

# Package ‘metagene’

November 26, 2021

**Version** 2.27.1

**Date** 2021-11-18

**Title** A package to produce metagene plots

**Author** Charles Joly Beuparlant

<charles.joly-beuparlant@crchul.ulaval.ca>, Fabien Claude Lamaze  
<fabien.lamaze.1@ulaval.ca>, Rawane Samb <rawane.samb.1@ulaval.ca>,  
Cedric Lippens <lippens.cedric@protonmail>,  
Astrid Louise Deschenes <Astrid-Louise.Deschenes@crchudequebec.ulaval.ca>  
and Arnaud Droit <arnaud.droit@crchuq.ulaval.ca>.

**Maintainer** Charles Joly Beuparlant <charles.joly-beuparlant@crchul.ulaval.ca>

**Description** This package produces metagene plots to compare the behavior of DNA-interacting proteins at selected groups of genes/features. Bam files are used to increase the resolution. Multiple combination of group of bam files and/or group of genomic regions can be compared in a single analysis. Bootstrapping analysis is used to compare the groups and locate regions with statistically different enrichment profiles.

**biocViews** ChIPSeq, Genetics, MultipleComparison, Coverage, Alignment, Sequencing

**License** Artistic-2.0 | file LICENSE

**LazyData** true

**BugReports** <https://github.com/CharlesJB/metagene/issues>

**VignetteBuilder** knitr

**Depends** R (>= 3.5.0), R6 (>= 2.0), GenomicRanges, BiocParallel

**Imports** rtracklayer, gplots, tools, GenomicAlignments, GenomeInfoDb, GenomicFeatures, IRanges, ggplot2, muStat, Rsamtools, matrixStats, purrr, data.table, magrittr, methods, utils, ensemblDb, EnsDb.Hsapiens.v86, stringr

**Suggests** BiocGenerics, similaRpeak, RUnit, knitr, BiocStyle, rmarkdown, similaRpeak

**RoxygenNote** 6.0.1

**git\_url** <https://git.bioconductor.org/packages/metagene>

**git\_branch** master

**git\_last\_commit** 432d1fd

**git\_last\_commit\_date** 2021-11-18

**Date/Publication** 2021-11-26

## R topics documented:

avoid_gaps_update . . . . .	2
Bam_Handler . . . . .	4
bed_file_filter . . . . .	6
exon_by_gene_with_observed_transcripts . . . . .	7
get_demo_bam_files . . . . .	8
get_demo_design . . . . .	8
get_demo_metagene . . . . .	9
get_demo_regions . . . . .	9
get_promoters_txdb . . . . .	10
metagene . . . . .	10
permutation_test . . . . .	13
plot_metagene . . . . .	14
promoters_hg18 . . . . .	15
promoters_hg19 . . . . .	16
promoters_mm10 . . . . .	16
promoters_mm9 . . . . .	17
write_bed_file_filter_result . . . . .	18

<b>Index</b>	<b>19</b>
--------------	-----------

---

avoid_gaps_update	<i>Is is a function designed to remove values <math>\leq</math> to 'gaps_threshold'. Nucleotides local and global positions, bins, size of regions/genes and exons will be recalculated. To use on metagene's table during RNA-seq analysis. Not made for ChIP-Seq analysis or to apply on matagene's data_frame. A similar function is implemented in produce_data_frame() with same arguments. The unique goal of this function is to allow permutation_test which match the plot created using avoid_gaps, bam_name and gaps_threshold arguments in the produce_data_frame function.</i>
-------------------	---

---

### Description

Is is a function designed to remove values  $\leq$  to 'gaps\_threshold'. Nucleotides local and global positions, bins, size of regions/genes and exons will be recalculated. To use on metagene's table during RNA-seq analysis. Not made for ChIP-Seq analysis or to apply on matagene's data\_frame. A similar function is implemented in produce\_data\_frame() with same arguments. The unique goal of this function is to allow permutation\_test which match the plot created using avoid\_gaps, bam\_name and gaps\_threshold arguments in the produce\_data\_frame function.

**Usage**

```
avoid_gaps_update(table, bam_name, gaps_threshold = 0)
```

**Arguments**

**table** A data.table from produce\_table(...) function of metagene.

**bam\_name** A reference bam\_name to allow the same removal (position in bam) of values for other bam file.

**gaps\_threshold** A threshold under which values will be removed.

**Value**

A data.table with values  $\leq$  to 'gaps\_threshold' removed

**Examples**

```
## Not run:
bam_files <- c(
  system.file("extdata/c_al4_945MLM_demo_sorted.bam", package="metagene"),
  system.file("extdata/c_al3_362PYX_demo_sorted.bam", package="metagene"),
  system.file("extdata/n_al4_310HII_demo_sorted.bam", package="metagene"),
  system.file("extdata/n_al3_588WMR_demo_sorted.bam", package="metagene")
)
region <- c(
  system.file("extdata/ENCFF355RXX_DPM1less.bed", package="metagene"),
  system.file("extdata/ENCFF355RXX_NDUFAB1less.bed", package="metagene"),
  system.file("extdata/ENCFF355RXX_SLC25A5less.bed", package="metagene")
)
mydesign <- matrix(c(1,1,0,0,0,0,1,1), ncol=2, byrow=FALSE)
mydesign <- cbind(c("c_al4_945MLM_demo_sorted.bam",
                  "c_al3_362PYX_demo_sorted.bam",
                  "n_al4_310HII_demo_sorted.bam",
                  "n_al3_588WMR_demo_sorted.bam"), mydesign)
colnames(mydesign) <- c('Samples', 'cyto', 'nucleo')
mydesign <- data.frame(mydesign)
mydesign[,2] <- as.numeric(mydesign[,2])-1
mydesign[,3] <- as.numeric(mydesign[,3])-1

mg <- metagene$new(regions = region, bam_files = bam_files,
                  assay = 'rnaseq')
mg$produce_table(flip_regions = FALSE, bin_count = 100,
               design = mydesign, normalization = 'RPM')
mg$produce_data_frame(avoid_gaps = TRUE,
                    bam_name = "c_al4_945MLM_demo_sorted",
                    gaps_threshold = 10)

mg$plot()
tab <- mg$get_table()
tab <- avoid_gaps_update(tab,
                        bam_name = 'c_al4_945MLM_demo_sorted', gaps_threshold = 10)
tab1 <- tab[which(tab0$design == "cyto"),]
tab2 <- tab[which(tab0$design == "nucleo"),]

library(similaRpeak)
```

```

perm_fun <- function(profile1, profile2) {
  sim <- similarity(profile1, profile2)
  sim[["metrics"]][["RATIO_NORMALIZED_INTERSECT"]]
}

ratio_normalized_intersect <-
  perm_fun(tab1[, .(moy=mean(value)), by=bin]$moy,
           tab2[, .(moy=mean(value)), by=bin]$moy)
ratio_normalized_intersect

permutation_results <- permutation_test(tab1, tab2, sample_size = 2,
                                       sample_count = 1000, FUN = perm_fun)

hist(permutation_results,
     main="ratio_normalized_intersect (1=total overlapping area)")
abline(v=ratio_normalized_intersect, col = 'red')
sum(ratio_normalized_intersect >= permutation_results) /
  length(permutation_results)

## End(Not run)

```

---

Bam\_Handler

*A class to manage BAM files.*


---

## Description

This class will allow to load, convert and normalize alignments and regions files/data.

## Usage

```
Bam_Handler
```

## Format

A BAM manager

## Value

Bam\_Handler\$new returns a Bam\_Handler object which contains coverage related information for every BAM files.

## Constructor

```
bh <- Bam_Handler$new(bam_files, cores = SerialParam())
```

**bam\_files** A vector of BAM filenames. The BAM files must be indexed. i.e.: if a file is named file.bam, there must be a file named file.bam.bai or file.bai in the same directory.

**cores** The number of cores available to parallelize the analysis. Either a positive integer or a BiocParallelParam. Default: SerialParam().

**paired\_end** If TRUE, metagene will deal with paired-end data. If FALSE, single-end data are expected

Bam\_Handler\$new returns a Bam\_Handler object that contains and manages BAM files. Coverage related information as alignment count can be obtain by using this object.

## Methods

bh\$get\_aligned\_count(bam\_file)

**bam\_file** The name of the BAM file.

bg\$get\_bam\_name(bam\_file)

**bam\_file** The name of the BAM file.

bh\$get\_rpm\_coefficient(bam\_file)

**bam\_file** The name of the BAM file.

bh\$index\_bam\_files(bam\_files)

**bam\_files** A vector of BAM filenames.

bh\$get\_bam\_files()

bh\$get\_coverage(bam\_file, regions) force\_seqlevels = FALSE)

**bam\_file** The name of the BAM file.

**regions** A not empty GRanges object.

**force\_seqlevels** If TRUE, Remove regions that are not found in bam file header. Default: FALSE. TRUE and FALSE respectively correspond to pruning.mode = "coarse" and "error" in ?seqinfo.

bh\$get\_normalized\_coverage(bam\_file, regions) force\_seqlevels = FALSE)

**bam\_file** The name of the BAM file.

**regions** A not empty GRanges object.

**force\_seqlevels** If TRUE, Remove regions that are not found in bam file header. Default: FALSE. TRUE and FALSE respectively correspond to pruning.mode = "coarse" and "error" in ?seqinfo.

bh\$get\_noise\_ratio(chip\_bam\_file, input\_bam\_file)

**chip\_bam\_file** The path to the chip bam file.

**input\_bam\_file** The path to the input (control) bam file.

## Examples

```
bam_file <- get_demo_bam_files()[1]
bh <- metagene::Bam_Handler$new(bam_files = bam_file)
bh$get_aligned_count(bam_file)
```

---

bed_file_filter	<i>Extract a list of ranges defined by the bed_file_content_gr argument from the ebgwot GRangesList. Equivalent to the exonsByOverlaps of GenomicFeatures.</i>
-----------------	--

---

## Description

Extract a list of ranges defined by the bed\_file\_content\_gr argument from the ebgwot GRangesList. Equivalent to the exonsByOverlaps of GenomicFeatures.

## Usage

```
bed_file_filter(ebgwot, bed_file_content_gr, reduce = TRUE)
```

## Arguments

ebgwot	A GRangesList object provided by the exon_by_gene_with_observed_transcripts function.
bed_file_content_gr	A GRanges object containing ranges of interest.
reduce	If the returned GRanges object will be reduce or not.

## Value

A GRanges object that contains exons by genes selected.

## Examples

```
## Not run:
require(EnsDb.Hsapiens.v86)
edb <- EnsDb.Hsapiens.v86
quantification_files <- 'file_path'
ebgwot <- exon_by_gene_with_observed_transcripts(edb,
                                                quantification_files)
bed_file_content_gr <- GRanges("chr16", ranges = IRanges(start=23581002,
                                                         end=23596356))
bed_file_filter(ebgwot, bed_file_content_gr)

## End(Not run)
```

---

`exon_by_gene_with_observed_transcripts`

*Extract exons by genes for which data are available in quantification files*

---

## Description

Extract exons by genes for which data are available in quantification files

## Usage

```
exon_by_gene_with_observed_transcripts(adb, quantification_files)
```

## Arguments

`adb` A valid EnsDb object.  
`quantification_files` the quantification files. A vector of pathes.

## Value

A GRangesList object containing exons by genes for which data are available in quantification files.

## Examples

```
## Not run:  
require(EnsDb.Hsapiens.v86)  
edb <- EnsDb.Hsapiens.v86  
quantification_files <- 'file_path'  
ebgwot <- exon_by_gene_with_observed_transcripts(edb,  
                                                quantification_files)  
bed_file_content_gr <- GRanges("chr16", ranges = IRanges(start=23581002,  
                                                         end=23596356))  
bed_file_filter(ebgwot, bed_file_content_gr)  
  
## End(Not run)
```

get\_demo\_bam\_files      *Get BAM filenames for demo*

---

**Description**

Get BAM filenames for demo

**Usage**

```
get_demo_bam_files()
```

**Value**

A vector of BAM filenames

**Examples**

```
bam_files <- get_demo_bam_files()
```

---

get\_demo\_design      *Get a demo design object*

---

**Description**

Get a demo design object

**Usage**

```
get_demo_design()
```

**Value**

A data.frame corresponding to a valid design.

**Examples**

```
mg <- get_demo_design()
```



---

`get_demo_metagene`      *Get a demo metagene object*

---

**Description**

Get a demo metagene object

**Usage**

```
get_demo_metagene()
```

**Value**

A metagene object

**Examples**

```
mg <- get_demo_metagene()
```

---

`get_demo_regions`      *Get regions filenames for demo*

---

**Description**

Get regions filenames for demo

**Usage**

```
get_demo_regions()
```

**Value**

A vector of regions filenames

**Examples**

```
regions <- get_demo_regions()
```

get\_promoters\_txdb      *Extract Entrez genes promoters from TxDb object.*

---

**Description**

Extract Entrez genes promoters from TxDb object.

**Usage**

```
get_promoters_txdb(txdb, upstream = 1000, downstream = 1000)
```

**Arguments**

txdb	A valid TxDb object.
upstream	The number of nucleotides upstream of TSS.
downstream	The number of nucleotides downstream of TSS.

**Value**

A GRanges object that contains the promoters infos.

**Examples**

```
## Not run:  
# require(TxDb.Hsapiens.UCSC.hg19.knownGene)  
txdb <- TxDb.Hsapiens.UCSC.hg19.knownGene  
promoters_hg19 <- get_promoters_txdb(txdb)  
  
## End(Not run)
```

---

metagene      *A class to manage metagene analysis.*

---

**Description**

This class will allow to load, convert and normalize alignments and regions files/data. Once the data is ready, the user can then chose to produce metagene plots on the data (or a subset of the data).

**Usage**

```
metagene
```

**Format**

A metagene experiment manager

**Value**

metagene\$new returns a metagene object which contains the normalized coverage values for every regions and for every BAM files.

**Constructor**

```
mg <- metagene$new(regions, bam_files, padding_size = 0, cores = SerialParam(), verbose = FALSE, force_seqlevels = FALSE, paired_end = FALSE, assay = 'chipseq')
```

**regions** Either a vector of BED, narrowPeak or broadPeak filenames, a GRanges object or a GRangesList object.

**bam\_files** A vector of BAM filenames. The BAM files must be indexed. i.e.: if a file is named file.bam, there must be a file named file.bam.bai or file.bai in the same directory.

**padding\_size** The regions will be extended on each side by the value of this parameter. The padding\_size must be a non-negative integer. Default = 0.

**cores** The number of cores available to parallelize the analysis. Either a positive integer or a BiocParallelParam. Default: SerialParam().

**verbose** Print progression of the analysis. A logical constant. Default: FALSE.

**force\_seqlevels** If TRUE, Remove regions that are not found in bam file header. Default: FALSE. TRUE and FALSE respectively correspond to pruning.mode = "coarse" and "error" in ?seqinfo.

**paired\_end** If TRUE, metagene will deal with paired-ended data. If FALSE, single-ended data are expected. Default: FALSE

**assay** 'chipseq' or 'rnaseq', the two available options. Default: 'chipseq'

metagene\$new returns a metagene object that contains the coverages for every BAM files in the regions from the regions param.

**Methods**

```
mg$plot(region_names = NULL, design_names = NULL, title = NULL, x_label = NULL)
```

**region\_names** The names of the regions to extract. If NULL, all the regions are returned. Default: NULL.

**design\_names** The names of the experiments to extract. If a design was added to the metagene object, design\_names correspond to the column names in the design, otherwise design\_names corresponds to the BAM name or the BAM filename. If NULL, all the experiments are returned. Default: NULL.

**title** A title to add to the graph. If NULL, will be automatically created. Default: NULL

**x\_label** X-axis label to add to the metagene plot. If NULL, metagene will use generic label. Default: NULL.

```
mg$produce_table(design, bin_count, noise_removal, normalization, flip_regions, bin_size = NULL)
```

**design** A data.frame that describe to experiment to plot. see plot function for more details. NA can be used keep previous design value. Default: NA.

**bin\_count** The number of bin to create. NA can be used to keep previous bin\_count value. A bin\_count value of 100 will be used if no value is specified. Default: NA.

**noise\_removal** The algorithm to use to remove control(s). Possible values are NA, NULL or "NCIS". By default, value is NULL. Use NA keep previous noise\_removal value (i.e. if produce\_table was called before). See Liand and Keles 2012 for the NCIS algorithm.

**normalization** The algorithm to use to normalize samples. Possible default, value is NULL and no normalization will be performed. Use NA keep previous normalization value (i.e. if produce\_table was called before).

**flip\_regions** Should regions on negative strand be flip\_regions? Default: FALSE.

**bin\_size** Deprecated.

```
mg$produce_data_frame(alpha = 0.05, sample_count = 1000, avoid_gaps = FALSE, gaps_threshold = 0)
```

**alpha** The range of the estimation to be shown with the ribbon.  $1 - \alpha / 2$  and  $\alpha / 2$  will be used. Default: 0.05.

**sample\_count** The number of draw to do in the bootstrap calculation. Default: 1000.

**avoid\_gaps** Provide the possibility to remove values = 0 and refit the data\_frame for this suppression. Default : FALSE.

**gaps\_threshold** It works with avoid\_gaps argument. It lets to remove values  $\leq$  at gaps\_threshold. Default : 0.

```
mg$get_params()
```

```
mg$get_design()
```

```
mg$get_regions(region_names = NULL)
```

**region\_names** The names of the regions to extract. If NULL, all the regions are returned. Default: NULL.

```
mg$get_table = function()
```

```
mg$get_matrices = function()
```

```
mg$get_data_frame(region_names = NULL, design_names = NULL)
```

**region\_names** The names of the regions to extract. If NULL, all the regions are returned. Default: NULL.

**design\_names** The names of the experiments to extract. If a design was added to the metagene object, design\_names correspond to the column names in the design, otherwise design\_names corresponds to the BAM name or the BAM filename. If NULL, all the experiments are returned. Default: NULL.

```
get_plot = function()
```

```
get_raw_coverages = function(filenamees)
```

**filenames** The name of the file to extract raw coverages. Can be the filename with the extension of the name of the bam file (if a named bam files was used during the creation of the metagene object). If NULL, returns the coverage of every bam files. Default: NULL.

```
get_normalized_coverages = function(filenamees)
```

**filenames** The name of the file to extract normalized coverages (in RPM). Can be the filename with the extension of the name of the bam file (if a named bam files was used during the creation of the metagene object). If NULL, returns the coverage every bam files. Default: NULL.

```
mg$export(bam_file, region, file)
```

**bam\_file** The name of the bam file to export.

**region** The name of the region to export.

**file** The name of the ouput file.

```
mg$add_design(design = NULL, check_bam_files = FALSE)
```

**design** A data.frame that describe to experiment to plot. See plot function for more details. NA can be used keep previous design value. Default: NA.

**check\_bam\_files** Force check that all the bam files from the first columns of the design are present in current metagene object. Default: FALSE

```
mg$unflip_regions()
```

```
mg$flip_regions()
```

```
mg$unflip_regions()
```

## Examples

```
region <- get_demo_regions()[1]
bam_file <- get_demo_bam_files()[1]
mg <- metagene$new(regions = region, bam_files = bam_file)
## Not run:
df <- metagene$plot()

## End(Not run)
```

---

permutation\_test

*Perform a permutation test on 2 tables*

---

## Description

The goal of this function is to calculate the values of a test performed by FUN after each of sample\_count permutations.

## Usage

```
permutation_test(table1, table2, sample_size, sample_count, FUN, ...)
```

**Arguments**

table1	The first table.
table2	The second table.
sample_size	The number of element to draw for each table.
sample_count	The number of permutations.
FUN	The function to use to compare the 2 table. First two params must be numeric vector and the return must be a single numeric value.
...	Extra param for FUN.

**Details**

Each round of the permutation test, two new matrices will be randomly sampled from using the combination of the two original tables. The means of each columns will be calculated to produce the vectors that will be sent FUN.

**Value**

A vector of numeric corresponding to the result of each permutation.

**Examples**

```
## Not run:
# Get some tables
mg <- get_demo_metagene()
mg$produce_table()
tab <- mg$get_table()
tab <- tab[which(tab$region == "list1"),]
tab1 <- tab[which(tab$design == "align1_rep1"),]
tab2 <- tab[which(tab$design == "align2_rep2"),]

# Perform permutation test
sample_size <- min(nrow(tab1), nrow(tab2))
FUN = function(a, b) { mean(a) - mean(b) } # Dummy function for demo purpose
# A sample_count >= 1000 should be used in a real analysis
permutation_results <- permutation_test(m1, m2, sample_size = sample_size,
                                       sample_count = 10, FUN = FUN)

## End(Not run)
```

---

plot\_metagene

---

*Produce a metagene plot*


---

**Description**

Produce a metagene plot

**Usage**

```
plot_metagene(df)
```

**Arguments**

`df` a data.frame obtained with the `get_data_frame` function. Must have the following columns: "region", "design", "bin", "value", "qinf" and "qsup".

**Value**

A 'ggplot' object.

**Examples**

```
region <- get_demo_regions()[1]
bam_file <- get_demo_bam_files()[1]
mg <- metagene$new(regions = region, bam_files = bam_file)
mg$produce_data_frame()
df <- mg$get_data_frame()
p <- plot_metagene(df)
```

---

promoters\_hg18

*Promoters regions of hg18 Entrez genes.*

---

**Description**

Each regions have a width of 2000 nucleotide centered at the transcription start site.

**Usage**

```
promoters_hg18
```

**Format**

A GRanges object with 19742 ranges.

**Value**

A GRanges.

**See Also**

[get\\_promoters\\_txdb](#)

**Examples**

```
data(promoters_hg18)
```

---

promoters_hg19	<i>Promoters regions of hg19 Entrez genes.</i>
----------------	--

---

**Description**

Each regions have a width of 2000 nucleotide centered at the transcription start site.

**Usage**

```
promoters_hg19
```

**Format**

A GRanges object with 23056 ranges.

**Value**

A GRanges.

**See Also**

[get\\_promoters\\_txdb](#)

**Examples**

```
data(promoters_hg19)
```

---

promoters_mm10	<i>Promoters regions of mm10 Entrez genes.</i>
----------------	--

---

**Description**

Each regions have a width of 2000 nucleotide centered at the transcription start site.

**Usage**

```
promoters_mm10
```

**Format**

A GRanges object with 23653 ranges.

**Value**

A GRanges.



**See Also**

[get\\_promoters\\_txdb](#)

**Examples**

```
data(promoters_mm10)
```

---

promoters\_mm9

*Promoters regions of mm9 Entrez genes.*

---

**Description**

Each regions have a width of 2000 nucleotide centered at the transcription start site.

**Usage**

```
promoters_mm9
```

**Format**

A GRanges object with 21677 ranges.

**Value**

A GRanges.

**See Also**

[get\\_promoters\\_txdb](#)

**Examples**

```
data(promoters_mm9)
```

---

write\_bed\_file\_filter\_result

*Transforms the bed\_file\_filter function output into a file.BED readable by metagene.*

---

### Description

Transforms the bed\_file\_filter function output into a file.BED readable by metagene.

### Usage

```
write_bed_file_filter_result(bed_file_filter_result, file = "file_name",
  path = "./")
```

### Arguments

bed_file_filter_result	A GRanges object : the output of bed_file_filter function.
file	the name of the output file without the extension
path	The path where the function will write the file

### Value

output of write function

### Examples

```
## Not run:
require(EnsDb.Hsapiens.v86)
edb <- EnsDb.Hsapiens.v86
quantification_files <- 'file_path'
ebgwot <- exon_by_gene_with_observed_transcripts(edb,
  quantification_files)
bed_file_content_gr <- GRanges("chr16",ranges = IRanges(start=23581002,
  end=23596356))

bffr <- bed_file_filter(ebgwot, bed_file_content_gr)
write_bed_file_filter_result(bffr, file='test','./')

## End(Not run)
```

# Index

## \* datasets

- Bam\_Handler, 4
- metagene, 10
- promoters\_hg18, 15
- promoters\_hg19, 16
- promoters\_mm10, 16
- promoters\_mm9, 17

avoid\_gaps\_update, 2

Bam\_Handler, 4

bed\_file\_filter, 6

exon\_by\_gene\_with\_observed\_transcripts,  
7

get\_demo\_bam\_files, 8

get\_demo\_design, 8

get\_demo\_metagene, 9

get\_demo\_regions, 9

get\_promoters\_txdb, 10, 15–17

metagene, 10

permutation\_test, 13

plot\_metagene, 14

- promoters\_hg18, 15
- promoters\_hg19, 16
- promoters\_mm10, 16
- promoters\_mm9, 17

write\_bed\_file\_filter\_result, 18