DiffBind : differential binding analysis of ChIP-Seq peak data

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

Gordon Brown
gordon.brown@cruk.cam.ac.uk

Cancer Research UK
Cambridge Research Institute

14 February 2013

Contents

1 Introduction **2**

2 Processing overview **2**

3 Example: obtaining differentially bound sites **4**
3.1 Reading in the peaksets . . . 4
3.2 Counting reads . . . 7
3.3 Establishing a contrast . . . 7
3.4 Performing the differential analysis . . . 7
3.5 Retrieving the differentially bound sites . . . 9

4 Example: plotting **11**
4.1 Venn diagrams . . . 12
4.2 MA plots . . . 12
4.3 PCA plots . . . 13
4.4 Boxplots . . . 14
4.5 Heatmaps . . . 16

5 Example: differential binding analysis using a blocking factor **18**

6 Example: occupancy analysis and overlaps **23**
6.1 Overlap rates . . . 23
6.2 Deriving consensus peaksets . . . 24
6.3 A complete occupancy analysis: identifying sites unique to a sample group . . . 29

6.4 Comparison of occupancy and affinity based analyses 31

7 Technical notes **34**
7.1 Loading peaksets 34
7.2 Merging peaks 35
7.3 edgeR analysis 35
7.4 DESeq analysis 36

8 Acknowledgements **37**

9 Setup **37**

1 Introduction

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

The primary emphasis of the package is on identifying sites that are differentially bound between two sample groups. It includes functions to support the processing of peak sets, including overlapping and merging peak sets, counting sequencing reads overlapping intervals in peak sets, and identifying statistically significantly differentially bound sites based on evidence of binding affinity (measured by differences in read densities). To this end it uses statistical routines developed in an RNA-Seq context (primarily the Bioconductor packages edgeR and DESeq). 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 three sections of examples: the first focusing on the core task of obtaining differentially bound sites based on affinity data, the second working through the main plotting routines, and the third revisiting occupancy data (peak calls) in more detail, as well as comparing the results of an occupancy-based analysis with an affinity-based one. Finally, some 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 (BAM//BED) can be associated with each peakset (one for the ChIP data, and optionally another representing a control dataset).

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 (Zhang et al. [2008]), or using some other criterion (e.g. 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. A single experiment can have more than one associated peakset, e.g. if multiple peak callers are used for comparison purposes, and hence have more than one line in the sample sheet. Once the peaksets are read in, a merging function finds all overlapping peaks and derives a single set of unique genomic intervals covering all the supplied peaks.

2. `Occupancy analysis`: Peaksets, especially those generated by peak callers, provide an insight into the potential *occupancy* of the protein being ChIPed for at specific genomic loci. After the peaksets have been loaded, it can be useful to perform some exploratory plotting to determine how these occupancy maps agree with each other, e.g. between experimental replicates (re-doing the ChIP under the same conditions), between different peak callers on the same experiment, and within groups of samples representing a common experimental condition. DiffBind provides functions to enable overlaps to be examined, as well as functions to determine how well similar samples cluster together. Beyond quality control, the product of an occupancy analysis may be a *consensus peakset*, representing an overall set of candidate binding sites to be used in further analysis.

3. `Counting reads`: Once a consensus peakset has been derived, DiffBind can use the supplied sequence read files to count how many reads overlap each interval for each unique sample. The result of this is a *binding affinity matrix* containing a (normalized) read count for each sample at every potential binding site. With this matrix, the samples can be re-clustered using affinity, rather than occupancy, data. The binding affinity matrix is used for QC plotting as well as for subsequent differential analysis.

4. `Differential binding affinity analysis`: The core functionality of DiffBind is the differential binding affinity analysis, which enables binding sites to be identified that are statistically significantly differentially bound between sample groups. To accomplish this, first a contrast (or contrasts) is established, dividing the samples into groups to be compared. Next the core analysis routines are executed, by default using `edgeR` . This will assign a p-value and FDR to each candidate binding site indicating the significance of their being differentially bound.

5. `Plotting and reporting`: Once one or more contrasts have been run, DiffBind provides a number of functions for reporting and plotting the results. MA plots give an overview of the results of the analysis, while correlation heatmaps and PCA plots show how the groups cluster based on differentially bound sites. Boxplots show the distribution of reads within differentially bound sites corresponding to whether they gain or lose affinity between the two

sample groups. A reporting mechanism enables differentially bound sites to be extracted for further processing, such as annotation and/or pathway analysis.

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 (Ross-Innes et al. [2012]). Three of these cell lines are responsive to tamoxifen, while two others are resistant to tamoxifen treatment. 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. Note that this experiment includes two types of MCF7 cells: the regular tamoxifen responsive line as well as MCF7 cells specially treated with tamoxifen until a tamoxifen resistant cell line is obtained. For each sample, we have one peakset originally derived using the MACS peak caller (Zhang et al. [2008]), for a total of eleven peaksets. Note that to save space in the package, only data for chromosome 18 is used. The metadata and peak data are available in the `extra` subdirectory of the DiffBind package directory; you can make this your working directory by entering:

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

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

```
> tamoxifen = dba(sampleSheet="tamoxifen.csv")
> tamoxifen = dba.count(tamoxifen)
> tamoxifen = dba.contrast(tamoxifen, categories=DBA_CONDITION)
> tamoxifen = dba.analyze(tamoxifen)
> tamoxifen.DB = dba.report(tamoxifen)
```

The following subsections describe these steps in more detail.

3.1 Reading in the peaksets

Table 1 shows the sample sheet, saved in a file called `tamoxifen.csv`. The peaksets are read in using the following DiffBind function:

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

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

Table 1: Tamoxifen dataset sample sheet (tamoxifen.csv).

SampleID	Tissue	Factor	Condition	Replicate	bamReads	bamControl	Peaks
BT474.1-	BT474	ER	Resistant	1	BT474_ER_1.bed.gz	BT474_Input.bed.gz	BT474_ER_1.bed.gz
BT474.2-	BT474	ER	Resistant	2	BT474_ER_2.bed.gz	BT474_Input.bed.gz	BT474_ER_2.bed.gz
MCF7.1+	MCF7	ER	Responsive	1	MCF7_ER_1.bed.gz	MCF7_Input.bed.gz	MCF7_ER_1.bed.gz
MCF7.2+	MCF7	ER	Responsive	2	MCF7_ER_2.bed.gz	MCF7_Input.bed.gz	MCF7_ER_2.bed.gz
MCF7.3+	MCF7	ER	Responsive	3	MCF7_ER_3.bed.gz	MCF7_Input.bed.gz	MCF7_ER_3.bed.gz
T47D.1+	T47D	ER	Responsive	1	T47D_ER_1.bed.gz	T47D_Input.bed.gz	T47D_ER_1.bed.gz
T47D.2+	T47D	ER	Responsive	2	T47D_ER_2.bed.gz	T47D_Input.bed.gz	T47D_ER_2.bed.gz
MCF7.1-	MCF7	ER	Resistant	1	TAMR_ER_1.bed.gz	TAMR_Input.bed.gz	TAMR_ER_1.bed.gz
MCF7.2-	MCF7	ER	Resistant	2	TAMR_ER_2.bed.gz	TAMR_Input.bed.gz	TAMR_ER_2.bed.gz
ZR75.1+	ZR75	ER	Responsive	1	ZR75_ER_1.bed.gz	ZR75_Input.bed.gz	ZR75_ER_1.bed.gz
ZR75.2+	ZR75	ER	Responsive	2	ZR75_ER_2.bed.gz	ZR75_Input.bed.gz	ZR75_ER_2.bed.gz

```
> tamoxifen

11 Samples, 2602 sites in matrix (3557 total):
         ID Tissue Factor  Condition Replicate Peak.caller Intervals
1  BT474.1-  BT474     ER  Resistant         1         raw      1084
2  BT474.2-  BT474     ER  Resistant         2         raw      1115
3   MCF7.1+   MCF7     ER Responsive         1         raw      1513
4   MCF7.2+   MCF7     ER Responsive         2         raw      1037
5   MCF7.3+   MCF7     ER Responsive         3         raw      1372
6   T47D.1+   T47D     ER Responsive         1         raw       509
7   T47D.2+   T47D     ER Responsive         2         raw       347
8   MCF7.1-   MCF7     ER  Resistant         1         raw      1148
9   MCF7.2-   MCF7     ER  Resistant         2         raw       933
10  ZR75.1+   ZR75     ER Responsive         1         raw      2111
11  ZR75.2+   ZR75     ER Responsive         2         raw      1975
```

This shows how many peaks are in each peakset, as well as (in the first line) total number of unique peaks after merging overlapping ones (3,557) and the default binding matrix of 11 samples by the 2,602 sites that overlap in at least two of the samples. This object is available for loading using `data(tamoxifen_peaks)`.

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

```
> plot(tamoxifen)
```

The resulting plot (Figure 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 MCF7-) 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.

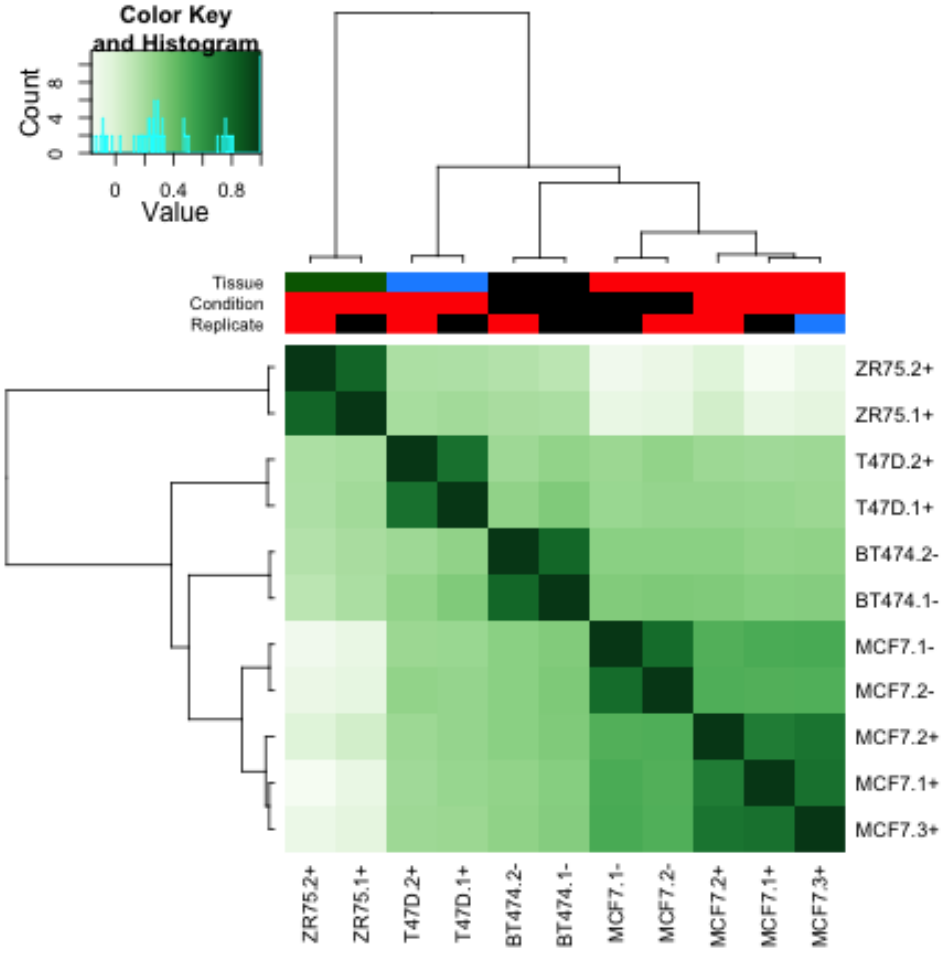


Figure 1: Correlation heatmap, using occupancy (peak caller score) data. Generated by: `plot(tamoxifen)`; can also be generated by: `dba.plotHeatmap(tamoxifen)`.

3.2 Counting reads

The next step is to calculate a binding matrix with scores based on read counts for every sample (affinity scores), rather than confidence scores for only those peaks called in a specific sample (occupancy scores). These reads are obtained using the `dba.count` function:[1]

```
> tamoxifen = dba.count(tamoxifen, minOverlap=3)
```

If you do not have the raw reads available to you, this object is available loading using `data(tamoxifen_counts)`. The `dba.count` call plots a new correlation heatmap based on the affinity scores, seen in Figure 2a. While this shows overall higher correlation levels (evidenced by the shift in the scale), and a slightly different clustering, responsiveness to tamoxifen treatment does not appear to form a basis for clustering when using all of the affinity scores. (Note that the clustering can change based on what scoring metric is used; see Section 4.4 for more details).

3.3 Establishing a contrast

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

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

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

3.4 Performing the differential analysis

The main differential analysis function is invoked as follows:

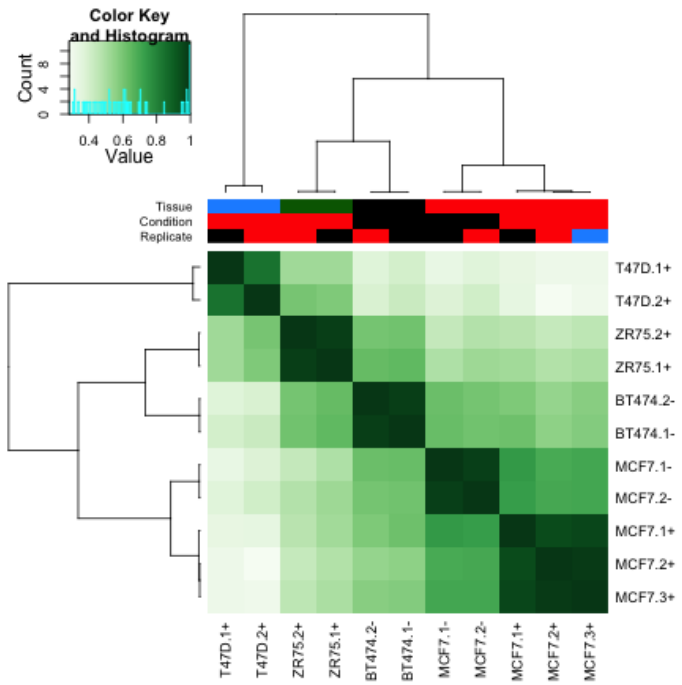
```
> tamoxifen = dba.analyze(tamoxifen)
```

This will run an `edgeR` analysis (see subsequent section discussing the technical details of the `edgeR` analysis) on the binding affinity matrix. Displaying the resultant `DBA object` shows that 235 of the 1,654 sites are identified as being significantly differentially bound (DB) using the default threshold of FDR <= 0.1:

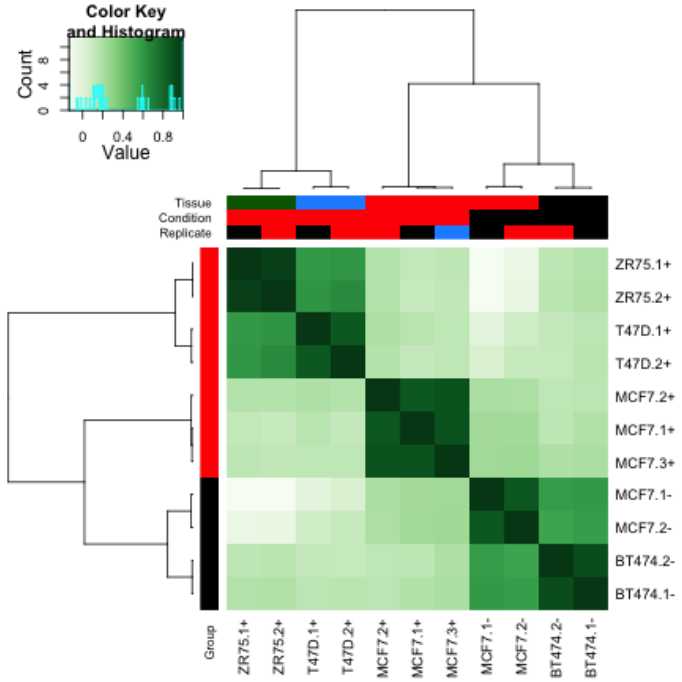
```
> tamoxifen
```

[1] Note that due to space limitations the reads are not shipped with the package. We hope to make them easily available in the future. Alternatively, you can get the result of the `dba.count` call by loading the supplied R object by invoking `data(tamoxifen_counts)`

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



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



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

Figure 2: Correlation heatmap plots, generated using `dba.plotHeatmap`.

```
11 Samples, 1654 sites in matrix:
         ID Tissue Factor  Condition Replicate Peak.caller Intervals   SN
1  BT474.1-  BT474     ER  Resistant         1      counts      1654 0.19
2  BT474.2-  BT474     ER  Resistant         2      counts      1654 0.18
3   MCF7.1+   MCF7     ER Responsive         1      counts      1654 0.35
4   MCF7.2+   MCF7     ER Responsive         2      counts      1654 0.22
5   MCF7.3+   MCF7     ER Responsive         3      counts      1654 0.28
6   T47D.1+   T47D     ER Responsive         1      counts      1654 0.14
7   T47D.2+   T47D     ER Responsive         2      counts      1654 0.08
8   MCF7.1-   MCF7     ER  Resistant         1      counts      1654 0.25
9   MCF7.2-   MCF7     ER  Resistant         2      counts      1654 0.16
10  ZR75.1+   ZR75     ER Responsive         1      counts      1654 0.35
11  ZR75.2+   ZR75     ER Responsive         2      counts      1654 0.24

1 Contrast:
     Group1 Members1     Group2 Members2 DB.edgeR
1 Resistant        4 Responsive        7      235
```

By default, `dba.analyze` plots a correlation heatmap if it finds any significantly differentially bound sites, shown in Figure 2b. Using only the differentially bound sites, we now see that the four tamoxifen resistant samples (representing two cell lines) cluster together, although the tamoxifen-responsive MCF7 replicates cluster closer to them than to the other tamoxifen responsive samples. Comparing Figure 2a, which uses all 1,654 consensus binding sites, with Figure 2b, which uses only the 235 differentially bound sites, demonstrates how the differential binding analysis isolates sites that help distinguish between the Resistant and Responsive sample groups. The fact that an ideal clustering is not apparent even when considering only the differentially bound sites indicates that there may be a confounding factor we are not taking into account in this analysis. See Section 5 for a more sophisticated analysis that takes this factor (the presence of MCF7 cell lines in both the responsive and resistant groups) into account, resulting in a more ideal clustering plot.

3.5 Retrieving the differentially bound sites

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

```
> tamoxifen.DB = dba.report(tamoxifen)
```

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

```
> tamoxifen.DB

GRanges with 235 ranges and 6 metadata columns:
       seqnames               ranges strand   |              Conc
          <Rle>            <IRanges>  <Rle>   |         <numeric>
```

```
1433    chr18 [62640747, 62642453]      *   | 6.93430344179521
   7    chr18 [  384853,   386517]      *   | 6.82514806459552
1606    chr18 [72497301, 72498984]      *   | 7.85563252157062
1496    chr18 [67583814, 67584869]      *   | 5.34228456529008
 156    chr18 [ 7635601,  7636863]      *   | 6.22190948600179
1446    chr18 [63180841, 63181720]      *   | 5.47822580597002
 806    chr18 [32851316, 32852623]      *   |  6.0812842143275
 245    chr18 [10013642, 10014854]      *   | 6.33184309988634
1440    chr18 [62935450, 62937265]      *   |  6.0651895142129
 ...      ...                  ...    ... ...              ...
 428    chr18 [17882950, 17883803]      *   | 5.79435211284434
 254    chr18 [10111717, 10112886]      *   | 7.12606255744744
  58    chr18 [ 2254244,  2255010]      *   | 5.47696852950419
1641    chr18 [74850775, 74851581]      *   | 4.79833863320484
1028    chr18 [44109798, 44112643]      *   | 7.56681993562183
1136    chr18 [50015964, 50016849]      *   | 5.65411683850322
 446    chr18 [18126948, 18127793]      *   | 4.42189452540606
1370    chr18 [58681684, 58682380]      *   | 3.02714855483702
1280    chr18 [54831153, 54832907]      *   | 7.56296441367182
       Conc_Resistant  Conc_Responsive              Fold              p.value
            <numeric>        <numeric>         <numeric>            <numeric>
1433 2.81775662440764 7.55581472986463 -4.73805810545699 2.36935418888045e-07
   7 8.09673253332829 4.44318491699971  3.65354761632857 3.43654787872711e-07
1606 4.25292499017943 8.46386612934426 -4.21094113916483 1.95913444261749e-06
1496 1.00320466121317 5.96820476143297  -4.9650001002198 4.29602532631596e-06
 156 7.39212878690454 4.41322947293372  2.97889931397082 4.43122183935907e-06
1446 2.17610712322116 6.07611036654751 -3.90000324332635  8.2261253253713e-06
 806 2.54502414089941 6.68741802918366 -4.14239388828425 1.18114309933857e-05
 245  7.5402835776779  4.3371851540492   3.2030984236287 1.68628188300138e-05
1440 2.71815696337253 6.66476578949797 -3.94660882612544 1.78330476303149e-05
 ...              ...              ...               ...                  ...
 428 4.05256047036229 6.28036877681275 -2.22780830645046   0.0129384007912487
 254 7.95343989625478 6.28298550703716  1.67045438921762   0.0131060004263551
  58 3.50861324107491 5.98833871766848 -2.47972547659357   0.0132036687844958
1641  3.2092073991694 5.26457625908654 -2.05536885991714   0.0133278094171427
1028 5.62471121953202 8.07547413469216 -2.45076291516013   0.0133499153572264
1136  3.9303597167757 6.13791795961264 -2.20755824283694   0.0135128329973973
 446 5.36293857260643 3.34586686642182  2.01707170618461   0.0136171750121922
1370  1.2818150999124 3.51359669064305 -2.23178159073065   0.0136194754348568
1280 5.42861079794735 8.09030962583931 -2.66169882789196   0.0138206950041009
                     FDR
                <numeric>
```

```
1433 0.000284202509570732
   7 0.000284202509570732
1606  0.00108013612269644
1496  0.00146584818445998
 156  0.00146584818445998
1446  0.00226766854802735
 806  0.00275293128076057
 245  0.00275293128076057
1440  0.00275293128076057
 ...                   ...
 428   0.0942736339591425
 254   0.0950759855490847
  58   0.0953662365482794
1641   0.0955877056313961
1028   0.0955877056313961
1136   0.0962675742275773
 446   0.0962675742275773
1370   0.0962675742275773
1280   0.0972741682416294
---
seqlengths:
 chr18
    NA
```

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

4 Example: plotting

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

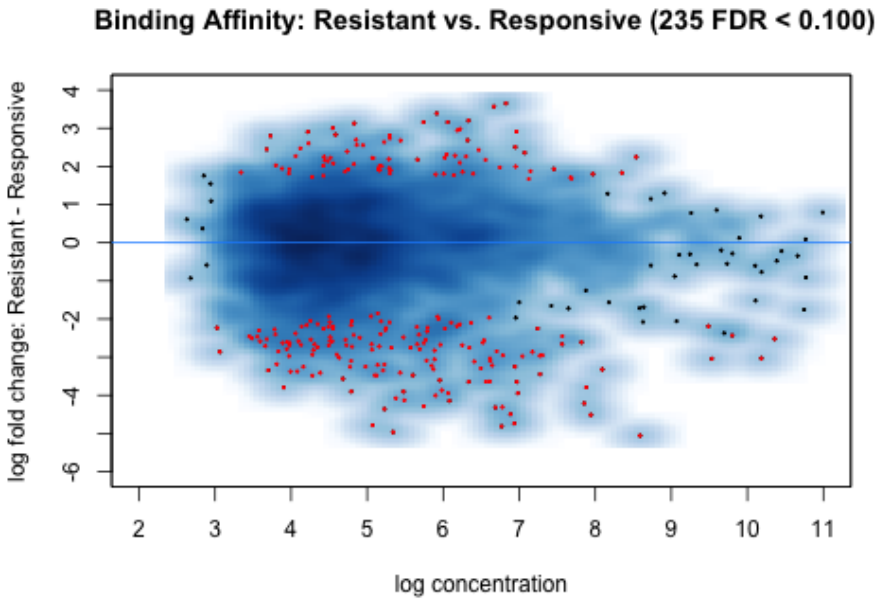


Figure 3: MA plot of Resistant-Responsive contrast, with sites identified as significantly differentially bound shown in red. Generated by: `dba.plotMA(tamoxifen)`

4.1 Venn diagrams

Venn diagrams are useful for examining overlaps between peaksets, particularly when determining how best to derive consensus peaksets for further analysis. Section 6.2, which discusses consensus peaksets, shows a number of Venn plots in context, and the help page for `dba.plotVenn` has a number of additional examples.

4.2 MA plots

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

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

The plot is shown in Figure 3. Each point represents a binding site, with point in red representing sites identified as differentially bound. The plot shows how the differentially bound sites have an absolute log fold difference of at least 2. This same data can also be shown with the concentrations of each sample groups plotted against each other plot using `dba.plotMA(tamoxifen, bXY=T)`.

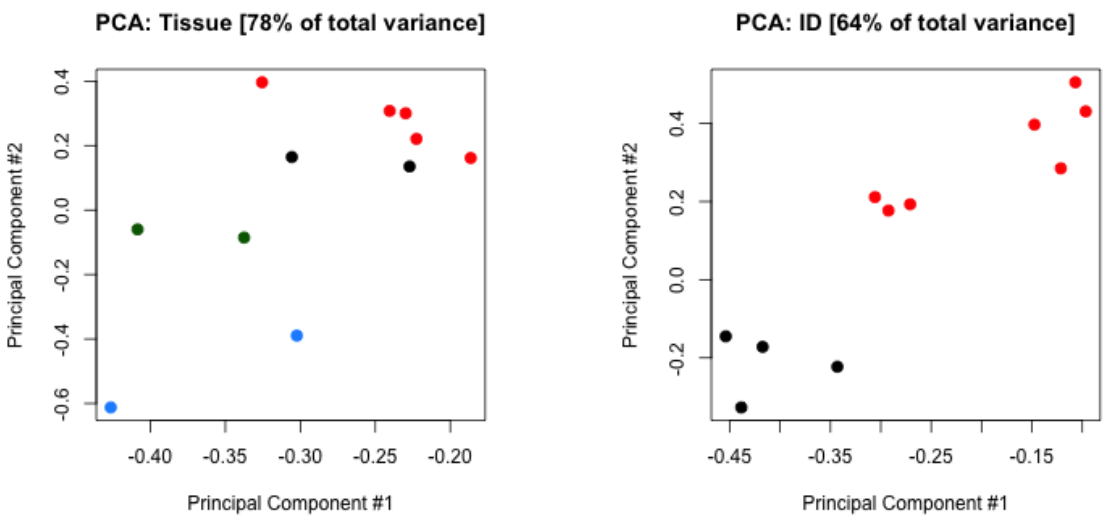
4.3 PCA plots

While the correlation heatmaps already seen are good for showing clustering, plots based on principal components analysis can be used to give a deeper insight into how samples are associated. A PCA plot corresponding to Figure 2a, which includes normalized read counts for all the binding sites, can be obtained as follows:

```
> dba.plotPCA(tamoxifen)

      Legend
BT474 "black"
MCF7  "red"
T47D  "dodgerblue"
ZR75  "darkgreen"
```

This draws the plot and returns a color legend. The resulting plot (Figure 4a) shows the four Resistant samples (black) not separable from the Responsive samples (red) in either the first (horizontal) or the second (vertical) components when looking at all the binding sites.



(a) PCA plot using affinity data for all sites. Generated by: `dba.plotPCA(tamoxifen)`

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

Figure 4: Principal Component Analysis (PCA) plots, generated using `dba.plotPCA`.

A PCA plot using only the differentially bound sites (corresponding to Figure 2b), using an FDR threshold of 0.05, can be drawn as follows:

```
> dba.plotPCA(tamoxifen, contrast=1,th=.05)
```

```
           Legend
Resistant  "black"
Responsive "red"
```

This plot (Figure 4b) shows that the differential analysis identifies sites than can be used to separate the sample groups along both the first and second components.

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

4.4 Boxplots

Boxplots provide a way to view how read distributions differ between classes of binding sites. Consider the example, where the 235 differentially bound sites are identified. The MA plot (Figure 3) 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] 157

> sum(tamoxifen.DB$Fold>0)

[1] 78
```

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

```
> pvals = dba.plotBox(tamoxifen)
```

The default plot (Figure 5) shows in the first two boxes that amongst differentially bound sites overall, the Responsive samples have a somewhat higher mean read concentration. The next two boxes show the distribution of reads in differentially bound sites that exhibit increased affinity in the Responsive samples, while the final two boxes show the distribution of reads in differentially bound sites that exhibit increased affinity in the Resistant samples.

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

```
> pvals
```

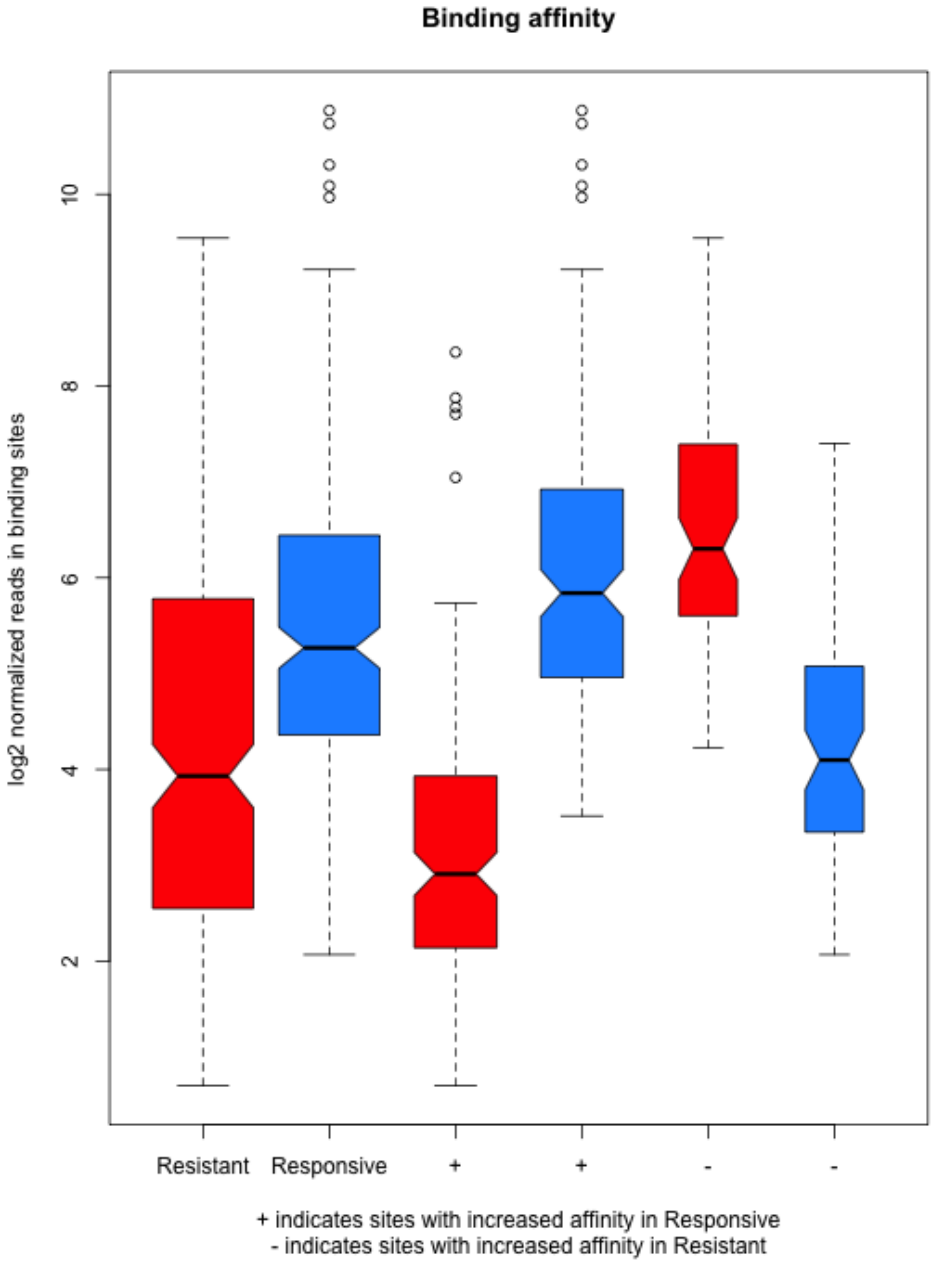


Figure 5: Box plots of read distributions for significantly differentially bound (DB) sites. Tamoxifen resistant samples are shown in red, and responsive samples are shown in blue. Left two boxes show distribution of reads over all DB sites in the Resistant and Responsive groups; middle two boxes show distributions of reads in DB sites that increase in affinity in the Responsive group; last two boxes show distributions of reads in DB sites that increase in affinity in the Resistant group. Generated by: `dba.plotBox(tamoxifen)`

```
               Resistant.DB Responsive.DB Resistant.DB+ Responsive.DB+
Resistant.DB   1.000000e+00  9.746727e-15  2.369406e-07   2.444282e-17
Responsive.DB  9.746727e-15  1.000000e+00  2.548594e-36   1.746585e-04
Resistant.DB+  2.369406e-07  2.548594e-36  1.000000e+00   1.644677e-27
Responsive.DB+ 2.444282e-17  1.746585e-04  1.644677e-27   1.000000e+00
Resistant.DB-  2.358740e-16  9.612111e-09  5.511712e-31   2.084464e-03
Responsive.DB- 4.283649e-01  2.569777e-09  8.436144e-10   4.225062e-17
               Resistant.DB- Responsive.DB-
Resistant.DB    2.358740e-16   4.283649e-01
Responsive.DB   9.612111e-09   2.569777e-09
Resistant.DB+   5.511712e-31   8.436144e-10
Responsive.DB+  2.084464e-03   4.225062e-17
Resistant.DB-   1.000000e+00   1.715161e-14
Responsive.DB-  1.715161e-14   1.000000e+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=9.673496e-15, while those between the Resistant and Responsive groups in the individual cases (increased in Responsive or Resistant) have p-values computed as 1.644677e-27 and 1.715161e-14.

4.5 Heatmaps

DiffBind provides two types of heatmaps. This first, correlation heatmaps, we have already seen. For example, the heatmap shown in Figure 2a 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_TMM_MINUS_FULL`, uses the TMM normalization procedure from `edgeR` , with control reads subtracted first and using the full library size. Another scoring method is to use RPKM fold (RPKM of the ChIP reads divided by RPKM of the control reads; a correlation heatmap for all the data using this scoring method can be obtained by typing `dba.plotHeatmap(tamoxifen, score=DBA_SCORE_RPKM_FOLD)`.

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

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

Figure 6 shows the affinities and clustering of the differentially bound sites (rows), as well as the sample clustering (columns). This plot can be tweaked to get more contrast, for example by using row-scaling `dba.plotHeatmap(tamoxifen, contrast=1, correlations=FALSE, scale=row)`.

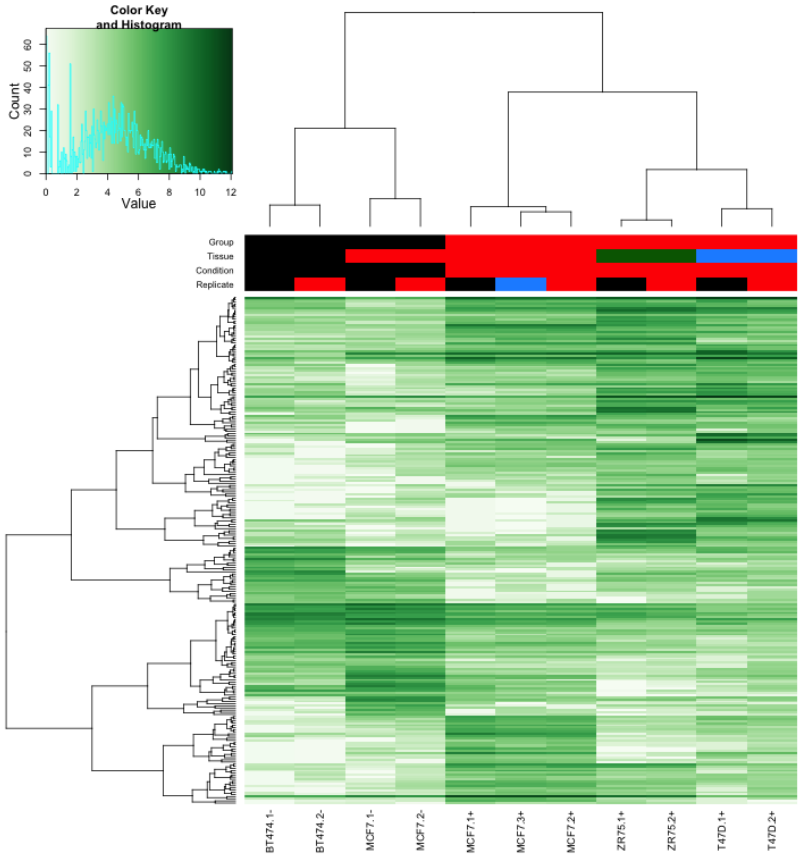


Figure 6: Binding affinity heatmap showing affinities for differentially bound sites. Samples cluster first by whether they are responsive to tamoxifen treatment, then by cell line. Clusters of binding sites show distinct patterns of affinity levels. Generated by: `dba.plotHeatmap(tamoxifen, contrast=1, correlations=FALSE)`

5 Example: differential binding analysis using a blocking factor

The previous example showed how to perform a differential binding analysis using a single factor with two values; that is, finding the significantly differentially bound sites between two sets of samples. This section extends the example by including a second factor, potentially with multiple values, that represents a confounding condition. Examples of experiments where it is appropriate to use a blocking factor include one where there are potential batch effects, with samples from the two conditions prepared together, or a matched design (e.g. matched normal and tumor pairs, where the primary factor of interest is to discover sites consistently differentially bound between normal and tumor samples. In the current example, the confounding effect we want to control for is the presence of two sets of samples, one tamoxifen responsive and one resistant, that are both derived from the same MCF7 cell line.

In the previous analysis, the two MCF7-derived cell lines tended to cluster together. While the differential binding analysis was able to identify sites that could be used to separate the resistance 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 2a).

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

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

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

```
> tamoxifen = dba.analyze(tamoxifen)
> tamoxifen

11 Samples, 1654 sites in matrix:
         ID Tissue Factor  Condition Replicate Peak.caller Intervals   SN
1  BT474.1-  BT474     ER  Resistant         1      counts      1654 0.19
2  BT474.2-  BT474     ER  Resistant         2      counts      1654 0.18
3   MCF7.1+   MCF7     ER Responsive         1      counts      1654 0.35
4   MCF7.2+   MCF7     ER Responsive         2      counts      1654 0.22
5   MCF7.3+   MCF7     ER Responsive         3      counts      1654 0.28
```

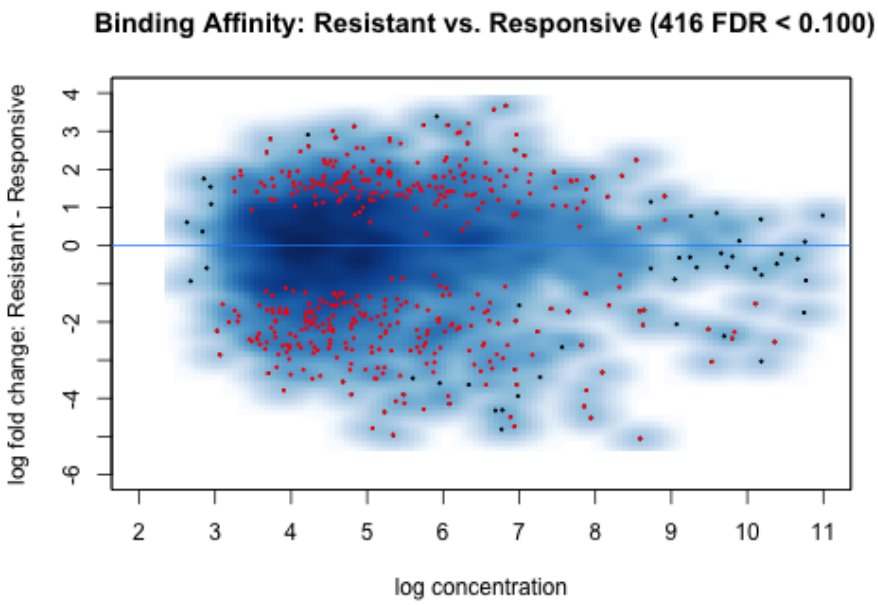


Figure 7: MA plot of Resistant-Responsive contrast, using MCF7 origin as a blocking factor, with sites identified as significantly differentially bound shown in red. Generated by: `dba.plotMA(tamoxifen, method=DBA_EDGER_BLOCK)`

```
6   T47D.1+   T47D     ER Responsive         1      counts      1654 0.14
7   T47D.2+   T47D     ER Responsive         2      counts      1654 0.08
8   MCF7.1-   MCF7     ER  Resistant         1      counts      1654 0.25
9   MCF7.2-   MCF7     ER  Resistant         2      counts      1654 0.16
10  ZR75.1+   ZR75     ER Responsive         1      counts      1654 0.35
11  ZR75.2+   ZR75     ER Responsive         2      counts      1654 0.24

1 Contrast:
     Group1 Members1     Group2 Members2 Block1Val InBlock1 Block2Val InBlock2
1 Resistant        4 Responsive        7      true        5     false        6
  DB.edgeR DB.edgeR.block
1      235            416
```

This indicates that where the standard, single-factor `edgeR` analysis identifies 235 differentially bound sites, the analysis using the blocking factor finds 416 such sites. An MA plot shows how the analysis has changed:

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

The resulting plot is shown in Figure 7. Comparing this to Figure 3, at least two differences can be observed. The analysis has become more sensitive, with sites being identified as significantly

differentially bound with lower magnitude fold changes (as low as twofold, as this plot is on a log2 scale). But it is not merely lowering a fold threshold: many sites with higher fold changes are no longer found to be significant. These were included in the earlier analysis because the confounding factor was not being modelled.

Consider the resulting separation and clustering using the newly discovered differentially bound sites:

```
> dba.plotHeatmap(tamoxifen,contrast=1,method=DBA_EDGER_BLOCK,
+                 attributes=c(DBA_TISSUE,DBA_CONDITION,DBA_REPLICATE))

> dba.plotPCA(tamoxifen,contrast=1,method=DBA_EDGER_BLOCK,
+             attributes=c(DBA_TISSUE,DBA_CONDITION))

                Legend
BT474:Resistant "black"
MCF7:Resistant  "red"
MCF7:Responsive "dodgerblue"
T47D:Responsive "darkgreen"
ZR75:Responsive "cyan"
```

Frequently, as more sites are included in these plots, the result is often worse clustering/separation along the grouping of primary interest. With this analysis, however, clustering and separation are improved, even though many more sites have been identified. The correlation heatmap (Figure 8, compare to Figure 2b) shows the tamoxifen resistant cell lines clustering together, with the tamoxifen responsive MCF7 sample clustering with the other responsive cell lines. Likewise, the PCA plot (Figure 9, compare to Figure 4b) shows a cleaner separation of the MCF7 samples, particularly in the second component.

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

```
> tamoxifen = dba.analyze(tamoxifen,method=DBA_DESEQ)

.
.
.
.

> tamoxifen

11 Samples, 1654 sites in matrix:
         ID Tissue Factor  Condition Replicate Peak.caller Intervals   SN
1  BT474.1-  BT474     ER  Resistant         1      counts      1654 0.19
2  BT474.2-  BT474     ER  Resistant         2      counts      1654 0.18
3   MCF7.1+   MCF7     ER Responsive         1      counts      1654 0.35
4   MCF7.2+   MCF7     ER Responsive         2      counts      1654 0.22
```

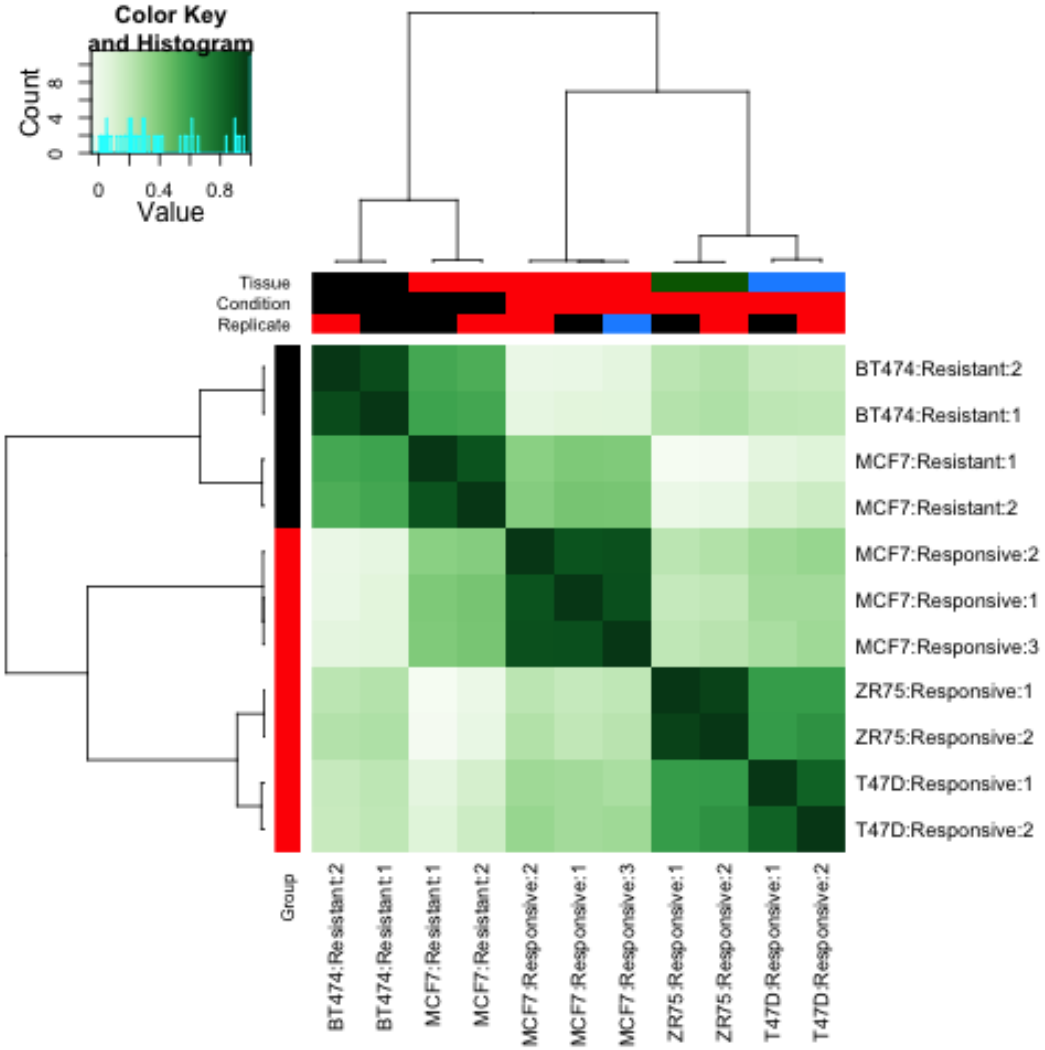


Figure 8: Correlation heatmap of using scores for significantly differentially bound sites for the Resistant-Responsive contrast, using MCF7 origin as a blocking factor. Generated by: `dba.plotHeatmap(tamoxifen, contrast=1, method=DBA_EDGER_BLOCK, attributes=c(DBA_TISSUE,DBA_CONDITION,DBA_REPLICATE))`

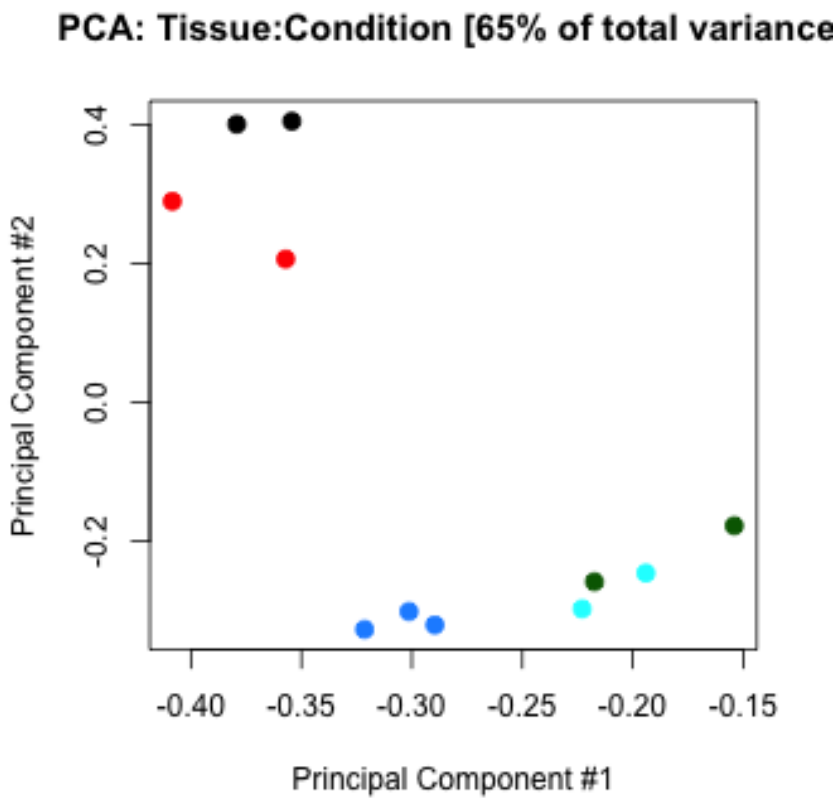


Figure 9: Plot of first two principal components, using scores for significantly differentially bound sites for the Resistant-Responsive contrast, using MCF7 origin as a blocking factor. Generated by: `dba.plotPCA(tamoxifen,contrast=1,method=DBA_EDGER_BLOCK, attributes=c(DBA_TISSUE,DBA_CONDITION))`

```
5   MCF7.3+   MCF7     ER Responsive         3      counts      1654 0.28
6   T47D.1+   T47D     ER Responsive         1      counts      1654 0.14
7   T47D.2+   T47D     ER Responsive         2      counts      1654 0.08
8   MCF7.1-   MCF7     ER  Resistant         1      counts      1654 0.25
9   MCF7.2-   MCF7     ER  Resistant         2      counts      1654 0.16
10  ZR75.1+   ZR75     ER Responsive         1      counts      1654 0.35
11  ZR75.2+   ZR75     ER Responsive         2      counts      1654 0.24

1 Contrast:
     Group1 Members1     Group2 Members2 Block1Val InBlock1 Block2Val InBlock2
1 Resistant        4 Responsive        7      true        5     false        6
  DB.edgeR DB.edgeR.block DB.DESeq DB.DESeq.block
1      235            416       54            550
```

While `DESeq` has a much more conservative approach to the single factor analysis, identifying only 54 sites as differentially bound, when modelling the confounding factor, the greater sensitivity results in many more sites being identified. You can check this by looking at the identified sites using `dba.report`, and performing MA, heatmap, and PCA plots.

6 Example: occupancy analysis and overlaps

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

```
> data(tamoxifen_peaks)
```

6.1 Overlap rates

One reason to do an occupancy-based analysis is to determine what candidate sites should be used in a subsequent affinity-based analysis. In the example so far, we took all sites that were identified in peaks in at least three of the eleven peaksets, reducing the number of sites from 3,557 overall to the 1,654 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 3,557 sites. In making the decision of what criteria to use many factors come into play, but it helps to get an idea of the rates at which the peaksets overlap (for more details on how overlaps are determined, see Section 7.2 on peak merging). A global overview can be obtained using the RATE mode of the `dba.overlap` function as follows:

```
> olap.rate = dba.overlap(tamoxifen,mode=DBA_OLAP_RATE)
> olap.rate

 [1] 3557 2602 1654 1299 1002  764  620  455  352  187  118
```

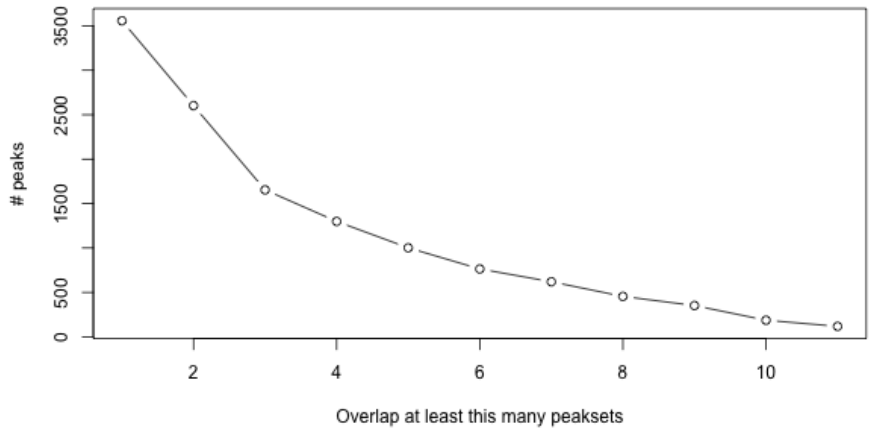


Figure 10: Overlap rate plot, showing how the number of overlapping peaks decreases as the overlap criteria becomes more stringent. X axis shows the number of peaksets in which the site is identified, while the Y axis shows the number of overlapping sites. Generated by plotting the result of: `dba.overlap(tamoxifen,mode=DBA_OLAP_RATE)`

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

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

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

The rate plot is shows in Figure 10. These curves typically exhibit a roughly geometric drop-off, with the number of overlapping sites halving as the overlap criterion become stricter by one site. When the drop-off is extremely steep, this is an indication that the peaksets do not agree very well. For example, if there are replicates you expect to agree, there may be a problem with the experiment. In the current example, peak agreement is high and the curve exhibits a better than geometric drop-off.

6.2 Deriving consensus peaksets

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

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

less powerful way to help accomplish this is to use multiple peak callers for each ChIP dataset and look for agreement between peak callers (Li et al. [2011]).

Consider for example the standard (tamoxifen responsive) MCF7 cell line, represented by three replicates in this dataset. How well do the replicates agree on their peak calls? The overlap rate for just the 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"  ""            "raw"         "Replicate.1"
[11] "Replicate.2" "Replicate.3" "All"         "None"
```

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

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

[1] 1767 1222  874
```

There are 874 peaks (out of 1,767) identified in all three replicates. A finer grained view of the overlaps can be obtained with the `dba.plotVenn` function:

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

The resultant plot is shown as Figure 11. This plot shows the 874 consensus peaks identified as common to all replicates, but further breaks down how the replicates relate to each other. The same can be done for each of the replicated cell line experiments, and rather than applying a global cutoff (3 of 11), each cell line could be dealt with individually in deriving a final peakset. A separate consensus peakset for each of the replicated sample types can be added to the DBA object using `dba.peakset`:

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

16 Samples, 2602 sites in matrix (3557 total):
                ID Tissue Factor  Condition Replicate Peak.caller Intervals
1         BT474.1-  BT474     ER  Resistant         1         raw      1084
2         BT474.2-  BT474     ER  Resistant         2         raw      1115
3          MCF7.1+   MCF7     ER Responsive         1         raw      1513
```

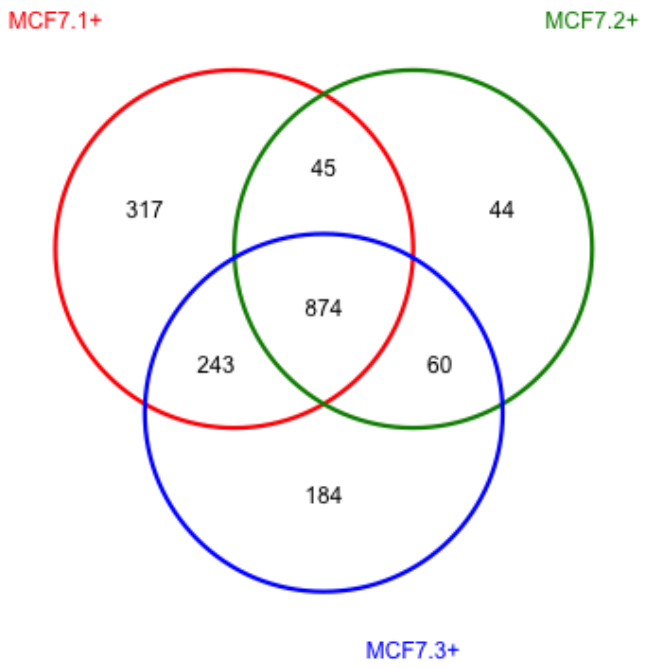


Figure 11: Venn diagram showing how the ER peak calls for three replicates of responsive MCF7 cell line overlap. Generated by plotting the result of: `dba.venn(tamoxifen,tamoxifen$masks$MCF7 & tamoxifen$masks$Responsive)`

```
4          MCF7.2+   MCF7     ER Responsive         2         raw      1037
5          MCF7.3+   MCF7     ER Responsive         3         raw      1372
6          T47D.1+   T47D     ER Responsive         1         raw       509
7          T47D.2+   T47D     ER Responsive         2         raw       347
8          MCF7.1-   MCF7     ER  Resistant         1         raw      1148
9          MCF7.2-   MCF7     ER  Resistant         2         raw       933
10         ZR75.1+   ZR75     ER Responsive         1         raw      2111
11         ZR75.2+   ZR75     ER Responsive         2         raw      1975
12 BT474:Resistant  BT474     ER  Resistant       1-2         raw       902
13 MCF7:Responsive   MCF7     ER Responsive     1-2-3         raw      1222
14 T47D:Responsive   T47D     ER Responsive       1-2         raw       298
15  MCF7:Resistant   MCF7     ER  Resistant       1-2         raw       795
16 ZR75:Responsive   ZR75     ER Responsive       1-2         raw      1633
```

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

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

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

```
5 Samples, 1163 sites in matrix (2444 total):
               ID Tissue Factor  Condition Replicate Peak.caller Intervals
1 BT474:Resistant  BT474     ER  Resistant       1-2         raw       902
2 MCF7:Responsive   MCF7     ER Responsive     1-2-3         raw      1222
3 T47D:Responsive   T47D     ER Responsive       1-2         raw       298
4  MCF7:Resistant   MCF7     ER  Resistant       1-2         raw       795
5 ZR75:Responsive   ZR75     ER Responsive       1-2         raw      1633
```

Alternatively, a master consensus peakset could be generated, and reads counted, directly using `dba.count`:

```
> tamoxifen = dba.count(tamoxifen, peaks=tamoxifen$masks$Consensus)
```

Finally, consider an analysis where we wished to treat all five MCF7 samples together to look for binding sites specific to that cell line irrespective of tamoxifien resistant/responsive status. We can create consensus peaksets for each cell type, and look at how the resultant peaks overlap (Figure 12):

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

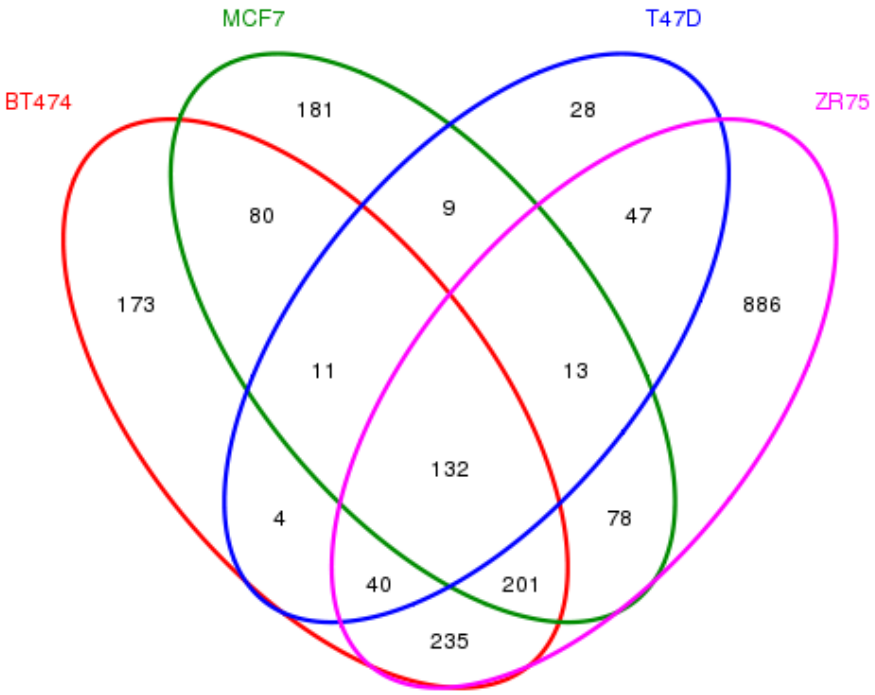


Figure 12: Venn diagram showing how the consensus peaks for each cell type overlap. Generated by plotting the result of: `dba.venn(tamoxifen,tamoxifen$masks$Consensus)`

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

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

Returning to the original tamoxifen dataset:

```
> data(tamoxifen_peaks)
```

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

```
> dba.overlap(tamoxifen,tamoxifen$masks$Resistant,mode=DBA_OLAP_RATE)

[1] 1875 1298  597  436

> dba.overlap(tamoxifen,tamoxifen$masks$Responsive,mode=DBA_OLAP_RATE)

[1] 3208 2293 1217  807  584  265  161
```

Requiring that consensus peaks overlap in at least one third of the samples in each group results in 1,298 sites for the Resistant group and 1,217 sites for the Responsive group:

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

Figure 13 shows that 448 sites are unique to the Resistant group, and 392 sites are unique to the Responsive group, with 819 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 819 common sites to be uninteresting, and focus on the 840 sites that are unique to a specific group. These unique sites can be obtained using `dba.overlap`:

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

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

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

Figure 13: 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)`

```
GRanges with 448 ranges and 1 metadata column:
       seqnames               ranges strand   |              score
          <Rle>            <IRanges>  <Rle>   |          <numeric>
     4    chr18     [301531, 302172]      *   | 0.0630939951318686
     6    chr18     [346557, 347362]      *   | 0.0202085601527258
     7    chr18     [361121, 362106]      *   | 0.0189999480119068
     8    chr18     [384853, 386517]      *   |  0.121797447215017
    11    chr18     [479200, 480032]      *   | 0.0834567115452259
    16    chr18     [562348, 563067]      *   | 0.0231533994443167
    18    chr18     [639178, 639957]      *   | 0.0474735365693182
    20    chr18     [647089, 647869]      *   | 0.0357705387285349
    25    chr18     [914869, 915486]      *   | 0.0499399950374473
   ...      ...                  ...    ... ...                ...
  1631    chr18 [74267807, 74268931]      *   | 0.0990811667533883
  1632    chr18 [74313069, 74313720]      *   | 0.0280953916852647
  1639    chr18 [74629529, 74630608]      *   | 0.0177467020159173
  1642    chr18 [74675317, 74676231]      *   | 0.0547749262961461
  1649    chr18 [75157775, 75158504]      *   | 0.0289404938256132
  1650    chr18 [75163026, 75163816]      *   | 0.0207221417463246
  1652    chr18 [75401417, 75402162]      *   | 0.0321108676071455
  1653    chr18 [75525519, 75526188]      *   | 0.0367471112551585
  1657    chr18 [75826088, 75826939]      *   | 0.0225979404192851
  ---
  seqlengths:
```

```
   chr18
      NA

> tamoxifen.OL$onlyB

GRanges with 392 ranges and 1 metadata column:
       seqnames               ranges strand   |              score
          <Rle>            <IRanges>  <Rle>   |          <numeric>
     5    chr18   [ 336592,  337347]      *   | 0.0412966688004544
    10    chr18   [ 439109,  440079]      *   | 0.0248480179180791
    27    chr18   [ 988767,  989698]      *   | 0.0378677595515141
    28    chr18   [1065304, 1066051]      *   | 0.0510851900868453
    33    chr18   [1231653, 1232311]      *   | 0.0365133439877049
    39    chr18   [1369773, 1370746]      *   | 0.0650460071208119
    42    chr18   [1470236, 1470902]      *   | 0.0296537980454761
    53    chr18   [2161751, 2162455]      *   | 0.0237545085977324
    57    chr18   [2197080, 2197908]      *   |  0.035595764948454
   ...      ...                  ...    ... ...                 ...
  1614    chr18 [72790966, 72791819]      *   | 0.0617312935186414
  1618    chr18 [72914352, 72914983]      *   | 0.0242428333633483
  1628    chr18 [73517930, 73518921]      *   | 0.0202448900241384
  1644    chr18 [74835590, 74836200]      *   | 0.0796813780786506
  1645    chr18 [74850775, 74851581]      *   | 0.0423465698740762
  1646    chr18 [74860617, 74861878]      *   | 0.0305023830489946
  1647    chr18 [74906349, 74907306]      *   | 0.0217141712183927
  1655    chr18 [75641971, 75642647]      *   | 0.0460080595885991
  1659    chr18 [76087923, 76089259]      *   |  0.104847503448091
  ---
  seqlengths:
   chr18
      NA
```

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

6.4 Comparison of occupancy and affinity based analyses

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

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

```
> tamoxifen = dba.peakset(tamoxifen,tamoxifen$masks$Consensus,
+             minOverlap=1,sampID="OL Consensus")
```

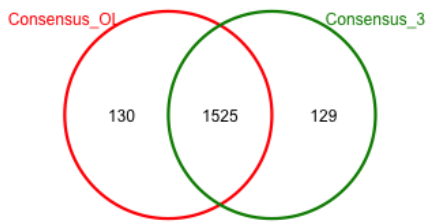


Figure 14: Venn diagram showing how the ER peak calls for two different ways of deriving consensus peaksets. Generated by plotting the result of: `dba.plotVenn(tamoxifen,14:15)`

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

Figure 14 shows that the two sets agree on about 85% of their sites, so the results should be directly comparable between the differing parameters used to establish the consensus peaksets.[4]

Next re-load the affinity analysis:

```
> data(tamoxifen_analysis)
```

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

```
> tamoxifen.rep = dba.report(tamoxifen,bCalled=T,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 )
```

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

```
[1] 313

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

[1] 391

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

[1] 821
```

Comparing these numbers verifies the similarity with those seen in Figure 13, showing again how the basic analysis is not oversensitive to differences in how the consensus peaksets are formed. This overlap analysis suggests that 704 of the sites are uniquely bound in either the Responsive or Resistant groups, while 821 sites are common to both.

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

```
> tamoxifen.DB = dba.report(tamoxifen,bCalled=T,th=.1)
> onlyResistant = tamoxifen.DB$Called1>=2 & tamoxifen.DB$Called2<3
> sum(onlyResistant)

[1] 41

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

[1] 119

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

[1] 72
```

There are a number of notable differences in the results. First there are many fewer sites identified as differentially bound (232) than are unique to one condition (704). Indeed, most of the sites identified in the occupancy analysis as unique to a sample group are not found to be significantly differentially bound using the affinity data. While partly this is a result of the stringency of the statistical tests, it shows how the affinity analysis can discriminate between sites where peak callers are making occupancy decisions that do not reflect significant differences in read densities at these sites. Note that about 16% of unique sites are significantly differential bound (113 out of 704). Secondly, differentially bound sites are as likely to be called in the consensus of

both response groups as they are to be unique to one group, as *nearly half of the sites identified as significantly differentially bound are called as peaks in both response groups*. Indeed, sites identified in both sample groups are almost as likely to be identified as significantly differentially bound as sites identified in only one sample group (14.5%, or 119 out of 821). Finally, the differentially bound peaks identified using the affinity analysis are associated with significance statistics (p-value and FDR) that can be used to rank them for further examination, while the occupancy analysis yields a relatively unordered list of peaks, as the peak caller statistics refer only to the significance of occupancy, and not of differential binding.

7 Technical notes

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

7.1 Loading peaksets

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

When loading in peaksets from files, specifying what peak caller generated the file enables peaks from supported peak callers to be read in. See the help page for `dba.peakset` for a list of supported peak callers. Any string 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 simple text files, no the expected MACS "xls" format. To maintain the peak caller in the metadata, we could specify the `PeakCaller` as "macs" but the `PeakFormat` as "raw". If we wanted to use peak scores in a different column than the fourth, the `scorecol` parameter could be set to indicate the appropriate column number. When handling scoring, DiffBind by default assumes that a higher score indicates a "better" peak. If this is not the case, for example if the score is a p-value or FDR, we could set `bLowerScoreBetter` to `TRUE`.

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

7.2 Merging peaks

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

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

7.3 `edgeR` analysis

When `dba.analyze` is invoked using the default `method=DBA_EDGER`, a standardized differential analysis is performed using the `edgeR` package (Robinson et al. [2010]). 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 peaks (the sum of each column). Alternatively, if the `bFullLibrarySize` parameter is set to TRUE, the total number of reads in the library (calculated from the source BAM//BED file) is used. The default setting is appropriate for situations when the overall signal is expected to be directly comparable between the samples; using the full library size may be preferable if samples are expected to have dramatically different signals (e.g., if some are expected to have very low binding rates compared to others). Next comes a call to `edgeR` 's `DGEList` function. The `DGEList` object that results is next passed to `calcNormFactors` with all other parameters retained as defaults (`method="TMM"`), returning an updated `DGEList` object. This is passed to `estimateCommonDisp` with default parameters.

If the method is `DBA_EDGER_CLASSIC`, then if `bTagwise` is TRUE (most useful when there are at least three members in each group of a contrast), the resulting `DGEList` object is then passed to `estimateTagwiseDisp`, with the prior set to 50 divided by two less than the total number of samples in the contrast, and `trend="none"`. The final steps are to perform testing to determine the significance measure of the differences between the sample groups by calling `exactTest` (Robinson and Smyth [2007]) 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 (McCarthy et al. [2012]) is performed, using either common or tagwise dispersion depending on the value specified for bTagwise.

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

DBA$contrasts[[n]]$edgeR

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

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

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

7.4 DESeq analysis

When dba.analyze is invoked using method=DBA_DESEQ[5], a standardized differential analysis is performed using the DESeq package (Anders and Huber [2010]). This section details the precise steps in that analysis.

For each contrast, a separate analysis is performed. First, a matrix of counts is constructed for the contrast, with columns for all the samples in the first group, followed by columns for all the samples in the second group. The raw read count is used for this matrix; if the bSubControl parameter is set to TRUE (as it is by default), the raw number of reads in the control sample (if available) will be subtracted. Next the library size is computed for each sample for use in subsequent normalization. By default, this is the total number of reads in peaks (the sum of each column). Alternatively, if the bFullLibrarySize parameter is set to TRUE, the total number of reads in the library (calculated from the source BAM/BED file) is used. The first step concludes with a call to DESeq's newCountDataSet function, which returns a CountDataSet object. If bFullLibrarySize is set to TRUE, then sizeFactors is called with the number of reads in the BAM/BED files for each ChIP sample, divided by the minimum of these; otherwise, estimateSizeFactors is invoked. Next, estimateDispersions is called with the CountDataSet object and fitType set to local. If there are no replicates, (only one sample in each group), method is set to blind. Otherwise, if bTagwise is TRUE, method is set to per-condition; if it is FALSE, method is set to pooled (or pooled-CR for a blocking analysis).

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

If the method is DBA_DESEQ_GLM (the default), two models are fitted using fitNbinomGLMs: a full model is fitted with all the coefficients, and a second model is fitted with the second coefficient

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

dropped. These are tested against each other using `nbinomGLMTest`, with the resulting p values adjusted using `p.adjust` (with `method="BH"`).

The final results are acessible within the `DBA object` as

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

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

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

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

8 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.

9 Setup

This vignette was built on:

```
> sessionInfo()

R version 2.15.2 (2012-10-26)
Platform: x86_64-unknown-linux-gnu (64-bit)

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
 [3] LC_TIME=en_US.UTF-8        LC_COLLATE=C
 [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=C                 LC_NAME=C
 [9] LC_ADDRESS=C               LC_TELEPHONE=C
[11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
```

```
attached base packages:
[1] parallel  stats     graphics  grDevices utils     datasets  methods
[8] base

other attached packages:
[1] DESeq_1.10.1         lattice_0.20-13      locfit_1.5-8
[4] DiffBind_1.4.2       Biobase_2.18.0       GenomicRanges_1.10.6
[7] IRanges_1.16.5       BiocGenerics_0.4.0

loaded via a namespace (and not attached):
 [1] AnnotationDbi_1.20.3 DBI_0.2-5            KernSmooth_2.23-8
 [4] MASS_7.3-23          RColorBrewer_1.0-5   RSQLite_0.11.2
 [7] XML_3.95-0.1         amap_0.8-7           annotate_1.36.0
[10] edgeR_3.0.8          gdata_2.12.0         genefilter_1.40.0
[13] geneplotter_1.36.0   gplots_2.11.0        grid_2.15.2
[16] gtools_2.7.0         limma_3.14.4         splines_2.15.2
[19] stats4_2.15.2        survival_2.37-2      tools_2.15.2
[22] xtable_1.7-0         zlibbioc_1.4.0
```

References

Simon Anders and Wolfgang Huber. Differential expression analysis for sequence count data. *Genome Biology*, 11(10):R106, Oct 2010. doi: 10.1186/gb-2010-11-10-r106.

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

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.

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. doi: 10.1093/bioinformatics/btp616. URL `http://bioinformatics.oxfordjournals.org/cgi/content/full/26/1/139`.

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

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.

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.