dba.plotProfile Demonstration

Rory Stark

18 May 2021

Contents

1 Download data and set working directory

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

```
tmpdir <- tempdir()
url <- 'https://content.cruk.cam.ac.uk/bioinformatics/software/DiffBind/DiffBind_vignette_data.tar.gz'
file <- basename(url)
```

```
options(timeout=600)
download.file(url, file.path(tmpdir,file))
untar(file.path(tmpdir,file), exdir = tmpdir )
knitr::opts_knit$set(root.dir=file.path(tmpdir,"DiffBind_Vignette"))
```

The examples in this workbook all follow the analysis or the impact of tamoxifen resistance on ER binding as discussed in detail in the `DiffBind` vignette.

2 Introduction

The `dba.plotProfile()` function enables the computation of peakset profiles and the plotting of complex heatmaps. It serves as a front-end to enable experiments analyzed using `DiffBind` to more easily use the profiling and plotting functionality provided by the `profileplyr` package written by Tom Carroll and Doug Barrows.

Processing proceed in two phases.

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

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

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

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

3 Default plots

The default plot depend on whether or not an analysis has been completed.

3.1 Default plot: no analysis

If no analysis has been completed, the default plot will include all samples, in a single group. They will be merged based on the `DBA_REPLICATE` attribute, such that each sample class will have one heatmap based on the normalized mean read counts for all the replicate samples in that class.

By default, up to 1,000 of the consensus sites (randomly sampled) will be included, in a single group. If the genome is supported, annotation to nearby genomic features (promoters, genes, intragenic) will be determined and plotted.

```
data(tamoxifen_counts)
tamoxifen$config$RunParallel <- TRUE
profiles <- dba.plotProfile(tamoxifen)
```

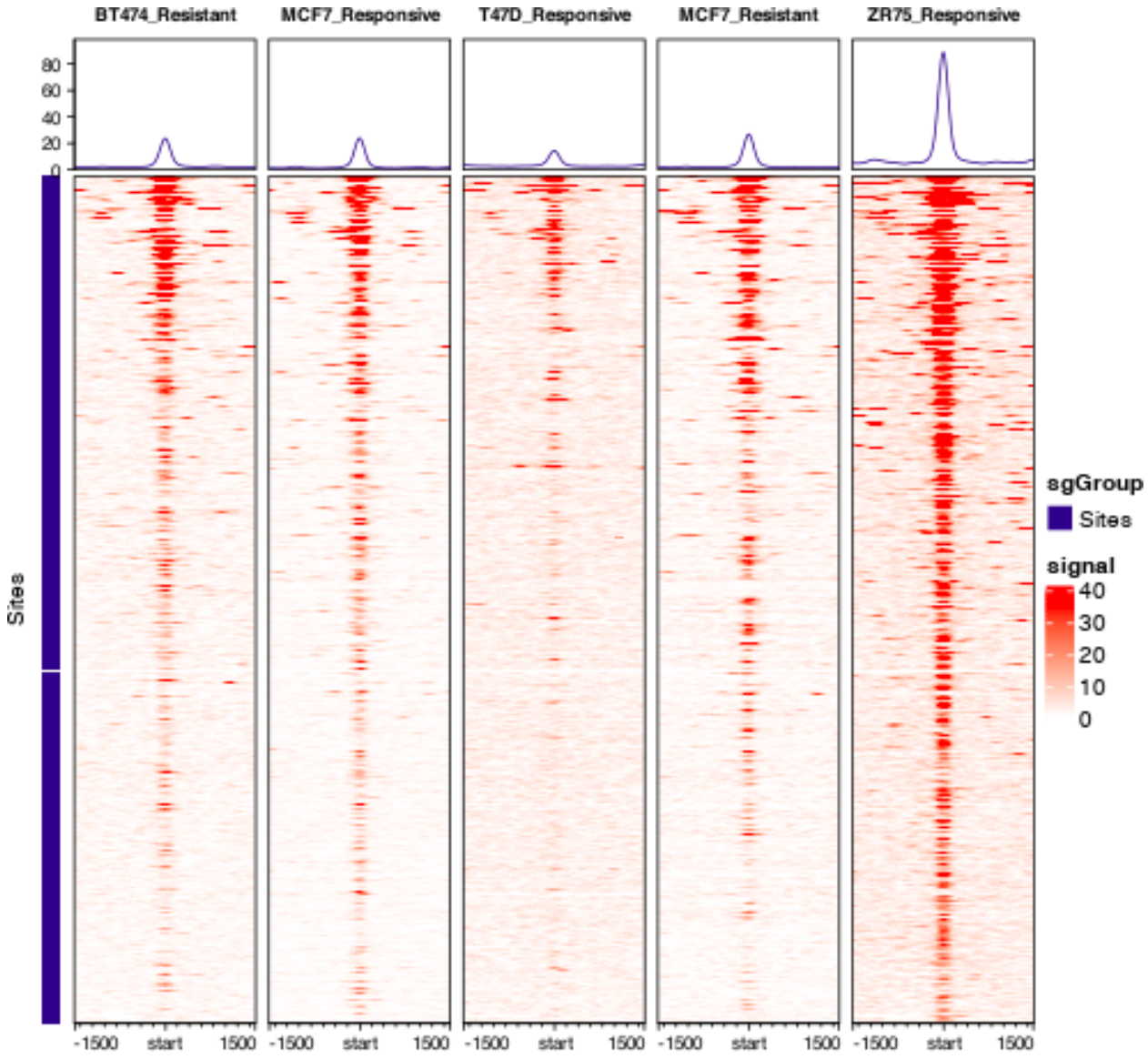
```
##
```

```
##
```

```
## Generating profiles...
```

```
dba.plotProfile(profiles)
```

```
## Plotting...
```



3.2 Default plot: analysis

If an analysis has been completed, the default will be based on the results of the first contrast. If the contrast compares two sample groups, the samples in those groups will be included, with the heatmaps colored

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

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

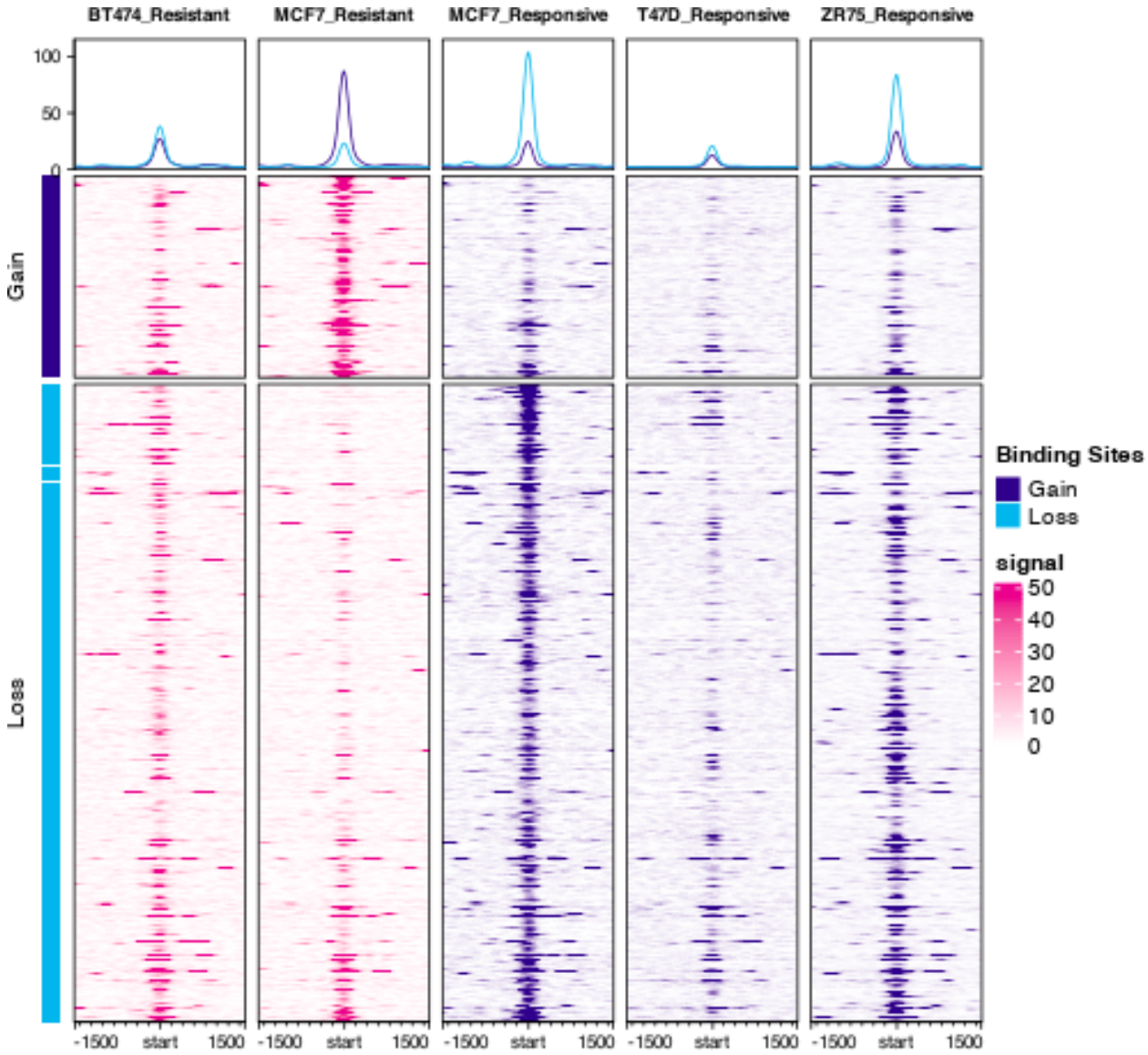
```
data(tamoxifen_analysis)
tamoxifen$config$RunParallel <- TRUE
profiles <- dba.plotProfile(tamoxifen)
```

```
## Generating report-based DBA object...
```

```
## Generating profiles...
```

```
dba.plotProfile(profiles)
```

```
## Plotting...
```



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

3.3 Default plot: merging cell types

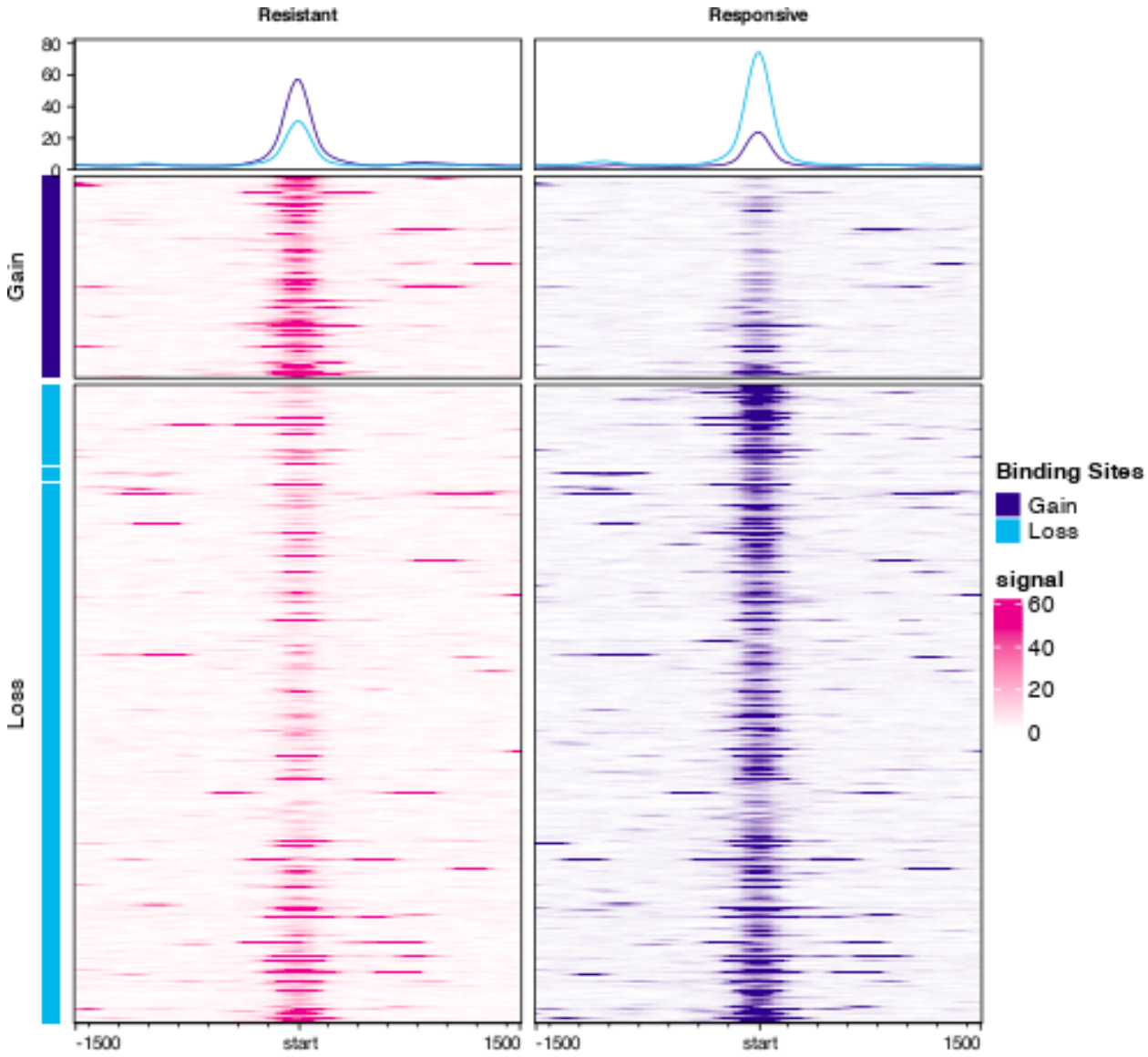
In the sample experiment, there are multiple sample groups comprising each side of the contrast: the Resistant class has two sample groups based on the BT474 and MCF7 cell lines, while the Responsive class has three groups, based on the MCF7, T47D, and ZR75 cell lines. If we want to generate composite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

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

```
## Generating report-based DBA object...
## Generating profiles...
```

```
dba.plotProfile(profiles)
```

```
## Plotting...
```



4 Specifying samples

The `samples` parameter can be used to explicitly specify a subset of samples to include in the plot.

4.1 Specifying samples: mask (sample vector)

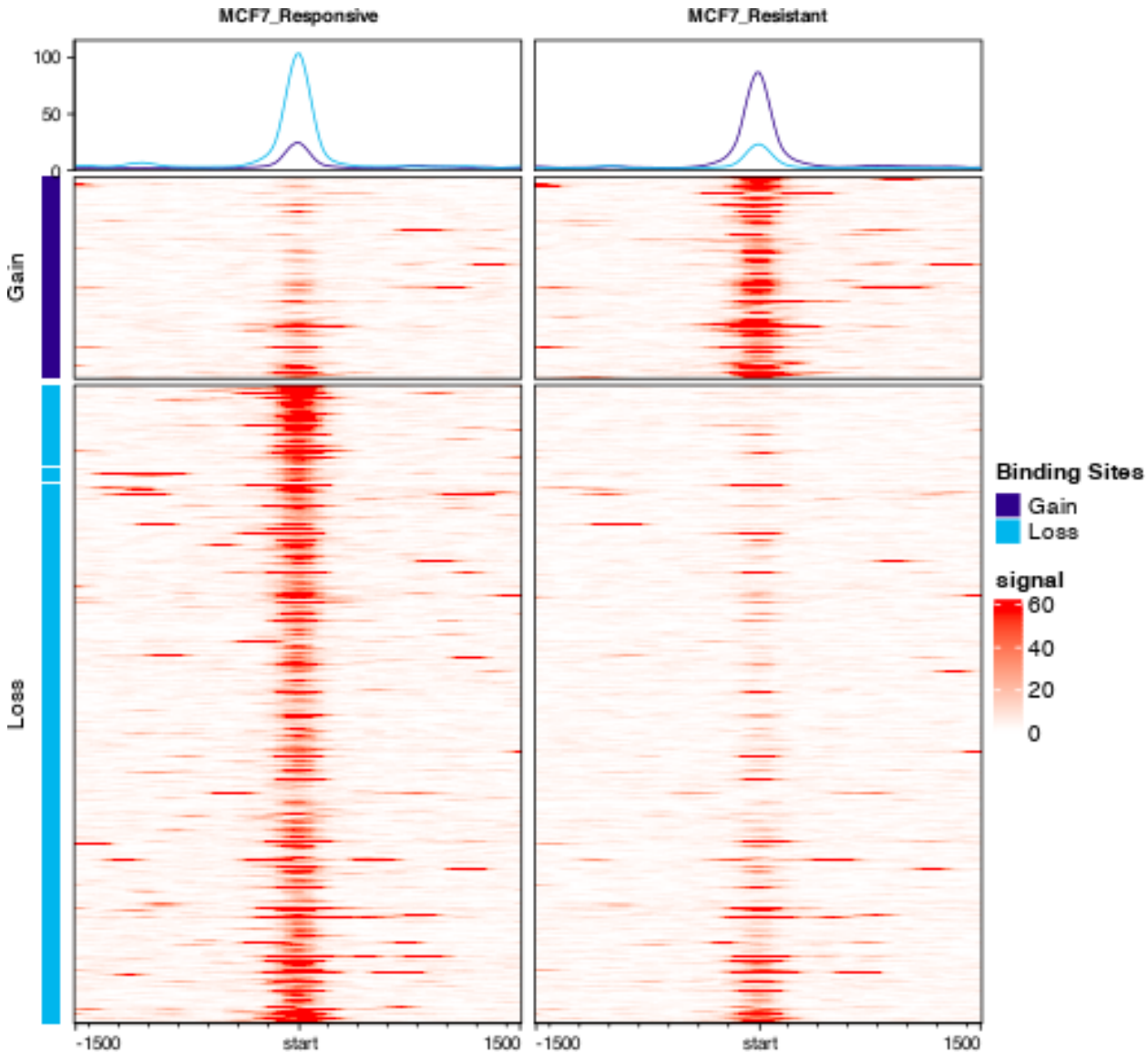
For example, we can limit the plot to only the MCF7 samples:

```
profiles <- dba.plotProfile(tamoxifen,samples=tamoxifen$mask$MCF7,
                            labels=c("MCF7_Responsive","MCF7_Resistant"))
```

```
## Generating report-based DBA object...
## Generating profiles...
```

```
dba.plotProfile(profiles)
```

```
## Plotting...
```



The resulting plot shows two sample groups: MCF7_Resistant and MCF7_Responsive. This is because a) by default, the plots are based on the first contrast, b) by default, all samples in a group that differ only in their `DBA_REPLICATE` values are merged. As a result, this plot is showing all of the Gain/Loss sites for the first (Resistant vs Responsive) contrast, but *only using data for the MCF7 samples in the profiles.*

4.2 Specifying samples: preventing sample merging

To see each of the MCF7 samples separately, specify `merge=NULL`, which will prevent the replicates from being merged:

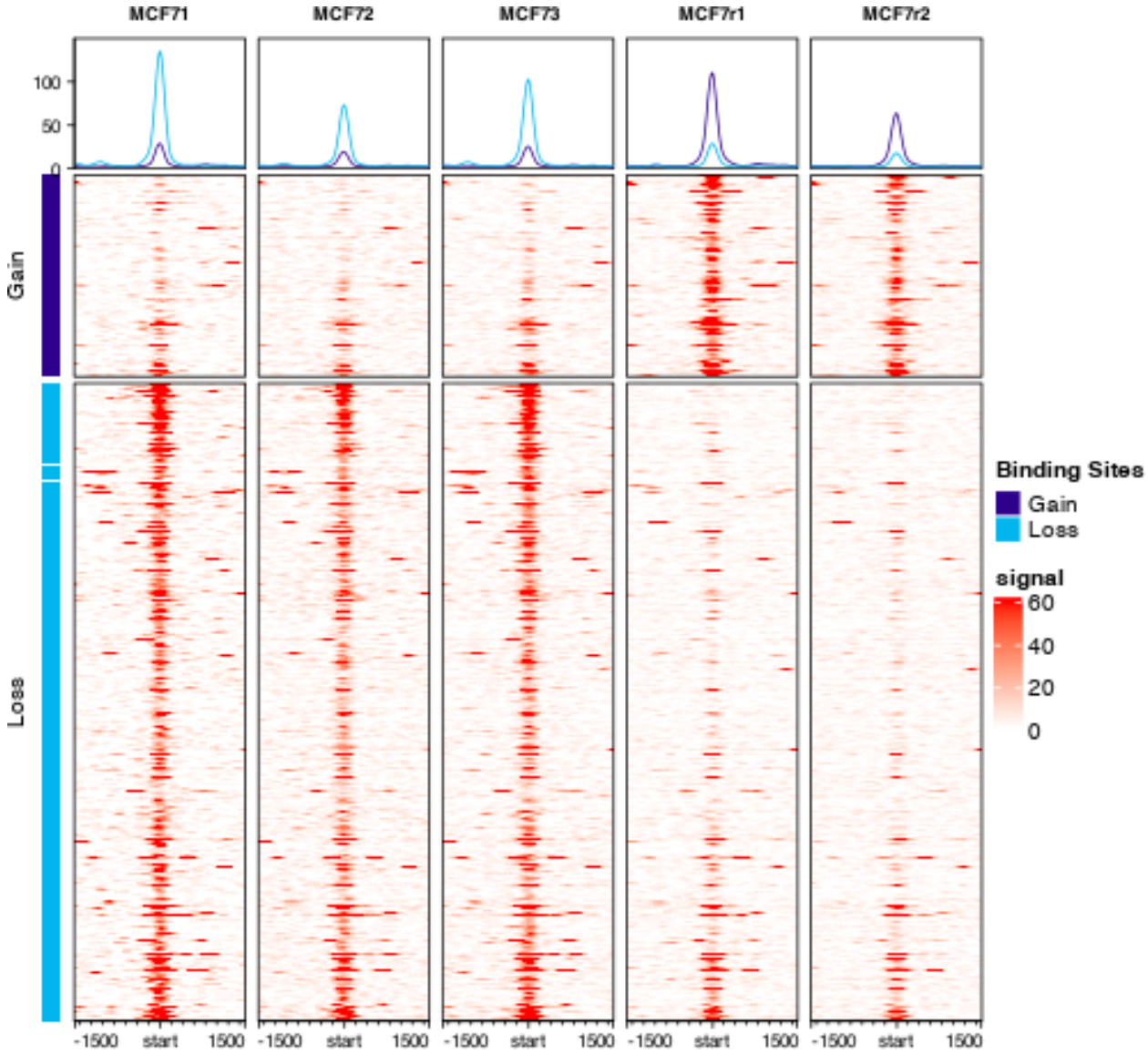
```
profiles <- dba.plotProfile(tamoxifen, samples=tamoxifen$mask$MCF7, merge=NULL)
```

```
## Generating report-based DBA object...
```

```
## Generating profiles...
```

```
dba.plotProfile(profiles)
```

```
## Plotting...
```



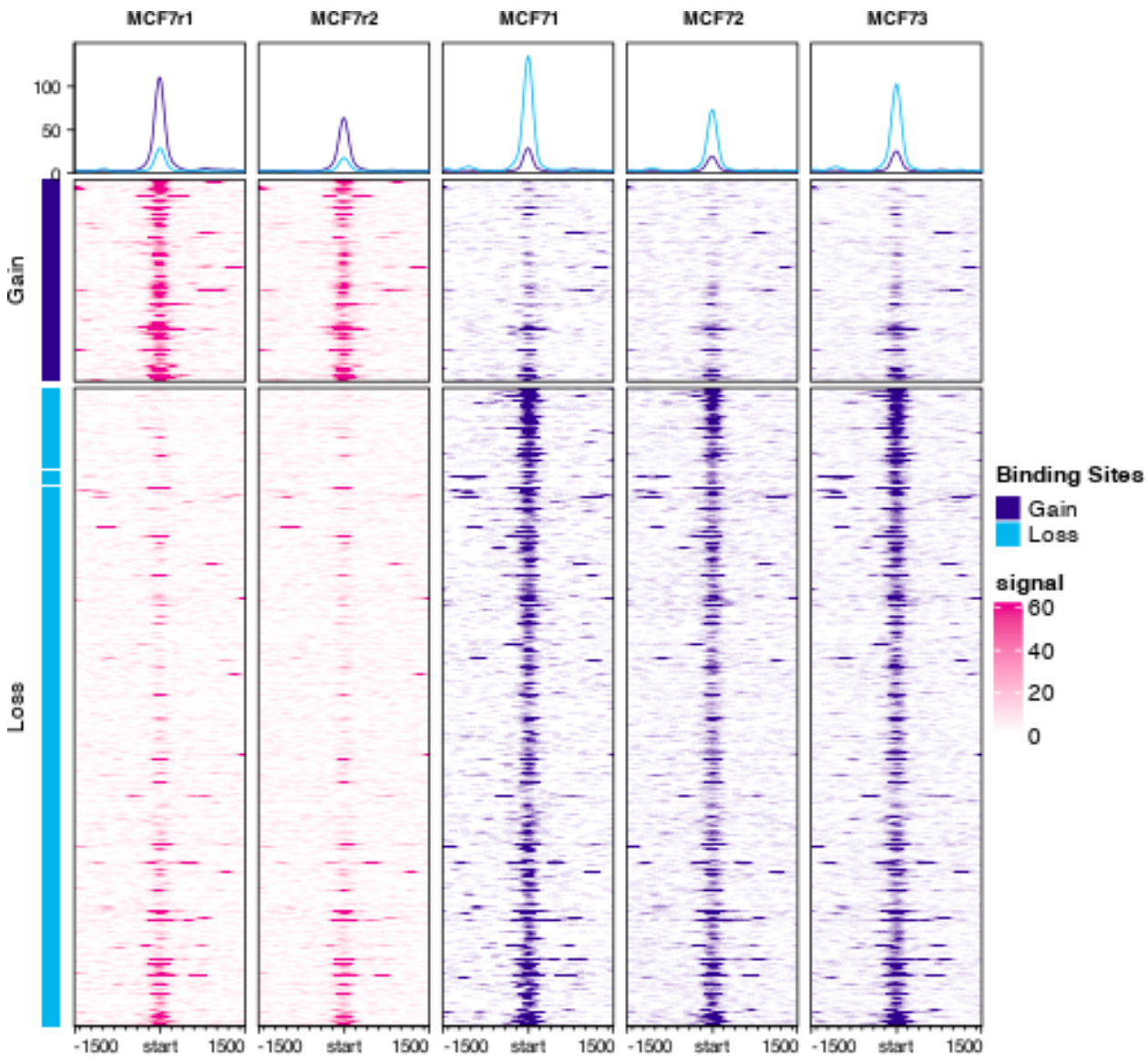
In this plot, the profile heatmaps are all plotted using the same color (red). This is because the specification for `samples` is a single vector.

4.3 Specifying samples: list of masks (sample vector list)

By specifying the samples as a list of two sample masks, the Resistant and Responsive MCF7 samples can be plotted using different colors:

```
profiles <- dba.plotProfile(tamoxifen,
                            samples=list(MCF7_Resistant=8:9,MCF7_Responsive=3:5),
                            merge=NULL)
```

```
## Generating report-based DBA object...
## Generating profiles...
dba.plotProfile(profiles)
## Plotting...
```



5 Specifying sites

There are a number of ways to specify which sites (peaks/genomic intervals) to include in the plot using the `sites` parameter.

5.1 Specifying sites: GRanges

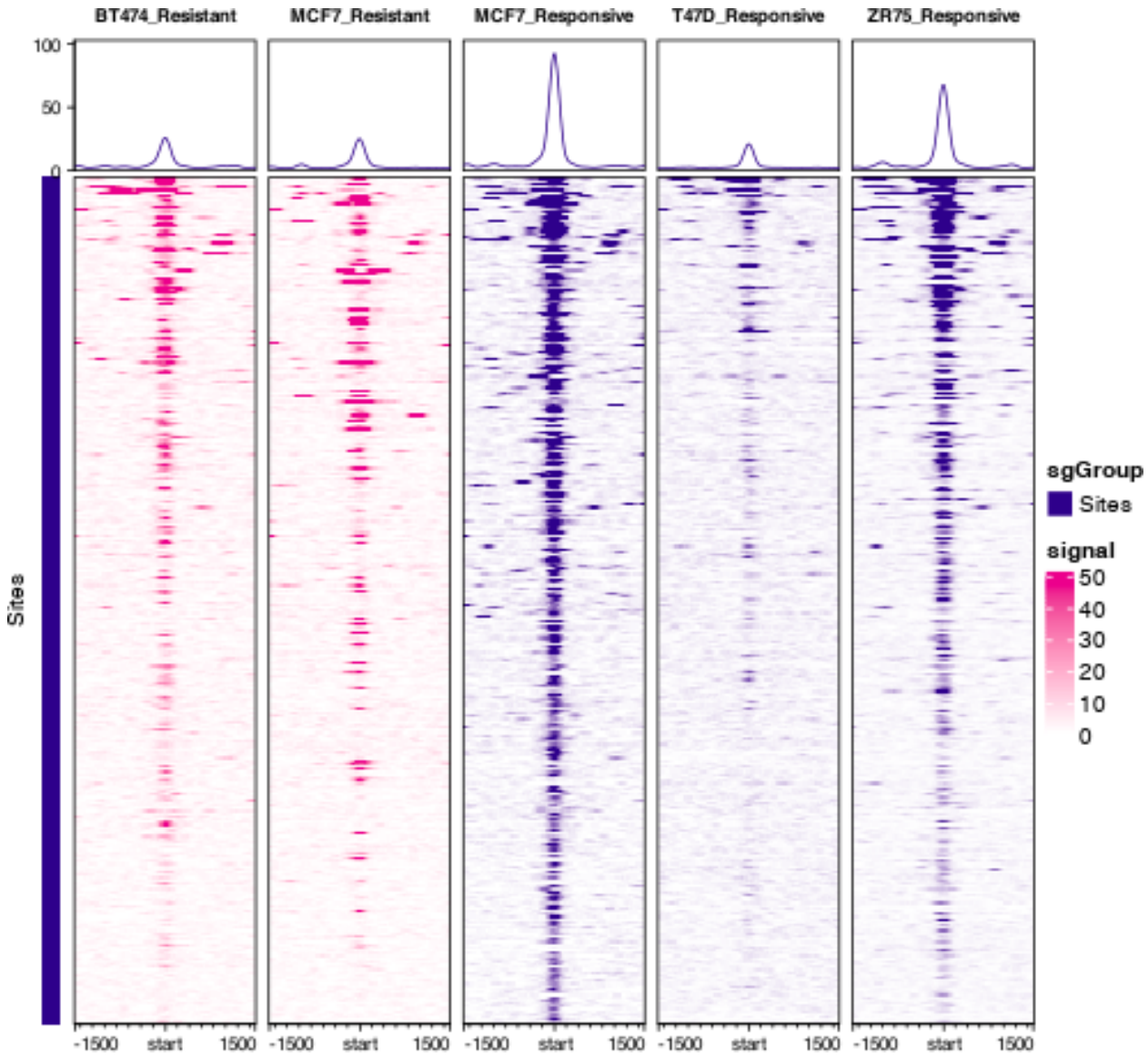
A set of intervals in the form of a GRanges object can be passed directly. For example, the dba.report() function can be used to obtain a set of differentially bound sites, then sub-setted to include those with a log fold change greater than 2, sorted from greatest positive fold change to greatest negative fold change:

```
rep <- dba.report(tamoxifen)
rep <- rep[abs(rep$Fold) > 2,]
rep <- rep[order(rep$Fold, decreasing=TRUE),]
profiles <- dba.plotProfile(tamoxifen, sites=rep,
                            samples=list(Resistant=tamoxifen$mask$Resistant,
                                         Responsive=tamoxifen$mask$Responsive))
```

```
## Generating profiles...
```

```
dba.plotProfile(profiles)
```

```
## Plotting...
```



5.2 Specifying sites: GRangesList

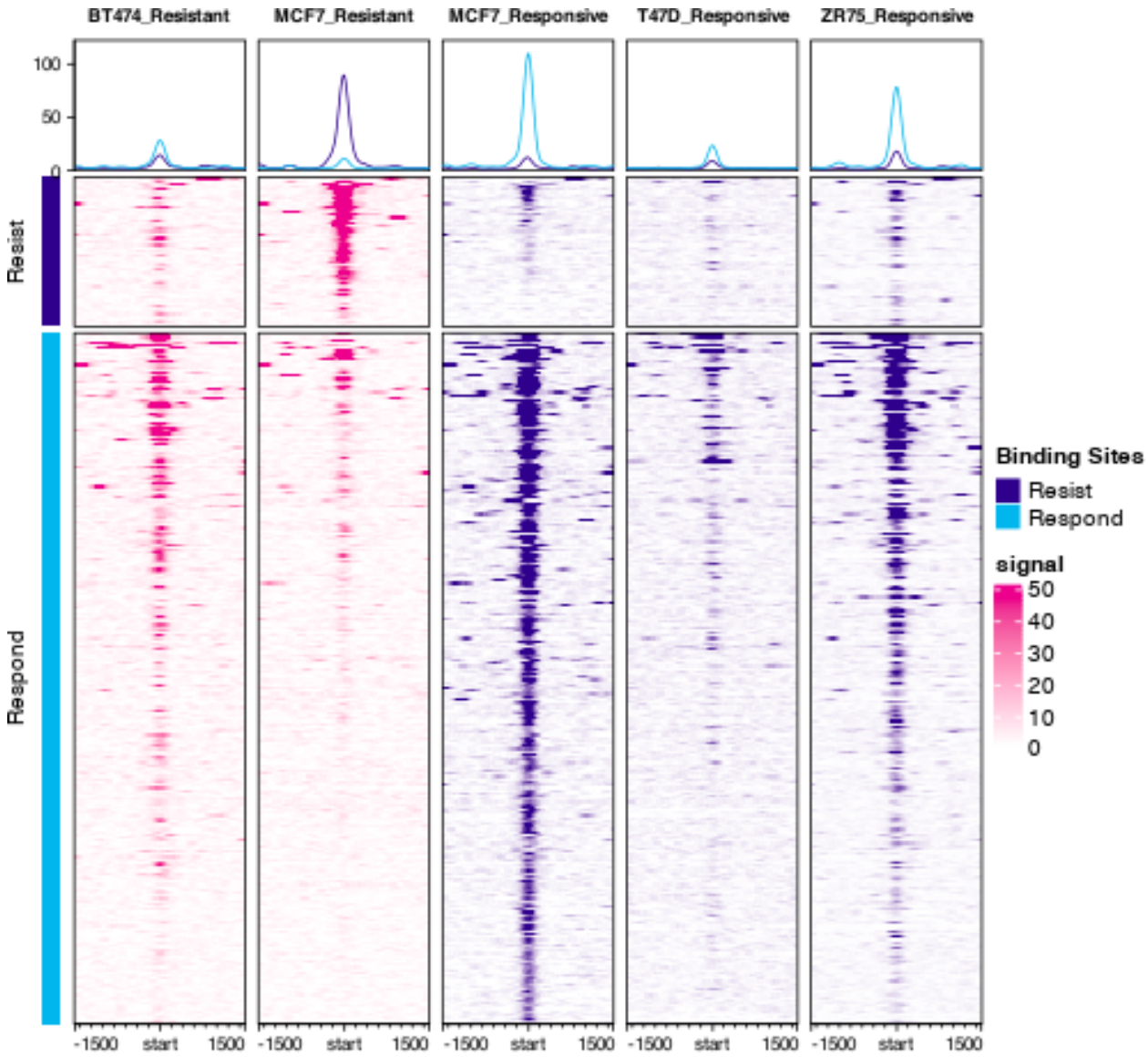
Alternatively, a GRangesList object can be used to specify **groups** of sites, in this case separating the Gain and Loss sites:

```
repList <- GRangesList(Gain=rep[rep$Fold>0,],Loss=rep[rep$Fold<0,])
profiles <- dba.plotProfile(tamoxifen, sites=repList,
                            samples=list(Resistant=tamoxifen$mask$Resistant,
                                         Responsive=tamoxifen$mask$Responsive),
                            labels=list(sites=c("Resist","Respond")),
                            score="Fold")
```

```
## Generating profiles...
```

```
dba.plotProfile(profiles)
```

```
## Plotting...
```



5.3 Specifying sites: by contrast number

Another way to specify sets of sites to profile is to use a report-based DBA object. This can be done implicitly, using a contrast, or explicitly, using the `dba.report()` function. We have already seen the implicit use in the default case where the first contrast is used (`sites=1`). If there are multiple contrasts, an alternative one may be selected:

```
tamoxifen <- dba.contrast(tamoxifen,contrast=c("Tissue","ZR75","BT474"))
tamoxifen <- dba.analyze(tamoxifen)
```

```
## Analyzing...
```

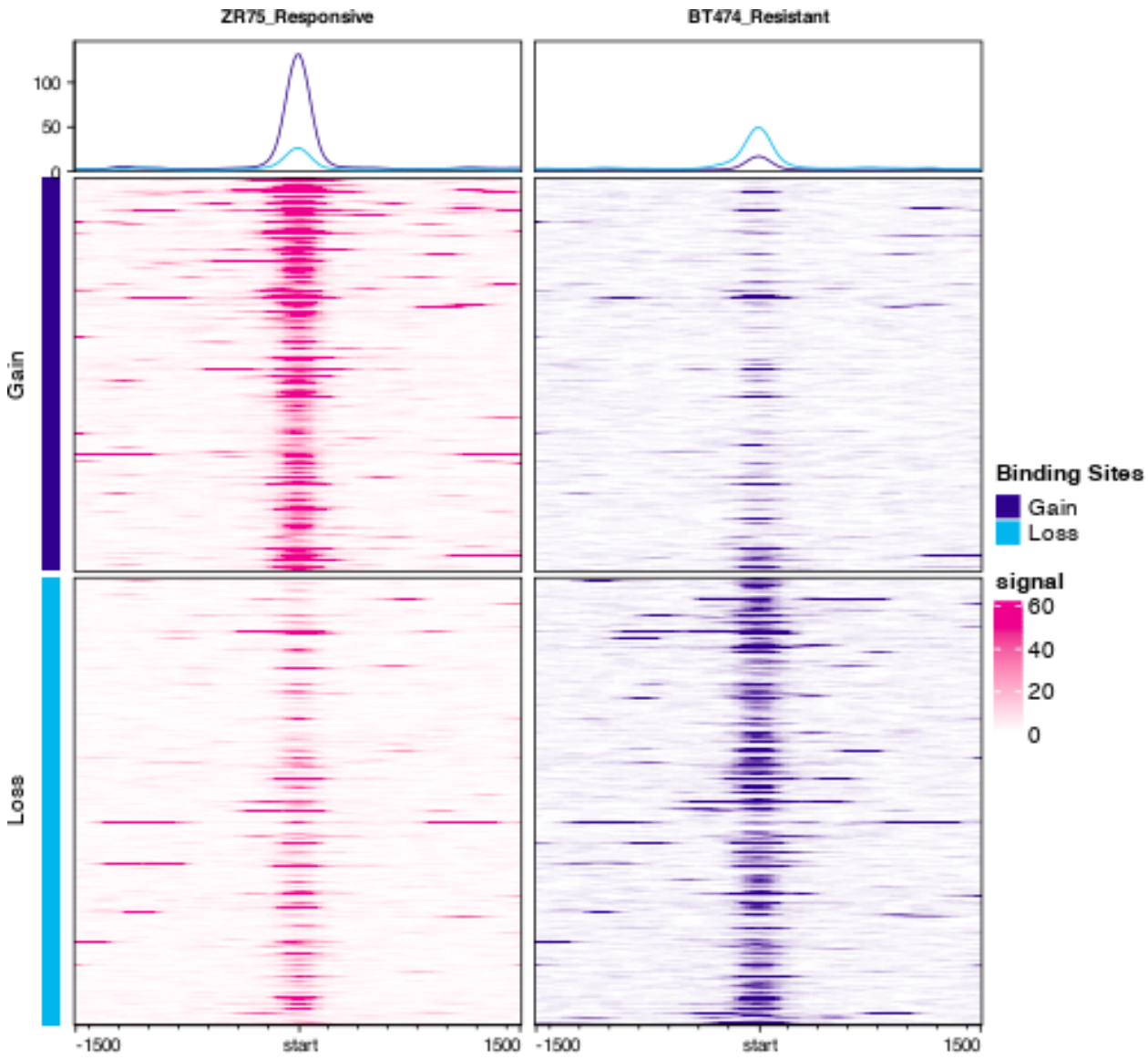
```
profiles  <- dba.plotProfile(tamoxifen, sites=2)
```

```
## Generating report-based DBA object...
```

```
## Generating profiles...
```

```
dba.plotProfile(profiles)
```

```
## Plotting...
```



5.4 Specifying sites: report-based DBA object

The report-based object can also be passed in explicitly:

```
repObj <- dba.report(tamoxifen, contrast=2,
                     bDB=TRUE, bGain=TRUE, bLoss=TRUE,bAll=FALSE)
```

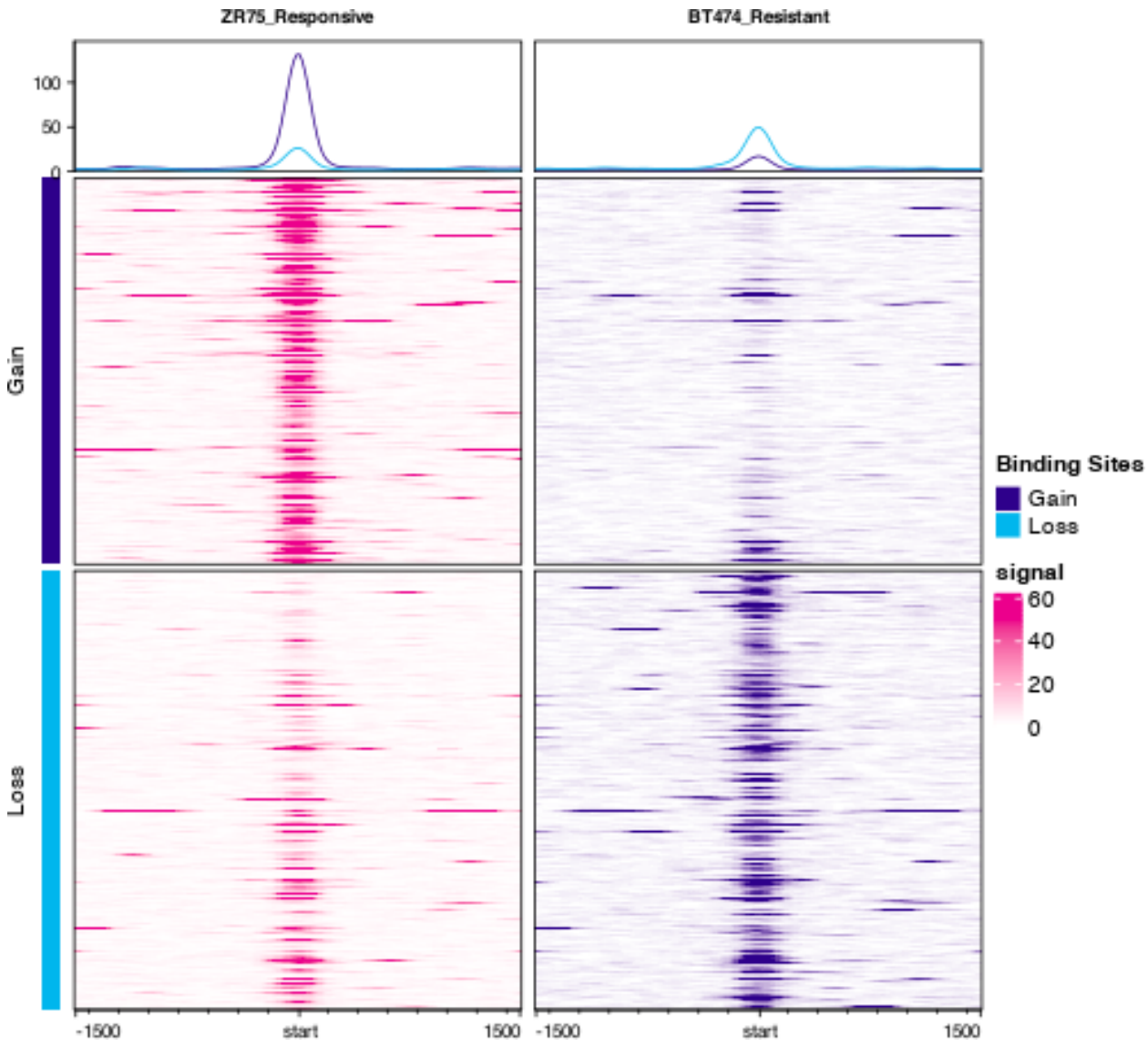
```
## Generating report-based DBA object...
```

```
profiles <- dba.plotProfile(tamoxifen, sites=repObj,
                            samples=list(ZR75=tamoxifen$mask$ZR75,
                                         BT474=tamoxifen$mask$BT474))
```

```
## Generating profiles...
```

```
dba.plotProfile(profiles)
```

```
## Plotting...
```



5.5 Specifying sites: report-based DBA object (multiple contrasts)

More complex report-based objects can be used as well. For example we can plot the differentially bound sites from each of the two contrasts across all the samples:

```
repObj <- dba.report(tamoxifen, contrast=1:2, bDB=TRUE)
```

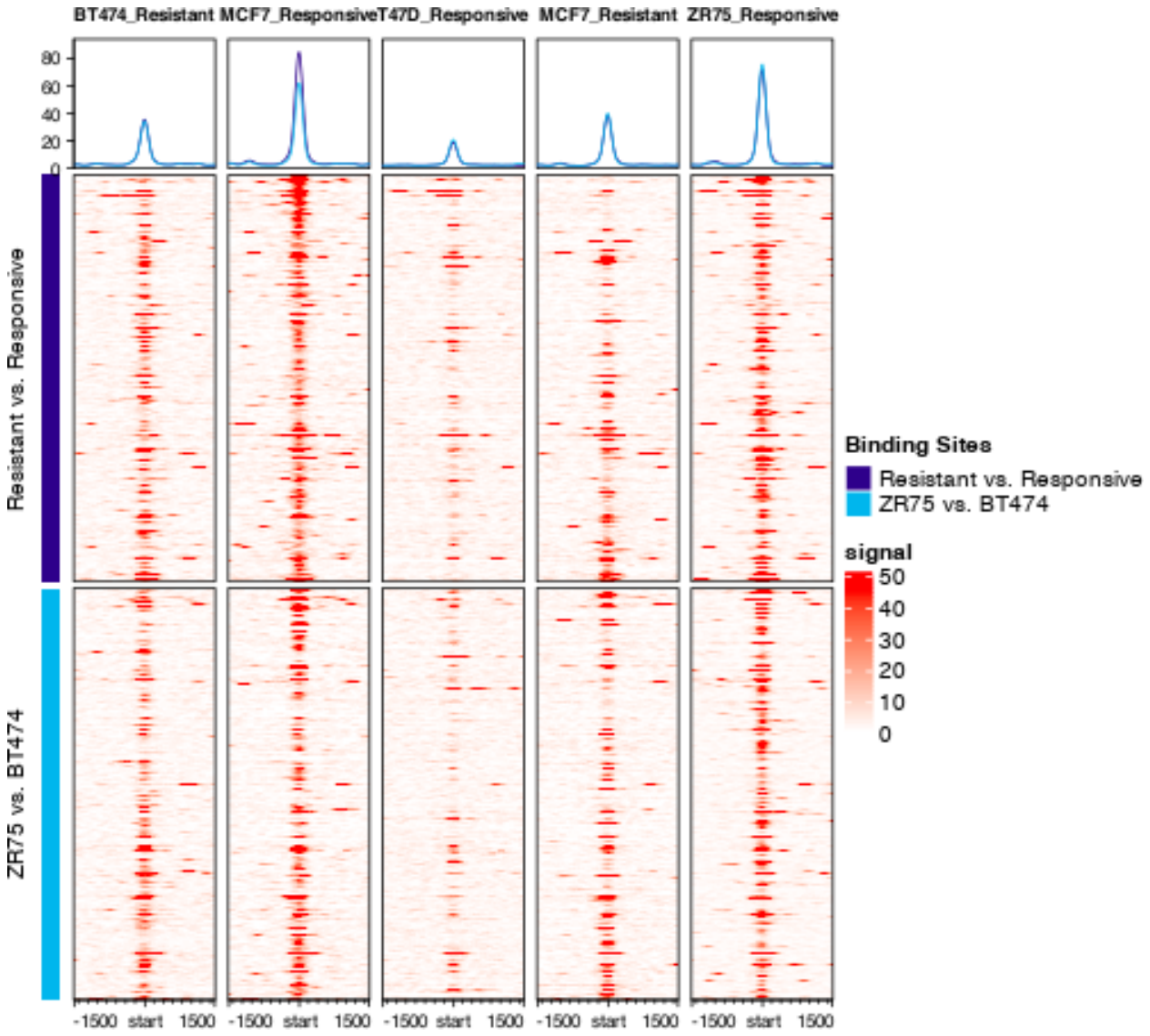
```
## Generating report-based DBA object...
```

```
profiles <- dba.plotProfile(tamoxifen, sites=repObj)
```

```
## Generating profiles...
```

```
dba.plotProfile(profiles)
```

```
## Plotting...
```



5.6 Specifying samples and sites

We can further refine the sites by considering the overlap between the differentially bound sites identified in the two contrasts:

```
overlaps <- dba.plotVenn(repObj,1:2)
```

Now we can generate separate profiles for the sites common to both contrasts, as well as those unique to one of them:

```
names(overlaps) <- c("Resistant/Responsive","ZR75/BT474","Both")
profiles <- dba.plotProfile(tamoxifen, sites=overlaps,
                            samples=list(ZR75=tamoxifen$masks$ZR75,
                                         BT474=tamoxifen$masks$BT474))
```

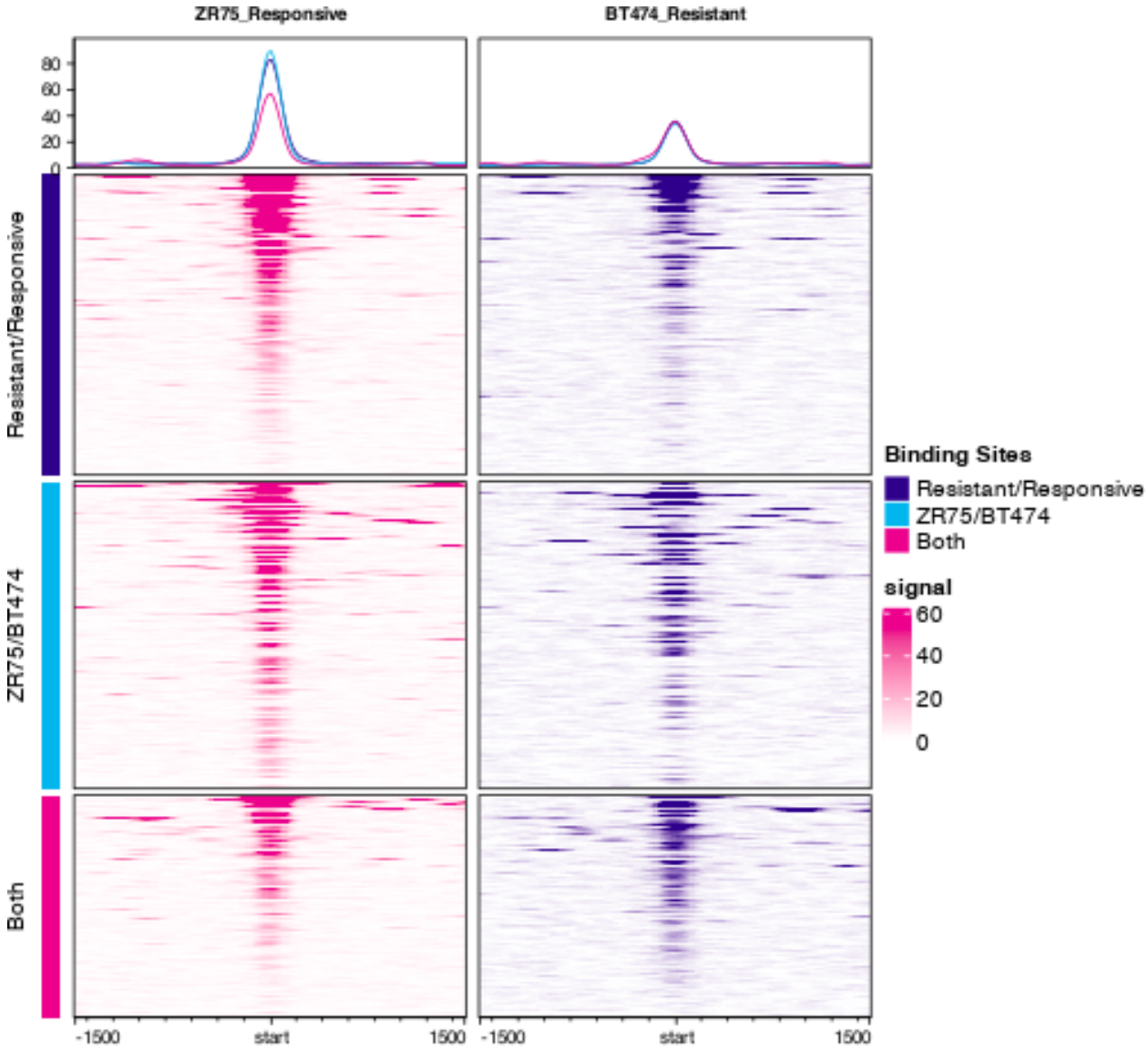
```
##

##

## Generating profiles...
```

```
dba.plotProfile(profiles)
```

```
## Plotting...
```



6 Normalization

By default, the read counts in the profiles are adjusted by dividing by the normalization factors. This can be altered by setting the `normalize` parameter.

6.1 Normalization: using raw (non-normalized) counts

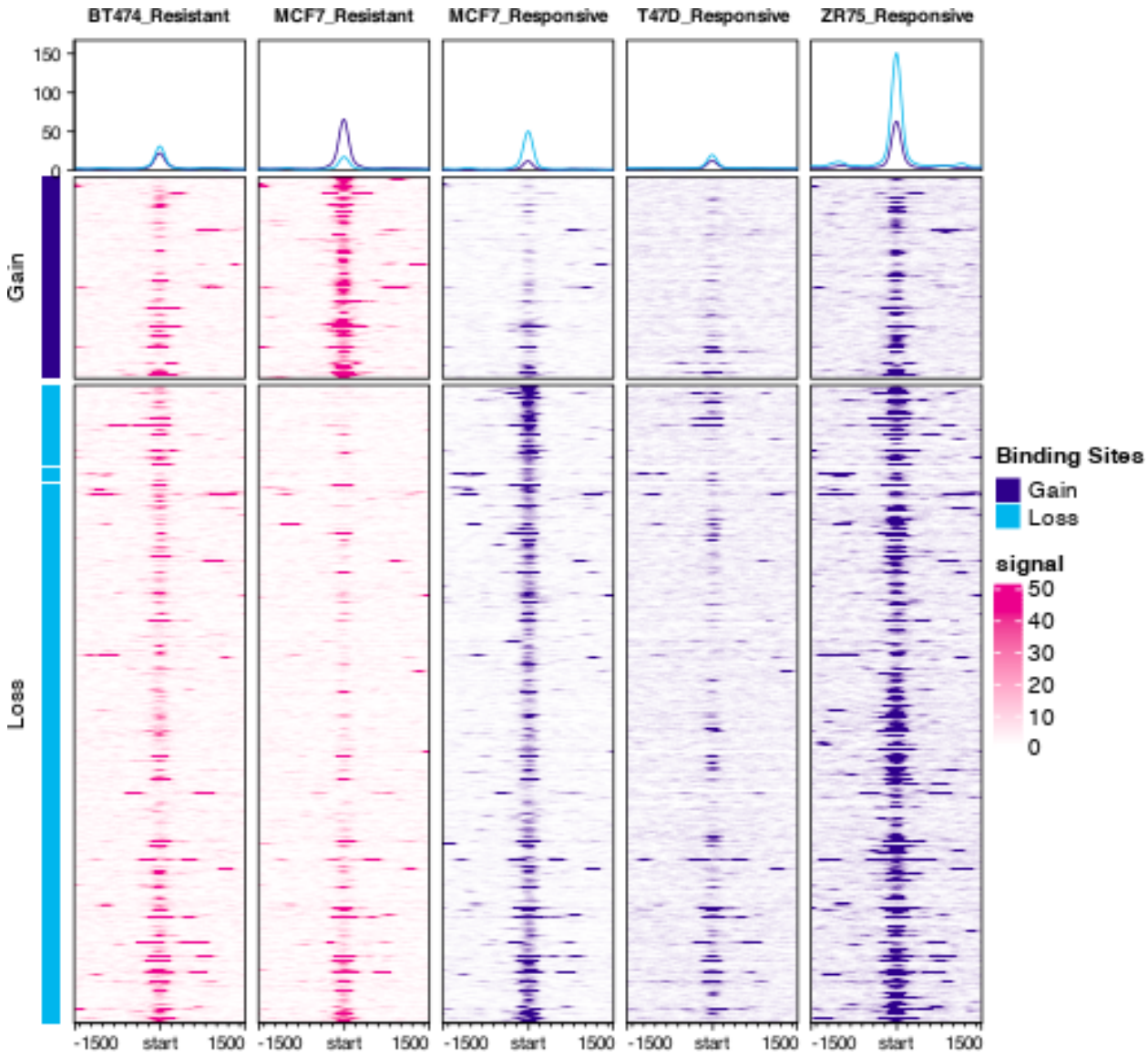
Normalization can be bypassed by setting `normalize=FALSE`, resulting in profiles based on the raw number of reads overlapping each bin:

```
profiles <- dba.plotProfile(tamoxifen, normalize=FALSE)
```

```
## Generating report-based DBA object...
## Generating profiles...
```

```
dba.plotProfile(profiles)
```

```
## Plotting...
```



Compare with the second plot (Default plot: analysis). The most obvious difference is in the profiles for the ZR75 samples, which are much stronger in the non-normalized plot, but which are adjusted down in the normalized version.

6.2 Normalization: specifying normalization factors

The relative strength of individual samples can be adjusted by explicitly passing in a vector of normalization values, overriding those computed by `DiffBind`. Factors must be passed in for each sample prior to merging.

For example, suppose we wanted to emphasize the T47D samples:

```
normfacs <- rep(1,11)
normfacs[8:9] <- .1
profiles <- dba.plotProfile(tamoxifen, normalize=normfacs)
```

```
##
```

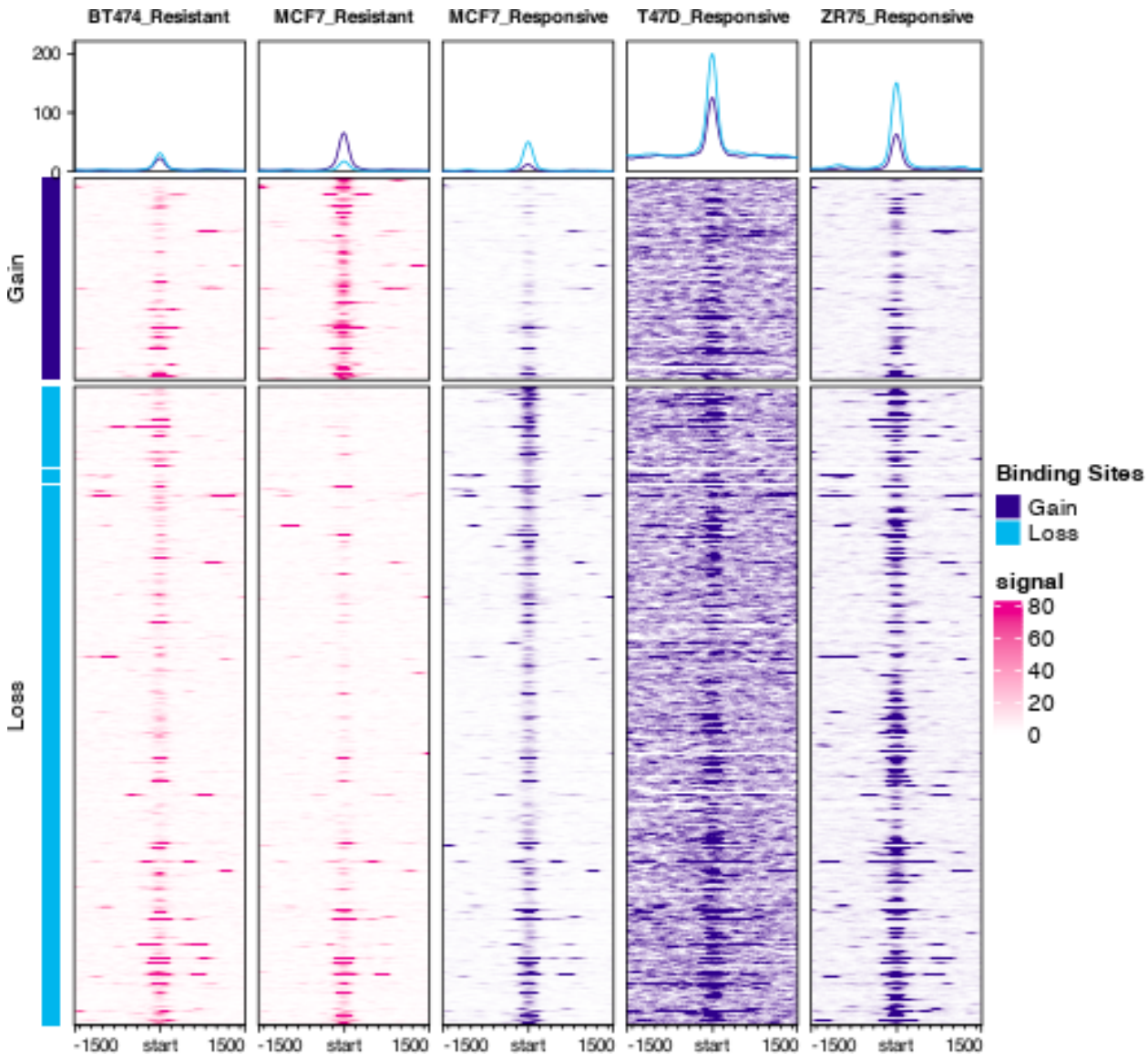
```
##
```

```
## Generating report-based DBA object...
```

```
## Generating profiles...
```

```
dba.plotProfile(profiles)
```

```
## Plotting...
```



7 Changing profiling parameters

Parameters the control the profiling computations can be supplied to `dba.plotProfile()`. These are a subset of the parameters associated with the `profileplyer::BamBigwig_to_chipProfile()` function.

7.1 Profiling parameters: `percentOfRegion`

The default profile `style` is "point", which counts reads overlapping fixed-size bins around each site. The default `bin_size` is 20 base pairs, and the default `distanceAround` value, which controls how many base pairs up- and down-stream of each site to profile, is 1500 base pairs. This means that each row in the profile plot comprises 3000 base pairs, divided into 150 bins (each being 20bp).

The alternative way to specify the profiling `style` is "percentOfRegion", where the bin size is set by dividing the width of the sites to be plotted by the value of the `nOfWindows` parameter. In this case `distanceAround` represents the percentage of the site length to plot (up- and down- stream). For example, if `distanceAround=300`, then the profiled region will extend 3x the width of each profiled site, for a total of seven site widths (3 downstream, the site, and 3 upstream). In the example data set, where the peak widths are standardized to 400bp, if `nOfWindows=20`, then the bin size is 400/20=20bp. If `distanceAround=300`, then the region to be profiled will be extended 300% of the 400bp, or 1200bp up and downstream, for a total of 1200+1200+400=2800bp, divided into seven sets of 20bp bins, for a total of 140 bins to be profiled:

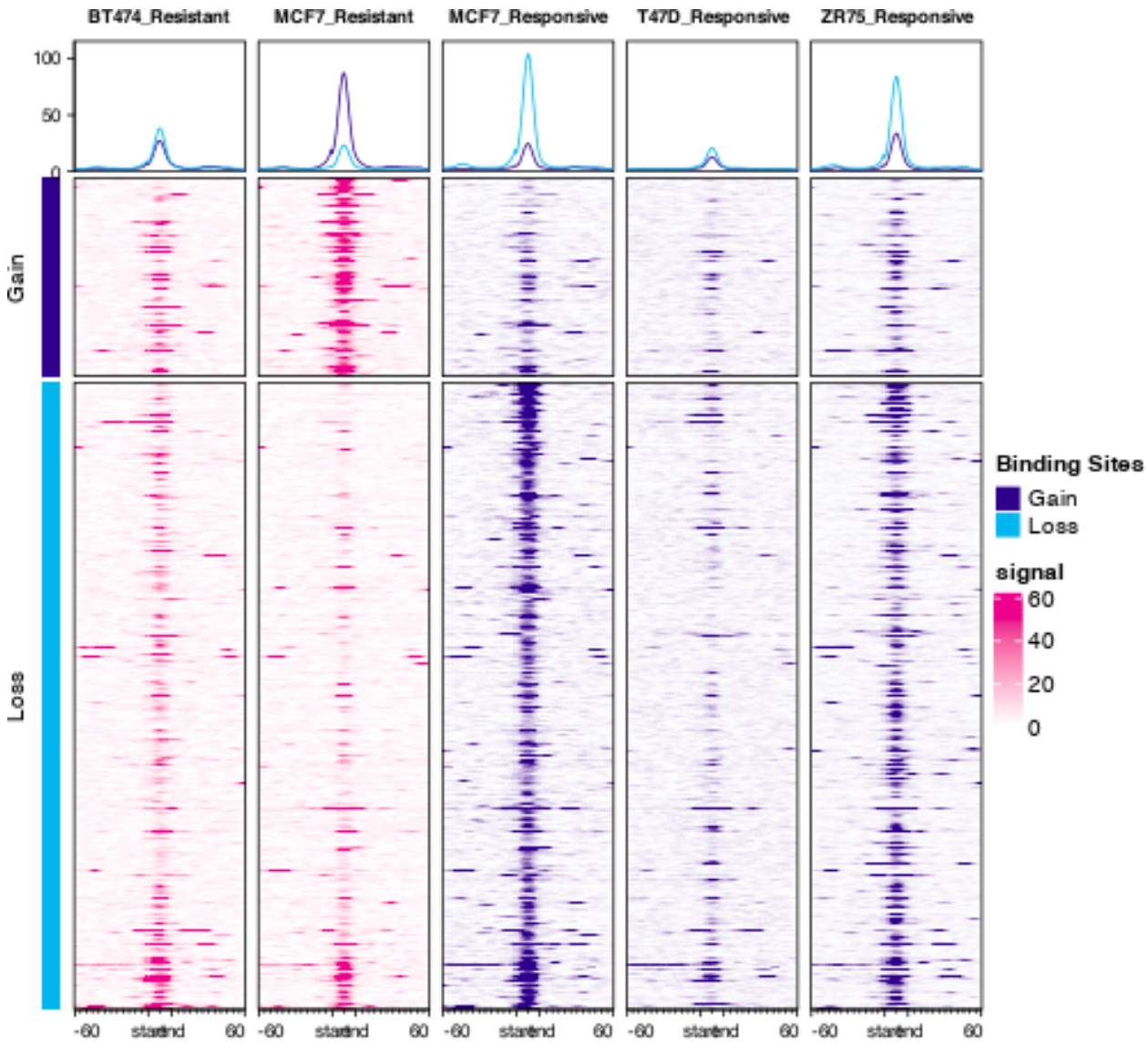
```
profiles <-  dba.plotProfile(tamoxifen, style="percentOfRegion",
                             nOfWindows=20, distanceAround=300)
```

```
## Generating report-based DBA object...
```

```
## Generating profiles...
```

```
dba.plotProfile(profiles)
```

```
## Plotting...
```



Other profiling parameters may also be changed; see the help page for `BamBigwig_to_chipProfile`.

8 Changing plotting parameters

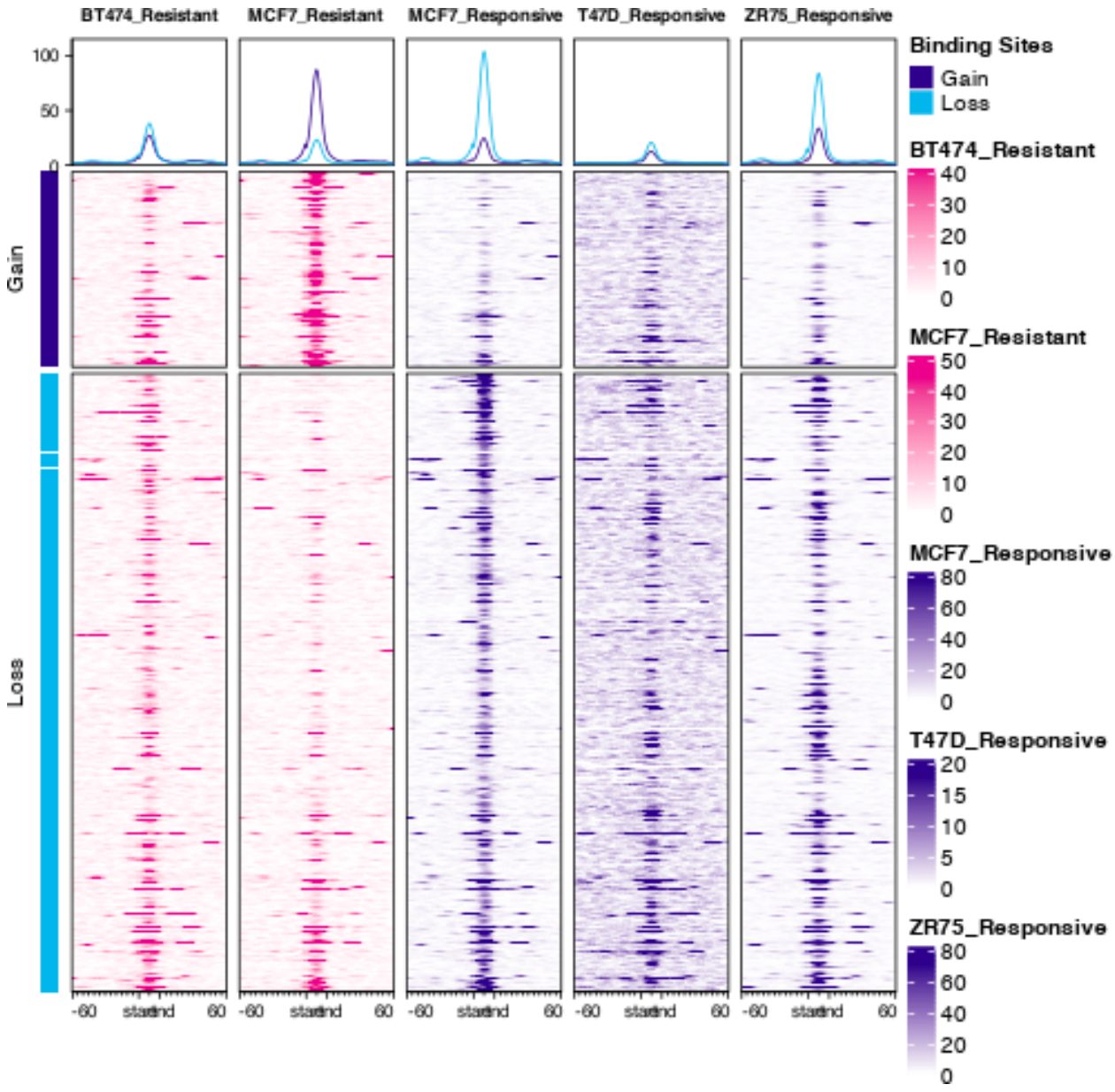
Parameters the control the plotting of the profile heatmaps can be supplied to `dba.plotProfile()`. These are a subset of the parameters associated with the `profileplyer::generateEnrichedHeatmap()` function.

8.1 Plotting parameters: separate scales for each sample (`all_color_scales_equal`)

By default, a single scale is used for all the heatmaps, with the maximum value being the global maximum intensity in any one bin. This can be changed so that every sample uses it's own scale, with the maximum value being the maximum intensity in any one bin *for that sample*. This is accomplished by setting the `generateEnrichedHeatmap` parameter `all_color_scales_equal`:

```
dba.plotProfile(profiles, all_color_scales_equal=FALSE)
```

```
## Plotting...
```



Note that while the scale is changed for the actual heatmaps, a single global scale is still used for the composite profile curves
at the top of the plot.

Other plotting parameters may also be changed; see the help page for`generateEnrichedHeatmap`.