Repitools: A collection of utilities for understanding epigenetic data.

Mark Robinson Aaron Statham Dario Strbenac

1 Introduction

Repitools is an R package to assist with the analysis of datasets typically found in epigenetics research. The main focus of the package is creating summarys of promoter tiling arrays and simple analyses of next generation sequencing, in the context of gene expression data. Functionality for Affymetrix promoter arrays and Solexa sequencing data is presented. However, the functions are simple to use on Nimblegen data too, usually with the only difference being an extra parameter being given. Consult the help page of each function for how to use it on Nimblegen data. Furthermore, there are many options for each function beyond those presented in this user manual.

2 Getting the Repitools package

The package can be acquired by running the command:

```
install.packages("Repitools", repos = "http://r-forge.r-project.org")
```

To get the data to use with these examples, download the RepitoolsExamples package from the R-Forge website:

http://repitools.r-forge.r-project.org/

and install the package using install.packages(). It can then be loaded by calling the function setupExamples(), after the package has been loaded into R (see below).

Repitools also depends on a number of packages, depending on the functions used. These include:

- aroma.affymetrix
- edgeR
- limma
- BSgenome, BSgenome.Hsapiens.UCSC.hg18
- IRanges
- ShortRead
- chipseq

Download and install these from their respective websites (e.g. Bioconductor, aroma.affymetrix) before using Repitools.

3 Getting help for the Repitools package

First, users should consult the documentation available within R. For example, to access the documentation for the cpgDensityCalc, type ?cpgDensityCalc after loading the package. Alternatively, you may call help.start() and browse the help documents (for all installed packages) through a web browser.

For questions that are not discussed in the function documentation, Repitools has a searchable mailing list, which can be accessed from:
https://lists.r-forge.r-project.org/pipermail/repitools-help/

Users can sign up for the mailing list through:
https://lists.r-forge.r-project.org/cgi-bin/mailman/listinfo/repitools-help

4 Setting up the aroma.affymetrix environment

For the analysis of Affymetrix tiling arrays, Repitools makes heavy use of the aroma.affymetrix package, which requires data and their annotations to be in a specific directory structure (and must be writable by the end-user). The RepitoolsExamples package has the included example data files laid out correctly, and can be setup by executing:

```
setupExamples()
```

```
Successfully setwd to example data directory
```

Or, if you wish to copy the example data into the current working directory (for instance if the package has been installed system-wide into a non-writable location), execute instead:

```
setupExamples(doCopy = TRUE)
```

For further information regarding aroma.affymetrix, consult the documentation at http://aroma-project.org/ and specifically the vignette 'MAT: (Promoter 1.0R) Tiling array analysis'.

5 Reading the example data

The data used in these examples are largely from our recent paper[1] where we integrated expression and epigenetic data of normal prostate epithelial cells (PrEC) and the LNCaP prostate cancer cell line. The example data includes Affymetrix Human Promoter 1.0R tiling arrays from a histone H3K9 acetylation chromatin immunoprecipitation experiment, with matching Affymetrix Human Gene 1.0ST expression arrays, and Illumina sequencing libraries.

First, references to the chip definition file (CDF) and CEL files (containing the data) are defined. Second, a design matrix specifying the difference ('contrast') of interest is created. Generally, the set of tiling array CEL files is MAT normalised and smoothed[2], here using the aroma.affymetrix implementation. Some operations take place directly on the normalized data while some analyses may make use of the smoothed values.

[1] Coolen et al. (2010) Consolidation of the cancer genome into domains of repressive chromatin by long-range epigenetic silencing (LRES) reduces transcriptional plasticity. *Nature Cell Biology*

[2] Johnson et al. (2006) Model-based analysis of tiling-arrays for ChIP-chip. *Proc. Natl. Acad. Sci. USA* 103: 12457-12462

```
capture <- function(...) UseMethod("capture")
getNames <- function(...) UseMethod("getNames")
cdfT <- AffymetrixCdfFile$byChipType("Hs_PromPR_v02")
csT <- AffymetrixCelSet$byName("Tiling", cdf = cdfT)
csTU <- convertToUnique(csT)
MN <- MatNormalization(csT, numChunks = 15)
csTMN <- process(MN)
csTMNU <- convertToUnique(csTMN)
design <- cbind(`PrEC K9Ac` = c(0, 0, -1, 1), `LNCAP K9Ac` = c(-1,
    1, 0, 0), `LNCaP-PrEC K9Ac` = c(-1, 1, 1, -1))
rownames(design) <- getNames(csT)
print(design)
```

```
                 PrEC K9Ac LNCAP K9Ac LNCaP-PrEC K9Ac
LNCaP_K9Ac_Input1         0         -1              -1
LNCaP_K9Ac_IP1            0          1               1
PrEC_K9Ac_Input1         -1          0               1
PrEC_K9Ac_IP1             1          0              -1
```

```
MS <- MatSmoothing(csTMNU, design = design, probeWindow = 300,
    tag = "300bp_smoothing", nProbes = 10)
csTS <- process(MS, units = NULL)
```

Next, we perform a standard Robust Multixchip Analyis (RMA)[3] on the example expression data.

```
cdfE <- AffymetrixCdfFile$byChipType("HuGene-1_0-st-v1")
csE <- AffymetrixCelSet$byName("geneExpression", cdf = cdfE)
bc <- RmaBackgroundCorrection(csE)
csEBC <- process(bc)
qn <- QuantileNormalization(csEBC, typesToUpdate = "pm")
csEN <- process(qn)
plm <- RmaPlm(csEN)
fit(plm)
ces <- getChipEffectSet(plm)
em <- log2(extractMatrix(ces))
rownames(em) <- getUnitNames(cdfE)
```

Next, we read in the positions of human genes with identifiers matched up to the Gene 1.0 ST platform (this was adapted from the annotation provided by Affymetrix):

```
genes <- read.csv("annotationData/humanGenomeAnnotation.csv")
em <- em[match(genes$name, rownames(em)), ]
head(genes)
```

```
     name  chr strand  start    end    symbol selected_identifier
1 7896759 chr1      + 781253 783614 LOC643837            AK096570
2 7896761 chr1      + 850983 869824    SAMD11           NM_152486
3 7896779 chr1      + 885829 890958    KLHL17           NM_198317
4 7896798 chr1      + 891739 900345   PLEKHN1           NM_032129
5 7896817 chr1      + 938709 939782     ISG15           NM_005101
6 7896822 chr1      + 945365 981355      AGRN           NM_198576
```

[3] Irizarry et al. (2003) Summaries of Affymetrix GeneChip probe level data *Nucleic Acids Research* 31(4):e15

We form a design matrix to specify the average expression levels, or differential expression between LNCaP and PrEC cells:

```
designE <- cbind(PrEC = c(0, 0, 0.5, 0.5), LNCaP = c(0.5, 0.5,
    0, 0), `LNCaP-PrEC` = c(0.5, 0.5, -0.5, -0.5))
rownames(designE) <- getNames(csE)
print(designE)
```

```
       PrEC LNCaP LNCaP-PrEC
LNCaP1  0.0   0.5        0.5
LNCaP2  0.0   0.5        0.5
PrEC1   0.5   0.0       -0.5
PrEC2   0.5   0.0       -0.5
```

We apply the design matrix to each gene (to give a gene-level score for each contrast):

```
emE <- em %*% designE
```

Finally, we load sequencing data for a different experiment. It has been stored compactly in a `BSgenome::GenomeDataList` object only containing the chromosome, strand and start position of each read (details on how to import data into this format are in the Section "Utility Functions"). This data is unpublished so we describe it here as just "IP" for each cell line, but we make it available in order to illustrate the examples. Further information will be made available when published, but the details are not important for illustrating the Repitools package.

```
load("rawData/sequencing/seq_data.Rdata")
print(rs)
```

```
GenomeDataList of length 4
names(4): PrEC_IP1 PrEC_IP2 LNCaP_IP1 LNCaP_IP2
```

```
designSeq <- cbind(`PrEC IP` = c(1, 1, 0, 0), `LNCaP IP` = c(0,
    0, 1, 1), `LNCaP-PrEC IP` = c(-1, -1, 1, 1))
rownames(designSeq) <- names(rs)
print(designSeq)
```

```
          PrEC IP LNCaP IP LNCaP-PrEC IP
PrEC_IP1        1        0            -1
PrEC_IP2        1        0            -1
LNCaP_IP1       0        1             1
LNCaP_IP2       0        1             1
```

6 Data summaries

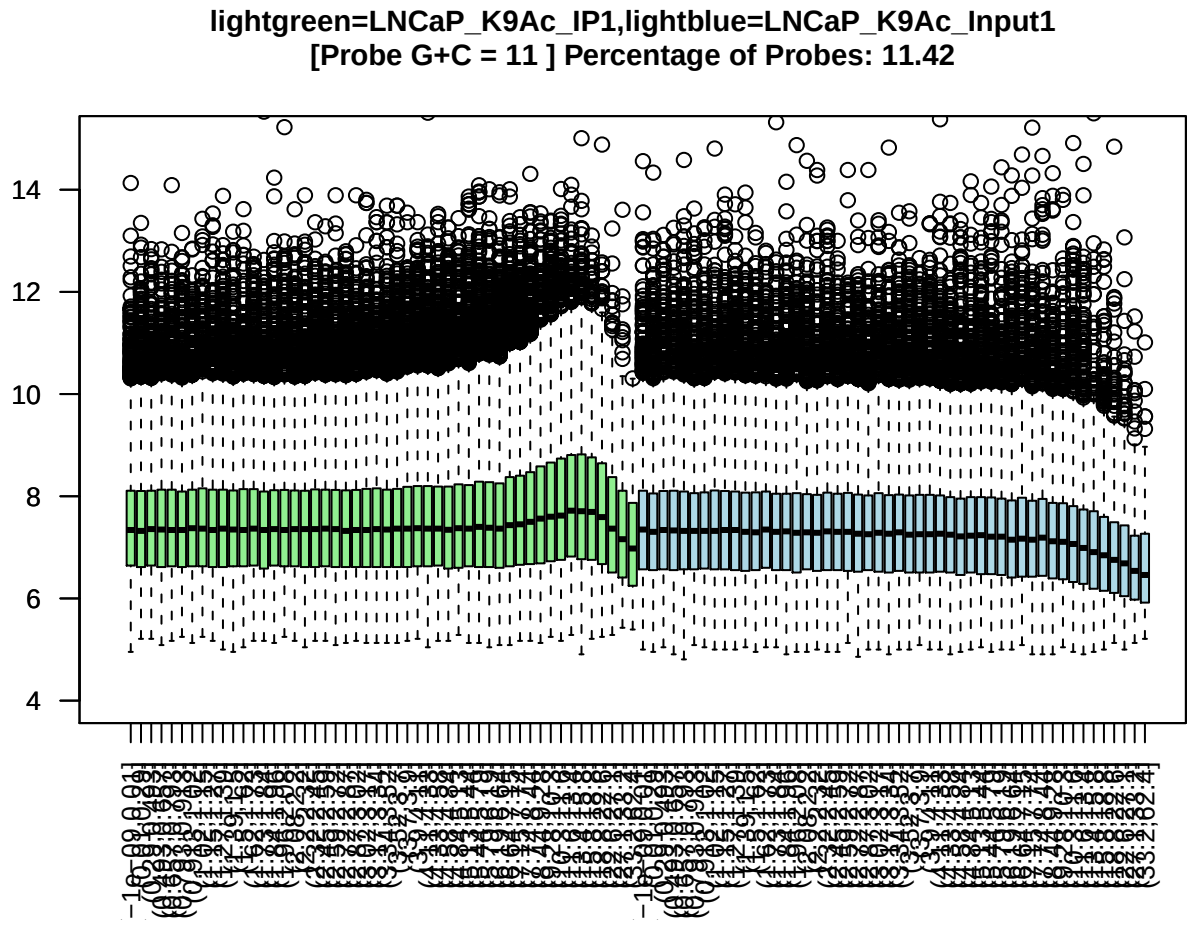
6.1 cpgBoxplots and cpgDensityPlot – Plotting Bias/Enrichment related to CpG Density

The function `cpgBoxplots` is used to create boxplots of intensity, with probes stratified by GC content and then grouped by the CpG density surrounding 600 bases of each probe (in our examples, we use a

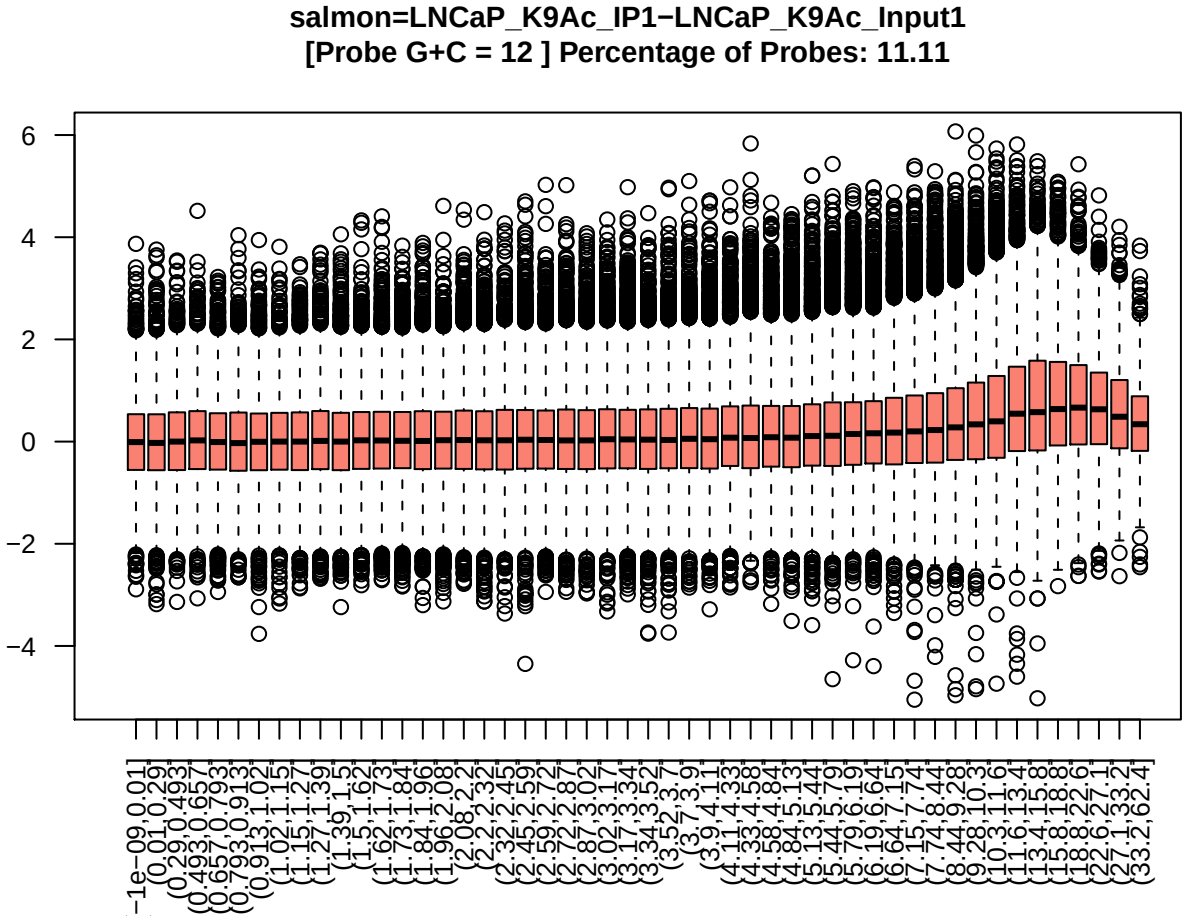
window of 600 bases, but this can be tailored to a given experiment). A range of probe GC content can be given, via the `gcContent` parameter, to narrow down analysis to probes with GC content in a certain range. Each level of GC content is plotted in a separate graph. The `samples` parameter must be a vector of length 2, that gives the columns of the intensity matrix to use. The number of bins that the CpG density is binned into is controlled with the `nBins` parameter. Setting the parameter `calcDiff` to **TRUE** will create a single plot of the difference of the first sample in the sample vector to the second sample.

Examples:

```
cpgBoxplots(csTU, samples = c(2, 1), gcContent = 11, nBins = 50)
```

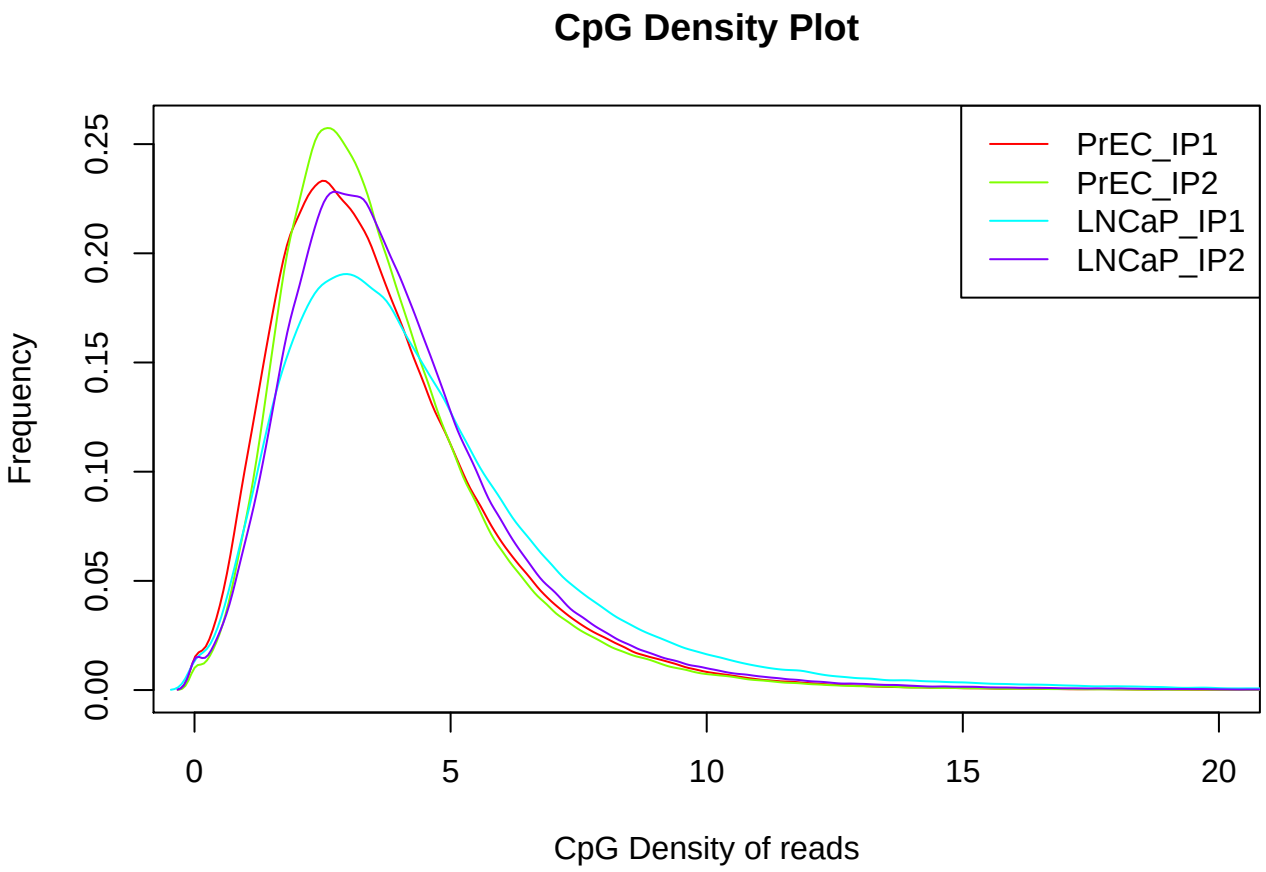



```
cpgBoxplots(csTU, samples = c(2, 1), gcContent = 12, nBins = 50,
    calcDiff = TRUE)
```



The function `cpgDensityPlot` uses `cpgDensityCalc` (see Section "Utility Functions") on sequencing data to examine the CpG density distribution of reads, a useful quality control step for methylated DNA enrichment experiments.

```
takeSample <- function(rs, f = 0.1) {
    gdapply(rs, function(x) {
        p <- x[["+"]]
        n <- x[["-"]]
        list(`+` = sample(p, f * length(p)), `-` = sample(n,
            f * length(n)))
    })
}
rsSub <- takeSample(rs, f = 0.1)
cpgDensityPlot(rsSub, seqLen = 300, organism = Hsapiens)
```

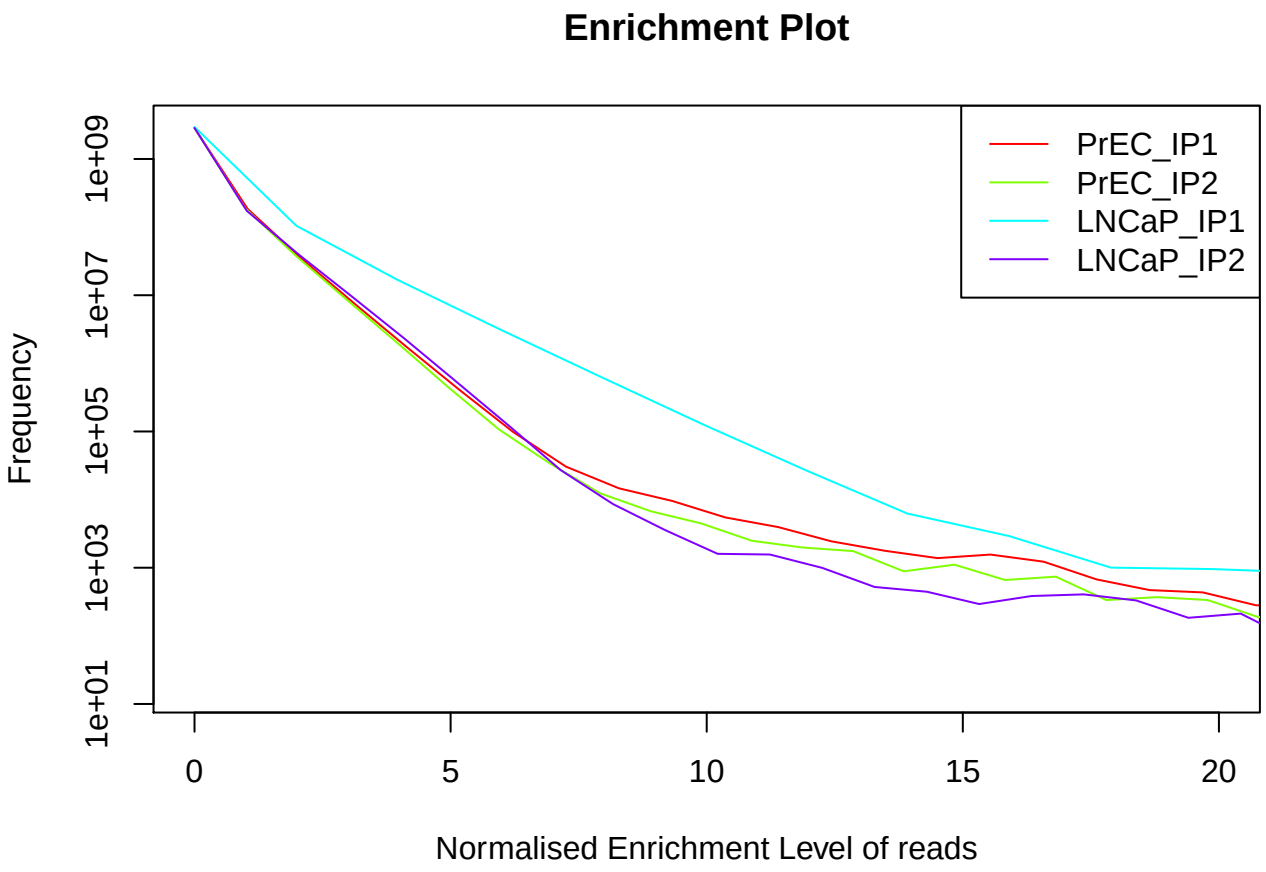


6.2 enrichmentPlot – Plotting Enrichment over the whole genome

Similarly, `enrichmentPlot` (which uses `enrichmentCalc`) can plot the distribution of enrichment over the whole genome, useful for quality control of any enrichment-based sequencing experiment.

```
enrichmentPlot(rsSub, seqLen = 300, organism = Hsapiens, verbose = TRUE)
```

```
Extending reads
Creating coverage object
Calculating enrichment
Normalising to reads per lane
```



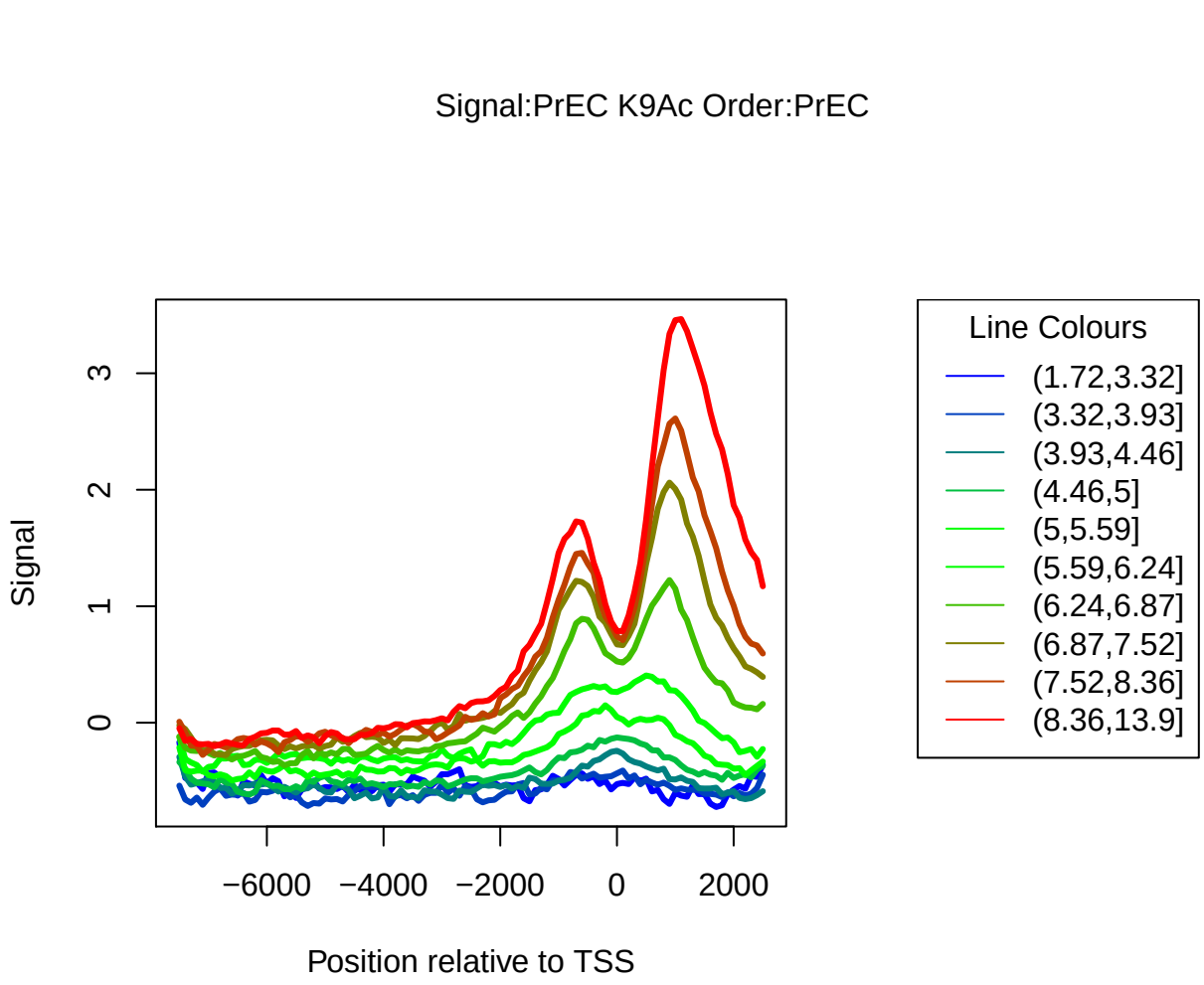
6.3 binPlots – average signal proximal to annotation in bins

Binned plots are a way of exploring the relationship between epigenomic signals and some external measure. For example, it is possible to bin the tiling array or sequencing data by the level of expression and visualize the signal across a region up- and downstream of the transcription start sites. There are three types of visualization: lineplots, heatmaps and "terrain" diagrams.

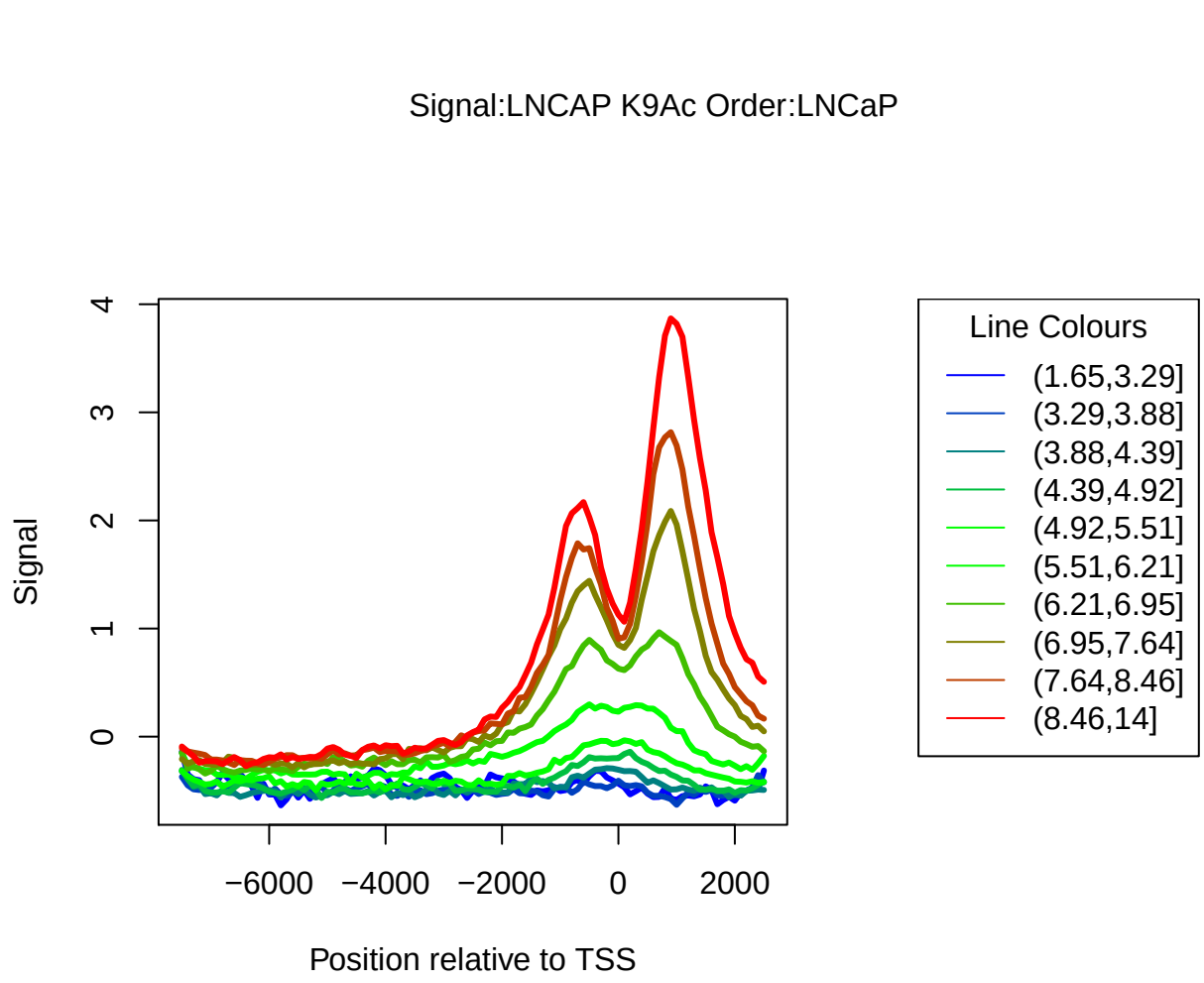
We use expression data processed through RMA from above (the 'em' data.frame) and use the table of annotation (the 'genes' data.frame).

The first time `binPlots` is called, it will create a mapping between the gene positions supplied and the probes on the tiling array, which is returned. Keeping this mapping and passing it to subsequent `binPlots` calls as the `probeMap` parameter will speed up plotting. Here are some examples:

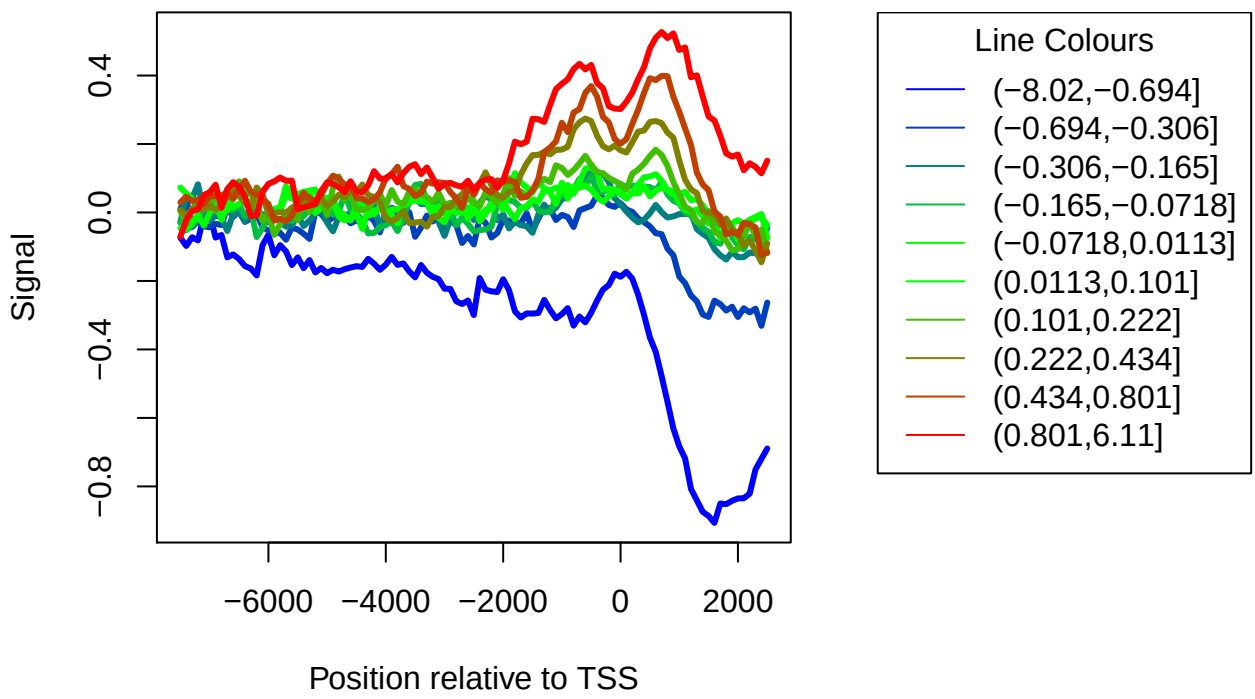
```
lookupT <- binPlots(extract(csTS, 1), coordinatesTable = genes,
    ordering = emE[, "PrEC", drop = FALSE], plotType = "line",
    nbins = 10)
```




```
binPlots(extract(csTS, 2), probeMap = lookupT, ordering = emE[,
    "LNCaP", drop = FALSE], plotType = "line", nbins = 10)
```

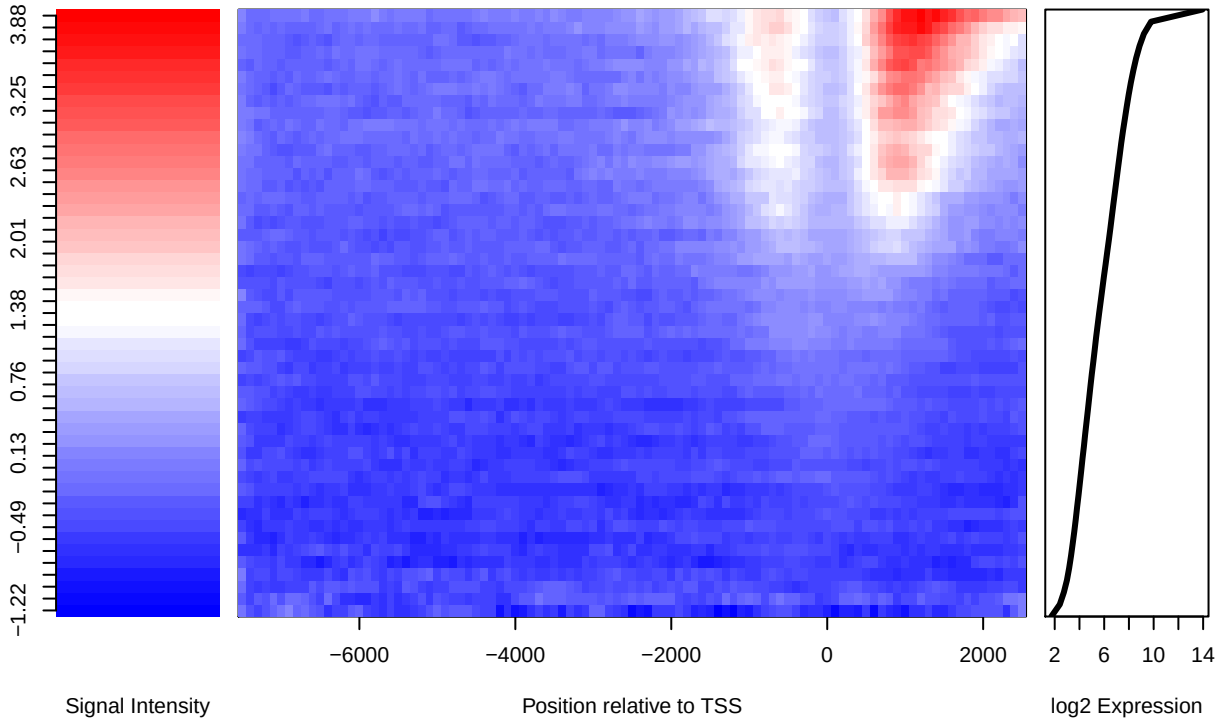



```
binPlots(extract(csTS, 3), probeMap = lookupT, ordering = emE[,
    "LNCaP-PrEC", drop = FALSE], plotType = "line", nbins = 10)
```



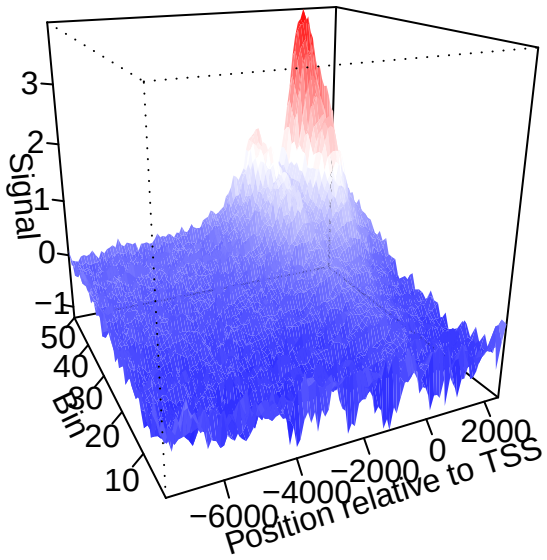
The `binPlots` functions also can plot a `heatmap` or `terrain` map, both of which benefit from an increase to the `nBins` argument.

```
binPlots(extract(csTS, 1), probeMap = lookupT, ordering = emE[,
    "PrEC", drop = FALSE], plotType = "heatmap", nbins = 50)
```

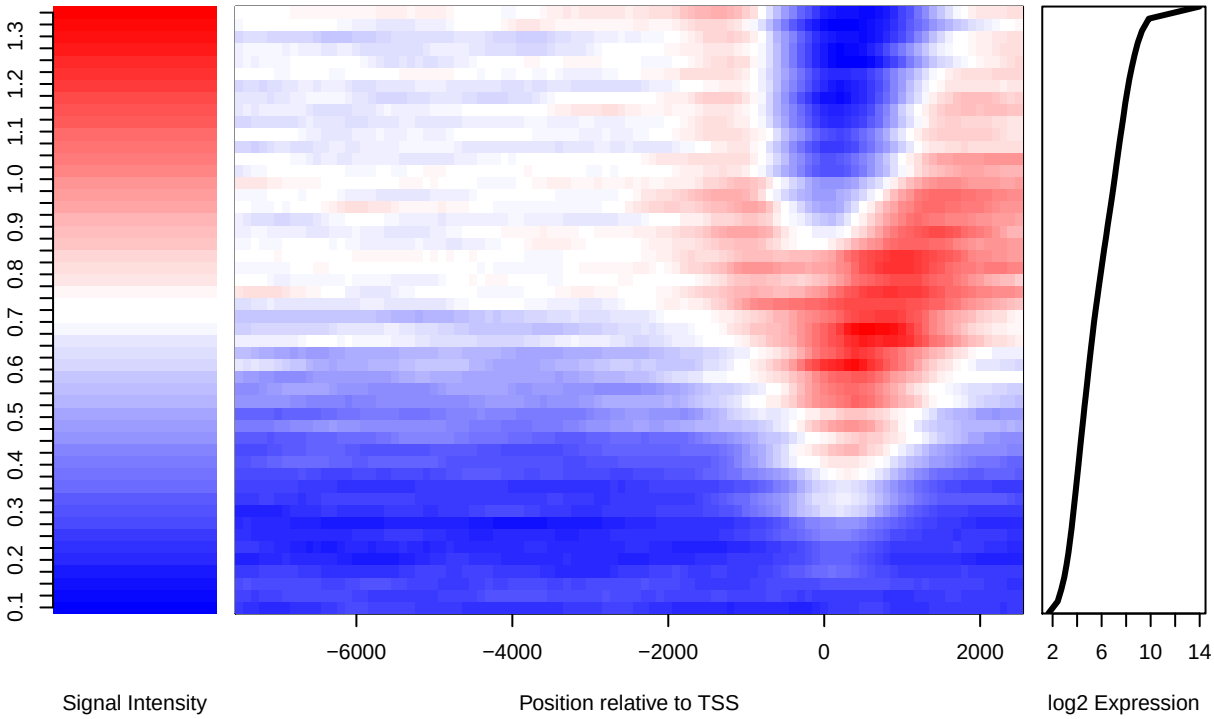



```
binPlots(extract(csTS, 1), probeMap = lookupT, ordering = emE[,
    "PrEC", drop = FALSE], plotType = "terrain", nbins = 50)
```

Signal:PrEC K9Ac Order:PrEC




```
binPlots(rs, coordinatesTable = genes, design = designSeq[, "LNCaP IP",
    drop = FALSE], ordering = emE[, "LNCaP", drop = FALSE], plotType = "heatmap",
    nbins = 50, seqLen = 300, libSize = "lane")
```



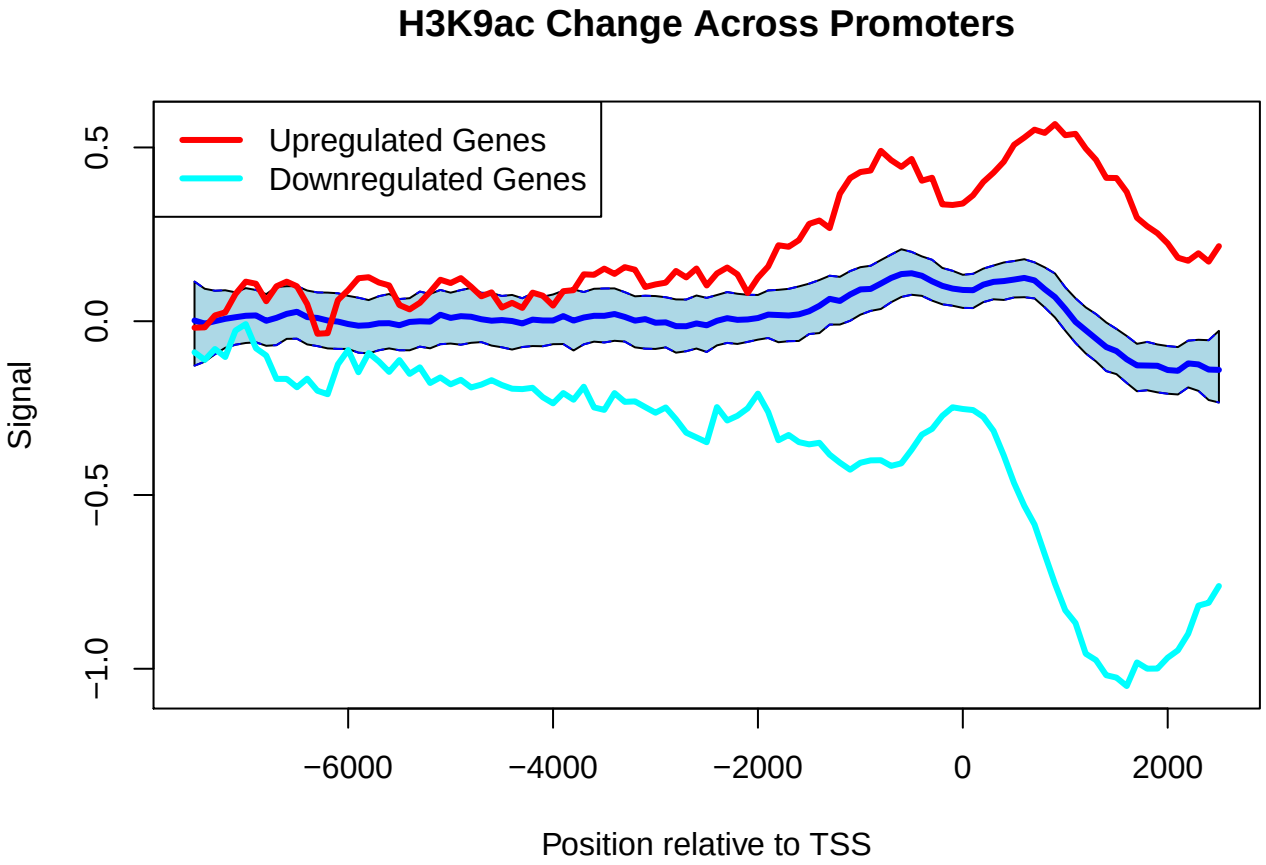
6.4 Gene set analysis of epigenetic marks

Sometimes, experiments may have a set of genes of interest (for example, genes highly expressed under a certain condition or a pathway). A number of graphs comparing the epigenetic marks of these genes, versus a random sample of genes, can be created in the one command. A lookup table, as created previously, must be supplied via the parameter `probeMap`. The genes of interest are given in the list `geneList`. Each element of the list can either be a logical vector for each gene, or alternatively, could be an integer vector specifying the rows of the data matrix that are of interest. The confidence interval defaults to 95%, but can be user - defined through the parameter `confidence`. The parameter `nSamples` determines how many probes will be sampled for the null distribution (default: 1000).

The examples below illustrate the profile of a random selection of genes, with confidence bounds, as well as the profile of the gene set of interest.

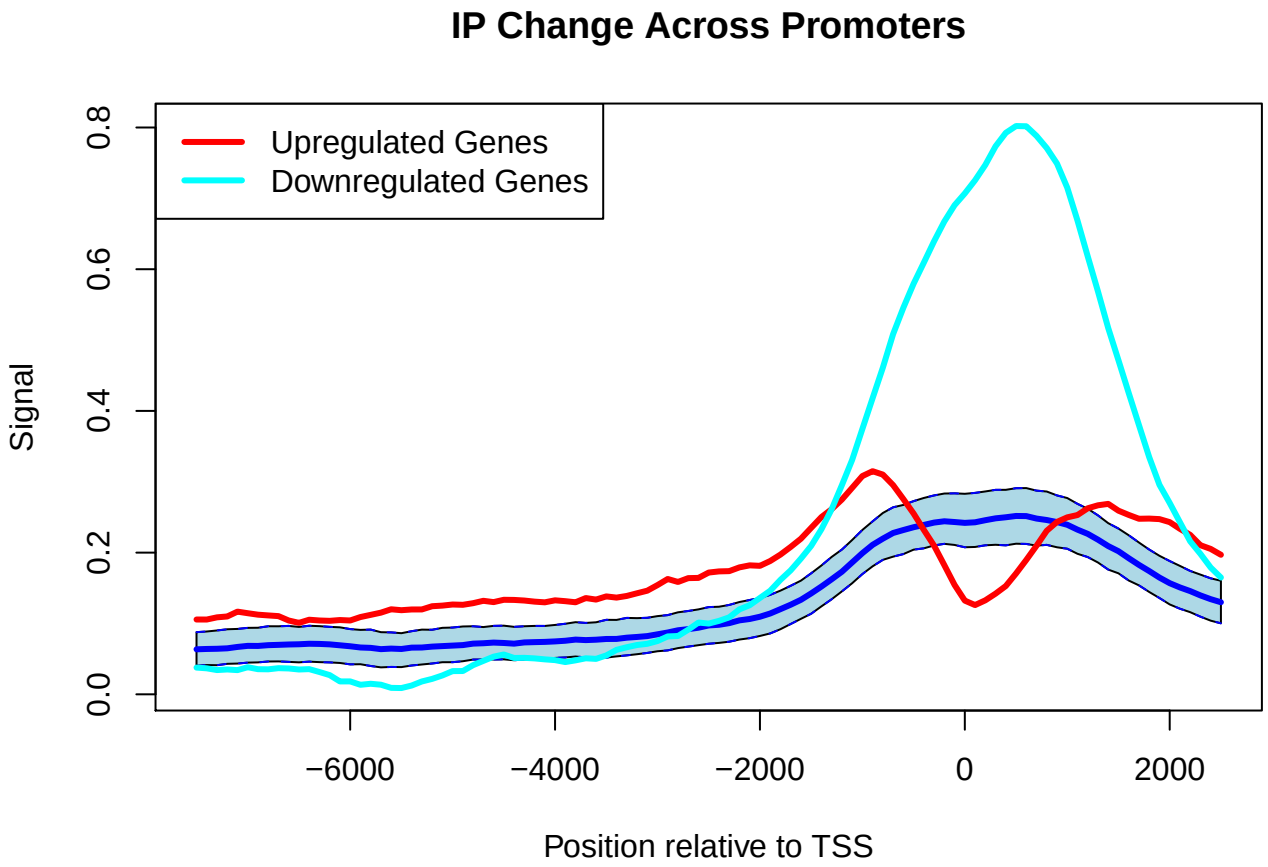
Example:

```
geneList <- list(`Upregulated Genes` = which(emE[, "LNCaP-PrEC"] >
    1), `Downregulated Genes` = which(emE[, "LNCaP-PrEC"] < -1))
significancePlots(extract(csTS, 3), probeMap = lookupT, geneList = geneList,
    titles = "H3K9ac Change Across Promoters")
```



And again for sequencing data:

```
significancePlots(rs, coordinatesTable = genes, design = designSeq[,
    "LNCaP-PrEC IP", drop = FALSE], geneList = geneList, seqLen = 300,
    titles = "IP Change Across Promoters")
```



7 Untargeted global analysis of Epigenetic Marks

To find regions that are statistically significantly different between immunoprecipitations and inputs, the `regionStats` function is used. Regions are called based on a trimmed mean score, and the score threshold is based on the lowest cutoff that gives an acceptable False Discovery Rate (FDR). The amount of trimming before calculation of the mean is given by `meanTrim`. The number of probes in the window that have to be above the threshold for the window to be called a significant region is given by `nProbes`. The FDR is calculated by randomly permuting the probe intensities a number of times, determined by `nPermutations`, and the minimum ratio of regions in the permuted sample to the actual sample is chosen as the FDR of the threshold. The threshold is progressively lowered until the FDR becomes greater than `fdrLevel`, and the previous score threshold is chosen to be used in deciding significant regions. Regions that are separated by less than `maxGap` bases are joined together into one larger region. The regions are returned as a list of data frames, in the list element `regions`. There is one data frame for each contrast. The examples below analyze (and randomize within) chromosome 7, in the interest of saving time.

Example:

```
ind <- getCellIndices(getUniqueCdf(cdfT), unit = indexOf(getUniqueCdf(cdfT),
    "chr7F"), unlist = TRUE, useNames = FALSE)
```

```
col <- "LNCaP-PrEC K9Ac"
regs <- regionStats(csTMNU, design[, col, drop = FALSE], ind = ind,
    probeWindow = 300, verbose = FALSE)
o <- order(-abs(regs$regions[[col]]$score))
head(regs$regions[[col]][o, ])
```

```
     chr     start       end  score startInd endInd
156 chr7 115927830 115929415 -7.514   151820 151864
58  chr7  17305627  17306493 -6.943    22540  22564
101 chr7  41707382  41709181 -6.138    56414  56464
162 chr7 116100065 116102081 -5.933   152375 152431
126 chr7  83660428  83663179 -5.460   104342 104413
139 chr7  98577243  98577788 -5.220   122411 122426
```

It may be desired to view these regions in a graphical way. The `aroma.affymetrix` function `writeSgr` is a quick method to export the scores into a format readable by the Integrated Genome Browser. The file created is a tab-delimited text file with three columns; chromosome name, probe positon, and probe score. There are no column headings in the file.

Example:

```
writeSgr(csTS)
```

This the IGB view of the top differentially K9Ac enriched TSS on the forward strand, as determined by the previous example.

At present, there is no analogous `regionStats` procedure for sequencing-based data.

8 Targeted analysis of epigenetic marks

Rather than looking across the entire genome (or entire region represented on a tiling microarray), one can target the analysis to regions of interest (e.g. promoters). The `blocksStats` procedure is a general-purpose tool for this. The function works in one of two modes. If the parameter `useAsRegions` is set to `TRUE`, the `start` and `end` columns of the `coordinatesTable` are used as the regions to consider. Otherwise, the start of the region of interest (e.g. transcription start site) is chosen based on the `strand` column, and the region to use for calculation is defined by the region that starts a number of bases upstream, defined by the parameter `upStream`, and a number of bases downstream, defined by the parameter `downStream`. The minimum number of probes that must be present in a defined region for statistics calculations to be carried out is given by the parameter `minNRobust`.

Columns for Region-level statistics are added to the input `coordinatesTable` data.frame, including the average difference of probes in that specified region, t-statistics, p-values and adjusted p-values for each column of the design matrix.

Example:

```
bs <- blocksStats(csTMNU, genes, design = design, upStream = 2000,
    downStream = 2000)
```

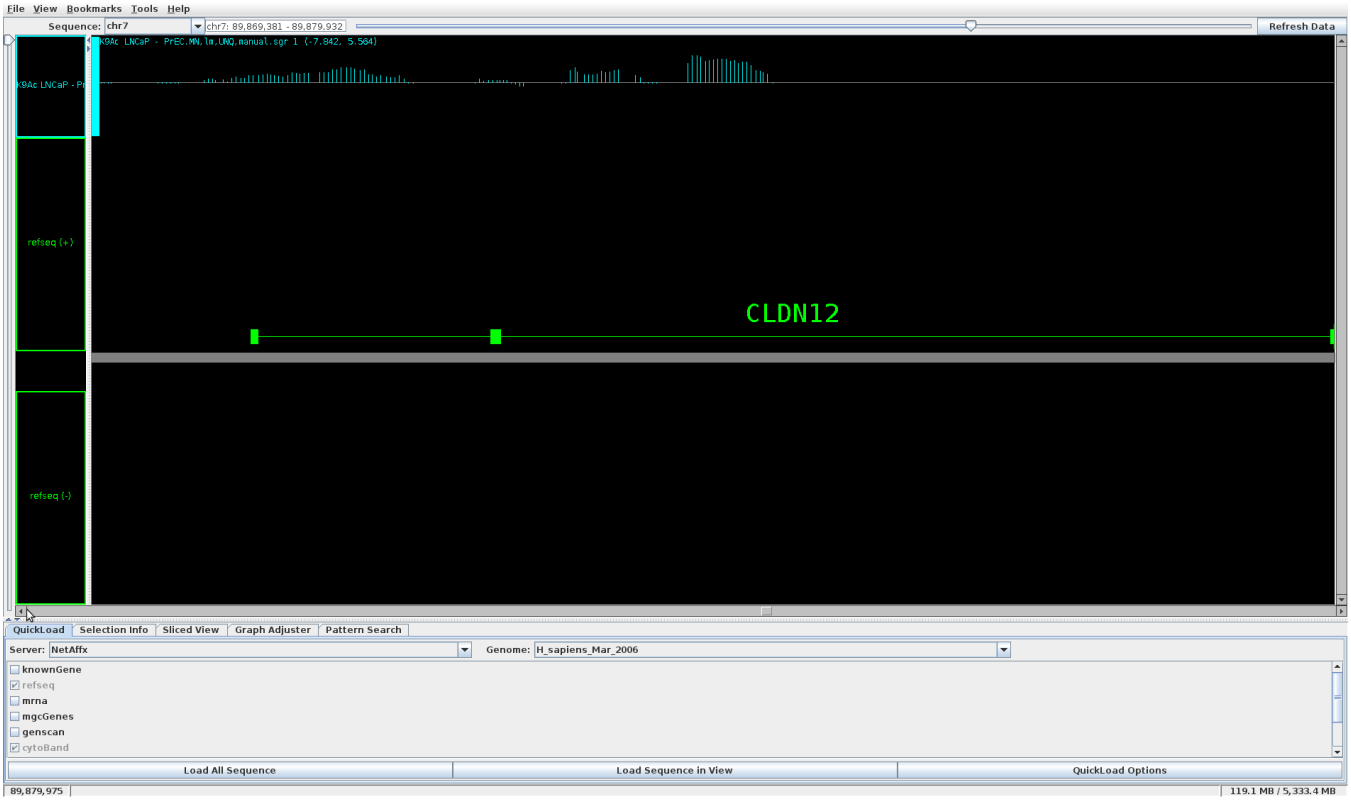


Figure 1: IGB view of the most differentially K9Ac enriched region.

```
Processing mapping between probes and genes.
Mapping done.
Processing mapping between probes and genes.
Mapping done.
```

```
o <- order(bs$pvals.LNCaP.PrEC)
head(bs[o, ])
```

```
        name   chr strand    start      end    symbol selected_identifier
17631 8095728  chr4      + 75449723 75473341      EREG           NM_001432
16072 8078330  chr3      + 29297946 30021624     RBMS3        NM_001003793
6109  7964733 chr12      - 64438069 64507021 LOC204010           BC107865
21673 8140668  chr7      - 83425594 83662153    SEMA3A           NM_006080
501   7902527  chr1      + 78729344 78776138     PTGFR           AY485530
10523 8016487 chr17      - 44157124 44161110    HOXB13           NM_006361
      df.2000.2000 meandiff.PrEC.K9Ac meandiff.LNCAP.K9Ac
17631           92         1.23502355         -0.34653331
16072          100         1.52320673          0.05972448
6109           102         1.33882712         -0.29788229
21673          100         1.26707508         -0.16779986
501             90        -0.11908496          1.09636003
10523          107        -0.01776204          0.77745199
      meandiff.LNCaP.PrEC.K9Ac tstats.PrEC.K9Ac tstats.LNCAP.K9Ac
17631                -1.581557         9.598198        -3.7053615
16072                -1.463482        11.049251         0.5434101
6109                 -1.636709         9.095209        -3.5769065
```

```
21673                -1.434875         9.477559        -2.0374225
501                   1.215445        -1.286236         8.0677212
10523                 0.795214        -0.232270         7.0425652
      tstats.LNCaP.PrEC.K9Ac pvals.PrEC.K9Ac pvals.LNCAP.K9Ac
17631              -9.502240    1.598607e-15     3.601448e-04
16072              -8.884494    5.026978e-19     5.880568e-01
6109               -8.779996    8.167226e-15     5.334785e-04
21673              -8.750984    1.384024e-15     4.424916e-02
501                 8.404246    2.016599e-01     2.956143e-12
10523               8.174333    8.167720e-01     1.899730e-10
      pvals.LNCaP.PrEC.K9Ac adjpvals.PrEC.K9Ac adjpvals.LNCAP.K9Ac
17631          2.545090e-15       3.417957e-13        1.175795e-03
16072          2.745724e-14       3.197620e-16        6.644236e-01
6109           4.040443e-14       1.198138e-12        1.670330e-03
21673          5.368239e-14       3.090680e-13        7.888235e-02
501            5.953247e-13       2.728532e-01        9.551859e-11
10523          6.522252e-13       8.563111e-01        3.294286e-09
      adjpvals.LNCaP.PrEC.K9Ac
17631             5.115122e-11
16072             2.697272e-10
6109              2.697272e-10
21673             2.697272e-10
501               2.184737e-09
10523             2.184737e-09
```

Quite conveniently, the same function can also be used to perform statistical analysis on sequencing data. The sequencing results are first read into a `GenomeDataList` object. There are two additional options that become available for `blocksStats` when it is being used to analyse sequencing data. The `seqLen` parameter can be used to extend the aligned reads to a certain length. Since the fragments of DNA sequenced are usually longer than the reads given by the experimental procedure, this single number argument will extend all reads based on where in the genome they align to, so that they all are `seqLen` in length. The other argument specific to sequencing is `total.lib.size`, which is a boolean value that specifies whether to use the total number of reads genome - wide (using the `laneCounts` function), or the total number of reads falling within the starts and ends as described by `coordinatesTable`, as the library size for statistical calculations. If it is `TRUE`, the library size will the the number of reads in the whole genome. `FALSE` will mean that the number of reads in the coorinatesTable regions will be taken as the library size. The design matrix needs to be modified, since `edgeR` does pairwise comparisons. Treatments should have 1, controls -1, and others 0.

The `regionStats` procedure for sequencing data will return (library size) quantile-normalized counts and statistics for differential expression from the edgeR functions.

Example:

```
statistics <- blocksStats(rs, genes, designSeq[, "LNCaP-PrEC IP",
    drop = FALSE], upStream = 2000, downStream = 2000, seqLen = 300,
    libSize = "lane")
```

```
Generating table of counts
PrEC_IP1: chr1 chr5 chr21 chr11 chr19 chr10 chr12 chr15 chr6 chr2 chrX chr13 chr20 chr14 chr4 chr3 chr16 ch
PrEC_IP2: chr1 chr5 chr21 chr11 chr19 chr10 chr12 chr15 chr6 chr2 chrX chr13 chr20 chr14 chr4 chr3 chr16 ch
LNCaP_IP1: chr1 chr5 chr21 chr11 chr19 chr10 chr12 chr15 chr6 chr2 chrX chr13 chr20 chr14 chr4 chr3 chr16 c
```

```
LNCaP_IP2: chr1 chr5 chr21 chr11 chr19 chr10 chr12 chr15 chr6 chr2 chrX chr13 chr20 chr14 chr4 chr3 chr16 c
Processing column 1 of design matrix
Comparison of groups:  1 - -1
```

```
head(statistics)
```

```
     name  chr strand  start    end    symbol selected_identifier position
1 7896759 chr1      + 781253 783614 LOC643837            AK096570   781253
2 7896761 chr1      + 850983 869824    SAMD11           NM_152486   850983
3 7896779 chr1      + 885829 890958    KLHL17           NM_198317   885829
4 7896798 chr1      + 891739 900345   PLEKHN1           NM_032129   891739
5 7896817 chr1      + 938709 939782     ISG15           NM_005101   938709
6 7896822 chr1      + 945365 981355      AGRN           NM_198576   945365
  PrEC_IP1 PrEC_IP2 LNCaP_IP1 LNCaP_IP2 PrEC_IP1_pseudo PrEC_IP2_pseudo
1       41       65        27        87        35.07838        53.89927
2       16       27        64        88        13.59353        22.49468
3       77       67        58        85        66.56343        54.97646
4       84       79        87       143        72.54215        64.93175
5       78       90        40        80        67.19515        74.20940
6       24       29        52        74        20.62102        23.96068
  LNCaP_IP1_pseudo LNCaP_IP2_pseudo logConc_LNCaP-PrEC IP logFC_LNCaP-PrEC IP
1         46.34933         74.33991              -17.28339           0.45840201
2        104.54489         74.52252              -17.66458           2.29711660
3         94.90800         72.02990              -16.83995           0.44843534
4        143.29426        121.42386              -16.41567           0.94109028
5         66.36191         68.03676              -16.88168          -0.07316343
6         84.90444         62.66353              -17.65248           1.71386182
  PValue_LNCaP-PrEC IP FDR_LNCaP-PrEC IP
1         1.736013e-01      3.338921e-01
2         7.725960e-11      2.787193e-09
3         1.423048e-01      2.891836e-01
4         1.537698e-03      8.416589e-03
5         8.663815e-01      9.890557e-01
6         9.254953e-07      1.315222e-05
```

The absolute number of counts within the regions can be determined with the functions annotationCounts and annotationBlocksCounts. For example:

```
genes$position <- ifelse(genes$strand == "+", genes$start, genes$end)
head(genes)
```

```
     name  chr strand  start    end    symbol selected_identifier position
1 7896759 chr1      + 781253 783614 LOC643837            AK096570   781253
2 7896761 chr1      + 850983 869824    SAMD11           NM_152486   850983
3 7896779 chr1      + 885829 890958    KLHL17           NM_198317   885829
4 7896798 chr1      + 891739 900345   PLEKHN1           NM_032129   891739
5 7896817 chr1      + 938709 939782     ISG15           NM_005101   938709
6 7896822 chr1      + 945365 981355      AGRN           NM_198576   945365
```

```
counts <- annotationCounts(rs, genes, 10000, 2500, 300)
```

```
PrEC_IP1: chr1 chr5 chr21 chr11 chr19 chr10 chr12 chr15 chr6 chr2 chrX chr13 chr20 chr14 chr4 chr3 chr16 ch
PrEC_IP2: chr1 chr5 chr21 chr11 chr19 chr10 chr12 chr15 chr6 chr2 chrX chr13 chr20 chr14 chr4 chr3 chr16 ch
LNCaP_IP1: chr1 chr5 chr21 chr11 chr19 chr10 chr12 chr15 chr6 chr2 chrX chr13 chr20 chr14 chr4 chr3 chr16 c
LNCaP_IP2: chr1 chr5 chr21 chr11 chr19 chr10 chr12 chr15 chr6 chr2 chrX chr13 chr20 chr14 chr4 chr3 chr16 c
```

```
head(counts)
```

```
        PrEC_IP1 PrEC_IP2 LNCaP_IP1 LNCaP_IP2
7896759       85      109        55       154
7896761      143      200       171       313
7896779      364      398       284       446
7896798      301      290       281       411
7896817      220      221       198       309
7896822      187      193       172       248
```

```
myregs <- data.frame(chr = c("chr2", "chr7"), start = c(5e+05,
    1e+06), end = c(550000, 1005000), strand = c("+", "-"), name = paste("Region",
    1:2))
print(myregs)
```

```
   chr start     end strand     name
1 chr2 5e+05  550000      + Region 1
2 chr7 1e+06 1005000      - Region 2
```

```
counts <- annotationBlocksCounts(rs, myregs, 200)
```

```
PrEC_IP1: chr2 chr7
PrEC_IP2: chr2 chr7
LNCaP_IP1: chr2 chr7
LNCaP_IP2: chr2 chr7
```

```
counts
```

```
         PrEC_IP1 PrEC_IP2 LNCaP_IP1 LNCaP_IP2
Region 1      364      383       184       375
Region 2      171      145       143       159
```

9 Utility Functions

Repitools contains a number of generally useful functions.

9.1 Importing Next Generation Sequencing Data

After alignment of Next Generation Sequencing data to the genome of interest by an external tool (e.g Bowtie, bwa, maq, ELAND etc) we need to import the data into R for analysis. The ShortRead packages AlignedRead function performs this, but keeps a lot of data we are not interested in anymore (the sequence and quality scores for each read for instance). So we follow the method of the Bioconductor chipseq package which only stores each reads chromosome, position and direction in a compact GenomeData object.

Example:

```
require(chipseq)
rs1 <- readAligned("./", paste("^", filename, "$", sep = ""),
    type = "Bowtie")
rs1 <- as(rs1, "GenomeData")
```

We then save this GenomeData object as an .Rdata file for later use. When working with multiple lanes of sequencing, it is much more convenient to combine them into a single GenomeDataList object.

Example:

```
rs <- GenomeDataList(list(`Sample 1` = rs1, `Sample 2` = rs2))
```

This GenomeDataList object now can be used as input to our sequencing processing functions.

9.2 Calculating CpG Density of Specific Regions

To calculate the CpG density of specific regions of the genome, the cpgDensityCalc function can be used. The parameter locationsTable can be a data frame with two columns, chr and position. The number windowSize is how many bases upstream and downstream of position to take into account for the calculation. You can weight the CpGs as a function of distance to the centre of the region of interest, if desired.

Example:

```
myLocations <- data.frame(chr = c("chr1", "chr17"), position = c(1e+05,
    250000))
cpgDensityCalc(myLocations, 500, organism = Hsapiens)
```

```
[1] 0.000 0.884
```

9.3 Reading Nimblegen Data Quickly

Using other packages usually requires definition text files or specific directory structures. The reading has been streamlined with the function loadSampleDirectory, which loads all .pair files is a specified directory. The first argument, path is the path of the directory where the .pair files are located. Additionally, the parameter ndf gives a data frame representation of the releveant Nimblegen Data File, which is the output from the processNDF function. The what argument is a great time - saver. It describes how to process the intensities as they are being read into the matrix. The strings that can be passed in for this argument and their effects are:

String	Effect
Cy3	The matrix will have the log_2 intensity of the green channel for each array.
Cy5	The matrix will have the log_2 intensity of the red channel for each array.
Cy3/Cy5	The matrix will have the log_2 of the ratio of the green channel to the red channel.
Cy5/Cy3	The matrix will have the log_2 of the ratio of the red channel to the green channel.
Cy3andCy5	The matrix will have two columns for each array, the first being the log_2 of the green channel, and the next being the log_2 of the red channel.
Cy5andCy3	The matrix will have two columns for each array, the first being the log_2 of the red channel, and the next being the log_2 of the green channel.

9.4 Visualization of multiple heatmaps

See the `multiHeatmap` example and documentation for the creating multi-panel heatmaps allowing complete control over colour scales and spacing between.

10 Future Directions

Analyses of next-generation sequencing data, especially for epigenomic data, are in their infancy. Further implementations will be made available in the near future. Experimenters are encouraged to provide feedback on the implemented routines as well as suggestions for further procedures.

11 Environment

This user manual was built using `Sweave` with the following $\mathbb{R}$ environment:

```
sessionInfo()
```

```
R version 2.10.1 (2009-12-14)
x86_64-pc-linux-gnu

locale:
 [1] LC_CTYPE=en_AU.UTF-8       LC_NUMERIC=C
 [3] LC_TIME=en_AU.UTF-8        LC_COLLATE=en_AU.UTF-8
 [5] LC_MONETARY=C              LC_MESSAGES=en_AU.UTF-8
 [7] LC_PAPER=en_AU.UTF-8       LC_NAME=C
 [9] LC_ADDRESS=C               LC_TELEPHONE=C
[11] LC_MEASUREMENT=en_AU.UTF-8 LC_IDENTIFICATION=C

attached base packages:
[1] grid      stats     graphics  grDevices utils     datasets  methods
[8] base

other attached packages:
 [1] chipseq_0.2.1                    ShortRead_1.4.0
 [3] lattice_0.18-3                   edgeR_1.4.7
```

```
 [5] gplots_2.7.4                       caTools_1.10
 [7] bitops_1.0-4.1                     gdata_2.6.1
 [9] gtools_2.6.1                       gsmoothr_0.1.4
[11] BSgenome.Hsapiens.UCSC.hg18_1.3.16 BSgenome_1.14.2
[13] Biostrings_2.14.7                  limma_3.2.1
[15] RepitoolsExamples_1.0.11           Repitools_1.0.6
[17] aroma.affymetrix_1.5.0             aroma.apd_0.1.7
[19] affxparser_1.18.0                  R.huge_0.2.0
[21] aroma.core_1.5.0                   aroma.light_1.15.1
[23] matrixStats_0.1.9                  R.rsp_0.3.6
[25] R.cache_0.2.0                      R.filesets_0.8.0
[27] digest_0.4.2                       R.utils_1.3.3
[29] R.oo_1.6.7                         IRanges_1.4.7
[31] R.methodsS3_1.1.0

loaded via a namespace (and not attached):
[1] Biobase_2.6.0 hwriter_1.1
```