Supplement: A tutorial on how to analyze ChIP-chip readouts using Bioconductor

Joern Toedling, Wolfgang Huber

June 4, 2008

Contents

1 Introduction **2**

2 Importing the data into R **2**

3 Quality assessment **4**

4 Mapping reporters to the genome **5**

5 Genome annotation **6**

6 Preprocessing **8**

7 Preprocessed reporter intensities around the *Crmp1* gene **9**

8 Smoothing of reporter intensities **9**

9 Comparing ChIP-enrichment between the tissues **10**

9.1 Enriched-region-wise comparison 10

10 ChIP results and expression microarray data **12**

10.1 Preprocess the microarray expression data 13

10.2 Map Ensembl identifier to Affymetrix probe sets 13

1 Introduction

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

This document has been written in the `Sweave` [1] format, which combines explanatory text and the actual R source code that has been used in this analysis. One advantage of this format is that the analysis can easily be reproduced by the reader. An R package containing all the data and scripts used herein is available from the Bioconductor web site.

2 Importing the data into R

The provided data are measurements of enrichment for H3K4me3 in heart and brain cells. For each microarray, the scanning 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.

The microarray platform was a set of 4 arrays containing about 390k reporters each and meant to tile selected promoter regions in the *Mus musculus* genome (assembly *mm5*) with one base every 100bp. Thus for every sample, we have 8 files (4 arrays $\times$ 2 dyes).

```
> 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"
```

In addition, there is one text file for each array that holds details on the samples, including which two *pair* files belong to which sample.

```
> read.delim(file.path(pairDir, "files_array1.txt"), header = TRUE)
```

```
  SlideNumber    FileNameCy3    FileNameCy5                    DESIGN_NAME
1       48153 48153_532.pair 48153_635.pair 2005-06-17_Ren_MM5Tiling_Set1
2       48170 48170_532.pair 48170_635.pair 2005-06-17_Ren_MM5Tiling_Set1
  SAMPLE_SPECIES   Cy3     Cy5 Tissue
1   Mus_musculus input H3K4me3  brain
2   Mus_musculus input H3K4me3  heart
```

The columns `FileNameCy3` and `FileNameCy5` hold which of the raw data files belong to which sample. The immuno-precipitated extract was tagged with the Cy5 dye in the experiment, so the column `Cy5` essentially holds which antibody has been used for the immuno-precipitation, in this case one against the histone modification `H3K4me3`.

Furthermore, there is a file `spottypes.text` describing the reporter categories on the array (you may know these Spot Types files from the Bioconductor packgae *limma*[2]).

Reading all these files, we can read in the raw reporter intensities and obtain four objects of class *RGList*, a class defined in package *limma*. Each object contains the readouts from all samples measured on the same array.

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

An *RGList* object is essentially a list and contains the raw intensities of the two hybridizations for the red and green channel plus information on the reporters on the array and on the analyzed samples.

```
> head(RGs[[1]]$R)
```

```
     48153_635 48170_635
[1,]  18680.45  27445.45
[2,]  19510.55  28003.45
[3,]  19269.45  26622.55
[4,]    348.67  24435.33
[5,]    312.78    214.00
[6,]    348.89  25187.22
```

```
> head(RGs[[1]]$G)
```

```
     48153_532 48170_532
[1,]  65535.00  65535.00
[2,]  65535.00  65535.00
[3,]  62598.89  65535.00
[4,]   1197.89  65535.00
[5,]   1433.00   1781.56
[6,]   1608.56  61490.45
```

```
> tail(RGs[[1]]$genes)
```

```
       GENE_EXPR_OPTION        PROBE_ID POSITION   X   Y Status              ID
390606              MM5 MM5000P03209684 52759608 140  32  Probe MM5000P03209684
390607              MM5 MM5000P03370281 82709533 146 534  Probe MM5000P03370281
390608              MM5 MM5000P03410577 89884304  33 495  Probe MM5000P03410577
390609              MM5 MM5000P03428135 93306337 404 786  Probe MM5000P03428135
390610              MM5 MM5000P03152702 42800575 548 836  Probe MM5000P03152702
390611              MM5 MM5000P03113901 35200666 687 907  Probe MM5000P03113901
```

Among the read-in values are those coming from reporters[1] matching the genome sequence as well as some from "control" reporters put on the array by the manufacturer.

[1] the misleading slot name "genes" is due to historical reasons, dating back to the time when cDNA microarrays were mostly used to measure gene expression. In our case, each reporter is not associated to one gene but to one or more genomic locations.

```
> table(RGs[[1]]$genes$Status)
```

```
  H_Code Negative    Probe   Random   V_Code
      40     2063   373896    14572       40
```

The *RGList* is a common class for raw two-color data, so the following steps can easily be applied to other, non-NimbleGen microarrays, which for example can be read in into R using *limma*'s function `read.maimages`.

3 Quality assessment

First, we look at the spatial distribution of the intensities on the array. This can be useful for detecting obvious artifacts on the array, such as scratches, bright spots, finger prints etc., which may render parts or all of the readouts of one hybridization useless.

We construct one picture showing the spatial distributions for all arrays and both channels.

```
> RG1breaks <- c(0, quantile(RGs[[1]]$G, probs = seq(0, 1, by = 0.1)),
+     2^16)
> layout(matrix(c(1, 2, 5, 6, 3, 4, 7, 8, 9, 10, 13, 14, 11, 12,
+     15, 16), ncol = 4, byrow = TRUE))
> for (this.set in 1:4) {
+     thisRG <- RGs[[this.set]]
+     for (this.channel in c("green", "red")) {
+         my.colors <- colorRampPalette(c("black", paste(this.channel,
+             c(4, 1), sep = "")))(length(RG1breaks) - 1)
+         for (arrayno in 1:2) {
+             image(thisRG, arrayno, channel = this.channel, mybreaks = RG1breaks,
+                 mycols = my.colors)
+         }
+     }
+ }
```

See Figure S1 for the image.

Minor artifacts can be seen. The arrays of the first set (48153 and 48172) show a blotch of lower intensities in the upper right area of the array. These artifacts affect only a small part of the array and thus probably have a negligible effect on the results. The reporters in those affected areas of the array will yield meaningless readouts but enriched regions will be determined based on a set of multiple reporters that are distributed over the microarray surface and not on single reporters only.

It may also be useful to look at the absolute distribution of the single-channel densities.

```
> plotDensities(RGs[[3]])
```

See Figure S2 for the densities. We see some obvious differences between red and green channel intensities. These effects could be due to a larger amounts of DNA being hybridized. The preprocessing step later on is able to correct for such shifts.

On all arrays in our set, the Cy3 channel holds the intensities from the untreated input sample, and the Cy5 channel holds the ChIP result for heart and heart, respectively. We investigate whether this experiment setup is reflected in the reporter intensity correlation per channel (see Figure S3). Compare these two plots:

```
> corPlot(log2(RGs[[2]]$G))
> corPlot(log2(RGs[[2]]$R))
```

See Figure S3 for plots comparing the two arrays. In the scatter plots of raw reporter intensities, the fraction of dots at the diagonal is higher for the *input* samples than for the ChIP samples. Concordantly, the correlation between the intensities of the *input* samples is higher than between the ChIP samples (0.877 versus 0.734).

4 Mapping reporters to the genome

We extracted the reporter nucleotide sequences from the downloaded NDF (NimbleGen Design Files) files. We re-mapped the reporter sequences to the genome, using the alignment tool *Exonerate* [3]. We required 97% sequence identity for a match[2]. Exonerate was run matching the reporter sequences in the Fasta file `RenMM5TilingProbeSequences.fsa` against each chromosome's sequence using the shell script `runExonerate.sh` and then condensing the per-chromosome output files into one single file using the Perl script `condenseExonerateOutput.pl`[3].

From this result file, we can construct an object of class *probeAnno* to store the mapping between reporters and genome positions.

```
> probeAnno <- posToProbeAnno(file.path(system.file("exonerateData",
+     package = "ccTutorial"), "allChromExonerateOut.txt"))
> allChrs <- chromosomeNames(probeAnno)

> genome(probeAnno) <- "M. musculus (mm9)"
> arrayName(probeAnno) <- "2005-06-17_Ren_MM5Tiling"

> show(probeAnno)

A 'probeAnno' object holding the mapping between
reporters and genomic positions.
Chromosomes: 10 11 12 13 14 15 16 17 18 19 1 2 3 4 5 6 7 8 9 X Y
Microarray platform: 2005-06-17_Ren_MM5Tiling
Genome: M. musculus (mm9)
```

[2]Remapping 1.5 million reporters took about 100 processor hours on an AMD Opteron Processor 275.
[3]all the scripts mentioned here are included in the `scripts` directory of the package

```
> ls(probeAnno)
```

```
 [1] "10.end"    "10.index"  "10.start"  "10.unique" "11.end"    "11.index"
 [7] "11.start"  "11.unique" "12.end"    "12.index"  "12.start"  "12.unique"
[13] "13.end"    "13.index"  "13.start"  "13.unique" "14.end"    "14.index"
[19] "14.start"  "14.unique" "15.end"    "15.index"  "15.start"  "15.unique"
[25] "16.end"    "16.index"  "16.start"  "16.unique" "17.end"    "17.index"
[31] "17.start"  "17.unique" "18.end"    "18.index"  "18.start"  "18.unique"
[37] "19.end"    "19.index"  "19.start"  "19.unique" "1.end"     "1.index"
[43] "1.start"   "1.unique"  "2.end"     "2.index"   "2.start"   "2.unique"
[49] "3.end"     "3.index"   "3.start"   "3.unique"  "4.end"     "4.index"
[55] "4.start"   "4.unique"  "5.end"     "5.index"   "5.start"   "5.unique"
[61] "6.end"     "6.index"   "6.start"   "6.unique"  "7.end"     "7.index"
[67] "7.start"   "7.unique"  "8.end"     "8.index"   "8.start"   "8.unique"
[73] "9.end"     "9.index"   "9.start"   "9.unique"  "X.end"     "X.index"
[79] "X.start"   "X.unique"  "Y.end"     "Y.index"   "Y.start"   "Y.unique"
```

5 Genome annotation

Later on, we are going to relate found enriched regions to annotated genome features, such as gene start and end positions. Using the Bioconductor package *biomaRt* [4], we can obtain an up-to-date annotation of the mouse genome from the Ensembl data base [5] (release 38, December 2007).

```
> ensembl <- useMart("ensembl", dataset = "mmusculus_gene_ensembl")
> gene.ids <- unique(unlist(lapply(as.list(c(1:19, "X", "Y")),
+     function(this.chr) getBM(attributes = "ensembl_gene_id",
+         filters = "chromosome_name", values = this.chr, mart = ensembl)[,
+         1]), use.names = FALSE))
> sel.attributes = c("ensembl_gene_id", "mgi_symbol", "chromosome_name",
+     "strand", "start_position", "end_position", "description")
> mm9genes <- getBM(attributes = sel.attributes, filters = "ensembl_gene_id",
+     value = gene.ids, mart = ensembl)
```

For later ease of use, we replace the formal element names retrieved from the data base by simpler ones.

```
> mm9genes$name <- mm9genes$ensembl_gene_id
> mm9genes$gene <- mm9genes$ensembl_gene_id
> mm9genes$chr <- mm9genes$chromosome_name
> mm9genes$symbol <- mm9genes$mgi_symbol
> mm9genes$start <- mm9genes$start_position
> mm9genes$end <- mm9genes$end_position
> mm9genes$feature <- rep("gene", nrow(mm9genes))
```

Some genes occur in multiples in the table because an Ensembl gene can have more than one MGI Symbol defined for it. We keep allow only one row in the table per gene and append additional MGI symbols to the *description* element of each gene.

```
> if (any(duplicated(mm9genes$name))) {
+     dupl <- unique(mm9genes$name[duplicated(mm9genes$name)])
+     G <- lapply(as.list(dupl), function(this.gene) {
+         this.gff <- subset(mm9genes, name == this.gene)
+         if (nrow(unique(this.gff[, c("name", "chr", "start",
+             "end", "description")])) > 1)
+             return(this.gff[1, , drop = FALSE])
+         non.zero.gff <- subset(this.gff, nchar(symbol) > 0)
+         this.other.sym <- NULL
+         if (nrow(non.zero.gff) > 0) {
+             shortest <- which.min(nchar(non.zerogff$symbol))
+             this.new.sym <- non.zero.gff$symbol[shortest]
+             if (nrow(non.zero.gff) > 1)
+                 this.other.sym <- paste("Synonyms", paste(non.zero.gff$symbol[-shortest],
+                   collapse = ","), sep = ":")
+         }
+         else {
+             this.new.sym <- ""
+         }
+         this.gff$symbol[1] <- this.new.sym
+         if (!is.null(this.other.sym))
+             this.gff$description[1] <- paste(this.gff$description[1],
+                 this.other.sym, sep = ";")
+         return(this.gff[1, , drop = FALSE])
+     })
+     mm9genes <- rbind(mm9genes[-which(mm9genes$name %in% dupl),
+         ], do.call("rbind", G))
+ }
```

Finally, we reorder the table rows by gene chromosome and start position.

```
> mm9genes <- mm9genes[order(mm9genes$chr, mm9genes$start), c("name",
+     "chr", "strand", "start", "end", "symbol", "description",
+     "feature")]
> rownames(mm9genes) <- NULL
```

The resulting table holds the coordinates, Ensembl gene identifiers, MGI symbols, and description of all the genes annotated for the *mm9* mouse assembly. Have a look at a few example lines from the table.

```
> mm9genes[sample(seq(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

We also retrieve the Gene Ontology (GO, [6]) annotation for each gene, but discard those annotations that have only been *inferred from electronic annotation* (evidence code: IEA), are based on a *non-traceable author statement* (NAS) or for which there is *no biological data* (ND) available.

```
> ensembl <- useMart("ensembl", dataset = "mmusculus_gene_ensembl")
> mm9GO <- getBM(attributes = c("ensembl_gene_id", "go", "evidence_code"),
+     filters = "ensembl_gene_id", value = mm9genes$name, mart = ensembl)
> mm9GO <- subset(mm9GO, !(evidence_code %in% c("", "IEA", "NAS",
+     "ND")))
> mm9.gene2GO <- with(mm9GO, split(go, ensembl_gene_id))
```

Finally, we create a mapping of gene identifiers to reporters that had been mapped into the gene or its upstream region.

```
> mm9.g2p <- features2Probes(gff = mm9genes, probeAnno = probeAnno)
```

```
> table(cut(listLen(mm9.g2p), breaks = c(-1, 0, 10, 50, 100, 500,
+     1200)))
```

```
       (-1,0]        (0,10]       (10,50]      (50,100]     (100,500]
        10790           640          7341          5949          2675
(500,1.2e+03]
           39
```

This last table shows how many genes have that number of reporters mapped into their upstream region or inside of them. The numbers of reporters are given in open interval notation with, e.g., (10,50] meaning 11 to 50 reporters.

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.

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

6 Preprocessing

Following quality assessment of the raw data, we derive log2 fold changes Cy5/Cy3 for each reporter and scale these by subtracting Tukey's biweight mean from each log2 ratio, the standard scaling procedure suggested by NimbleGen. Each of the four microarrays used contains a unique set of reporters. Thus, we preprocess the arrays separately by type and only then 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 design (four in this case).

```
> show(X)

ExpressionSet (storageMode: lockedEnvironment)
assayData: 1495582 features, 2 samples
  element names: exprs
phenoData
  sampleNames: H3K4me3.brain, H3K4me3.heart
  varLabels and varMetadata description:
    SlideNumber: NA
    FileNameCy3: NA
    ...: ...
    Tissue: NA
    (8 total)
  additional varMetadata: varLabel
featureData
  featureNames: 16716, 16717, ..., 3906113  (1495582 total)
  fvarLabels and fvarMetadata description:
    GENE_EXPR_OPTION: NA
    PROBE_ID: NA
    ...: ...
    ID: NA
    (7 total)
experimentData: use 'experimentData(object)'
Annotation:
```

7 Preprocessed reporter intensities around the *Crmp1* gene

We visualize the preprocessed H3K4me3 ChIP-chip reporter-wise readouts around the start of the *Crmp1* gene. H3K4me3 has frequently been shown to be associated to active transcription (e.g., [7]) and the gene Crmp1 has been reported as being expressed in brain cells [8].

```
> chipAlongChrom(X, chrom = "5", xlim = c(37630000, 37640000),
+     probeAnno = probeAnno, gff = mm9genes, ylim = c(-3, 5), paletteName = "Set2")
```

See the result in Figure S4. In brain cells, the intensities for enrichment of H3K4me3 around the gene's start position tend to be positive, while the signal for heart cells is around or below zero.

8 Smoothing of reporter intensities

```
> 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 non-smoothed ones around the start of the gene *Crmp1*.

```
> chipAlongChrom(X, chrom = "5", xlim = c(37630000, 37640000),
+     probeAnno = probeAnno, gff = mm9genes, paletteName = "Set2",
+     ylim = c(-3, 5))
> chipAlongChrom(smoothX, chrom = "5", xlim = c(37630000, 37640000),
+     probeAnno = probeAnno, ilwd = 4, paletteName = "Dark2", add = TRUE)
```

See Figure S5 for a comparison of the original and smoothed reporter levels around the gene *Crmp1*.

9 Comparing ChIP-enrichment between the tissues

First, we have taken a gene-centric position and consider which genes are associated to each tissue specifically.

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

9.1 Enriched-region-wise comparison

We compute the base-pair overlap between enriched regions found in brain cells with those found in heart cells. We define that region A is said to *overlap* with region B if

$$length(A \cap B) \quad \geq \quad 0.7 \cdot \min(length(A), length(B))$$

where $length(X)$ denotes the length of region X in nucleotides. We define an enriched region as tissue-specific if it does not overlap with any region from another tissue according to the definition above.

```
> brainRegions <- subset(chersXD, cellType == "brain")
> heartRegions <- subset(chersXD, cellType == "heart")
> chersOBL <- as.matrix(regionOverlap(brainRegions, heartRegions))
> minRegLen <- outer(with(brainRegions, end - start + 1), with(heartRegions,
+     end - start + 1), pmin)
> fracOverlap <- chersOBL/minRegLen
```

```
> brainSpecReg <- brainRegions[rowMax(fracOverlap) < 0.7, ]
> heartSpecReg <- heartRegions[rowMax(t(fracOverlap)) < 0.7, ]
> mean(is.element(unlist(strsplit(brainSpecReg$features, split = "[[:space:]]"),
+     use.names = FALSE), brainOnlyGenes))
```

```
[1] 0.799639
```

```
> selGenes <- intersect(unlist(strsplit(brainSpecReg$features,
+     split = "[[:space:]]"), use.names = FALSE), heartGenes)
```

Note that only 80% of the genes related to non-overlapping ChIP-enriched regions show such regions in brain cells only. The other genes show such regions in both tissues but their positions differ between the tissues.

We can assess whether these 690 genes show a typical positioning of H3K4me3 to each other, such as 'in heart cells they display H3K4me3 enriched regions upstream of the genes, while in brain cells the show H3K4me3 between gene start and stop coordinates'. We use the genes-to-reporters mapping that we have created earlier for this investigation.

```
> targetPos <- seq(-5000, 10000, by = 250)
```

For each gene, we assess the genomic region from 5kb upstream to 10 kb downstream of the gene start.

```
> selGenesVal <- vector("list", length(selGenes))
> for (i in 1:length(selGenes)) {
+     if (i%%1000 == 0)
+         cat(i, "... ")
+     ix <- mm9.g2p[[selGenes[i]]]
+     if (length(ix) == 0)
+         next
+     iy <- exprs(smoothX)[names(ix), , drop = FALSE]
+     if (sum(!is.na(iy[, 1])) < 2)
+         next
+     iy <- apply(iy, 2, function(a) approx(x = ix, y = a, xout = targetPos)$y)
+     selGenesVal[[i]] <- iy
+ }
> names(selGenesVal) <- selGenes
```

```
> quants <- c(0.1, 0.3, 0.5, 0.7, 0.9)
> valsBySample <- lapply(list("H3K4me3.brain.smoothed", "H3K4me3.heart.smoothed"),
+     function(thisSample) {
+         do.call("cbind", lapply(selGenesVal, function(theseValues) {
+             return(theseValues[, thisSample, drop = FALSE])
+         }))
+     })
> names(valsBySample) <- c("H3K4me3.brain.smoothed", "H3K4me3.heart.smoothed")
> quantsBySample <- lapply(valsBySample, function(theseValues) apply(theseValues,
+     1, quantile, probs = quants, na.rm = TRUE))
```

We normalize these densities of the observed fold changes by genomic positions by the densities of mapped reporters.

```
> probePosDens <- density(unlist(mm9.g2p, use.names = FALSE))
> probePosY <- approx(x = probePosDens$x, y = probePosDens$y, xout = targetPos)$y
> probePosY <- probePosY/max(probePosY)


> plot(x = 0, y = 0, ylim = c(-0.2, 3.5), xaxt = "n", xlim = range(targetPos),
+     type = "n", xlab = "Distance to gene start [bp]", ylab = "smoothed intensity")
> axis(side = 1, at = seq(-5000, 10000, by = 1000), labels = paste(-5:10,
+     "kb", sep = ""))
> mycolors <- c("green", "orange")
> for (i in 1:2) {
+     lines(x = targetPos, y = quantsBySample[[i]][3, ] * probePosY,
+         lwd = 2, col = mycolors[i])
+     lines(x = targetPos, y = quantsBySample[[i]][5, ] * probePosY,
+         lwd = 2, lty = 2, col = mycolors[i])
+ }
> legend(x = "topleft", fill = c("green", "orange"), legend = c("H3K4me3 brain",
+     "H3K4me3 heart"), bty = "n")
> legend(x = "topright", lty = c(2, 1), lwd = 2, legend = c("90% quantile",
+     "median"), bty = "n")
```

See Figure S6 for the densities. There are no clear tissue-wise trends where these enriched regions are in relation to the gene start coordinate. In both tissues, the smoothed intensities on average are highest within 1kb after the gene start coordinate, while in brain cells the density shows a second, smaller cher within 1kb upstream of the gene start.

We also investigate genes that have separate enriched regions in both tissues for over-represented GO annotations.

```
> sepRegRes <- sigGOTable(selGenes = selGenes)
> print(sepRegRes)
```

GO.ID	Term	Annotated	Significant	Expected	p.value
GO:0045648	positive regulation of erythrocyte differentiation	5	4	0.26	3.5e-05
GO:0045176	apical protein localization	8	4	0.42	0.00043
GO:0007406	negative regulation of neuroblast proliferation	4	3	0.21	0.00054
GO:0050803	regulation of synapse structure and activity	23	6	1.20	0.00090

Table S1: *GO terms that are significantly over-represented among genes that show different H3K4me3 regions in heart and brain cells*

See the results in Table S1.

10 ChIP results and expression microarray data

Barrera et al. [9] also provide expression microarray data for five their analyzed *M. musculus* tissues.

10.1 Preprocess the microarray expression data

The data were obtained from the publication's supplementary web page and were then imported into R and preprocessed as follows

```
> library("affy")
> library("mouse4302cdf")
> AB <- ReadAffy(celfile.path=system.file("expression", package="ccTutorial"))
> barreraExpressionX <- mas5(AB)
> barreraExpressionX$Tissue <- sapply(
+    strsplit(sampleNames(barreraExpressionX),split="\\."),"[",3)
```

10.2 Map Ensembl identifier to Affymetrix probe sets

The expression data was generated using the `Mouse_430_2` oligonucleotide microarray platform from Affymetrix. Using biomaRt, we create a mapping of Ensembl gene identifiers to the probe set identifiers on that microarray design.

```
> ensembl <- useMart("ensembl", dataset="mmusculus_gene_ensembl")

Checking attributes and filters ... ok

> bmRes <- getBM(attributes=c("ensembl_gene_id","affy_mouse430_2"),
+                filters="ensembl_gene_id", value=arrayGenes, mart=ensembl)
> bmRes <- subset(bmRes, nchar(affy_mouse430_2)>0)
> arrayGenesToProbeSets <- split(bmRes$"affy_mouse430_2",
+                                bmRes$"ensembl_gene_id")
```

How many probe sets are mapped to each gene?

```
> table(listLen(arrayGenesToProbeSets))

   1    2    3    4    5    6    7    8    9   11   12
6252 4625 2215  977  377  133   47   12    3    1    1
```

Software versions

This supplement 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 1.17.2, affyio 1.5.11, annotate 1.17.2, AnnotationDbi 1.3.0, Biobase 2.1.0, biomaRt 1.11.15, ccTutorial 0.9.1, codetools 0.2-1, DBI 0.2-4, digest 0.3.1, fortunes 1.3-3, genefilter 1.21.0, geneplotter 1.19.0, GO.db 2.2.0, graph 1.19.0, lattice 0.17-8, limma 2.15.0, preprocessCore 0.99.21, RColorBrewer 1.0-2, RCurl 0.9-3, Ringo 1.5.3, RSQLite 0.6-8, SparseM 0.77, survival 2.34-1, topGO 1.8.1, vsn 3.7.0, weaver 1.5.0, xtable 1.5-2
- Loaded via a namespace (and not attached): cluster 1.11.8, grid 2.8.0, KernSmooth 2.22-22, XML 1.92-1

Acknowledgments

This work was supported by the European Union (FP6 HeartRepair, LSHM-CT-2005-018630).

References

[1] Leisch F (2002) Dynamic generation of statistical reports using literate data analysis. In: Haerdle W, Roenz B, editors, Compstat 2002 - Proceedings in Computational Statistics. Physika Verlag, Heidelberg, Germany, pp. 575–580. 2

[2] 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

[3] Slater GSC, Birney E (2005) Automated generation of heuristics for biological sequence comparison. BMC Bioinformatics 6:31. 5

[4] 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. 6

[5] Birney E, Andrews TD, Bevan P, Caccamo M, Chen Y, et al. (2004) An overview of Ensembl. Genome Res 14:925–928. 6

[6] 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. 8

[7] Fischer JJ, Toedling J, Krueger T, Schueler M, Huber W, et al. (2008) Combinatorial effects of four histone modifications in transcription and differentiation. Genomics 91:41–51. 9

[8] 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. 9

[9] 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. 12

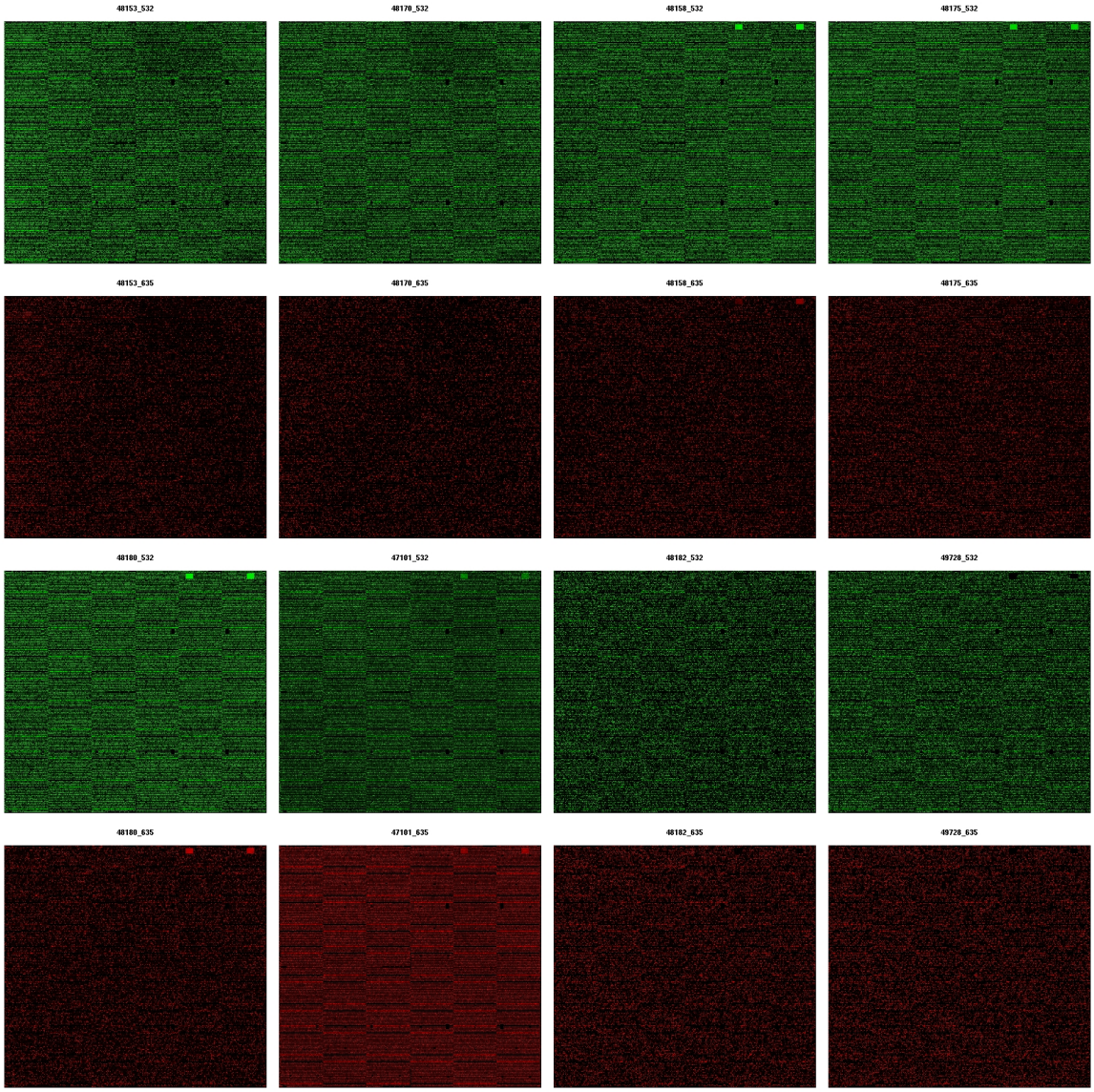


Figure S1: *Spatial distribution of raw reporter intensities laid out by the reporter position on the microarray surface. Each pair of one green and one red image on top of each other are the Cy3 and Cy5 readouts of the same hybridized microarray.*

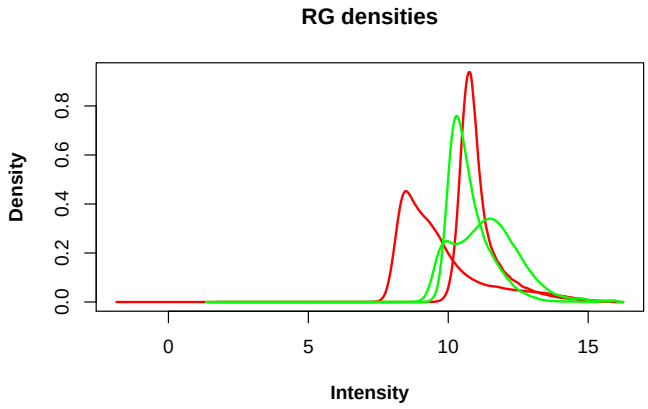
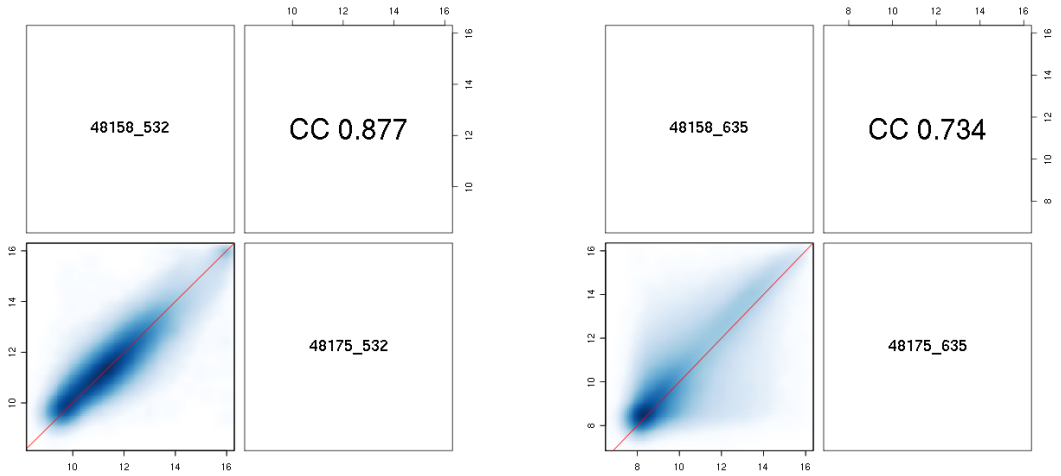


Figure S2: *Densities of the raw Cy3 (green, input) and Cy5 (red, ChIP sample) intensities as read from the third microarray of the brain and heart samples, respectively*



(a) (b)

Figure S3: *Scatterplot and Spearman correlation of the raw intensities from the two microarrays for* (a) *the Cy3 channel, the genomic* input *samples* (b) *the Cy5 channel, the H3K4me3-ChIP sample for* M. musculus *brain and heart cells.*

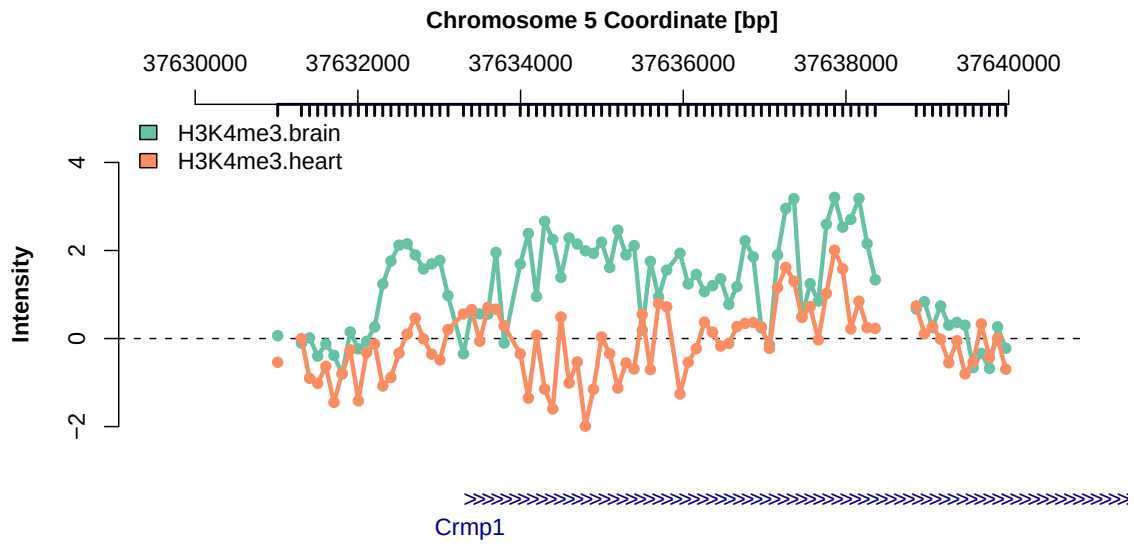


Figure S4: *Normalized reporter intensities for H3K4me3 ChIP around the TSS of the* `Crmp1` *gene in* M. musculus *brain and heart cells. The ticks below the genomic coordinate axis on top indicate genomic positions matched by reporters on the microarray. The blue arrows on the bottom mark the Crmp1 gene with the arrow direction indicating that the gene is located on the Watson strand.*

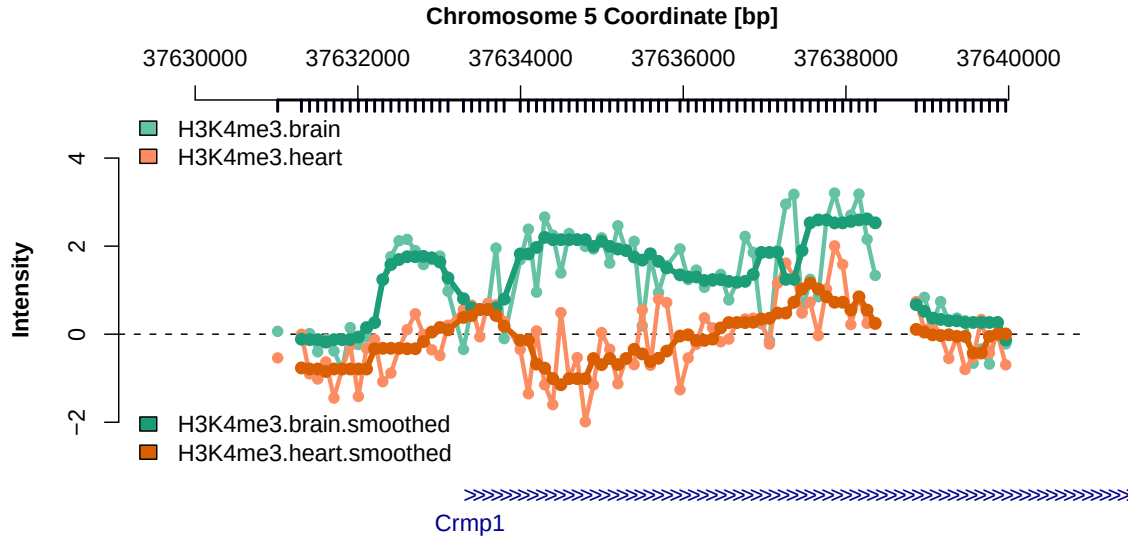


Figure S5: *Normalized and smoothed reporter intensities for H3K4me3 ChIP around the TSS of the gene* Crmp1 *in* M. musculus *brain and heart cells.*

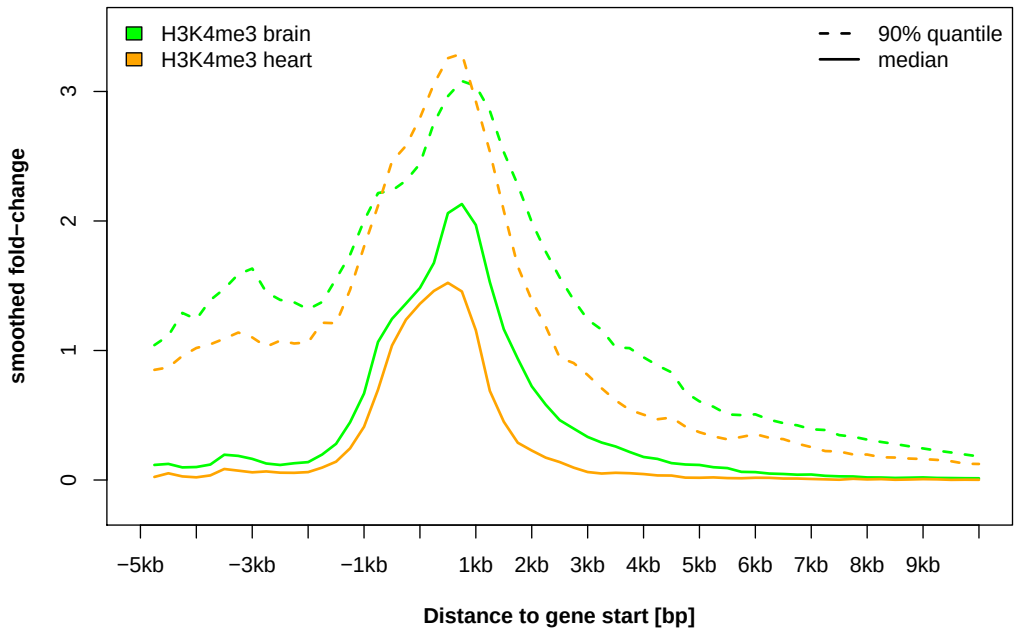


Figure S6: *Densities of selected quantiles of the smoothed fold-changes for H3K4me3 ChIP in* M. musculus *brain and heart cells for genes that show H3K4me3 enriched regions in both tissues but in separate positions.*