

DiffBind : differential binding analysis of ChIP-Seq peak data

Rory Stark

`rory.stark@cancer.org.uk`

Gordon Brown

`gordon.brown@cancer.org.uk`

Cancer Research UK
Cambridge Research Institute

20 March 2012

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/SAM/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:

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

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

```
> tamoxifen
```

```
11 Samples, 2602 sites in matrix (3557 total):
```

	ID	Tissue	Factor	Condition	Replicate	Intervals
1	BT474.1-	BT474	ER	Resistant	1	1084
2	BT474.2-	BT474	ER	Resistant	2	1115
3	MCF7.1+	MCF7	ER	Responsive	1	1513
4	MCF7.2+	MCF7	ER	Responsive	2	1037
5	MCF7.3+	MCF7	ER	Responsive	3	1372
6	T47D.1+	T47D	ER	Responsive	1	509
7	T47D.2+	T47D	ER	Responsive	2	347
8	MCF7.1-	MCF7	ER	Resistant	1	1148
9	MCF7.2-	MCF7	ER	Resistant	2	933
10	ZR75.1+	ZR75	ER	Responsive	1	2111
11	ZR75.2+	ZR75	ER	Responsive	2	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.

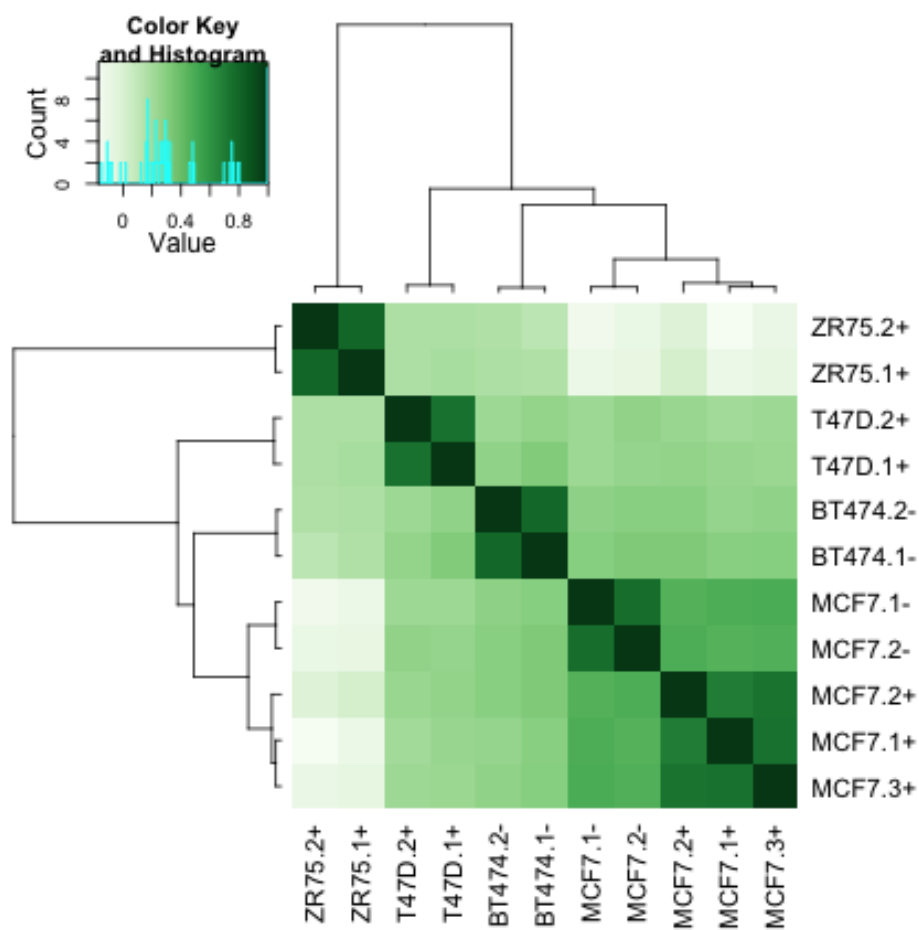


Figure 1: Correlation heatmap, using occupancy (peak caller score) data. Generated by: `plot(tamoxifen)`; can also be generated by: `dba.plotHeatmap(tamoxifen)`.

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 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:

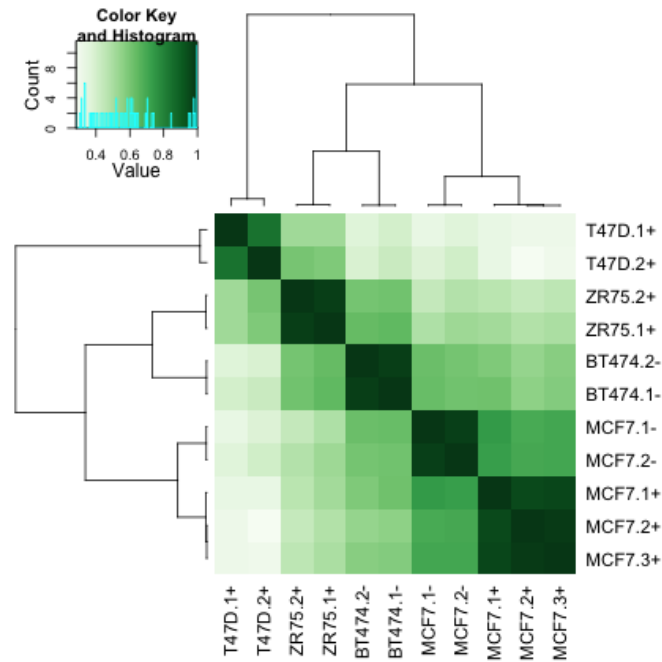
```
> tamoxifen = dba.analyze(tamoxifen)
```

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 $FDR \leq 0.1$:

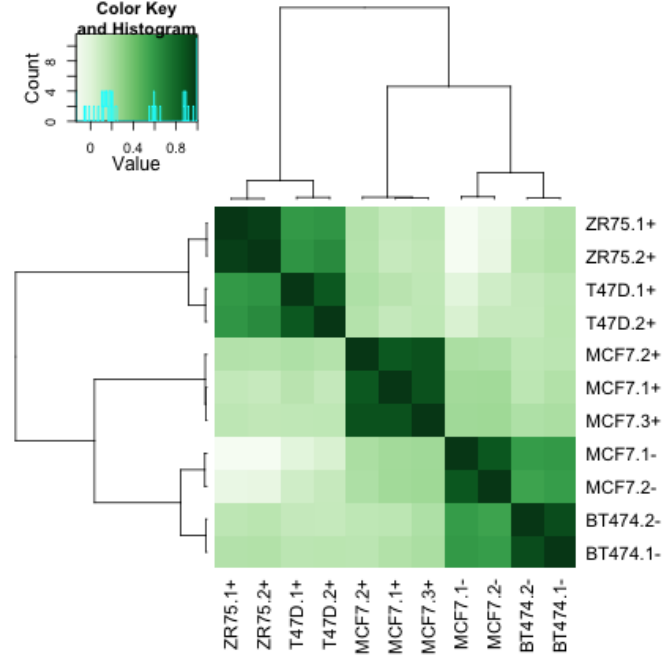
```
> tamoxifen
```

¹Note 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)`

²This 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	Intervals
1	BT474.1-	BT474	ER	Resistant	1	1654
2	BT474.2-	BT474	ER	Resistant	2	1654
3	MCF7.1+	MCF7	ER	Responsive	1	1654
4	MCF7.2+	MCF7	ER	Responsive	2	1654
5	MCF7.3+	MCF7	ER	Responsive	3	1654
6	T47D.1+	T47D	ER	Responsive	1	1654
7	T47D.2+	T47D	ER	Responsive	2	1654
8	MCF7.1-	MCF7	ER	Resistant	1	1654
9	MCF7.2-	MCF7	ER	Resistant	2	1654
10	ZR75.1+	ZR75	ER	Responsive	1	1654
11	ZR75.2+	ZR75	ER	Responsive	2	1654

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.

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 elementMetadata cols:

seqnames	ranges	strand		Conc
<Rle>	<IRanges>	<Rle>		<numeric>

1433	chr18	[62640747, 62642453]	*		6.93446985062311
7	chr18	[384853, 386517]	*		6.82360489448543
1606	chr18	[72497301, 72498984]	*		7.85584497172355
1496	chr18	[67583814, 67584869]	*		5.34220917917666
156	chr18	[7635601, 7636863]	*		6.22184608056736
1446	chr18	[63180841, 63181720]	*		5.478248039011
806	chr18	[32851316, 32852623]	*		6.08124222246203
245	chr18	[10013642, 10014854]	*		6.33180178601248
1440	chr18	[62935450, 62937265]	*		6.06536039712491
...
428	chr18	[17882950, 17883803]	*		5.79464564596883
254	chr18	[10111717, 10112886]	*		7.12607764157981
58	chr18	[2254244, 2255010]	*		5.47718414424491
1641	chr18	[74850775, 74851581]	*		4.79827285007796
1028	chr18	[44109798, 44112643]	*		7.56671182432961
1136	chr18	[50015964, 50016849]	*		5.65424269289693
1370	chr18	[58681684, 58682380]	*		3.0273434181576
446	chr18	[18126948, 18127793]	*		4.42187310620638
1280	chr18	[54831153, 54832907]	*		7.56285633435402

	Conc_Resistant	Conc_Responsive		Fold	p.value
	<numeric>	<numeric>		<numeric>	<numeric>
1433	2.81763656456302	7.55598727203811	-4.73835070747509	2.37229151228923e-07	
7	8.09661247348367	4.43136660594618	3.66524586753749	3.11744663551109e-07	
1606	4.25280493033481	8.46408883827149	-4.21128390793668	1.96106558171827e-06	
1496	1.00308460136855	5.96813019264189	-4.96504559127334	4.29769211377866e-06	
156	7.39200872705993	4.41342127109062	2.97858745596931	4.45120764507587e-06	
1446	2.17598706337654	6.07613804593089	-3.90015098255435	8.23082715536047e-06	
806	2.5449040810548	6.68737856334493	-4.14247448229014	1.18162889298005e-05	
245	7.54016351783328	4.33755816864945	3.20260534918383	1.69491776518319e-05	
1440	2.71803690352792	6.66494745384203	-3.94691055031411	1.78419422470735e-05	
...	
428	4.05244041051767	6.28071275609926	-2.22827234558159	0.0129313717851089	
254	7.95331983641017	6.28324637776154	1.67007345864862	0.0131376168059196	
58	3.50849318123029	5.98858871613736	-2.48009553490707	0.0131983073471148	
1641	3.20908733932479	5.26451793767293	-2.05543059834815	0.0133278887347908	
1028	5.62459115968741	8.07536727228396	-2.45077611259655	0.013351288346433	
1136	3.93023965693108	6.13807423425685	-2.20783457732577	0.0135093821863586	
1370	1.28169504006778	3.51382986114437	-2.23213482107659	0.0136148637101148	
446	5.36281851276181	3.3460735695309	2.01674494323091	0.0136381678825304	
1280	5.42849073810273	8.09020262841551	-2.66171189031277	0.0138221035775294	
	FDR				
	<numeric>				

```

1433 0.000257812836756767
    7 0.000257812836756767
1606 0.00108120082405401
1496 0.0014724594889911
    156 0.0014724594889911
1446 0.0022689646858277
    806 0.00275784220184094
    245 0.00275784220184094
1440 0.00275784220184094
...
428 0.0942224182051544
254 0.0953053429692591
    58 0.0953275124547066
1641 0.0955975364718621
1028 0.0955975364718621
1136 0.0963125781734357
1370 0.0963996994773729
    446 0.0963996994773729
1280 0.0972840822009938
---
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 MA plots, PCA plots, Boxplots, and Heatmaps.

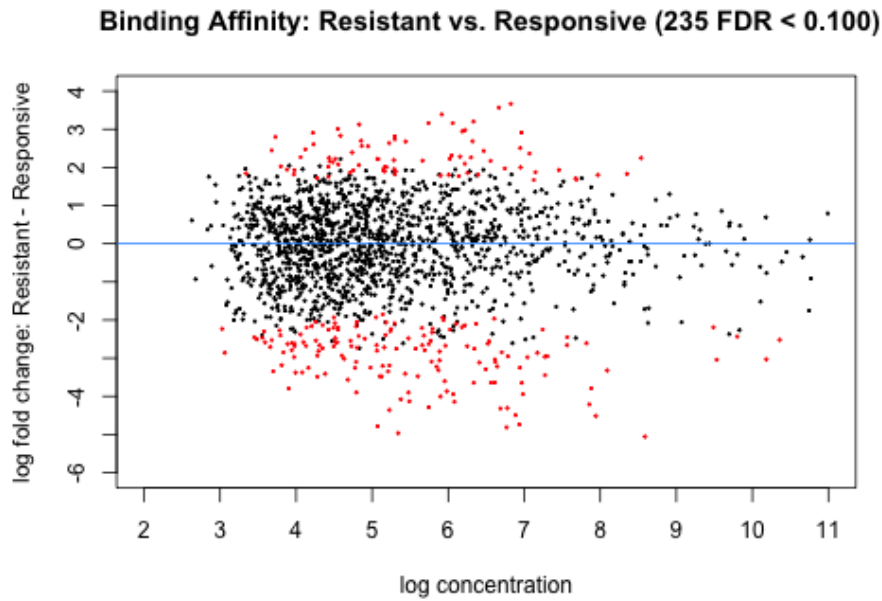


Figure 3: MA plot of Resistant-Responsive contrast, with sites identified as significantly differentially bound shown in red. Generated by: `dba.plotMA(tamoxifen)`

4.1 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)`.

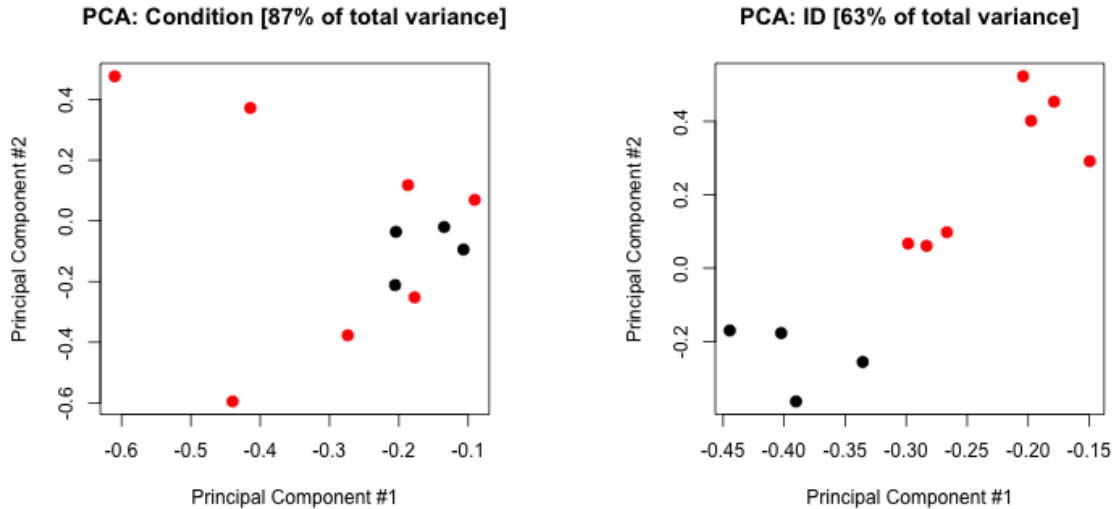
4.2 PCA plots

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

```
> dba.plotPCA(tamoxifen)
```

Legend
Resistant "black"
Responsive "red"

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 0.1, can be drawn as follows:

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

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)`. You may have to rotate the resultant plot top-to-bottom to align it with the standard two-dimension plot, but seeing the first three principal components can be a useful exploratory exercise.

4.3 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(values(tamoxifen.DB)$Fold<0)
```

```
[1] 157
```

```
> sum(values(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) indicating which of these distributions are significantly different from another distribution.

```
> pvals
```

	Resistant.DB	Responsive.DB	Resistant.DB+	Responsive.DB+
Resistant.DB	1.000000e+00	2.135309e-11	2.369406e-07	2.444282e-17
Responsive.DB	2.135309e-11	1.000000e+00	2.578124e-36	1.752937e-04
Resistant.DB+	2.369406e-07	2.578124e-36	1.000000e+00	2.926571e-41
Responsive.DB+	2.444282e-17	1.752937e-04	2.926571e-41	1.000000e+00
Resistant.DB-	2.358740e-16	9.612111e-09	5.511712e-31	2.084464e-03
Responsive.DB-	4.292073e-01	2.592570e-09	8.544980e-10	4.299065e-17
	Resistant.DB-	Responsive.DB-		
Resistant.DB	2.358740e-16	4.292073e-01		
Responsive.DB	9.612111e-09	2.592570e-09		
Resistant.DB+	5.511712e-31	8.544980e-10		
Responsive.DB+	2.084464e-03	4.299065e-17		
Resistant.DB-	1.000000e+00	2.588615e-18		
Responsive.DB-	2.588615e-18	1.000000e+00		

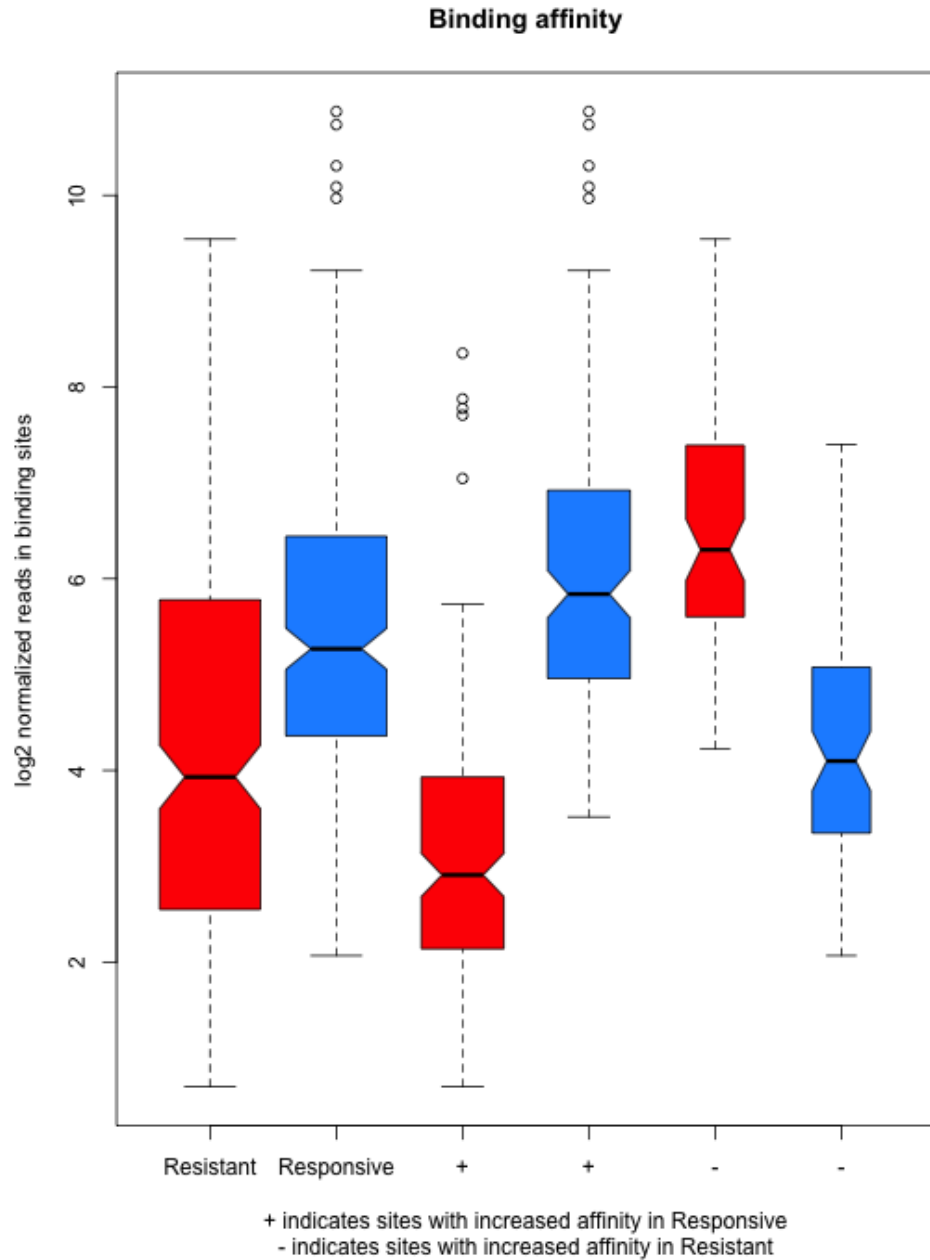


Figure 5: Box plots of read distributions for significantly differentially bound (DB) sites. Tamoxifen resistant samples are shown in red, and responsive samples are shown in blue. Left two boxes show distribution of reads over all DB sites in the Resistant and Responsive groups; middle two boxes show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

The significance of the overall difference in distribution of concentrations amongst the differentially bound sites in the two groups is shown to be $p\text{-value}=2.135309\text{e-}11$, while those between the Resistant and Responsive groups in the individual cases (increased in Responsive or Resistant) have $p\text{-values}$ computed as $2.926571\text{e-}41$ and $2.588615\text{e-}18$.

4.4 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)`.

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

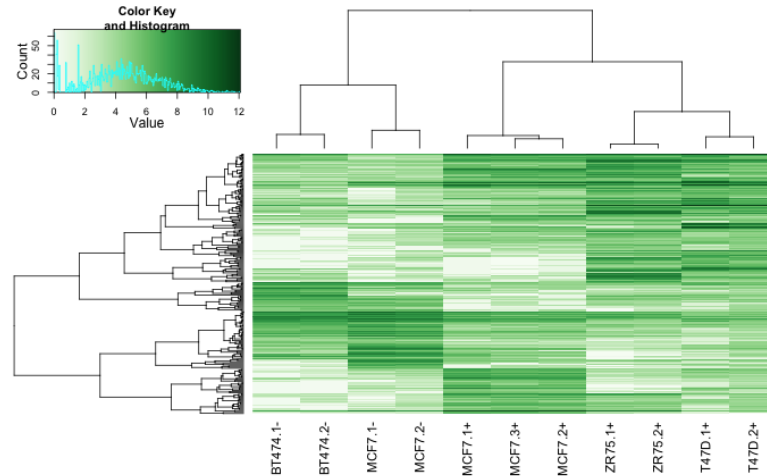


Figure 6: Binding affinity heatmap showing affinities for differentially bound sites. Samples cluster first by whether they are responsive to tamoxifen treatment, then by cell line. Clusters of binding sites show distinct patterns of affinity levels. Generated by: `dba.plotHeatmap(tamoxifen, contrast=1,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

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	Intervals
1	BT474.1-	BT474	ER	Resistant	1	1654

2	BT474.2-	BT474	ER Resistant	2	1654
3	MCF7.1+	MCF7	ER Responsive	1	1654
4	MCF7.2+	MCF7	ER Responsive	2	1654
5	MCF7.3+	MCF7	ER Responsive	3	1654
6	T47D.1+	T47D	ER Responsive	1	1654
7	T47D.2+	T47D	ER Responsive	2	1654
8	MCF7.1-	MCF7	ER Resistant	1	1654
9	MCF7.2-	MCF7	ER Resistant	2	1654
10	ZR75.1+	ZR75	ER Responsive	1	1654
11	ZR75.2+	ZR75	ER Responsive	2	1654

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"
```

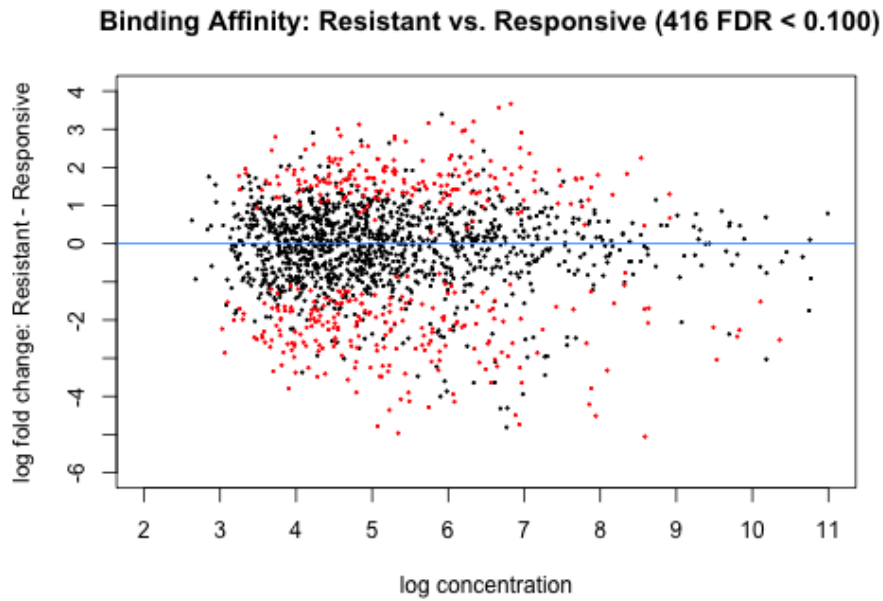


Figure 7: MA plot of Resistant-Responsive contrast, using MCF7 origin as a blocking factor, with sites identified as significantly differentially bound shown in red. Generated by: `dba.plotMA(tamoxifen, method=DBA_EDGER_BLOCK)`

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 Intervals
1  BT474.1-  BT474      ER   Resistant         1      1654
```

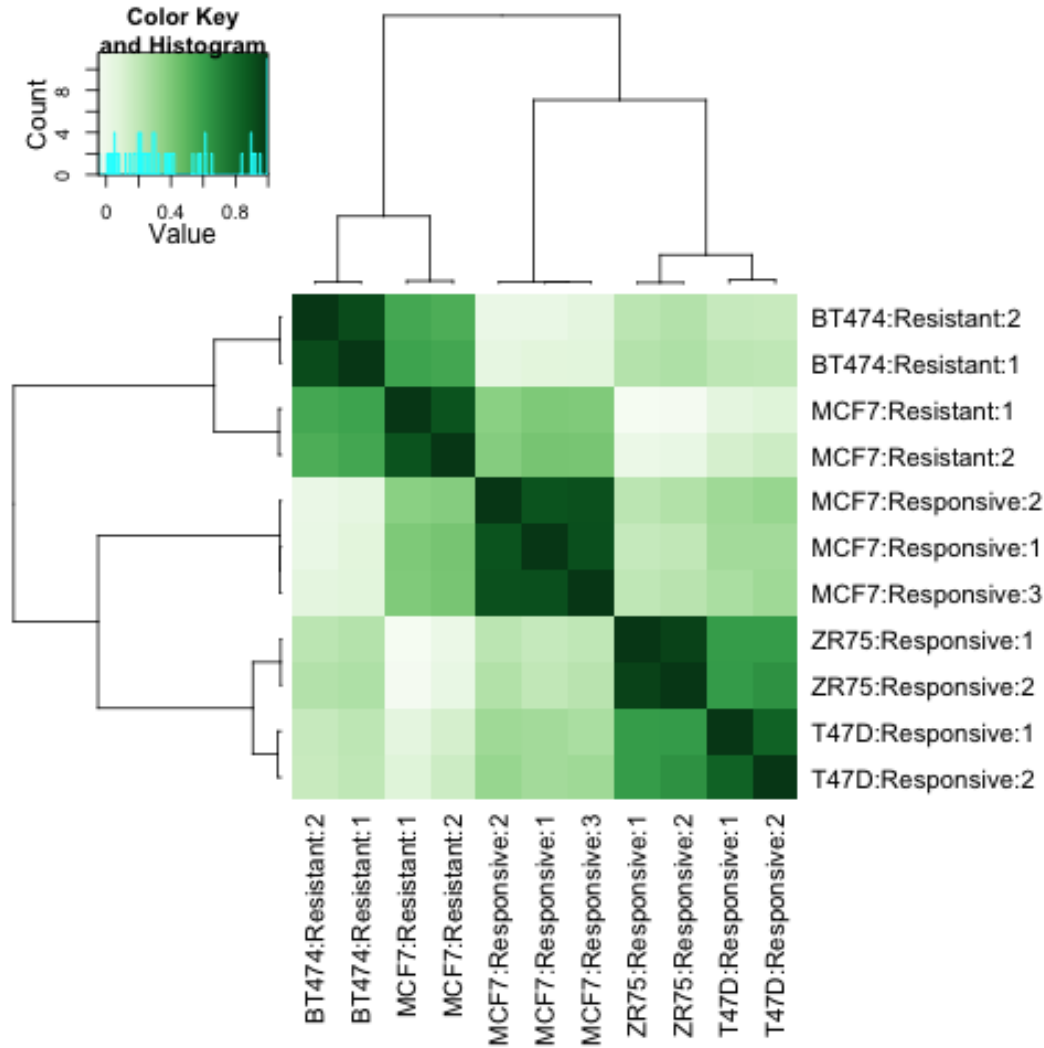


Figure 8: Correlation heatmap of using scores for significantly differentially bound sites for the Resistant-Responsive contrast, using MCF7 origin as a blocking factor. Generated by: `dba.plotHeatmap(tamoxifen, contrast=1, method=DBA_EDGER_BLOCK, attributes=c(DBA_TISSUE,DBA_CONDITION,DBA_REPLICATE))`

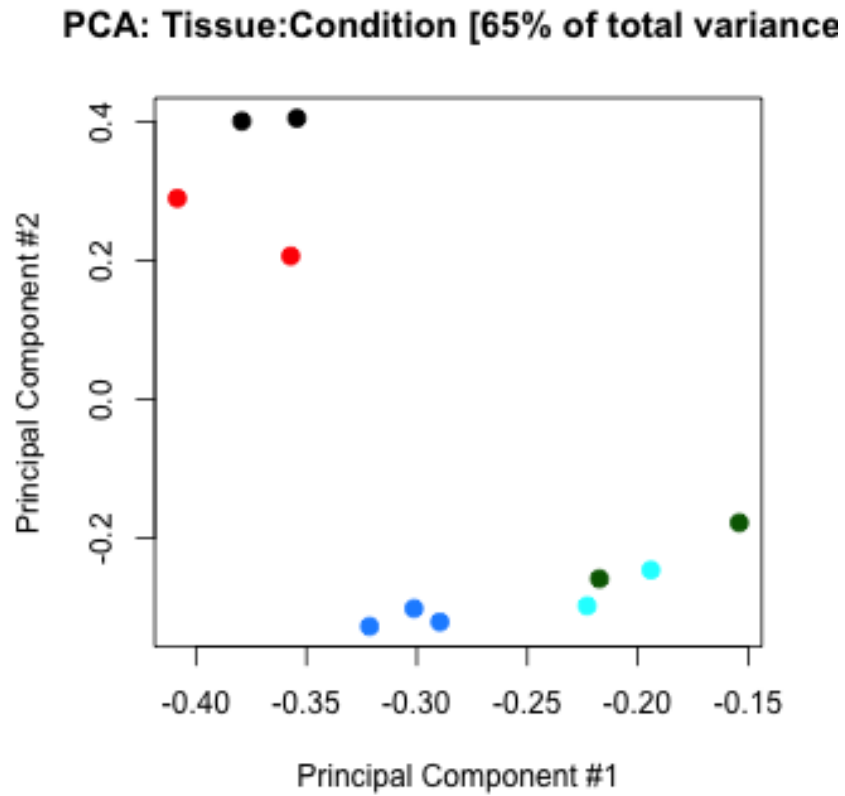


Figure 9: Plot of first two principal components, using scores for significantly differentially bound sites for the Resistant-Responsive contrast, using MCF7 origin as a blocking factor. Generated by: `dba.plotPCA(tamoxifen,contrast=1,method=DBA_EDGER_BLOCK, attributes=c(DBA_TISSUE,DBA_CONDITION))`

2	BT474.2-	BT474	ER Resistant	2	1654
3	MCF7.1+	MCF7	ER Responsive	1	1654
4	MCF7.2+	MCF7	ER Responsive	2	1654
5	MCF7.3+	MCF7	ER Responsive	3	1654
6	T47D.1+	T47D	ER Responsive	1	1654
7	T47D.2+	T47D	ER Responsive	2	1654
8	MCF7.1-	MCF7	ER Resistant	1	1654
9	MCF7.2-	MCF7	ER Resistant	2	1654
10	ZR75.1+	ZR75	ER Responsive	1	1654
11	ZR75.2+	ZR75	ER Responsive	2	1654

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	418				

While DESeq has a much more conservative approach to the single factor analysis, identifying only 54 sites as differentially bound, modelling the confounding fact results in an almost identical result. 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 7.1 on peak merging). A global overview can be obtained using the RATE mode of the `dba.overlap` function as follows:

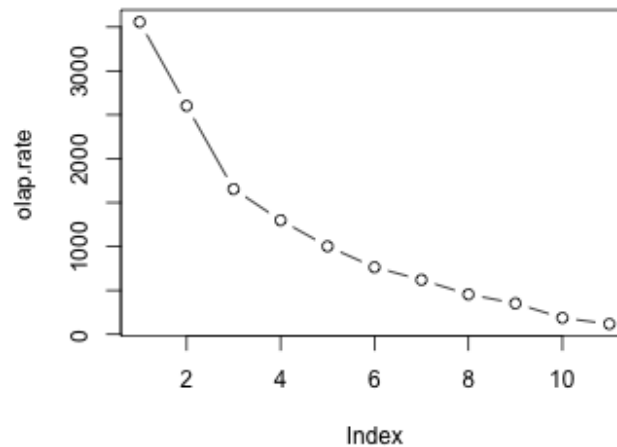


Figure 10: Overlap rate plot, showing how the number of overlapping peaks decreases as the overlap criteria becomes more stringent. X axis shows the number of peaksets in which the site is identified, while the Y axis shows the number of overlapping sites. Generated by plotting the result of: `dba.overlap(tamoxifen,mode=DBA_OLAP_RATE)`

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

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

`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')
```

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

6.2 Deriving consensus peaksets

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

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. For example we could replace the three tamoxifen-responsive MCF7 peaksets with one including the 1,222 peaks identified in at least two replicates by masking the non-consensus MCF7 peaksets (the `$Consensus` mask filters peaksets formed from existing peaksets using `dba.peakset`) as follows:

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

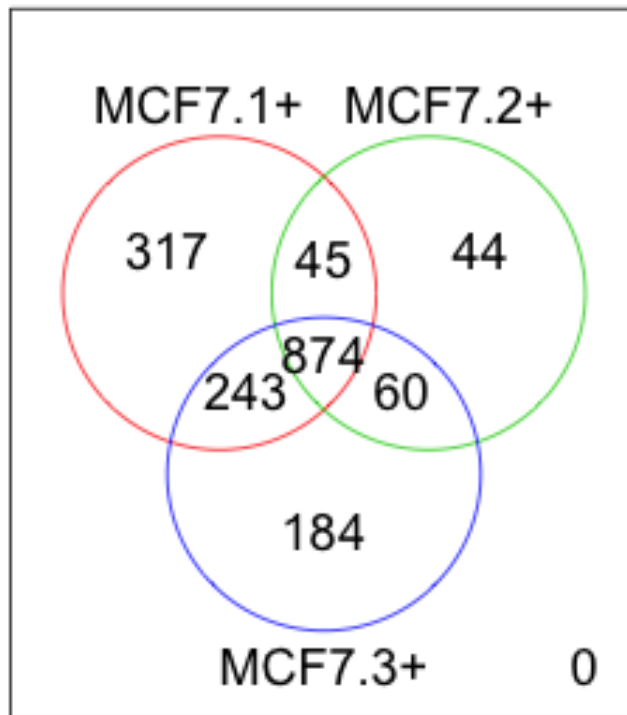


Figure 11: Venn diagram showing how the ER peak calls for three replicates of responsive MCF7 cell line overlap. Generated by plotting the result of: `dba.venn(tamoxifen,tamoxifen$mask$MCF7 & tamoxifen$mask$Responsive)`


```
> tamoxifen = dba.peakset(tamoxifen,tamoxifen$mask$MCF7&tamoxifen$mask$Responsive,
+                          sampID="MCF7+")
> tamoxifen = dba(tamoxifen,!(!tamoxifen$mask$Consensus&tamoxifen$mask$MCF7))
> tamoxifen
```

7 Samples, 2107 sites in matrix (3072 total):

	ID	Tissue	Factor	Condition	Replicate	Intervals
1	BT474.1-	BT474	ER	Resistant	1	1084
2	BT474.2-	BT474	ER	Resistant	2	1115
3	T47D.1+	T47D	ER	Responsive	1	509
4	T47D.2+	T47D	ER	Responsive	2	347
5	ZR75.1+	ZR75	ER	Responsive	1	2111
6	ZR75.2+	ZR75	ER	Responsive	2	1975
7	MCF7+	MCF7	ER	Responsive	1-2-3	1222

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$mask$Resistant,mode=DBA_OLAP_RATE)
[1] 1875 1298 597 436
> dba.overlap(tamoxifen,tamoxifen$mask$Responsive,mode=DBA_OLAP_RATE)
[1] 3208 2293 1217 807 584 265 161
```

For the Resistant group, we choose an overlap rate of two, leaving 1,298 sites, while for the Responsive group, we use an overlap rate of 3, leaving 1,217 sites, and look at the overlap between the two groups:

```
> tamoxifen = dba.peakset(tamoxifen,tamoxifen$mask$Resistant,
+                          sampID="Resistant",minOverlap=2)
> tamoxifen = dba.peakset(tamoxifen,tamoxifen$mask$Responsive,
+                          sampID="Responsive",minOverlap=3)
> dba.plotVenn(tamoxifen,tamoxifen$mask$Consensus)
```

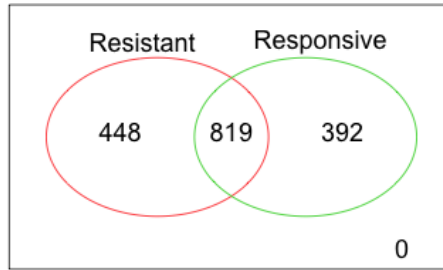


Figure 12: Venn diagram showing how the ER peak calls for two response groups overlap. Generated by plotting the result of: `dba.plotVenn(tamoxifen, tamoxifen$masks$Consensus)`

Figure 12 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. The sites can be obtained using `dba.overlap`:

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

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

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

GRanges with 211 ranges and 1 elementMetadata col:

	seqnames	ranges	strand	score
	<Rle>	<IRanges>	<Rle>	<numeric>
14	chr18	[521507, 522443]	*	0.128978267622691
18	chr18	[639178, 639957]	*	0.0254536579689849
20	chr18	[647089, 647869]	*	0.0178065034135036
22	chr18	[834959, 835851]	*	0.0887190625790319
27	chr18	[988767, 989698]	*	0.086873645309346
36	chr18	[1284247, 1285028]	*	0.0320210489586112
42	chr18	[1470236, 1470902]	*	0.0552776401270308
44	chr18	[1607943, 1608842]	*	0.0552169277363557
56	chr18	[2181095, 2181967]	*	0.0659388066011975

```

...      ...      ...      ...      ...
1556    chr18 [70126424, 70129225]    * | 0.0371779493864449
1575    chr18 [70877774, 70878592]    * | 0.0479562916713206
1599    chr18 [72231522, 72232225]    * | 0.0318654599563659
1600    chr18 [72235659, 72236252]    * | 0.0455384132542688
1601    chr18 [72252473, 72253355]    * | 0.0780761572604726
1610    chr18 [72679104, 72679670]    * | 0.0639424216427045
1620    chr18 [72934091, 72934934]    * | 0.0552534440090404
1641    chr18 [74667531, 74668298]    * | 0.0432033104075515
1643    chr18 [74753658, 74755251]    * | 0.0553678343818583

```

```
---
```

```
seqlengths:
```

```
chr18
```

```
NA
```

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

```
GRanges with 182 ranges and 1 elementMetadata col:
```

	seqnames	ranges	strand	score
	<Rle>	<IRanges>	<Rle>	<numeric>
17	chr18	[579002, 579929]	*	0.0466440809907735
25	chr18	[914869, 915486]	*	0.0331046835979069
62	chr18	[2254244, 2255010]	*	0.047010413661554
65	chr18	[2306710, 2307486]	*	0.0695151519760949
75	chr18	[2528718, 2529414]	*	0.0552311386757906
82	chr18	[2866735, 2867500]	*	0.0892392452619188
91	chr18	[3097746, 3098504]	*	0.0657304041711667
117	chr18	[3501724, 3502670]	*	0.0392796883450164
121	chr18	[3560991, 3561575]	*	0.0460720325127482
...
1566	chr18	[70675631, 70676438]	*	0.149469635564882
1581	chr18	[71510377, 71510930]	*	0.262298383582386
1588	chr18	[71664005, 71665146]	*	0.0240431677566015
1590	chr18	[71668140, 71669131]	*	0.0391031324657972
1592	chr18	[71767043, 71767558]	*	0.0520175791599085
1604	chr18	[72413975, 72414794]	*	0.0549224423910508
1607	chr18	[72527374, 72529329]	*	0.0463255405424242
1622	chr18	[72968464, 72969990]	*	0.197588276744784
1623	chr18	[72978978, 72980033]	*	0.1229542152509

```
---
```

```
seqlengths:
```

```
chr18
```

```
NA
```

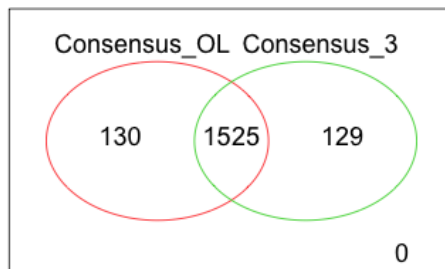


Figure 13: Venn diagram showing how the ER peak calls for two different ways of deriving consensus peaksets. Generated by plotting the result of: `dba.plotVenn(tamoxifen,14:15)`

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

6.4 Comparison of occupancy and affinity based analyses

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

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

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

Figure 13 shows that the two sets agree on about 85% of their sites, so the results should be directly comparable.⁴

Next re-load the affinity analysis:

```
> data(tamoxifen_analysis)
```

⁴Alternatively, we could re-run the analysis using the newly derived consensus peakset by passing it into the counting function: `> tamoxifen = dba.count(tamoxifen, peaks = dba.peakset(tamoxifen, peaks=14, bRetrieve=T))`

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

```
[1] 313
```

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

```
[1] 391
```

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

```
[1] 821
```

Comparing these numbers verifies the similarity with those seen in Figure 12.

Focusing on only those sites identified as significantly differentially bound ($\text{FDR} \leq 0.1$), however, shows a different story than that obtainable using only occupancy data:

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

```
[1] 41
```

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

```
[1] 119
```

```
> bothGroups = values(tamoxifen.DB)$Called1>=2 & values(tamoxifen.DB)$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. 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. Secondly, while the sites unique to a sample group are more likely to be identified as differentially bound, 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*. 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 internal technical details of DiffBind processing.

7.1 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.

7.2 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/SAM/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** using the **DGEList** with the **dispersion** set based on the **bTagwise** parameter.

If the method is **DBA_EDGER_GLM** (the default), then 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, **exactTest** is performed, using either common or tagwise dispersion depending on the value specified for **bTagwise**.

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

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

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

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

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

7.3 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/SAM/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/SAM/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`.

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

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

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

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

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

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 2.15.1 (2012-06-22)
```

```
Platform: i386-pc-mingw32/i386 (32-bit)
```

```
locale:
```

```
[1] LC_COLLATE=C
```

```
[2] LC_CTYPE=English_United States.1252
```

```
[3] LC_MONETARY=English_United States.1252
```

```
[4] LC_NUMERIC=C
```

```
[5] LC_TIME=English_United States.1252
```

```
attached base packages:
```

```
[1] parallel stats graphics grDevices utils datasets methods
```

```
[8] base
```

```
other attached packages:
```

```
[1] DESeq_1.8.3 locfit_1.5-8 DiffBind_1.2.4
```

```
[4] Biobase_2.16.0 GenomicRanges_1.8.13 IRanges_1.14.4
```

```
[7] BiocGenerics_0.2.0
```

```
loaded via a namespace (and not attached):
```

```
[1] AnnotationDbi_1.18.3 DBI_0.2-5 RColorBrewer_1.0-5
```

```
[4] RSQLite_0.11.1 XML_3.9-4.1 amap_0.8-7
```

```
[7] annotate_1.34.1 edgeR_2.6.12 gdata_2.11.0
```

```
[10] genefilter_1.38.0 geneplotter_1.34.0 gplots_2.11.0
```

```
[13] grid_2.15.1 gtools_2.7.0 lattice_0.20-10
```

```
[16] limma_3.12.3 splines_2.15.1 stats4_2.15.1
```

[19] survival_2.36-14 tools_2.15.1 xtable_1.7-0
[22] zlibbioc_1.2.0

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.
- 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>.
- 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.