette data:

This workbook requires the sample data used for the 'DiffBind' vignette. These data can be obtained as follows:

```
> tmpdir <- tempdir()
> url <- 'https://content.cruk.cam.ac.uk/bioinformatics/software/DiffBind/DiffBind_vignette_data.tar.gz'
> file <- basename(url)
> options(timeout=600)
> download.file(url, file.path(tmpdir,file))
> untar(file.path(tmpdir,file), exdir = tmpdir )
> setwd(file.path(tmpdir,"DiffBind_Vignette"))
```

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

13 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.

14 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.

15 Session Info

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

- R version 4.6.1 (2026-06-24), `x86_64-pc-linux-gnu`
- Locale: `LC_CTYPE=en_US.UTF-8`, `LC_NUMERIC=C`, `LC_TIME=en_GB`, `LC_COLLATE=C`, `LC_MONETARY=en_US.UTF-8`, `LC_MESSAGES=en_US.UTF-8`, `LC_PAPER=en_US.UTF-8`, `LC_NAME=C`, `LC_ADDRESS=C`, `LC_TELEPHONE=C`, `LC_MEASUREMENT=en_US.UTF-8`, `LC_IDENTIFICATION=C`
- Time zone: `America/New_York`
- TZcode source: `system (glibc)`

- Running under: `Ubuntu 24.04.5 LTS`
- Matrix products: default
- BLAS: `/home/biocbuild/bbs-3.24-bioc/R/lib/libRblas.so`
- LAPACK: `/usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.12.0`
- Base packages: base, datasets, grDevices, graphics, methods, stats, stats4, utils
- Other packages: Biobase 2.74.0, BiocGenerics 0.60.0, DiffBind 3.24.0, GenomicRanges 1.66.0, IRanges 2.48.0, MatrixGenerics 1.26.0, S4Vectors 0.52.0, Seqinfo 1.4.0, SummarizedExperiment 1.44.0, generics 0.1.4, matrixStats 1.5.0, xtable 1.8-8
- Loaded via a namespace (and not attached): BSgenome 1.82.0, BiocBaseUtils 1.16.0, BiocIO 1.24.0, BiocManager 1.30.27, BiocParallel 1.48.0, BiocStyle 2.42.0, Biostrings 2.82.0, DESeq2 1.54.0, DelayedArray 0.40.0, GenomeInfoDb 1.50.0, GenomicAlignments 1.50.0, GreyListChIP 1.46.0, KernSmooth 2.23-27, MASS 7.3-66, Matrix 1.7-6, R6 2.6.1, RColorBrewer 1.1-3, RCurl 1.98-1.20, Rcpp 1.1.2, Rsamtools 2.30.0, S4Arrays 1.14.0, S7 1.0.0, SQUAREM 2026.1, ShortRead 1.72.0, SparseArray 1.14.0, UCSC.utils 1.10.0, XML 3.99-0.25, XVector 0.54.0, abind 1.4-8, amap 0.8-20, apeglm 1.36.0, ashr 2.2-63, bbmle 1.0.26, bdsmatrix 1.3-7, bitops 1.1-0, caTools 1.18.4, cigarillo 1.4.0, cli 3.6.6, coda 0.19-4.1, codetools 0.2-20, compiler 4.6.1, crayon 1.5.3, csaw 1.48.0, curl 8.2.0, deldir 2.0-4, dichromat 2.0-1, digest 0.6.39, dplyr 1.2.1, edgeR 5.0.0, emdbook 1.3.14, evaluate 1.0.5, farver 2.1.2, fastmap 1.2.0, ggplot2 4.0.4, ggrepel 0.9.8, glue 1.8.1, gplots 3.3.0, grid 4.6.1, gtable 0.3.6, gtools 3.9.5, htmltools 0.5.9, htmlwidgets 1.6.4, httr 1.4.9, hwriter 1.3.2.1, interp 1.1-6, invgamma 1.2, irlba 2.4.1, jpeg 0.1-11, jsonlite 2.0.0, knitr 1.52, labeling 0.4.3, lattice 0.23-1, latticeExtra 0.6-31, lifecycle 1.0.5, limma 4.0.0, locfit 1.5-9.12, magrittr 2.0.5, metapod 1.22.0, mixsqp 0.3-54, mvtnorm 1.4-2, numDeriv 2016.8-1.1, otel 0.3.0, parallel 4.6.1, pillar 1.11.1, pkgconfig 2.0.3, plyr 1.8.9, png 0.1-9, pwalign 1.10.0, restfulr 0.0.17, rjson 0.2.23, rlang 1.3.0, rmarkdown 2.32, rtracklayer 1.74.0, scales 1.4.0, splines 4.6.1, statmod 1.5.2, stringi 1.8.9, stringr 1.6.0, systemPipeR 2.20.0, tibble 3.3.1, tidyselect 1.2.1, tools 4.6.1, truncnorm 1.0-9, vctrs 0.7.3, withr 3.0.3, xfun 0.62, yaml 2.3.12

References

[1] Y. Zhang, T. Liu, C.A. Meyer, J. Eeckhoute, D.S. Johnson, B.E. Bernstein, C. Nussbaum, R.M. Myers, M. Brown, W. Li, et al. Model-based analysis of ChIP-Seq (MACS). *Genome Biol*, 9(9):R137, 2008.

[2] C.S. Ross-Innes, R. Stark, A.E. Teschendorff, K.A. Holmes, H.R. Ali, M.J. Dunning, G.D. Brown, O. Gojis, I.O. Ellis, A.R. Green, et al. Differential oestrogen receptor binding is associated with clinical outcome in breast cancer. *Nature*, 481(7381):389–393, 2012.

[3] Haley M. Amemiya, Anshul Kundaje, and Alan P. Boyle. The ENCODE blacklist: Identification of problematic regions of the genome. *Scientific Reports*, 9(1):9354, 2019.

[4] Gord Brown. GreyListChIP: Grey lists–mask artefact regions based on ChIP inputs, 2015. URL: https://bioconductor.org/packages/GreyListChIP/, doi:10.18129/B9.bioc.GreyListChIP.

[5] Thomas S. Carroll, Ziwei Liang, Rafik Salama, Rory Stark, and Ines de Santiago. Impact of artifact removal on ChIP quality metrics in ChIP-seq and ChIP-exo data. *Frontiers in Genetics*, 5:75, 2014. URL: https://www.frontiersin.org/article/10.3389/fgene.2014.00075, doi:10.3389/fgene.2014.00075.

[6] Michael I. Love, Wolfgang Huber, and Simon Anders. Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2. *Genome Biology*, 15:550, 2014. URL: http://dx.doi.org/10.1186/s13059-014-0550-8.

[7] Mark D Robinson and Alicia Oshlack. A scaling normalization method for differential expression analysis of RNA-seq data. *Genome biology*, 11(3):1–9, 2010.

[8] Aaron TL Lun and Gordon K Smyth. csaw: a Bioconductor package for differential binding analysis of ChIP-seq data using sliding windows. *Nucleic acids research*, 44(5):e45–e45, 2016.

[9] Jake J Reske, Mike R Wilson, and Ronald L Chandler. ATAC-seq normalization method can significantly affect differential accessibility analysis and interpretation. *Epigenetics & chromatin*, 13:1–17, 2020.

[10] Michael J Guertin, Amy E Cullen, Florian Markowetz, and Andrew N Holding. Parallel factor ChIP provides essential internal control for quantitative differential ChIP-seq. *Nucleic acids research*, 46(12):e75–e75, 2018.

[11] Manuel Allhoff, Kristin Seré, Juliana F. Pires, Martin Zenke, and Ivan G. Costa. Differential peak calling of chip-seq signals with replicates with thor. *Nucleic acids research*, 44(20):e153–e153, 2016.

[12] Q. Li, J.B. Brown, H. Huang, and P. Bickel. Measuring reproducibility of high-throughput experiments. *Annals of Applied Statistics*, 2011.

[13] Mark D Robinson, Davis J McCarthy, and Gordon K Smyth. edgeR: a Bioconductor package for differential expression analysis of digital gene expression data. *Bioinformatics*, 26(1):139–40, Jan 2010. URL: http://bioinformatics.oxfordjournals.org/cgi/content/full/26/1/139, doi:10.1093/bioinformatics/btp616.

[14] M.D. Robinson and G.K. Smyth. Moderated statistical tests for assessing differences in tag abundance. *Bioinformatics*, 23(21):2881–2887, 2007.

[15] D.J. McCarthy, Y. Chen, and G.K. Smyth. Differential expression analysis of multifactor RNA-Seq experiments with respect to biological variation. *Nucleic Acids Research*, 2012.