`MMDiff`: Statistical Testing for ChIP-Seq data sets

Gabriele Schweikert
`G.Schweikert@ed.ac.uk`
University of Edinburgh, UK

1 October 2012

1 Introduction

ChIP-Seq has rapidly become the dominant experimental technique to determine the location of transcription factor binding sites and histone modifications. Typically, computational peak finders, such as Macs (**?**), are used to identify potential candidate regions, i.e. regions with significantly enriched read coverage compared to some background. In the following, we will simply call these regions *peaks* and will assume that their *genomic coordinates* are provided. Going beyond this basic analysis, it is often of interest to detect a subset of peaks where significant *changes of read coverage* occur in a treatment experiment relative to a control (see Figure **??**). Statistical analysis of ChIP-Seq data however remains challenging, due to the highly structured nature of the data and the paucity of replicates. Current approaches to detect differentially bound regions are mainly borrowed from RNA-Seq data analysis, thus focusing on total counts of fragments mapped to a region, ignoring any information encoded in the shape of the peak profile.

Higher order features of ChIP-Seq peak enrichment profiles carry important and often complementary information to total counts, and hence are potentially important in assessing differential binding. We therefore incorporate higher order information into testing for differential binding by adapting recently proposed kernel-based statistical tests to ChIP-Seq data.

2 Analysis pipeline

Suppose we have performed an experiment to study the binding of a certain transcription factor or the occurrence of a certain histone modification. We obtained $n_{Samples1}$ ChIP-Seq data sets for a control group and $n_{Samples2}$ ChIP-Seq data sets for a treatment group. On each of the sets we have run a peak finder, e.g. Macs (**?**), to determine the regions of read enrichment, i.e. binding sites. Eventually, we came up with a set of n_{Peaks} regions, which we will examine across all data sets. The task now will be to determine the set of peaks that have significantly changed read coverage between the control group and the treatment group. A change in this context may correspond to a difference in overall binding (i.e. a change in the total number of reads mapping to the region after

normalisation) and/or to a change that affects the shape of the peak. The first kind of changes can be detected with packages like DiffBind (**??**). Here, we will be most interested in the second type of changes. However, this package is compatible with DiffBind and results can easily be compared.

As an example we use data from **?**. It examines the role of CxxC finger protein 1 (Cfp1) in the establishment of Trimethylation of histone H3 Lys 4 (H3K4me3). The experiment consists of H3K4me3 ChIP-Seq measurements from three different cell lines: (1) a wild-type mouse embryonic stem cell line (WT), (2) a mutant line lacking the protein Cfp1 (Null) and (3) a rescue cell line where Cfp1 was re-inserted into the genome (Resc). Cfp1 is known to be a conserved DNA-binding subunit of the H3K4 histone methyltransferase Set1 complex. It is therefore expected that H3K4me3 is reduced in the Null cells. However, as the H3K4 histone methyltransferase activity is redundantly encoded in at least six different complexes in mammals, the precise target regions of Cfp1 are unknown. In addition, under the assumption that the different enzymes potentially act cooperatively at the same target regions, it is to be expected that binding is not completely abolished at these regions but rather reduced, potentially leading to altered H3K4me3 peak profiles. In the rescue cell line 'normal' H3K4me3 levels should be observed and it will therefore be treated as a replicate of the WT cell lines.

We have aligned the short reads from the ChIP-Seq experiments using BWA (**?**), creating bam files. Subsequently, we run Macs (**?**), on each bam file creating Peak lists containing the coordinates of enriched regions. The complete bam files are available as part of ArrayExpress Experiment E-ERAD-79. To run the following examples, we provide subsets of the BAM files with reads mapping to chr1:3000000...75000000 in the data package MMDiffBamSubset. Due to space limitation these files include only one replicate for each cell line (WT, Resc, Null) as well as a control input sample. The package MMDiffBamSubset also contains a corresponding subset of the called peaks. In analogy to DiffBind, information on the experiment is stored in a csv sample sheet. In the case of our example, it is called Cfp1.csv (Table **??**) and it is also available with the provided data package. Among other information the sample sheet contains for each sample the path to the sample bam files and to the peak files.

Table 1: Cfp1 dataset sample sheet (Cfp1.csv).

SampleID	Tissue	Factor	Condition	Replicate	bamReads	bamControl	Peaks	PeakCaller
WT.AB2	WT	H3K4me3	1	2	reads/WT_2.bam	reads/Input.bam	peaks/WT_2_Macs.xls	macs
Null.AB2	Null	H3K4me3	2	2	reads/Null_2.bam	reads/Input.bam	peaks/Null_2_Macs.xls	macs
Resc.AB2	Resc	H3K4me3	1	2	reads/Resc_2.bam	reads/Input.bam	peaks/Resc_2_Macs.xls	macs

Our analysis pipeline consists of 5 steps, which will be described in more detail in the following subsections.

1. **Building histograms**
2. **Outlier detection and normalisation**
3. **Computing of distances between histograms**
4. **Determination of p values**

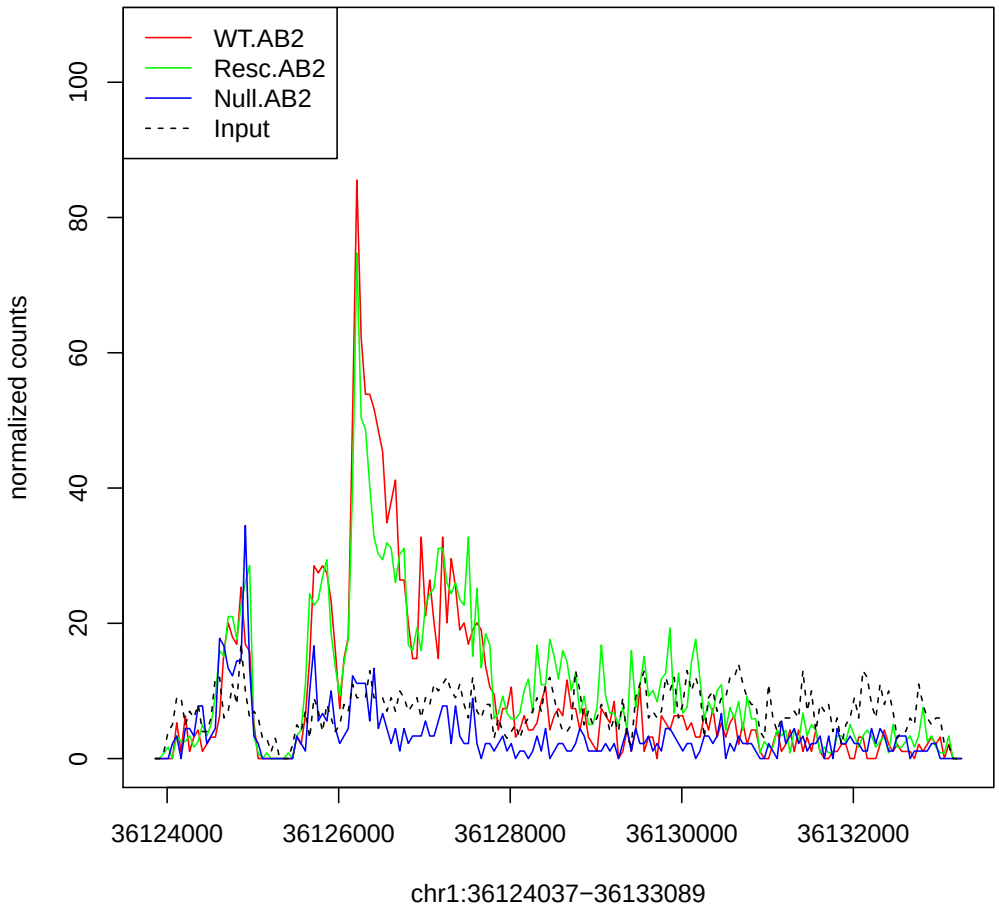

Figure 1: Example Peak: H3K4me3 read enrichment profiles at chromosome 1 :36124037-36133089. The Null sample shows strong signal depletion between 36126000 and 36128000 when compared to the WT and Rescue samples.

5. **Analysis of results**

2.1 Step 1: Building histograms

The experiment details along with the provided peak sets detected by Macs are read in using the following DiffBind function:

```
> library('MMDiff')
> library('MMDiffBamSubset')
> oldwd <- setwd(system.file("extdata", package="MMDiffBamSubset"))
> Cfp1 <- dba(sampleSheet="Cfp1.csv", minOverlap=3)
```

As a result, an S3 object (class DBA) is created. This class is defined in the DiffBindpackage and additional information can be found in the DiffBindvignette. An overview over methods for this class can be viewed by typing `?summary.DBA`. This DBA object will form the basic data structure for the following analysis. However, we will extend the original DBA object by an element called `MD` (see below). Note, that we called `dba()` such that a consensus peak set is created consisting of regions which occur in at least `minOverlap=3` samples. Therefore out of a total of 7370 peaks 2466 regions are selected:

```
> Cfp1

3 Samples, 2468 sites in matrix (7372 total):
        ID Tissue  Factor Condition Replicate Caller Intervals
1   WT.AB2     WT H3K4me3         1         2   macs      4976
2 Null.AB2   Null H3K4me3         2         2   macs      4976
3 Resc.AB2   Resc H3K4me3         1         2   macs      4976
```

Next, we will specify the regions of interest, i.e. the peaks, with a "GRanges" Object. For this example we will examine the first 1000 consensus peaks:

```
> Peaks <- dba.peakset(Cfp1,bRetrieve=TRUE)
> Peaks <- Peaks[1:1000]
```

Note, that we don't have to use the consensus peaks but we could chose an arbitrary set of regions, for example, these 200 100bp consecutive regions on chromosome 1:

```
> Peaks.2 <- GRanges(seqnames = Rle('chr1'),
+                    ranges = IRanges(start=seq(3200000,3219900,100),
+                                     width=100))
```

We now want to create read count profiles across each peak for each sample. To do so, we call the MMDiff function getPeakProfiles(), which uses Rsamtools to collect reads mapping to a certain region in a bam file. Additionally, strand shifts between forward and reverse strand are determined and corrected for and reads in each region are binned (default bin.length = 20bp). As a result, histograms are obtained for each sample and peak.

To correct for the strand shift, only peaks are selected whose total number of reads mapping to each strand is in the 9th decile. If draw.on=TRUE, a plot is generated for each bam file (all bamRead and bamControl files), showing smoothscatter plots of total number of reads mapping to the peaks on forward vs reverse strand, see Figure **??**. This can be used as a quality control (Points should lie on the diagonal). The peaks used to determine the strand shift are shown in red. For each of the selected peaks the shift between forward and reverse strand is determined using the cross-correlation function ccf. If draw.on=TRUE, histograms are plotted for each bam file, showing the distribution of shifts (Figure **??**). The median is used to correct all reads mapping to any peak in the respective bam file. (Note, the shifts can vary between samples i.e. different bam files.)

```
> Cfp1Profiles <- getPeakProfiles(Cfp1, Peaks,
+                                 bin.length=50,
+                                 save.files=FALSE,
+                                 run.parallel=FALSE)
> setwd(oldwd)
```

This object is available for loading using data(Cfp1Profiles). The original DBA object has been extended by an element called MD, which is a list containing elements such as RawTotalCounts and PeakRawHists:

```
> names(Cfp1Profiles$MD)
```

```
[1] "PeakRawHists"   "RawTotalCounts"
```

RawTotalCounts is a simple matrix ($n_{Samples}$ x n_{Peaks}) containing the total number of reads found in the corresponding bam file mapping to a given peak.

PeakRawHists is a list with one element per peak. The name of the elements are the peak coordinates.

```
> PeakRawHists <- Cfp1Profiles$MD$PeakRawHists
> names(PeakRawHists)[1:10]
```

```
 [1] "chr1:3140628-3141907" "chr1:3334413-3335494" "chr1:3659128-3663465"
 [4] "chr1:3842066-3843358" "chr1:3945970-3946781" "chr1:3970025-3973719"
 [7] "chr1:3976887-3978372" "chr1:3984014-3985682" "chr1:4074813-4076118"
[10] "chr1:4139636-4141895"
```

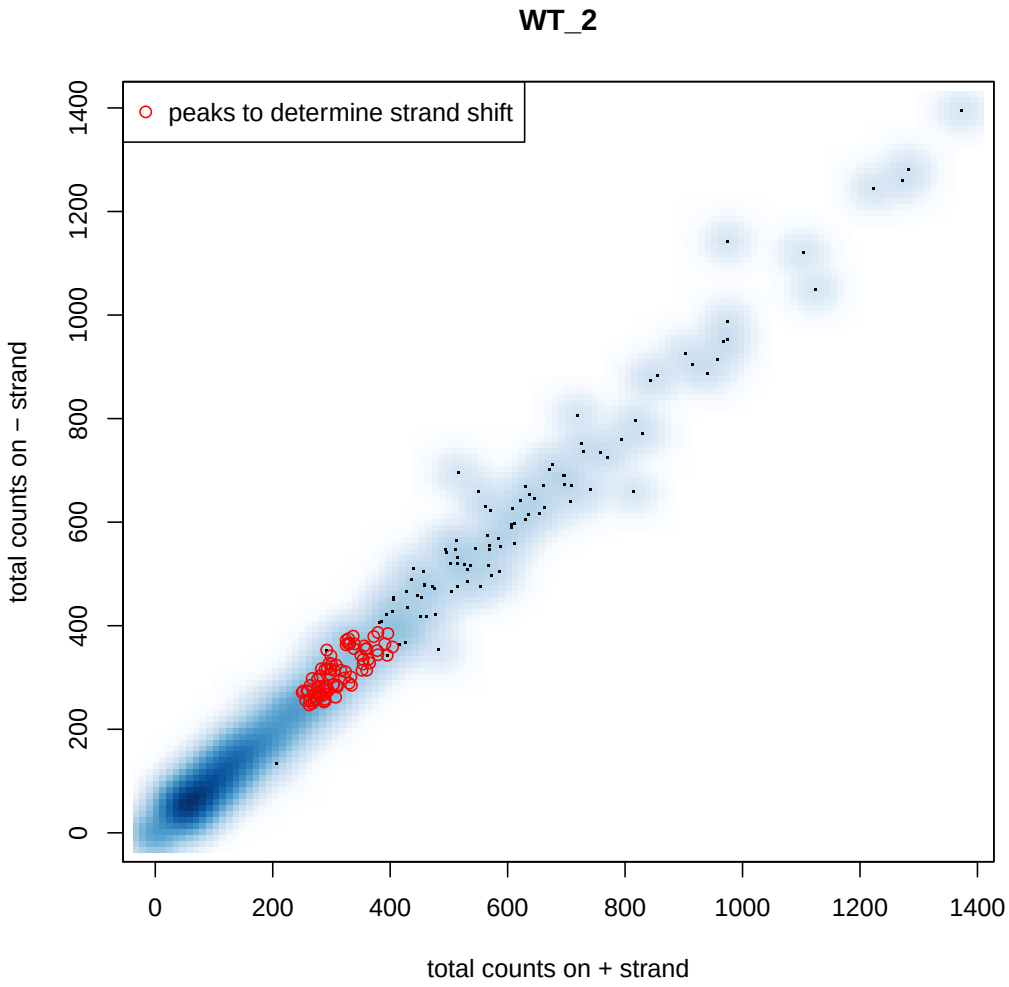

Figure 2: Smooth scatter plot of total number of reads mapping to the peaks on forward vs reverse strand. The red circles indicate peaks used to determine the strand shift.

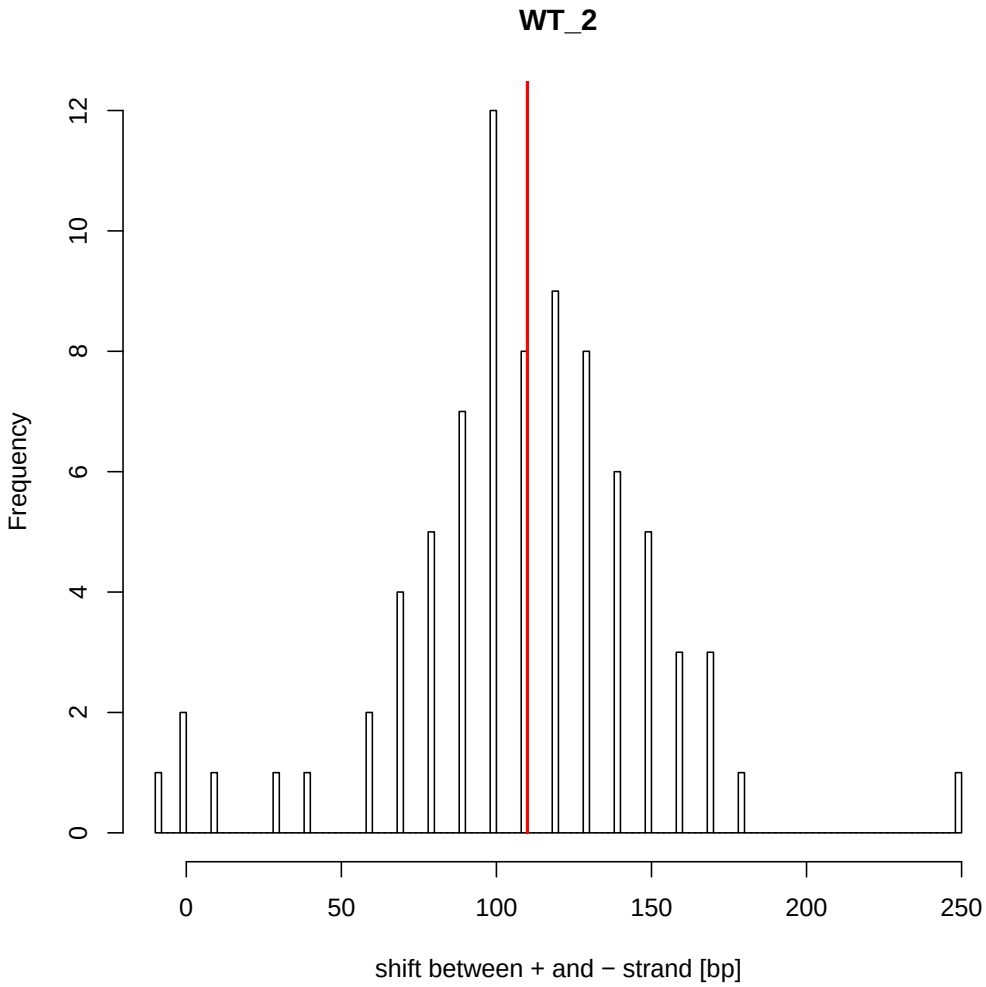

Figure 3: Histogram of determined strand shifts. Red line indicates the median which is used for correction.

For each peak, a matrix can be accessed which contains histograms for each sample, e.g. for peak 5 ("chr1:3842066-3843373"), this is a 4 by 24 matrix, containing counts on 24 bins for 4 samples (WT, Resc, Null, Input):

```
> peak.id <- 5
> dim(PeakRawHists[[peak.id]])

[1]  4 24
```

The column names of the matrix correspond to histogram mid points (genomic coordinates), and the row names signify the bam file from which the histogram was generated. Note that histogram length can vary between different peaks, as the peaks are not necessary of the same length.

```
> colnames(PeakRawHists[[peak.id]])

 [1] "3945795" "3945845" "3945895" "3945945" "3945995" "3946045" "3946095"
 [8] "3946145" "3946195" "3946245" "3946295" "3946345" "3946395" "3946445"
[15] "3946495" "3946545" "3946595" "3946645" "3946695" "3946745" "3946795"
[22] "3946845" "3946895" "3946945"

> rownames(PeakRawHists[[peak.id]])

[1] "WT_2"   "Null_2" "Resc_2" "Input"
```

Profiles for a given peak (e.g. peak 33) can be plotted like this:

```
> Peak.id <- 33
> Sample.ids <- c("WT.AB2", "Null.AB2", "Resc.AB2")
> plotPeak(Cfp1Profiles, Peak.id, Sample.ids, NormMethod=NULL)
```

Note, additional information can be kept if `getPeakProfiles()` is called with the parameter `keep.extra = TRUE`. In this case an extra element `RawHists` is appended to the list `MD`, which contains a list for each bam file:
For instance:

```
> oldwd <- setwd(system.file("extdata", package="MMDiffBamSubset"))
> Cfp1Profiles2 <- getPeakProfiles(Cfp1, Peaks,
+                                  bin.length=50,
+                                  save.files=FALSE,
+                                  keep.extra=TRUE)
> names(Cfp1Profiles2$MD$RawHists$WT_2)
> setwd(oldwd)
```

where "Counts" contains the histogram counts, "Mids" the histogram mid points, "Counts.p" and "Counts.n" histograms for the forward and reverse strand histograms, respectively. "Meta" contains meta information such as the applied shift and bin length.

2.2 Step 2: Normalisation

So far we have looked at raw read counts. However, each sample may have been sequenced to a different depth and in order to make samples comparable, they have to be normalised. Different normalisation strategies have been suggested, here we use the method proposed by **?**: Briefly, a size factor estimate for data set s is computed as the median of the ratios of the s-th data set's counts to those of a pseudo-reference obtained by taking the geometric mean across data sets. Note, that under certain situations only a subset of samples should be normalised with respect to each other (e.g. when groups of the samples have different signal-to-noise ratios, one might want to analyse the groups independently.). In this case we can specify which samples should be used. It is also possible to specify a subset of peaks that are used to determine the normalisation factors. For example, it might be necessary to determine outliers with extreme total counts to be excluded. The function `findOutliers` can be used for this task (see the example in the man page).

```
> SampleIDs <- c("WT.AB2", "Null.AB2", "Resc.AB2")
> Cfp1Norm <- getNormFactors(Cfp1Profiles,
+                            method = "DESeq",
+                            SampleIDs=SampleIDs)

[1] "WT.AB2"   "Null.AB2" "Resc.AB2"
$`WT.AB2,Null.AB2,Resc.AB2`
     WT_2    Null_2    Resc_2
0.9439729 0.8910581 1.1986156
```

The estimated factors can be accessed as:

```
> Cfp1Norm$MD$NormFactors$DESeq

$`WT.AB2,Null.AB2,Resc.AB2`
     WT_2    Null_2    Resc_2
0.9439729 0.8910581 1.1986156
```

Additionally, NormTotalCounts is appended which contains normalised total counts (RawTotalCounts/NormFactors) for all specified samples (all others are set to 0).
Note, that histograms (e.g. `PeakRawHists`) are not normalised and normalisation factors have to be specified in the subsequent steps.
To plot normalised peaks, e.g. peak 33, use:

```
> Peak.id <- 33
> Sample.ids <- c("WT.AB2", "Null.AB2", "Resc.AB2")
> plotPeak(Cfp1Norm, Peak.id, Sample.ids, NormMethod='DESeq')
```

2.3 Step 3: Determine distances between histograms

Next, we want to determine a distance measure for each peak comparing WT, Resc, and Null, i.e. for each peak we need the three pairwise distances: 'WT vs Null', 'WT vs Resc' and 'Resc vs Null'.

When running the `MMDiff`function `compHistDists()`, we can specify which 'distance measure' we want to compute. We recommend the default method 'MMD' (**?**), which computes the maximum mean discrepancy between pairs of histograms (**?**). Alternatively, `method=GMD` (Generalized Minimum Distances) (**?**) can be used, which we will use here, because it is faster:

```
> Cfp1Dists <- compHistDists(Cfp1Norm, method='GMD',
+    overWrite=FALSE, NormMethod='DESeq')
```

We could also specify the comparisons directly, for example:

```
> SampleIDs <- Cfp1Norm$samples$SampleID
> ID <- which(upper.tri(matrix(1, length(SampleIDs), length(SampleIDs)),
+                       diag = F), arr.ind=T)
> CompIDs <- rbind(SampleIDs[ID[,1]], SampleIDs[ID[,2]])
> CompIDs

     [,1]       [,2]       [,3]
[1,] "WT.AB2"   "WT.AB2"   "Null.AB2"
[2,] "Null.AB2" "Resc.AB2" "Resc.AB2"
```

and run the function `compHistDists`:

```
> Cfp1Dists <- compHistDists(Cfp1Norm, method='GMD', CompIDs=CompIDs,
+                            overWrite=FALSE, NormMethod='DESeq')
```

As a result, `Cfp1$MD` is appended with a new element called `DISTS`. In `DISTS$GMD` you will find a matrix of dimension (n_{Peaks} x n_{Comps}) containing all pairwise distances for each peak:

```
> Cfp1Dists$MD$DISTS$GMD[1:10,]

                     WT.AB2 vs Null.AB2 WT.AB2 vs Resc.AB2 Null.AB2 vs Resc.AB2
chr1:3140628-3141907          3.4013630          0.5440379            3.3779956
chr1:3334413-3335494          0.8647619          0.5857143            0.5904762
chr1:3659128-3663465          5.9160605          1.7950362            5.1298997
chr1:3842066-3843358          1.4085859          0.5855556            1.1090909
chr1:3945970-3946781          1.3552036          1.1712912            1.0892598
chr1:3970025-3973719          2.5547082          3.7979556            2.5355312
chr1:3976887-3978372          0.7562044          1.5630954            1.2100154
chr1:3984014-3985682          1.5121951          0.8061350            1.5109981
chr1:4074813-4076118          1.1166755          1.2309383            0.6931514
chr1:4139636-4141895          2.1014550          1.2483766            1.7017797
```

We provide `Cfp1Dists`, which was created running `compHistDists` three times, once with each methods.

```
> data(Cfp1Dists)
> names(Cfp1Dists$MD$DISTS)

[1] "MMD"     "GMD"     "Pearson"
```

2.4 Step 4: Determine p-values

Finally we want to detect peaks that are different in a treatment group relative to a control group. In this example, we only have one replicate for the treatment group (Null.AB2) and we will use WT.AB2 and Resc.AB2 as replicates for the control group. We compute empirical p-values pooling peaks with similar mean total counts. The p-values are adjusted for multiple testing with the method by Benjamini and Hochberg (1995).

```
> data(Cfp1Dists)
> group1 <- c("WT.AB2","Resc.AB2")
> group2 <- c("Null.AB2")
> Cfp1Pvals <- detPeakPvals(Cfp1Dists, group1=group1, group2=group2,
+                     name1='Wt/Resc', name2='Null', method='MMD')

  0%   5%  10%  15%  20%  25%  30%  35%  40%  45%  50%  55%  60%  65%  70%  75%
  32   78   96  110  124  139  155  169  191  210  236  264  297  335  385  453
 80%  85%  90%  95% 100%
 520  639  828 1176 3699

> Cfp1Pvals <- detPeakPvals(Cfp1Pvals, group1=group1, group2=group2,
+                     name1='Wt/Resc', name2='Null', method='GMD')

  0%   5%  10%  15%  20%  25%  30%  35%  40%  45%  50%  55%  60%  65%  70%  75%
  32   78   96  110  124  139  155  169  191  210  236  264  297  335  385  453
 80%  85%  90%  95% 100%
 520  639  828 1176 3699

> Cfp1Pvals <- detPeakPvals(Cfp1Pvals, group1=group1, group2=group2,
+                     name1='Wt/Resc', name2='Null', method='Pearson')

  0%   5%  10%  15%  20%  25%  30%  35%  40%  45%  50%  55%  60%  65%  70%  75%
  32   78   96  110  124  139  155  169  191  210  236  264  297  335  385  453
 80%  85%  90%  95% 100%
 520  639  828 1176 3699
```

2.5 Step 5: Analysis results

The indices of peaks with a p-value < 0.05 can be obtained, by:

```
> idxMMD <- which(Cfp1Pvals$MD$Pvals$MMD<0.05)
```

and similarly for the other methods:

```
> idxGMD <- which(Cfp1Pvals$MD$Pvals$GMD<0.05)
> idxPearson <- which(Cfp1Pvals$MD$Pvals$Pearson<0.05)
```

The coordinates of the peaks can be obtained by:

```
> rownames(Cfp1Pvals$MD$Pvals$MMD)[idxMMD[1:10]]
```

```
 [1] "chr1:3140628-3141907" "chr1:3659128-3663465" "chr1:4773572-4777983"
 [4] "chr1:4796535-4801155" "chr1:4844171-4852624" "chr1:5072531-5074975"
 [7] "chr1:5666675-5668852" "chr1:5905892-5908412" "chr1:6000694-6004040"
[10] "chr1:6906272-6908003"
```

For sanity checks, computed distances can be plotted as a function of mean total count (Figure ??- ??):

```
> group1 <- c("WT.AB2", "Resc.AB2")
> group2 <- c("Null.AB2")
> plotHistDists(Cfp1Pvals, group1=group1, group2=group2, method='MMD')
> plotHistDists(Cfp1Pvals, group1=group1, group2=group2, method='GMD')
> plotHistDists(Cfp1Pvals, group1=group1, group2=group2, method='Pearson')
```

Note, that with MMD, one can observe a number of peaks which have higher distances in the between group comparison than any of the with-in group distances. If we assume that the within group-distances give us a good estimate of naturally occurring variation between peaks under the same condition, these are likely to be peaks that are truly affected when Cfp1 is knocked out. In this example, MMD therefore seems to be best suited to detect target regions of Cfp1.

3 Acknowledgements

The package was developed at the University of Edinburgh, School of Informatics and the Wellcome Trust Centre for Cell Biology with support from Guido Sanguinetti and other members of his group. The data used to illustrate the usage of the package was kindly provided by Thomas Clouaire and Adrian Bird. System administration and computing support was provided by Shaun Webb and Alastair Kerr. Arthur Gretton was very helpful in discussing MMD-based statistical testing and Rory Stark gave valuable insights into ChIP-Seq data analysis and his package DiffBind.

The research leading to these results has received funding from the People Program (Marie Curie Actions) of the European Union's Seventh Framework Programme (FP7/2007-2013) under REA grant agreement number n^o PIEF-GA-2011-299192.

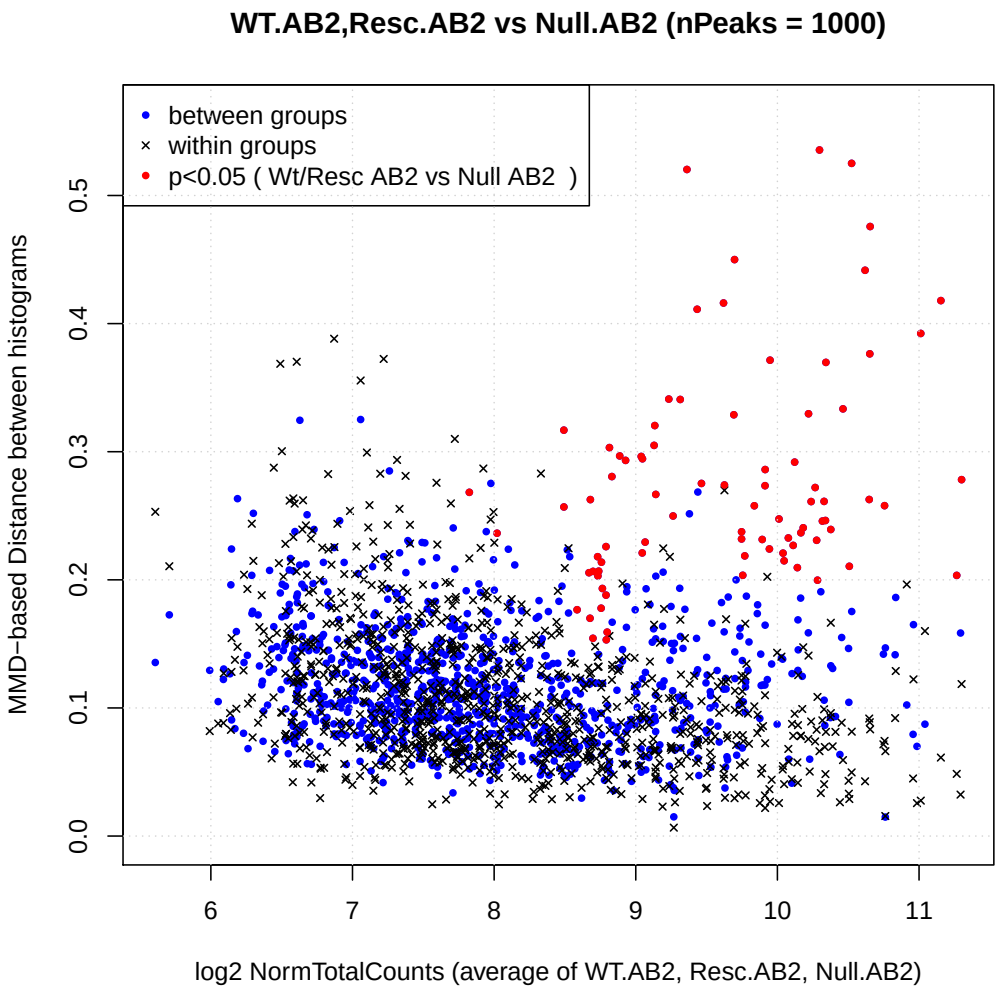

Figure 4: MMD based distances as a function of averaged normalised total counts. Shown are between-group distances (i.e. Wt/Resc vs Null), where each black cross corresponds to a peak. In addition, within-group distances (i.e. between WT and Resc) are overlayed (blue dots). Differentially called peaks with large enough between-group distances are additionally labelled with red circles.

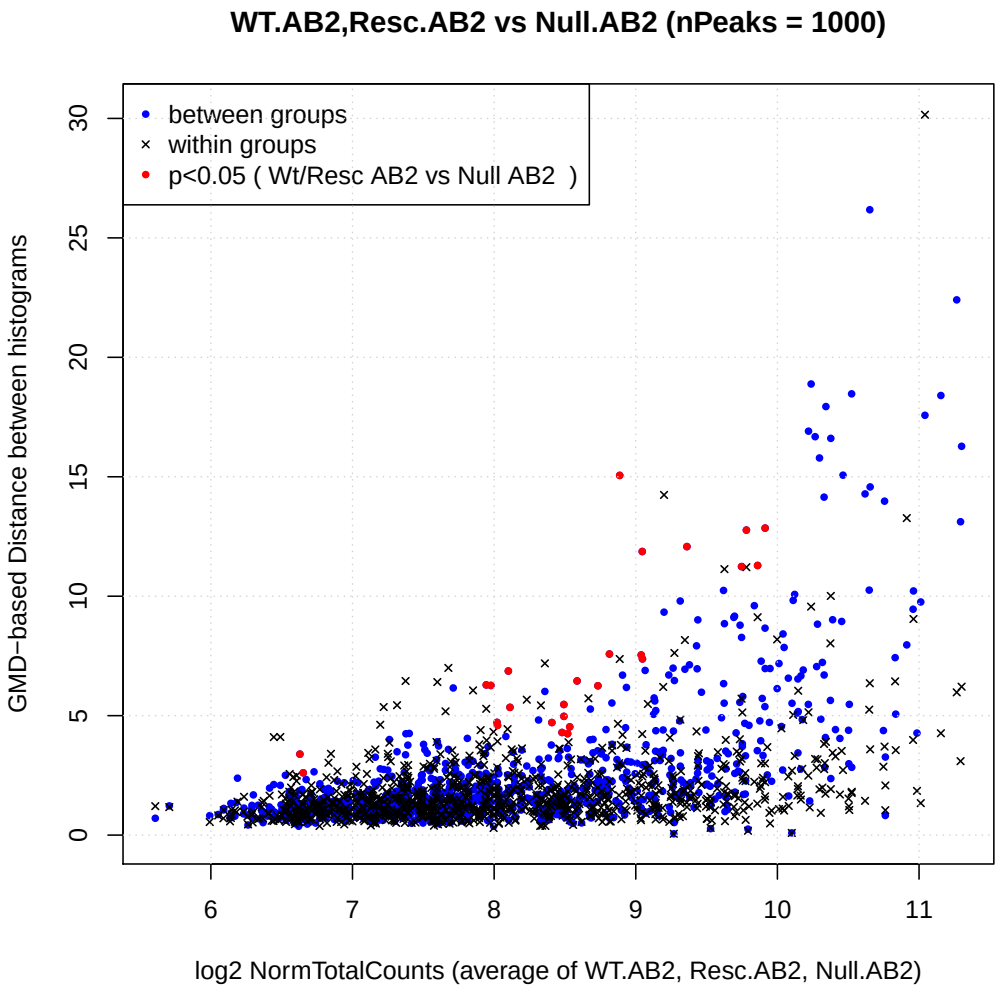

Figure 5: GMD based distances as a function of averaged normalised total counts.

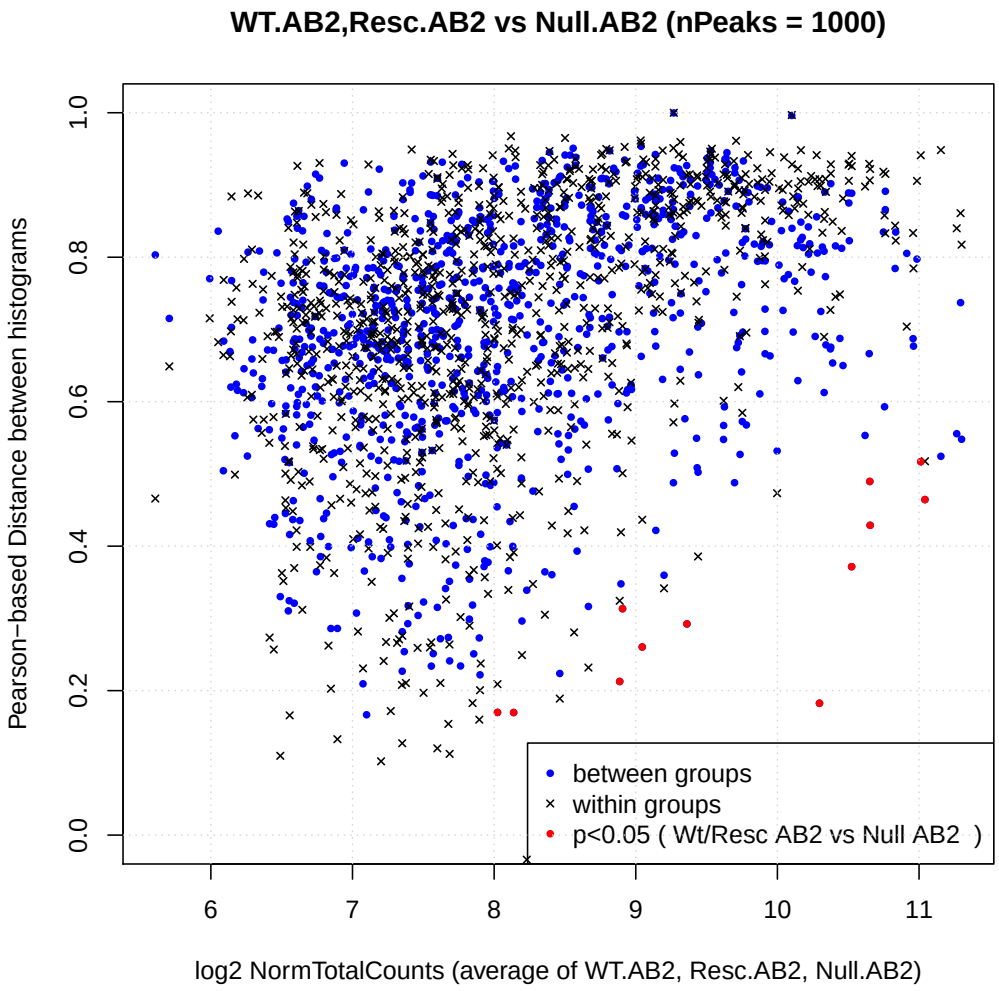

Figure 6: Pearson correlation as a function of averaged normalised total counts

4 Setup

This vignette was built on:

```
> sessionInfo()

R version 3.3.0 RC (2016-04-26 r70550)
Platform: x86_64-apple-darwin13.4.0 (64-bit)
Running under: OS X 10.9.5 (Mavericks)

locale:
[1] C/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

attached base packages:
[1] stats4    parallel  stats     graphics  grDevices utils     datasets
[8] methods   base

other attached packages:
 [1] MMDiffBamSubset_1.7.0      MMDiff_1.12.0
 [3] GMD_0.3.3                  DiffBind_2.0.0
 [5] locfit_1.5-9.1             GenomicAlignments_1.8.0
 [7] Rsamtools_1.24.0           Biostrings_2.40.0
 [9] XVector_0.12.0             limma_3.28.0
[11] SummarizedExperiment_1.2.0 Biobase_2.32.0
[13] GenomicRanges_1.24.0       GenomeInfoDb_1.8.0
[15] IRanges_2.6.0              S4Vectors_0.10.0
[17] BiocGenerics_0.18.0

loaded via a namespace (and not attached):
 [1] Rcpp_0.12.4.5          lattice_0.20-33        GO.db_3.3.0
 [4] gtools_3.5.0           assertthat_0.1         digest_0.6.9
 [7] R6_2.1.2               plyr_1.8.3             BatchJobs_1.6
[10] backports_1.0.2        ShortRead_1.30.0       RSQLite_1.0.0
[13] ggplot2_2.1.0          gplots_3.0.1           zlibbioc_1.18.0
[16] GenomicFeatures_1.24.0 annotate_1.50.0        gdata_2.17.0
[19] Matrix_1.2-6           checkmate_1.7.4        systemPipeR_1.6.0
[22] GOstats_2.38.0         splines_3.3.0          BiocParallel_1.6.0
[25] stringr_1.0.0          pheatmap_1.0.8         RCurl_1.95-4.8
[28] biomaRt_2.28.0         munsell_0.4.3          sendmailR_1.2-1
[31] rtracklayer_1.32.0     base64enc_0.1-3        BBmisc_1.9
[34] fail_1.3               edgeR_3.14.0           XML_3.98-1.4
[37] AnnotationForge_1.14.0 dplyr_0.4.3            bitops_1.0-6
[40] grid_3.3.0             RBGL_1.48.0            xtable_1.8-2
```

```
[43] GSEABase_1.34.0        gtable_0.2.0           DBI_0.4
[46] magrittr_1.5           scales_0.4.0           graph_1.50.0
[49] KernSmooth_2.23-15     amap_0.8-14            stringi_1.0-1
[52] hwriter_1.3.2          genefilter_1.54.0      latticeExtra_0.6-28
[55] brew_1.0-6             rjson_0.2.15           RColorBrewer_1.1-2
[58] tools_3.3.0            Category_2.38.0        survival_2.39-2
[61] AnnotationDbi_1.34.0   colorspace_1.2-6       caTools_1.17.1
```