Ringo - R Investigation of NimbleGen Oligoarrays

Joern Toedling

May 3, 2016

1 Introduction

The package *Ringo* deals with the analysis of two-color oligonucleotide microarrays used in ChIP-chip projects. The package was started to facilitate the analysis of two-color microarrays from the company NimbleGen[1], but the package has a modular design, such that the platform-specific functionality is encapsulated and analogous two-color tiling array platforms can also be processed. The package employs functions from other packages of the Bioconductor project (Gentleman et al., 2004) and provides additional ChIP-chip-specific and NimbleGen-specific functionalities.

```
> library("Ringo")
```

If you use Ringo for analyzing your data, please cite:

- Joern Toedling, Oleg Sklyar, Tammo Krueger, Jenny J Fischer, Silke Sperling, Wolfgang Huber (2007). Ringo - an R/Bioconductor package for analyzing ChIP-chip readouts. *BMC Bioinformatics*, 8:221.

Getting help

If possible, please send questions about *Ringo* to the Bioconductor mailing list.
See http://www.bioconductor.org/docs/mailList.html
Their archive of questions and responses may prove helpful, too.

2 Reading in the raw data

For each microarray, the scanning output consists of two files, one holding the Cy3 intensities, the other one the Cy5 intensities. These files are tab-delimited text files.

The package comes with (shortened) example scanner output files, in NimbleGen's *pair* format. These files are excerpts of the ChIP-chip demo data that NimbleGen provide at

[1] for NimbleGen one-color microarrays, we recommend the Bioconductor package *oligo*

their FTP site for free download. Their biological context, identification of DNA binding sites of complexes containing Suz12 in human cells, has been described before (Squazzo et al., 2006).

```
> exDir <- system.file("exData",package="Ringo")
> list.files(exDir, pattern="pair.txt")
```

```
[1] "MOD_20551_PMT1_pair.txt" "MOD_20742_PMT1_pair.txt"
```

```
> head(read.delim(file.path(exDir,"MOD_20551_PMT1_pair.txt"),
+                 skip=1))[,c(1,4:7,9)]
```

```
    IMAGE_ID          PROBE_ID POSITION   X   Y      PM
1 20551_PMT1 SUZ100P0000021781        1 269  78 1149.33
2 20551_PMT1 SUZ100P0000021783       16 682 779 1192.00
3 20551_PMT1 SUZ100P0000021785       31  92 405  685.56
4 20551_PMT1 SUZ100P0000021787       46 219 608  562.67
5 20551_PMT1 SUZ100P0000021789       61 217 418  584.56
6 20551_PMT1 SUZ100P0000021791       76 147 406  636.22
```

In addition, there is a text file that holds details on the samples, including which two *pair* files belong to which sample[2].

```
> read.delim(file.path(exDir,"example_targets.txt"), header=TRUE)
```

```
  SlideNumber             FileNameCy3             FileNameCy5
1       Suz12 MOD_20551_PMT1_pair.txt MOD_20742_PMT1_pair.txt
               Species   Cy3   Cy5
1 Homo sapiens (human) total Suz12
```

The columns `FileNameCy3` and `FileNameCy5` hold which of the raw data files belong to which sample. The immuno-precipitated extract was colored 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 protein `Suz12`.

Furthermore, there is a file describing the reporter categories on the array (you might know these Spot Types files from *limma* (Smyth, 2005))[3]

```
> read.delim(file.path(exDir,"spottypes.txt"), header=TRUE)
```

[2] You may have to construct such a targets file for your own data. The `scripts` directory of this package contains a script `convertSampleKeyTxt.R` as an inspiration how the file `SampleKey.txt` provided by NimbleGen could be used for this.

[3] The spot types file is usally not provided by the array manufacturer, but needs to be created manually. You can use the file that comes with the package as a template and extend it as needed. See: <your-install-directory-of-Rpackages>/Ringo/exData/spottypes.txt.

```
  SpotType GENE_EXPR_OPTION PROBE_ID  Color
1    Probe         FORWARD*        *  black
2    Probe         REVERSE*        *  black
3    Probe           BLOCK*        *  black
4 Negative    NGS_CONTROLS*        * yellow
5    Empty            EMPTY        *  white
6   H_Code           H_CODE        *    red
7   V_Code           V_CODE        *   blue
8   Random           RANDOM        *  green
```

Reading all these files, we can read in the raw reporter intensities and obtain an object of class *RGList*, a class defined in package *limma*.

```
> exRG <- readNimblegen("example_targets.txt","spottypes.txt",path=exDir)
```

This 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(exRG$R)
```

```
     MOD_20742_PMT1_pair
[1,]              613.22
[2,]              841.67
[3,]              659.56
[4,]              494.44
[5,]              469.33
[6,]              544.11
```

```
> head(exRG$G)
```

```
     MOD_20551_PMT1_pair
[1,]             1149.33
[2,]             1192.00
[3,]              685.56
[4,]              562.67
[5,]              584.56
[6,]              636.22
```

```
> head(exRG$genes)
```

```
  GENE_EXPR_OPTION          PROBE_ID POSITION   X   Y Status                ID
1         FORWARD1 SUZ100P0000021781        1 269  78  Probe SUZ100P0000021781
2         FORWARD1 SUZ100P0000021783       16 682 779  Probe SUZ100P0000021783
3         FORWARD1 SUZ100P0000021785       31  92 405  Probe SUZ100P0000021785
4         FORWARD1 SUZ100P0000021787       46 219 608  Probe SUZ100P0000021787
5         FORWARD1 SUZ100P0000021789       61 217 418  Probe SUZ100P0000021789
6         FORWARD1 SUZ100P0000021791       76 147 406  Probe SUZ100P0000021791
```

```
> exRG$targets
```

```
  SlideNumber             FileNameCy3             FileNameCy5
1       Suz12 MOD_20551_PMT1_pair.txt MOD_20742_PMT1_pair.txt
               Species   Cy3   Cy5
1 Homo sapiens (human) total Suz12
```

Users can alternatively supply raw two-color ChIP-chip readouts from other platforms in *RGList* format and consecutively use *Ringo* to analyze that data. See Section 9 for an example.

3 Mapping reporters to genomic coordinates

By *reporters*, we mean the oligo-nucleotides or PCR products that have been fixated on the array for measuring the abundance of corresponding genomic fragments in the ChIP-chip experiment.

Each reporter has a unique identifier and (ideally) a unique sequence, but can, and probably does, appear in multiple copies as *features* on the array surface.

A mapping of reporters to genomic coordinates is usually provided by the array manufacturer, such as in NimbleGen's *.POS files. If the reporter sequences are provided as well, you may consider to perform a custom mapping of these sequences to the genome of interest, using alignment tools such as *Exonerate* (Slater and Birney, 2005) or functions provided by the Bioconductor package *Biostrings* (Pages et al., 2008).

Such a re-mapping of reporters to the genome can sometimes be necessary, for example when the array has designed on an outdated assembly of the genome. Re-mapping also provides the advantage that you can allow non-perfect matches of reporters to the genome, if desired.

Once reporters have been mapped to the genome, this mapping needs to be made available to the data analysis functions. While a *data.frame* may be an obvious way of representing such a mapping, repeatedly extracting sub-sets of the data frame related to a genomic region of interest turns out to be too slow for practical purposes. *Ringo*, similar to the Bioconductor package *tilingArray*, employs an object of class *probeAnno* to store the mapping between reporters on the microarray and genomic positions. Per chromosome, the object holds four vectors of equal length and ordering that specify at which genomic positions reporter matches start and end, what identifiers or indices these reporters have in the intensities data, and whether these reporters match uniquely to the genomic positions.

```
> load(file.path(exDir,"exampleProbeAnno.rda"))
> ls(exProbeAnno)
```

```
[1] "9.end"    "9.index"  "9.start"  "9.unique"
```

```
> show(exProbeAnno)
```

```
A 'probeAnno' object holding the mapping between
reporters and genomic positions.
Chromosomes: 9
Microarray platform: NimbleGen MOD_2003-12-05_SUZ12_1in2
Genome: H.sapiens (hg18)
```

```
> head(exProbeAnno["9.start"])
```

```
[1]  531857  861532 1269645 2497214 2685042 3795381
```

```
> head(exProbeAnno["9.end"])
```

```
[1]  531916  861591 1269704 2497273 2685101 3795440
```

The function `posToProbeAnno` allows generation of a valid *probeAnno* object, either from a file that corresponds to a NimbleGen `POS` file or from a *data.frame* objects that holds the same information. The package's `scripts` directory contains a script `mapReportersWithBiostrings.R`, which shows how to use *Biostrings* for mapping the reporter sequences of the provided example data, and some Perl scripts that allow the conversion of multiple output files from common alignment tools such as Exonerate into one file that corresponds to a POS file. The function `validObject` can be used to perform a quick check whether a generated *probeAnno* object will probably work with other *Ringo* functions.

4 Quality assessment

The `image` function allows us to look at the spatial distribution of the intensities on a chip. This can be useful to detect obvious artifacts on the array, such as scratches, bright spots, finger prints etc. that might render parts or all of the readouts useless.

```
> par(mar=c(0.01,0.01,0.01,0.01), bg="black")
> image(exRG, 1, channel="green", mycols=c("black","green4","springgreen"))
```

See figure 1 for the image. Since the provided example data set only holds the intensities for reporters mapped to the forward strand of chromosome 9, the image only shows the few green dots of these reporters' positions. We see, however, that these chromosome 9 reporters are well distributed over the whole array surface rather than being clustered together in one part of the array.

It may also be useful to look at the absolute distribution of the single-channel densities. *limma*'s function `plotDensities` may be useful for this purpose.

```
> plotDensities(exRG)
```

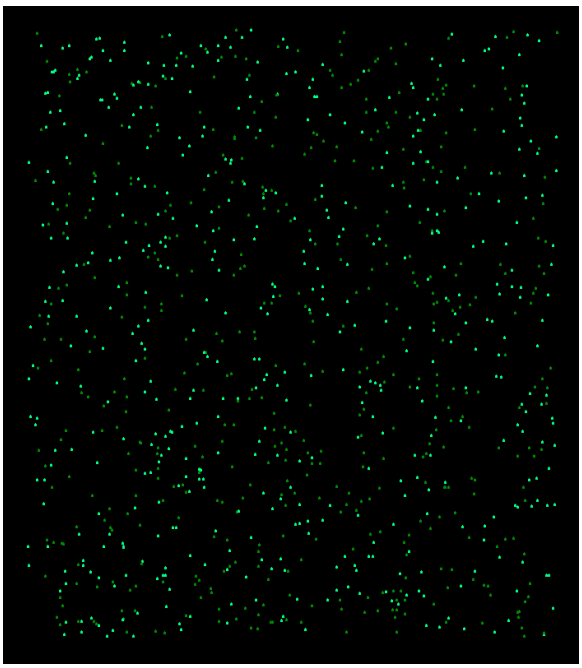
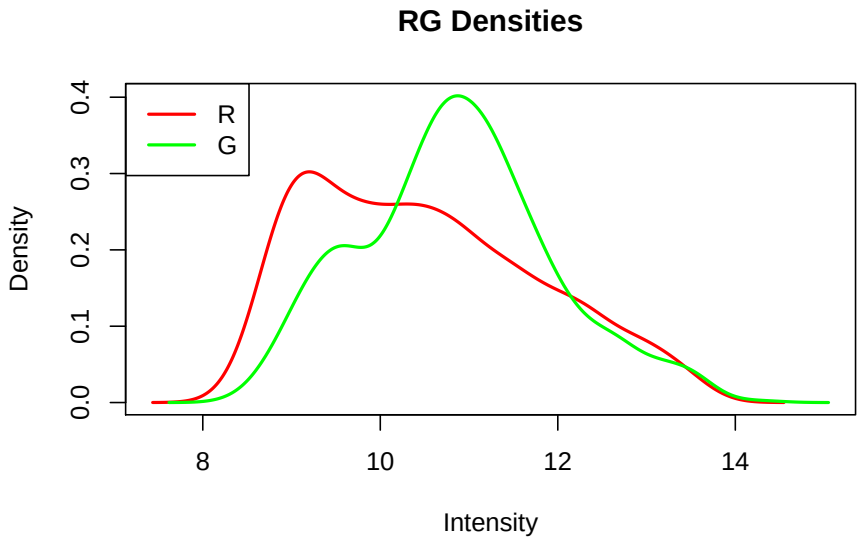


Figure 1: *Spatial distribution of raw reporter intensities laid out by the reporter position on the microarray surface.*



In addition, the data file loaded above also contains a *GFF (General Feature Format)* file of all transcripts on human chromosome 9 annotated in the Ensembl database (release 46, August 2007). The script `retrieveGenomicFeatureAnnotation.R` in the package's scripts directory contains example source code showing how the Bioconductor package *biomaRt* can be used to generate such an annotated genome features *data.frame*.

```
> head(exGFF[,c("name","symbol","chr","strand","start","end")])
```

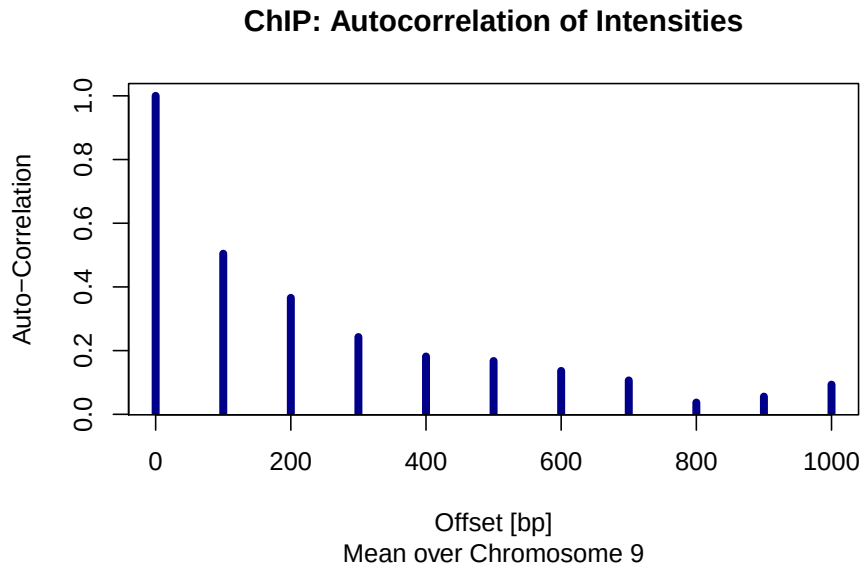
```
            name symbol chr strand  start    end
```

```
1 ENST00000382507           9      1   1056   1620
2 ENST00000326592           9     -1   5841  19587
3 ENST00000382502           9     -1  24394  25841
4 ENST00000355822           9     -1  24921  25277
5 ENST00000382501           9     -1  25028  25856
6 ENST00000382500  FOXD4    9     -1 105991 108951
```

To assess the impact of the small distance between reporters on the data, one can look at the autocorrelation plot. For each base-pair lag d, it is assessed how strong the intensities of reporters at genomic positions $x+d$ are correlated with the probe intensities at positions x.

The computed correlation is plotted against the lag d.

```
> exAc <- autocor(exRG, probeAnno=exProbeAnno, chrom="9", lag.max=1000)
> plot(exAc)
```



We see some auto-correlation between probe position up to 800 base pairs apart. Since the sonicated fragments that are hybridized to the array have an average size in the range of up to 1000 bp, such a degree of auto-correlation up to this distance can be expected.

5 Preprocessing

Following quality assessment of the raw data, we perform normalization of the probe intensities and derive fold changes of reporters' intensities in the enriched sample divided by their intensities in the non-enriched *input* sample and take the (generalized) logarithm of these ratios.

We use the variance-stabilizing normalization (Huber et al., 2002) or probe intensities and generate an `ExpressionSet` object of the normalized probe levels.

```
> exampleX <- preprocess(exRG)
> sampleNames(exampleX) <-
+  with(exRG$targets, paste(Cy5,"vs",Cy3,sep="_"))
> print(exampleX)
```

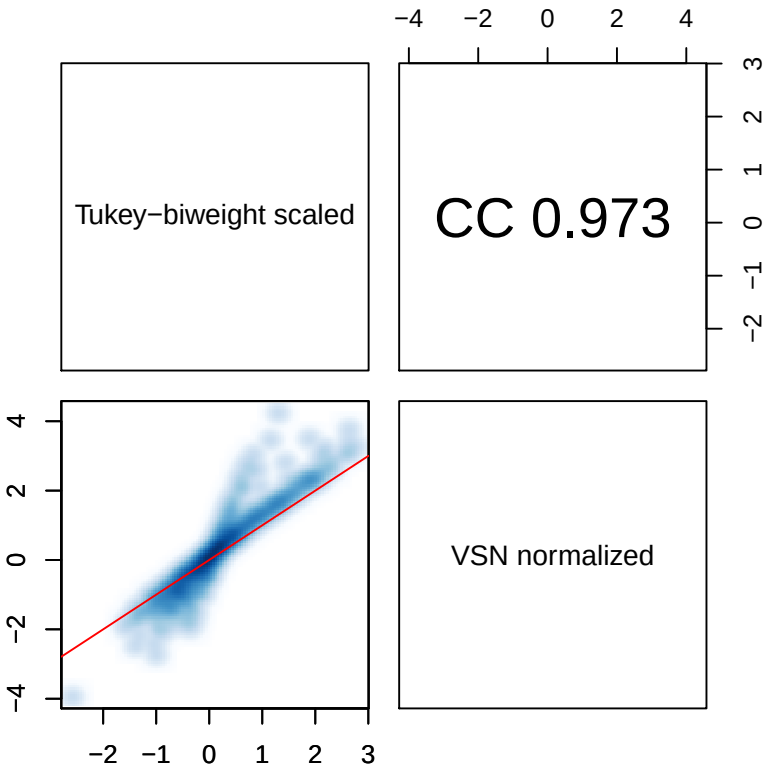
```
ExpressionSet (storageMode: lockedEnvironment)
assayData: 991 features, 1 samples
  element names: exprs
protocolData: none
phenoData
  sampleNames: Suz12_vs_total
  varLabels: SlideNumber FileNameCy3 ... Cy5 (6 total)
  varMetadata: varLabel labelDescription
featureData: none
experimentData: use 'experimentData(object)'
Annotation:
```

Among the provided alternative preprocessing options is also the Tukey-biweight scaling procedure that NimbleGen have used to scale ChIP-chip readouts so that the data is centered on zero.

```
> exampleX.NG <- preprocess(exRG, method="nimblegen")
> sampleNames(exampleX.NG) <- sampleNames(exampleX)
```

The effects of different preprocessing procedures on the data, can be assessed using the `corPlot` function.

```
> corPlot(cbind(exprs(exampleX),exprs(exampleX.NG)),
+         grouping=c("VSN normalized","Tukey-biweight scaled"))
```



The same function can also be used to assess the correlation between biological and technical replicates among the microarray samples.

6 Visualize intensities along the chromosome

The function `chipAlongChrom` provides a way to visualize the ChIP-chip data in a specified genome region. For convenience, this function can also be invoked by using the function `plot` with an *ExpressionSet* object as first argument and a *probeAnno* object as second argument.

```
> plot(exampleX, exProbeAnno, chrom="9", xlim=c(34318000,34321000),
+      ylim=c(-2,4), gff=exGFF, colPal=c("skyblue", "darkblue"))
```

See the result in figure 2.

7 Smoothing of probe intensities

Since the response of reporters to the same amount of hybridized genome material varies greatly, due to probe GC content, melting temperature, secondary structure etc., it is suggested to do a smoothing over individual probe intensities before looking for ChIP-enriched regions.

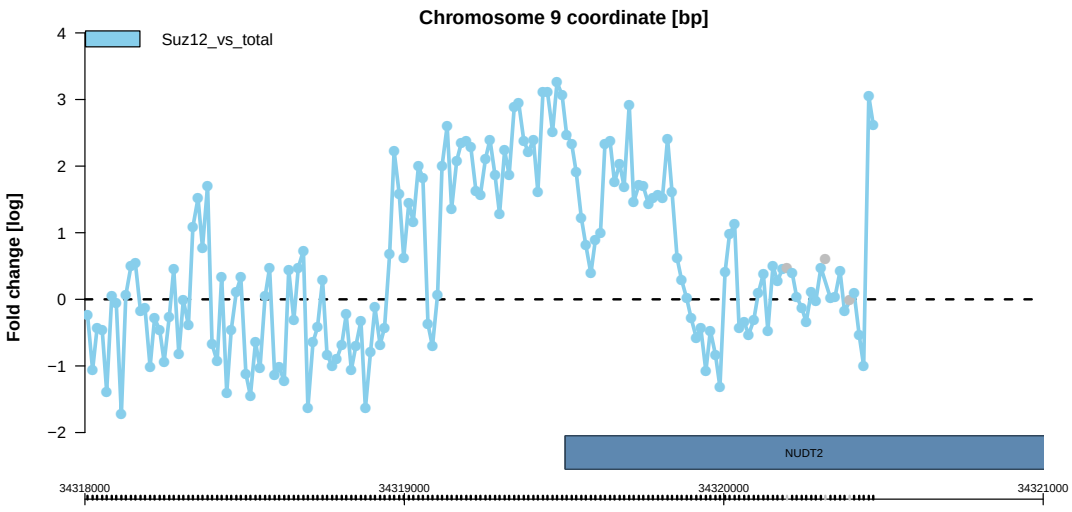


Figure 2: *Normalized probe intensities around the TSS of the* `Nudt2` *gene.*

Here, we slide a window of 800 bp width along the chromosome and replace the intensity at e genomic position x_0 by the median over the intensities of those reporters inside the window that is centered at x_0.

```
> smoothX <- computeRunningMedians(exampleX, probeAnno=exProbeAnno,
+ modColumn = "Cy5", allChr = "9", winHalfSize = 400)
> sampleNames(smoothX) <- paste(sampleNames(exampleX),"smoothed")

> combX <- combine(exampleX, smoothX)
> plot(combX, exProbeAnno, chrom="9", xlim=c(34318000,34321000),
+      ylim=c(-2,4), gff=exGFF, colPal=c("skyblue", "steelblue"))
```

See the smoothed probe levels in figure 3.

8 Finding ChIP-enriched regions

To identify antibody-enriched genomic regions, we require the following:

- smoothed intensities of reporters mapped to this region exceed a certain threshold y_0
- the region contains at least three probe match positions
- each affected position is less than a defined maximum distance d_{max} apart from another affected position in the region (we require a certain probe spacing to have confidence in detected peaks[4])

[4]Note that the term "peak", while commonly used in ChIP-chip context, is slightly misleading and the term "ChIP-enriched region", or "cher" in shorthand, is more appropriate. Within such regions the actual signal could show two or more actual signal peaks or none at all (long plateau).

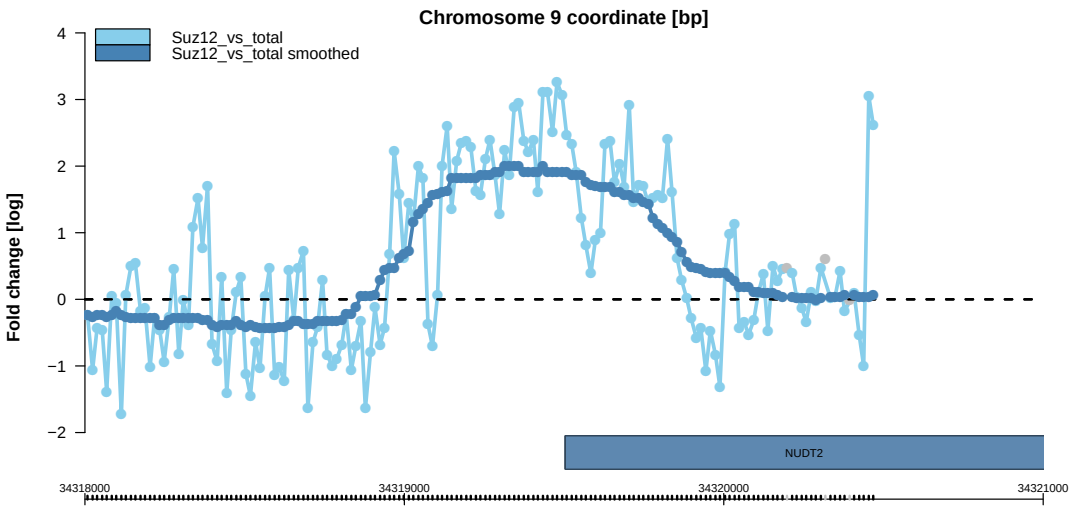


Figure 3: *Normalized and smoothed probe intensities around the TSS of the* `Nudt2` *gene.*

For setting the threshold y_0, one has to assess the expected (smoothed) probe levels in non-enriched genomic regions, i.e. the *null distribution* of probe levels. In a perfect world, we could use a log ratio of 0 as definite cut-off. In this case the "enriched" DNA and the input DNA sample would be present in equal amounts, so no antibody-bound epitope, could be found at this genomic site. In practice, there are some reasons why zero may be a too naive cut-off for calling a probe-hit genomic site *enriched* in our case. See Bourgon (2006) for an extensive discussion on problematic issues with ChIP-chip experiments. We will just briefly mention a few issues here. For once, during the immuno-precipitation, some non-antibody-bound regions may be pulled down in the assay and consequently enriched or some enriched DNA may cross-hybridize to other reporters. Furthermore, since genomic fragments after sonication are mostly a lot larger than the genomic distance between two probe-matched genomic positions, auto-correlation between reporters certainly is existent. Importantly, different reporters measure the same DNA amount with a different efficiency even after normalizing the probe levels, due to sequence properties of the probe, varying quality of the synthesis of reporters on the array and other reasons. To ameliorate this fact, we employ the sliding-window smoothing approach.

The aforementioned issues make it difficult to come up with a reasonable estimate for the null distribution of smoothed probe levels in non-enriched genomic regions. See Figure 4 for the two histograms. We present one way (out of many) for objectively choosing the threshold y_0. The histograms suggest the smoothed reporter levels follow a mixture of two distributions, one being the null distribution of non-affected reporters and the other one the alternative one for the smoothed reporter values in ChIP-enriched regions. We assume the null distribution is symmetric and its mode is the one close to zero in the histogram. By mirroring its part left of the mode over the mode, we end up with an estimated null distribution. For the alternative distribution, we only assume that it is stochastically larger than the null distribution and that its mass to the left of the estimated mode of the null distribution is negligible. We estimate an upper bound y_0 for values arising from the null distribution and conclude that smoothed probe levels $y > y_0$ are more likely to arise from the ChIP enrichment distribution than from the null distribution. These estimates are indicated by red vertical lines in the histograms.

```
> (y0 <- apply(exprs(smoothX),2,upperBoundNull))
```

```
Suz12_vs_total smoothed
             0.7392157
```

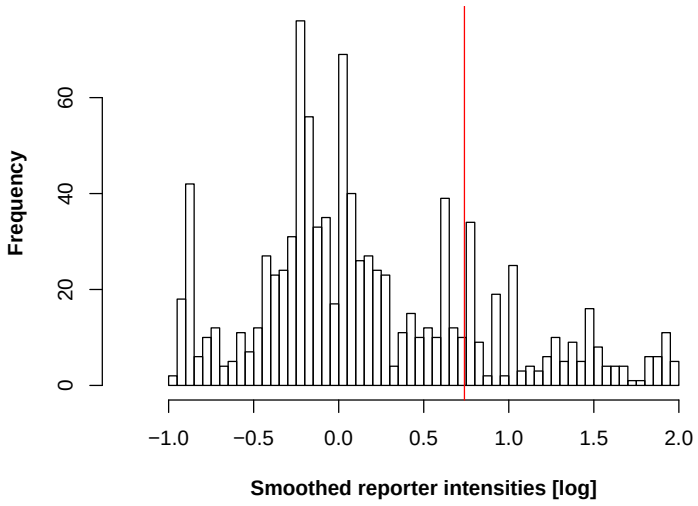


Figure 4: *Histograms of reporter intensities after smoothing of reporter level. The red vertical line is the cutoff values suggested by the histogram.*

Since antibodies vary in their efficiency to bind to their target epitope, we suggest to obtain a different threshold for each antibody. In the example data, however, we have only one antibody against `Suz12`.

While this threshold worked well for us, we do not claim this way to be a gold standard for determining the threshold. In particular, it does not take into account the auto-correlation between near-by reporters. See Bourgon (2006) for a more sophisticated algorithm that does take it into account.

```
> chersX <- findChersOnSmoothed(smoothX, probeAnno=exProbeAnno, thresholds=y0,
+    allChr="9", distCutOff=600, cellType="human")
> chersX <- relateChers(chersX, exGFF)
> chersXD <- as.data.frame.cherList(chersX)
```

```
> chersXD[order(chersXD$maxLevel, decreasing=TRUE),]
```

```
                                      name chr    start      end cellType
1 human.Suz12_vs_total smoothed.chr9.cher1   9 34319028 34319854    human
3 human.Suz12_vs_total smoothed.chr9.cher3   9 34580420 34582384    human
2 human.Suz12_vs_total smoothed.chr9.cher2   9 34579444 34579760    human
                 antibody
1 Suz12_vs_total smoothed
3 Suz12_vs_total smoothed
2 Suz12_vs_total smoothed
                                                                  features
```

```
1 ENST00000379158 ENST00000379154 ENST00000379155 ENST00000346365 ENST00000337747
3                                                  ENST00000378980 ENST00000351266
2                                                  ENST00000378980 ENST00000351266
   maxLevel      score
1 1.9958907 52.1036588
3 1.5341497 47.8557696
2 0.7882005  0.5865187
```

Note that in *Ringo* functions, "ChIP-enriched region" is abbreviated to "cher".

One characteristic of enriched regions that can be used for sorting them is the element `maxLevel`, that is the highest smoothed probe level in the enriched region. Alternatively, one can sort by the `score`, that is the sum of smoothed probe levels minus the threshold. It is a discretized version of to the area under the curve with the baseline being the threshold.

```
> plot(chersX[[1]], smoothX, probeAnno=exProbeAnno, gff=exGFF,
+      paletteName="Spectral")
```

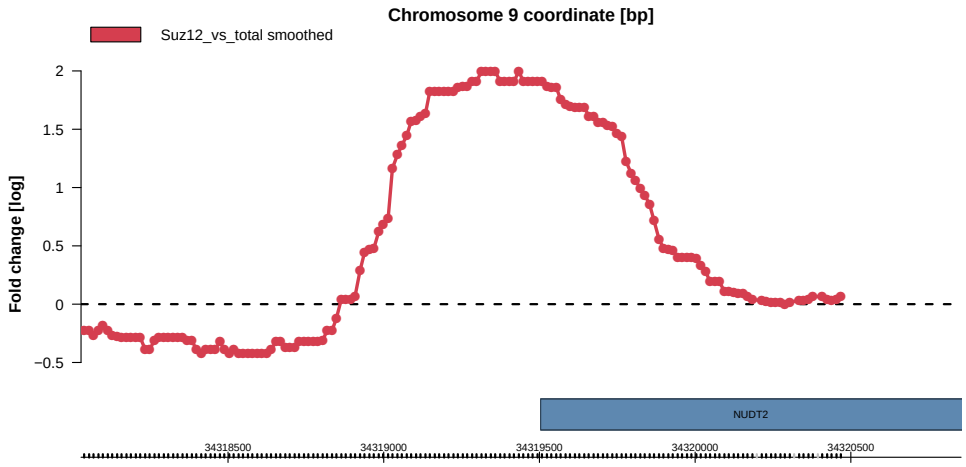


Figure 5: *One of the identified Suz12-antibody enriched regions on chromosome 9.*

Figure 5 displays an identified enriched region, which is located upstream of the `Nudt2` gene. This ChIP-enriched region was already obvious in plots of the normalized data (see Figure 3). While it is reassuring that our method recovers it as well, a number of other approaches would undoubtedly have reported it as well.

9 Agilent data

The package *Ringo* can also be applied to ChIP-chip data from manufacturers other than NimbleGen. As long as the data is supplied as an *RGList* or *ExpressionSet*, the functions of the package can be used, although certain function arguments may need to be changed from their default setting. As an example, we demonstrate how *Ringo* can be used for the analysis of ChIP-chip data generated on two-color microarrays from Agilent. These data

have been described in Schmidt et al. (2008), and the raw data files were downloaded from the ArrayExpress database (Parkinson et al., 2009, accession: E-TABM-485). The data are ChIP-chip measurements of the histone modification H3K4me3 in a 300 kb region of chromosome 17 in mouse Tc1 liver cells. These demo data files included in this package are only excerpts of the original data files.

First we read in the raw data using the function `read.maimages` from package *limma*.

```
> agiDir <- system.file("agilentData", package="Ringo")
> arrayfiles <- list.files(path=agiDir,
+  pattern="H3K4Me3_Tc1Liver_sol1_mmChr17_part.txt")
> RG <- read.maimages(arrayfiles, source="agilent", path=agiDir)
```

Annotation of the one sample was provided and we created a targets file that contains this sample annotation.

```
> at <- readTargets(file.path(agiDir,"targets.txt"))
> RG$targets <- at
```

Have a look at the raw data structure in R.

```
> show(RG)
```

```
An object of class "RGList"
$G
[1] 135.0  52.0  50.0  86.0 134.5
774 more rows ...

$Gb
[1] 48.0 49.0 49.0 49.0 49.5
774 more rows ...

$R
[1] 111.0  67.0  68.0  94.0 253.5
774 more rows ...

$Rb
[1] 65.0 64.0 65.0 65.5 66.0
774 more rows ...

$targets
                                        SlideNumber
H3K4Me3_Tc1Liver_sol1_mmChr17_part H3K4me3Tc1Liver
                                                                FileName
H3K4Me3_Tc1Liver_sol1_mmChr17_part H3K4Me3_Tc1Liver_sol1_mmChr17_part.txt
                                       Species Tissue   Cy3     Cy5 Antibody
H3K4Me3_Tc1Liver_sol1_mmChr17_part Mus musculus  Liver input H3K4me3  H3K4me3

$genes
```

```
  Row Col ProbeUID ControlType         ProbeName                  GeneName
1   1   1        0           1 MmCGHBrightCorner         MmCGHBrightCorner
2   1   2        1           1        DarkCorner                 DarkCorner
3   1   3        1           1        DarkCorner                 DarkCorner
4   1   4        3           0    A_68_P31052153 chr17:012323564-012323623
5   1  95      185           0    A_68_P31152765 chr17:033822110-033822154
             SystematicName
1         MmCGHBrightCorner
2                DarkCorner
3                DarkCorner
4 chr17:012323564-012323623
5 chr17:033822110-033822154
774 more rows ...

$source
[1] "agilent"
```

We can only perform limited quality assessment of these data, as the data only consist of one sample and the demo data files are only a short excerpt of the full raw data from that microarray. Have a look at the spatial distribution of the raw intensities on the microarray surface.

```
> par(mar=c(0.01,0.01,0.01,0.01), bg="black")
> image(RG, 1, channel="red", dim1="Col", dim2="Row",
+       mycols=c("sienna","darkred","orangered"))
```

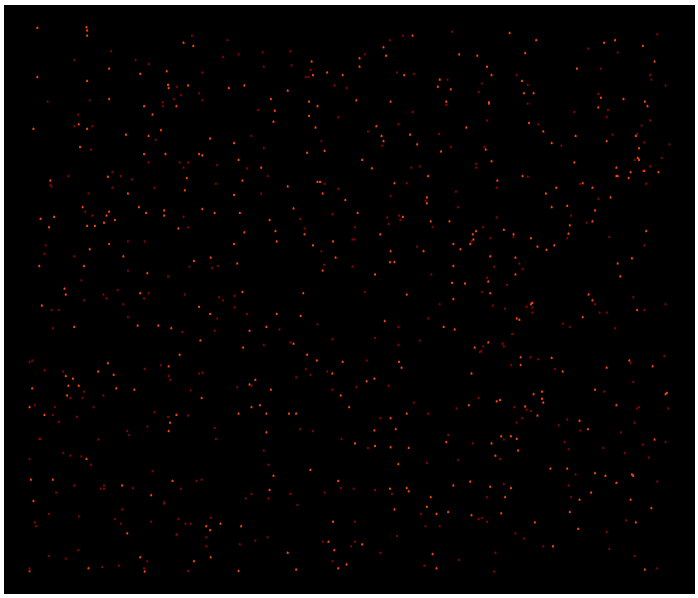


Figure 6: *Spatial distribution of raw reporter intensities of the Agilent microarray, laid out by the reporter position on the microarray surface.*

See the result in Figure 6. Again, the sparseness of the image is due to the fact that the example data is only a short excerpt of the original raw data file from the microarray.

To create a *probeAnno* object for this array, there are two possibilites:

1. one option – in many cases the preferable one – is to remap all the probe sequences from the array description file to the current genome build
2. often, as in this case, the systematic name Agilent gives to reporters on the array corresponds to the genomic coordinates these reporters were designed to match. Therefore, if one decides to accept the mapping provided by the manufacturer, these systematic names can be used to generate a *probeAnno* object.

Here, we show the second option and use the systematic names of the reporters, as provided in the raw data file.

```
> pA <- extractProbeAnno(RG, "agilent", genome="mouse",
+     microarray="Agilent Tiling Chr17")
```

Genome annotation The provided Agilent example data relate to chromosome 17 of the Mouse genome. We have used the package *biomaRt* to retrieve annotated genes on this chromosome from the Ensembl database[5].

```
> load(file=file.path(agiDir,"mm9chr17.RData"))
```

Data processing We preprocess the data in the same way as the previous example data set.

```
> X <- preprocess(RG[RG$genes$ControlType==0,], method="nimblegen",
+                 idColumn="ProbeName")
> sampleNames(X) <- X$SlideNumber
```

We visualize the data in the region around the start site of the gene *Rab11b* on chromosome 17. For setting the parameters of the sliding-window smoothing, we first investigate the spacing between adjacent reporter matches on the genome.

```
> probeDists <- diff(pA["17.start"])
> br <- c(0, 100, 200, 300, 500, 1000, 10000, max(probeDists))
> table(cut(probeDists, br))
```

```
        (0,100]        (100,200]        (200,300]        (300,500]
              0              338              221              104
  (500,1e+03]   (1e+03,1e+04] (1e+04,2.13e+07]
             71               39                2
```

The majority of match positions are 100–300 bp apart.

```
> smoothX <- computeRunningMedians(X, modColumn="Antibody",
+               winHalfSize=500, min.probes=3, probeAnno=pA)
> sampleNames(smoothX) <- paste(sampleNames(X),"smooth",sep=".")
```

[5]See the script `retrieveGenomicFeatureAnnotation.R` in the package's `scripts` directory for details.

We visualize the data after smoothing in the region around the start site of the gene *Rab11b* on chromosome 17.

```
> combX <- combine(X, smoothX)
> plot(combX, pA, chr="17", coord=33887000+c(0, 13000),
+      gff=mm9chr17, maxInterDistance=450, paletteName="Paired")
```

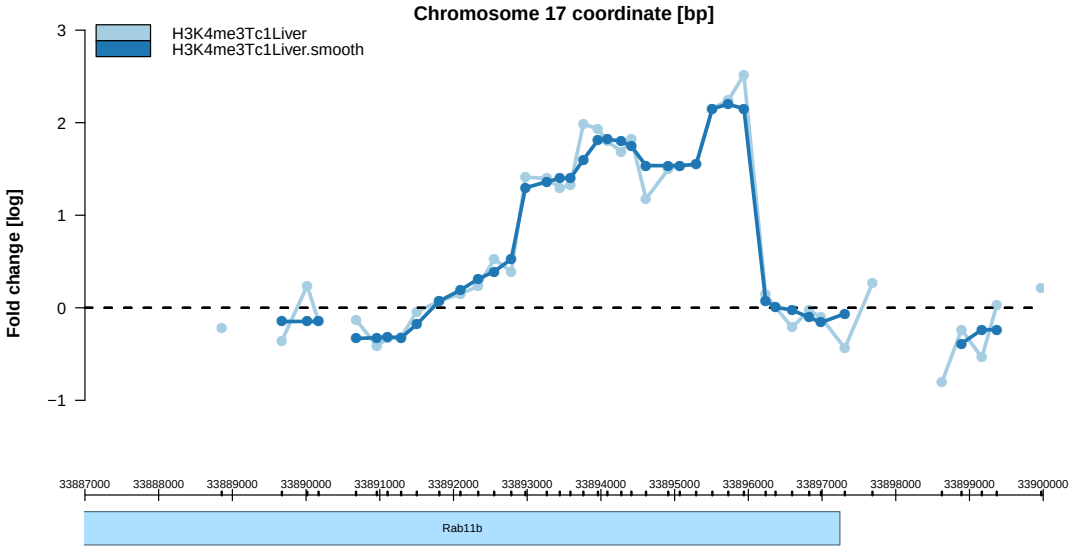


Figure 7: *Normalized and smoothed Agilent reporter intensities around the TSS of the gene Rap11b.*

See the result in Figure 7.

Find ChIP-enriched regions We compare two approaches to determine the threshold y_0 above which smoothed reporter levels are considered to indicate enrichment. The first one is the non-parametric approach that we introduced before (see Section 8). The second one is similar. Only both the null distribution and the alternative distribution are assumed to be Gaussians. The threshold is minimal reporter levels with a sufficiently small p-value under the Gaussian null distribution.

```
> y0  <- upperBoundNull(exprs(smoothX))
> y0G <- twoGaussiansNull(exprs(smoothX))
```

```
> hist(exprs(smoothX), n=100, main=NA,
+      xlab="Smoothed expression level [log2]")
> abline(v=y0, col="red", lwd=2)
> abline(v=y0G, col="blue", lwd=2)
> legend(x="topright", lwd=2, col=c("red","blue"),
+        legend=c("Non-parametric symmetric Null", "Gaussian Null"))
```

Even though the non-parametric estimate is usually preferable, with few data points such as in this case, the estimate derived from a Gaussian null distribution might provide a better threshold for enrichment. We are going to use this threshold y_{0G} for the identification of ChIP-enriched regions.

```
> chersX <- findChersOnSmoothed(smoothX, probeAnno=pA, threshold=y0G,
+                               cellType="Tc1Liver")
> chersX <- relateChers(chersX, gff=mm9chr17, upstream=5000)
```

We find 14 ChIP-enriched regions.

```
> chersXD <- as.data.frame(chersX)
> head(chersXD[order(chersXD$maxLevel, decreasing=TRUE),])
```

```
                                          name chr    start      end cellType
6   Tc1Liver.H3K4me3Tc1Liver.smooth.chr17.cher6  17 33725511 33734380 Tc1Liver
4   Tc1Liver.H3K4me3Tc1Liver.smooth.chr17.cher4  17 33640444 33642033 Tc1Liver
5   Tc1Liver.H3K4me3Tc1Liver.smooth.chr17.cher5  17 33642962 33644147 Tc1Liver
2   Tc1Liver.H3K4me3Tc1Liver.smooth.chr17.cher2  17 33634150 33635943 Tc1Liver
10 Tc1Liver.H3K4me3Tc1Liver.smooth.chr17.cher10  17 33795820 33798894 Tc1Liver
12 Tc1Liver.H3K4me3Tc1Liver.smooth.chr17.cher12  17 33814429 33816303 Tc1Liver
                 antibody           features maxLevel     score
6  H3K4me3Tc1Liver.smooth ENSMUSG00000024300 5.504729 121.60989
4  H3K4me3Tc1Liver.smooth                    5.081668  23.04169
5  H3K4me3Tc1Liver.smooth                    5.002873  15.12560
2  H3K4me3Tc1Liver.smooth                    4.889622  31.76257
10 H3K4me3Tc1Liver.smooth ENSMUSG00000059208 4.868514  43.09071
12 H3K4me3Tc1Liver.smooth ENSMUSG00000059208 4.498009  17.69241
```

This concludes the example of how *Ringo* can be used to analyze other types of ChIP-chip data.

10 Concluding Remarks

The package *Ringo* aims to facilitate the analysis ChIP-chip readouts. We constructed it during the analysis of a ChIP-chip experiment for the genome-wide identification of modified histone sites on data gained from NimbleGen two-color microarrays. Analogous two-color microarray platforms, however, can also be processed. Key functionalities of *Ringo* are data read-in, quality assessment, preprocessing of the raw data, and visualization of the raw and preprocessed data. The package also contains algorithms for the detection of for ChIP-enriched genomic regions. While one of these algorithm worked quite well with our data, we do not claim it to be the definite algorithm for that task.

Further reading For an extended tutorial about how to use *Ringo* for the analysis of ChIP-chip data, please refer to Toedling and Huber (2008) and the corresponding Bioconductor package *ccTutorial*.

Package versions This vignette was generated using the following package versions:

- R version 3.3.0 RC (2016-04-25 r70549), `x86_64-w64-mingw32`

- Base packages: base, datasets, grDevices, graphics, grid, methods, parallel, stats, utils
- Other packages: Biobase 2.32.0, BiocGenerics 0.18.0, Matrix 1.2-6, RColorBrewer 1.1-2, Ringo 1.36.0, lattice 0.20-33, limma 3.28.0, mclust 5.2
- Loaded via a namespace (and not attached): AnnotationDbi 1.34.0, BiocInstaller 1.22.0, DBI 0.4, IRanges 2.6.0, KernSmooth 2.23-15, RSQLite 1.0.0, Rcpp 0.12.4.5, S4Vectors 0.10.0, XML 3.98-1.4, affy 1.50.0, affyio 1.42.0, annotate 1.50.0, colorspace 1.2-6, genefilter 1.54.0, ggplot2 2.1.0, gtable 0.2.0, munsell 0.4.3, plyr 1.8.3, preprocessCore 1.34.0, scales 0.4.0, splines 3.3.0, stats4 3.3.0, survival 2.39-2, tools 3.3.0, vsn 3.40.0, xtable 1.8-2, zlibbioc 1.18.0

Acknowledgments

Many thanks to Wolfgang Huber, Oleg Sklyar, Tammo Krüger, Richard Bourgon, and Matt Ritchie for source code contributions to and lots of helpful suggestions on Ringo, Todd Richmond and NimbleGen Systems, Inc. for providing us with the example ChIP-chip data. This work was supported by the European Union (FP6 HeartRepair, LSHM-CT-2005-018630).

References

R. W. Bourgon. *Chromatin-immunoprecipitation and high-density tiling microarrays: a generative model, methods for analysis, and methodology assessment in the absence of a "gold standard"*. PhD thesis, University of California, Berkley, Berkley, California, USA, 2006. URL http://www.ebi.ac.uk/~bourgon/papers/bourgon_dissertation_public.pdf.

R. C. Gentleman, V. J. Carey, D. J. Bates, B. M. Bolstad, M. Dettling, S. Dudoit, B. Ellis, L. Gautier, Y. Ge, J. Gentry, K. Hornik, T. Hothorn, W. Huber, S. Iacus, R. Irizarry, F. Leisch, C. Li, M. Maechler, A. J. Rossini, G. Sawitzki, C. Smith, G. K. Smyth, L. Tierney, Y. H. Yang, and J. Zhang. Bioconductor: Open software development for computational biology and bioinformatics. *Genome Biology*, 5:R80, 2004.

W. Huber, A. von Heydebreck, H. Sültmann, A. Poustka, and M. Vingron. Variance stabilization applied to microarray data calibration and to the quantification of differential expression. *Bioinformatics*, 18:S96–S104, 2002.

H. Pages, R. Gentleman, P. Aboyoun, and S. DebRoy. *Biostrings: String objects representing biological sequences, and matching algorithms*, 2008. R package version 2.9.29.

H. Parkinson, M. Kapushesky, N. Kolesnikov, G. Rustici, M. Shojatalab, N. Abeygunawardena, H. Berube, M. Dylag, I. Emam, A. Farne, E. Holloway, M. Lukk, J. Malone, R. Mani, E. Pilicheva, T. F. Rayner, F. Rezwan, A. Sharma, E. Williams, X. Z. Bradley, T. Adamusiak, M. Brandizi, T. Burdett, R. Coulson, M. Krestyaninova, P. Kurnosov, E. Maguire, S. G. Neogi, P. Rocca-Serra, S.-A. Sansone, N. Sklyar, M. Zhao, U. Sarkans, and A. Brazma. ArrayExpress update–from an archive of functional genomics experiments to the atlas of gene expression. *Nucleic Acids Research*, 37(Database issue):D868–D872, Jan 2009. URL http://www.ebi.ac.uk/arrayexpress.

D. Schmidt, R. Stark, M. D. Wilson, G. D. Brown, and D. T. Odom. Genome-scale validation of deep-sequencing libraries. *PLoS ONE*, 3(11):e3713, 2008.

G. S. C. Slater and E. Birney. Automated generation of heuristics for biological sequence comparison. *BMC Bioinformatics*, 6:31, 2005.

G. K. Smyth. Limma: linear models for microarray data. In R. Gentleman, V. Carey, W. Huber, R. Irizarry, and S. Dudoit, editors, *Bioinformatics and Computational Biology Solutions Using R and Bioconductor*, pages 397–420. Springer, 2005.

S. L. Squazzo, H. O'Geen, V. M. Komashko, S. R. Krig, V. X. Jin, S. wook Jang, R. Margueron, D. Reinberg, R. Green, and P. J. Farnham. Suz12 binds to silenced regions of the genome in a cell-type-specific manner. *Genome Res*, 16(7):890–900, 2006.

J. Toedling and W. Huber. Analyzing ChIP-chip Data Using Bioconductor. *PLoS Computational Biology*, 4(11):e1000227, Nov 2008.

J. Toedling, O. Sklyar, T. Krueger, J. J. Fischer, S. Sperling, and W. Huber. Ringo - an R/Bioconductor package for analyzing ChIP-chip readouts. *BMC Bioinformatics*, 8:221, 2007.