SOFTWARE TOOL ARTICLE

REVISED From reads to regions: a Bioconductor workflow to detect differential binding in ChIP-seq data [version 2; referees: 2 approved, 1 approved with reservations]

Aaron T. L. Lun[1,2], Gordon K. Smyth[1,3]

[1]The Walter and Eliza Hall Institute of Medical Research, Melbourne, Australia
[2]Department of Medical Biology, The University of Melbourne, Melbourne, Australia
[3]Department of Mathematics and Statistics, The University of Melbourne, Melbourne, Australia

v2 **First published:** 16 Oct 2015, **4**:1080 (doi: 10.12688/f1000research.7016.1)
Latest published: 11 Jan 2016, **4**:1080 (doi: 10.12688/f1000research.7016.2)

Abstract

Chromatin immunoprecipitation with massively parallel sequencing (ChIP-seq) is widely used to identify the genomic binding sites for protein of interest. Most conventional approaches to ChIP-seq data analysis involve the detection of the absolute presence (or absence) of a binding site. However, an alternative strategy is to identify changes in the binding intensity between two biological conditions, i.e., differential binding (DB). This may yield more relevant results than conventional analyses, as changes in binding can be associated with the biological difference being investigated. The aim of this article is to facilitate the implementation of DB analyses, by comprehensively describing a computational workflow for the detection of DB regions from ChIP-seq data. The workflow is based primarily on R software packages from the open-source Bioconductor project and covers all steps of the analysis pipeline, from alignment of read sequences to interpretation and visualization of putative DB regions. In particular, detection of DB regions will be conducted using the counts for sliding windows from the csaw package, with statistical modelling performed using methods in the edgeR package. Analyses will be demonstrated on real histone mark and transcription factor data sets. This will provide readers with practical usage examples that can be applied in their own studies.

This article is included in the Bioconductor channel.

Open Peer Review

Referee Status: ✓ ? ✓

	Invited Referees 1	2	3
REVISED version 2 published 11 Jan 2016			
version 1 published 16 Oct 2015	✓ report	? report	✓ report

1 **Lihua Julie Zhu**, University of Massachusetts Medical School USA, **Jianhong Ou**, University of Massachusetts Medical School USA

2 **Cenny Taslim**, Ohio State University Medical Center USA

3 **Rory Stark**, University of Cambridge UK

Discuss this article

Comments (0)

Corresponding authors: Aaron T. L. Lun (alun@wehi.edu.au), Gordon K. Smyth (smyth@wehi.edu.au)

How to cite this article: Lun ATL and Smyth GK. **From reads to regions: a Bioconductor workflow to detect differential binding in ChIP-seq data [version 2; referees: 2 approved, 1 approved with reservations]** *F1000Research* 2016, **4**:1080 (doi: 10.12688/f1000research.7016.2)

Grant information: National Health and Medical Research Council (Program Grant 1054618 to G.K.S., Fellowship to G.K.S.); Victorian State Government Operational Infrastructure Support; Australian Government NHMRC IRIIS.
The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Competing interests: No competing interests were disclosed.

First published: 16 Oct 2015, **4**:1080 (doi: 10.12688/f1000research.7016.1)

REVISED **Amendments from Version 1**

This revision contains minor changes to the workflow, mostly involving additional elaboration in text. This includes suggestions for the number of replicates in the experimental design, guidelines for interpreting the mapping statistics and BCV plots, explanations of the choices for some parameter settings (in particular, mapping quality thresholds, spacings and bin sizes for the CBP data set) and additional notes about alternative strategies for blacklisting and annotation. In the code blocks, calls to the 'normalize' function were updated to 'normOffsets' to reflect changes in the latest version of the csaw package. Calls to 'align' were similarly updated to make use of new options in the latest version of the Rsubread package. Figure 2 was modified to include "log-CPM" in the x-axis label for clarification. An additional figure (Figure 12) was added upon a request from a reviewer and contains a MDS plot for the CBP data set.

See referee reports

Introduction

Chromatin immunoprecipitation with sequencing (ChIP-seq) is a popular technique for identifying the genomic binding sites of a target protein. Conventional analyses of ChIP-seq data aim to detect absolute binding (i.e., the presence or absence of a binding site) based on peaks in the read coverage. However, a number of recent studies have focused on the detection of changes in the binding profile between conditions (Pal *et al.*, 2013; Ross-Innes *et al.*, 2012). These differential binding (DB) analyses involve counting reads into genomic intervals, and then testing those counts for significant differences between conditions. This defines a set of putative DB regions for further examination. DB analyses are easier to perform than their conventional counterparts, as the effect of genomic biases is largely mitigated when counts for different libraries are compared at the same genomic region. DB regions may also be more relevant as the change in binding can be associated with the biological difference between conditions.

The key step in the DB analysis is the manner in which reads are counted. The most obvious strategy is to count reads into pre-defined regions of interest, like promoters or gene bodies (Pal *et al.*, 2013). This is simple but will not capture changes outside of those regions. In contrast, *de novo* analyses do not depend on pre-specified regions, instead using empirically defined peaks or sliding windows for read counting. Peak-based methods are implemented in the *DiffBind* and *DBChIP* software packages (Liang & Keles, 2012; Ross-Innes *et al.*, 2012), which count reads into peak intervals that have been identified with software like MACS (Zhang *et al.*, 2008). This requires some care to maintain statistical rigour, as peaks are called with the same data used to test for DB. Alternatively, window-based approaches count reads into sliding windows across the genome. This is a more direct strategy that avoids problems with data re-use and can provide increased DB detection power (Lun & Smyth, 2014). However, its correct implementation is not straightforward due to the subtleties with interpretation of the false discovery rate (FDR).

This article describes a computational workflow for performing a DB analysis with sliding windows. The aim is to facilitate the practical implementation of window-based DB analyses by providing detailed code and expected output. The workflow described here applies to any ChIP-seq experiment with multiple experimental conditions and with multiple biological samples within one or more of the conditions. It detects and summarizes DB regions between conditions in a *de novo* manner, i.e., without making any prior assumptions about the location or width of bound regions. Detected regions are then annotated according to their proximity to annotated genes. In addition, the code can be easily adapted to accommodate batch effects, covariates and multiple experimental factors.

The workflow is based primarily on software packages from the open-source Bioconductor project (Huber *et al.*, 2015). It contains all steps that are necessary for detecting DB regions, starting from the raw read sequences. Reads are first aligned to the genome using the *Rsubread* package (Liao *et al.*, 2013). These are counted into sliding windows with *csaw*, to quantify binding intensity across the genome (Lun & Smyth, 2014; Lun & Smyth, 2015). Statistical modelling is based on the negative binomial (NB) distribution with generalized linear models (GLMs) in the *edgeR* package (McCarthy *et al.*, 2012; Robinson *et al.*, 2010), with additional sophistication provided by quasi-likelihood (QL) methods (Lund *et al.*, 2012). Code is also provided for filtering, normalization and region-level control of the FDR. Finally, annotation and visualization of the DB regions is described using *Gviz* and other packages.

The application of the methods in this article will be demonstrated on two publicly available ChIP-seq data sets. The first data set studies changes in H3K9ac marking between pro-B and mature B cells (Revilla-I-Domingo *et al.*, 2012). The second data set studies changes in CREB-binding protein (CBP) binding between wild-type and CBP knock-out cells (Kasper *et al.*, 2014). These two studies were chosen to represent common situations where a DB analysis can be applied – one involving sharp binding with CBP, and the other involving broader marking with H3K9ac. A separate

workflow is described for the analysis of each data set, using the sliding window approach in both cases but with different parameter settings. The intention is to provide readers with a variety of usage examples from which they can construct DB analyses of their own data.

Aligning reads in the H3K9ac libraries

The first task is to download the relevant ChIP-seq libraries from the NCBI Gene Expression Omnibus (GEO) (Edgar *et al.*, 2002). These are obtained from the data series GSE38046, using the Sequence Read Accession (SRA) numbers listed below. The experiment contains two biological replicates in total for each of the two cell types, i.e., pro-B and mature B. Multiple technical replicates exist for some of the biological replicates, and are indicated as those files with the same `grouping`.

```
sra.numbers <- c("SRR499718", "SRR499719", "SRR499720", "SRR499721",
    "SRR499734", "SRR499735", "SRR499736", "SRR499737", "SRR499738")
grouping <- c("proB-8113", "proB-8113", "proB-8108", "proB-8108",
    "matureB-8059", "matureB-8059", "matureB-8059", "matureB-8059", "matureB-8086")
all.sra <- paste0(sra.numbers, ".lite.sra")
data.frame(SRA=all.sra, Condition=grouping)
```

```
##                   SRA    Condition
## 1 SRR499718.lite.sra    proB-8113
## 2 SRR499719.lite.sra    proB-8113
## 3 SRR499720.lite.sra    proB-8108
## 4 SRR499721.lite.sra    proB-8108
## 5 SRR499734.lite.sra matureB-8059
## 6 SRR499735.lite.sra matureB-8059
## 7 SRR499736.lite.sra matureB-8059
## 8 SRR499737.lite.sra matureB-8059
## 9 SRR499738.lite.sra matureB-8086
```

These files are downloaded in the SRA format, and need to be unpacked to the FASTQ format prior to alignment. This can be done using the `fastq-dump` utility from the SRA Toolkit.

```
for (sra in all.sra) {
    code <- system(paste("fastq-dump", sra))
    stopifnot(code==0L)
}
all.fastq <- paste0(sra.numbers, ".fastq")
```

Reads from technical replicates are pooled together into a single FASTQ file prior to further processing. This reflects the fact that they originate from a single library of DNA fragments.

```
by.group <- split(all.fastq, grouping)
for (group in names(by.group)) {
    code <- system(paste(c("cat", by.group[[group]], ">",
        paste0(group, ".fastq")), collapse=" "))
    stopifnot(code==0L)
}
group.fastq <- paste0(names(by.group), ".fastq")
```

Reads in each library are aligned to the mm10 build of the mouse genome, using the `align` function in the *Rsubread* package (Liao *et al.*, 2013). This assumes that an index has already been constructed with the prefix `index/mm10`. The function uses a seed-and-vote paradigm to quickly and accurately map reads to the genome by focusing on locations that receive a number of votes above some consensus threshold. Here, a threshold of 2 votes is used instead of the default of 3, to accommodate the short length of the reads (32–36 bp). The `type` parameter is also set to optimize for genomic alignment, rather than alignment to the transcriptome.

```
library(Rsubread)
bam.files <- paste0(names(by.group), ".bam")
align(index="index/mm10", readfile1=group.fastq, TH1=2, type=1,
    input_format="FASTQ", output_file=bam.files)
```

In each of the resulting BAM files, alignments are re-sorted by their mapping locations. This is required for input into *csaw*, but is also useful for other programs like genome browsers that depend on sorting and indexing for rapid retrieval of reads.

```
library(Rsamtools)
for (bam in bam.files) {
    out <- suppressWarnings(sortBam(bam, "h3k9ac_temp"))
    file.rename(out, bam)
}
```

Potential PCR duplicates are marked using the `MarkDuplicates` tool from the Picard software suite. These are identified as alignments at the same genomic location, such that they may have originated from PCR-amplified copies of the same DNA fragment.

```
temp.bam <- "h3k9ac_temp.bam"
temp.file <- "h3k9ac_metric.txt"
temp.dir <- "h3k9ac_working"
dir.create(temp.dir)
for (bam in bam.files) {
    code <- system(sprintf("MarkDuplicates I=%s O=%s M=%s \\
        TMP_DIR=%s AS=true REMOVE_DUPLICATES=false \\
        VALIDATION_STRINGENCY=SILENT", bam, temp.bam,
        temp.file, temp.dir))
    stopifnot(code==0L)
    file.rename(temp.bam, bam)
}
```

The behaviour of the alignment pipeline for this data set can be easily summarized with some statistics. Ideally, the proportion of mapped reads should be high (70–80% or higher), while the proportion of marked reads should be low (below 20%). Note that only reads with unique mapping locations are reported by *Rsubread* as being successfully mapped.

```
diagnostics <- list()
for (bam in bam.files) {
    total <- countBam(bam)$records
    mapped <- countBam(bam, param=ScanBamParam(
        flag=scanBamFlag(isUnmapped=FALSE)))$records
    marked <- countBam(bam, param=ScanBamParam(
        flag=scanBamFlag(isUnmapped=FALSE, isDuplicate=TRUE)))$records
    diagnostics[[bam]] <- c(Total=total, Mapped=mapped, Marked=marked)
}
diag.stats <- data.frame(do.call(rbind, diagnostics))
diag.stats$Prop.mapped <- diag.stats$Mapped/diag.stats$Total*100
diag.stats$Prop.marked <- diag.stats$Marked/diag.stats$Mapped*100
diag.stats
```

```
##                     Total  Mapped  Marked Prop.mapped Prop.marked
## matureB-8059.bam 16675372 7752077 1054591    46.48818   13.603980
## matureB-8086.bam  6347683 4899961  195100    77.19291    3.981664
## proB-8108.bam    10413135 8213980  297796    78.88095    3.625478
## proB-8113.bam    10724526 9145743  489177    85.27876    5.348685
```

Finally, the libraries are indexed for rapid retrieval by genomic location. This generates a number of index files at the same location as the BAM files.

```
indexBam(bam.files)
```

Obtaining the ENCODE blacklist for mm10

A number of genomic regions contain high artifactual signal in ChIP-seq experiments. These often correspond to genomic features like telomeres or microsatellite repeats. For example, multiple tandem repeats in the real genome are reported as a single unit in the genome build. Alignment of all (non-specifically immunoprecipitated) reads from the former will result in artificially high coverage of the latter. Moreover, differences in repeat copy numbers between conditions can lead to detection of spurious DB.

As such, these regions must be removed prior to further analysis. This can be done with an annotated blacklist of problematic regions in the mm9 build of the mouse genome. All reads in the blacklist will be ignored during processing in *csaw*. The blacklist itself was constructed by identifying consistently problematic regions in the ENCODE and modENCODE data sets (ENCODE Project Consortium, 2012).

Recall that the alignments have been performed to the mm10 build, so the mm9 blacklist coordinates must be transferred to their mm10 equivalents. This is done using the `liftOver` function in the *rtracklayer* package (Lawrence *et al.*, 2009). The chain file specifies the corresponding coordinates between the two builds and can be obtained here. The new blacklist coordinates are then saved to file for future use.

```
library(rtracklayer)
ch <- import.chain("mm9ToMm10.over.chain")
original <- import("mm9-blacklist.bed")
blacklist <- liftOver(x=original, chain=ch)
blacklist <- unlist(blacklist)
saveRDS(file="mm10-blacklist.rds", blacklist)
```

Any user-defined set of regions can be used as a blacklist in this analysis. For example, one could use predicted repeat regions from the UCSC genome annotation (Rosenbloom *et al.*, 2015). This tends to remove a greater number of problematic regions (especially microsatellites) compared to the ENCODE blacklist. However, the size of the UCSC list means that genuine DB sites may also be removed. Thus, the ENCODE blacklist is preferred for most applications. Alternatively, if negative control libraries are available, they can be used to empirically identify problematic regions with the *GreyListChIP* package. These regions should be ignored as they have high coverage in the controls and are unlikely to be genuine binding sites.

Testing for DB between pro-B and mature B cells

Setting up the analysis parameters

Here, the settings for the DB analysis are specified. Recall that the paths to the BAM files are stored in the `bam.files` vector after alignment. The cell type for each file can be conveniently extracted from the file name.

```
celltype <- sub("-.*",  "", bam.files)
data.frame(BAM=bam.files,  CellType=celltype)
```

```
##                BAM  CellType
## 1 matureB-8059.bam   matureB
## 2 matureB-8086.bam   matureB
## 3    proB-8108.bam      proB
## 4    proB-8113.bam      proB
```

In the *csaw* package, the `readParam` object determines which reads are extracted from the BAM files. The idea is to set this up once and to re-use it in all relevant functions. For this analysis, reads are only used if they have a mapping quality (MAPQ) score equal to or above 50. This avoids spurious results due to weak or non-unique alignments. While a MAPQ threshold of 50 is quite conservative, a stringent threshold is necessary here due to the short length of the reads. Reads are also ignored if they map within blacklist regions or if they do not map to the standard set of mouse nuclear chromosomes.

```
library(csaw)
standard.chr <- paste0("chr", c(1:19, "X", "Y"))
param <- readParam(minq=50, discard=blacklist, restrict=standard.chr)
```

Computing the average fragment length

Strand bimodality is often observed in ChIP-seq experiments involving narrow binding events like H3K9ac marking. This refers to the presence of distinct subpeaks on each strand and can be quantified with cross-correlation plots (Kharchenko *et al.*, 2008). A strong peak in the cross-correlations should be observed if immunoprecipitation was successful. The delay distance at the peak corresponds to the distance between forward-/reverse-strand subpeaks. This is identified from Figure 1 and is used as the average fragment length for this analysis.

```
x <- correlateReads(bam.files, param=reform(param, dedup=TRUE))
frag.len <- which.max(x) - 1
frag.len
```

```
## [1] 148
```

```
plot(1:length(x)-1, x, xlab="Delay (bp)", ylab="CCF", type="l")
abline(v=frag.len, col="red")
text(x=frag.len, y=min(x), paste(frag.len, "bp"), pos=4, col="red")
```

Only unmarked reads (i.e., not potential PCR duplicates) are used here. This tends to give better signal by reducing the size of the "phantom" peak at the read length (Landt *et al.*, 2012). However, removal of marked reads is risky as it caps the signal in high-coverage regions of the genome. This can result in loss of power to detect DB, or introduction of spurious DB when the same cap is applied to libraries of different sizes. Thus, the marking status of each read will be ignored in the rest of the analysis, i.e., no duplicates will be removed in downstream steps.

Counting reads into windows

csaw uses a sliding window strategy to quantify binding intensity across the genome. Each read is directionally extended to the average fragment length, to represent the DNA fragment from which that read was sequenced. The number of extended reads overlapping a window is counted. The window is then moved to its next position on the genome, and counting is repeated. (Each read is usually counted into multiple windows, which will introduce correlations between adjacent windows but will not otherwise affect the analysis.) This is done for all libraries such that a count is obtained for each window in each library. The `windowCounts` function produces a `RangedSummarizedExperiment` object containing these counts in matrix form, where each row corresponds to a window and each column represents a library.

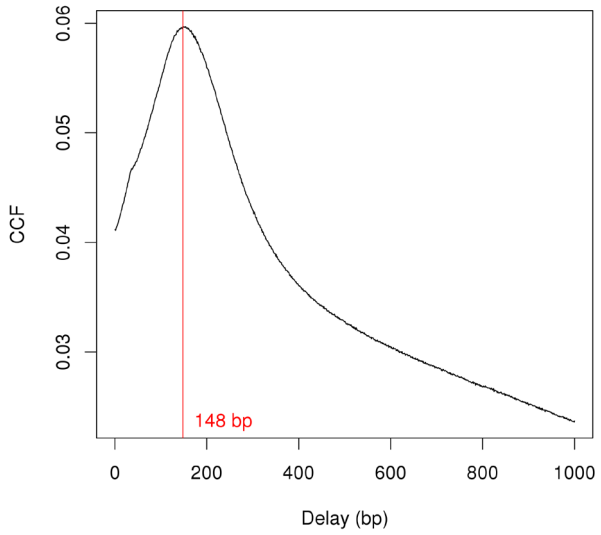

Figure 1. Cross-correlation function (CCF) against delay distance for the H3k9ac data set. The delay with the maximum correlation is shown as the red line.

```
win.data <- windowCounts(bam.files, param=param, width=150, ext=frag.len)
win.data
```

```
## class: RangedSummarizedExperiment
## dim: 1569624 4
## metadata(4): spacing width shift final.ext
## assays(1): counts
## rownames: NULL
## rowRanges metadata column names(0):
## colnames: NULL
## colData names(4): bam.files totals ext param
```

To analyze H3K9ac data, a window size of 150 bp is used here. This corresponds roughly to the length of the DNA in a nucleosome (Humburg *et al.*, 2011), which is the smallest relevant unit for studying histone mark enrichment. The spacing between windows is set to the default of 50 bp, i.e., the start positions for adjacent windows are 50 bp apart. Smaller spacings can be used to improve spatial resolution, but will increase memory usage and runtime by increasing the number of windows required to cover the genome. This is unnecessary as increased resolution confers little practical benefit for this data set – counts for very closely spaced windows will be practically identical. Finally, windows with very low counts (by default, less than a sum of 10 across all libraries) are removed to reduce memory usage. This represents a preliminary filter to remove uninteresting windows corresponding to likely background regions.

Filtering windows by abundance

As previously mentioned, low-abundance windows contain no binding sites and need to be filtered out. This improves power by removing irrelevant tests prior to the multiple testing correction; avoids problems with discreteness in downstream statistical methods; and reduces computational work for further analyses. Here, filtering is performed using the average abundance of each window (McCarthy *et al.*, 2012), which is defined as the average log-count per million for that window. This performs well as an independent filter statistic for NB-distributed count data (Lun & Smyth, 2014).

The filter threshold is defined based on the assumption that most regions in the genome are not marked by H3K9ac. Reads are counted into large bins and the median coverage across those bins is used as an estimate of the background abundance. This estimate is then compared to the average abundances of the windows, after rescaling to account for differences in the window and bin sizes. A window is only retained if its coverage is 3-fold higher than that of the background regions, i.e., the abundance of the window is greater than the background abundance estimate by $\log_2(3)$ or more. This removes a large number of windows that are weakly or not marked and are likely to be irrelevant.

```
bins <- windowCounts(bam.files, bin=TRUE, width=2000, param=param)
filter.stat <- filterWindows(win.data,  bins, type="global")
min.fc <- 3
keep <- filter.stat$filter > log2(min.fc)
summary(keep)
```

```
##     Mode    FALSE    TRUE    NA's
##  logical   906406  663218       0
```

The effect of the fold-change threshold can be examined visually in Figure 2. The chosen threshold is greater than the abundances of most bins in the genome – presumably, those that contain background regions. This suggests that the filter will remove most windows lying within background regions.

```
hist(filter.stat$back.abundances, main="", breaks=50,
    xlab="Background abundance (log2-CPM)")
threshold <- filter.stat$abundances[1] - filter.stat$filter[1] + log2(min.fc)
abline(v=threshold, col="red")
```

The actual filtering itself is done by simply subsetting the `RangedSummarizedExperiment` object.

```
filtered.data <- win.data[keep,]
```

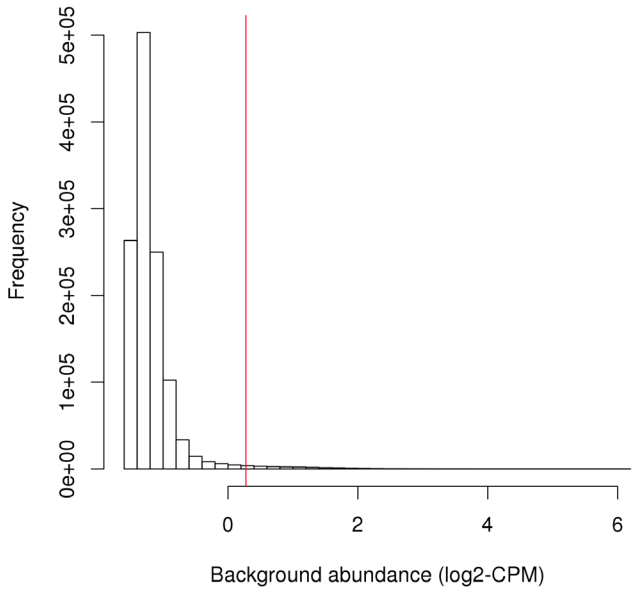

Figure 2. Histogram of average abundances across all 2 kbp genomic bins. The filter threshold is shown as the red line.

Normalizing for library-specific trended biases

Normalization is required to eliminate confounding library-specific biases prior to any comparisons between libraries. In particular, a trended bias is often observed between libraries in Figure 3. This refers to a systematic fold-difference in window coverage between libraries that changes according to the average abundance of the window.

```
win.ab <- filter.stat$abundances[keep]
adjc <- log2(assay(filtered.data)+0.5)
logfc <- adjc[,1] - adjc[,4]
smoothScatter(win.ab, logfc, ylim=c(-6, 6), xlim=c(0, 5),
     xlab="Average abundance", ylab="Log-fold change")
```

Trended biases cannot be removed by scaling methods like TMM normalization (Robinson & Oshlack, 2010), as the amount of scaling required varies with the abundance of the window. Rather, non-linear normalization methods must be used. *csaw* implements a version of the fast loess method (Ballman *et al.*, 2004) that has been modified to handle count data (Lun & Smyth, 2015). This produces a matrix of offsets that can be used during GLM fitting.

```
offsets <- normOffsets(filtered.data, type="loess")
head(offsets)
```

```
##             [,1]       [,2]      [,3]      [,4]
## [1,] -0.5878496 -0.4019382 0.3954267 0.5943611
## [2,] -0.5673338 -0.3789731 0.3770978 0.5692091
## [3,] -0.6261679 -0.4720746 0.4397909 0.6584516
## [4,] -0.6528790 -0.5453416 0.4789700 0.7192507
## [5,] -0.6713098 -0.5838111 0.5015881 0.7535328
## [6,] -0.7028331 -0.6463783 0.5390876 0.8101237
```

The effect of non-linear normalization can be visualized with a mean-difference plot comparing the first and last libraries. Once the offsets are applied to adjust the log-fold changes, the trend is eliminated from the plot (Figure 4). The cloud of points is also centred at a log-fold change of zero. This indicates that normalization was successful in removing the differences between libraries.

```
norm.adjc <- adjc - offsets/log(2)
norm.fc <- norm.adjc[,1]-norm.adjc[,4]
smoothScatter(win.ab, norm.fc, ylim=c(-6, 6), xlim=c(0, 5),
     xlab="Average abundance", ylab="Log-fold change")
```

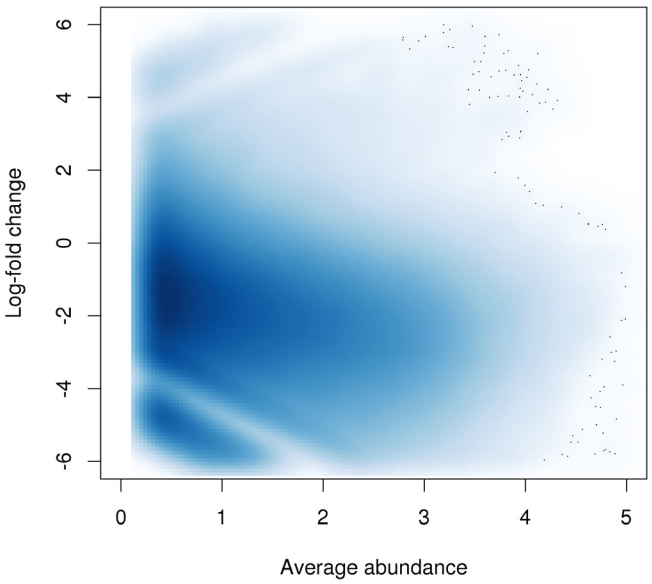

Figure 3. Abundance-dependent trend in the log-fold change between two H3K9ac libraries (mature B over pro-B), across all windows retained after filtering.

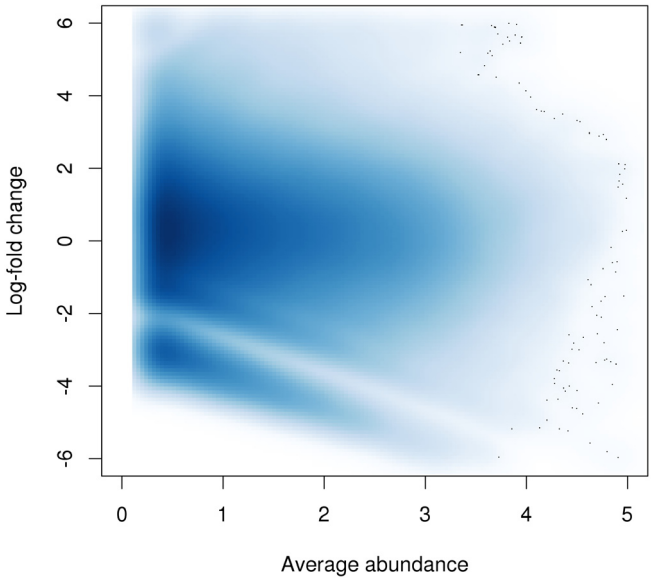

Figure 4. Effect of non-linear normalization on the trended bias between two H3K9ac libraries. Normalized log-fold changes are shown for all windows retained after filtering.

The implicit assumption of non-linear methods is that most windows at each abundance are not DB. Any systematic difference between libraries is attributed to bias and is removed. The assumption of a non-DB majority is reasonable for this data set, given that the cell types being compared are quite closely related. However, it is not appropriate in situations where large-scale DB is expected, as removal of the difference would result in loss of genuine DB. An alternative normalization strategy for these situations will be described later in the CBP analysis.

Statistical modelling of biological variability

Introduction. Counts are modelled using NB GLMs in the *edgeR* package (McCarthy *et al.*, 2012; Robinson *et al.*, 2010). The NB distribution is useful as it can handle low, discrete counts for each window. The NB dispersion parameter allows modelling of biological variability between replicate libraries. GLMs can also accommodate complex experimental designs, though a simple design is sufficient for this study.

```
celltype <- factor(celltype)
design <- model.matrix(~0+celltype)
colnames(design) <- levels(celltype)
design
```

```
##    matureB  proB
## 1        1     0
## 2        1     0
## 3        0     1
## 4        0     1
## attr(,"assign")
## [1] 1 1
## attr(,"contrasts")
## attr(,"contrasts")$celltype
## [1] "contr.treatment"
```

As a general rule, the experimental design should contain at least two replicates in each of the biological conditions. This ensures that the results for each condition are replicable and are not the result of technical artifacts such as PCR duplicates. Obviously, more replicates will provide more power to detect DB accurately and reliability, albeit at the cost of time and experimental resources.

Estimating the NB dispersion. The `RangedSummarizedExperiment` object is coerced into a `DGEList` object (plus offsets) prior to entry into *edgeR*. Estimation of the NB dispersion is then performed. Specifically, a NB dispersion trend is fitted to all windows against the average abundance. This means that empirical mean-dispersion trends can be flexibly modelled.

```
library(edgeR)
y <- asDGEList(filtered.data)
y$offset <- offsets
y <- estimateDisp(y, design)
summary(y$trended.dispersion)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##  0.03156  0.04174  0.04274  0.04168  0.04311  0.04371
```

The NB dispersion trend is visualized in Figure 5 as the biological coefficient of variation (BCV), i.e., the square root of the NB dispersion. Note that only the trended dispersion will be used in the downstream steps – the common and tagwise values are only shown for diagnostic purposes. Specifically, the common BCV provides an overall measure of the variability in the data set, averaged across all windows. Data sets with common BCVs ranging from 10 to 20% are considered to have low variability, i.e., counts are highly reproducible. The tagwise BCVs should also be dispersed above and below the fitted trend, indicating that the fit was successful.

```
plotBCV(y)
```

For most data sets, one would expect to see a trend that decreases to a plateau with increasing average abundance. This reflects the greater reliability of large counts, where the effects of stochasticity and technical artifacts (e.g., mapping errors, PCR duplicates) are averaged out. In Figure 5, the range of abundances after filtering is such that the plateau has already been reached. This is still a satisfactory result, as it indicates that the retained windows have low variability and more power to detect DB.

Estimating the QL dispersion. Additional modelling is provided with the QL methods in *edgeR* (Lund *et al.*, 2012). This introduces a QL dispersion parameter for each window, which captures variability in the NB dispersion around the fitted trend for each window. Thus, the QL dispersion can model window-specific variability, whereas the NB dispersion trend is averaged across many windows. However, with limited replicates, there is not enough information for each window to stably estimate the QL dispersion. This is overcome by sharing information between windows with empirical Bayes (EB) shrinkage. The instability of the QL dispersion estimates is reduced by squeezing the estimates towards an abundance-dependent trend (Figure 6).

```
fit <- glmQLFit(y, design, robust=TRUE)
plotQLDisp(fit)
```

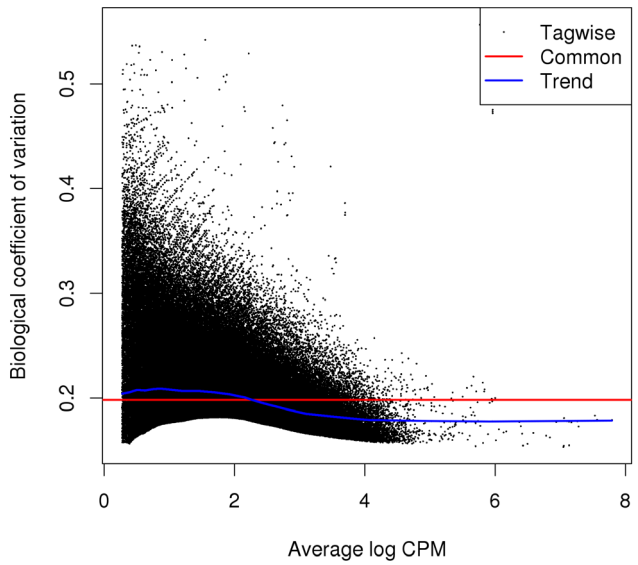

Figure 5. Abundance-dependent trend in the BCV for each window, represented by the blue line. Common (red) and tagwise estimates (black) are also shown.

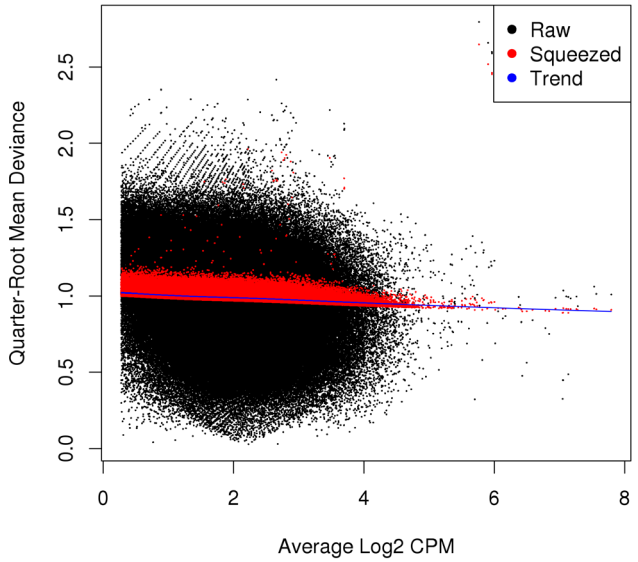

Figure 6. Effect of EB shrinkage on the raw QL dispersion estimate for each window (black) towards the abundance-dependent trend (blue) to obtain squeezed estimates (red).

The extent of shrinkage is determined by the prior degrees of freedom (d.f.). Large prior d.f. indicates that the dispersions were similar across windows, such that strong shrinkage to the trend could be performed to increase stability and power. Small prior d.f. indicates that the dispersions were more variable. In such cases, less squeezing is performed as strong shrinkage would be inappropriate. Also note the use of `robust=TRUE`, which reduces the sensitivity of the EB procedures to outlier windows.

```
summary(fit$df.prior)
```

```
##    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##  0.4903  22.6900  22.6900  22.6900  22.6900  22.6900
```

Examining the data with MDS plots. Multi-dimensional scaling (MDS) plots can be used to examine the similarities between libraries. The distance between a pair of libraries on this plot represents the overall log-fold change between those libraries. Ideally, replicates should cluster together while samples from different conditions should be separate. In Figure 7, strong separation in the first dimension is observed between libraries from different cell types. This indicates that significant differences are likely to be present between cell types in this data set.

```
plotMDS(norm.adjc, labels=celltype,
     col=c("red", "blue")[as.integer(celltype)])
```

Testing for DB and controlling the FDR

Testing for DB with QL F-tests. Each window is tested for significant differences between cell types using the QL F-test (Lund *et al.*, 2012). This is superior to the likelihood ratio test that is typically used for GLMs, as the QL F-test accounts for the uncertainity in dispersion estimation. One *p*-value is produced for each window, representing the evidence against the null hypothesis (i.e., that no DB is present in the window). For this analysis, the comparison is parametrized such that the reported log-fold change for each window represents that of the coverage in pro-B cells over their mature B counterparts.

```
contrast <- makeContrasts(proB-matureB, levels=design)
res <- glmQLFTest(fit, contrast=contrast)
head(res$table)
```

```
##       logFC      logCPM           F       PValue
## 1 0.8071199   0.3987193   0.9350894   0.34292110
## 2 0.7892698   0.3531386   0.8977361   0.35257125
## 3 2.0508458   0.5770295   5.3124205   0.02987028
## 4 1.1952436   0.8317769   2.6800790   0.11429478
## 5 0.9751114   0.9868770   2.0577431   0.16397982
## 6 0.6472745   1.2487216   1.0906847   0.30643720
```

Controlling the FDR across regions. One might attempt to control the FDR by applying the Benjamini-Hochberg (BH) method to the window-level *p*-values (Benjamini & Hochberg, 1995). However, the features of interest are not windows, but the genomic regions that they represent. Control of the FDR across windows does not guarantee control of the FDR across regions (Lun & Smyth, 2014). The latter is arguably more relevant for the final interpretation of the results.

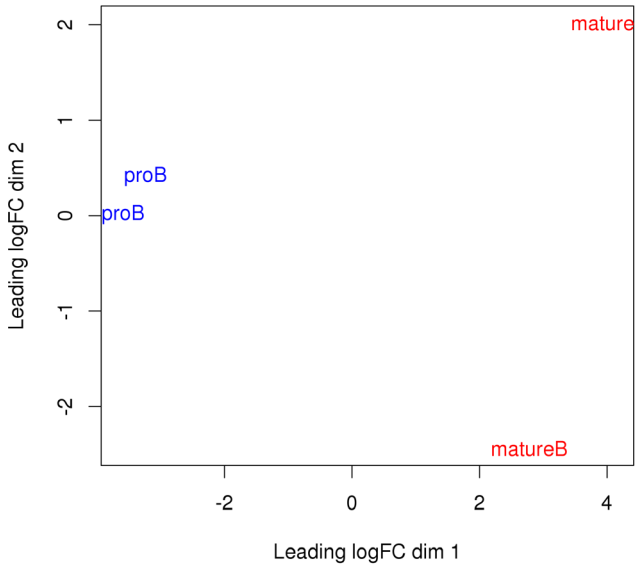

Figure 7. MDS plot with two dimensions for all libraries in the H3K9ac data set. Libraries are labelled and coloured according to the cell type.

Control of the region-level FDR can be provided by aggregating windows into regions and combining the *p*-values. Here, adjacent windows less than 100 bp apart are aggregated into clusters. Each cluster represents a genomic region. Smaller values of `tol` allow distinct marking events to kept separate, while larger values provide a broader perspective, e.g., by considering adjacent co-regulated sites as a single entity. Chaining effects are mitigated by setting a maximum cluster width of 5 kbp.

```
merged <- mergeWindows(rowRanges(filtered.data),tol=100, max.width=5000)
```

A combined *p*-value is computed for each cluster using the method of Simes (1986), based on the *p*-values of the constituent windows. This represents the evidence against the global null hypothesis for each cluster, i.e., that no DB exists in any of its windows. Rejection of this global null indicates that the cluster (and the region that it represents) contains DB. Applying the BH method to the combined *p*-values allows the region-level FDR to be controlled.

```
tabcom <- combineTests(merged$id, res$table)
head(tabcom)
```

```
##    nWindows   logFC.up   logFC.down      PValue        FDR
## 1         2          2            0 0.35257125 0.48433052
## 2        24         10            0 0.03965980 0.09329214
## 3         8          1            3 0.38231836 0.51205966
## 4        11          1            2 0.84578866 0.91987081
## 5        36         14            6 0.01481647 0.04558244
## 6        18          7            9 0.00671594 0.02656732
```

Each row of the output table contains the statistics for a single cluster, including the combined *p*-value before and after the BH correction. The `nWindows` field describes the total number of windows in the cluster. The `logFC.up` and `logFC.down` fields describe the number of windows with a log-fold change above 0.5 or below -0.5 in each cluster, respectively. This can be used to determine the direction of DB in each cluster.

Examining the scope and direction of DB. The total number of DB regions at a FDR of 5% can be easily calculated.

```
is.sig <- tabcom$FDR <= 0.05
summary(is.sig)
```

```
##    Mode   FALSE    TRUE   NA's
## logical   26121   13402       0
```

Determining the direction of DB is more complicated, as clusters could potentially contain windows that are changing in opposite directions. One approach is to define the direction based on the number of windows changing in each direction, as described above. Another approach is to use the log-fold change of the most significant window as a proxy for the log-fold change of the cluster. This is generally satisfactory, though it will not capture multiple changes in opposite directions. It also tends to overstate the change in each cluster.

```
tabbest <- getBestTest(merged$id, res$table)
head(tabbest)
```

```
##   best       logFC    logCPM           F      PValue        FDR
## 1    1  0.8071199 0.3987193  0.9350894 0.68584219 0.89635068
## 2   14  6.4894914 0.7814903 12.3651181 0.04305271 0.10940477
## 3   29 -0.8951569 1.4182105  3.1716524 0.70058621 0.91053977
## 4   42 -0.9100013 0.9724194  2.4590005 1.00000000 1.00000000
## 5   64  6.5014465 0.7867585 14.3069870 0.03337001 0.09138600
## 6   88  6.5134616 0.7920288 15.6865615 0.01067998 0.04156789
```

In the above table, each row contains the statistics for each cluster. Of interest are the `best` and `logFC` fields. The former is the index of the window that is the most significant in each cluster, while the latter is the log-fold change of that window. This can be used to obtain a summary of the direction of DB across all clusters/regions.

```
is.sig.pos <- (tabbest$logFC > 0)[is.sig]
summary(is.sig.pos)
```

```
##    Mode   FALSE    TRUE   NA's
## logical    8137    5265       0
```

Saving results to file

Results can be saved to file prior to further manipulation. One approach is to store all statistics in the metadata of a `GRanges` object. This is useful as it keeps the statistics and coordinates together for each cluster, avoiding problems with synchronization in downstream steps. The midpoint and log-fold change of the best window are also stored.

```
out.ranges <- merged$region
elementMetadata(out.ranges) <- data.frame(tabcom,
   best.pos=mid(ranges(rowRanges(filtered.data[tabbest$best]))),
   best.logFC=tabbest$logFC)
saveRDS(file="h3k9ac_results.rds", out.ranges)
```

For input into other programs like genome browsers, results can be saved in a more conventional format. Here, coordinates of DB regions are saved in BED format via *rtracklayer*, using a log-transformed FDR as the score.

```
simplified <- out.ranges[is.sig]
simplified$score <- -10*log10(simplified$FDR)
export(con="h3k9ac_results.bed", object=simplified)
```

Saving the `RangedSummarizedExperiment` objects is also recommended. This avoids the need to re-run the time-consuming read counting steps if parts of the analysis need to be repeated. Similarly, the `DGEList` object is saved so that the *edgeR* statistics can be easily recovered.

```
save(file="h3k9ac_objects.Rda", win.data, bins, y)
```

Interpreting the DB results

Adding gene-centric annotation

Using the `detailRanges` function. *csaw* provides its own annotation function, `detailRanges`. This identifies all genic features overlapping each region and reports them in a compact string form. Briefly, features are reported as `SYMBOL|EXONS|STRAND` where `SYMBOL` represents the gene symbol, `EXONS` lists the overlapping exons (`0` for promoters, `I` for introns), and `STRAND` reports the strand. Multiple overlapping features for different genes are separated by commas within the string for each region.

```
library(org.Mm.eg.db)
library(TxDb.Mmusculus.UCSC.mm10.knownGene)
anno <- detailRanges(out.ranges, orgdb=org.Mm.eg.db,
     txdb=TxDb.Mmusculus.UCSC.mm10.knownGene)
head(anno$overlap)
```

```
## [1] "Mrpl15|5|-"       "Mrpl15|0-1|-"   "Lypla1|0|+"     "Lypla1|0,2|+"
## [5] "Tcea1|0-2|+"      "Atp6v1h|0-1|+"
```

Annotated features that flank the region of interest are also reported. The description for each feature is formatted as described above but with an extra `[DISTANCE]` field, representing the distance (in base pairs) between that feature and the region. By default, only flanking features within 5 kbp of each region are considered.

```
head(anno$left)
```

```
## [1] "Mrpl15|6|-[935]"  "Mrpl15|2-3|-[896]" ""
## [4] "Lypla1|1|+[19]"   ""                  ""
```

```
head(anno$right)
```

```
## [1] "Mrpl15|4|-[1875]" ""              "Lypla1|1-2|+[143]"
## [4] ""                 ""              "Atp6v1h|2|+[517]"
```

The annotation for each region can then be stored in metadata of the `GRanges` object. The compact string form is useful for human interpretation, as it allows rapid examination of all genic features neighbouring each region.

```
meta <- elementMetadata(out.ranges)
elementMetadata(out.ranges) <- data.frame(meta, anno)
```

Using the ChIPpeakAnno package. As its name suggests, the *ChIPpeakAnno* package is designed to annotate peaks from ChIP-seq experiments (Zhu *et al.*, 2010). A `GRanges` object containing all regions of interest is supplied to the relevant function after removing all previous metadata fields to reduce clutter. The gene closest to each region is then reported. Gene coordinates are taken from the NCBI mouse 38 annotation, which is roughly equivalent to the annotation in the mm10 genome build.

```
library(ChIPpeakAnno)
data(TSS.mouse.GRCm38)
minimal <- out.ranges
elementMetadata(minimal) <- NULL
anno.regions <- annotatePeakInBatch(minimal, AnnotationData=TSS.mouse.GRCm38)
colnames(elementMetadata(anno.regions))
```

```
## [1] "peak"                      "feature"
## [3] "start_position"            "end_position"
## [5] "feature_strand"            "insideFeature"
## [7] "distancetoFeature"         "shortestDistance"
## [9] "fromOverlappingOrNearest"
```

Alternatively, identification of all overlapping features within, say, 5 kbp can be achieved by setting `maxgap=5000` and `output="overlapping"` in `annotatePeakInBatch`. This will report each overlapping feature in a separate entry of the returned `GRanges` object, i.e., each input region may have multiple output values. In contrast, `detailRanges` will report all overlapping features for a region as a single string, i.e., each input region has one output value. Which is preferable depends on the purpose of the annotation – the `detailRanges` output is more convenient for direct annotation of a DB list, while the `annotatePeakInBatch` output contains more information and is more convenient for further manipulation.

Reporting gene-based results. Another approach to annotation is to flip the problem around, such that DB statistics are reported directly for features of interest like genes. This is more convenient when the DB analysis needs to be integrated with, e.g., DE analyses of matching RNA-seq data. In the code below, promoter coordinates are obtained by running `detailRanges` without specifying any regions. All windows overlapping each promoter are defined as a cluster, and DB statistics are computed as previously described for each cluster/promoter. This directly yields DB results for annotated features, along with some `NA` values representing promoters that have no overlapping windows (these are filtered out in the code below for demonstration purposes).

```
anno.ranges  <- detailRanges(orgdb=org.Mm.eg.db,
     txdb=TxDb.Mmusculus.UCSC.mm10.knownGene)
promoters  <- anno.ranges[anno.ranges$exon==0L]
olap <- findOverlaps(promoters, rowRanges(filtered.data))
tabprom <- combineOverlaps(olap, res$table)
head(data.frame(Gene=promoters$symbol,  tabprom)[!is.na(tabprom$PValue),])
```

```
##       Gene  nWindows  logFC.up  logFC.down         PValue          FDR
## 6   Ldlrap1        19        11           0 0.224741404877 0.2707231899
## 7     Mdn1        29        12          11 0.000004447727 0.0001346831
## 8    Pydc3         8         0           6 0.051183399851 0.0781862855
## 9   Wfdc17         6         0           6 0.000069604922 0.0008739367
## 10  Mfap1b        19         1          10 0.107116609335 0.1441133306
## 13 Gm15772        30        12           7 0.085543435687 0.1193045223
```

Note that this strategy is distinct from counting reads across promoters. Using promoter-level counts would not provide enough spatial resolution to detect sharp binding events that only occur in a subinterval of the promoter. In particular, detection may be compromised by non-specific background or the presence of multiple opposing DB events in the same promoter. Combining window-level statistics is preferable as resolution is maintained for optimal performance.

Visualizing DB results

Overview. Here, the *Gviz* package is used to visualize read coverage across the data set at regions of interest. Coverage in each BAM file will be represented by a single track. Several additional tracks will also be included in each plot. One is the genome axis track, to display the genomic coordinates across the plotted region. The other is the annotation track containing gene models, with gene IDs replaced by symbols (where possible) for easier reading.

```
library(Gviz)
gax <- GenomeAxisTrack(col="black", fontsize=15, size=2)
greg <- GeneRegionTrack(TxDb.Mmusculus.UCSC.mm10.knownGene, showId=TRUE,
     geneSymbol=TRUE, name="", background.title="transparent")
symbols <- unlist(mapIds(org.Mm.eg.db, gene(greg), "SYMBOL",
     "ENTREZID", multiVals = "first"))
symbol(greg) <- symbols[gene(greg)]
```

Simple DB across a broad region. To begin with, the top-ranking DB region will be visualized. This represents a simple DB event where the entire region changes in one direction (Figure 8). Specifically, it represents an increase in H3K9ac marking at the *H2-Aa* locus. This is consistent with the expected biology – H3K9ac is a mark of active gene expression (Karmodiya *et al.*, 2012) and MHCII components are upregulated in mature B cells (Hoffmann *et al.*, 2002).

```
o <- order(out.ranges$PValue)
cur.region <- out.ranges[o[1]]
cur.region
```

```
## GRanges object with 1 range and 10 metadata columns:
##       seqnames                 ranges strand |  nWindows  logFC.up
##          <Rle>              <IRanges>  <Rle> | <integer> <integer>
##   [1]  chr17    [34285101, 34289950]      * |        94         0
##       logFC.down                  PValue              FDR  best.pos
##        <integer>               <numeric>          <numeric> <integer>
##   [1]         94   0.00000000000004471753 0.000000001195279  34287575
##       best.logFC                           overlap             left
##        <numeric>                          <factor>          <factor>
##   [1]  -7.176575 H2-Aa|0-1|-,H2-Eb1|I|+,Notch4|I|+ H2-Aa|2-6|-[278]
##          right
##       <factor>
##   [1]
##   -------
##   seqinfo: 21 sequences from an unspecified genome
```

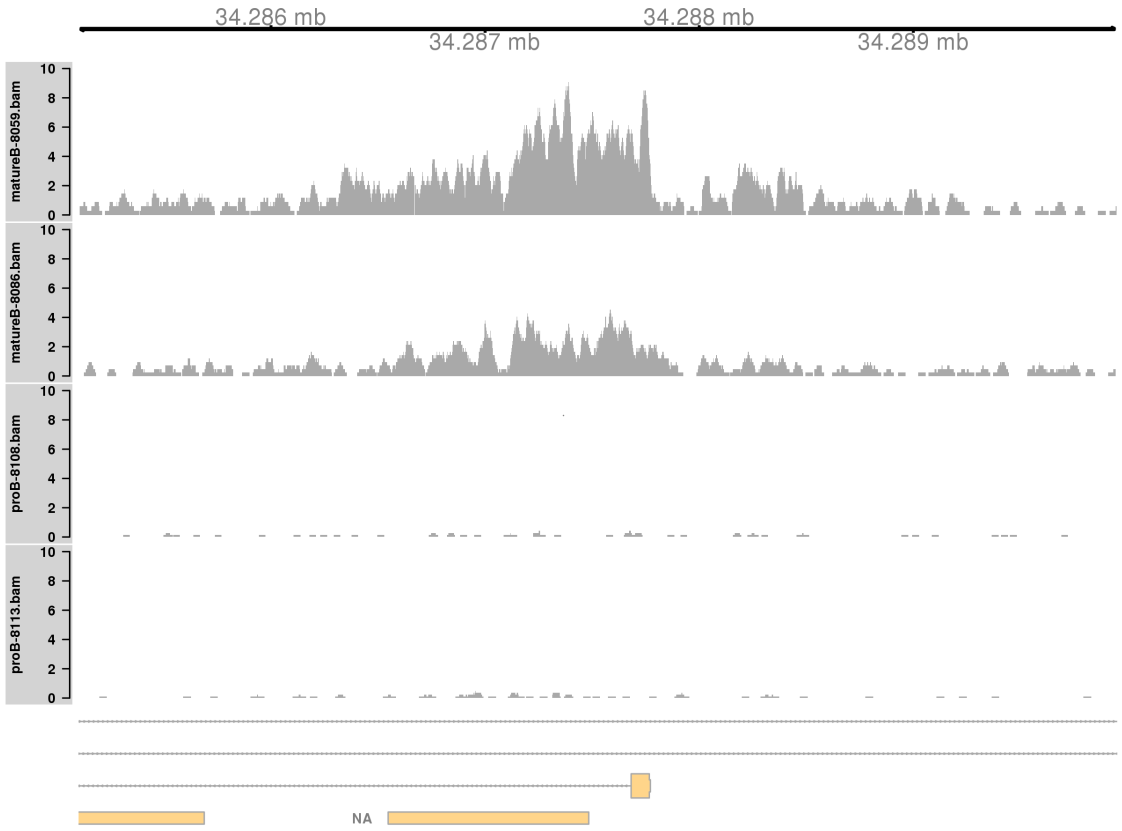

Figure 8. Coverage tracks for a simple DB event between pro-B and mature B cells, across a broad region in the H3K9ac data set. Read coverage for each library is shown as a per-million value at each base.

One track is plotted for each library, in addition to the coordinate and annotation tracks. Coverage is plotted in terms of sequencing depth-per-million at each base. This corrects for differences in library sizes between tracks.

```
collected <- list()
lib.sizes <- filtered.data$totals/1e6
for (i in 1:length(bam.files)) {
    reads <- extractReads(bam.file=bam.files[i], cur.region, param=param)
    cov <- as(coverage(reads)/lib.sizes[i], "GRanges")
    collected[[i]] <- DataTrack(cov, type="histogram", lwd=0, ylim=c(0,10),
        name=bam.files[i],  col.axis="black", col.title="black",
        fill="darkgray", col.histogram=NA)
}
plotTracks(c(gax, collected, greg), chromosome=as.character(seqnames(cur.region)),
    from=start(cur.region), to=end(cur.region))
```

Complex DB across a broad region. Complex DB refers to situations where multiple DB events are occurring within the same enriched region. These are identified as those clusters that contain windows changing in both directions. Here, the second-ranking complex cluster is selected for visualization (the top-ranking complex cluster is adjacent to the region used in the previous example, so another region is chosen for some variety).

```
complex <- out.ranges$logFC.up > 0 & out.ranges$logFC.down > 0
cur.region <- out.ranges[o[complex[o]][2]]
cur.region
```

```
## GRanges object with 1 range and 10 metadata columns:
##      seqnames               ranges strand |  nWindows  logFC.up
##         <Rle>            <IRanges>  <Rle> | <integer> <integer>
##   [1]    chr5 [122987201, 122991450]      * |        83        17
##        logFC.down             PValue              FDR  best.pos best.logFC
##         <integer>          <numeric>        <numeric> <integer>  <numeric>
##   [1]         43 0.0000000002201102 0.0000001962277 122990925  -5.466918
##                              overlap              left
##                             <factor>          <factor>
##   [1] A930024E05Rik|0-1|+,Kdm2b|0-3|- Kdm2b|4-5|-[2661]
##                            right
##                           <factor>
##   [1] A930024E05Rik|2|+[2913]
##   -------
##   seqinfo: 21 sequences from an unspecified genome
```

This region contains a bidirectional promoter where different genes are marked in the different cell types (Figure 9). Upon differentiation to mature B cells, loss of marking in one part of the region is balanced by a gain in marking in another part of the region. This represents a complex DB event that would not be detected if reads were counted across the entire region.

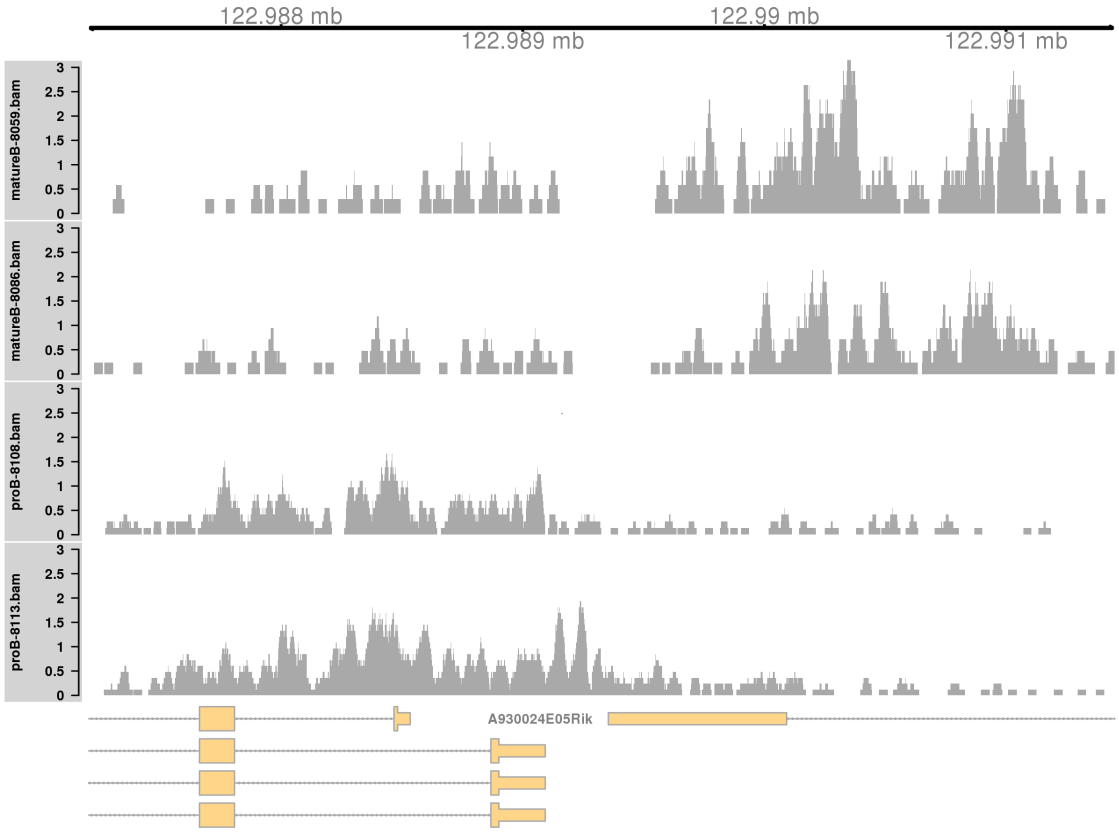

Figure 9. Coverage tracks for a complex DB event in the H3K9ac data set, shown as per-million values.

```
collected <- list()
for (i in 1:length(bam.files)) {
    reads <- extractReads(bam.file=bam.files[i], cur.region, param=param)
    cov <- as(coverage(reads)/lib.sizes[i], "GRanges")
    collected[[i]] <- DataTrack(cov, type="histogram", lwd=0, ylim=c(0,3),
        name=bam.files[i],  col.axis="black", col.title="black",
        fill="darkgray", col.histogram=NA)
}
plotTracks(c(gax, collected, greg), chromosome=as.character(seqnames(cur.region)),
    from=start(cur.region), to=end(cur.region))
```

Simple DB across a small region. Both of the examples above involve differential marking within broad regions spanning several kilobases. This is consistent with changes in the marking profile across a large number of nucleosomes. However, H3K9ac marking can also be concentrated into small regions, involving only a few nucleosomes. *csaw* is equally capable of detecting sharp DB within these small regions. This can be demonstrated by examining those clusters that contain a smaller number of windows.

```
sharp <- out.ranges$nWindows < 20
cur.region <- out.ranges[o[sharp[o]][1]]
cur.region
```

```
## GRanges object with 1 range and 10 metadata columns:
##       seqnames               ranges strand |  nWindows  logFC.up
##          <Rle>            <IRanges>  <Rle> | <integer> <integer>
##   [1]    chr16 [36665551,  36666200]      * |         11         0
##       logFC.down            PValue             FDR  best.pos best.logFC
##        <integer>         <numeric>       <numeric> <integer>  <numeric>
##   [1]         11 0.0000000003412784 0.0000002593913  36665925  -4.887727
##          overlap     left    right
##         <factor> <factor> <factor>
##   [1] Cd86|0-1|-
##   -------
##   seqinfo: 21 sequences from an unspecified genome
```

Marking is increased for mature B cells within a 500 bp region (Figure 10), which is sharper than the changes in the previous two examples. This also coincides with the promoter of the *Cd86* gene. Again, this makes biological sense as CD86 is involved in regulating immunoglobulin production in activated B-cells (Podojil & Sanders, 2003).

```
collected <- list()
for (i in 1:length(bam.files)) {
    reads <- extractReads(bam.file=bam.files[i], cur.region, param=param)
    cov <- as(coverage(reads)/lib.sizes[i], "GRanges")
    collected[[i]] <- DataTrack(cov, type="histogram", lwd=0, ylim=c(0,3),
          name=bam.files[i], col.axis="black", col.title="black",
          fill="darkgray",  col.histogram=NA)
}
plotTracks(c(gax, collected, greg), chromosome=as.character(seqnames(cur.region)),
    from=start(cur.region), to=end(cur.region))
```

Note that the window size will determine whether sharp or broad events are preferentially detected. Larger windows provide more power to detect broad events (as the counts are higher), while smaller windows provide more resolution to detect sharp events. Optimal detection of all features can be obtained by performing analyses with multiple window sizes and consolidating the results, though – for brevity – this will not be described here. In general, smaller window sizes are preferred as strong DB events with sufficient coverage will always be detected. For larger windows, detection may be confounded by other events within the window that distort the log-fold change in the counts between conditions.

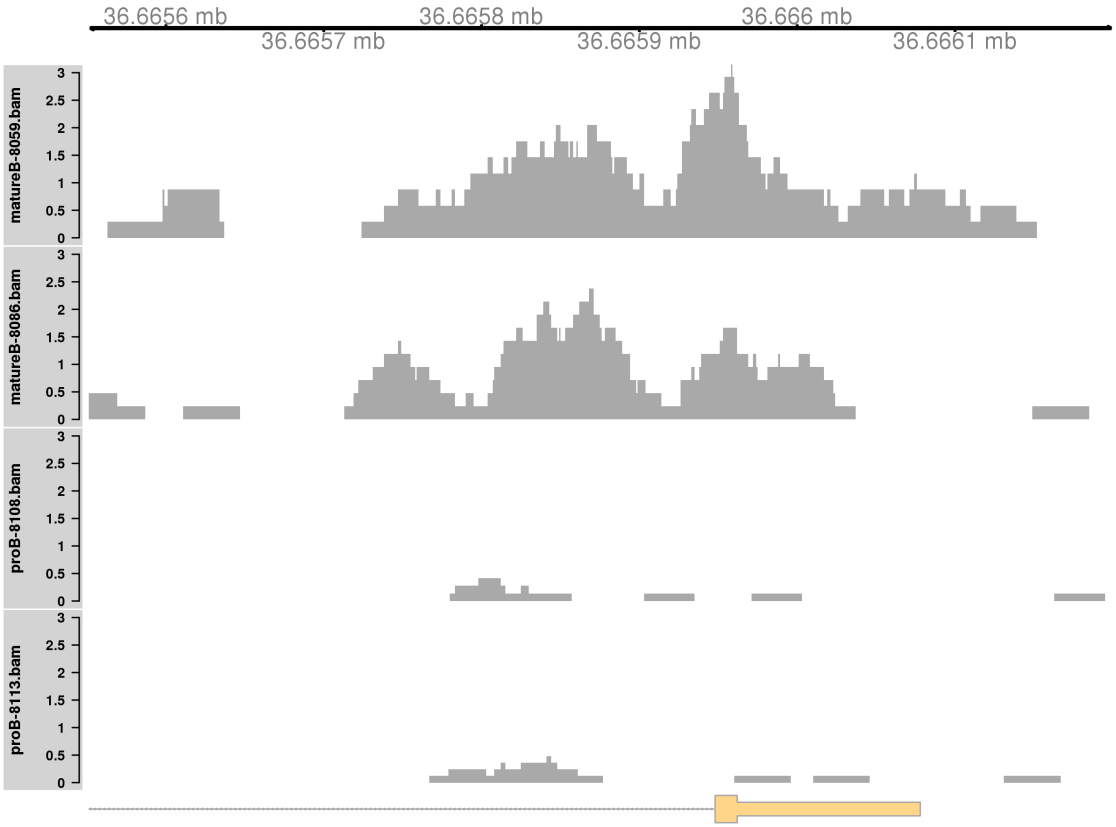

Figure 10. Coverage tracks for a sharp and simple DB event in the H3K9ac data set, shown as per-million values.

Repeating the analysis for the CBP data

Overview

A window-based DB analysis will be shown for transcription factor (TF) data, to complement the histone mark analysis above. This data set compares CBP binding between wild-type (WT) and CBP knock-out (KO) animals (Kasper *et al.*, 2014). The aim is to use *csaw* and other Bioconductor packages to identify DB sites between genotypes. Most, if not all, of these sites should be increased in the WT, given that protein function should be compromised in the KO.

Aligning reads from CBP libraries

Libraries are downloaded from the NCBI GEO data series GSE54453, using the SRA accessions listed below. The data set contains two biological replicates for each of the two genotypes. One file is available for each library, i.e., no technical replicates.

```
sra.numbers <- c("SRR1145787", "SRR1145788", "SRR1145789", "SRR1145790")
genotype <- c("wt", "wt", "ko", "ko")
all.sra <- paste0(sra.numbers, ".sra")
data.frame(SRA=all.sra, Condition=genotype)
```

```
##              SRA  Condition
## 1 SRR1145787.sra         wt
## 2 SRR1145788.sra         wt
## 3 SRR1145789.sra         ko
## 4 SRR1145790.sra         ko
```

SRA files are unpacked to yield FASTQ files with the raw read sequences.

```
for (sra in all.sra) {
    code <- system(paste("fastq-dump", sra))
    stopifnot(code==0L)
}
all.fastq <- paste0(sra.numbers, ".fastq")
```

Reads are aligned to the mm10 genome using *Rsubread*. Here, the default consensus threshold is used as the reads are longer (75 bp). A Phred offset of +64 is also used, instead of the default +33 used in the previous analysis.

```
bam.files <- paste0(sra.numbers, ".bam")
align(index="index/mm10", readfile1=all.fastq, type=1, phredOffset=64,
    input_format="FASTQ", output_file=bam.files)
```

Alignments in each BAM file are sorted by coordinate. Duplicate reads are marked, and the resulting files are indexed.

```
temp.bam <- "cbp_temp.bam"
temp.file <- "cbp_metric.txt"
temp.dir <- "cbp_working"
dir.create(temp.dir)
for (bam in bam.files) {
    out <- suppressWarnings(sortBam(bam, "cbp_temp"))
    file.rename(out, bam)
    code <- system(sprintf("MarkDuplicates I=%s O=%s M=%s \\
        TMP_DIR=%s AS=true REMOVE_DUPLICATES=false \\
        VALIDATION_STRINGENCY=SILENT",
        bam, temp.bam, temp.file, temp.dir))
    stopifnot(code==0L)
    file.rename(temp.bam, bam)
}
indexBam(bam.files)
```

Some mapping statistics can be reported as previously described. For brevity, the code will not be shown here, as it is identical to that used for the H3K9ac analysis.

```
##                   Total   Mapped  Marked Prop.mapped Prop.marked
## SRR1145787.bam 28525952 24015041 2244935    84.18664    9.348037
## SRR1145788.bam 25514465 21288115 2062157    83.43547    9.686893
## SRR1145789.bam 34476967 28830024 2678297    83.62111    9.289958
## SRR1145790.bam 32624587 27067108 2912659    82.96537   10.760880
```

Detecting DB between genotypes for CBP

Counting reads into windows. First, a `readParam` object is constructed to standardize the parameter settings in this analysis. The ENCODE blacklist is again used to remove reads in problematic regions. For consistency, the MAPQ threshold of 50 is also re-used here for removing poorly aligned reads. Lower thresholds (e.g., from 10 to 20) can be used for longer reads with more reliable mapping locations - though in practice, the majority of long read alignments reported by *Rsubread* tend to have very high or very low MAPQ scores, such that the exact choice of the MAPQ threshold is not a critical parameter.

```
param <- readParam(minq=50, discard=blacklist)
```

The average fragment length is estimated by maximizing the cross-correlation function, as previously described.

```
x <- correlateReads(bam.files, param=reform(param, dedup=TRUE))
frag.len <- which.max(x) - 1
frag.len
```

```
## [1] 162
```

Reads are then counted into sliding windows. For TF data analyses, smaller windows are necessary to capture sharp binding sites. A large window size will be suboptimal as the count for a particular site will be "contaminated" by non-specific background in the neighbouring regions. In this case, a window size of 10 bp is used.

```
win.data <- windowCounts(bam.files, param=param, width=10, ext=frag.len)
win.data
```

```
## class: RangedSummarizedExperiment
## dim: 9127613 4
## metadata(4): spacing width shift final.ext
## assays(1): counts
## rownames: NULL
## rowRanges metadata column names(0):
## colnames: NULL
## colData names(4): bam.files totals ext param
```

The default spacing of 50 bp is also used here. This may seem inappropriate, given that the windows are only 10 bp. However, reads lying in the interval between adjacent windows will still be counted into several windows. This is because reads are extended to the value of `frag.len`, which is substantially larger than the 50 bp spacing. Again, smaller spacings can be used but will provide little benefit, given that each extended read already overlaps multiple windows.

Normalization for composition biases. Composition biases are introduced when the amount of DB in each condition is unbalanced (Lun & Smyth, 2014; Robinson & Oshlack, 2010). More binding in one condition means that more reads are sequenced at the binding sites, leaving fewer reads for the rest of the genome. This suppresses the genomic coverage at non-DB sites, resulting in spurious differences between libraries. To remove this bias, reads are counted into large genomic bins. Most bins are assumed to represent non-DB background regions. Any systematic differences in the coverage of those bins is attributed to composition bias and is normalized out. Specifically, the TMM method (Robinson & Oshlack, 2010) is applied to compute normalization factors from the bin counts. These factors can then be applied to the DB analysis with the window counts.

```
bins <- windowCounts(bam.files, bin=TRUE, width=10000, param=param)
normfacs <- normOffsets(bins)
normfacs
```

```
## [1] 1.011851 0.908138 1.044806 1.041588
```

The effect of normalization can be visualized with some mean-difference plots between pairs of libraries (Figure 11). The dense cloud in each plot represents the majority of bins in the genome. These are assumed to mostly contain background regions. A non-zero log-fold change for these bins indicates that composition bias is present between libraries. The red line represents the log-ratio of normalization factors and passes through the centre of the cloud in each plot, indicating that the bias has been successfully identified and removed.

```
y.bin <- asDGEList(bins)
bin.ab <- aveLogCPM(y.bin)
adjc <- cpm(y.bin, log=TRUE)
par(cex.lab=1.5, mfrow=c(1,3))
smoothScatter(bin.ab, adjc[,1]-adjc[,4], ylim=c(-6, 6),
     xlab="Average abundance", ylab="Log-ratio (1 vs 4)")
abline(h=log2(normfacs[1]/normfacs[4]), col="red")
smoothScatter(bin.ab, adjc[,2]-adjc[,4], ylim=c(-6, 6),
     xlab="Average abundance", ylab="Log-ratio (2 vs 4)")
abline(h=log2(normfacs[2]/normfacs[4]), col="red")
smoothScatter(bin.ab, adjc[,3]-adjc[,4], ylim=c(-6, 6),
     xlab="Average abundance", ylab="Log-ratio (3 vs 4)")
abline(h=log2(normfacs[3]/normfacs[4]), col="red")
```

Note that this normalization strategy is quite different from that in the H3K9ac analysis. Here, systematic DB in one direction is expected between conditions, given that CBP function is lost in the KO genotype. This means that the assumption of a non-DB majority (required for non-linear normalization of the H3K9ac data) is not valid. No such assumption is made by the binned-TMM approach described above, which makes it more appropriate for use in the CBP analysis.

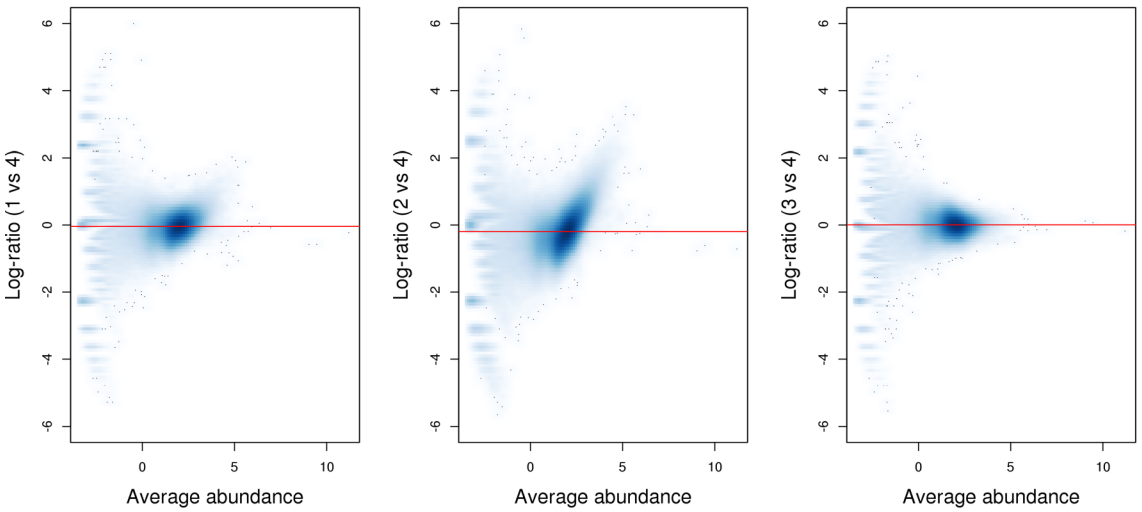

Figure 11. Mean-difference plots for the bin counts, comparing library 4 to all other libraries. The red line represents the log-ratio of the normalization factors between libraries.

Filtering of low-abundance windows. Removal of low-abundance windows is performed as previously described. The majority of windows in background regions are filtered out upon applying a modest fold-change threshold. This leaves a small set of relevant windows for further analysis.

```
filter.stat <- filterWindows(win.data, bins, type="global")
min.fc <- 3
keep <- filter.stat$filter > log2(min.fc)
summary(keep)
```

```
##     Mode   FALSE    TRUE    NA's
##  logical 8862335  265278       0
```

```
filtered.data <- win.data[keep,]
```

It should be noted that the 10 kbp bins are used here for filtering, while smaller 2 kbp bins were used in the corresponding step for the H3K9ac analysis. This is purely for convenience – the 10 kbp counts for this data set were previously loaded for normalization, and can be re-used during filtering to save time. Changes in bin size will have little impact on the results, so long as the bins (and their counts) are large enough for precise estimation of the background abundance. While smaller bins provide greater spatial resolution, this is irrelevant for quantifying coverage in large background regions that span most of the genome.

Statistical modelling of biological variability. Counts for each window are modelled using *edgeR* as previously described. First, a design matrix needs to be constructed.

```
genotype <- factor(genotype)
design <- model.matrix(~0+genotype)
colnames(design) <- levels(genotype)
design
```

```
##   ko wt
## 1  0  1
## 2  0  1
## 3  1  0
## 4  1  0
## attr(,"assign")
## [1] 1 1
## attr(,"contrasts")
## attr(,"contrasts")$genotype
## [1] "contr.treatment"
```

Estimation of the NB and QL dispersions is then performed. The estimated NB dispersions are substantially larger than those observed in the H3K9ac data set. In addition, the estimated prior d.f. is infinite.

```
y <- asDGEList(filtered.data, norm.factors=normfacs)
y <- estimateDisp(y, design)
summary(y$trended.dispersion)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.1376  0.1641  0.1835  0.1895  0.2127  0.2572
```

```
fit <- glmQLFit(y, design, robust=TRUE)
summary(fit$df.prior)
```

```
##    Min. 1st Qu. Median  Mean  3rd Qu.   Max.
##     Inf     Inf    Inf   Inf     Inf    Inf
```

These statistics are consistent with the presence of a batch effect between replicates. The dispersions for all windows are inflated to a similarly large value by the batch effect, resulting in low variability in the dispersions across windows. This is illustrated in Figure 12 where the WT libraries are clearly separated in both dimensions of the MDS plot. In particular, separation of replicates on the first dimension is indicative of a systematic difference of size comparable to that between genotypes.

```
plotMDS(cpm(y, log=TRUE), top=10000, labels=genotype,
    col=c("red", "blue")[as.integer(genotype)])
```

The presence of a large batch effect between replicates is not ideal. Nonetheless, the DB analysis can proceed, albeit with some loss of power due to the inflated NB dispersions.

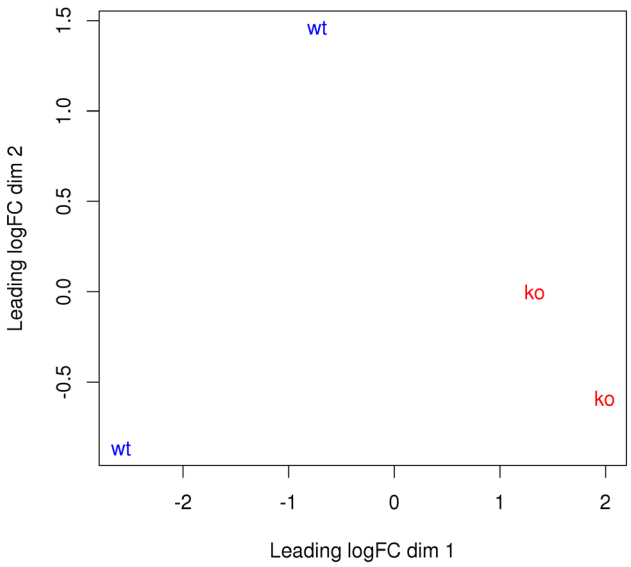

Figure 12. MDS plot with two dimensions for all libraries in the CBP data set. Libraries are labelled and coloured according to the genotype. A larger top set of windows was used to improve the visualization of the genome-wide differences between the WT libraries.

Testing for DB. DB windows are identified using the QL F-test. Windows are clustered into regions, and the region-level FDR is controlled using Simes' method. All significant regions have increased CBP binding in the WT genotype. This is expected, given that protein function should be lost in the KO genotype.

```
contrast <- makeContrasts(wt-ko, levels=design)
res <- glmQLFTest(fit, contrast=contrast)
merged <- mergeWindows(rowRanges(filtered.data), tol=100, max.width=5000)
tabcom <- combineTests(merged$id, res$table)
tabbest <- getBestTest(merged$id, res$table)
is.sig <- tabcom$FDR <= 0.05
summary(is.sig)
```

```
##    Mode   FALSE    TRUE   NA's
## logical   55444    1969       0
```

```
is.sig.pos <- (tabbest$logFC > 0)[is.sig]
summary(is.sig.pos)
```

```
##    Mode    TRUE   NA's
## logical    1969       0
```

These results can be saved to file, as previously described. Key objects are also saved for convenience.

```
out.ranges <- merged$region
elementMetadata(out.ranges) <- data.frame(tabcom,
    best.pos=mid(ranges(rowRanges(filtered.data[tabbest$best]))),
    best.logFC=tabbest$logFC)
saveRDS(file="cbp_results.rds", out.ranges)
save(file="cbp_objects.Rda", win.data, bins, y)
```

Annotation and visualization

Annotation is added using the `detailRanges` function, as previously described.

```
anno <- detailRanges(out.ranges, orgdb=org.Mm.eg.db,
            txdb=TxDb.Mmusculus.UCSC.mm10.knownGene)
meta <- elementMetadata(out.ranges)
elementMetadata(out.ranges) <- data.frame(meta, anno)
```

The top-ranked DB event will be visualized here. This corresponds to a simple DB event, as all windows are changing in the same direction, i.e., up in the WT. The binding region is also quite small relative to some of the H3K9ac examples, consistent with sharp TF binding to a specific recognition site.

```
o <- order(out.ranges$PValue)
cur.region <- out.ranges[o[1]]
cur.region
```

```
## GRanges object with 1 range and 10 metadata columns:
##       seqnames              ranges strand |  nWindows  logFC.up
##          <Rle>           <IRanges>  <Rle> | <integer> <integer>
##   [1]    chr16 [70313851, 70314860]      * |        21        21
##      logFC.down           PValue       FDR  best.pos best.logFC
##       <integer>        <numeric> <numeric> <integer>  <numeric>
##   [1]          0  0.0000001802112 0.00348259  70314405   5.273053
##         overlap       left     right
##        <factor>   <factor>  <factor>
##   [1] Gbe1|0-1|+
##   -------
##   seqinfo: 66 sequences from an unspecified genome
```

Plotting is performed using two tracks for each library – one for the forward-strand coverage, another for the reverse-strand coverage. This allows visualization of the strand bimodality that is characteristic of genuine TF binding sites. In Figure 13, two adjacent sites are present at the *Gbe1* promoter, both of which exhibit increased binding in the WT genotype. Coverage is also substantially different between the WT replicates, consistent with the presence of a batch effect.

```
collected <- list()
lib.sizes <- filtered.data$totals/1e6
for (i in 1:length(bam.files)) {
    reads <- extractReads(bam.file=bam.files[i], cur.region, param=param)
    pcov <- as(coverage(reads[strand(reads)=="+"])/lib.sizes[i], "GRanges")
    ncov <- as(coverage(reads[strand(reads)=="-"])/-lib.sizes[i], "GRanges")
    ptrack <- DataTrack(pcov, type="histogram",  lwd=0, ylim=c(-5, 5),
          name=bam.files[i], col.axis="black", col.title="black",
          fill="blue", col.histogram=NA)
    ntrack <- DataTrack(ncov, type="histogram", lwd=0, ylim=c(-5, 5),
          fill="red", col.histogram=NA)
    collected[[i]] <- OverlayTrack(trackList=list(ptrack, ntrack))
}
plotTracks(c(gax, collected, greg), chromosome=as.character(seqnames(cur.region)),
    from=start(cur.region), to=end(cur.region))
```

Note that that the `gax` and `greg` objects are the same as those used in the visualization of the H3k9ac data.

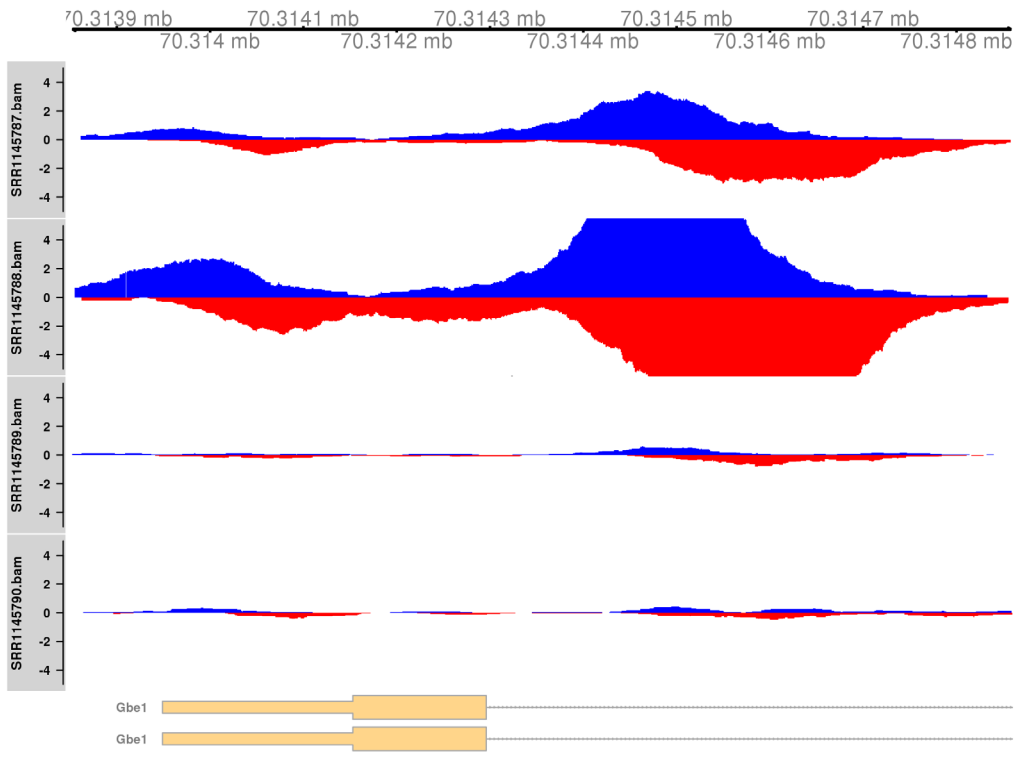

Figure 13. Coverage tracks for TF binding sites that are differentially bound in the WT (top two tracks) against the KO (last two tracks). Blue and red tracks represent forward- and reverse-strand coverage, respectively, on a per-million scale (capped at 5 in SRR1145788, for visibility).

Summary

This workflow describes the steps of a window-based DB analysis, from read alignment through to visualization of DB regions. All steps are performed within the R environment and mostly use functions from Bioconductor packages. In particular, the core of the workflow – the detection of DB regions – is based on a combination of *csaw* and *edgeR*. Analyses are shown for histone mark and TF data sets, with differences in parametrization that are appropriate to each data type. Readers are encouraged to apply the concepts and code presented in this article to their own data.

Software availability

This workflow depends on various packages from version 3.2 of the Bioconductor project, running on *R* version 3.2.2 or higher. It requires a number of software packages, including *csaw*, *edgeR*, *Rsubread*, *Rsamtools*, *Gviz*, *rtracklayer* and *ChIPpeakAnno*. It also depends on the annotation packages *org.Mm.eg.db* and *TxDb.Mmusculus.UCSC.mm10.knownGene*. Version numbers for all packages used are shown below.

```
sessionInfo()
```

```
## R version 3.2.2 Patched (2015-10-30 r69588)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: CentOS release 6.4 (Final)
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=en_US.UTF-8        LC_COLLATE=en_US.UTF-8
##  [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
##  [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
##  [1] grid    parallel stats4  methods  stats  graphics  grDevices
##  [8] utils   datasets base
##
## other attached packages:
##  [1] Gviz_1.14.0
##  [2] ChIPpeakAnno_3.4.3
##  [3] VennDiagram_1.6.16
##  [4] futile.logger_1.4.1
##  [5] TxDb.Mmusculus.UCSC.mm10.knownGene_3.2.2
##  [6] GenomicFeatures_1.22.7
##  [7] org.Mm.eg.db_3.2.3
##  [8] RSQLite_1.0.0
##  [9] DBI_0.3.1
## [10] AnnotationDbi_1.32.2
## [11] edgeR_3.12.0
## [12] limma_3.26.3
## [13] locfit_1.5-9.1
## [14] statmod_1.4.22
## [15] csaw_1.4.1
## [16] SummarizedExperiment_1.0.1
## [17] Biobase_2.30.0
## [18] rtracklayer_1.30.1
## [19] Rsamtools_1.22.0
## [20] Biostrings_2.38.2
## [21] XVector_0.10.0
## [22] GenomicRanges_1.22.2
## [23] GenomeInfoDb_1.6.1
## [24] IRanges_2.4.6
## [25] S4Vectors_0.8.5
## [26] BiocGenerics_0.16.1
## [27] Rsubread_1.20.2
```

```
## [28] knitr_1.11
## [29] BiocStyle_1.8.0
##
## loaded via a namespace (and not attached):
##  [1] httr_1.0.0                      regioneR_1.2.0
##  [3] AnnotationHub_2.2.2             splines_3.2.2
##  [5] Formula_1.2-1                   shiny_0.12.2
##  [7] interactiveDisplayBase_1.8.0    latticeExtra_0.6-26
##  [9] RBGL_1.46.0                     BSgenome_1.38.0
## [11] lattice_0.20-33                 biovizBase_1.18.0
## [13] digest_0.6.8                    RColorBrewer_1.1-2
## [15] colorspace_1.2-6                htmltools_0.2.6
## [17] httpuv_1.3.3                    plyr_1.8.3
## [19] XML_3.98-1.3                    biomaRt_2.26.1
## [21] zlibbioc_1.16.0                 xtable_1.8-0
## [23] GO.db_3.2.2                     scales_0.3.0
## [25] BiocParallel_1.4.3              ggplot2_2.0.0
## [27] nnet_7.3-11                     survival_2.38-3
## [29] magrittr_1.5                    mime_0.4
## [31] memoise_0.2.1                   evaluate_0.8
## [33] MASS_7.3-45                     foreign_0.8-66
## [35] graph_1.48.0                    BiocInstaller_1.20.1
## [37] tools_3.2.2                     formatR_1.2.1
## [39] matrixStats_0.50.1              stringr_1.0.0
## [41] munsell_0.4.2                   cluster_2.0.3
## [43] ensembldb_1.2.1                 lambda.r_1.1.7
## [45] RCurl_1.95-4.7                  dichromat_2.0-0
## [47] VariantAnnotation_1.16.4        bitops_1.0-6
## [49] gtable_0.1.2                    multtest_2.26.0
## [51] R6_2.1.1                        gridExtra_2.0.0
## [53] GenomicAlignments_1.6.1         Hmisc_3.17-1
## [55] futile.options_1.0.0            KernSmooth_2.23-15
## [57] stringi_1.0-1                   Rcpp_0.12.2
## [59] rpart_4.1-10                    acepack_1.3-3.3
```

For the command-line tools, the `fastq-dump` utility (version 2.4.2) from the SRA Toolkit must be installed on the system, along with the `MarkDuplicates` command from the Picard software suite (version 1.117). Readers should note that the read alignment steps for each data set can only be performed on Unix or Mac OS. This is because the various `system` calls assume that a Unix-style command-line interface is present. In addition, *Rsubread* is not supported for Windows. However, downstream analyses of the BAM files can be performed using any platform on which *R* can be installed. The entire workflow takes 7–8 hours to run and requires 10 GB of RAM.

Author contributions

A.T.T.L. developed and tested the workflow on the H3K9ac and CBP data sets. G.K.S. provided direction on the design of the workflow. Both A.T.T.L. and G.K.S. wrote the article.

Competing interests

No competing interests were disclosed.

Grant information

National Health and Medical Research Council (Program Grant 1054618 to G.K.S., Fellowship to G.K.S.); Victorian State Government Operational Infrastructure Support; Australian Government NHMRC IRIIS.

I confirm that the funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Acknowledgements

The authors would like to thank Prof. Stephen Nutt for his valuable insights on B-cell biology.

References

Ballman KV, Grill DE, Oberg AL, *et al.*: **Faster cyclic loess: normalizing RNA arrays via linear models.** *Bioinformatics.* 2004; **20**(16): 2778–2786.
PubMed Abstract | Publisher Full Text

Benjamini Y, Hochberg Y: **Controlling the false discovery rate: a practical and powerful approach to multiple testing.** *J Royal Stat Soc B.* 1995; **57**(1): 289–300.
Reference Source

Edgar R, Domrachev M, Lash AE: **Gene Expression Omnibus: NCBI gene expression and hybridization array data repository.** *Nucleic Acids Res.* 2002; **30**(1): 207–210.
PubMed Abstract | Publisher Full Text | Free Full Text

ENCODE Project Consortium: **An integrated encyclopedia of DNA elements in the human genome.** *Nature.* 2012; **489**(7414): 57–74.
PubMed Abstract | Publisher Full Text | Free Full Text

Hoffmann R, Seidl T, Neeb M, *et al.*: **Changes in gene expression profiles in developing B cells of murine bone marrow.** *Genome Res.* 2002; **12**(1): 98–111.
PubMed Abstract | Publisher Full Text | Free Full Text

Huber W, Carey VJ, Gentleman R, *et al.*: **Orchestrating high-throughput genomic analysis with Bioconductor.** *Nat Methods.* 2015; **12**(2): 115–121.
PubMed Abstract | Publisher Full Text | Free Full Text

Humburg P, Helliwell CA, Bulger D, *et al.*: **ChIPseqR: analysis of ChIP-seq experiments.** *BMC Bioinformatics.* 2011; **12**: 39.
PubMed Abstract | Publisher Full Text | Free Full Text

Karmodiya K, Krebs AR, Oulad-Abdelghani M, *et al.*: **H3K9 and H3K14 acetylation co-occur at many gene regulatory elements, while H3K14ac marks a subset of inactive inducible promoters in mouse embryonic stem cells.** *BMC Genomics.* 2012; **13**: 424.
PubMed Abstract | Publisher Full Text | Free Full Text

Kasper LH, Qu C, Obenauer JC, *et al.*: **Genome-wide and single-cell analyses reveal a context dependent relationship between CBP recruitment and gene expression.** *Nucleic Acids Res.* 2014; **42**(18): 11363–11382.
PubMed Abstract | Publisher Full Text | Free Full Text

Kharchenko PV, Tolstorukov MY, Park PJ: **Design and analysis of ChIP-seq experiments for DNA-binding proteins.** *Nat Biotechnol.* 2008; **26**(12): 1351–1359.
PubMed Abstract | Publisher Full Text | Free Full Text

Landt SG, Marinov GK, Kundaje A, *et al.*: **ChIP-seq guidelines and practices of the ENCODE and modENCODE consortia.** *Genome Res.* 2012; **22**(9): 1813–1831.
PubMed Abstract | Publisher Full Text | Free Full Text

Lawrence M, Gentleman R, Carey V: **rtracklayer: an R package for interfacing with genome browsers.** *Bioinformatics.* 2009; **25**(14): 1841–1842.
PubMed Abstract | Publisher Full Text | Free Full Text

Liang K, Keles S: **Detecting differential binding of transcription factors with ChIP-seq.** *Bioinformatics.* 2012; **28**(1): 121–122.
PubMed Abstract | Publisher Full Text | Free Full Text

Liao Y, Smyth GK, Shi W: **The Subread aligner: fast, accurate and scalable read mapping by seed-and-vote.** *Nucleic Acids Res.* 2013; **41**(10): e108.
PubMed Abstract | Publisher Full Text | Free Full Text

Lun AT, Smyth GK: ***De novo* detection of differentially bound regions for ChIP-seq data using peaks and windows: controlling error rates correctly.** *Nucleic Acids Res.* 2014; **42**(11): e95.
PubMed Abstract | Publisher Full Text | Free Full Text

Lun AT, Smyth GK: **csaw: a Bioconductor package for differential binding analysis of ChIP-seq data using sliding windows.** *Nucleic Acids Res.* 2015; pii: gkv1191.
PubMed Abstract | Publisher Full Text

Lund SP, Nettleton D, McCarthy DJ, *et al.*: **Detecting differential expression in RNA-sequence data using quasi-likelihood with shrunken dispersion estimates.** *Stat Appl Genet Mol Biol.* 2012; **11**(5): 1544–6115.
PubMed Abstract | Publisher Full Text

McCarthy DJ, Chen Y, Smyth GK: **Differential expression analysis of multifactor RNA-Seq experiments with respect to biological variation.** *Nucleic Acids Res.* 2012; **40**(10): 4288–4297.
PubMed Abstract | Publisher Full Text | Free Full Text

Pal B, Bouras T, Shi W, *et al.*: **Global changes in the mammary epigenome are induced by hormonal cues and coordinated by Ezh2.** *Cell Rep.* 2013; **3**(2): 411–426.
PubMed Abstract | Publisher Full Text

Podojil JR, Sanders VM: **Selective regulation of mature IgG1 transcription by CD86 and beta 2-adrenergic receptor stimulation.** *J Immunol.* 2003; **170**(10): 5143–5151.
PubMed Abstract | Publisher Full Text

Revilla-I-Domingo R, Bilic I, Vilagos B, *et al.*: **The B-cell identity factor Pax5 regulates distinct transcriptional programmes in early and late B lymphopoiesis.** *EMBO J.* 2012; **31**(14): 3130–3146.
PubMed Abstract | Publisher Full Text | Free Full Text

Robinson MD, McCarthy DJ, Smyth GK: **edgeR: a Bioconductor package for differential expression analysis of digital gene expression data.** *Bioinformatics.* 2010; **26**(1): 139–140.
PubMed Abstract | Publisher Full Text | Free Full Text

Robinson MD, Oshlack A: **A scaling normalization method for differential expression analysis of RNA-seq data.** *Genome Biol.* 2010; **11**(3): R25.
PubMed Abstract | Publisher Full Text | Free Full Text

Rosenbloom KR, Armstrong J, Barber GP, *et al.*: **The UCSC Genome Browser database: 2015 update.** *Nucleic Acids Res.* 2015; **43**(Database issue): D670–681.
PubMed Abstract | Publisher Full Text | Free Full Text

Ross-Innes CS, Stark R, Teschendorff AE, *et al.*: **Differential oestrogen receptor binding is associated with clinical outcome in breast cancer.** *Nature.* 2012; **481**(7381): 389–393.
PubMed Abstract | Publisher Full Text | Free Full Text

Simes RJ: **An improved Bonferroni procedure for multiple tests of significance.** *Biometrika.* 1986; **73**(3): 751–754.
Publisher Full Text

Zhang Y, Liu T, Meyer CA, *et al.*: **Model-based analysis of ChIP-Seq (MACS).** *Genome Biol.* 2008; **9**(9): R137.
PubMed Abstract | Publisher Full Text | Free Full Text

Zhu LJ, Gazin C, Lawson ND, *et al.*: **ChIPpeakAnno: a Bioconductor package to annotate ChIP-seq and ChIP-chip data.** *BMC Bioinformatics.* 2010; **11**: 237.
PubMed Abstract | Publisher Full Text | Free Full Text

Open Peer Review

Current Referee Status: ✓ ? ✓

Version 1

Referee Report 18 December 2015

doi:10.5256/f1000research.7553.r10879

Rory Stark
Cancer Research UK Cambridge Institute, University of Cambridge, Cambridge, UK

This article represents a comprehensive and useful presentation of a window-based differential binding analysis. Currently, most quantitative differential binding studies rely on a peak calling step; this demonstration of the benefits of avoiding such a step is of great interest. The workflow is a valuable complement to the author's related published discussions (Lun and Smyth 2014, 2015). It is comprehensive in that is takes the user from archived sequencing reads, through two core analyses, and includes annotation and visualization code as well. Notably the authors include interesting exploration of many important details, particularly in the area of normalization, that are frequently overlooked and can have a crucial impact on analytical results.

I include below a number of questions for the authors, the answer to which may make the article even more useful.

- **Computational resources.** It would be useful to include some discussion of the computational resources required to perform this analysis (memory and compute time). How do memory/compute requirements change as a function of lowering the window size? Can this handle a large number of samples?

- **Experimental design**. Are there any guidelines of how many replicates should be included for a successful analysis?

- **Duplication rates**. Duplicates are included in the analysis, are there any guidelines for acceptable duplication rates? The authors say that "ideally, the proportion of mapped reads should be high, while the proportion of marked reads should be low" -- how high and how low? Is there some point where there are too many duplicates to expect a successful analysis?

- **Blacklists**. The workflow uses a published blacklist to mask areas of the genome where reads will be not be counted. Some of these are attributable repeat regions, but some are anomalous, and there may be tissue-specific issues. Given that the workflow includes only very limited use of control reads (such as Input) for filtering non-enriched windows, perhaps it would be good to use blacklists generated from the controls, such as those made by the GreyListChIP Bioconductor package.This is especially important for experiments that use multiple tissue types or cell lines.

- **Read counting.** The authors state that “the number of extended reads overlapping a window is counted”. As almost all reads will overlap more than one window (fragment lengths generally being longer than the window size), it would be helpful to be explicit regarding if a read is counted in more

than one window. If so, do single reads resulting in multiple counts have an implication for the assumption of binomial distribution of reads?

- **Filtering windows by abundance.**
  - By **using read abundance to remove windows** prior to testing, is there an issue with the same information being used to choosing which windows to compare as is used for the comparison itself? Can window filtering be compared to peak calling in that it uses read counts to reduce the proportion of the genome being considered for differential analysis?
  - By **using a fold measure as a threshold,** in many cases, very small changes in the number of background reads can have a big impact on calculated fold change. How sensitive is this filter process? How important is it to final results?

- **Normalization**
  - Is there way to **determine computationally if a trended-bias normalization is appropriate**, or is this best done by visual examination of the mean-difference plot?
  - If **the scale of differential binding is not known *a priori***, what normalization method should be used? Is it safe to use a non-linear trended correction? Or TMM on large "background" windows?

- **Merging windows**
  - It appears possible to create **merged regions with higher FDR than the minimum FDR of constituent windows**. Windows that would have an FDR lower than some “significance” threshold may be “lost” in a merged region. Is there a way to constrain the merging function to not create a “non-significant” region that contains “significant” windows (according to a specified FDR threshold)?
  - Another reviewer commented that it would be nice to not **merge regions that have windows with both positive and negative fold changes**, this seems useful as well.
  - Regarding regions with positive and negative fold changes, it seems worth referencing MMDiff (Schweikert et al BMC Genomics 2013), which is designed to **detect differences in the gain/loss patterns of binding profiles.**

- **For the CBP example**:
  - One or more **plots visualizing the batch effec**t (MDS/PCA) would be helpful!
  - **Is a batch effect the only possible explanation** for large dispersion estimates and infinite prior d.f.? Can we always assume a batch effect if we see this?
  - If the batch effect applies to a more than one sample, **can this be modeled in a multi-factor design**? If it applies only to one sample, perhaps this is a ChIP efficiency issue?

References

1. Schweikert G, Cseke B, Clouaire T, Bird A, Sanguinetti G: MMDiff: quantitative testing for shape changes in ChIP-Seq data sets.*BMC Genomics*. 2013; **14**: 826 PubMed Abstract | Publisher Full Text
2. Lun AT, Smyth GK: De novo detection of differentially bound regions for ChIP-seq data using peaks and windows: controlling error rates correctly.*Nucleic Acids Res*. 2014; **42** (11): e95 PubMed Abstract | Publisher Full Text
3. Lun AT, Smyth GK: csaw: a Bioconductor package for differential binding analysis of ChIP-seq data using sliding windows.*Nucleic Acids Res*. 2015. PubMed Abstract | Publisher Full Text

I have read this submission. I believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard.

Competing Interests: No competing interests were disclosed.

Author Response 19 Dec 2015

Aaron Lun, Walter and Eliza Hall Institute of Medical Research, Australia

This article represents a comprehensive and useful presentation of a window-based differential binding analysis.

Thanks Rory.

It would be useful to include some discussion of the computational resources required to perform this analysis (memory and compute time). How do memory/compute requirements change as a function of lowering the window size? Can this handle a large number of samples?

Including the alignment steps, the entire analysis of the two data sets takes approximately 7-8 hours. It uses around 10 GB of memory in total, most of which is spent in alignment or in processing the large numbers of windows in edgeR. Memory requirements will not change as a function of window size but will change linearly as a function of window spacing, as this determines the number of windows that are extracted from the genome. We recommend that, for large windows, the spacing can be increased to avoid unnecessary computational work (given that any additional loss of spatial resolution is irrelevant for large windows). As for the number of samples, we typically use csaw to analyze data from small contained experiments (5-10 samples). Large numbers (>50) of samples will probably require high-performance computing resources and some additional care, e.g., chromosome-by-chromosome processing. As suggested, we have added some expected time/memory requirements to the "Software availability" section.

Are there any guidelines of how many replicates should be included for a successful analysis?

edgeR requires at least two replicates in one group to estimate the NB/QL dispersion. We usually analyze data with two replicates in each of our biological conditions of interest. This ensures that the results for each condition are replicable and are not the result of technical artifacts such as PCR duplicates. We have added some comments regarding this to the text. Of course, the more replicates, the better, but we appreciate that ChIP-seq is one of the more technically challenging techniques and that obtaining high levels of replication is not always feasible.

Duplicates are included in the analysis, are there any guidelines for acceptable duplication rates? The authors say that "ideally, the proportion of mapped reads should be high, while the proportion of marked reads should be low" -- how high and how low? Is there some point where there are too many duplicates to expect a successful analysis?

We have added some ballpark figures to the workflow. In our experience, mapping proportions above 70 to 80% are quite satisfactory (with the missing reads probably lying in unmappable repeat regions), along with duplicate proportions below 20%. While some of the marked reads will inevitably correspond to non-duplicate fragments that happen to overlap in enriched regions, this is

unlikely to explain very high proportions of marked reads (> 40%). Such cases are more likely to be caused by high levels of PCR duplication.

Given that the workflow includes only very limited use of control reads (such as Input) for filtering non-enriched windows, perhaps it would be good to use blacklists generated from the controls, such as those made by the GreyListChIP Bioconductor package.This is especially important for experiments that use multiple tissue types or cell lines.

Done.

As almost all reads will overlap more than one window (fragment lengths generally being longer than the window size), it would be helpful to be explicit regarding if a read is counted in more than one window. If so, do single reads resulting in multiple counts have an implication for the assumption of binomial distribution of reads?

Yes, reads are often counted in more than one window. We have added a statement regarding this to the workflow. Multiple counting of reads introduces technical correlations between windows, but this will not affect the downstream analysis. Estimation of the EB shrinkage statistics in edgeR are robust to correlations between features. Similarly, the BH correction used to control the FDR is robust to correlations between tests.

By using read abundance to remove windows prior to testing, is there an issue with the same information being used to choosing which windows to compare as is used for the comparison itself? Can window filtering be compared to peak calling in that it uses read counts to reduce the proportion of the genome being considered for differential analysis?

The use of an independent filtering criterion is the important concept here. In a NB model, the average abundance (i.e., the log-NB mean) is a filter statistic that is independent of the DB status of each test, i.e., knowing the average abundance doesn't tell you whether or not the window is DB. This ensures that the selection of high-abundance windows will not bias the analysis towards or against detecting DB in those windows, and maintains the validity of the downstream multiplicity correction procedures.

Conceptually, filtering and peak calling have some similarities, as you have pointed out. However, from a statistical perspective, filtering on the average abundance is only equivalent to peak calling if all libraries were pooled together and the pooled library was used for peak calling (and even then, only if those libraries are of the same size). Indeed, most peak callers were not designed for processing multi-sample data sets, especially not in a manner that preserves the validity of downstream DB analyses.

By using a fold measure as a threshold, in many cases, very small changes in the number of background reads can have a big impact on calculated fold change. How sensitive is this filter process? How important is it to final results?

We protect against this effect by using large bins to compute the background abundance. This increases the number of reads in each bin and stabilizes the estimated abundance against stochastic changes in coverage. Further stability is provided by taking the median of the abundance across all bins. We find that random differences in coverage make little difference to the estimated background and to the filter threshold. Instead, filtering is more sensitive to the

choice of the fold-increase over the background. Obviously, requiring a larger fold-increase will result in the loss of weak binding sites. This is an arbitrary decision that depends on what the user considers to be relevant and cannot be avoided, even with peak callers (in which case, the choice of threshold is that of the significance of each peak).

Is there way to determine computationally if a trended-bias normalization is appropriate, or is this best done by visual examination of the mean-difference plot?

This is best done with visual examination. Abundance-dependent normalization involves fitting a trend, and then adjusting the observations in order to "flatten out" the trend. This means that if you were to repeat the trend fitting process on the adjusted data, you would end up with a flat trend by definition. This would not be helpful in diagnosing problems in the normalization procedure.

If the scale of differential binding is not known a priori, what normalization method should be used? Is it safe to use a non-linear trended correction? Or TMM on large "background" windows?

We would suggest repeating the analysis with both normalization strategies. If they give similar results, then it doesn't matter which method is used. However, if they yield different results, this indicates that there are systematic differences in read coverage between some of the libraries. The origin of these differences cannot be conclusively determined from the data alone - they may be due to differences in IP efficiency, or due to systematic and genuine DB in one set of conditions. Thus, care will be required in the interpretation of the DB results. Regions detected with both approaches are most likely to be reliable.

We note that systematic differences between replicates within a condition can be directly interpreted as efficiency biases. However, this does not guarantee that there are no systematic differences in binding between conditions. Applying normalization to remove efficiency biases will also remove any systematic and genuine DB between conditions. In short, an assumption about the underlying biology is still required in order to apply one method or the other. In practice, normalization of the efficiency biases is usually the lesser of two evils, as large differences between replicates will inflate the dispersions and interfere with DB detection.

A third possibility is to use spike-ins (e.g., *Drosophila* chromatin) for normalization. This should be able to distinguish between genuine DB and efficiency bias, as only the latter should affect spike-in coverage. See the csaw user's guide for some guidelines on accommodating spike-in data in csaw.

Is there a way to constrain the merging function to not create a “non-significant” region that contains “significant” windows (according to a specified FDR threshold)?

As a general rule, the merging algorithm must be independent of the significance of the window in order to maintain statistical validity. This means that it is not allowed to know which windows are significant or not during its operation. If this is not true, FDR control across the regions cannot be guaranteed.

That said, it may be useful in some applications to get tighter intervals containing only the significantly DB regions. This can be achieved by defining putative DB windows based on a window-level FDR threshold, and then clustering them to obtain DB regions. An informal estimate

of the region-level FDR for these DB regions can be computed using the clusterFDR() function, and the window-level threshold can then be adjusted until the region-level FDR is below some desired threshold. This two-step procedure is necessary as the window-level and region-level FDRs can be quite different for many overlapping windows. However, it is less statistically rigorous than the default approach which is blind to the significance of each window.

Another reviewer commented that it would be nice to not merge regions that have windows with both positive and negative fold changes, this seems useful as well.

Such a merging procedure would result in loss of detection power - see our comments below.

One or more plots visualizing the batch effect (MDS/PCA) would be helpful!

Added. Note that we use a larger 'top' set of windows to make the MDS plot. This is simply to improve the visualization of the systematic differences between libraries that are not captured with the default top value (500).

Is a batch effect the only possible explanation for large dispersion estimates and infinite prior d.f.? Can we always assume a batch effect if we see this?

Very large dispersions in conjunction with infinite prior d.f. means there are large differences between the replicates that are too consistent to be random. In other words, there are systematic differences between the replicates, and that is almost the definition of a batch effect.

If the batch effect applies to a more than one sample, can this be modeled in a multi-factor design? If it applies only to one sample, perhaps this is a ChIP efficiency issue?

Yes, the batch effect can easily be modelled in the GLM, provided that it does not confound detection of DB between the conditions of interest.

Batch effects and differences in ChIP efficiency are not mutually exclusive - the fact that samples are processed in separate batches is often the cause for a difference in efficiency. In this case, systematic differences in peak heights are observed between the two CBP WT replicates. This can be seen most clearly in the coverage tracks of Figure 12 (13 in version 2). This would suggest that the batch effect corresponds to an increase in ChIP efficiency for one of the WT libraries.

Competing Interests: No competing interests were disclosed.

Referee Report 25 November 2015

doi:10.5256/f1000research.7553.r10877

Cenny Taslim
Comprehensive Cancer Center, Ohio State University Medical Center, Columbus, OH, USA

This article provides a useful workflow on detecting differential binding in ChIP-seq data with examples and codes from R packages and other softwares. This paper will help researchers on analyzing their ChIP-seq using R. However, it would be better if there was more description on the methods used in the

workflow.

“Reads are first aligned to the genome using the *Rsubread* package (Liao *et al.*, 2013).” Please describe what algorithm is used to align the reads.

"Technical replicates are merged together prior to further processing." What does merged together mean? Concatenate? Or average? Or pick one randomly?

"Ideally, the proportion of mapped reads should be high, while the proportion of marked reads should be low." Proportion mapped is those reads that can be uniquely mapped to the genome?

"Thus, the marking status of each read will be ignored in the rest of the analysis, i.e., no duplicates will be removed in downstream steps." I believe generally people exclude duplicates in downstream steps?

"By default, windows with very low counts are removed to reduce memory use." What is the definition of very low counts? <=1? If it is described by the section filtering windows by abundance, please mention it.

```
bins <- windowCounts(bam.files, bin=TRUE, width=2000, param=param)
filter.stat <- filterWindows(win.data,  bins, type="global")
min.fc <- 3
keep <- filter.stat$filter > log2(min.fc)
summary(keep)
##    Mode     FALSE    TRUE   NA's
##  logical   906406 663218     0

hist(filter.stat$back.abundances, xlab="Background abundance", main="", breaks=50)
threshold <- filter.stat$abundances[1] - filter.stat$filter[1] + log2(min.fc)
abline(v=threshold, col="red")
```

In code above, where is the background? I assumed it is filter.stat$filter[1]? However, it looks like the filter is background + log2(3)? If it’s at least 3-fold background, shouldn’t it be filter.stat$filter[1]*3?

Figure 2 doesn’t make sense. Why is there windows with < 0 abundance?

Normalizing for library-specific trended biases. Figure 3 only shows log-fold change between mature B and pro-B but there is more than one sample in mature B and pro-B. How do you apply the normalization? Do you take average of all mature B samples and average of all pro-B samples and then do the loess normalization?

```
##     Gene nWindows logFC.up logFC.down    PValue       FDR
## 6  Ldlrap1     19      11           0 0.224741404877 0.2705479855
## 7  Mdn1        29      12          11 0.000004447727 0.0001347924
## 8  Pydc3        8       0           6 0.051183399851 0.0781822366
## 9  Wfdc17       6       0           6 0.000069604922 0.0008738790
## 10 Mfap1b      19       1          10 0.107116609335 0.1440819313
## 13 Gm15772     30      12           7 0.085543435687 0.1192823092
```

How do you interpret this results? For Mdn1, there are 29 DB windows? What are the logFC.up and logFC.down.

I have read this submission. I believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard, however I have significant reservations, as outlined above.

Competing Interests: No competing interests were disclosed.

Author Response 03 Dec 2015

Aaron Lun, Walter and Eliza Hall Institute of Medical Research, Australia

Thank you for your report. You asked for more details of the methodology in a number of places, and we have added further details as appropriate. We are puzzled about your approval "with reservations", as we did not find any criticisms in your report, apart from the Figure 2 x-label not being explicit about the log-scale used.

“Reads are first aligned to the genome using the Rsubread package (Liao et al., 2013).” Please describe what algorithm is used to align the reads.

The alignment algorithm in Rsubread package uses a seed-and-vote paradigm. We have added a very brief mention of this, though for a complete description of the algorithm, it would be advisable to consult the Liao et al. reference. We note that the differences between different aligners are not relevant to this workflow article. What is relevant is that the subread aligner is an appropriate aligner for ChIP-seq data, and that it is available as a native implementation in a Bioconductor package.

"Technical replicates are merged together prior to further processing." What does merged together mean? Concatenate? Or average? Or pick one randomly?

Merging simply refers to pooling the reads from all replicates into a single library. We have altered the wording to make this more obvious.

"Ideally, the proportion of mapped reads should be high, while the proportion of marked reads should be low." Proportion mapped is those reads that can be uniquely mapped to the genome?

Yes. By default, Rsubread only reports mapped reads as those with unique mapping locations. We have added a mention of this to the text.

"Thus, the marking status of each read will be ignored in the rest of the analysis, i.e., no duplicates will be removed in downstream steps." I believe generally people exclude duplicates in downstream steps?

Duplicate removal is not recommended for routine DB analyses with edgeR. It will reduce detection power by capping read coverage at strongly bound sites, such that a region that is DB may not be detected if the coverage of that region gets capped to the same level in two libraries. It may also increase false positives for analyses involving different library sizes, when the same cap is applied across libraries; subsequent normalization for library size will result in a spurious difference in the capped heights.

In theory, duplicate removal should only be necessary and appropriate when a large portion of

duplicate reads arise from PCR duplicates of the same DNA fragment. This should seldom occur unless the amount of DNA is overly small, in which case there are likely to be more general problems with data quality.

"By default, windows with very low counts are removed to reduce memory use." What is the definition of very low counts? <=1? If it is described by the section filtering windows by abundance, please mention it.

The internal filter in windowCounts removes windows with count sums less than 10 across all libraries, simply to reduce memory usage. Windows with such low counts will not provide sufficent evidence for detecting DB, so their removal does not seem like a major loss. We have added a mention of this threshold to the text.

In code above, where is the background? I assumed it is filter.stat$filter[1]? However, it looks like the filter is background + log2(3)? If it's at least 3-fold background, shouldn't it be filter.stat$filter[1]*3?

By default, abundances in edgeR are reported as log-CPMs. So, a 3-fold increase over the background coverage corresponds to a log2(3) increase in the abundance. This has been reworded for more clarity.

Figure 2 doesn't make sense. Why is there windows with < 0 abundance?

Again, abundances are reported as log-CPMs, for which it is entirely possible to obtain negative values. We have clarified this on the x-axis label.

Normalizing for library-specific trended biases. Figure 3 only shows log-fold change between mature B and pro-B but there is more than one sample in mature B and pro-B. How do you apply the normalization? Do you take average of all mature B samples and average of all pro-B samples and then do the loess normalization?

The algorithm constructs an average library containing average counts across all samples for each window. It then performs loess normalization between each sample and this average sample. A full description is available in our recently published paper (doi: 10.1093/nar/gkv1191, to which we have added a reference), but is beyond the scope of this workflow article.

How do you interpret this results? For Mdn1, there are 29 DB windows? What are the logFC.up and logFC.down.

We have already described the meaning of these fields in the text following the first call to combineTests(), in the section "Controlling the FDR across regions". The same interpretation can be applied to the output of all combineTests() calls.

Competing Interests: No competing interests were disclosed.

Referee Report 30 October 2015

doi:10.5256/f1000research.7553.r10876

Lihua Julie Zhu[1], **Jianhong Ou**[2]
[1] Department of Molecular, Cell and Cancer Biology, Program in Bioinformatics and Integrated Biology, Program in Molecular Medicine, University of Massachusetts Medical School, Worcester, MA, USA
[2] Department of Molecular, Cell and Cancer Biology, University of Massachusetts Medical School, Worcester, MA, USA

This article “From reads to regions: a Bioconductor workflow to detect differential binding in ChIP-seq data” clearly describes a comprehensive computational workflow of Differential Binding (DB) analysis of ChIP-seq data set, based primarily on R software packages from the open-source Bioconductor project. It provides readers with practical usage examples of DB analyses of two typical types of ChIP-seq data sets, that covers all steps of the analysis pipeline, from alignment of read sequences, normalization, DB identification, annotation to interpretation and visualization of putative DB regions. We believe that this well-written paper will greatly help users to apply the workflow to their own DB analysis of ChIP-seq data sets.

In the following section, please explicitly explain that the two data sets were chosen to represent two common ChIP-seq use cases, one data set for dealing with wider peaks while the other data set is for working with sharp peaks

"The application of the methods in this article will be demonstrated on two publicly available ChIP-seq data sets. The first data set studies changes in H3K9ac marking between pro-B and mature B cells (Revilla-I-Domingo *et al.*, 2012). The second data set studies changes in CREB-binding protein (CBP) binding between wild-type and CBP knock-out cells (Kasper *et al.*, 2014). "

According to the Rsubread documentation about parameter ***type*** at http://bioconductor.org/packages/release/bioc/manuals/Rsubread/man/Rsubread.pdf, ***type*** is an integer giving the type of sequencing data. Possible values include ***0 (RNA- seq data) and 1 (genomic DNA-seq data such as WGS, WES, ChIP-seq data etc). By default, it is set to 0.*** Therefore, please set type to 1 in the following section.

"
```
library(Rsubread)
bam.files <- paste0(names(by.group), ".bam")
align(index=“index/mm10", readfile1=group.fastq, TH1=2,
   input_format=“FASTQ", output_file=bam.files)
```
"
"
```
align(index="index/mm10", readfile1=all.fastq, phredOffset=64,
    input_format="FASTQ", output_file=bam.files)"
```

Suggest adding file.exists check to see if temp.dir exists already before dir.create(temp.dir).

For system call to fastq-dump and MarkDuplicates, suggest to add path information to these programs in case the path to these programs are not in the search path.

"For this analysis, reads are only used if they have a mapping quality score above 50."
50 is pretty high although we understand that this is for illustration purposes. To prevent readers from getting the idea that 50 is the recommended cutoff, could you please also provide a quality score

threshold commonly used in the field?

"The filter threshold is defined based on the assumption that most regions in the genome are not marked by H3K9ac. Reads are counted into large bins and the median coverage across those bins is used as an estimate of the background abundance. Windows are only retained if they have abundances 3-fold higher than the background."
In the example, the window size (width) for calculating background (bg) coverage is set to 2000 bp, and the width is set to 150 bp for win.data. When calculating fold enrichment over background, is the bg coverage scaled down using the ratio of the window size (2000 vs. 150)?

"The implicit assumption of non-linear methods is that most windows at each abundance are not DB. Any systematic difference between libraries is attributed to bias and is removed. This is not appropriate in situations where large-scale DB is expected, as removal of the difference would result in loss of genuine DB. However, there is no indication that such changes are present in this data set, so non-linear methods can be applied without too much concern."
Could you please suggest a normalization method for situations where large-scale DB is expected (perhaps TMM as mentioned in the later section)?

"Note that only the trended dispersion will be used here – the common and tagwise values are only shown for diagnostic purposes"
Could you please comment on how common and tag wise values help with diagnosis of the data set?

"Determining the direction of DB is more complicated, as clusters could potentially contain windows that are changing in opposite directions."
How about controlling this by allowing nearby windows to merge only if the changing directions are the same?

"The behaviour of *ChIPpeakAnno* complements that of detailRanges. The latter reports all overlapping and flanking genes, while the former reports only the closest gene (but in greater detail). Which is preferable depends on the proclivities of the user and the purpose of the annotation."
Actually, ChIPpeakAnno can also report all overlapping and flanking genes by setting output="both" (or output ="overlapping") and maxgap. For example, it outputs all overlapping and flanking genes within 5kb if set maxgap = 5000L and output ="overlapping". Here is an example using Txdb annotation data.

```
library(TxDb.Mmusculus.UCSC.mm10.knownGene)
ucsc.mm10.knownGene <- genes(TxDb.Mmusculus.UCSC.mm10.knownGene)
anno.peaks <- annotatePeakInBatch(minimal, annotationData=ucsc.mm10.knownGene,
        output="overlapping", maxgap=5000L)
```

"Reads are then counted into sliding windows. For TF data analyses, smaller windows are necessary to capture sharp binding sites. A large window size will be suboptimal as the count for a particular site will be “contaminated” by non-specific background in the neighbouring regions. In this case, a window size of 10 bp is used.
win.data <- windowCounts(bam.files, param=param, width=10, ext=frag.len)"
It seems that space is set to 50bp for both sharp peaks and broad peaks. Could you please comment on this?

We notice that the width for local background is set to 2kb for broad peaks and 10kb for sharp peaks. Could you please justify? Would adding a fitted line in Figure 3 and Figure 4 help with the visualization? In addition, it would be great if you could provide recommendations on how to evaluate the quality of the BCV in Figure 5. Is there a range or specific shape we are targeting?

We have read this submission. We believe that we have an appropriate level of expertise to confirm that it is of an acceptable scientific standard.

Competing Interests: No competing interests were disclosed.

Author Response 02 Nov 2015

Aaron Lun, Walter and Eliza Hall Institute of Medical Research, Australia

We believe that this well-written paper will greatly help users to apply the workflow to their own DB analysis of ChIP-seq data sets.

Thanks Julie.

In the following section, please explicitly explain that the two data sets were chosen to represent two common ChIP-seq use cases, one data set for dealing with wider peaks while the other data set is for working with sharp peaks

Done.

Therefore, please set type to 1 in the following section.

Done. We note that, at the time of publication, the workflow was constructed using packages in BioC 3.1, at which time the "type" parameter was not available in Rsubread. This has now been changed, along with minor code updates to various other parts of the workflow for BioC 3.2.

Suggest adding file.exists check to see if temp.dir exists already before dir.create(temp.dir).

There's no need for the extra check. By default, "dir.create" gives a warning if the directory already exists. We want users to be able to run the code multiple times if they wish, overwriting earlier files if they exist.

For system call to fastq-dump and MarkDuplicates, suggest to add path information to these programs in case the path to these programs are not in the search path.

We believe that implementing this suggestion may be counter-productive for most readers; the absolute path to these programs on our machines will not be useful to the general audience, given that the installation directory will change on different machines. It is the user's responsibility to make sure that these system commands are available prior to analysis - how this is done is largely irrelevant to successful execution of the workflow. The Picard software suite and the SRA Toolkit are quite widely used, so we do not think that installation and access would pose a major problem for most users.

To prevent readers from getting the idea that 50 is the recommended cutoff, could you

please also provide a quality score threshold commonly used in the field?

We use 50 for the first data set as the reads are very short (< 40 bp) such that strict filtering is required to avoid mapping errors, non-uniquely-mapped reads, etc. We use the same value for the second data set for consistency, though we concede that a lower value could be used in practice. We have added a comment regarding this to the workflow.

We also note that the MAPQ values reported by (R)subread tend to be quite binary for long reads, i.e., either very low or very high. For example, in SRR1145790, there are 24212282 reads with a MAPQ score >= 50, and 24907156 with a MAPQ score >= 10. This equates to a (relatively minor) 3% increase in available reads from a large change in the MAPQ threshold. All in all, we believe that analyses based on Rsubread alignments are generally robust to the choice of threshold.

When calculating fold enrichment over background, is the bg coverage scaled down using the ratio of the window size (2000 vs. 150)?

Yes. This is done automatically within the "filterWindows" function. We have added a note to the workflow to point this out.

Could you please suggest a normalization method for situations where large-scale DB is expected (perhaps TMM as mentioned in the later section)?

As you have noted, this issue is addressed more comprehensively in the analysis for the CBP data set. We have added a comment here regarding the existence of alternative normalization strategies when large-scale DB is expected.

Could you please comment on how common and tag wise values help with diagnosis of the data set?

The common BCV provides a measure of the overall variability of the data set, averaged across all windows. The tagwise BCVs should be dispersed around the fitted trend to indicate that the fit was successful. These comments have been added to the workflow.

How about controlling this by allowing nearby windows to merge only if the changing directions are the same?

This possibility has not escaped us. However, the sign of the log-FC is not independent of the DB status of each window in this particular application. In fact, clustering windows based on the signs of the log-FCs will result in loss of detection power for DB events.

To illustrate, consider a non-DB site with a number of overlapping windows. Due to stochasticity, the log-FCs of those windows fluctuate around zero. Now, assume that we only allow merging of adjacent windows where the signs of the log-FCs are identical. For our non-DB site, we end up with lots of small clusters, comprised of windows with positive or negative log-FCs due to chance. In contrast, for a genuinely DB site, all windows will have positive (or negative) log-FCs, as stochasticity will not change the sign of the log-FC.

When this behaviour is considered on a genome-wide level, we can see that non-DB sites will generate several non-DB clusters, while DB sites will only generate one DB cluster. This results in

an increase in the number of tests with large p-values, increasing the severity of the BH correction without any increase in the number of potential discoveries. Ultimately, this results in the loss of detection power for DB events. One might be tempted to overcome this by filtering out small non-DB clusters, but this will compromise FDR control and result in liberalness.

We consider loss of power to be more problematic than the existence of complex DB events. The former means that you don't detect anything, whereas the latter just requires some more consideration during interpretation. In fact, proper identification of complex events is not irrelevant - the context of a change in binding will affect its interpretation (e.g., if it occurs adjacent to an opposing change in the same general region) and this would be lost if changes were reported separately.

Of course, these theoretical issues are beyond the scope of this workflow article. However, considerations of proper FDR control across regions have been discussed in our methodological articles (Lun and Smyth, NAR 2014; Lun and Smyth, NAR 2015).

Actually, ChIPpeakAnno can also report all overlapping and flanking genes by setting output="both" (or output ="overlapping") and maxgap. For example, it outputs all overlapping and flanking genes within 5kb if set maxgap = 5000L and output ="overlapping".

Thanks for letting us know. We have amended our comment regarding the differences between the two annotation strategies. We note that there are still some differences; in particular, for any given input region, "annotatePeakInBatch" with output="overlapping" reports each overlapping feature as a separate entry of the output object, while "detailRanges" reports all overlapping features in one string. The former provides more detail and is easier to use for further analysis, while the latter is more convenient for annotating an existing DB list that needs to be saved to file.

It seems that space is set to 50bp for both sharp peaks and broad peaks. Could you please comment on this?

In general, the spacing governs the compromise between spatial resolution and computational convenience. Smaller spacings increase resolution, at the cost of increasing memory usage and runtime. However, for most ChIP-seq analyses, resolution is fundamentally limited by the width of the binding event and by the size of the post-sonication fragments. As long as the spacing is smaller than either of these two parameters (i.e., the window size and the extension length), there will be enough windows to profile each interval of the genome for a satisfactory analysis. Smaller spacings and additional windows will provide no benefit, and will only increase computational work. This is true for both TF and histone mark analyses, such that the choice of spacing interval does not need to be changed between them. We have added a brief remark about this to the workflow.

We notice that the width for local background is set to 2kb for broad peaks and 10kb for sharp peaks. Could you please justify?

A local background of 2 kbp can also be used for sharp peaks. We have used 10 kbp for convenience; the counts for large bins were already loaded for normalization in the previous step, so it makes sense to re-use them for filtering. In general, smaller bins provide a more accurate estimate of the background abundance, as unbound regions can be distinguished from binding sites more effectively. However, loss of spatial resolution is negligible for large background regions

in the CBP data set. We have added a comment about this to the workflow.

Would adding a fitted line in Figure 3 and Figure 4 help with the visualization?

We believe that the trend in the log-fold changes is obvious enough without the need for a fitted line.

In addition, it would be great if you could provide recommendations on how to evaluate the quality of the BCV in Figure 5. Is there a range or specific shape we are targeting?

As a general rule, we expect to see a curve that decreases to a plateau with increasing average abundance. This reflects the increased reliability of the data at large counts, where the effects of stochasticity and technical artifacts (e.g., mapping errors, PCR duplicates) are averaged out. In Figure 5, the range of abundances is such that the plateau has already been reached; a more dramatic decrease can be observed by including more lower-abundance windows.

Competing Interests: No competing interests were disclosed.