The chromstaR user's guide

Aaron Taudt*

May 28, 2015

Contents

1 Introduction

ChIP-seq has become the standard technique for assessing the genome-wide chromatin state of DNA. *chromstaR* provides functions for the joint analysis of multiple ChIP-seq samples. It allows peak calling for transcription

*a.s.taudt@umcg.nl

factor binding and histone modifications with a narrow (e.g. H3K4me3, H3K27ac, ...) or broad (e.g. H3K36me3, H3K27me3, ...) profile. All analysis can be performed on each sample individually (=univariate), or in a joint analysis considering all samples simultaneously (=multivariate).

2 Outline of workflow

Every analysis with the *chromstaR* package starts from aligned reads in either BAM or BED format. In the first step, the genome is partitioned into non-overlapping, equally sized bins and the reads that fall into each bin are counted. These read counts serve as the basis for both the univariate and the multivariate peak- and broad-region calling. Univariate peak calling is done by fitting a three-state Hidden Markov Model to the binned read counts. Multivariate peak calling for $\mathcal{S}$ samples is done by fitting a $2^{\mathcal{S}}$-state Hidden Markov Model to all binned read counts.

3 Univariate analysis

3.1 Task 1: Peak calling for a narrow histone modification

Examples of histone modifications with a narrow profile are H3K4me3, H3K9ac and H3K27ac. For such peak-like modifications, the bin size should be set to a value between 200bp and 1000bp.

If you want to do this example starting from a BAM file, you should have the *chromstaRExampleData* package installed. Otherwise you can skip the first step (Binning) and start from already binned data. If your input is in BED format, use function `bed2binned` instead. Please refer to the FAQ section 6 for more details and troubleshooting on the binning step.

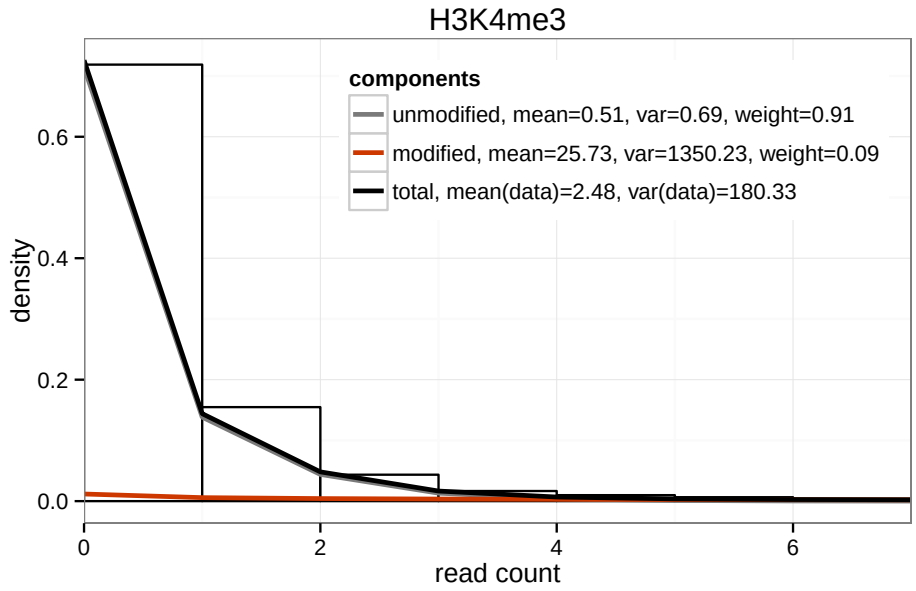
```
library(chromstaR)
```

```
## === Step 1: Binning ===
# We use bin size 1000bp and chromosome 12 to keep the example quick
library(chromstaRExampleData)
bamfile <- getExampleFilesBAM('H3K4me3')[1]
binned.data <- bam2binned(bamfile, bamindex=bamfile, binsize=1000,
                          chromosomes='chr12')
```

```
## === Step 2: Peak calling ===
# We load the binned.data from step 1 (this is only necessary if step 1 was skipped)
data("liver-H3K4me3-BN-male-bio2-tech1_chr12.bam_binsize_1000")
# We restrict the peak calling to 60 seconds to keep this example quick.
model <- callPeaksUnivariate(binned.data, ID='H3K4me3', max.time=60)
```

```
## Replaced read counts > 163 (99.9% quantile) by 163 in 46 bins.  Set option 'read.cutoff.quantile=1'
to disable this filtering.  This filtering was done to increase the speed of the HMM and
should not affect the results.
## ---------------------------------- Try 1 of 1 ----------------------------------

## HMM: number of states = 3
## HMM: number of bins = 46782
## HMM: maximum number of iterations = none
## HMM: maximum running time = 60 sec
## HMM: epsilon = 0.01
## HMM: data mean = 2.47595, data variance = 180.327
##  Iteration              log(P)              dlog(P)     Diff in state 2    Time in sec
##          0                -inf                   -                   -              0
##          1       -60305.655970                 inf               27150              0
##          2       -57564.343441         2741.312529                7264              0
##          3       -55096.051255         2468.292186               16265              1
##        ...                 ...                 ...                 ...            ...
##        124       -50021.551800            0.010576               23718              2
##        125       -50021.541731            0.010068               23717              2
##        126       -50021.532143            0.009588               23715              2
## HMM: Convergence reached!
## HMM: Recoding posteriors ...

## Calculating states from posteriors ...  0.24s
## Making segmentation ...  0.44s
```

```
## === Step 3: Checking the fit ===
# For a narrow modification, the fit should look something like this,
# with the 'modified'-component near the bottom of the figure
plot(model)
```




```
## === Step 4: Export to genome browser ===
# We can export peak calls and binned read counts with
exportUnivariates(list(model), filename='your-file', what=c('peaks','reads'))
```

3.2 Task 2: Peak calling for a broad histone modification

Examples of histone modifications with a broad profile are H3K9me3, H3K27me3, H3K36me3, H4K20me1. These modifications usually cover broad domains of the genome, and the enrichment is best captured with a bin size between 500bp and 2000bp.

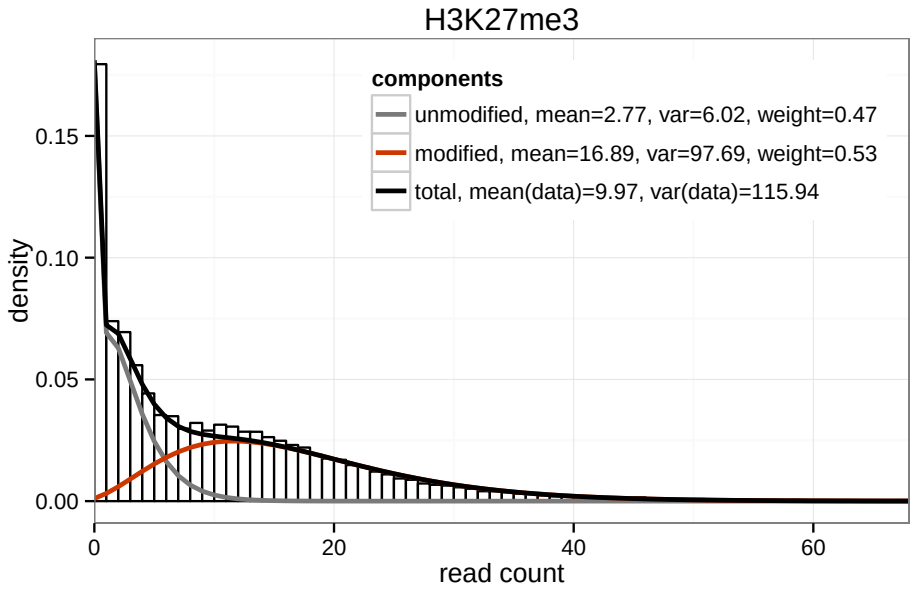
If you want to do this example starting from a BAM file, you should have the *chromstaRExampleData* package installed. Otherwise you can skip the first step (Binning) and start from already binned data. If your input is in BED format, use function `bed2binned` instead. Please refer to the FAQ section 6 for more details and troubleshooting on the binning step.

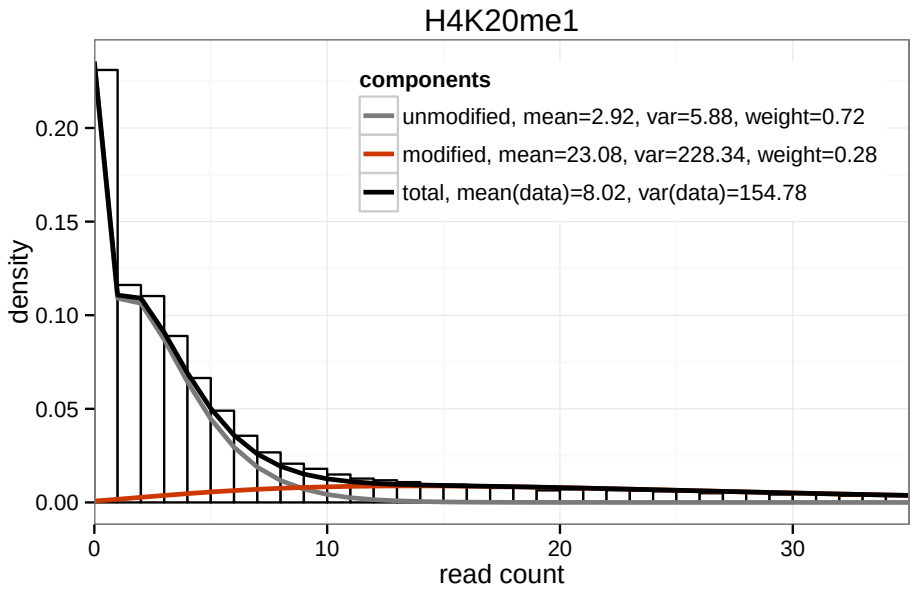
```
library(chromstaR)
```

```
## === Step 1: Binning ===
# We use bin size 1000bp and chromosome 12 to keep the example quick
library(chromstaRExampleData)
bamfile <- getExampleFilesBAM('H3K27me3')[4]
binned.data <- bam2binned(bamfile, bamindex=bamfile, binsize=1000,
                          chromosomes='chr12')
```

```
## === Step 2: Peak calling ===
# We load the binned.data from step 1 (this is only necessary if step 1 was skipped)
data("liver-H3K27me3-BN-male-bio3-tech1_chr12.bam_binsize_1000")
# We restrict the peak calling to 60 seconds to keep this example quick.
model <- callPeaksUnivariate(binned.data, ID='H3K27me3', max.time=60)

## Replaced read counts > 70 (99.9% quantile) by 70 in 44 bins.  Set option 'read.cutoff.quantile=1'
to disable this filtering.  This filtering was done to increase the speed of the HMM and
should not affect the results.
## ------------------------------------ Try 1 of 1 -------------------------------------

## HMM: number of states = 3
## HMM: number of bins = 46782
## HMM: maximum number of iterations = none
## HMM: maximum running time = 60 sec
## HMM: epsilon = 0.01
## HMM: data mean = 9.9711, data variance = 115.938
##  Iteration              log(P)             dlog(P)     Diff in state 2    Time in sec
##          0                -inf                   -                   -              0
##          1      -145371.198541                 inf               26373              0
##          2      -138286.718658         7084.479883                4015              0
##          3      -135011.421891         3275.296767                3861              0
##        ...                 ...                 ...                 ...            ...
##         70      -130368.590285            0.011413                1908              1
##         71      -130368.580012            0.010274                1909              1
##         72      -130368.570759            0.009252                1907              1
## HMM: Convergence reached!
## HMM: Recoding posteriors ...

## Calculating states from posteriors ...  0.03s
## Making segmentation ...  0.41s
```

```
## === Step 3: Checking the fit ===
# For a broad modification, the fit should look something like this,
# with a 'modified'-component that fits the thick tail of the distribution.
plot(model)
```






```
## === Step 4: Export to genome browser ===
# We can export peak calls and binned read counts with
exportUnivariates(list(model), filename='your-file', what=c('peaks','reads'))
```

3.3 Task 3: Peak calling for ATAC-seq, DNase-seq, FAIRE-seq, ...

Peak calling for ATAC-seq and DNase-seq is similar to the peak calling of a narrow histone modification (see section 3.1). FAIRE-seq experiments seem to exhibit a broad profile with our model, so the procedure is similar to the domain calling of a broad histone modification (see section 3.2).

4 Multivariate analysis

4.1 Task 1: Integrating multiple replicates

chromstaR can be used to call peaks with multiple replicates, without the need of prior merging. The underlying statistical model integrates information from all replicates to identify common peaks. It is, however, important to note that replicates with poor quality can affect the joint peak calling negatively. It is therefore recommended to first check the replicate quality and discard poor-quality replicates. The necessary steps for peak calling for an example ChIP-seq experiment with 4 replicates are detailed below.

```
library(chromstaR)
```

```
## === Step 1: Binning ===
# !!! Skip this step if you do not have package 'chromstaRExampleData' installed !!!
library(chromstaRExampleData)
# Let's get some example data with 3 replicates
bamfiles.good <- getExampleFilesBAM('H3K27me3')[3:5]
# We fake a replicate with poor quality by taking a different mark entirely
bamfiles.poor <- getExampleFilesBAM('H4K20me1')[1]
bamfiles <- c(bamfiles.good, bamfiles.poor)
# We use bin size 1000bp and chromosome 12 to keep the example quick
binned.data <- list()
for (bamfile in bamfiles) {
  binned.data[[basename(bamfile)]] <- bam2binned(bamfile, bamindex=bamfile,
                                                 binsize=1000, chromosomes='chr12')
}
```

```
## === Step 2: Univariate peak calling ===
# We load the binned.data from step 1 (this is only necessary if step 1 was skipped)
data(replicateExample_binnedData)
# The univariate fit is obtained for each replicate
models <- list()
for (i1 in 1:length(binned.data)) {
  models[[i1]] <- callPeaksUnivariate(binned.data[[i1]], ID=paste0('Rep',i1),
                                      max.time=60)
}
```

```
## === Step 3: Check replicate correlation ===
# We run a multivariate peak calling on all 4 replicates
# A warning is issued because replicate 4 is very different from the others
multi.model <- callPeaksReplicates(models, max.time=60)

## Warning in callPeaksReplicates(models, max.time = 60): Your replicates cluster in
2 groups. Consider redoing your analysis with only the group with the highest average
coverage:
## Rep1
## Rep2
## Rep3
## Replicates from groups with lower coverage are:
## Rep4
```

```
# Checking the correlation confirms that Rep4 is very different from the others
multi.model$replicateInfo$correlation

##             Rep1       Rep2       Rep3       Rep4
## Rep1  1.0000000  0.9976476  0.9977751 -0.4126249
## Rep2  0.9976476  1.0000000  0.9984169 -0.4115774
## Rep3  0.9977751  0.9984169  1.0000000 -0.4115570
## Rep4 -0.4126249 -0.4115774 -0.4115570  1.0000000
```

```
## === Step 4: Peak calling with replicates ===
# We redo the previous step without the "bad" replicate
# Also, we force all replicates to agree in their peak calls
multi.model <- callPeaksReplicates(models[1:3], force.equal=TRUE, max.time=60)
```

```
## === Step 5: Export to genome browser ===
# Finally, we can export the results as BED file
exportMultivariate(multi.model, filename='your-file', what=c('peaks','reads'))
```

4.2 Task 2: Detecting differentially modified regions

chromstaR is extremely powerful in detecting differentially modified regions in two or more samples. The following example illustrates this on ChIP-seq data for H3K36me3 in 7 different human brain tissues. With 7 samples we can have $2^7 = 128$ combinatorial states, which can be readily interpreted as '0: all samples unmodified', '1-126: DMR' and '127: all samples modified'. Having several replicates for each sample makes it more complicated, but you get the idea ...

```
library(chromstaR)
```

```r
## === Step 1: Binning ===
# !!! Skip this step if you do not have package 'chromstaRExampleData' installed !!!
library(chromstaRExampleData)
# Let's get some example data with 3 replicates
bedfiles <- getExampleFilesBED()
# We use bin size 1000bp and chromosome 22 to keep the example quick
binned.data <- list()
for (bedfile in bedfiles) {
  binned.data[[basename(bedfile)]] <- bed2binned(bedfile, assembly='hg19',
                                                 binsize=1000, chromosomes='chr22')
}
```

```r
## === Step 2: Univariate peak calling ===
# We load the binned.data from step 1 (this is only necessary if step 1 was skipped)
data(differentialExample_binnedData)
# The univariate fit is obtained for each sample
models <- list()
for (i1 in 1:length(binned.data)) {
  message("Fitting model ", i1)
  models[[i1]] <- callPeaksUnivariate(binned.data[[i1]], ID=names(binned.data)[i1],
                                      max.time=60, verbosity=0)
}
```

```r
## === Step 3: Constructing the combinatorial states ===
# This step is only necessary if you have replicates for each sample.
# To ensure that replicates are treated as such, and not as independent
# samples, we have to construct the proper combinatorial states:

# First, we get all the tissues (we could specify them by hand, but we are lazy)
IDs <- names(binned.data)
tissues <- unlist(lapply(strsplit(IDs, '.Brain_|\\.'), '[[', 2))
print(tissues)

##  [1] "Angular_Gyrus"          "Angular_Gyrus"
##  [3] "Anterior_Caudate"       "Anterior_Caudate"
##  [5] "Cingulate_Gyrus"        "Cingulate_Gyrus"
##  [7] "Hippocampus_Middle"     "Hippocampus_Middle"
##  [9] "Inferior_Temporal_Lobe" "Inferior_Temporal_Lobe"
## [11] "Mid_Frontal_Lobe"       "Mid_Frontal_Lobe"
## [13] "Substantia_Nigra"       "Substantia_Nigra"

# Second, we obtain the combinatorial states
# Look up ?stateBrewer on how to use this function
states <- stateBrewer(statespec = paste0('r.', tissues))
# Third, we construct common states
common.states <- c(stateBrewer(statespec = paste0('1.', tissues)),
                   stateBrewer(statespec = paste0('0.', tissues)))
```

```r
## === Step 4: Multivariate peak calling ===
multi.model <- callPeaksMultivariate(models, use.states=states, eps=1, max.time=60)
```

```
## Extracting reads from modellist...  0.02s
## Getting combinatorial states...  0.08s
## Computing pre z-matrix...  0.01s
## Transfering values into z-matrix...  0.1s
## Computing inverse of correlation matrix...  4.52s
##
## Starting multivariate HMM
## Using the following combinatorial states, covering 86.8840636207703% of the bins:
## 0 16383 3072 16380 192 13116 16371 12 768 3264 12300 13308 3084 15375 3120 3840 4095
15420 16188 3075 16128 16191 3 48 960 3123 3315 3903 4092 12540 15363 15612 16179 15 51
60 63 195 204 207 240 243 252 255 771 780 783 816 819 828 831 963 972 975 1008 1011 1020
1023 3087 3132 3135 3267 3276 3279 3312 3324 3327 3843 3852 3855 3888 3891 3900 4032 4035
4044 4047 4080 4083 12288 12291 12303 12336 12339 12348 12351 12480 12483 12492 12495 12528
12531 12543 13056 13059 13068 13071 13104 13107 13119 13248 13251 13260 13263 13296 13299
13311 15360 15372 15408 15411 15423 15552 15555 15564 15567 15600 15603 15615 16131 16140
16143 16176 16320 16323 16332 16335 16368

## HMM: number of states = 128
## HMM: number of bins = 51304
## HMM: maximum number of iterations = none
## HMM: maximum running time = 60 sec
## HMM: epsilon = 1
## HMM: number of modifications = 14
##  Iteration              log(P)             dlog(P)    Time in sec
##          0                -inf                  -              0
## HMM: Precomputing densities ...
##  Iteration              log(P)             dlog(P)    Time in sec
##          0                -inf                  -             13
##          1     -1470427.292494                inf             20
##          2     -1463343.769468         7083.523026             27
##          3     -1463199.730689          144.038780             33
##          4     -1463181.468977           18.261712             40
##          5     -1463178.007584            3.461393             47
##          6     -1463176.926714            1.080870             54
##          7     -1463176.529079            0.397635             61
## HMM: Convergence reached!
## HMM: Recoding posteriors ...

## Transforming posteriors to 'per sample' representation ...  0.54s
## Calculating states from posteriors ...  0.01s
## Making segmentation ...  0.77s
```

```
## === Step 5: Export to genome browser ===
# Export only differential peaks by excluding the 'common.states'
exportMultivariate(multi.model, filename='your-file', what=c('peaks','reads'),
                   exclude.states=common.states)
```

4.3 Task 3: Finding combinatorial chromatin states

TODO: Introduction

```r
library(chromstaR)
```

```r
## === Step 1: Binning ===
# !!! Skip this step if you do not have package 'chromstaRExampleData' installed !!!
library(chromstaRExampleData)
# Let's get some example data with 3 replicates
bamfiles <- getExampleFilesBAM()
# We use bin size 1000bp and chromosome 22 to keep the example quick
binned.data <- list()
for (bamfile in bamfiles) {
  binned.data[[basename(bamfile)]] <- bam2binned(bamfile, bamindex=bamfile,
                                                 binsize=1000, chromosomes='chr12')
}
```

```r
## === Step 2: Univariate peak calling ===
# We load the binned.data from step 1 (this is only necessary if step 1 was skipped)
data(combinatorialExample_binnedData)
# The univariate fit is obtained for each sample
models <- list()
for (i1 in 1:length(binned.data)) {
  message("Fitting model ", i1)
  models[[i1]] <- callPeaksUnivariate(binned.data[[i1]], ID=names(binned.data)[i1],
                                      max.time=60, verbosity=0)
}
```

```r
## === Step 3: Constructing the combinatorial states ===
# This step is only necessary if you have replicates for each sample.
# To ensure that replicates are treated as such, and not as independent
# samples, we have to construct the proper combinatorial states:

# First, we get all the histone marks (we could specify them by hand, but we are lazy)
IDs <- names(binned.data)
marks <- unlist(lapply(strsplit(IDs, 'liver-|-BN'), '[[', 2))
print(marks)

##  [1] "H3K27me3" "H3K27me3" "H3K27me3" "H3K27me3" "H3K27me3" "H3K4me1"
##  [7] "H3K4me1"  "H3K4me1"  "H3K4me3"  "H3K4me3"  "H3K4me3"  "H4K20me1"
## [13] "H4K20me1" "H4K20me1"

# Second, we obtain the combinatorial states
# Look up ?stateBrewer on how to use this function
states <- stateBrewer(statespec = paste0('r.', marks))
```

```r
## === Step 4: Multivariate peak calling ===
multi.model <- callPeaksMultivariate(models, use.states=states, eps=1, max.time=60)

## Extracting reads from modellist...  0.02s
## Getting combinatorial states...  0.08s
## Computing pre z-matrix...  0.01s
```

```
## Transfering values into z-matrix...  0.09s
## Computing inverse of correlation matrix...  0.7s
##
## Starting multivariate HMM
## Using the following combinatorial states, covering 75.3879697319482% of the bins:
## 15872 0 455 7 511 16320 16327 448 15879 16376 504 16383 15928 56 63 15935
```

```
## Transforming posteriors to 'per sample' representation ...  0.02s
## Calculating states from posteriors ...  0.01s
## Making segmentation ...  2.67s
```

```
# Let's have a look at the results
multi.model$bins
```

```
##       [2]     H3K4me3
##       [3]     H3K4me3
##       [4]     H3K4me3
##       [5]
##       ...         ...
##   [46778]    H3K27me3
##   [46779]    H3K27me3
##   [46780]    H3K27me3
##   [46781]    H3K27me3
##   [46782]    H3K27me3
##   -------
##   seqinfo: 1 sequence from an unspecified genome
```

```
## === Step 5: Export to genome browser ===
# Export combinatorial states
exportMultivariate(multi.model, filename='your-file',
                   what=c('combstates','peaks','reads'))
```

5 Example workflows

6 FAQ

7 Session Info

```
sessionInfo()

## R version 3.1.0 (2014-04-10)
## Platform: x86_64-pc-linux-gnu (64-bit)
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=nl_NL.UTF-8        LC_COLLATE=en_US.UTF-8
##  [5] LC_MONETARY=nl_NL.UTF-8    LC_MESSAGES=en_US.UTF-8
##  [7] LC_PAPER=nl_NL.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=nl_NL.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] parallel  stats4    stats     graphics  grDevices utils     datasets
## [8] methods   base
##
## other attached packages:
##  [1] XVector_0.6.0            chromstaRExampleData_0.9
##  [3] chromstaR_0.9            reshape2_1.4.1
##  [5] ggplot2_1.0.1            GenomicRanges_1.18.4
##  [7] GenomeInfoDb_1.2.5       IRanges_2.0.1
##  [9] S4Vectors_0.4.0          BiocGenerics_0.12.1
## [11] knitr_1.10.5             devtools_1.8.0
```

```
##
## loaded via a namespace (and not attached):
##  [1] base64enc_0.1-2         BatchJobs_1.6
##  [3] BBmisc_1.9              BiocParallel_1.0.3
##  [5] Biostrings_2.34.1       bitops_1.0-6
##  [7] brew_1.0-6              BSgenome_1.34.1
##  [9] checkmate_1.5.3         codetools_0.2-11
## [11] colorspace_1.2-6        DBI_0.3.1
## [13] digest_0.6.8            evaluate_0.7
## [15] fail_1.2                foreach_1.4.2
## [17] formatR_1.2             GenomicAlignments_1.2.2
## [19] git2r_0.10.1            grid_3.1.0
## [21] gtable_0.1.2            highr_0.5
## [23] iterators_1.0.7         labeling_0.3
## [25] magrittr_1.5            MASS_7.3-40
## [27] memoise_0.2.1           munsell_0.4.2
## [29] plyr_1.8.2              proto_0.3-10
## [31] Rcpp_0.11.6             RCurl_1.95-4.6
## [33] Rsamtools_1.18.3        RSQLite_1.0.0
## [35] rtracklayer_1.26.3      rversions_1.0.0
## [37] scales_0.2.4            sendmailR_1.2-1
## [39] stringi_0.4-1           stringr_1.0.0
## [41] tools_3.1.0             XML_3.98-1.1
## [43] zlibbioc_1.12.0

warnings()

## NULL
```