

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 NimbleGen¹, 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 (?) and provides additional ChIP-chip-specific and NimbleGen-specific functionalities.

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

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.

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

```
[1] "55773_532_pair.txt" "55773_635_pair.txt" "56577_532_pair.txt"
[4] "56577_635_pair.txt"
```

```
> head(read.delim(file.path(exDir, "56577_532_pair.txt"), skip = 1))[,
+       c(1, 4:8, 10)]
```

	IMAGE_ID	PROBE_ID	POSITION	X	Y	MATCH_INDEX	PM
1	56577_532	CHR11P020230890	20230890	672	600	39853667	36933.89
2	56577_532	CHR11P020231216	20231216	541	603	39867116	30607.78
3	56577_532	CHR11P020231584	20231584	592	812	40048471	23702.22

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

```

4 56577_532 CHR2P028894632 28894632 388 48 39974238 10601.33
5 56577_532 CHR4P056640054 56640054 78 838 39726901 8223.00
6 56577_532 CHR4P090182945 90182945 366 280 39939651 14373.11

```

In addition, there is a file with more details on the samples, including which files belong to which sample.

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

	SlideNumber	FileNameCy3	FileNameCy5	Set	Cell.line	Cy3	Cy5
1	56577	56577_532_pair.txt	56577_635_pair.txt	1	HL1	input	H3ac
2	55773	55773_532_pair.txt	55773_635_pair.txt	2	HL1	input	H3ac

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 our experiment, so the column `Cy5` essentially holds which antibody has been used for the immuno-precipitation, in this case one against H3ac that is acetylated Histone 3 residues.

Furthermore, there is a file describing the probe categories on the array (you might know these Spot Types files from *limma* (?)).

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

	SpotType	GENE_EXPR_OPTION	PROBE_ID	Color
1	Probe	BLOCK1	*	black
2	Negative	NGS_CONTROLS	*	yellow
3	H_Code	H_CODE	*	red
4	V_Code	V_CODE	*	blue
5	Random	RANDOM	*	green

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

```
> exRG <- readNimblegen("example_files.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)
```

	56577_635_pair	55773_635_pair
[1,]	18994.89	30161.33
[2,]	17016.11	18996.22
[3,]	17485.67	7521.11
[4,]	7915.89	9846.78
[5,]	3795.33	8704.11
[6,]	8212.33	10070.22

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

	56577_532_pair	55773_532_pair
[1,]	36933.89	46984.78
[2,]	30607.78	35818.00
[3,]	23702.22	21651.67
[4,]	10601.33	35601.11
[5,]	8223.00	24904.11
[6,]	14373.11	37386.67

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

	GENE_EXPR_OPTION	PROBE_ID	POSITION	X	Y	Status	ID
1	BLOCK1	CHR11P020230890	20230890	672	600	Probe	CHR11P020230890
2	BLOCK1	CHR11P020231216	20231216	541	603	Probe	CHR11P020231216
3	BLOCK1	CHR11P020231584	20231584	592	812	Probe	CHR11P020231584
4	BLOCK1	CHR2P028894632	28894632	388	48	Probe	CHR2P028894632
5	BLOCK1	CHR4P056640054	56640054	78	838	Probe	CHR4P056640054
6	BLOCK1	CHR4P090182945	90182945	366	280	Probe	CHR4P090182945

```
> exRG$targets
```

	SlideNumber	FileNameCy3	FileNameCy5	Set	Cell.line	Cy3	Cy5
1	56577	56577_532_pair.txt	56577_635_pair.txt	1	HL1 input	H3ac	
2	55773	55773_532_pair.txt	55773_635_pair.txt	2	HL1 input	H3ac	

The user can alternatively supply raw two-color microarray readouts from other platforms in *RGList* format, and the other functions of *Ringo* can work equally well on 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"))
```

See figure 1 for the image. Since the provided example data set only holds the probe intensities of probes that could be mapped to chromosome 8, the image only shows the few green dots of these probes' positions. We see, however, that these chromosome 8 probes are well distributed over the whole array surface rather than being bundled 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.

56577_532_pair

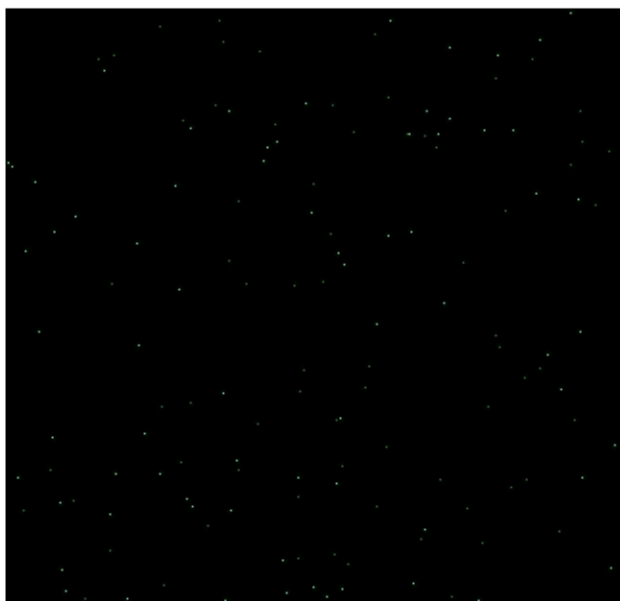
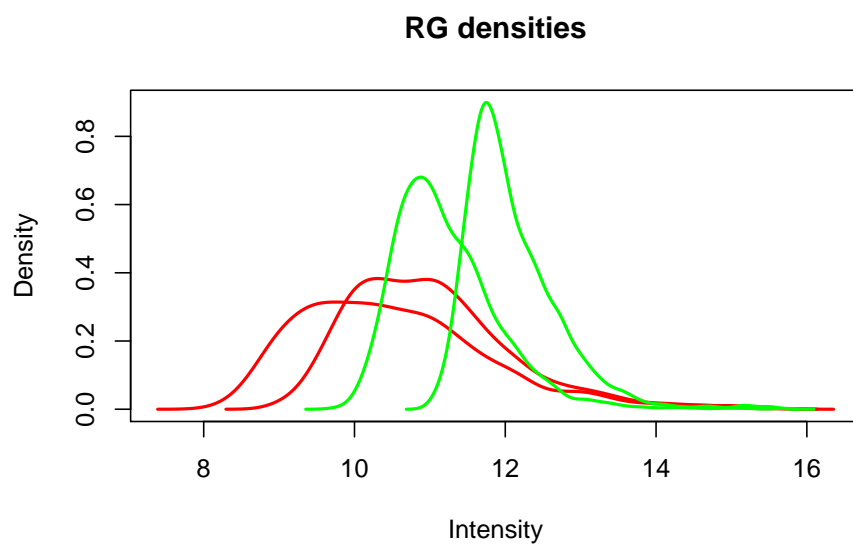


Figure 1: *Spatial distribution of raw probe intensities laid out by the probe position on the microarray surface.*

```
> plotDensities(exRG)
```



A *probeAnno* environment contains the mapping between probes and genomic positions.

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

```
[1] "8.end"      "8.index"    "8.start"    "8.unique"
```

```
> head(get("8.start", exProbeAnno))
```

```
[1] 60209176 60209261 60209346 60209431 60209516 60209601
```

```
> head(get("8.index", exProbeAnno))
```

```
[1] "CHR8P056101891" "CHR8P056101976" "CHR8P056102061" "CHR8P056102146"
[5] "CHR8P056102231" "CHR8P056102316"
```

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 mouse chromosome 8 annotated in the [Ensembl](#) database (version 39, June 2006).

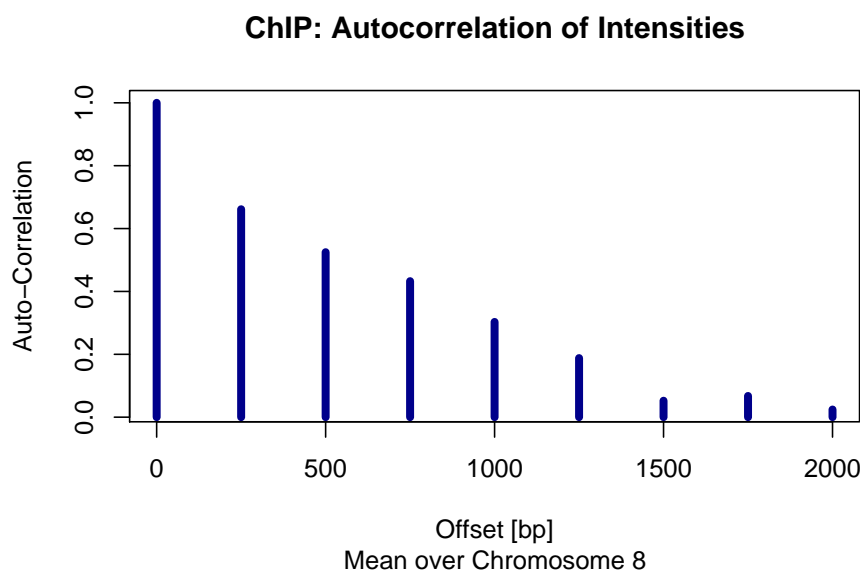
```
> head(exGFF[, c(2:6, 14, 17)])
```

	name	chr	strand	start	end	symbol	refseq
26696	ENSMUST000000040104	8	1	60213125	60216659	Hand2	NM_010402
26697	ENSMUST000000034023	8	1	60348065	60369727	Scrg1	NM_009136
26698	ENSMUST000000034022	8	-1	60374849	60380004		NM_021788
26699	ENSMUST000000067925	8	1	60404049	60406636	Hmgb2	NM_008252
26700	ENSMUST000000054134	8	-1	60412298	60413030		XR_002441
26701	ENSMUST000000034021	8	-1	60416711	60545145	Galnt7	NM_144731

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 = "8", lag.step = 250)
> plot(exAc)
```



We see a high autocorrelation between probes up to 750 base-pairs apart. Since the sonicated fragments that are hybridized to the array have an average size of 900 bp, such a high auto-correlation up to this distance is to 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 (?) 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)
```

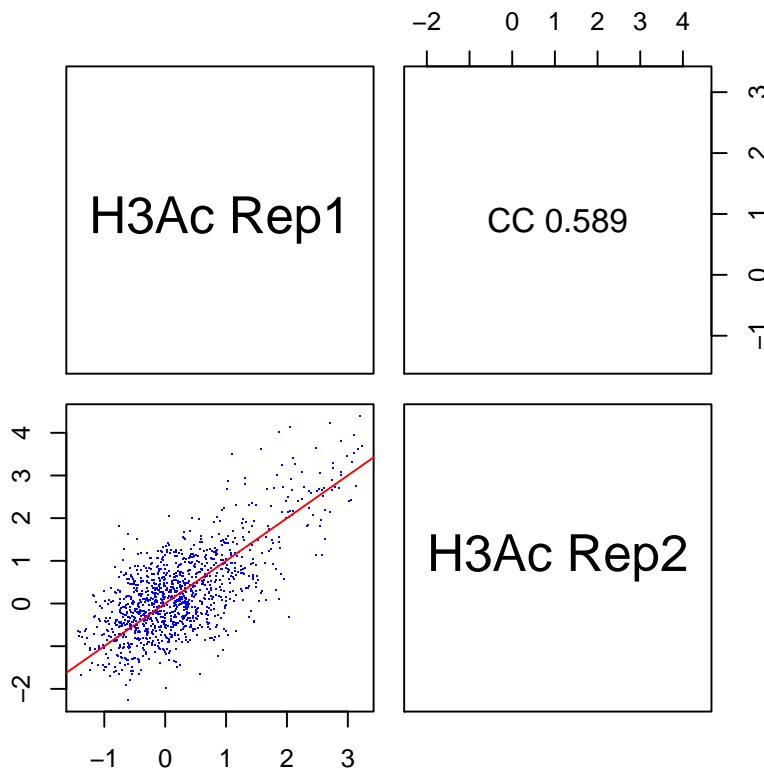
```
ExpressionSet (storageMode: lockedEnvironment)
assayData: 1201 features, 2 samples
  element names: exprs
phenoData
  sampleNames: HL1.H3ac.1, HL1.H3ac.2
  varLabels and varMetadata:
    SlideNumber: NULL
    FileNameCy3: NULL
    ...: ...
    Cy5: NULL
    (7 total)
```

```
featureData
  featureNames: 1, 2, ..., 1201 (1201 total)
  varLabels and varMetadata: none
experimentData: use 'experimentData(object)'
Annotation [1] ""
```

5 Correlation between replicates

After preprocessing, we assess the degree of correlation between our two samples, which in fact are biological replicates of the same experiment.

```
> corPlot(exampleX, grouping = c("H3Ac Rep1", "H3Ac Rep2"))
```



Remember that these are true biological replicates rather than technical replicates, and thus we do not expect perfect correlation.

6 Visualize intensities along the chromosome

```
> load(file.path(exDir, "exampleX.rda"))
> chipAlongChrom(exampleX, chrom = "8", xlim = c(60208500, 60216000),
+   ylim = c(-1, 6), colPal = 2:3, probeAnno = exProbeAnno, gff = exGFF)
```

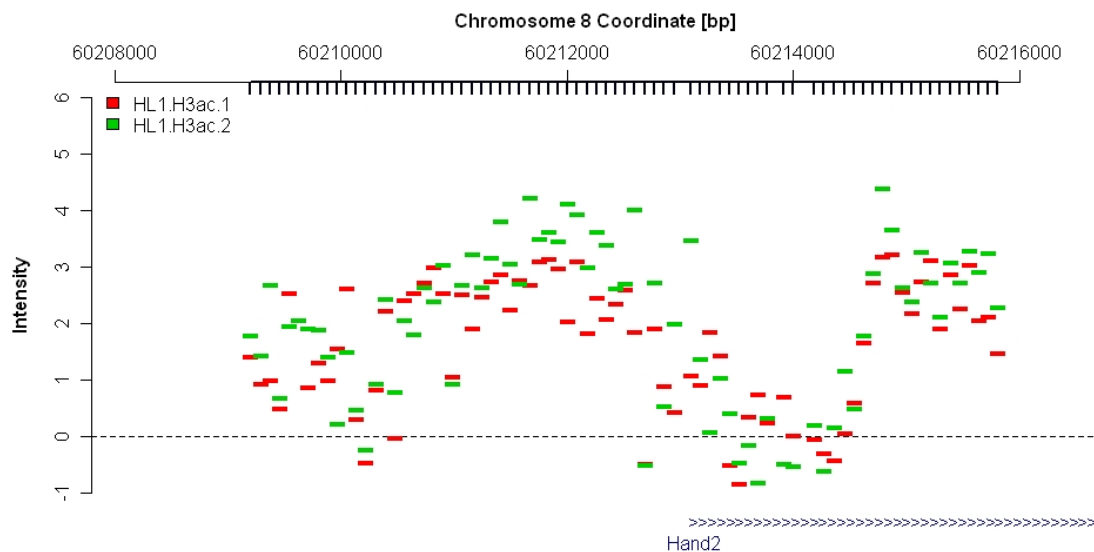


Figure 2: *Normalized probe intensities around the TSS of the **Hand2** gene.*

See the result in figure 2.

7 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 peaks.

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

```
> smoothX <- computeRunningMedians(exampleX, probeAnno = exProbeAnno,
+   modColumn = "Cy5", allChr = c("8"), winHalfSize = 400, combineReplicates = TRUE,
+   verbose = TRUE)

> chipAlongChrom(exampleX, chrom = "8", xlim = c(60208500, 60216000),
+   ylim = c(-1, 6), colPal = 2:3, probeAnno = exProbeAnno, gff = exGFF)
> chipAlongChrom(smoothX, chrom = "8", xlim = c(60208000, 60216000),
+   probeAnno = exProbeAnno, itype = "l", ilwd = 3, paletteName = "Spectral",
+   add = TRUE)
```

See the smoothed probe levels in figure 3.

8 Peak finding

To identify genomic regions, in which the histones are modified, we require the following:

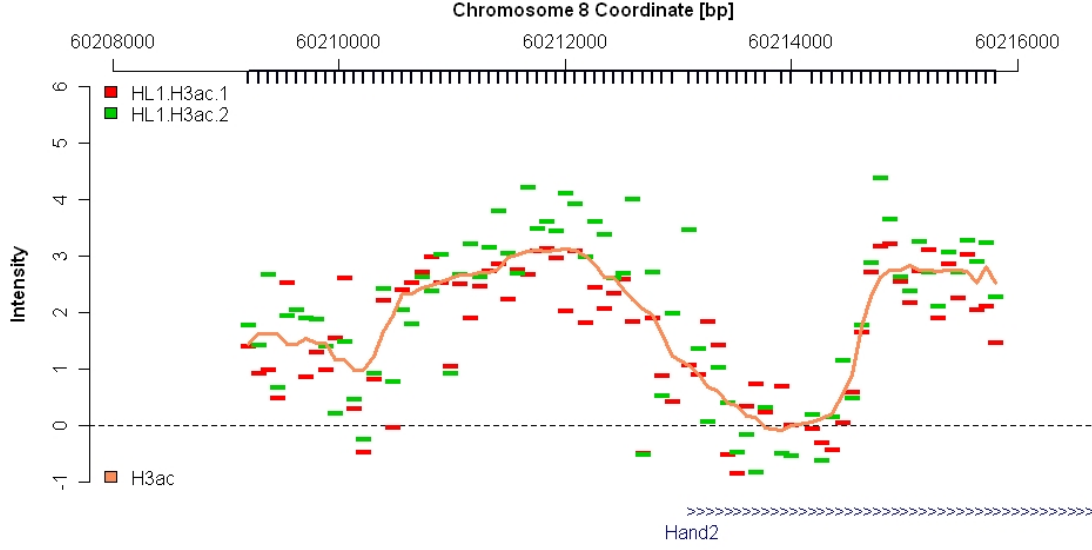


Figure 3: *Smoothed probe intensities around the TSS of the *Hand2* gene.*

- 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)

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, i.e. a modified histone residue in our case, 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 ? 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.

Most importantly, in the current setting we are looking for regions enriched for histone modifications. 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. Thus, we cannot assume that the large majority of probes does not show enrichment.

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 ².

```
> permProbeAnno <- copyEnv(exProbeAnno)
> sampled.index <- sample(get("8.index", env = exProbeAnno))
> assign("8.index", sampled.index, env = permProbeAnno)
> sampledX <- computeRunningMedians(exampleX, probeAnno = permProbeAnno,
+   modColumn = "Cy5", allChr = c("8"), winHalfSize = 400, verbose = FALSE)
> (y0 <- quantile(exprs(sampledX), 0.99, na.rm = TRUE))
```

```
99%
1.035892
```

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 histone 3 acetylation.

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 = c("8"), distCutOff = 600, cellType = "HL1")
> peaksX <- relatePeaks(peaksX, exGFF)
> peaksXD <- as.data.frame.peakList(peaksX)
```

```
> peaksXD[2, ]
```

	name	chr	start	end	cellType	modification	nUpstream
2	HL1.H3ac.chr8.peak2	8	60210306	60213094	HL1	H3ac	1
	nDownstream		transcripts	maxPeak	score		
2	0	ENSMUST00000040104	3.132845	47.2402			

```
> plot(peaksX[[2]], smoothX, probeAnno = exProbeAnno, gff = exGFF)
```

Figure 4 displays the identified peak, which is the one upstream of the HAND2. This peak was obvious in plots of the normalized data (see figure 3) and it is reassuring that the algorithm recovers it as well.

²To get a more realistic estimate for such a null distribution, one should shuffle all chromosomes' positions and repeat the whole process many times.

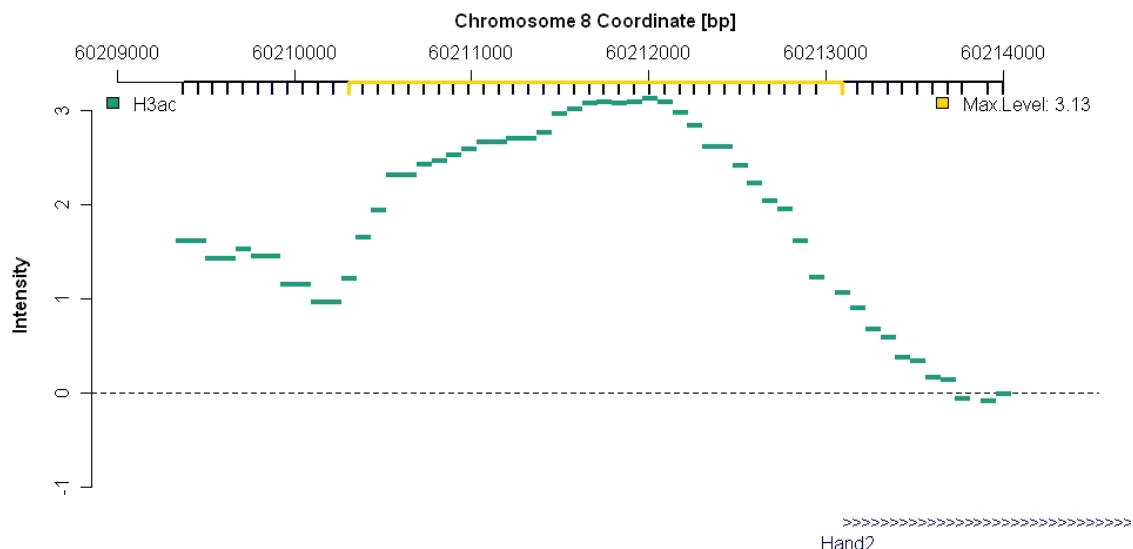


Figure 4: *One of the discovered peaks for H3ac antibody enrichment on chromosome 8.*

9 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 2.5.0 (2007-04-23), i386-pc-mingw32
- Locale: LC_COLLATE=English_United States.1252;LC_CTYPE=English_United States.1252;LC_MONETARY=English_United States.1252;LC_NUMERIC=English_United States.1252;LC_TIME=English_United States.1252
- Base packages: base, datasets, graphics, grDevices, methods, stats, tools, utils
- Other packages: affy 1.14.0, affyio 1.4.0, annotate 1.14.1, Biobase 1.14.0, geneplotter 1.14.0, lattice 0.15-4, limma 2.10.0, RColorBrewer 0.2-3, Ringo 1.2.0, vsn 2.2.0

References