diffHic: Differential analysis of ChIA-PET data

User's Guide

Aaron Lun

First edition 12 December 2012

Last revised 28 July 2014

Contents

Chapter 1

Introduction

1.1 Scope

This document describes the analysis of Hi-C data with the diffHic package. Differential interactions are identified as those with significant changes in intensity between conditions. This is achieved in a statistically rigorous manner using the methods in the edgeR package [Robinson et al., 2010]. Knowledge of edgeR is useful but is not necessary for reading.

1.2 How to get help

Most questions about individual functions should be answered by the documentation. For example, if you want to know more about `preparePairs`, you can bring up the documentation by typing `?preparePairs` or `help(preparePET)` at the R prompt. Otherwise, try reading this guide or contacting one of the authors. Considered suggestions for improvements are appreciated.

1.3 A brief description of Hi-C

The Hi-C protocol was originally developed by Lieberman-Aiden et al. [2009]. It is used to study chromatin organization by identifying pairwise interactions between two distinct genomic loci. Briefly, chromatin is cross-linked and digested with a restriction enzyme. This releases chromatin complexes into solution, where each complex contains multiple restriction fragments corresponding to interacting loci. Overhangs are filled in with biotin-labelled nucleotides to form blunt ends.

Proximity ligation is performed whereby ligation between blunt ends in the same complex is favoured. Sonication of the ligated DNA is performed and the fragments of DNA containing ligation junctions are purified by a biotin pulldown. The ligation product is then subjected to paired-end sequencing. Mapping of the reads in each pair can identify the interacting

loci. Of course, some caution is required due to the presence of non-specific ligation between blunt ends in different complexes.

1.4 Quick start

A typical differential analysis of Hi-C data is described below. For simplicity, assume that the the BAM files have already been processed into index files in `input`. Let `design` contain the design matrix for this experiment. Also assume that the boundaries of the relevant restriction fragments are present in `fragments`.

```
> require(diffHic)
> data <- squareCounts(input, width=1e6, fragments=fragments)
> require(edgeR)
> keep <- aveLogCPM(asDGEList(data)) > 0
> data <- data[keep,]
> y <- asDGEList(data)
> require(csaw)
> y$offset <- normalize(data, type="loess")
> y <- estimateDisp(y, design)
> result <- glmQLFTest(y, design, robust=TRUE)
```

There are several components of the pipeline that shall be discussed:

1. converting BAM files to index files
2. counting read pairs into pairs of genomic bins
3. normalizing counts between libraries
4. modelling biological variability
5. testing for significant differences between groups

In the various examples for this guide, data will be used from two studies. The first dataset is smaller and examines chromatin structure in K562 and GM06990 cell lines [Lieberman-Aiden et al., 2009]. The second compares interactions between wild-type and cohesin-deficient murine neural stem cells [Sofueva et al., 2013]. Obviously, readers will have to modify the code for their own analyses.

Chapter 2

Preparing index files from BAM

2.1 Matching mapped reads to restriction fragments

The Hi-C protocol is based on ligation between restriction fragments, i.e., the genomic interval between adjacent restriction sites [Lieberman-Aiden et al., 2009]. Sequencing of the ligation product is performed to identify the interacting loci - or, more precisely, the two restriction fragments containing the interacting loci. The resolution of Hi-C data is inherently limited by the frequency of restriction sites and the size of the restriction fragments. Thus, it makes sense to report the read alignment location in terms of the restriction fragment to which that read was mapped. The boundaries of each restriction fragment can be obtained with the `cutGenome` function, as shown below for the human genome after digestion with the *Hin*dIII restriction enzyme (recognition site of `AAGCTT`, 5′ overhang of 4 bp).

```
> require(BSgenome.Hsapiens.UCSC.hg19)
> hs.frag <- cutGenome(BSgenome.Hsapiens.UCSC.hg19, "AAGCTT", 4)
> hs.frag

GRanges with 846225 ranges and 0 metadata columns:
                 seqnames         ranges strand
                    <Rle>      <IRanges>  <Rle>
       [1]           chr1 [    1, 16011]      *
       [2]           chr1 [16008, 24575]      *
       [3]           chr1 [24572, 27985]      *
       [4]           chr1 [27982, 30433]      *
       [5]           chr1 [30430, 32157]      *
       ...            ...            ...    ...
  [846221] chrUn_gl000249 [22854, 25904]      *
  [846222] chrUn_gl000249 [25901, 31201]      *
  [846223] chrUn_gl000249 [31198, 36757]      *
  [846224] chrUn_gl000249 [36754, 36891]      *
  [846225] chrUn_gl000249 [36888, 38502]      *
  ---
  seqlengths:
```

```
      chr1       chr2 ...  chrUn_gl000249
 249250621  243199373 ...           38502
```

The `preparePairs` function can be used to match the mapping location to the restriction fragment. This converts the exact mapping position of a read into the index of the corresponding restriction fragment in `hs.frag`. The resulting pairs of indices are stored in an index file using the HDF5 format [The HDF Group, 1997-2014]. The larger index is designated as the "anchor" whereas the smaller is the "target". Results for each pair of chromosomes are stored in a separate dataframe for efficient retrieval. This is demonstrated using Hi-C data from GM06990 cells.

```
> preparePairs("SRR027957.bam", fragments=hs.frag, file="SRR027957.h5", dedup=TRUE, minq=10)

$pairs
   total   marked filtered   mapped
 7068675   103547  1657440  5337807

$same.id
   dangling self.circle
     423211      135008

$singles
[1] 0

$chimeras
  total  mapped   multi invalid
2297481 1656730 1072926   69482
```

The function itself returns a list of diagnostics showing the number of read pairs that are lost for various reasons. Of particular note is the removal of reads that are potential PCR duplicates with `dedup=TRUE`. This requires marking of the reads beforehand using an appropriate program such as Picard's MarkDuplicates. Filtering on the minimum mapping quality score with `minq` is also recommended to remove spurious alignments.

Read pairs mapping to the same restriction fragment provide little information on interactions between fragments. Dangling ends are inward-facing read pairs that are mapped to the same fragment [Belton et al., 2012]. These are uninformative as they are usually formed from sequencing of the restriction fragment prior to ligation. Self-circles are outward-facing read pairs that are formed when two ends of the same restriction fragment ligate to one another. Interactions within a fragment cannot be easily distinguished from these self-circularization events. Both structures are removed to avoid downstream confusion.

2.2 A comment on chimeric read alignment

On occasion, sequencing is performed over the ligation junction between two restriction fragments. This means that the resulting read will be chimeric, i.e., containing sequences

from two distinct genomic loci. Such reads require careful alignment that favours mapping of the 5′ end. This is because the location of the 3′ end of a chimeric read is already provided by the mapping location of the 5′ end of the mate read. Such alignment can be achieved with a number of approaches such as iterative mapping [Imakaev et al., 2012] or read splitting [Seitan et al., 2013].

For `preparePairs`, chimeric reads can be handled by recording two separate alignments for each read. Hard clipping is used to denote the length trimmed from each sequence in each alignment, and to determine which alignment corresponds to the 5′ or 3′ end of the read. Only the 5′ end(s) will be used to determine the restriction fragment index for that read pair. The total number of chimeric read pairs will be reported, along with the number where 5′ ends or 3′ ends are mapped. Of course, the function will just work normally if the mapping location is only given for the 5′ end, e.g., as with iterative mapping.

The proportion of invalid chimeric pairs can be determined if both 5′ and 3′ end information is provided. Invalid pairs are those where the 3′ location of a chimeric read does not agree with the 5′ location of the mate. The invalid proportion can be used as an empirical measure of the mapping error rate - or, at least, the upper bound of the error rate, given that split alignments are shorter and more uncertain than those using the full read sequence. Invalid chimeric pairs can be discarded by setting `ichim=FALSE` in `preparePairs`. However, this is not recommended as mapping errors for short 3′ ends may result in invalidity and loss of the otherwise correct 5′ alignments.

2.3 Filtering and counting read pairs

2.3.1 Reprocessing index files to store counts

Multiple read pairs may be assigned to a given pair of restriction fragments. It is more efficient to store the count associated with each pair of fragment indices, rather than storing the indices for each pair of reads. This can be done with the `countPairs` function, which overwrites the existing index file with the reformatted count data by default. Alternatively, it can be directed to an alternative file path if the original data is to be reused.

```
> min.ingap <- 1000
> min.outgap <- 25000
> countPairs("SRR027957.h5", file.out="SRR027957_counted.h5",
+     max.length=600, min.ingap=min.ingap, min.outgap=min.outgap)

   total   length    ingap   outgap retained
 4779588   858203    93666    82131  3758082
```

The `max.length` argument removes read pairs where the inferred length of the sequencing fragment (i.e., the ligation product) is greater than a specified value. The length of the sequencing fragment is inferred by summing the distance between the mapping location of

the 5′ end of each read and the nearest restriction site on the 3′ end of that read. Excessively large lengths are indicative of offsite cleavage, i.e., where the restriction enzyme or some other agent cuts the DNA at a location other than the restriction site. While not completely uninformative, these are discarded as they are not expected from the Hi-C protocol. The threshold value can be chosen based on the size selection interval in library preparation.

The `min.ingap` paramater removes inward-facing intra-chromosomal read pairs where the distance between the two reads on the linear chromosome is less than the specified value. The `min.outgap` parameter does the same for outward-facing intra-chromosomal read pairs. This is designed to remove dangling ends or self-circles involving undigested restriction fragments [Jin et al., 2013]. Such read pairs are technical artifacts but would not be removed in `preparePairs` as they would be mapped to different restriction fragments.

In all cases, filtering is performed on the read pairs in the input file prior to counting. The return value of `countPairs` simply contains the number of read pairs that were removed for the various reasons described above. The number of read pairs remaining is also recorded.

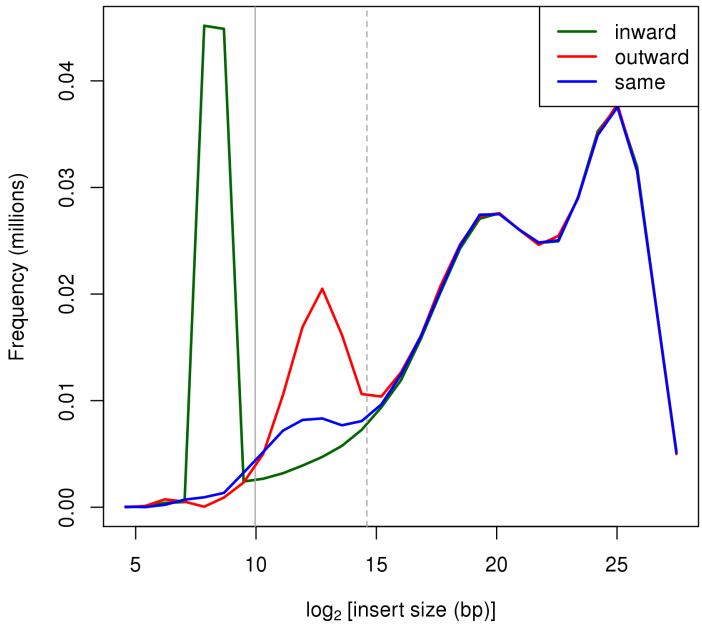
2.3.2 Setting parameter values with strand orientation plots

The strand orientation for a read pair refers to the combination of strands for the anchor/target reads. These are stored as flags where setting `0x1` and/or `0x2` means that the anchor or target reads, respectively, are mapped on the reverse strand. If different pieces of DNA were randomly ligated together, one would expect to observe equal proportions of all strand orientations. This can be tested by examining the distribution of strand orientations for inter-chromosomal read pairs with `getPairData`. A roughly equal number of read pairs are present for each strand orientation, which is expected as different chromosomes represent different pieces of DNA.

```
> diags <- getPairData("SRR027957.h5")
> intra <- !is.na(diags$gap)
> table(diags$orientation[!intra])

     0      1      2      3
750007 746596 746616 746497
```

The artificial nature of adjacent intra-chromosomal read pairs can be examined with a plot of gap distances for each strand orientation [Jin et al., 2013]. The two same-strand distributions are averaged for convenience. At high gap distances, the distributions will converge for all strand orientations. This is consistent with random ligation between two separate restriction fragments. At lower gap distances, spikes are observed in the ouward- and inward-facing distributions due to self-circularization and dangling ends, respectively. Thresholds should be chosen in `countPairs` to remove these spikes, as represented by the grey lines.

```
> llgap <- log2(diags$gap + 1L)
> intra <- !is.na(llgap)
> breaks <- seq(min(llgap[intra]), max(llgap[intra]), length.out=30)
> inward <- hist(llgap[diags$orientation==1L], plot=FALSE, breaks=breaks)
> outward <- hist(llgap[diags$orientation==2L] ,plot=FALSE, breaks=breaks)
> samestr <- hist(llgap[diags$orientation==0L | diags$orientation==3L], plot=FALSE, breaks=breaks)
> samestr$counts <- samestr$counts/2
> ymax <- max(inward$counts, outward$counts, samestr$counts)/1e6
> xmax <- max(inward$mids, outward$mids, samestr$mids)
> xmin <- min(inward$mids, outward$mids, samestr$mids)
> plot(0,0,type="n", xlim=c(xmin, xmax), ylim=c(0, ymax),
+     xlab=expression(log[2]~"[insert size (bp)]"), ylab="Frequency (millions)")
> lines(inward$mids, inward$counts/1e6, col="darkgreen", lwd=2)
> abline(v=log2(min.ingap), col="darkgrey")
> lines(outward$mids, outward$counts/1e6, col="red", lwd=2)
> abline(v=log2(min.outgap), col="darkgrey", lty=2)
> lines(samestr$mids, samestr$counts/1e6, col="blue", lwd=2)
> legend("topright", c("inward", "outward", "same"), col=c("darkgreen", "red", "blue"), lwd=2)
```



It is possible that these spikes reflect some genuine aspect of chromatin organization. For example, the spike in the outward-facing read pairs may be the result of systematic looping in chromosomal packaging. Removal would then result in loss of power to detect these features. Nonetheless, filtering is recommended as any genuine interactions are likely

to be dominated by technical artifacts. Indeed, one would still expect a variety of strand orientations for non-random ligation events generated across all genuine interactions.

2.4 Merging technical replicates

Hi-C experiments often involve deep sequencing as read pairs are sparsely distributed across the possible interactions. As a result, multiple index files may be present for multiple technical replicates of a single Hi-C library. These can be merged together using the `mergePairs` function prior to downstream processing. This is equivalent to summing the counts for each pair of restriction fragment indices, and is valid if one assumes Poisson sampling for each sequencing run [Marioni et al., 2008]. An example is provided with technical replicates for GM06990 cells in the Lieberman-Aiden et al. dataset.

```
> prepped <- preparePairs("SRR027958.bam", hs.frag, file="SRR027958.h5", dedup=TRUE, minq=10)
> counted <- countPairs("SRR027958.h5", "SRR027958_counted.h5", max.length=600,
+     min.ingap=min.ingap, min.outgap=min.outgap)
> mergePairs(files=c("SRR027957_counted.h5", "SRR027958_counted.h5"), "merged.h5")
```

Any Hi-C dataset that is processed manually by the user can be stored in an index file using the `savePairs` function. This takes a dataframe with anchor and target indices, as well as any additional information that might be useful. The idea is to allow entry into the diffHic analysis from other pipelines. If the dataset is too large, one can save chunks at a time before merging them all together with `mergePairs`.

```
> anchor.id <- as.integer(runif(100, 1, length(hs.frag)))
> target.id <- as.integer(runif(100, 1, length(hs.frag)))
> dummy <- data.frame(anchor.id, target.id, other.data=as.integer(runif(100, 1, 100)))
> savePairs(dummy, "example.h5", hs.frag)
```

Chapter 3

Counting read pairs into interactions

3.1 Overview

Prior to any statistical analysis, the Hi-C data must be summarized in terms of counts for each interaction. This acts as an experimental measure of the interaction intensity. Each interaction is parameterized by two genomic intervals representing the interacting loci. The count for that interaction is defined as the number of read pairs with one read mapping to each of the intervals. This must be performed for each sample in the dataset, such that each interaction is associated with a set of counts.

It is also worth defining the concept of the interaction space. This refers to the genome-by-genome space over which read pairs are distributed, i.e., all index pairs (x, y) for $x, y \in [1..N]$ where $x \geq y$ and N is the number of restriction fragments. Rectangular areas in the interaction space represent interactions between the genomic intervals spanned by the sides of the rectangle. The number of read pairs in each area is used as the count for the corresponding interaction. Non-rectangular areas also represent interactions, but these are more difficult to interpret and will not be considered here.

The examples shown here will use the Sofueva et al. dataset. Thus, some work is required to obtain the restriction fragment coordinates for the *Hin*dIII-digested mouse genome.

```
> require(BSgenome.Mmusculus.UCSC.mm10)
> mm.frag <- cutGenome(BSgenome.Mmusculus.UCSC.mm10, "AAGCTT", 4)
> input <- c("merged_flox_1.h5", "merged_flox_2.h5", "merged_ko_1.h5", "merged_ko_2.h5")
```

3.2 Counting into bin pairs

3.2.1 Overview

Here, the genome is partitioned into contiguous non-overlapping bins of constant size. Each interaction is defined as a pair of these bins. This approach avoids the need for prior knowl-

edge when summarizing Hi-C counts for each interaction. Counting of read pairs between bin pairs can be achieved using the `squareCounts` function.

```
> bin.size <- 1e6
> data <- squareCounts(input, mm.frag, width=bin.size, filter=1)
> data

DIList object for 4 libraries with 3319104 pairs across 2739 regions

Counts:
      [,1]  [,2]  [,3]  [,4]
[1,]    83    48    33    19
[2,] 21332 17151 12894 12357
[3,]    20    20    17     6
[4,]  8215  7023  4399  4237
[5,] 14729 12460  8984  8443
... and 3319099 more rows

Totals:
[1] 85786442 74685222 60860615 54596195

Anchors:
   Chr   Start     End
1 chr1 3004106 4000741
2 chr1 3004106 4000741
3 chr1 4000738 5001375
4 chr1 4000738 5001375
5 chr1 4000738 5001375
... and 3319099 more rows

Targets:
   Chr   Start     End
1 chr1       1 3004109
2 chr1 3004106 4000741
3 chr1       1 3004109
4 chr1 3004106 4000741
5 chr1 4000738 5001375
... and 3319099 more rows
```

This generates a `DIList` object containing the relevant information. Each row of the count matrix represents the counts for an interaction, while each column represents a library. Each interaction is characterized as a pair of genomic intervals, i.e., bins. Again, anchor and target notation is used for these intervals, whereby the anchor bin is that with the higher genomic coordinate. The total vector just contains the total number of read pairs in each library.

Bin pairs can also be filtered to remove those with to a count sum below `filter`. This removes uninformative bin pairs with very few read pairs, and reduces the memory footprint of the function. A higher value of `filter` may be necessary for analyses of large datasets with limited memory. More sophisticated filtering strategies are discussed in Section 4.

3.2.2 Choosing a bin width

The `width` of the bin is specified in base pairs and determines the spatial resolution of the analysis. Smaller bins will have greater resolution as adjacent features can be distinguished in the interaction space. Larger bins will have greater counts as a larger area is used to collect reads. Optimal summarization will not be achieved if bins are too small or too large for the features of interest.

Determination of the ideal bin size is not trivial as the features of interest are not usually known in advance. Instead, repeated analyses with multiple bin sizes is recommended to provide some degree of robustness (see Section 7.2.4). In practice, the boundary of each bin is rounded to the closest restriction site on the genome. This is due to the inherent limits on spatial resolution in a Hi-C experiment. The number of restriction fragments in each bin is recorded in the `nfrags` field of the metadata.

```
> head(regions(data))

GRanges with 6 ranges and 1 metadata column:
      seqnames             ranges strand |    nfrags
         <Rle>          <IRanges>  <Rle> | <integer>
  [1]     chr1 [      1, 3004109]      * |         1
  [2]     chr1 [3004106, 4000741]      * |       389
  [3]     chr1 [4000738, 5001375]      * |       334
  [4]     chr1 [5001372, 5997485]      * |       340
  [5]     chr1 [5997482, 7000260]      * |       342
  [6]     chr1 [7000257, 8000015]      * |       349
  ---
  seqlengths:
                   chr1                 chr2 ...       chrUn_JH584304
              195471971            182113224 ...               114452
```

3.3 Counting with pre-defined regions

On occasion, prior knowledge may be available with respect to the regions of interest. For example, a researcher may be interested in examining interactions between gene bodies. The coordinates can be easily obtained from existing annotation, as shown below for the mouse genome. Other pre-specified regions can be used such as those of known enhancers or protein binding sites.

```
> require(org.Mm.eg.db)
> chrstart <- toTable(org.Mm.egCHRLOC)
> chrend <- toTable(org.Mm.egCHRLOCEND)
> gene.body <- GRanges(paste0("chr", chrstart$Chromosome),
+     IRanges(abs(chrstart$start_location), abs(chrend$end_location)))
```

Counting can be directly performed for these defined regions using the `connectCounts` function. Interactions are defined between each pair of regions in the pre-specified set. This can be easier to interpret than pairs of bins as the interacting regions have some biological significance. The count matrix and the vector of totals are defined as previously described.

```
> redata <- connectCounts(input, fragments=mm.frag, regions=gene.body)
> redata

DIList object for 4 libraries with 16266783 pairs across 27294 regions

Counts:
     [,1] [,2] [,3] [,4]
[1,] 9540 7469 5967 5739
[2,]  701  604  329  334
[3,] 1182  933  835  799
[4,]  127  134   68   52
[5,]  424  315  278  274
... and 16266778 more rows

Totals:
[1] 85786442 74685222 60860615 54596195

Anchors:
   Chr   Start     End
1 chr1 3211067 3675240
2 chr1 4286646 4409818
3 chr1 4286646 4409818
4 chr1 4340420 4362532
5 chr1 4340420 4362532
... and 16266778 more rows

Targets:
   Chr   Start     End
1 chr1 3211067 3675240
2 chr1 3211067 3675240
3 chr1 4286646 4409818
4 chr1 3211067 3675240
5 chr1 4286646 4409818
... and 16266778 more rows
```

Again, anchor and target notation applies where the interval with the larger start coordinate in the genome is defined as the anchor. Note that the anchor may not have a larger end coordinate if the supplied `regions` are nested. In addition, each region is rounded to the nearest restriction site. Resorting is also performed though the indices of the original regions is supplied in the metadata as `original`.

```
> head(regions(redata))
```

```
GRanges with 6 ranges and 2 metadata columns:
      seqnames             ranges strand |    nfrags  original
         <Rle>          <IRanges>  <Rle> | <integer> <integer>
  [1]     chr1 [3211067, 3675240]      * |       197     24086
  [2]     chr1 [4286646, 4409818]      * |        49      4725
  [3]     chr1 [4340420, 4362532]      * |         8      4724
  [4]     chr1 [4487488, 4498343]      * |         2      5193
  [5]     chr1 [4772601, 4786029]      * |         3      6667
  [6]     chr1 [4802092, 4847111]      * |         8      4156
  ---
  seqlengths:
                   chr6                chr11 ... chr5_JH584297_random
                     NA                   NA ...                    NA
```

One obvious limitation of this approach is that interactions involving unspecified regions will be ignored. This is obviously problematic when searching for novel interacting loci. Another issue is that the width of the regions cannot be easily changed. This means that the compromise between spatial resolution and count size cannot be tuned. For example, interactions will not be detected around smaller genes as the counts will be too small. Conversely, interactions between distinct loci within a single large gene body will not be resolved.

3.4 Counting into single bins

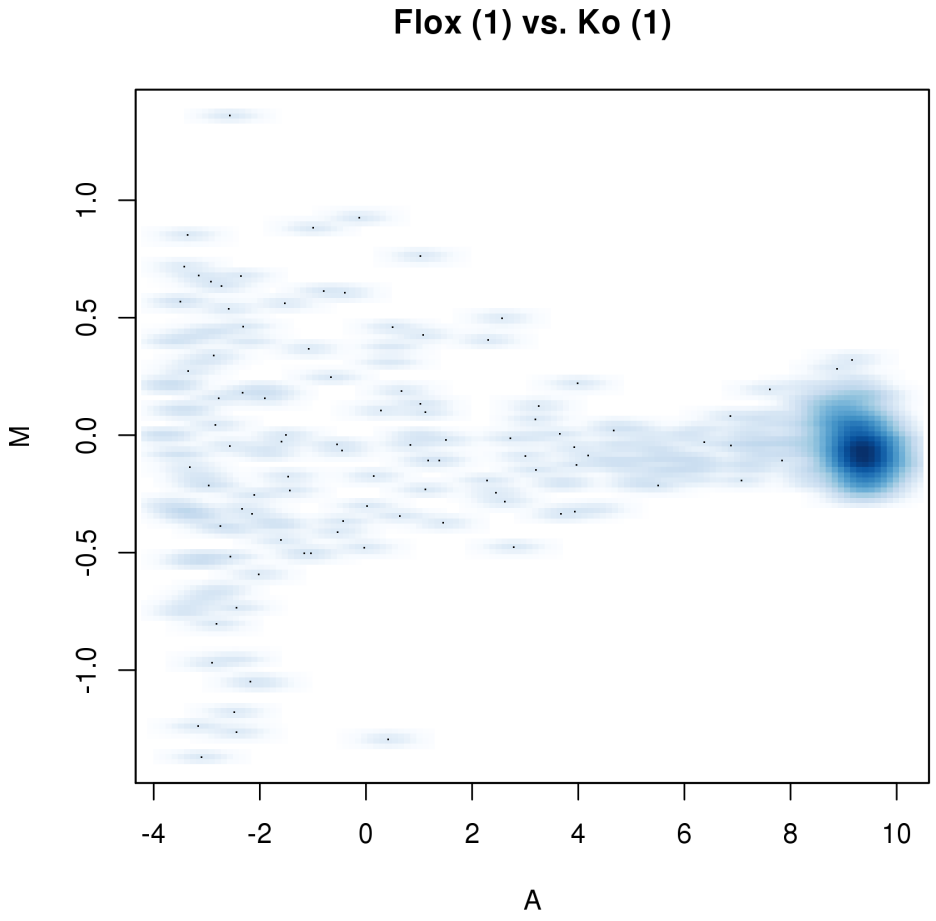
For each bin, the number of read pairs with at least one read mapped inside that bin can be counted with the `marginalCounts` function. This effectively uses the Hi-C data to examine the genomic coverage of each bin. One can use these "marginal" counts to determine whether there are systematic differences in coverage between libraries for a given bin. This implies that copy number variations are present, which may confound detection of differential interactions.

```
> margin.data <- marginCounts(input, fragments=mm.frag, width=bin.size)
> margin.data

DIList object for 4 libraries with 2732 pairs across 2739 regions

Counts:
      [,1]  [,2]  [,3]  [,4]
[1,]   203   128   106    82
[2,] 78448 65876 57030 54474
[3,] 72665 63262 52369 48886
[4,] 68766 59567 49858 46784
[5,] 70338 62625 53813 50920
... and 2727 more rows

Totals:
[1] 85786442 74685222 60860615 54596195
```

```
Anchors:
   Chr   Start     End
1 chr1       1 3004109
2 chr1 3004106 4000741
3 chr1 4000738 5001375
4 chr1 5001372 5997485
5 chr1 5997482 7000260
... and 2727 more rows

Targets:
   Chr   Start     End
1 chr1       1 3004109
2 chr1 3004106 4000741
3 chr1 4000738 5001375
4 chr1 5001372 5997485
5 chr1 5997482 7000260
... and 2727 more rows
```

For this dataset, there are no changes in coverage for the vast majority of bins. The largest fold changes occur at low abundances and are likely to be imprecise. The lack of changes indicate that a direct comparison of the interaction intensities will be valid. Remedial action in the presence of possible copy number variations is not trivial and will be discussed in Section 5.3.

```
> adjc <- cpm(asDGEList(margin.data), log=TRUE, prior.count=5)
> smoothScatter(0.5*(adjc[,1]+adjc[,3]), adjc[,1]-adjc[,3],
+     xlab="A", ylab="M", main="Flox (1) vs. Ko (1)")
```



To avoid any confusion in the notation, the count for each bin pair is defined as the interaction count. It may be necessary later to process the interaction count based on the marginal counts for the corresponding bins in the pair. In such cases, one should ensure that the same `width` (and other parameters) is used in `squareCounts` and `marginCounts`. One can check that this is the case by ensuring that the regions are the same.

```
> identical(regions(data), regions(margin.data))
```

```
[1] TRUE
```

Also note that, technically speaking, the anchor and target notation here is superfluous. This is because the counts refer to bins on the linear genome, rather than pairs of bins in the interaction space. Nonetheless, a `DIList` is still returned by `marginCounts` for convenience and consistency.

3.5 Summary

Counting into bin pairs is the most general method for interaction quantification. It does not require any prior knowledge regarding the regions of interest. The bin size can be easily adjusted to obtain the desired spatial resolution. It is also easier/safer to compare between bin pairs (e.g., during filtering) when each bin is roughly of the same size. Thus, bin-based counting will be the method of choice for the rest of this guide.

Chapter 4

Filtering out uninteresting interactions

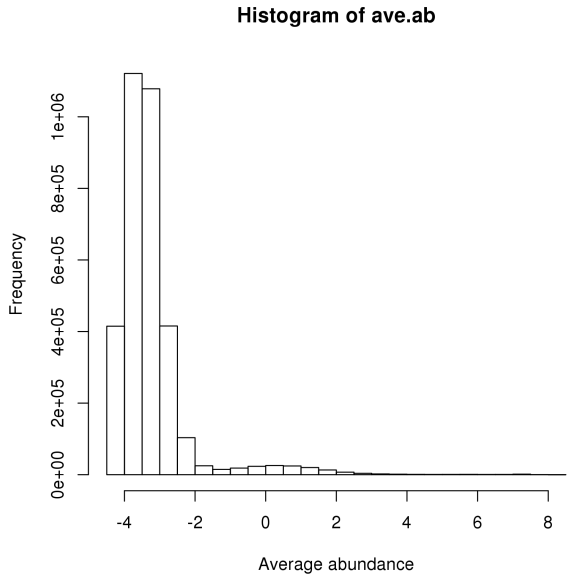
4.1 Overview

4.1.1 Computing the average abundance in a NB model

Filtering is often performed to remove uninteresting features in analyses of high-throughput experiments. This reduces the severity of the multiple testing correction and increases detection power among the remaining tests. The filter statistic should be independent of the p-value under the null hypothesis, but correlated to the p-value under the alternative [Bourgon et al., 2010]. The aim is to enrich for false nulls without affecting type I error for the true nulls.

Assume that the counts for each bin pair are sampled from the negative binomial (NB) distribution. Here, the average count across all libraries is (probably) an independent filter statistic. This is also called the average abundance and can be computed using the `average` function [McCarthy et al., 2012]. The calculation is demonstrated below, after adding a prior count to stabilize the estimates at low counts.

```
> require(edgeR)
> prior.count <- 3
> ave.ab <- aveLogCPM(asDGEList(data), prior.count=prior.count)
> hist(ave.ab, xlab="Average abundance")
```



Bin pairs with an average abundance less than a certain threshold can be discarded. At the very least, the threshold should be chosen to screen out bin pairs with very low absolute counts. This is because these bin pairs will never have sufficient evidence to reject the null hypothesis. The example below removes those bin pairs with an average count below 5 across all libraries. Note that a prior count is added for a valid comparison with the computed abundances.

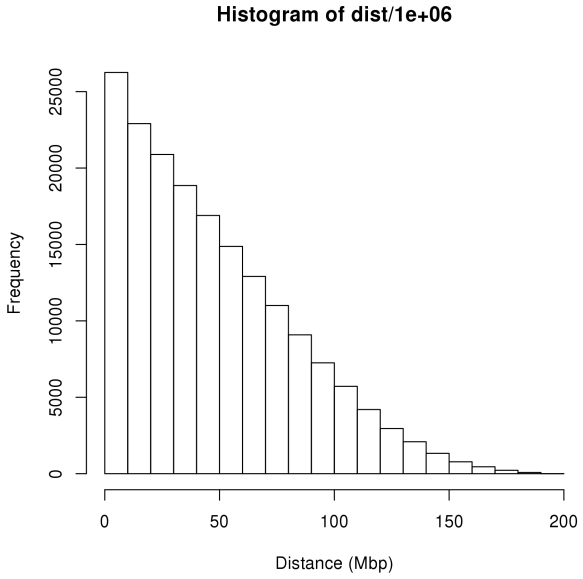
```
> count.keep <- ave.ab >= log2(5 + prior.count) - mean(log2(totals(data)/1e6))
> summary(count.keep)

   Mode   FALSE    TRUE    NA's
logical 2478528  840576       0

> dummy <- data[count.keep,]
```

This count-based approach is fairly objective yet is still effective, i.e., removes a large number of bin pairs. More sophisticated strategies can be implemented where the choice of threshold is motivated by some understanding of the Hi-C protocol. These strategies are described in the rest of this chapter, and will be combined with the count-based filter to maintain a minimum count size among the retained bin pairs.

4.1.2 Computing the interaction distance

The linear distance between each pair of the interacting bins is worth mentioning as it can also be used in filtering. This is obtained using the `getDistance` function, which computes the distance between the bin midpoints by default.

```
> dist <- getDistance(data, type="mid")
> hist(dist/1e6, xlab="Distance (Mbp)")
```



Inter-chromosomal bin pairs are marked here as `NA` for completeness. These tend to dominate the output as they constitute most of the interaction space. Of course, the majority of these bin pairs will have low counts due to the sparseness of the data.

```
> summary(is.na(dist))
```

```
   Mode   FALSE    TRUE    NA's
logical  178758 3140346       0
```

4.2 Directly removing low-abundance interactions

The simplest definition of an "uninteresting" interaction is that resulting from non-specific ligation. These are represented by low-abundance bin pairs where no underlying interaction is present to drive ligation events between the corresponding bins. Any changes in the counts for these bin pairs are not interesting and are ignored. In particular, the filter threshold can be defined by mandating some minimum fold change above the level of non-specific ligation.

The magnitude of non-specific ligation can be estimated from the data by assuming that most inter-chromosomal contacts are not genuine. This is reasonable given that most chromosomes are arranged in self-interacting territories [Bickmore, 2013]. The median normalized abundance across the inter-chromosomal bin pairs is used as the estimate of the non-specific ligation rate. The threshold is then defined by requiring a minimum fold change of 10 above the non-specific estimate.

```
> direct.threshold <- median(ave.ab[is.na(dist)], na.rm=TRUE)
> direct.keep <- count.keep & ave.ab > log2(10) + direct.threshold
> summary(direct.keep)

   Mode   FALSE    TRUE    NA's
logical 3207706  111398       0
```

The `direct.keep` vector can then be applied for filtering of `data`, as previously shown with `count.keep`. This approach is named here as "direct" filtering, as the average count is directly compared against a fixed threshold value. Note that the construction of `direct.keep` is based off `count.keep` to ensure that retained bins have large absolute counts.

4.3 Filtering as a function of interaction distance

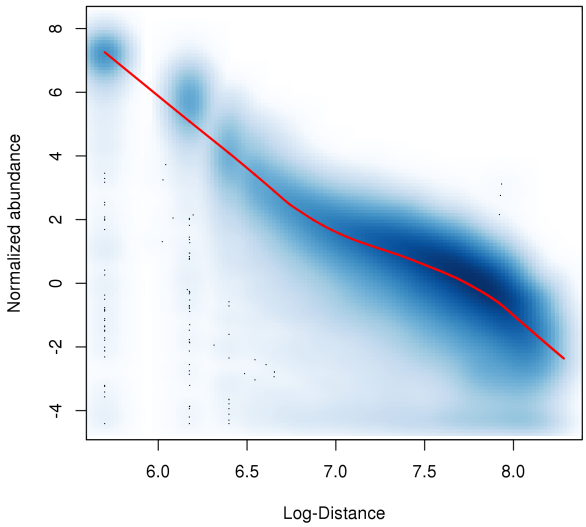
A more complex filter adjusts the threshold according to the distance between the bins in each bin pair. Larger counts are observed at lower distances, suggesting that a concomitantly higher threshold is necessary. This expands the definition of "uninteresting" events to include those interactions that are formed by simple compaction of chromatin [Lin et al., 2012].

In this strategy, a trend is first fitted to the normalized abundance for all intra-chromosomal bin pairs using the log-distance as the covariate. Half of the bin size is added as a prior, to avoid undefined values when distances are zero. Inter-chromosomal bin pairs will not be involved in trend fitting as the distance will be `NA`. Nonetheless, the fitted value for these bin pairs is set to the fitted value of the largest intra-chromosomal distance.

```
> log.dist <- log10(dist + bin.size/2)
> trend.threshold <- loessFit(x=log.dist, y=ave.ab)$fitted
> trend.threshold[is.na(log.dist)] <- trend.threshold[which.max(log.dist)]
```

The effect of this strategy can be visualized by plotting the interaction distance against the normalized abundance. A power-law relationship between distance and abundance is typically observed in Hi-C data [Lieberman-Aiden et al., 2009]. The filter threshold decreases as the distance between the interacting loci increases.

```
> smoothScatter(log.dist, ave.ab, xlab="Log-Distance", ylab="Normalized abundance")
> o <- order(log.dist)
> lines(log.dist[o], trend.threshold[o], col="red", lwd=2)
```



The assumption here is that the majority of interactions are generated by non-specific packaging of the linear genome. Each bin pair is only retained if its abundance is greater than the corresponding fitted value, i.e., above that expected from compaction. This favours selection of bin pairs corresponding to long-range interactions.

```
> trend.keep <- count.keep & ave.ab > trend.threshold
> summary(trend.keep)
```

```
   Mode   FALSE    TRUE    NA's
logical 3154317  164787       0
```

4.4 Computing filter thresholds with limited memory

These filtering procedures assume that no filtering has been performed during count loading with `squareCounts`, i.e., `filter=1`. Any pre-filtering that removes low-abundance bin pairs will lead to overestimation of the filter thresholds. However, it may not be practical to load counts without pre-filtering, as too many non-empty bin pairs may be present. In such cases, two options are available:

- Pick an arbitrary threshold and use it to directly filter on the average count. This is the simplest approach and can be justified from a minimum allowable count for detection of differences.
- Load counts and perform filtering for larger bin sizes. This has lower memory requirements as the interaction space is partitioned into fewer bin pairs. Then, convert the computed filter thresholds into values for the original bin sizes.

An example of the second option is shown here. For this section, imagine that a bin size of 100 kbp is actually of interest, but filter thresholds can only be efficiently computed with 1 Mbp bins. The first task is to match up the smaller bin pairs with the larger bin pairs using the `boxPairs` function.

```
> deflation <- 10
> smaller.data <- squareCounts(input, fragments=mm.frag, width=bin.size/deflation, filter=50)
> matched <- boxPairs(reference=bin.size, larger=data, smaller=smaller.data, fragments=mm.frag)
> small2large <- match(matched$indices$smaller, matched$indices$larger)
```

The threshold for each 100 kbp bin pair can be calculated from the threshold of the larger 1 Mbp bin pair in which it is nested. However, the thresholds must be adjusted to account for the differences in the read counting area of the interaction space. A quick and dirty adjustment can be obtained by squaring the ratio of the bin sizes. An example is shown below, using the thresholds from trend-based filtering.

```
> new.threshold <- trend.threshold[small2large] - 2*log2(deflation)
```

The average abundance must also be computed with some consideration of the differences in bin size. Specifically, the prior count must be scaled down before addition. This avoids spurious differences in the abundances of small and large bin pairs after downscaling of the latter. If the same prior is used, downscaling of the abundances for large bin pairs will reduce the size of the effective prior added to those counts.

```
> small.ab <- aveLogCPM(asDGEList(smaller.data),
+     prior.count=prior.count/deflation^2)
```

Finally, filtering can be performed by comparing the abundances of the smaller bin pairs to the adjusted threshold. Though more bin pairs are retained, remember that the bins themselves are smaller than those used in the rest of the examples. Thus, the retained area of the interaction space is actually lower. Also note that a non-unity `filter` is used in `squareCounts`, so no additional filtering is necessary to enforce a minimum count size.

```
> small.keep <- small.ab > new.threshold
> summary(small.keep)

   Mode   FALSE    TRUE    NA's
logical  125118  326079       0

> smaller.data <- smaller.data[small.keep,]
```

The same procedure can be used in direct filtering by replacing `trend.threshold[...]` with `direct.threshold` when computing `new.threshold`. In this case, matching is not required as the threshold is constant for all bin pairs.

4.5 Filtering for pre-specified regions

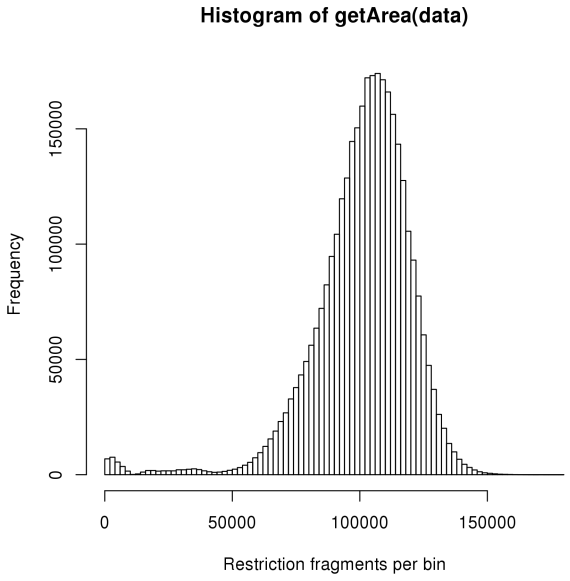
The methods described above for calculation of filter thresholds rely on bin pair counts. Filtering for arbitrary regions is slightly difficult as the width of the regions is not guaranteed to be reasonably constant. One must adjust the threshold for the different read counting areas, as shown below with the `getArea` function.

```
> re.ab <- aveLogCPM(asDGEList(redata))
> re.norm.ab <- re.ab - log2(getArea(redata, fragments=mm.frag))
> summary(re.norm.ab)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-22.070 -15.280 -13.450 -13.360 -11.560  -1.539
```

Filtering can then be performed with `re.norm.ab`, in the same manner as described for the bin pairs. However, users should keep in mind that only a subset of the interaction space is counted here. Calculation of the filter thresholds may be inaccurate, e.g., if only high-abundance areas are counted. Similarly, threshold estimation may be imprecise if not enough areas are non-empty.

On a related note, filtering with bin pairs should technically be performed after adjusting for the area of each bin pair. This is because some bins may match up to fewer restriction fragments, despite all bins having similar base pair widths. In practice, this is unnnecessary for large bins as the vast majority will contain similar numbers of restriction fragments. Normalization like that for `re.norm.ab` above tends to select for bins with very few fragments, e.g., telomeres, centromeres.

```
> hist(getArea(data), breaks=100, xlab="Restriction fragments per bin")
```



Note that area-based adjustment adjustment fails to account for other differences between bins, e.g., in mappability or sequenceability. Full consideration of these bin-specific biases requires methods such as iterative correction (see Section 5.2).

4.6 Summary of the filtering strategies

Each filtering strategy is arbitrarily tunable, as one can simply increase or decrease the minimum fold change required for retention. Nonetheless, they are still useful as they can guide the user towards a sensible interpretation of the filter threshold. This would be possible if the threshold value was just arbitrarily chosen. The effect of each approach can also be summarized by examining the distribution of interaction distances after filtering.

```
> summary(dist[direct.keep]/1e6)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
   0.00   11.00   25.01   30.50   45.00  163.00     108

> summary(dist[trend.keep]/1e6)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
   0.00   19.00   41.00   48.18   71.00  190.70   74515
```

As expected, the direct method preferentially retains short-range interactions whereas the trended method selects for long-range interactions. The choice of filtering method depends on the features that are most likely to be of interest in each analysis. A tentative recommendation is provided for retention of short-range interactions, given that short-range packaging dominates genomic organization. On a practical level, the simplicity of the direct approach is attractive and will be used throughout the rest of the guide.

```
> original <- data
> data <- data[direct.keep,]
```

The original unfiltered data is also retained for later use. In general, less aggressive filtering should be performed if the features of interest are not clearly defined.

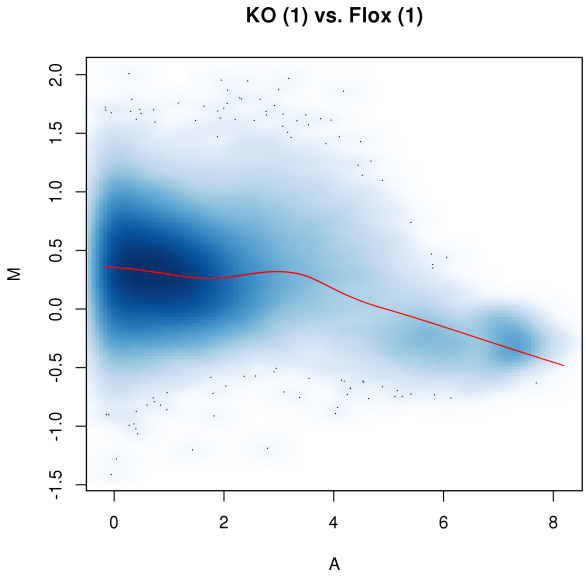
Chapter 5

Normalization strategies for Hi-C data

5.1 Removing trended biases between libraries

Library-specific biases can be generated from uncontrolled differences in library preparation. This is particularly problematic for Hi-C data given the complexity of the protocol. Changes in cross-linking efficiency or ligation specificity can lead to redistribution of read pairs throughout the interaction space. Such technical differences may manifest as a trended bias across bin pairs in a MA plot.

```
> ab <- aveLogCPM(asDGEList(data))
> o <- order(ab)
> adj.counts <- cpm(asDGEList(data), log=TRUE)
> mval <- adj.counts[,3]-adj.counts[,1]
> smoothScatter(ab, mval, xlab="A", ylab="M", main="KO (1) vs. Flox (1)")
> fit <- loessFit(x=ab, y=mval)
> lines(ab[o], fit$fitted[o], col="red")
```



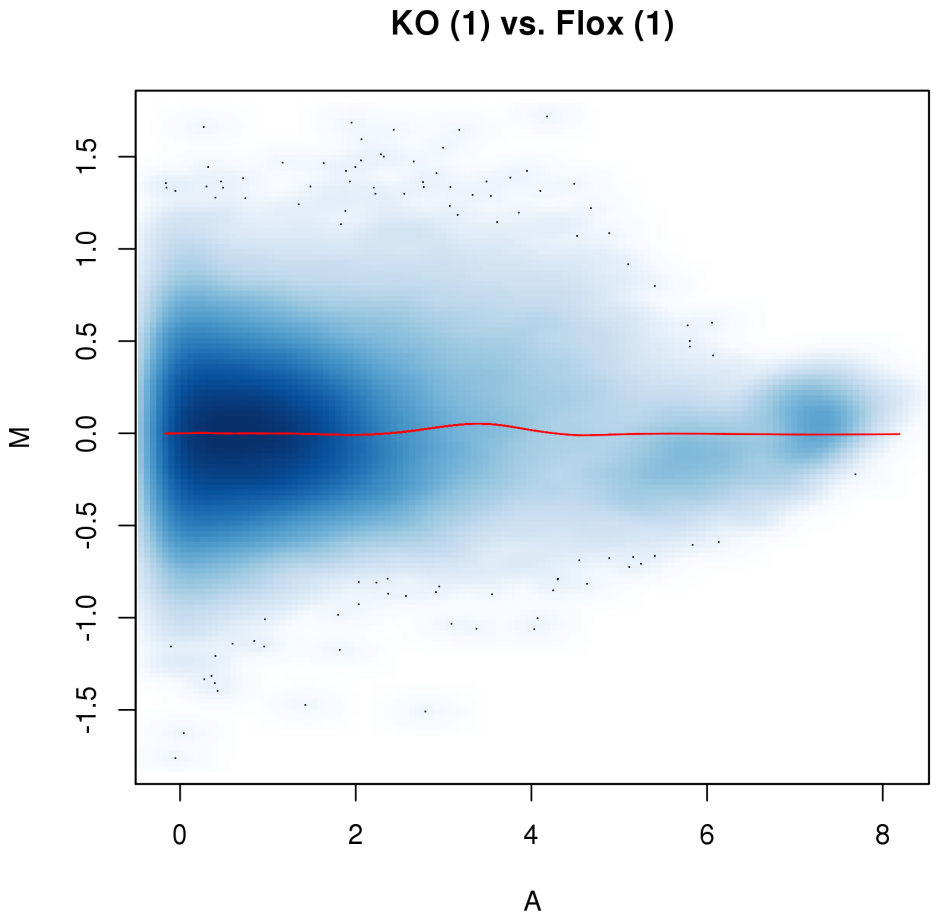
Trended biases are problematic as they can inflate the variance estimates or fold-changes for some bin pairs. Thus, they must be eliminated with non-linear normalization. Here, the NB-loess method is used to account for the discrete nature of the counts [**?**], based on the `normalizeCounts` function in the csaw package. This computes an offset term for each bin pair in each library for use in a generalized linear model (GLM). A large offset for an observation is equivalent to downscaling the corresponding count relative to the counts of the other libraries.

```
> require(csaw)
> nb.off <- normalize(data, type="loess")
> head(nb.off)
```

```
          [,1]      [,2]       [,3]       [,4]
[1,] 0.3893146 0.2108572 -0.2544951 -0.3456766
[2,] 0.3050890 0.1543921 -0.1858283 -0.2736528
[3,] 0.3592914 0.1908865 -0.2301172 -0.3200607
[4,] 0.2272537 0.1020850 -0.1223381 -0.2070006
[5,] 0.2956056 0.1479464 -0.1780463 -0.2655056
[6,] 0.3604638 0.1916682 -0.2310706 -0.3210614
```

The MA plot can then be examined after adjusting the log-counts with the computed offsets. Most of the trend is removed which indicates that normalization was successful. Of course, this assumes that most bin pairs at each abundance are constant between libraries. Any systematic differences must be technical in origin and should be removed. However, if this assumption does not hold, removal of the trend may result in loss genuine differences between conditions.

```
> adj.counts <- log2(counts(data) + 0.5) - nb.off/log(2)
> mval <- adj.counts[,3]-adj.counts[,1]
> smoothScatter(ab, mval, xlab="A", ylab="M", main="KO (1) vs. Flox (1)")
> fit <- loessFit(x=ab, y=mval)
> lines(ab[o], fit$fitted[o], col="red")
```



5.2 Iterative correction of interaction intensities

While not the forte of diffHic, a method is also provided here for removal of biases between genomic regions. The `correctedContact` function performs the iterative correction procedure described by [Imakaev et al., 2012] with some modifications. Briefly, if multiple libraries are used to generate the `data` object, then correction is performed using the average count for each bin pair. Winsorizing through `winsor.high` is also performed to mitigate the impact of high-abundance bin pairs.

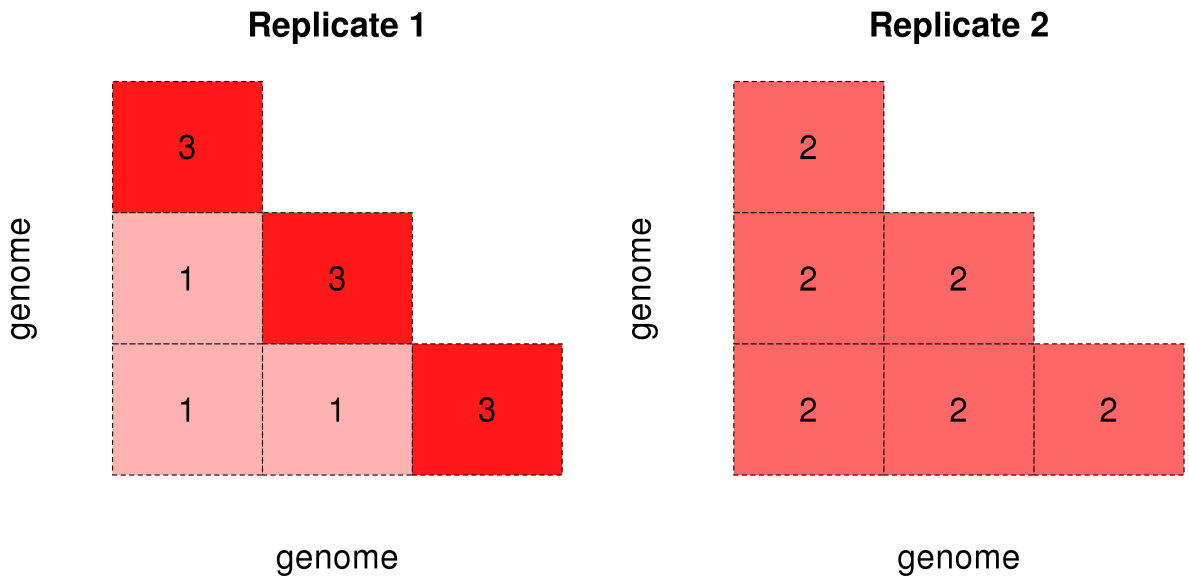
```
> corrected <- correctedContact(original, winsor.high=0.02, ignore.low=0.02)
> head(corrected$truth)

[1]         NA         NA 0.14653918         NA         NA 0.03264732
```

The returned `truth` contains the "true" contact probability for each bin pair in `data`. This accounts for differences in sequencibility, mappability, restriction site frequency, etc. between bins. Comparisons can then be directly performed between the contact probabilities of different bin pairs. Some `NA` values will be present due to the removal of low-abundance bins that do not exhibit stable behaviour during correction.

Note that `original` is used as no filtering should be performed prior to `correctedContact`. All non-empty bin pairs are needed for correction as information is collated across the entire interaction space. The convergence of the correction procedure can be checked by examining the maximum fold change to the truth at each iteration.

```
> corrected$max

 [1] 186.770783   8.028471   2.842165   1.703590   1.325088   1.171107
 [7]   1.099043   1.061208   1.039449   1.026089   1.017523   1.011879
[13]   1.008099   1.005541   1.003799   1.002608   1.001792   1.001232
[19]   1.000847   1.000583   1.000401   1.000276   1.000190   1.000130
[25]   1.000090   1.000062   1.000042   1.000029   1.000020   1.000014
[31]   1.000009   1.000007   1.000005   1.000003   1.000002   1.000002
[37]   1.000001   1.000001   1.000001   1.000000   1.000000   1.000000
[43]   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000
[49]   1.000000   1.000000
```

Of course, iterative correction only removes biases between different bins. It is not guaranteed to remove (trended) biases between libraries. For example, consider two replicates that have the same genomic biases. Assume that the Hi-C protocol was more efficient in the second replicate, such that more weak long-range interactions were captured (at the expense of the strong short-range interactions). This is visualized below with plots of the genome-by-genome interaction space for each replicate, where the counts represent the relative intensity of each interaction.



Interactions are shown between pairs of genomic intervals, based on the partitioning of each axis by the dotted lines. A fold change of 1.5 will be obtained at an average intensity of 2.5 for the diagonal elements, whereas a fold change of 0.5 will be obtained at an average intensity of 1.5 for all other elements. This mean-dependent fold change represents a trended bias that can be eliminated with non-linear methods. In contrast, the sum of counts for each genomic interval is the same for all intervals in each replicate ($1 + 3 + 1$ for the first and $2+2+2$ for the second). This means that iterative correction will have no effect as it operates on the differences in these sums within a single sample.

5.3 Accounting for copy number variations

As one might imagine, copy number variations (CNVs) in the interacting regions will affect the interaction intensity. These CNV-driven differences in intensities are generally uninteresting and must be removed to avoid spurious detection. This is done using the `normalizeCNV` function, which simultaneously removes the trended biases described earlier. The function is based on multi-dimensional smoothing across several covariates with the `locfit` package [Loader, 1999].

```
> cnv.offs <- normalizeCNV(data, margin.data)
> head(cnv.offs)
```

```
          [,1]      [,2]       [,3]       [,4]
[1,] 0.4290858 0.1569355 -0.2925458 -0.2934754
[2,] 0.3069194 0.1167794 -0.1934355 -0.2302633
[3,] 0.3709111 0.1866080 -0.2655925 -0.2919266
```

```
[4,] 0.1961233 0.0637020 -0.1238676 -0.1359576
[5,] 0.2787581 0.1384220 -0.1943861 -0.2227940
[6,] 0.3700657 0.1776827 -0.2630845 -0.2846639
```

For any two libraries, the ratio of marginal counts for a bin can be used as a proxy for the relative CNV between libraries in that bin. Each bin pair is subsequently associated with two marginal log-ratios. Note that the marginal counts should be collected with the same parameters as the bin pair (i.e., interaction) counts. The third covariate for each bin pair is that of the average abundance across all libraries. This will account for any abundance-dependent trends in the biases.

The response for each bin pair is defined as the log-ratio of interaction counts between two libraries. A locally weighted surface is fitted to the response against all three covariates. The idea is that, at any combination of covariate values, most bin pairs are not expected to be differential. Any systematic differences between libraries is attributable to CNV (or trended) biases and is removed. The function does so by returning GLM offsets, which can be supplied to the statistical analysis to eliminate the bias.

The use of a empirical trend fit reduces the number of assumptions involved in translating a CNV into its effect on interaction intensities. For example, iterative correction can also be used to correct for CNVs by accounting for differences in coverage of each bin. It does so by assuming factorizability [Imakaev et al., 2012], i.e., the effect on intensity is a product of the CNVs of the interacting regions. This may not be reliable given that random ligation is not the only mechanism for read pair generation.

That said, the use of many covariates may lead to overfitting and removal of genuine differences. The function also assumes that CNVs in different parts of the genome have the same effect. Caution is therefore required when using `normalizeCNV`. The safest choice is to avoid it when CNVs are not present, according to the diagnostic plots in Section 3.4.

Chapter 6

Assessing biological variability

6.1 Overview

The magnitude of biological variability can be determined from biological replicates, i.e., Hi-C libraries prepared from different biological samples. This reduces the significance of detected interactions when the data is highly variable. For count-based data, this can be achieved using the NB model in edgeR [Robinson et al., 2010]. Estimation of the NB dispersion parameter allows modelling of the variation between biological replicates. Similarly, estimation of the quasi-likelihood (QL) dispersion can be performed to account for heteroskedasticity [Lund et al., 2012].

Dispersion estimation requires fitting of a GLM to the counts for each interaction [McCarthy et al., 2012]. This means that a design matrix must be specified to describe the experimental setup. Here, a simple one-way layout is sufficient. Two groups are present, each of which contains two replicates. At this point, the aim is to compute the dispersion from the variability in counts within each group.

```
> design <- model.matrix(~factor(c("flox", "flox", "ko", "ko")))
> colnames(design) <- c("Intercept", "KO")
> design

  Intercept KO
1         1  0
2         1  0
3         1  1
4         1  1
attr(,"assign")
[1] 0 1
attr(,"contrasts")
attr(,"contrasts")$`factor(c("flox", "flox", "ko", "ko"))`
[1] "contr.treatment"
```

It is also necessary to assemble a `DGEList` object for entry into edgeR. Each feature of the

`DGEList` corresponds to an interaction - in this case, a pair of 1 Mbp bins. Note the inclusion of the normalization offsets that were previously computed with the NB-loess method.

```
> y <- asDGEList(data)
> y$offset <- nb.off
```

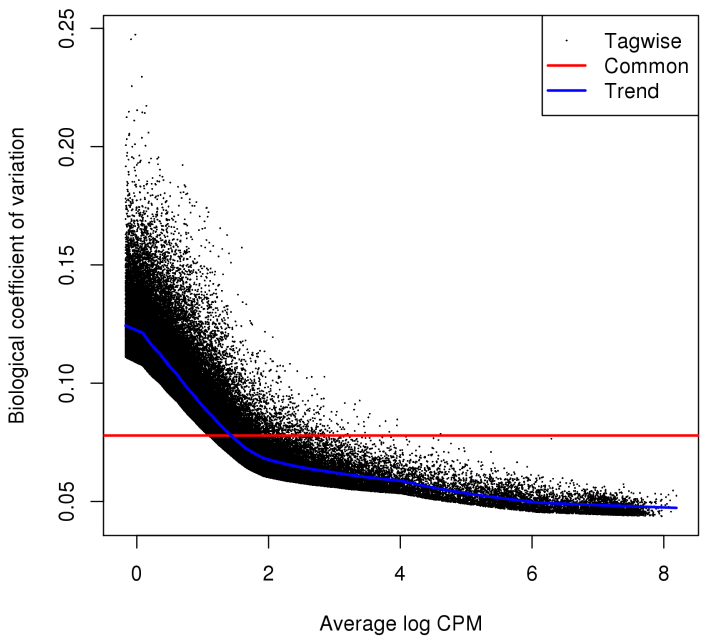
6.2 Estimating the NB dispersion

Estimation of the NB dispersion is performed by maximizing the Cox-Reid adjusted profile likelihood (APL) [McCarthy et al., 2012]. This adjusts the likelihood for the uncertainty in the estimated means. Of course, when replication is limited, there is not enough information per bin pair to estimate the dispersion. This is overcome by computing APLs across many bin pairs to stablize the estimates.

```
> y <- estimateDisp(y, design)
> y$common.dispersion

[1] 0.006065707
```

A more sophisticated strategy is also used whereby an abundance-dependent trend is fitted to the APLs. This should manifest as a smooth trend in the final NB dispersion estimates. The aim is to improve accuracy by empirically modelling any non-NB mean-variance relationships.

```
> plotBCV(y)
```



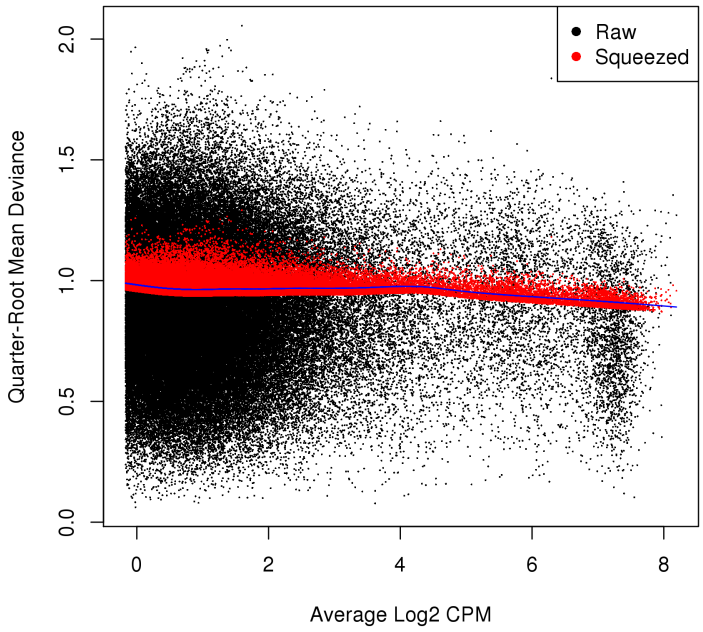
In most cases, the relationship should be monotonic decreasing as the counts become more precise with increasing size. Minor deviations are probably due to the imperfect nature of non-linear normalization. Major increases suggest that there are substantial batch effects that are still present. For example, a cluster of outliers indicates that there may be copy number changes between replicates.

6.3 Estimating the QL dispersion

The QL dispersion is estimated using the deviance estimate for GLMs. This may seem a bit superfluous given that the NB dispersion already accounts for biological variability. However, the QL dispersion can account for heteroskedasticity in the bin-pair-specific estimates. Estimation can be performed with the `glmQLFTest` function in edgeR.

```
> result <- glmQLFTest(y, design, robust=TRUE, plot=TRUE)
```

Again, there is not enough information for each bin pair to provide a precise estimate of the QL dispersion. Instead, information is shared between bin pairs using an empirical Bayes (EB) approach. Per-bin-pair estimates are shrunk towards the mean (trended) QL dispersion across all bin pairs. This stabilizes the estimates and improves precision for downstream applications.



The extent of the EB shrinkage is determined by the heteroskedasticity in the data. If the true dispersions are highly variable, shrinkage to a common value would be inappropriate. If the true dispersions are not variable, more shrinkage can be performed to increase precision. This is quantified as the prior degrees of freedom, for which smaller values correspond to more heteroskedasticity.

```
> summary(result$df.prior)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  15.77   16.65   16.65   16.64   16.65   16.65
```

It's important to use the `robust=TRUE` argument in `glmQLFTest`. This protects against any large positive outliers that might be present. It also protects against large negative outliers when the counts are small. In both cases, such outliers would inflate the apparant heteroskedasticity and increase the estimated prior degrees of freedom.

Chapter 7

Testing for significant interactions

7.1 Using the quasi-likelihood F-test

The `glmQLFTest` function also performs the F-test for each interaction so a new function call is not required. Nonetheless, the command is repeated below for the reader's convenience. Users should check that the contrast has been specified correctly through the `coef` or `contrast` arguments. In this case, the coefficient of interest refers to the change in the KO counts over the WT counts. The null hypothesis for each bin pair is that the coefficient is equal to zero, i.e., there is no change.

```
> result <- glmQLFTest(y, design, robust=TRUE, coef=2)
> topTags(result)

Coefficient:  KO
          logFC   logCPM        F       PValue          FDR
65738 1.5782129 4.179440 289.6546 8.204117e-13 9.139222e-08
41082 1.3160962 4.679997 255.0940 2.494727e-12 1.389538e-07
60091 1.5561953 3.178065 237.3638 4.668093e-12 1.733387e-07
30073 1.1562636 4.323671 193.0424 2.767373e-11 6.580457e-07
65739 1.1268713 4.442631 191.5757 2.953580e-11 6.580457e-07
2353  1.2249625 4.103014 183.3362 4.295484e-11 7.180829e-07
65693 1.1961975 3.856083 179.9750 5.026795e-11 7.180829e-07
41858 1.2120282 4.014323 179.4341 5.156882e-11 7.180829e-07
28009 1.0846384 4.518359 176.2516 6.002039e-11 7.429057e-07
25809 0.9776271 5.101400 173.8218 6.750787e-11 7.520241e-07
```

Users might wonder why the likelihood ratio test (LRT) is not used. Indeed, the LRT is the more obvious test for inferences with GLMs. However, the QL F-test is preferred as it accounts for the variability and uncertainty of the QL dispersion estimates [Lund et al., 2012]. This means that it can maintain type I error control in the presence of heteroskedasticity whereas the LRT does not.

7.2 Multiplicity correction and the FDR

7.2.1 Overview

Many bin pairs are tested for differences across the interaction space. Correction for multiple testing is necessary to avoid detection of many spurious differences. For genome-wide analyses, this correction can be performed by controlling the false discovery rate (FDR) with the Benjamini-Hochberg (BH) method [Benjamini and Hochberg, 1995]. This provides a suitable comprimise between specificity and sensitivity. In contrast, traditional methods of correction are often too conservative, e.g., Bonferroni.

7.2.2 Direct application of the BH method

The BH method can be applied directly to the p-values for the individual bin pairs. In this case, the FDR refers to the proportion of detected bin pairs that are false positives. Significantly specific interactions are defined as those that are detected at an FDR of 5%.

```
> adj.p <- p.adjust(result$table$PValue, method="BH")
> sum(adj.p <= 0.05)
```

```
[1] 5853
```

These can be saved to file as necessary. The resorting by p-value just makes it easier to parse the final, as the most interesting differential interactions are placed at the top.

```
> ax <- anchors(data)
> tx <- targets(data)
> final <- data.frame(anchor.chr=seqnames(ax), anchor.start=start(ax), anchor.end=end(ax),
+     target.chr=seqnames(tx), target.start=start(tx), target.end=end(tx),
+     result$table, FDR=adj.p)
> o <- order(final$PValue)
> write.table(final[o,], file="results.tsv", sep="\t", quote=FALSE, row.names=FALSE)
```

7.2.3 Aggregating small bin pairs

Smaller bins may be more useful as they can identify sharp features that would otherwise be "averaged out" from counting with larger bins. For purposes of demonstration, a quick-and-dirty analysis of the previously loaded 100 kbp bin pairs is performed here.

```
> y.small <- DGEList(counts(smaller.data), lib.size=totals(smaller.data))
> y.small$offset <- normalize(smaller.data, type="loess")
> y.small <- estimateDisp(y.small, design)
> result.small <- glmQLFTest(y.small, design, robust=TRUE)
```

Use of smaller bin pairs can lead to problems when diffuse interactions are present. Recall that the FDR refers to the proportion of bin pairs that are false positives. As the bin pair is just an analytical construct, and the actual statistic of interest is the equivalent proportion for the underlying interactions. The FDR over bin pairs can be used as a proxy for the FDR over interactions, if one assumes that each bin pair roughly corresponds to an interaction. This is reasonable for pairs of large bins, but is less so when multiple smaller bin pairs are used to represent a diffuse interaction. This can result in misinterpretation of the FDR such that control is lost [Lun and Smyth, 2014].

A typical approach might be to cluster bin pairs in the interaction space to identify the underlying true interaction. One can then combine p-values across all bin pairs in the cluster, to obtain a single combined p-value for the cluster. This approach avoids misinterpretation for diffuse interactions as each cluster (and thus interaction) has one p-value. However, this is confounded by the density of the interaction space, particularly at short distances and/or in TADs. The boundaries of each cluster become ambiguous and difficult to interpret, e.g., if a whole TAD is absorbed into a cluster.

Boundary ambiguity can be avoided by performing pre-defined clustering based on large bin pairs. Specifically, a large bin pair is defined and the set of all smaller bin pairs nested therein is defined as a cluster. This is robust to high-density space as the definition of the cluster does not depend on the neighbouring bin pairs. Identification of these clusters can be achieved with the `boxPairs` function, as shown below. Note that the larger bin size must be a multiple of the `width` used for the smaller bins - in this case, 1 Mbp.

```
> matched <- boxPairs(reference=bin.size, smaller=smaller.data, fragments=mm.frag)
```

The p-values for all of the smaller bin pairs can then be combined using Simes' method. The resulting combined p-value represents the evidence against the global null hypothesis for the larger bin pair. That is, there are no significant differences in any of the smaller bin pairs nested within the larger bin pair. The BH method is applied to the combined p-values to control the FDR across all larger bin pairs/interactions. Rejection of the global null indicates that there is a change and that the larger bin pair is worth further investigation.

```
> tabcom <- combineTests(matched$indices$smaller, result.small$table)
> head(tabcom)

         logFC     logCPM       PValue          FDR
1 -0.820220116 -1.4307054 1.517308e-02 3.282635e-02
2  0.030581207  2.0648678 2.286361e-02 4.633919e-02
3 -0.104999654 -0.1344230 3.956528e-01 4.782809e-01
4 -0.003126787  2.0871058 6.744154e-09 1.232663e-07
5  0.189886488 -1.4693763 2.662355e-01 3.495479e-01
6 -0.197359453 -0.5803001 1.238441e-05 7.175996e-05

> sum(tabcom$FDR <= 0.05)

[1] 5566
```

Of course, this bin clustering method will not be optimal for diffuse interactions. One could obtain larger counts with larger bin pairs, given that spatial resolution is not an issue for diffuse interactions. Nonetheless, bin clustering will protect against misinterpretation when both sharp and diffuse interactions are present. The use of small bin pairs will allow detection of the former, while bin clustering will prevent the results being dominated by the latter. This method is also a good introduction to the next section, which might be more useful.

7.2.4 Merging results from different bin widths

The choice of bin size is not clear when there are both sharp and diffuse changes in the interaction space. Robustness can be provided by combining the differential testing results between small and large bin pairs. All smaller bin pairs that are nested within each of the larger bin pairs are identified using the `boxPairs` function. This yields identifiers for each bin pair where `smaller` bin pairs are nested in `larger` bin pairs with the same ID.

```
> matched <- boxPairs(bin.size, larger=data, smaller=smaller.data, fragments=mm.frag)
> ldex <- matched$indices$larger
> sdex <- matched$indices$smaller
```

Each larger bin pair is associated with its own p-value and those of the smaller nested bin pairs. All of these p-values can be combined using Simes' method, similar to that previously described. The idea is to provide a compromise between larger counts and spatial resolution in the final combined p-value.

Some finesse is necessary to assign different weights to each p-value during the combining procedure [Benjamini and Hochberg, 1997]. For each larger bin pair, the p-value for that bin pair has the same weight as those of all the smaller nested bin pairs. This means that the analysis with the larger bin pair has the same contribution as that for the smaller bin pairs. The aim is to avoid increasing the contribution of the latter simply because there are more smaller bin pairs covering the interaction space.

```
> weight <- c(1/counts(matched$pairs)[sdex,"smaller"],
+     1/counts(matched$pairs)[ldex,"larger"])
```

The code snippet above makes sense as the count matrix from the `DIList` in `boxPairs` records the number of nested bin pairs at each bin size. So, as the number of nested bin pairs increases, the weight assigned to each bin pair decreases. This effectively means that the p-values from each bin size are assigned equal weight. Calculation of the combined p-value is then performed using the `combineTests` function in the csaw package.

```
> result.com <- combineTests(ids=c(sdex, ldex),
+     tab=rbind(result.small$table, result$table), weight=weight)
> head(result.com)
```

```
        logFC    logCPM       PValue          FDR
1 -0.82022012 -1.430705 1.517308e-02 1.235083e-01
2  0.14841658  4.952530 2.700122e-04 6.622190e-03
3 -0.12843844  3.137101 7.391431e-02 3.125385e-01
4  0.07667818  4.705547 1.348831e-08 2.261431e-06
5  0.26301985  1.788312 6.715889e-04 1.343094e-02
6 -0.21740652  2.833992 2.476882e-05 9.761578e-04
```

The BH method is then applied to the combined p-values. The FDR refers to the proportion of larger bin pairs that are false discoveries. Each large bin pair is safe to use as a proxy for the underlying interaction (or at least, safer than smaller bin pairs) to avoid misinterpretation of the FDR. Note the difference in the results after combining results with the smaller bin pairs, relative to direct application of the BH method to the larger bin pairs. An increase in the number of detections suggests that more features can be detected at higher resolution. However, decreases are also possible as the effective number of tests increases the severity of the correction.

```
> sum(result.com$FDR <= 0.05)

[1] 8757
```

Finally, the results can be saved to file. The IDs in identifiers can be matched to those of the `reference` that was used in `boxPairs`. The corresponding coordinates of the bin pairs in `original` can then be saved to file. Of course, the coordinates of the smaller bin pairs can also be saved, though users should keep in mind that the FDR is computed with respect to the larger bin pairs.

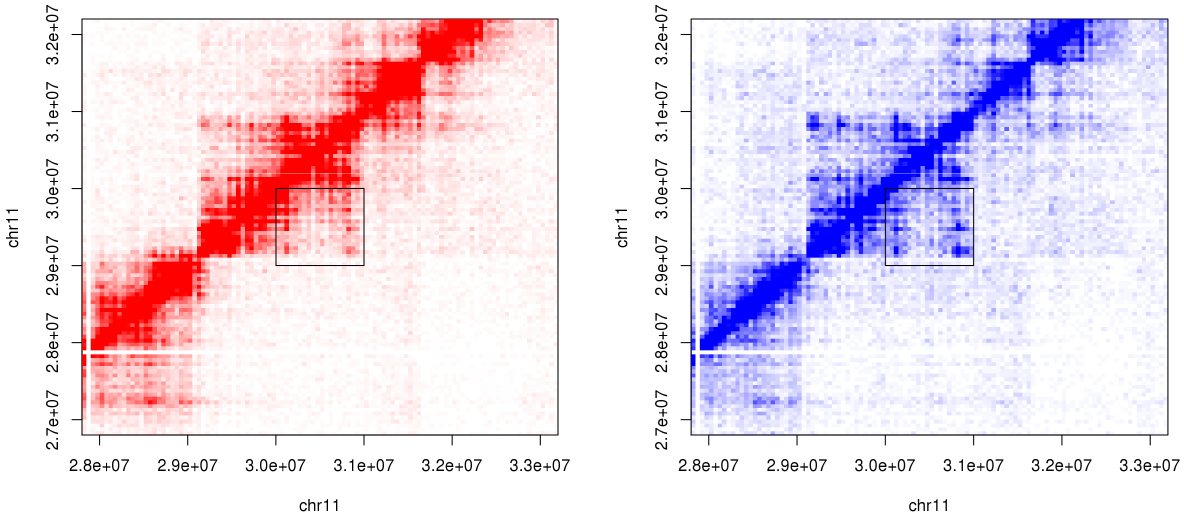
```
> ax.2 <- anchors(matched$pairs)
> tx.2 <- targets(matched$pairs)
> final.2 <- data.frame(anchor.chr=seqnames(ax.2), anchor.start=start(ax.2),
+     anchor.end=end(ax.2), target.chr=seqnames(tx.2), target.start=start(tx.2),
+     target.end=end(tx.2), result.com)
> o2 <- order(final.2$PValue)
> write.table(final.2[o2,], file="results.2.tsv", sep="\t", quote=FALSE, row.names=FALSE)
```

7.3 Visualization with plaid plots

Plaid plots can be used to visualize the distribution of read pairs in the interaction space [Lieberman-Aiden et al., 2009]. Briefly, each axis is a chromosome segment. The box represents an interaction between the corresponding intervals on each axis. The colour of the box is proportional to the number of read pairs mapped between the interacting loci.

```
> chosen <- o2[1]
> expanded.a <- resize(ax.2[chosen], fix="center", width=bin.size*5)
> expanded.t <- resize(tx.2[chosen], fix="center", width=bin.size*5)
> cap1 <- 50
> cap3 <- cap1*totals(data)[3]/totals(data)[1]
> plotPlaid(input[1], anchor=expanded.a, target=expanded.t, cap=cap1, width=5e4, fragments=mm.frag)
> rect(start(ax.2[chosen]), start(tx.2[chosen]), end(ax.2[chosen]), end(tx.2[chosen]))
> plotPlaid(input[3], anchor=expanded.a, target=expanded.t, cap=cap3,
+     width=5e4, col="blue", fragments=mm.frag)
> rect(start(ax.2[chosen]), start(tx.2[chosen]), end(ax.2[chosen]), end(tx.2[chosen]))
```

Expansion of the plot boundaries is often desirable. This ensures that the context of the interaction can be determined by examining the features in the surrounding interaction space. It is also possible to tune the size of the boxes through a parameter that is, rather unsurprisingly, named `width`. In this case, the side of each box represents a 50 kbp bin, rounded to the nearest restriction site. The actual bin pair occurs at the center of the plot and is marked by a rectangle.

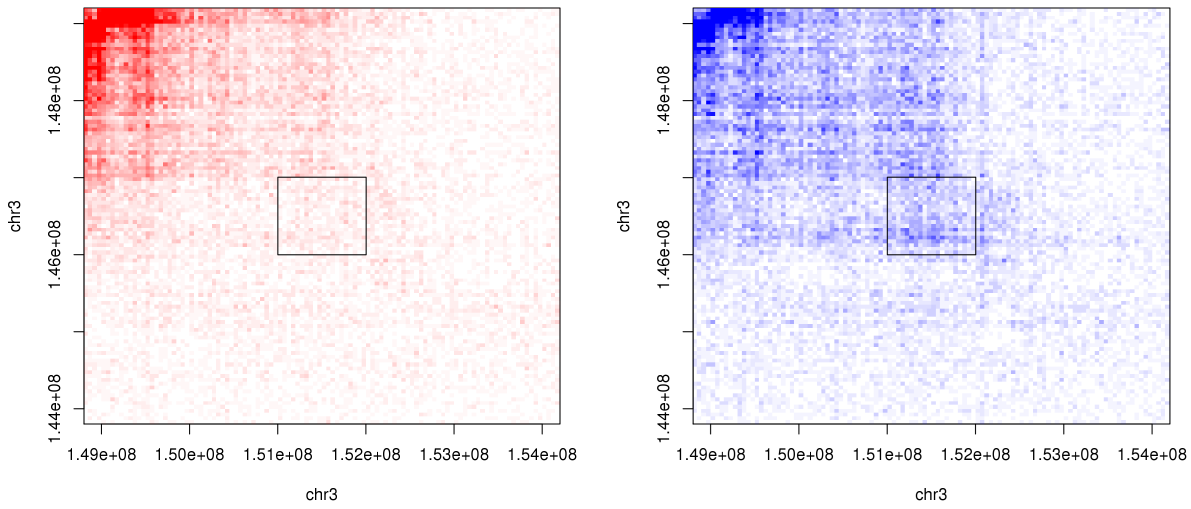


The `cap` value controls the relative scale of the colours. A smaller `cap` is necessary for smaller libraries so that the intensity of the colours is comparable. The colour can also be controlled by specifying a different string in `col`.

In the example above, the differential interaction is driven mainly by the smaller bin pairs. Changes in intensities are particularly prevalent at the top left and bottom right corners of the rectangle. By comparison, the fold change for the entire interaction is a little less than 30% between groups. This highlights the usefulness of including information from analyses with smaller bin sizes.

It is also possible to examine a scenario involving larger changes. The following plots are constructed for the top differential interaction, detected using only the larger bin size. Because the counts are "averaged" across the area of the interaction space, the change must be consistent throughout that area (and thus, more obvious) for detection to be successful. Of course, any sharp changes within each of these large bin pairs will be overlooked as the smaller bin pairs are not used.

```
> chosen <- o[1]
> expanded.a <- resize(ax[chosen], fix="center", width=bin.size*5)
> expanded.t <- resize(tx[chosen], fix="center", width=bin.size*5)
> cap1 <- 30
> cap3 <- cap1*totals(data)[3]/totals(data)[1]
> plotPlaid(input[1], anchor=expanded.a, target=expanded.t, cap=cap1,
+     width=5e4, fragments=mm.frag, col="red")
> rect(start(ax[chosen]), start(tx[chosen]), end(ax[chosen]), end(tx[chosen]))
> plotPlaid(input[3], anchor=expanded.a, target=expanded.t, cap=cap3,
+     width=5e4, col="blue", fragments=mm.frag)
> rect(start(ax[chosen]), start(tx[chosen]), end(ax[chosen]), end(tx[chosen]))
```



Chapter 8

Epilogue

8.1 Session information

```
> sessionInfo()

R version 3.1.0 (2014-04-10)
Platform: x86_64-unknown-linux-gnu (64-bit)

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
 [3] LC_TIME=en_US.UTF-8        LC_COLLATE=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] splines   parallel  stats     graphics  grDevices utils     datasets
[8] methods   base

other attached packages:
 [1] org.Mm.eg.db_2.14.0                  RSQLite_0.11.4
 [3] DBI_0.2-7                            AnnotationDbi_1.26.0
 [5] Biobase_2.24.0                       BSgenome.Mmusculus.UCSC.mm10_1.3.1000
 [7] BSgenome.Hsapiens.UCSC.hg19_1.3.1000 BSgenome_1.32.0
 [9] Biostrings_2.32.1                    XVector_0.4.0
[11] statmod_1.4.20                       locfit_1.5-9.1
[13] csaw_0.99.0                          edgeR_3.7.10
[15] limma_3.20.9                         diffHic_0.0.3
[17] GenomicRanges_1.16.4                 GenomeInfoDb_1.0.2
[19] IRanges_1.22.10                      BiocGenerics_0.10.0

loaded via a namespace (and not attached):
 [1] BatchJobs_1.3           BBmisc_1.7              BiocParallel_0.6.1
 [4] biomaRt_2.20.0          bitops_1.0-6            brew_1.0-6
```

```
 [7] checkmate_1.3           codetools_0.2-9         digest_0.6.4
[10] fail_1.2                foreach_1.4.2           GenomicAlignments_1.0.6
[13] GenomicFeatures_1.16.2  grid_3.1.0              iterators_1.0.7
[16] KernSmooth_2.23-12      lattice_0.20-29         RCurl_1.95-4.3
[19] rhdf5_2.8.0             Rsamtools_1.16.1        rtracklayer_1.24.2
[22] sendmailR_1.1-2         stats4_3.1.0            stringr_0.6.2
[25] tools_3.1.0             XML_3.98-1.1            zlibbioc_1.10.0
```

8.2 References

J. M. Belton, R. P. McCord, J. H. Gibcus, N. Naumova, Y. Zhan, and J. Dekker. Hi-C: a comprehensive technique to capture the conformation of genomes. *Methods*, 58(3):268–276, Nov 2012.

Y. Benjamini and Y. Hochberg. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *J. Roy. Statist. Soc. B*, pages 289–300, 1995.

Y. Benjamini and Y. Hochberg. Multiple hypotheses testing with weights. *Scand. J. Stat.*, 24:407–418, 1997.

W. A. Bickmore. The spatial organization of the human genome. *Annu. Rev. Genomics Hum. Genet.*, 14:67–84, 2013.

R. Bourgon, R. Gentleman, and W. Huber. Independent filtering increases detection power for high-throughput experiments. *Proc. Natl. Acad. Sci. U.S.A.*, 107(21):9546–9551, May 2010.

M. Imakaev, G. Fudenberg, R. P. McCord, N. Naumova, A. Goloborodko, B. R. Lajoie, J. Dekker, and L. A. Mirny. Iterative correction of Hi-C data reveals hallmarks of chromosome organization. *Nat. Methods*, 9(10):999–1003, Oct 2012.

F. Jin, Y. Li, J. R. Dixon, S. Selvaraj, Z. Ye, A. Y. Lee, C. A. Yen, A. D. Schmitt, C. A. Espinoza, and B. Ren. A high-resolution map of the three-dimensional chromatin interactome in human cells. *Nature*, 503(7475):290–294, Nov 2013.

E. Lieberman-Aiden, N. L. van Berkum, L. Williams, M. Imakaev, T. Ragoczy, A. Telling, I. Amit, B. R. Lajoie, P. J. Sabo, M. O. Dorschner, R. Sandstrom, B. Bernstein, M. A. Bender, M. Groudine, A. Gnirke, J. Stamatoyannopoulos, L. A. Mirny, E. S. Lander, and J. Dekker. Comprehensive mapping of long-range interactions reveals folding principles of the human genome. *Science*, 326(5950):289–293, Oct 2009.

Y. C. Lin, C. Benner, R. Mansson, S. Heinz, K. Miyazaki, M. Miyazaki, V. Chandra, C. Bossen, C. K. Glass, and C. Murre. Global changes in the nuclear positioning of genes

and intra- and interdomain genomic interactions that orchestrate B cell fate. *Nat. Immunol.*, 13(12):1196–1204, Dec 2012.

C. Loader. *Local Regression and Likelihood*. Statistics and Computing. Springer New York, 1999. ISBN 9780387987750. URL `http://books.google.com.au/books?id=D7GgBAfL4ngC`.

A. T. Lun and G. K. Smyth. De novo detection of differentially bound regions for ChIP-seq data using peaks and windows: controlling error rates correctly. *Nucleic Acids Res.*, May 2014.

S. P. Lund, D. Nettleton, D. J. McCarthy, and G. K. Smyth. Detecting differential expression in RNA-sequence data using quasi-likelihood with shrunken dispersion estimates. *Stat. Appl. Genet. Mol. Biol.*, 11(5), 2012.

J. C. Marioni, C. E. Mason, S. M. Mane, M. Stephens, and Y. Gilad. RNA-seq: an assessment of technical reproducibility and comparison with gene expression arrays. *Genome Res.*, 18(9):1509–1517, Sep 2008.

D. J. McCarthy, Y. Chen, and G. K. Smyth. Differential expression analysis of multifactor RNA-Seq experiments with respect to biological variation. *Nucleic Acids Res.*, 40(10): 4288–4297, May 2012.

M. D. Robinson, D. J. McCarthy, and G. K. Smyth. edgeR: a Bioconductor package for differential expression analysis of digital gene expression data. *Bioinformatics*, 26(1):139–140, Jan 2010.

V. C. Seitan, A. J. Faure, Y. Zhan, R. P. McCord, B. R. Lajoie, E. Ing-Simmons, B. Lenhard, L. Giorgetti, E. Heard, A. G. Fisher, P. Flicek, J. Dekker, and M. Merkenschlager. Cohesin-based chromatin interactions enable regulated gene expression within preexisting architectural compartments. *Genome Res.*, 23(12):2066–2077, Dec 2013.

S. Sofueva, E. Yaffe, W. C. Chan, D. Georgopoulou, M. Vietri Rudan, H. Mira-Bontenbal, S. M. Pollard, G. P. Schroth, A. Tanay, and S. Hadjur. Cohesin-mediated interactions organize chromosomal domain architecture. *EMBO J.*, 32(24):3119–3129, Dec 2013.

The HDF Group. Hierarchical Data Format, version 5, 1997-2014. http://www.hdfgroup.org/HDF5/.