Ringo - R Investigation of NimbleGen Oligoarrays

Joern Toedling

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, 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 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).

---

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

```
> 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 probe categories on the array (you might know these Spot Types files from *limma* (Smyth, 2005)).

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

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

---

[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.

Reading all these files, we can read in the raw probe 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 probes 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.

3   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.

```
> image(exRG, 1, channel = "green", mycols = c("black", "green4",
+       "springgreen"))
```
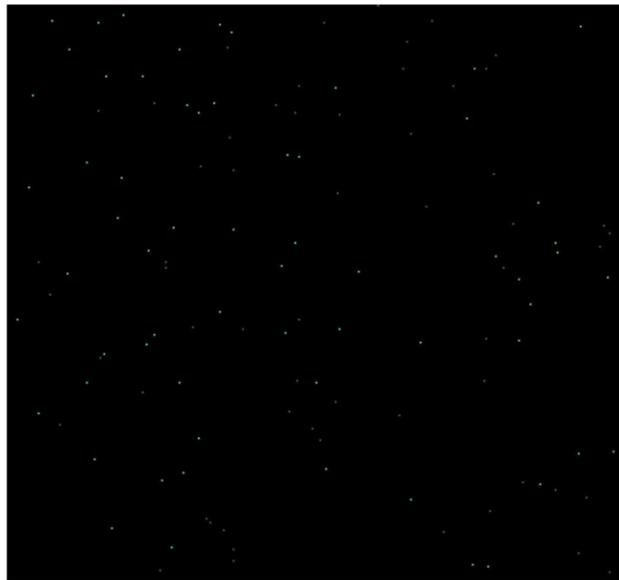
MOD_20551_PMT1_pair

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

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 probes' 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)
```

RG densities
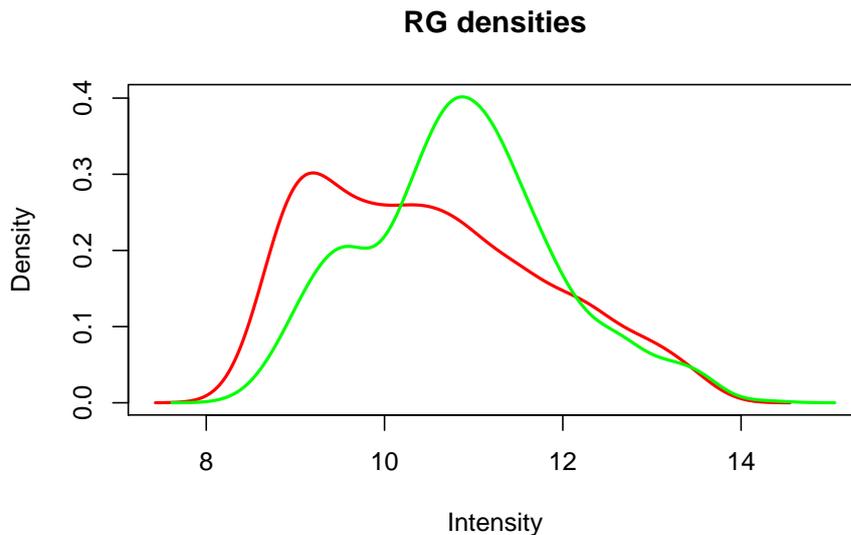
Ringo, similar to the Bioconductor package *tilingArray*, employs a *probeAnno* environment to describe the mapping between reporters on the microarray and genomic positions. Per chromosome, four vectors of equal length and ordering specify at which genomic positions reporter matches start and end, what identifiers or indices these reporters have in the intensities data, and whether these reporters match uniquely to those genomic positions.

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

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

```
> head(get("9.start", exProbeAnno))
```

```
[1] 34315504 34315519 34315534 34315549 34315564 34315579
```

```
> head(get("9.index", exProbeAnno))
```

```
[1] "SUZ100P0000057421" "SUZ100P0000057423" "SUZ100P0000057425"
[4] "SUZ100P0000057427" "SUZ100P0000057429" "SUZ100P0000057431"
```

The package's `scripts` directory contains a script `makeProbeAnno.R` that demonstrates how to generate such a mapping object either from a NimbleGen `POS` file or from result files of aligning the probe sequences to the genome.

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 aforementioned script `makeProbeAnno.R` also contains some example source code showing how the Bioconductor package *biomaRt* can be used to generate such a GFF 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 probes on the data, one can look at the autocorrelation plot. For each base-pair lag d, it is assessed how strong the intensities of probes 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)
```

ChIP: Autocorrelation of Intensities

Offset [bp]
Mean over Chromosome 9
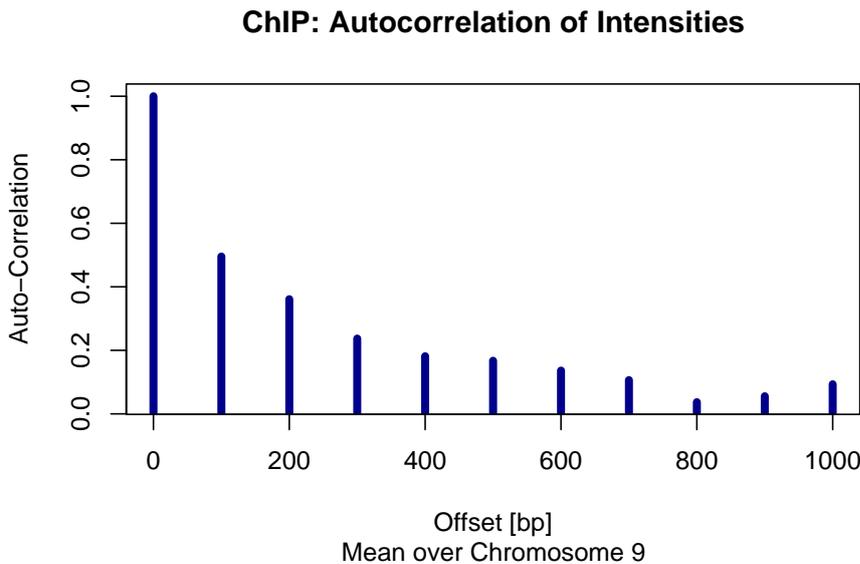
We see some auto-correlation between probe position up to 800 base pairs apart. Since the sonicated fragments that are hybridized to the array have an average size in the range of up to 1000 bp, such a degree of auto-correlation up to this distance can be expected, but nevertheless has to be taken into account later on during data analysis.

4   Preprocessing

Following quality assessment of the raw data, we perform normalization of the probe intensities and derive fold changes of probes' intensities in the enriched sample divided by their intensities in the non-enriched *input* sample and take the (generalized) logarithm of this 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) <- paste(exRG$targets$Cell.line, exRG$targets$Cy5,
+     exRG$targets$Set, sep = ".")
> print(exampleX)


ExpressionSet (storageMode: lockedEnvironment)
assayData: 991 features, 1 samples
  element names: exprs
phenoData
  sampleNames: Suz12_vs_total
  varLabels and varMetadata description:
    SlideNumber: NA
    FileNameCy3: NA
    ...: ...
    Cy5: NA
    (6 total)
  additional varMetadata: labelDescription
featureData
  featureNames: SUZ100P0000021781, SUZ100P0000021783, ..., SUZ100P0000058079  (991 total)
  fvarLabels and fvarMetadata description: 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")


Normalizing...


> sampleNames(exampleX.NG) <- sampleNames(exampleX)
```
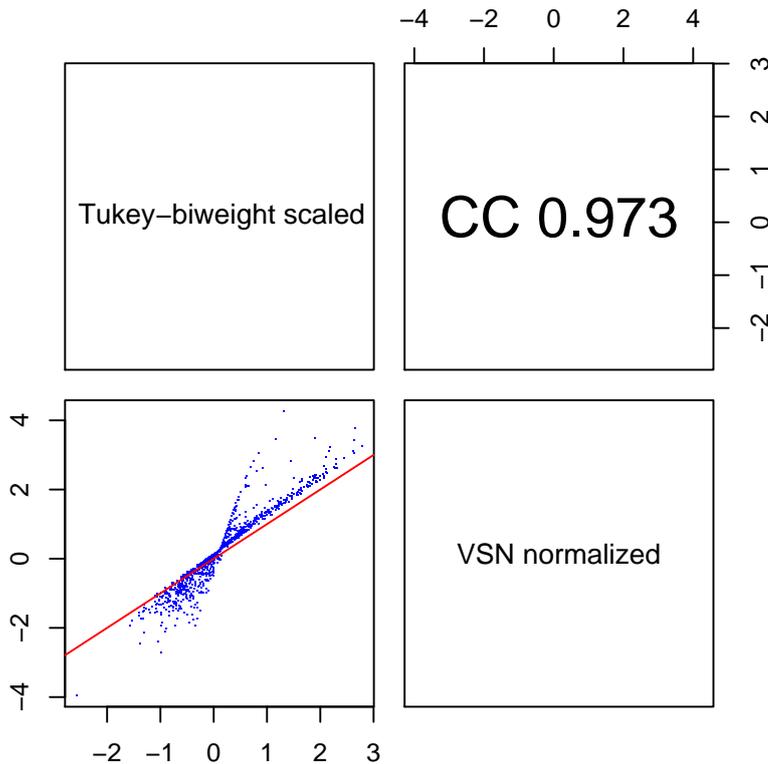
The effects of different preprocessing procedures on the data, can be assessed using the `corPlot` function.

```
> 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.

5   Visualize intensities along the chromosome

```
> chipAlongChrom(exampleX, chrom = "9", xlim = c(34318000, 34321000),
+     ylim = c(-2, 4), probeAnno = exProbeAnno, gff = exGFF)
```

See the result in figure 2.

6   Smoothing of probe intensities

Since the response of probes 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.

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 probes inside the window that is centered at x_0.
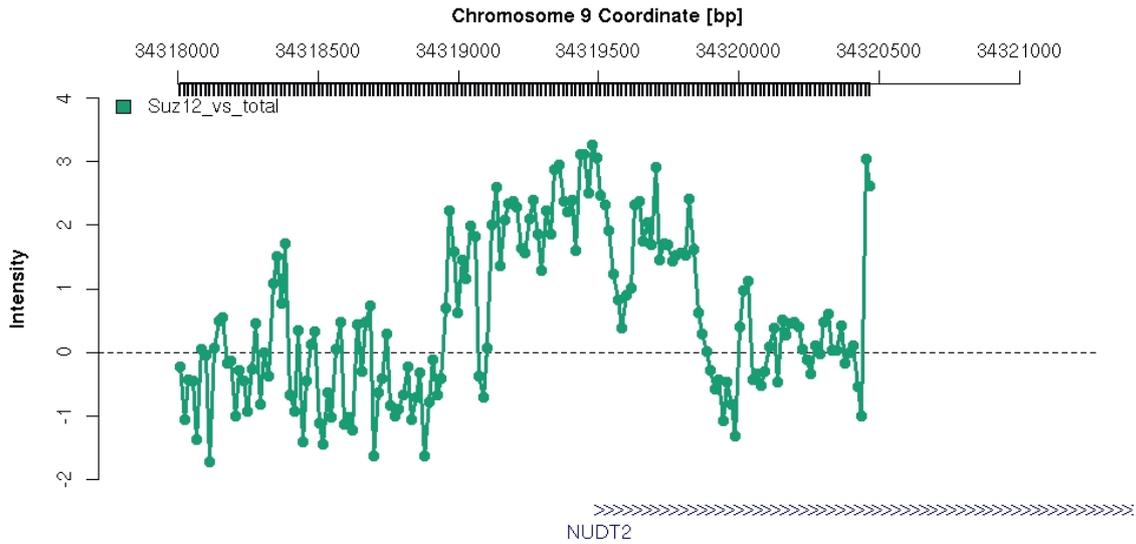
Figure 2: *Normalized probe intensities around the TSS of the* `Nudt2` *gene.*

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


> chipAlongChrom(exampleX, chrom = "9", xlim = c(34318000, 34321000),
+     ylim = c(-2, 4), probeAnno = exProbeAnno, gff = exGFF)
> chipAlongChrom(smoothX, chrom = "9", xlim = c(34318000, 34321000),
+     probeAnno = exProbeAnno, itype = "l", ilwd = 4, paletteName = "Spectral",
+     add = TRUE)
```
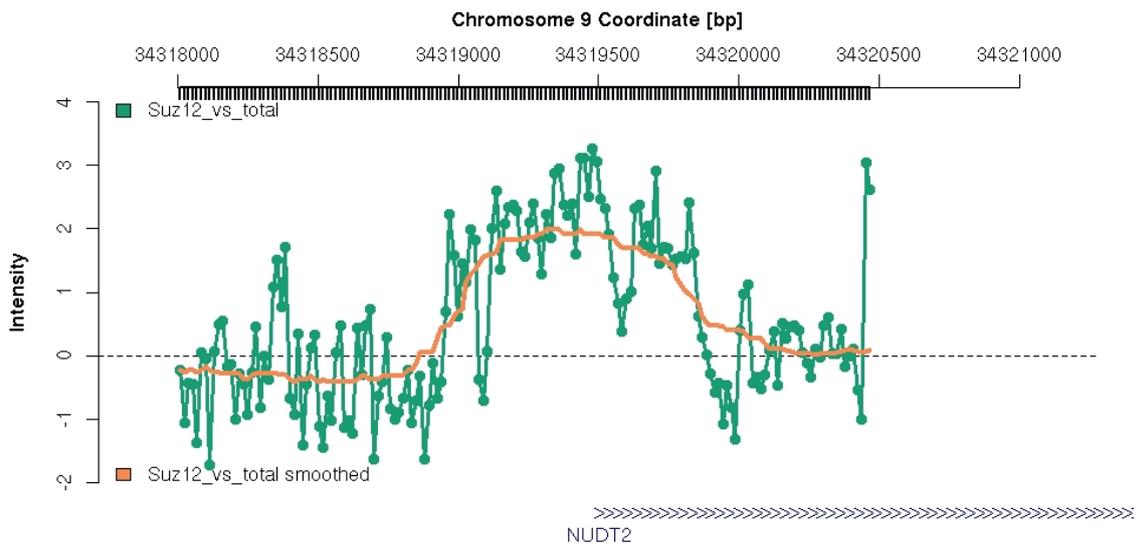

Figure 3: *Normalized and smoothed probe intensities around the TSS of the* `Nudt2` *gene.*

See the smoothed probe levels in figure 3.

7 Peak finding

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

- smoothed intensities of probes mapped to this region are 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[3])

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 probes. Furthermore, since genomic fragments after sonication are mostly a lot larger than the genomic distance between two probe-matched genomic positions, auto-correlation between probes certainly is existent. Importantly, different probes 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 probes on the array and other reasons. To ameliorate this fact, we employ the sliding-window smoothing approach.

Please note that this procedure was developed to identify genomic regions enriched for histone modifications rather than transcription factor binding sites. The frequency and extent of genomic regions with modified histones is expected to be much larger than, say, regions containing binding sites for a certain transcription factor. In that case one cannot assume that the large majority of probes does not show enrichment. In the example data case, however, we look for enriched transcription-factor-bound regions. For this setup a number of alternative, and likely superior, algorithms for detecting enriched regions have been suggested, some being available in other Bioconductor packages, too.

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. We simulate such a distribution by permuting the probe match positions on the chromosome. We take the 99% quantile of these smoothed intensities as the threshold y_0 [4].

---

[3]Note that the term "peak", while commonly used in ChIP-chip context, is slightly misleading and the term "ChIP-enriched region" would be more appropriate. Within such regions the actual signal could show two or more actual signal peaks or none at all (long plateau). For succinctness sake, we use the term "peak" for a ChIP-enriched region, though.

[4]To get a more realistic estimate for such a null distribution, one should shuffle all chromosomes' positions and repeat the whole process many times.

```
> permProbeAnno <- copyEnv(exProbeAnno)
> sampled.index <- sample(get("9.index", env = exProbeAnno))
> assign("9.index", sampled.index, env = permProbeAnno)
> sampledX <- computeRunningMedians(exampleX, probeAnno = permProbeAnno,
+     modColumn = "Cy5", allChr = c("9"), winHalfSize = 400, verbose = FALSE)
> (y0 <- quantile(exprs(sampledX), 0.99, na.rm = TRUE))


      99%
0.4956420
```

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 probes.

```
> peaksX <- findPeaksOnSmoothed(smoothX, probeAnno = exProbeAnno,
+     thresholds = y0, allChr = "9", distCutOff = 600, cellType = "human")
> peaksX <- relatePeaks(peaksX, exGFF)
> peaksXD <- as.data.frame.peakList(peaksX)


> peaksXD


                                     name chr    start      end cellType
1 human.Suz12_vs_total smoothed.chr9.peak1   9 34318984 34319884    human
2 human.Suz12_vs_total smoothed.chr9.peak2   9 34579054 34582400    human
              antibody
1 Suz12_vs_total smoothed
2 Suz12_vs_total smoothed
                                                              transcripts
1 ENST00000379158 ENST00000379154 ENST00000379155 ENST00000346365 ENST00000337747
2                                 ENST00000378980 ENST00000351266
   maxPeak    score
1 1.995891 66.26737
2 1.534150 94.12441


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

Figure 4 displays the identified peak, which is the one upstream of the Nudt2 gene. This peak was obvious in plots of the normalized data (see figure 3). And while it is reassuring that our method recovers it as well, a number of other approaches would certainly have reported it as well.
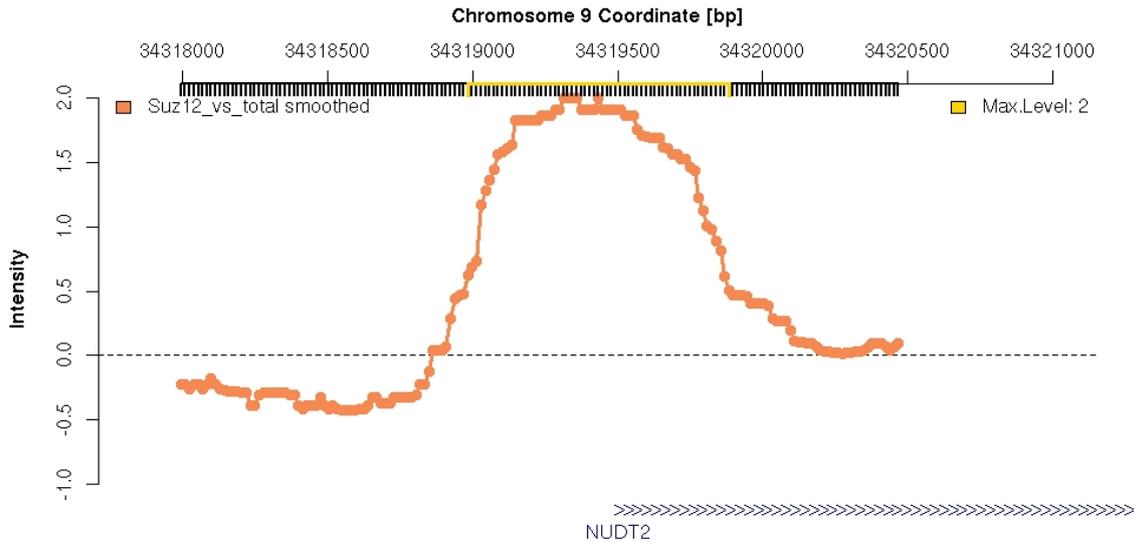
Figure 4: *One of the identified Suz12-antibody enriched regions on chromosome 9.*

8 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 contains a heuristic algorithm for the detection of for ChIP-enriched genomic regions, too. While this algorithm worked quite well on our data, we do not claim it to be the definite algorithm for that task.

This vignette was generated using the following package versions:

- R version 2.6.0 (2007-10-03), `x86_64-unknown-linux-gnu`

- Locale: `LC_CTYPE=en_US;LC_NUMERIC=C;LC_TIME=en_US;LC_COLLATE=en_US;LC_MONETARY=en_US;LC_`

- Base packages: base, datasets, graphics, grDevices, methods, stats, tools, utils

- Other packages: affy 1.16.0, affyio 1.6.0, annotate 1.16.0, AnnotationDbi 1.0.0, Biobase 1.16.0, DBI 0.2-3, geneplotter 1.16.0, lattice 0.16-5, limma 2.12.0, preprocessCore 1.0.0, RColorBrewer 1.0-1, Ringo 1.2.0, RSQLite 0.6-3, vsn 3.2.1, xtable 1.5-1

- Loaded via a namespace (and not attached): grid 2.6.0, KernSmooth 2.22-21

Acknowledgments

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.

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, O. Sklyar, and W. Huber. Ringo - an R/Bioconductor package for analyzing ChIP-chip readouts. *BMC Bioinformatics*, 8:221, 2007.