Analyzing ChIP-chip data using Bioconductor

Joern Toedling, Wolfgang Huber

EMBL European Bioinformatics Institute, Wellcome Trust Genome Campus,
Hinxton CB10 1SD, United Kingdom

1 Introduction

ChIP-chip, chromatin immunoprecipitation combined with DNA microarrays, is a currently widely used assay for DNA-protein binding and chromatin plasticity, which are of fundamental interest for the understanding of gene regulation.

The interpretation of ChIP-chip data poses two computational challenges: first, what can be termed primary statistical analysis, which includes quality assessment, data normalization and transformation, and the calling of regions of interest; second, integrative bioinformatic analysis, which interprets the data in the context of existing genome annotation and of related experimental results obtained, for example, from other ChIP-chip or (m)RNA abundance microarray experiments.

Both tasks rely heavily on visualization, which helps to explore the data as well as to present the analysis results. For the primary statistical analysis, some standardization is possible and desirable: commonly used experimental designs and microarray platforms allow the development of relatively standard workflows and statistical procedures. Most software available for ChIP-chip data analysis can be employed in such standardized approaches [1, 2, 3, 4, 5, 6]. Yet even for primary analysis steps, it may be beneficial to adapt them to specific experiments, and hence it is desirable that software offers flexibility in the choice of algorithms for normalization, visualization and identification of enriched regions.

For the second task, integrative bioinformatic analysis, the datasets, questions and the applicable methods are diverse, and a degree of flexibility is needed that often can only be achieved in a programmable environment.

Bioconductor [7] is an open source and open development software project for the analysis and comprehension of genomic data, and it offers tools that cover a broad range of computational methods, visualizations and experimental data types, and is designed to allow the construction of scalable, reproducible and interoperable workflows. A consequence of the wide range of functionality of Bioconductor and its concurrency with research progress in biology and computational statistics is that using its tools can be daunting for a new user. This tutorial covers basic ChIP-chip data analysis with Bioconductor. Among the packages used are *Ringo* [5], *biomaRt* [8] and *topGO* [9].

We wrote this document in the `Sweave` [10] format, which combines explanatory text and the actual R source code used in this analysis [11]. Thus that the analysis can easily be reproduced by the reader. An R package *ccTutorial* that contains the input data, the text and code presented here, as well as supplementary text and code is available from the Bioconductor web site.

```
> library("Ringo")
> library("biomaRt")
> library("topGO")
> library("ccTutorial")
```

1.1 The data

We consider a ChIP-chip data set on a post-translational modification of histone protein H3, namely tri-methylation of its Lysine residue 4, in short *H3K4me3*. H3K4me3 has been associated with active transcription (e.g., [12]). Here, enrichment for H3K4me3 was investigated in *Mus musculus* brain and heart cells. The microarray platform is a set of 4 arrays manufactured by NimbleGen containing 390k reporters each. The reporters were designed to tile 32,482 selected regions of the *Mus musculus* genome (assembly *mm5*) with one base every 100bp, with a different set of promoters represented on each of the four arrays [13, Methods: Condensed array ChIP-chip]. We obtained the data from the GEO repository [14] (accession GSE7688).

Glossary

Reporters are the DNA sequences fixed to the microarray; they are designed to specifically hybridize with corresponding genomic fragments from the immuno-precipitate. A reporter has a unique identifier and a unique sequence, and it can appear in one or multiple *features* on the array surface. The *sample* is the aliquot of immuno-precipitated or input DNA that is hybridized to the microarray. We shall call a genomic region apparently enriched by ChIP a *ChIP-enriched region*.

2 Importing the data into R

For each microarray, the scanner output consists of two files, one holding the Cy3 intensities (the untreated *input* sample), the other one the Cy5 intensities, coming from the immuno-precipitated sample. These files are tab-delimited text files in NimbleGen's *pair* format. Since the reporters are distributed over 4 arrays, we have 16 files (4 microarrays × 2 dyes × 2 tissues).

```
> pairDir <- system.file("PairData",package="ccTutorial")
> list.files(pairDir, pattern="pair$")
```

```
 [1] "47101_532.pair" "47101_635.pair" "48153_532.pair" "48153_635.pair"
 [5] "48158_532.pair" "48158_635.pair" "48170_532.pair" "48170_635.pair"
 [9] "48175_532.pair" "48175_635.pair" "48180_532.pair" "48180_635.pair"
[13] "48182_532.pair" "48182_635.pair" "49728_532.pair" "49728_635.pair"
```

One text file per array describes the samples, including which two *pair* files belong to which sample. Another file, `spottypes.text`, describes the reporter categories on the arrays.

We read in the raw reporter intensities and obtain four objects of class *RGList*, a class defined in package *limma* [15], one object per array type.

```
> RGs <- lapply(sprintf("files_array%d.txt",1:4),
+   readNimblegen, "spottypes.txt", path=pairDir)
```

See the Supplement for an extended description of the data import.

3 Quality assessment

In this step, we check the arrays for obvious artifacts and inconsistencies between array subsets.

First, we look at the spatial distribution of the intensities on each array. See Figure S1 and the Supplement text for the source code. We do not see any artifacts such as scratches, bright spots, or finger prints that would render parts of the readouts useless.

On all arrays in our set, the Cy3 channel holds the intensities from the untreated *input* sample, and the Cy5 channel holds the immuno-precipiate from brain and heart, respectively. This experiment setup is reflected in the reporter intensity correlation per channel (see Figure S3). The correlation between the intensities of the *input* samples is higher than between the ChIP samples (0.877 versus 0.734).

The Bioconductor package *arrayQualityMetrics* [16] offers an extensive set of visualizations and metrics for assessing microarray data quality. Applied to this data set, *arrayQualityMetrics* also indicates the data are of good quality.

4 Mapping reporters to the genome

A mapping of reporters to genomic coordinates is usually provided by the array manufacturer. Often, however, remapping the reporter sequences to the genome may be required. Here, the microarray had been designed on an outdated assembly of the mouse genome (mm5, May 2004). We remapped the reporter sequences to the current assembly (mm9, July 2007).

We used *Exonerate* [17] for the remapping, requiring 97% sequence similarity for a match. See the Supplement for details and the used scripts.

In *Ringo*, the mapping of reporters to the genome is stored in a *probeAnno* class object. The supplement contains details on its construction.

```
> data("probeAnno")
> allChrs <- chromosomeNames(probeAnno)
```

5 Genome annotation

We want to relate ChIP-enriched regions to annotated genome elements, such as potential regulatory regions and transcripts. Using the Bioconductor package *biomaRt* [8], we obtain an up-to-date annotation of the mouse genome from the Ensembl database [18].

The source code for creating the annotation table `mm9genes` is given in the Supplement. This table holds the coordinates, Ensembl gene identifiers, MGI symbols, and description of all genes annotated for the *mm9* mouse assembly.

```
> data("mm9genes")
> mm9genes[sample(nrow(mm9genes), 4),
+     c("name", "chr", "strand", "start", "end", "symbol")]
```

```
                    name chr strand     start       end        symbol
7284  ENSMUSG00000057903  14      1  51044196  51045125       Olfr739
10209 ENSMUSG00000039615  17     -1  25967581  25970306         Stub1
15715 ENSMUSG00000068823   3      1 102824530 102862108         Csde1
24914 ENSMUSG00000006241   9      1  21731915  21740316 2510048L02Rik
```

Moreover, we used *biomaRt* to retrieve the Gene Ontology (GO)[19] annotation for all genes in the table. Find the source code and further details in the Supplement.

```
> data("mm9.gene2GO")
```

For all genes, we stored which reporters, if any, are mapped inside the gene or in its 5kb upstream region.

```
> data("mm9.g2p")
```

For later use, we determine which genes have a sufficient number - arbitrarily we say 5 - of reporters mapped to their upstream region or inside of them and the subset of them that also have ≥ 1 GO term(s) annotated to them.

```
> arrayGenes <- names(mm9.g2p)[listLen(mm9.g2p)>=5]
> arrayGenesWithGO <- intersect(arrayGenes, names(mm9.gene2GO))
```

6 Preprocessing

For each sample, we compute the $\log_2$ ratios $\log_2(\mathrm{Cy5}/\mathrm{Cy3})$ for all reporters. To adjust for systematic dye and labeling biases, we compute Tukey's biweight mean across each sample's $\log_2$ ratios and subtract it from the individual $\log_2$ ratios. Each of the four microarray types used contains a unique set of reporters. Thus, we preprocess the arrays separately by type and combine the results into one object holding the preprocessed readouts for all reporters.

```
> MAs <- lapply(RGs, function(thisRG)
+   preprocess(thisRG[thisRG$genes$Status=="Probe",],
+              method="nimblegen", returnMAList=TRUE))
> MA <- do.call(rbind, MAs)
> X  <- asExprSet(MA)
> sampleNames(X) <- paste(X$Cy5, X$Tissue, sep=".")
```

The result is an object of class *ExpressionSet*, the Bioconductor class for storing preprocessed microarray data. Note that first creating an *MAList* for each array type, combining them with `rbind` and then converting the result into an *ExpressionSet* is only necessary if the reporters are distributed over more than one microarray type (four in this case). For data of one microarray type only, you can call `preprocess` with argument `returnMAList=-FALSE` and directly obtain the result as an *ExpressionSet*.

The above procedure is the standard method suggested by NimbleGen for ChIP-chip. More sophisticated normalization methods exist, *e.g.*, using the genomic DNA hybridization as reference [20]. However, due to the smaller dynamic range of the data in the input channel such additional effort seems less worthwhile than, say, for transcription microarrays.

7 Visualizing intensities along the chromosome

We visualize the preprocessed H3K4me3 ChIP-chip reporter-wise readouts around the start of the *Actc1* gene, which encodes the cardiac actin protein.

```
> chipAlongChrom(X, chrom="2", xlim=c(113.873e6,113.883e6), ylim=c(-3,5),
+                probeAnno=probeAnno, gff=mm9genes, paletteName='Set2')
```

The degree of H3K4me3 enrichment over the reporters mapped to this region seems stronger in heart cells than in brain cells (see Figure 1). However, the signal is highly variable and individual reporters give different readouts from reporters matching genomic positions only 100bp away, even though the DNA fragments after sonication are hundreds of base pairs long.

See Figure S4 for the corresponding intensities around the start of the brain-specific gene *Crpm1* [21].

8 Smoothing of reporter intensities

The signal variance arises from systematic and stochastic noise. Individual reporters measure the same amount of DNA with different efficiency due to reporter sequence characteristics [22], such as GC content, secondary structure, and cross-hybridization. To ameliorate these reporter effects as well as the stochastic noise, we perform a smoothing over individual reporter intensities before looking for ChIP-enriched regions. We slide a window of 900 bp width along the chromosome and replace the intensity at genomic position x_0 by the median over the intensities of those reporters mapped inside the window centered at x_0. Factors to take into account when choosing the width of the sliding window are the size distribution of DNA fragments after sonication and the spacing between reporter matches on the genome.

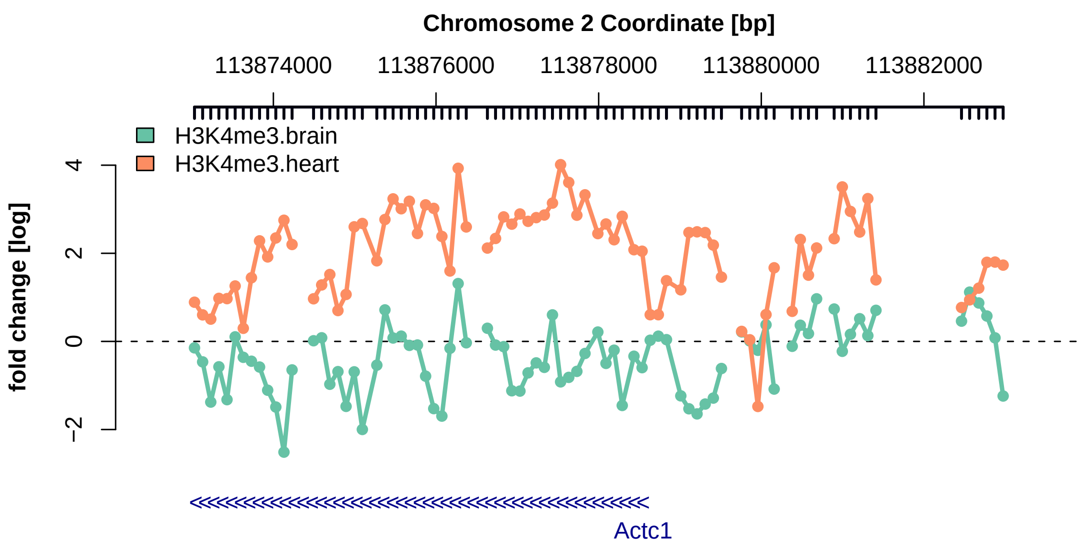

Figure 1: *Normalized reporter intensities for H3K4me3 ChIP around the TSS of the* Actc1 *gene in* M. musculus *brain and heart cells. The ticks below the genomic coordinate axis on top indicate genomic positions matched by reporters on the microarray. The blue arrows on the bottom mark the Actc1 gene with the arrow direction indicating that the gene is located on the Crick strand.*

```
> smoothX <- computeRunningMedians(X, probeAnno=probeAnno,
+   modColumn="Tissue", allChr=allChrs, winHalfSize=450, min.probes=5)
> sampleNames(smoothX) <- paste(sampleNames(X),"smoothed",sep=".")
```

Compare the smoothed reporter intensities with the original ones around the start of the gene *Actc1*.

```
> chipAlongChrom(X, chrom="2", xlim=c(113.873e6,113.883e6), ylim=c(-3,5),
+                  probeAnno=probeAnno, gff=mm9genes, paletteName='Set2')
> chipAlongChrom(smoothX, chrom="2", xlim=c(113.873e6,113.883e6), ilwd=4,
+                  probeAnno=probeAnno, paletteName='Dark2', add=TRUE)
```

See the result in Figure 2. After smoothing, the reporters give a more concise picture that there is H3K4me3 enrichment inside and upstream of *Actc1* in heart but not in brain cells.

9 Finding ChIP-enriched regions

We would like to determine a discrete set of regions that appear antibody-enriched, together with a quantitative score of our confidence in that and a measure of their enrichment strength. Which approach is best for this purpose depends on the microarray design, on the biological question and on the subsequent use of the regions e. g. in a follow-up experiment or computational analysis. Below, we describe one possible approach, but before we discuss two more conceptual aspects.

In the literature, a computed confidence score is often mixed up with the term "*p*-value". Speaking of a *p*-value is meaningful only if there is a defined null hypothesis and a probability

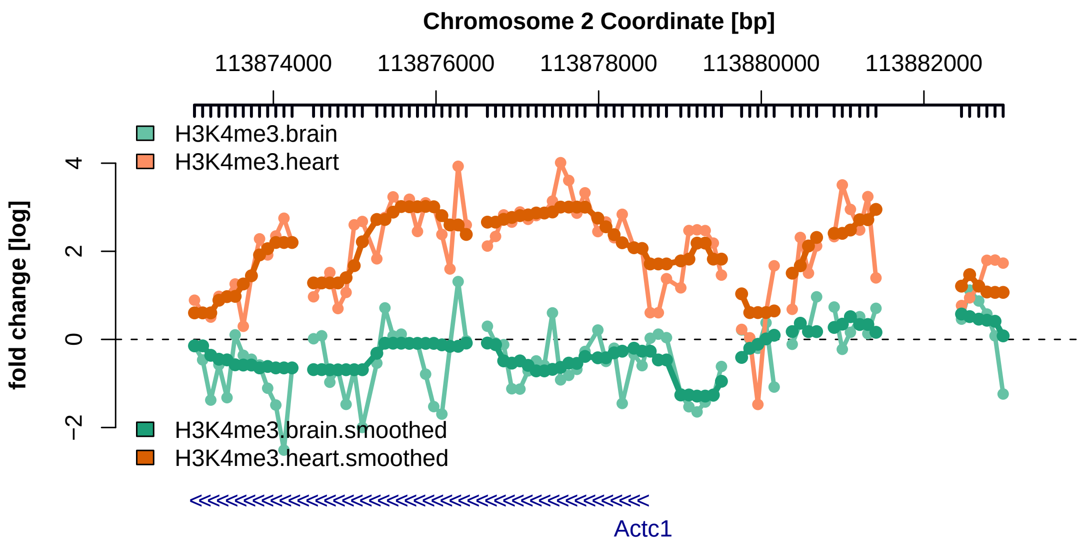

Figure 2: *Normalized and smoothed reporter intensities for H3K4me3 ChIP around the TSS of the* Actc1 *gene in* M. musculus *brain and heart cells.*

interpretation; these complications are not necessary if the goal is simply to find and rank regions in some way that can be reasonably calibrated.

Furthermore, it is helpful to distinguish between our confidence in an enrichment being present, and the strength of the enrichment. Although stronger enrichments tend to result in stronger signals and hence less ambiguous calls, our certainty about an enrichment should also be affected by reporter coverage, sequence, cross-hybridization etc.

Let us now consider the following simple approach: for an enriched region, require that the smoothed reporter levels all exceed a certain threshold y_0, that the region contains at least $n_{\min}$ reporter match positions, and that each of these is less than $d_{\max}$ basepairs apart from the nearest other affected position in the region.

The minimum number of reporters rule ($n_{\min}$) might seem redundant with the smoothing median computation (since a smoothed reporter intensity is already the median of all the reporter intensities in the window), but it plays its role in reporter sparse regions, where a window may only contain one or a few reporters. One wants to avoid making calls supported only by few reporters.
The $d_{\max}$ rule prevents us from calling disconnected regions.

For setting the threshold y_0, the probably best approach would be to tune it by considering sets of positive and negative control regions. As such are not available with the current data, we choose a mixture modeling approach.

We assume that the distribution of the smoothed reporter levels y is a mixture of two underlying distributions, a null distribution $\mathcal{L}_0$ of reporter levels in non-enriched regions, and an alternative distribution $\mathcal{L}_{\text{alt}}$ of levels in enriched regions.

The problem at hand is how to estimate $\mathcal{L}_0$. Let us look at histograms of the smoothed reporter levels.

```
> y0 <- apply(exprs(smoothX), 2, upperBoundNull, prob=0.99)
> myPanelHistogram <- function(x, ...){
+   panel.histogram(x, col=brewer.pal(8,"Dark2")[panel.number()], ...)
+   panel.abline(v=y0[panel.number()], col="red")
+ }
> h = histogram( ~ y | z,
+       data = data.frame(
+         y = as.vector(exprs(smoothX)),
+         z = rep(X$Tissue, each = nrow(smoothX))),
+       layout = c(1,2), nint = 50,
+       xlab = "smoothed reporter level [log2]",
+       panel = myPanelHistogram)
> print(h)
```

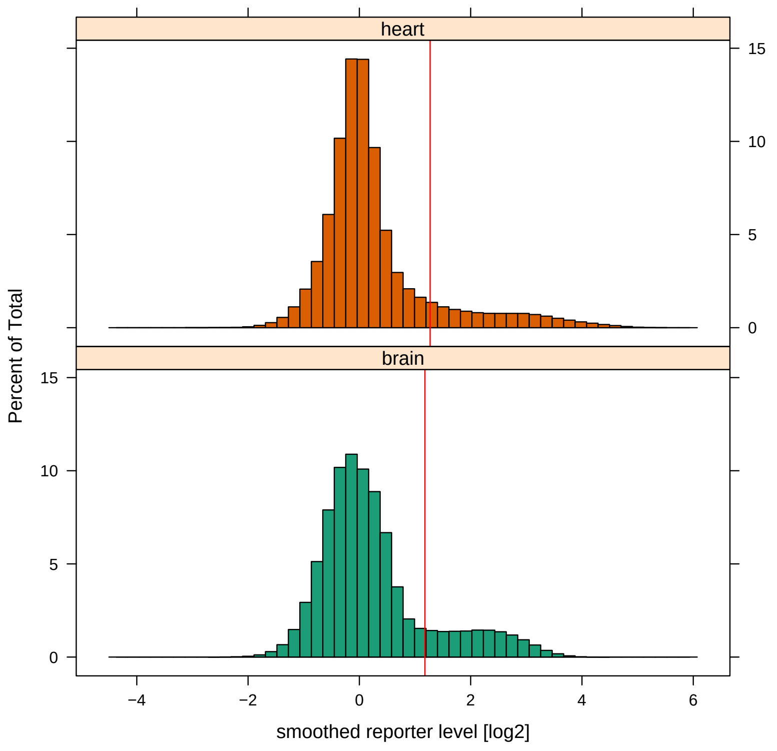

Figure 3: *Histograms of reporter intensities after smoothing of reporter levels, measured in* M. musculus *heart and brain cells. The red vertical lines are the cutoff values suggested by the algorithm described in the text.*

The first line of the code above already computes the suggested thresholds, which we will explain below. The resulting histograms are shown in Figure 3. They suggest that the null distribution $\mathcal{L}_0$ has most of its mass close to its mode m_0, which is near $y = 0$. In contrast, $\mathcal{L}_{\text{alt}}$ is more spread out, with its mass mostly well above 0.

Let us now consider the following estimator of $\mathcal{L}_0$. First, we estimate m_0 by the midpoint of the shorth of the empirical distribution of those y that fall into the interval $[-1, 1]$. Then we estimate the distribution $\mathcal{L}_0$ from the empirical distribution of $m_0 - |y - m_0|$. The assumptions that underly this estimator are that the null distribution $\mathcal{L}_0$ is symmetric about m_0, that the alternative distribution $\mathcal{L}_{\text{alt}}$ is stochastically larger than $\mathcal{L}_0$ and that it contains negligible mass at $y < m_0$.

From the estimated null distribution, we determine a threshold y_0, for example we take its 99% quantile.

The values y_0 estimated in this way are indicated by red vertical lines in the histograms in Figure 3. Antibodies vary in their efficiency to bind to their target epitope, and the noise level in the data depends on the complexity of the sample DNA. We suggest to compute y_0 separately for each antibody and tissue.

The algorithm described above has been used in the literature, for example in [23]. There are other algorithms that use more complex models of ChIP-chip data [24, 25].

We are now ready to identify H3K4me3 ChIP-enriched regions in the data. We set $n_{\min} = 5$ and $d_{\max} = 450$.

```
> chersX <- findChersOnSmoothed(smoothX,
+     probeAnno = probeAnno,
+     thresholds = y0,
+     allChr = allChrs,
+     distCutOff = 450,
+     minProbesInRow = 5,
+     cellType = X$Tissue)
```

We relate found ChIP-enriched regions to gene coordinates we retrieved from Ensembl (see Section 5). We decide to regard an enriched region as *related* to a gene if its middle position is located less than 5 kb upstream of a gene's start coordinate or between a gene's start and end coordinates, as they are annotated in the Ensembl database.

```
> chersX <- relateChers(chersX, mm9genes, upstream=5000)
```

One characteristic of enriched regions that can be used for ranking them is the *area under the curve* score, that is the sum of the smoothed reporter levels each minus the threshold. Alternatively, one can rank them by the highest smoothed reporter level in the enriched region.

```
> chersXD <- as.data.frame(chersX)
> head(chersXD[
+   order(chersXD$maxLevel, decreasing=TRUE),
+   c("chr", "start", "end", "cellType", "features", "maxLevel", "score")])
```

```
      chr     start       end cellType                               features
21721   X   7338726   7343630    heart                     ENSMUSG00000000134
22035   X  98834348  98838572    heart                     ENSMUSG00000034160
13847  17  10508374  10511376    heart                     ENSMUSG00000062078
22165   X 148236854 148239554    heart                     ENSMUSG00000025261
12760  15  10414592  10416734    heart ENSMUSG00000022248 ENSMUSG00000022247
14193  17  35972156  35975830    heart ENSMUSG00000061607 ENSMUSG00000001525
      maxLevel score
```

```
21721      5.56  83.6
22035      5.45  93.1
13847      5.44  76.3
22165      5.40  80.3
12760      5.39  53.2
14193      5.37  62.1
```

We visualize the intensities around the region with the highest smoothed level.

```
> plot(chersX[[which.max(chersXD$maxLevel)]], smoothX, probeAnno=probeAnno,
+      gff=mm9genes, paletteName="Dark2", ylim=c(-1,6))
```

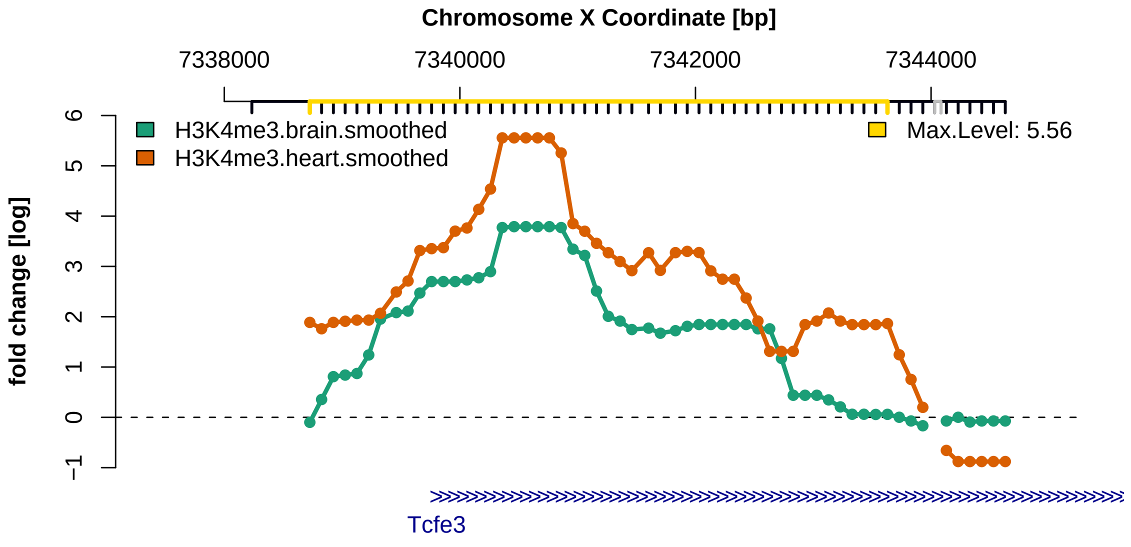

Figure 4: *The genomic region that scores highest for H3K4me3 enrichment.*

Figure 4 displays this region, which covers the gene *Tcfe3*.

10 Comparing ChIP-enrichment between the tissues

There are a number of ways to compare the H3K4me3 enrichment between the two tissues. How many ChIP-enriched regions do we find in each tissue?

```
> table(chersXD$cellType)
```

```
brain heart
11852 10391
```

Brain cells show a higher number of H3K4me3-enriched regions than heart cells. Which genes show tissue-specific association to H3K4me3 ChIP-enriched regions?

```
> brainGenes <- getFeats(chersX[sapply(chersX, cellType)=="brain"])
> heartGenes <- getFeats(chersX[sapply(chersX, cellType)=="heart"])
> brainOnlyGenes <- setdiff(brainGenes, heartGenes)
> heartOnlyGenes <- setdiff(heartGenes, brainGenes)
```

We use the Bioconductor package *topGO* [9] to investigate whether tissue-specific H3K4me3-enriched genes can be summarized by certain biological themes. *topGO* employs the Fisher test to assess whether among a list of genes, the fraction annotated with a certain GO term is significantly higher than expected by chance, considering all genes that are represented on the microarrays and have GO annotation. We set a p-value cutoff of 0.001 and the evaluation starts from the most specific GO nodes in a bottom-up approach. Genes that are used for evaluating one node are not used for evaluating any of its ancestor nodes [9, *elim* algorithm].

```
> sigGOTable <- function(selGenes, GOgenes=arrayGenesWithGO,
+  gene2GO=mm9.gene2GO[arrayGenesWithGO], ontology="BP", maxP=0.001)
+ {
+   inGenes <- factor(as.integer(GOgenes %in% selGenes))
+   names(inGenes) <- GOgenes
+   GOdata <- new("topGOdata", ontology=ontology, allGenes=inGenes,
+                 annot=annFUN.gene2GO, gene2GO=gene2GO)
+   myTestStat <- new("elimCount", testStatistic=GOFisherTest,
+                     name="Fisher test", cutOff=maxP)
+   mySigGroups <- getSigGroups(GOdata, myTestStat)
+   sTab <- GenTable(GOdata, mySigGroups, topNodes=length(usedGO(GOdata)))
+   names(sTab)[length(sTab)] <- "p.value"
+   sTab <- subset(sTab, as.numeric(p.value) < maxP)
+   sTab$Term <- sapply(mget(sTab$GO.ID, env=GOTERM), Term)
+   return(sTab)
+ }
> brainRes <- sigGOTable(brainOnlyGenes)
> print(brainRes)
```

GO.ID	Term	Annotated	Significant	Expected	p.value
GO:0007268	synaptic transmission	137	44	24.65	3.7e-05
GO:0007610	behavior	180	54	32.39	4.4e-05
GO:0007409	axonogenesis	119	38	21.41	0.00014
GO:0006887	exocytosis	40	17	7.20	0.00026
GO:0007420	brain development	136	40	24.47	0.00066

Table 1: *GO terms that are significantly over-represented among genes showing H3K4me3 enrichment specifically in brain cells*

See the result GO terms in Table 1. We perform the same analysis for genes showing heart-specific relation to H3K4me3 enrichment.

```
> heartRes <- sigGOTable(heartOnlyGenes)
> print(heartRes)
```

See the result in Table 2. Genes that show H3K4me3 in brain but not in heart cells are significantly often involved in neuron-specific biological processes. Genes marked by H3K4me3 specifically in heart cells show known cardiomyocyte functions, amongst others.

One could repeat this process using the *cellular component* and *molecular function* ontologies of the GO. Besides GO, other databases that collect gene lists can be used for this kind of gene set enrichment analysis. For, example the Kyoto Encyclopedia of Genes and Genomes (KEGG) [26] is also readily available in Bioconductor.

In the supplement, we present an additional way for comparing H3K4me3 enrichment between the two tissue, an enriched-region-wise comparison considering the actual overlap of the enriched regions.

11 ChIP results and expression microarray data

We compare the H3K4me3 ChIP-chip results with the expression microarray data, which Barrera et al. [13] provide for the same *M. musculus* tissues they analyzed with ChIP-chip.

```
> data("barreraExpressionX")
```

The data were generated using the `Mouse_430_2` oligonucleotide microarray platform from Affymetrix and preprocessed using Affymetrix's MAS5 method. Using *biomaRt*, we created a mapping of Ensembl gene identifiers to the probe set identifiers on that microarray platform (see the Supplement for the source code).

```
> data("arrayGenesToProbeSets")
```

We obtain the expression values for genes related to H3K4me3-enriched regions in heart or brain cells.

```
> bX <- exprs(barreraExpressionX)
> allH3K4me3Genes  <- union(brainGenes, heartGenes)
> allH3K4ProbeSets <- unlist(arrayGenesToProbeSets[allH3K4me3Genes])
> noH3K4ProbeSets  <- setdiff(rownames(bX), allH3K4ProbeSets)
> brainH3K4ExclProbeSets <- unlist(arrayGenesToProbeSets[brainOnlyGenes])
> heartH3K4ExclProbeSets <- unlist(arrayGenesToProbeSets[heartOnlyGenes])
> brainIdx <- barreraExpressionX$Tissue=="Brain"
> brainExpression <- list(
+   H3K4me3BrainNoHeartNo  = bX[noH3K4ProbeSets, brainIdx],
+   H3K4me3BrainYes        = bX[allH3K4ProbeSets, brainIdx],
+   H3K4me3BrainYesHeartNo = bX[brainH3K4ExclProbeSets, brainIdx],
```

GO.ID	Term	Annotated	Significant	Expected	p.value
GO:0006936	muscle contraction	56	13	2.95	4.5e-06
GO:0002526	acute inflammatory response	17	6	0.90	0.00015
GO:0009887	organ morphogenesis	339	34	17.86	0.00017
GO:0008016	regulation of heart contraction	32	8	1.69	0.00019
GO:0030878	thyroid gland development	7	4	0.37	0.00023
GO:0007512	adult heart development	8	4	0.42	0.00045
GO:0055003	cardiac myofibril assembly	4	3	0.21	0.00056
GO:0007507	heart development	148	21	7.80	0.00085

Table 2: *GO terms that are significantly over-represented among genes showing H3K4me3 enrichment specifically in heart cells*

```
+   H3K4me3BrainNoHeartYes = bX[heartH3K4ExclProbeSets, brainIdx]
+ )
```

We use boxplots to compare the brain expression levels of genes with and without H3K4me3 enriched regions in brain/heart cells.

```
> boxplot(brainExpression, col=c("#666666","#999966","#669966","#996666"),
+         names=NA, varwidth=TRUE, log="y",
+         ylab='gene expression level in brain cells')
> mtext(side=1, at=1:length(brainExpression), padj=1, font=2,
+       text=rep("H3K4me3",4), line=1)
> mtext(side=1, at=c(0.2, 1:length(brainExpression)), padj=1, font=2,
+       text=c("brain/heart","-/-","+/+","+/-","-/+"), line=2)
```

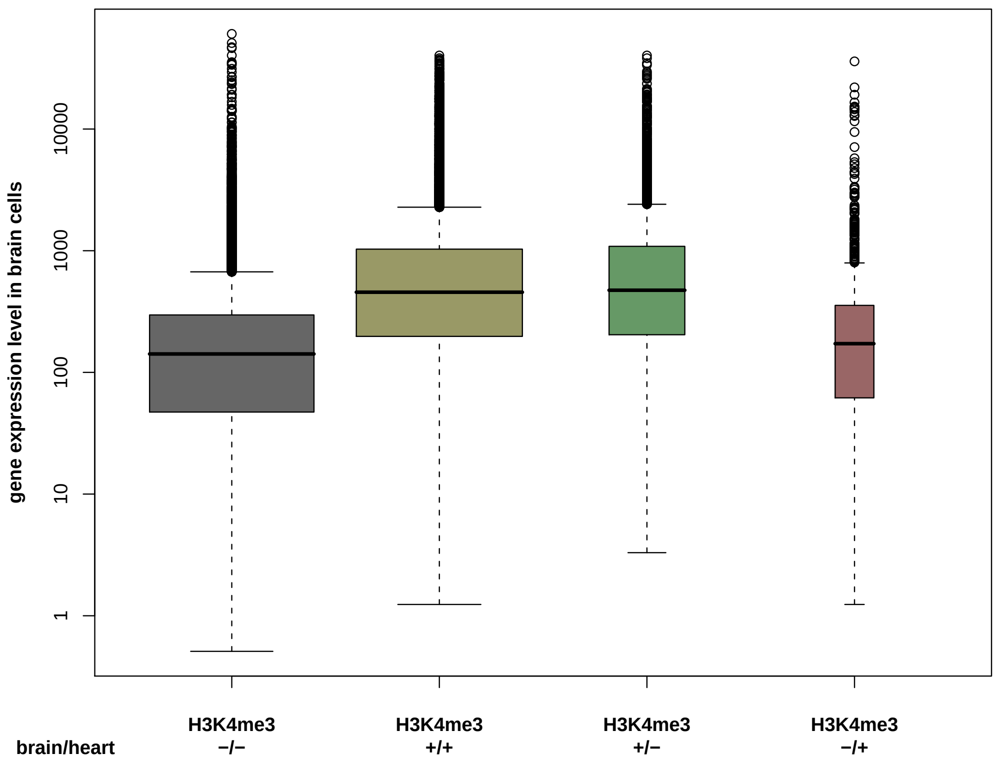

Figure 5: *Boxplots for comparing gene expression levels in brain cells. Genes are stratified by whether or not they are related to H3K4me3 ChIP-enriched regions in brain and/or heart cells according to ChIP-chip. The width of the boxes is proportional to the number of genes in each stratification group.*

See the boxplots in Figure 5. Genes related to H3K4me3 ChIP-enriched regions show higher expression levels than those that are not, as we can assess using the Wilcoxon rank sum test.

```
> with(brainExpression,
+      wilcox.test(H3K4me3BrainYesHeartNo, H3K4me3BrainNoHeartNo,
+                  alternative="greater"))
```

```
        Wilcoxon rank sum test with continuity correction

data:  H3K4me3BrainYesHeartNo and H3K4me3BrainNoHeartNo
W = 88159233, p-value < 2.2e-16
alternative hypothesis: true location shift is greater than 0
```

12 Discussion

We have shown how to use the freely available tools R and Bioconductor for the analysis of ChIP-chip data. We have shown ways to assess data quality, to visualize the data and to find ChIP-enriched regions.

Besides the ones introduced here, there are other Bioconductor packages that provide further functionality, e. g. *ACME* [27], *oligo* and *tilingArray* [20].

For analyses that go beyond pairwise comparisons of samples and use more complex (multi-)-factorial experimental designs or retrospective studies of collections of tissues from patients, the package *limma* [15] offers a powerful statistical modeling interface and facilitates computation of appropriate reporter-wise statistics.

We also demonstrated a few conceivable follow-up investigations. Bioconductor allows for easy integration of ChIP-chip results with other resources, such as annotated genome elements, gene expression data or DNA-protein interaction networks.

Software versions

This tutorial was generated using the following package versions:

- R version 2.8.0 Under development (unstable) (2008-05-28 r45808), `x86_64-unknown-linux-gnu`
- Locale: `LC_CTYPE=en_US.ISO-8859-1;LC_NUMERIC=C;LC_TIME=en_US.ISO-8859-1;LC_COLLATE=en_US.ISO-885`
- Base packages: base, datasets, graphics, grDevices, methods, splines, stats, tools, utils
- Other packages: affy 1.19.2, affyio 1.9.0, annotate 1.19.2, AnnotationDbi 1.3.6, Biobase 2.1.7, biomaRt 1.15.0, ccTutorial 0.9.4, codetools 0.2-1, DBI 0.2-4, digest 0.3.1, fortunes 1.3-5, genefilter 1.21.1, geneplotter 1.19.1, GO.db 2.2.0, graph 1.19.1, lattice 0.17-10, limma 2.15.9, preprocessCore 1.3.1, RColorBrewer 1.0-2, RCurl 0.9-3, Ringo 1.5.5, RSQLite 0.6-9, SparseM 0.78, survival 2.34-1, topGO 1.9.0, vsn 3.7.5, weaver 1.7.0, xtable 1.5-2
- Loaded via a namespace (and not attached): cluster 1.11.11, grid 2.8.0, KernSmooth 2.22-22, XML 1.95-2

Acknowledgments

We thank Richard Bourgon for valuable comments on the manuscript and Leah O. Barrera, Bing Ren and co-workers for making their ChIP-chip data publicly available. This work was supported by the European Union (FP6 HeartRepair, LSHM-CT-2005-018630).

References

[1] Buck MJ, Nobel AB, Lieb JD (2005) ChIPOTle: a user-friendly tool for the analysis of ChIP-chip data. Genome Biology 6:R97. 1

[2] Ji H, Wong WH (2005) TileMap: create chromosomal map of tiling array hybridizations. Bioinformatics 21:3629–3636. 1

[3] Johnson WE, Li W, Meyer CA, Gottardo R, Carroll JS, et al. (2006) Model-based analysis of tiling-arrays for ChIP-chip. Proc Natl Acad Sci USA 103:12457–12462. 1

[4] Keleş S (2007) Mixture modeling for genome-wide localization of transcription factors. Biometrics 63:10–21. 1

[5] Toedling J, Sklyar O, Krueger T, Sperling S, Fischer JJ, et al. (2007) Ringo - an R/Bioconductor package for analyzing ChIP-chip readouts. BMC Bioinformatics 8:221. 1

[6] Zheng M, Barrera LO, Ren B, Wu YN (2007) ChIP-chip: data, model, and analysis. Biometrics 63:787–796. 1

[7] Gentleman RC, Carey VJ, Bates DJ, Bolstad BM, Dettling M, et al. (2004) Bioconductor: Open software development for computational biology and bioinformatics. Genome Biology 5:R80. 1

[8] Durinck S, Moreau Y, Kasprzyk A, Davis S, Moor BD, et al. (2005) BioMart and Bioconductor: a powerful link between biological databases and microarray data analysis. Bioinformatics 21:3439–3440. 1, 4

[9] Alexa A, Rahnenführer J, Lengauer T (2006) Improved scoring of functional groups from gene expression data by decorrelating GO graph structure. Bioinformatics 22:1600–1607. 1, 11

[10] Gentleman R (2005) Reproducible research: A bioinformatics case study. Statistical Applications in Genetics and Molecular Biology 4:2. 1

[11] Knuth D (1992) Literate programming. Technical report, Center for the Study of Language and Information, Stanford, California. 1

[12] Santos-Rosa H, Schneider R, Bannister AJ, Sherriff J, Bernstein BE, et al. (2002) Active genes are tri-methylated at K4 of histone H3. Nature 419:407–11. 2

[13] Barrera LO, Li Z, Smith AD, Arden KC, Cavenee WK, et al. (2008) Genome-wide mapping and analysis of active promoters in mouse embryonic stem cells and adult organs. Genome Res 18:46–59. 2, 12

[14] Edgar R, Domrachev M, Lash AE (2002) Gene Expression Omnibus: NCBI gene expression and hybridization array data repository. Nucleic Acids Res 30:207–210. 2

[15] Smyth GK (2005) Limma: linear models for microarray data. In: Gentleman R, Carey V, Huber W, Irizarry R, Dudoit S, editors, Bioinformatics and Computational Biology Solutions Using R and Bioconductor. Springer, pp. 397–420. 3, 14

[16] Kauffmann A, Huber W (2008) arrayQualityMetrics: Quality metrics on microarray data sets. R package version 1.7.5. 3

[17] Slater GSC, Birney E (2005) Automated generation of heuristics for biological sequence comparison. BMC Bioinformatics 6:31. 3

[18] Birney E, Andrews TD, Bevan P, Caccamo M, Chen Y, et al. (2004) An overview of Ensembl. Genome Res 14:925–928. 4

[19] Ashburner M, Ball CA, Blake JA, Botstein D, Butler H, et al. (2000) Gene ontology: tool for the unification of biology. The Gene Ontology Consortium. Nat Genet 25:25–29. 4

[20] Huber W, Toedling J, Steinmetz LM (2006) Transcript mapping with high-density oligonucleotide tiling arrays. Bioinformatics 22:1963–1970. 5, 14

[21] Hamajima N, Matsuda K, Sakata S, Tamaki N, Sasaki M, et al. (1996) A novel gene family defined by human dihydropyrimidinase and three related proteins with differential tissue distribution. Gene 180:157–163. 5

[22] Royce TE, Rozowsky JS, Gerstein MB (2007) Assessing the need for sequence-based normalization in tiling microarray experiments. Bioinformatics 23:988–997. 5

[23] Schwartz YB, Kahn TG, Nix DA, Li XY, Bourgon R, et al. (2006) Genome-wide analysis of Polycomb targets in Drosophila melanogaster. Nat Genet 38:700–705. 9

[24] Bourgon RW (2006) Chromatin-immunoprecipitation and high-density tiling microarrays: a generative model, methods for analysis, and methodology assessment in the absence of a ”gold standard”. Ph.D. thesis, University of California, Berkley, Berkley, California, USA. URL `http://www.ebi.ac.uk/~bourgon/papers/bourgon_dissertation_public.pdf`. 9

[25] Kuan PF, Chun H, Keleş S (2008) CMARRT: a tool for the analysis of ChIP-chip data from tiling arrays by incorporating the correlation structure. Pac Symp Biocomput :515–526. 9

[26] Kanehisa M (1997) A database for post-genome analysis. Trends Genet 13:375–376. 12

[27] Scacheri PC, Crawford GE, Davis S (2006) Statistics for ChIP-chip and DNase hypersensitivity experiments on NimbleGen arrays. Methods Enzymol 411:270–282. 14