Package 'nullranges'

November 12, 2026

Title Generation of null ranges via bootstrapping or covariate matching

Version 1.18.0

Description Modular package for generation of sets of ranges representing the null hypothesis. These can take the form of bootstrap samples of ranges (using the block bootstrap framework of Bickel et al 2010), or sets of control ranges that are matched across one or more covariates. nullranges is designed to be inter-operable with other packages for analysis of genomic overlap enrichment, including the plyranges Bioconductor package.

Suggests testthat, knitr, rmarkdown, ks, DNAcopy, RcppHMM, AnnotationHub, ExperimentHub, GenomeInfoDb, nullrangesData, ensembldb, EnsDb.Hsapiens.v86, BSgenome.Hsapiens.UCSC.hg38, patchwork, plotgardener, dplyr, magrittr, tidyr, cobalt, DiagrammeR, MatchIt, mariner

Imports stats, IRanges, GenomicRanges, Seqinfo, methods, rlang, S4Vectors, scales, InteractionSet, ggplot2, grDevices, plyranges, data.table, progress, ggridges

biocViews Visualization, GeneSetEnrichment, FunctionalGenomics, Epigenetics, GeneRegulation, GeneTarget, GenomeAnnotation, Annotation, GenomeWideAssociation, HistoneModification, ChIPSeq, ATACSeq, DNaseSeq, RNASeq, HiddenMarkovModel

URL https://nullranges.github.io/nullranges, https://github.com/nullranges/nullranges

BugReports https://support.bioconductor.org/tag/nullranges/

License GPL-3

Encoding UTF-8

LazyData true

VignetteBuilder knitr

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

git_url https://git.bioconductor.org/packages/nullranges

git_branch RELEASE_3_23

git_last_commit b4dd7bf

git_last_commit_date 2026-04-28

Repository Bioconductor 3.23

Date/Publication 2026-11-11

Author Michael Love [aut, cre] (ORCID:
<https://orcid.org/0000-0001-8401-0545>),
Wancen Mu [aut] (ORCID: <https://orcid.org/0000-0002-5061-7581>),
Eric Davis [aut] (ORCID: <https://orcid.org/0000-0003-4051-3217>),
Douglas Phanstiel [aut] (ORCID:
<https://orcid.org/0000-0003-2123-0051>),
Stuart Lee [aut] (ORCID: <https://orcid.org/0000-0003-1179-8436>),
Mikhail Dozmorov [ctb],
Tim Triche [ctb],
CZI [fnd]

Maintainer Michael Love <michaelisaiahlove@gmail.com>

Contents

bootRanges . . . 3
BootRanges-class . . . 4
combnCov . . . 5
covariates . . . 5
focal . . . 6
indices . . . 7
makeExampleMatchedDataSet . . . 8
matched . . . 9
Matched-class . . . 10
matchedData . . . 11
MatchedDataFrame-class . . . 12
MatchedGInteractions-class . . . 14
MatchedGRanges-class . . . 16
matchitToMatched . . . 18
matchRanges . . . 19
method . . . 22
names.MatchedOverview . . . 23
oneRegionSegment . . . 24
overview . . . 24
plotCovariate . . . 25
plotPropensity . . . 27
plotSegment . . . 28
pool . . . 29
reduceSegment . . . 30

segmentDensity 31
unmatched 32
withReplacement 33

Index **34**

`bootRanges` *Block bootstrap for genomic ranges*

Description

Performs a block bootstrap, optionally with respect to a genome segmentation. Returns a `bootRanges` object, which is a GRanges object with all the ranges from the bootstrap iterations concatenated.

Usage

```
bootRanges(
  y,
  blockLength,
  R = 1,
  seg = NULL,
  exclude = NULL,
  excludeOption = c("drop", "trim"),
  proportionLength = TRUE,
  type = c("bootstrap", "permute"),
  withinChrom = FALSE,
  storeBlockLength = FALSE
)
```

Arguments

`y`	the GRanges to bootstrap sample
`blockLength`	the length of the blocks (for proportional blocks, this is the maximal length of a block)
`R`	the number of bootstrap samples to generate
`seg`	the segmentation GRanges, with a column ("state") indicating segmentation state (optional)
`exclude`	the GRanges of excluded regions (optional)
`excludeOption`	whether to `"drop"` or `"trim"` bootstrap ranges that overlap a excluded region
`proportionLength`	for the segmented block bootstrap, whether to use scaled block lengths (scaling by the proportion of the segmentation state out of the total genome length). That is, the resulting blocks will be of size less than `blockLength`
`type`	the type of null generation (un-segmented bootstrap only)
`withinChrom`	whether to re-sample (bootstrap) ranges across chromosomes (default) or only within chromosomes (un-segmented bootstrap only)
`storeBlockLength`	whether to save blockLength as a metadata column

Details

Note that this function requires input ranges have associated `seqlengths`, and that these must not be shorter than `blockLength`. See `Seqinfo`, `seqlevels`, and `keepStandardChromosomes` functions and their use in the Quick Start section of the vignette.

Value

a BootRanges (GRanges object) with the bootstrapped ranges, where iteration and block length are recorded as metadata columns

References

bootRanges manuscript:

Wancen Mu, Eric S. Davis, Stuart Lee, Mikhail G. Dozmorov, Douglas H. Phanstiel, Michael I. Love. 2023. "bootRanges: Flexible generation of null sets of genomic ranges for hypothesis testing." Bioinformatics. doi: 10.1093/bioinformatics/btad190

Original method describing the segmented block bootstrap for genomic features:

Bickel, Peter J., Nathan Boley, James B. Brown, Haiyan Huang, and Nancy R. Zhang. 2010. "Subsampling Methods for Genomic Inference." The Annals of Applied Statistics 4 (4): 1660–97. doi: 10.1214/10-AOAS363

Examples

```
set.seed(1)
library(GenomicRanges)
gr <- GRanges("chr1", IRanges(0:4 * 10 + 1, width=5),
              seqlengths=c(chr1=50))
br <- bootRanges(gr, blockLength=10)
```

`BootRanges-class` *BootRanges object*

Description

This class extends the GRanges class. As produced by bootRanges, it will have an `iter` column indicating the iterations of the bootstrap, and a optionally a `blockLength` column indicating the blockLength parameter.

combnCov *Function for creating combinations of covariates*

Description

Function for creating combinations of covariates

Usage

```
combnCov(x, ...)

## S4 method for signature 'character'
combnCov(x)
```

Arguments

x	Character vector of covariates to combine.
...	Additional arguments.

Value

Returns a character vector of formulae combinations.

Examples

```
combnCov(x = c('a', 'b', 'c'))
```

covariates *Get covariates from a Matched object*

Description

Get covariates from a Matched object

Usage

```
covariates(x, ...)

## S4 method for signature 'Matched'
covariates(x, ...)
```

Arguments

x	Matched object.
...	Additional arguments.

Value

A character vector of covariates

Examples

```
set.seed(123)
mdf <- makeExampleMatchedDataSet(matched = TRUE)
covariates(mdf)
```

`focal` *Get focal set from a Matched object*

Description

Get focal set from a Matched object

Usage

```
focal(x, ...)

## S4 method for signature 'MDF_OR_MGR_OR_MGI'
focal(x, ...)
```

Arguments

`x`	A `MatchedDataFrame`, `MatchedGRanges`, or `MatchedGInteractions` object.
`...`	Additional options.

Value

An object of the same class as `x` representing the focal set.

Examples

```
set.seed(123)
x <- makeExampleMatchedDataSet(matched = TRUE)
focal(x)
```

`indices` *Get indices of matched set*

Description

Extracts the indices of a specified matched set from a Matched object.

Usage

```
indices(x, set = "matched", ...)

## S4 method for signature 'Matched'
indices(x, set)
```

Arguments

`x`	Matched object.
`set`	A character string describing from which set to extract indices. can be one of 'focal', 'matched', 'pool', or 'unmatched'.
`...`	Additional arguments.

Details

Indices from 'focal' come from the `focal` set of `matchRanges()` used to construct the `Matched` object while indices from 'matched', 'pool', and 'unmatched' come from the `pool` set. Default returns the 'matched' indices from the `pool` set.

Value

An integer vector corresponding to the indices in the `focal` or `pool` which comprise the "focal" or c("matched", "pool", "unmatched") sets.

Examples

```
set.seed(123)
mdf <- makeExampleMatchedDataSet(matched = TRUE)
head(indices(mdf))

head(indices(mdf, set = 'focal'))
head(indices(mdf, set = 'pool'))
head(indices(mdf, set = 'matched'))
head(indices(mdf, set = 'unmatched'))
```

`makeExampleMatchedDataSet` *Function for generating an example matchRanges or Matched dataset*

Description

This function will generate an example dataset as either 1) input for `matchRanges()` (when `matched = FALSE`) or 2) a Matched Object (when `matched = TRUE`).

Usage

```
makeExampleMatchedDataSet(
  type = "DataFrame",
  matched = FALSE,
  method = "rejection",
  replace = FALSE,
  ...
)

## S4 method for signature
## 'character_OR_missing,
##   logical_OR_missing,
##   character_OR_missing,
##   logical_OR_missing'
makeExampleMatchedDataSet(type, matched, method, replace)
```

Arguments

`type`	Character designating which type of dataset to make. options are one of 'data.frame', 'data.table', 'DataFrame', 'GRanges', or 'GInteractions'.
`matched`	TRUE/FALSE designating whether to generate a Matched dataset (`matched = TRUE`) or an input dataset for `matchRanges()` (`matched = FALSE`).
`method`	Character describing which matching method to use. Supported options are either 'nearest', 'rejection', or 'stratified'.
`replace`	TRUE/FALSE describing whether to select matches with or without replacement.
`...`	Additional arguments

Details

When `matched = FALSE`, the data returned contains 3 different features that can be subset to perform matching.

Value

Returns an example Matched dataset or an example dataset for input to `matchRanges()`.

Examples

```
## Make examples for matchRanges() (i.e matched = FALSE)
set.seed(123)
makeExampleMatchedDataSet()
head(makeExampleMatchedDataSet(type = 'data.frame', matched = FALSE))
makeExampleMatchedDataSet(type = 'data.table', matched = FALSE)
makeExampleMatchedDataSet(type = 'DataFrame', matched = FALSE)
makeExampleMatchedDataSet(type = 'GRanges', matched = FALSE)
makeExampleMatchedDataSet(type = 'GInteractions', matched = FALSE)

## Make Matched class examples (i.e. matched = TRUE)
set.seed(123)
makeExampleMatchedDataSet(matched = TRUE)
makeExampleMatchedDataSet(type = 'DataFrame', matched = TRUE,
                          method = 'rejection',
                          replace = FALSE)
makeExampleMatchedDataSet(type = 'GRanges', matched = TRUE,
                          method = 'rejection',
                          replace = FALSE)
# throwing error (April 2023)
#makeExampleMatchedDataSet(type = 'GInteractions', matched = TRUE,
#                           method = 'rejection',
#                           replace = FALSE)
```

`matched` *Get matched set from a Matched object*

Description

Get matched set from a Matched object

Usage

```
matched(x, ...)

## S4 method for signature 'MDF_OR_MGR_OR_MGI'
matched(x, ...)
```

Arguments

`x`	A `MatchedDataFrame`, `MatchedGRanges`, or `MatchedGInteractions` object.
`...`	Additional options.

Value

An object of the same class as `x` representing the matched set.

Examples

```
set.seed(123)
x <- makeExampleMatchedDataSet(matched = TRUE)
matched(x)
```

`Matched-class` *Matched objects*

Description

The Matched class is a container for attributes of covariate-matched data resulting from `matchRanges()`.

Slots

`matchedData` A `data.table` with matched data

`matchedIndex` An integer vector corresponding to the indices in the `pool` which comprise the `matched` set.

`covar` A character vector describing the covariates used for matching.

`method` Character describing replacement method used for matching.

`replace` TRUE/FALSE describing if matching was done with or without replacement.

Accessor methods for Matched Class

Functions that get data from Matched subclasses (`x`) such as MatchedDataFrame, MatchedGRanges, and MatchedGInteractions are listed below:

- `matchedData(x)`: Get matched data from a Matched object
- `covariates(x)`: Get covariates from a Matched object
- `method(x)`: Get matching method used for Matched object
- `withReplacement(x)`: Get replace method
- `indices(x, set)`: Get indices of matched set

For more detail check the help pages for these functions.

See Also

matchedData, covariates, method, withReplacement, indices

Examples

```
## Make Matched example
set.seed(123)
x <- makeExampleMatchedDataSet(matched = TRUE)
## Accessor functions for Matched class
matchedData(x)
covariates(x)
method(x)
withReplacement(x)
head(indices(x, set = 'matched'))
```

`matchedData` *Get matched data from a Matched object*

Description

Get matched data from a Matched object

Usage

```
matchedData(x, ...)

## S4 method for signature 'Matched'
matchedData(x, ...)
```

Arguments

`x`	Matched object.
`...`	Additional arguments.

Value

A `data.table` with matched data.

Examples

```
set.seed(123)
mdf <- makeExampleMatchedDataSet(matched = TRUE)
matchedData(mdf)
```

MatchedDataFrame-class *MatchedDataFrame objects*

Description

The MatchedDataFrame class is a subclass of both Matched and DFrame. Therefore, it contains slots and methods for both of these classes.

Details

The MatchedDataFrame class uses a delegate object during initialization to assign its DFrame slots. MatchedDataFrame behaves as a DataFrame but also includes additional Matched object functionality (see ?Matched). For more information about DataFrame see ?S4Vectors::DataFrame.

Slots

`focal` A DataFrame object containing the focal data to match.

`pool` A DataFrame object containing the pool from which to select matches.

`delegate` A DataFrame object used to initialize DataFrame-specific slots. `matchRanges()` assigns the matched set to the slot.

`matchedData` A `data.table` with matched data

`matchedIndex` An integer vector corresponding to the indices in the `pool` which comprise the `matched` set.

`covar` A character vector describing the covariates used for matching.

`method` Character describing replacement method used for matching.

`replace` TRUE/FALSE describing if matching was done with or without replacement.

`rownames rownames(delegate)`

`nrows nrows(delegate)`

`listData as.list(delegate)`

`elementType elementType(delegate)`

`elementMetadata elementMetadata(delegate)`

`metadata metadata(delegate)`

Accessor methods for Matched Class

Functions that get data from Matched subclasses (`x`) such as MatchedDataFrame, MatchedGRanges, and MatchedGInteractions are listed below:

- `matchedData(x)`: Get matched data from a Matched object
- `covariates(x)`: Get covariates from a Matched object
- `method(x)`: Get matching method used for Matched object
- `withReplacement(x)`: Get replace method
- `indices(x, set)`: Get indices of matched set

For more detail check the help pages for these functions.

Accessor methods for Matched subclass objects

Additional functions that get data from Matched subclasses (x) such as MatchedDataFrame, MatchedGRanges, and MatchedGInteractions include:

- `focal(x)`: Get focal set from a Matched object
- `pool(x)`: Get pool set from a Matched object
- `matched(x)`: Get matched set from a Matched object
- `unmatched(x)`: Get unmatched set from a Matched object

For more detail check the help pages for these functions.

See Also

S4Vectors::DataFrame

matchedData, covariates, method, withReplacement, indices

focal, pool, matched, unmatched

Examples

```
## Constructing MatchedDataFrame with matchRanges
## data.frame
set.seed(123)
x <- makeExampleMatchedDataSet(type = "data.frame")
mx <- matchRanges(
  focal = x[x$feature1, ],
  pool = x[!x$feature1, ],
  covar = ~ feature2 + feature3,
  method = "rejection",
  replace = FALSE
)
class(mx)

## data.table
set.seed(123)
x <- makeExampleMatchedDataSet(type = "data.table")
mx <- matchRanges(
  focal = x[x$feature1],
  pool = x[!x$feature1],
  covar = ~ feature2 + feature3,
  method = "rejection",
  replace = FALSE
)
class(mx)

## DataFrame
set.seed(123)
x <- makeExampleMatchedDataSet(type = "DataFrame")
mx <- matchRanges(
  focal = x[x$feature1, ],
  pool = x[!x$feature1, ],
```

```
  covar = ~ feature2 + feature3,
  method = "rejection",
  replace = FALSE
)
class(mx)

## Make MatchedDataFrame example
set.seed(123)
x <- makeExampleMatchedDataSet(type = "DataFrame", matched = TRUE)
## Accessor functions for Matched class
matchedData(x)
covariates(x)
method(x)
withReplacement(x)
head(indices(x, set = 'matched'))

## Accessor functions for Matched subclasses
focal(x)
pool(x)
matched(x)
unmatched(x)
```

`MatchedGInteractions-class` *MatchedGInteractions objects*

Description

The `MatchedGInteractions` class is a subclass of both `Matched` and `GInteractions`. Therefore, it contains slots and methods for both of these classes.

Details

The `MatchedGInteractions` class uses a delegate object during initialization to assign its `GInteractions` slots. `MatchedGInteractions` behaves as a `GInteractions` but also includes additional `Matched` object functionality (see `?Matched`). For more information about `GInteractions` see `?InteractionSet::GInteractions`.

Slots

`focal` A `GInteractions` object containing the focal data to match.

`pool` A `GInteractions` object containing the pool from which to select matches.

`delegate` A `GInteractions` object used to initialize `GInteractions`-specific slots. `matchRanges()` assigns the matched set to the slot.

`matchedData` A `data.table` with matched data

`matchedIndex` An integer vector corresponding to the indices in the `pool` which comprise the `matched` set.

`covar` A character vector describing the covariates used for matching.

`method` Character describing replacement method used for matching.

`replace` TRUE/FALSE describing if matching was done with or without replacement.

`anchor1` `anchorIds(delegate)$first`

`anchor2` `anchorIds(delegate)$second`

`regions` `regions(delegate)`

`NAMES` `names(delegate)`

`elementMetadata` `elementMetadata(delegate)`

`metadata` `metadata(delegate)`

Accessor methods for Matched Class

Functions that get data from Matched subclasses (`x`) such as MatchedDataFrame, MatchedGRanges, and MatchedGInteractions are listed below:

- `matchedData(x)`: Get matched data from a Matched object
- `covariates(x)`: Get covariates from a Matched object
- `method(x)`: Get matching method used for Matched object
- `withReplacement(x)`: Get replace method
- `indices(x, set)`: Get indices of matched set

For more detail check the help pages for these functions.

Accessor methods for Matched subclass objects

Additional functions that get data from Matched subclasses (`x`) such as MatchedDataFrame, MatchedGRanges, and MatchedGInteractions include:

- `focal(x)`: Get focal set from a Matched object
- `pool(x)`: Get pool set from a Matched object
- `matched(x)`: Get matched set from a Matched object
- `unmatched(x)`: Get unmatched set from a Matched object

For more detail check the help pages for these functions.

See Also

InteractionSet::GInteractions

matchedData, covariates, method, withReplacement, indices

focal, pool, matched, unmatched

Examples

```
## Constructing MatchedGInteractions with matchRanges
set.seed(123)
gi <- makeExampleMatchedDataSet(type = "GInteractions")
mgi <- matchRanges(
  focal = gi[gi$feature1, ],
  pool = gi[!gi$feature1, ],
  covar = ~ feature2 + feature3,
  method = "rejection",
  replace = FALSE
)
class(mgi)

## Make MatchedGInteractions example
set.seed(123)
x <- makeExampleMatchedDataSet(type = "GInteractions", matched = TRUE)
## Accessor functions for Matched class
matchedData(x)
covariates(x)
method(x)
withReplacement(x)
head(indices(x, set = 'matched'))

## Accessor functions for Matched subclasses
focal(x)
pool(x)
matched(x)
unmatched(x)
```

MatchedGRanges-class *MatchedGRanges objects*

Description

The MatchedGRanges class is a subclass of both Matched and GRanges. Therefore, it contains slots and methods for both of these classes.

Details

The MatchedGRanges class uses a delegate object during initialization to assign its GRanges slots. MatchedGRanges behaves as a GRanges but also includes additional Matched object functionality (see ?Matched). For more information about GRanges see ?GenomicRanges::GRanges.

Slots

focal A GRanges object containing the focal data to match.

pool A GRanges object containing the pool from which to select matches.

`delegate` A `GRanges` object used to initialize `GRanges`-specific slots. `matchRanges()` assigns the matched set to the slot.

`matchedData` A `data.table` with matched data

`matchedIndex` An integer vector corresponding to the indices in the `pool` which comprise the `matched` set.

`covar` A character vector describing the covariates used for matching.

`method` Character describing replacement method used for matching.

`replace` TRUE/FALSE describing if matching was done with or without replacement.

`seqnames seqnames(delegate)`

`ranges ranges(delegate)`

`strand strand(delegate)`

`seqinfo seqinfo(delegate)`

`elementMetadata elementMetadata(delegate)`

`elementType elementType(delegate)`

`metadata metadata(delegate)`

Accessor methods for Matched Class

Functions that get data from Matched subclasses (`x`) such as MatchedDataFrame, MatchedGRanges, and MatchedGInteractions are listed below:

- `matchedData(x)`: Get matched data from a Matched object
- `covariates(x)`: Get covariates from a Matched object
- `method(x)`: Get matching method used for Matched object
- `withReplacement(x)`: Get replace method
- `indices(x, set)`: Get indices of matched set

For more detail check the help pages for these functions.

Accessor methods for Matched subclass objects

Additional functions that get data from Matched subclasses (`x`) such as MatchedDataFrame, MatchedGRanges, and MatchedGInteractions include:

- `focal(x)`: Get focal set from a Matched object
- `pool(x)`: Get pool set from a Matched object
- `matched(x)`: Get matched set from a Matched object
- `unmatched(x)`: Get unmatched set from a Matched object

For more detail check the help pages for these functions.

See Also

GenomicRanges::GRanges

matchedData, covariates, method, withReplacement, indices

focal, pool, matched, unmatched

Examples

```
## Contructing MatchedGRanges with matchRanges
set.seed(123)
gr <- makeExampleMatchedDataSet(type = "GRanges")
mgr <- matchRanges(
  focal = gr[gr$feature1, ],
  pool = gr[!gr$feature1, ],
  covar = ~ feature2 + feature3,
  method = "rejection",
  replace = FALSE
)
class(mgr)

## Make MatchedGRanges example
set.seed(123)
x <- makeExampleMatchedDataSet(type = "GRanges", matched = TRUE)
## Accessor functions for Matched class
matchedData(x)
covariates(x)
method(x)
withReplacement(x)
head(indices(x, set = 'matched'))

## Accessor functions for Matched subclasses
focal(x)
pool(x)
matched(x)
unmatched(x)
```

---

matchitToMatched *Coerce* matchit *to* Matched *object*

Description

Coerce matchit to Matched object

Usage

```
matchitToMatched(x, ranges = NULL, keep_mcols = TRUE)

## S4 method for signature 'matchit'
matchitToMatched(x, ranges = NULL, keep_mcols = TRUE)
```

Arguments

x	A matchit object from MatchIt package.
ranges	A GRanges or GInteractions object that is the same length as the data used to create the matchit object, x.

keep_mcols — Logical whether to keep or existing mcols of the object supplied in ranges or not. Default is TRUE.

Value

A Matched subclass object, depending on the value of ranges. If is(ranges, "GRanges") then a MatchedGRanges object is returned. If is(ranges, "GInteractions") then a MatchedGInteractions object is returned. If is.null(ranges) then a the function first attempts to coerce into the other classes before coercing to MatchedDataFrame.

Examples

```
if (!requireNamespace("MatchIt", quietly=TRUE)) {
 set.seed(123)
 x <- makeExampleMatchedDataSet(type="GRanges")

 ## Convert GRanges to data.frame, pass to matchit,
 ## and convert to MatchedGRanges object
 set.seed(123)
 mgr <-
   as.data.frame(x) |>
   matchit(formula=feature1 ~ feature2 + feature3,
           data=_,
           method='nearest',
           replace=FALSE) |>
   matchitToMatched()

 ## Compatible with GRanges & Matched functions
 mgr
 plotCovariate(mgr)
} else {
 message("The 'MatchIt' package is required to run this example.")
}
```

---

matchRanges — *Generate a covariate-matched control set of ranges*

Description

matchRanges() uses a propensity score-based method to generate a covariate-matched control set of DataFrame, GRanges, or GInteractions objects.

Usage

```
matchRanges(focal, pool, covar, method = "nearest", replace = TRUE, ...)

## S4 method for signature
```

```
## 'DF_OR_df_OR_dt,
##   DF_OR_df_OR_dt,
##   formula,
##   character_OR_missing,
##   logical_OR_missing'
matchRanges(focal, pool, covar, method, replace)

## S4 method for signature
## 'GRanges,GRanges,formula,character_OR_missing,logical_OR_missing'
matchRanges(focal, pool, covar, method, replace)

## S4 method for signature
## 'GInteractions,
##   GInteractions,
##   formula,
##   character_OR_missing,
##   logical_OR_missing'
matchRanges(focal, pool, covar, method, replace)
```

Arguments

focal	A DataFrame, GRanges, or GInteractions object containing the focal data to match.
pool	A DataFrame, GRanges, or GInteractions object containing the pool from which to select matches.
covar	A rhs formula with covariates on which to match.
method	A character describing which matching method to use. supported options are either 'nearest', 'rejection', or 'stratified'.
replace	TRUE/FALSE describing whether to select matches with or without replacement.
...	Additional arguments.

Details

Available inputs for `focal` and `pool` include `data.frame`, `data.table`, `DataFrame`, `GRanges`, or `GInteractions`. `data.frame` and `data.table` inputs are coerced to `DataFrame` objects and returned as `MatchedDataFrame` while `GRanges` and `GInteractions` objects are returned as `MatchedGRanges` or `MatchedGInteractions`, respectively.

Value

A covariate-matched control set of data.

Methodology

`matchRanges` uses propensity scores to perform subset selection on the `pool` set such that the resulting `matched` set contains similar distributions of covariates to that of the `focal` set. A propensity score is the conditional probability of assigning an element (in our case, a genomic range) to a

particular outcome (`Y`) given a set of covariates. Propensity scores are estimated using a logistic regression model where the outcome `Y=1` for `focal` and `Y=0` for `pool`, over the provided covariates `covar`.

Matching methods

- `method = 'nearest'`: Nearest neighbor matching with replacement. Finds the nearest neighbor by using a rolling join with `data.table`. Matching without replacement is not currently supported.
- `method = 'rejection'`: (Default) Rejection sampling with or without replacement. Uses a probability-based approach to select options in the `pool` that match the `focal` distribition.
- `method = 'stratified'`: Iterative stratified sampling with or without replacement. Bins `focal` and `pool` propensity scores by value and selects matches within bins until all `focal` items have a corresponding match in `pool`.

References

matchRanges manuscript:

Eric S. Davis, Wancen Mu, Stuart Lee, Mikhail G. Dozmorov, Michael I. Love, Douglas H. Phanstiel. 2023. "matchRanges: Generating null hypothesis genomic ranges via covariate-matched sampling." Bioinformatics. doi: 10.1093/bioinformatics/btad197

Examples

```
## Match with DataFrame
set.seed(123)
x <- makeExampleMatchedDataSet(type = 'DataFrame')
matchRanges(focal = x[x$feature1,],
            pool = x[!x$feature1,],
            covar = ~feature2 + feature3)

## Match with GRanges
set.seed(123)
x <- makeExampleMatchedDataSet(type = "GRanges")
matchRanges(focal = x[x$feature1,],
            pool = x[!x$feature1,],
            covar = ~feature2 + feature3)

## Match with GInteractions
set.seed(123)
x <- makeExampleMatchedDataSet(type = "GInteractions")
matchRanges(focal = x[x$feature1,],
            pool = x[!x$feature1,],
            covar = ~feature2 + feature3)

## Nearest neighbor matching with replacement
set.seed(123)
x <- makeExampleMatchedDataSet(type = 'DataFrame')
matchRanges(focal = x[x$feature1,],
            pool = x[!x$feature1,],
```

```
            covar = ~feature2 + feature3,
            method = 'nearest',
            replace = TRUE)

## Rejection sampling without replacement
set.seed(123)
x <- makeExampleMatchedDataSet(type = 'DataFrame')
matchRanges(focal = x[x$feature1,],
            pool = x[!x$feature1,],
            covar = ~feature2 + feature3,
            method = 'rejection',
            replace = FALSE)

## Stratified sampling without replacement
set.seed(123)
x <- makeExampleMatchedDataSet(type = 'DataFrame')
matchRanges(focal = x[x$feature1,],
            pool = x[!x$feature1,],
            covar = ~feature2 + feature3,
            method = 'stratified',
            replace = FALSE)
```

`method` *Get matching method used for Matched object*

Description

Get matching method used for Matched object

Usage

```
method(x, ...)

## S4 method for signature 'Matched'
method(x, ...)
```

Arguments

`x`	Matched object.
`...`	Additional arguments.

Value

A character describing the matched method

Examples

```
set.seed(123)
mdf <- makeExampleMatchedDataSet(matched = TRUE)
method(mdf)
```

`names.MatchedOverview` *Names S3 method for autocomplete*

Description

Names S3 method for autocomplete

Extract `$` operator for MatchedOverview

Extract `[[` operator for MatchedOverview

Show method for `overview`

Usage

```
## S3 method for class 'MatchedOverview'
names(x)

## S4 method for signature 'MatchedOverview'
x$name

## S4 method for signature 'MatchedOverview,ANY,ANY'
x[[i]]

## S4 method for signature 'MatchedOverview'
show(object)
```

Arguments

`x`	A `MatchedOverview` object.
`name`	Name of slot.
`i`	A character or numeric to extract.
`object`	A `MatchedOverview` object.

Value

different data or metadata concerning a MatchedOverview object

oneRegionSegment *Segmentation based on one region*

Description

This function makes a segmentation (GRanges) based on one region of one chromosome (seqnames).

Usage

```
oneRegionSegment(x, seqlength)
```

Arguments

x	a single region as GRanges object
seqlength	optional, the length of the chromosome, if not provided, the function will attempt to pull this using genome(x) and the Seqinfo function

Value

a segmentation (GRanges object) with the region of interest designated as state 2, and the rest of the chromosome as state 1.

Examples

```
library(GenomicRanges)
x <- GRanges("chr1", IRanges(10e6+1,width=1e6))
genome(x) <- "hg19"
seg <- oneRegionSegment(x)
```

overview *Overview of matching quality*

Description

The overview function provides a quick assessment of overall matching quality by reporting the N, mean, and s.d. of focal, matched, pool, and unmatched sets for all covariates as well as the propensity scores ('ps'). The mean and s.d. difference in focal - matched is also reported.

Usage

```
overview(x, digits = 2, ...)

## S4 method for signature 'Matched,numeric_OR_missing'
overview(x, digits)
```

Arguments

x	Matched object
digits	Integer indicating the number of significant digits to be used. Negative values are allowed (see ?signif).
...	Additional arguments.

Details

Factor, character, or logical covariates are reported by N per set, rather than with mean and s.d.

Value

- A printed overview of matching quality.
- (Internally) a MatchedOverview object.

Examples

```
set.seed(123)
mdf <- makeExampleMatchedDataSet(matched = TRUE)
overview(mdf)
```

---

plotCovariate *Covariate plotting for Matched objects*

Description

This function plots the distributions of a covariate from each matched set of a Matched object.

Usage

```
plotCovariate(
  x,
  covar = NULL,
  sets = c("focal", "matched", "pool", "unmatched"),
  type = NULL,
  log = NULL,
  ...
)

## S4 method for signature
## 'Matched,
##   character_OR_missing,
##   character_OR_missing,
##   character_OR_missing,
##   character_OR_missing'
plotCovariate(x, covar, sets, type, log, thresh = 12)
```

Arguments

x	Matched object
covar	Character naming the covariate to plot. If multiple are provided, only the first one is used.
sets	Character vector describing which matched set(s) to include in the plot. Options are 'focal', 'matched', 'pool', or 'unmatched'. Multiple options are accepted.
type	Character naming the plot type. Available options are one of either 'ridges', 'jitter', 'lines', or 'bars'. Note that for large datasets, use of 'jitter' is discouraged because the large density of points can stall the R-graphics device.
log	Character vector describing which axis or axes to apply log-transformation. Available options are 'x' and/or 'y'.
...	Additional arguments.
thresh	Integer describing the number of unique values required to classify a numeric variable as discrete (and convert it to a factor). If the number of unique values exceeds thresh then the variable is considered continuous.

Details

By default, plotCovariate will sense the class of covariate and make a plot best suited to that data type. For example, if the covariate class is categorical in nature then the type argument defaults to 'bars'. type is set to 'lines' for continuous covariates. These settings can also be overwritten manually.

Value

Returns a plot of a covariate's distribution among matched sets.

See Also

plotPropensity() to plot propensity scores.

Examples

```
## Matched example dataset
set.seed(123)
mdf <- makeExampleMatchedDataSet(matched = TRUE)

## Visualize covariates
plotCovariate(mdf)
plotCovariate(mdf, covar = 'feature3')
plotCovariate(mdf,
              covar = 'feature2',
              sets = c('focal', 'matched', 'pool'))
plotCovariate(mdf,
              covar = 'feature2',
              sets = c('focal', 'matched', 'pool'),
              type = 'ridges')
plotCovariate(mdf,
```

```
covar = 'feature2',
sets = c('focal', 'matched', 'pool'),
type = 'jitter')
```

`plotPropensity` *Propensity score plotting for Matched objects*

Description

This function plots the distribution of propensity scores from each matched set of a Matched object.

Usage

```
plotPropensity(
  x,
  sets = c("focal", "matched", "pool", "unmatched"),
  type = NULL,
  log = NULL,
  ...
)

## S4 method for signature
## 'Matched,
##   character_OR_missing,
##   character_OR_missing,
##   character_OR_missing'
plotPropensity(x, sets, type, log, thresh = 12)
```

Arguments

`x`	Matched object
`sets`	Character vector describing which matched set(s) to include in the plot. Options are 'focal', 'matched', 'pool', or 'unmatched'. Multiple options are accepted.
`type`	Character naming the plot type. Available options are one of either 'ridges', 'jitter', 'lines', or 'bars'. Note that for large datasets, use of 'jitter' is discouraged because the large density of points can stall the R-graphics device.
`log`	Character vector describing which axis or axes to apply log-transformation. Available options are 'x' and/or 'y'.
`...`	Additional arguments.
`thresh`	Integer describing the number of unique values required to classify a numeric variable as discrete (and convert it to a factor). If the number of unique values exceeds `thresh` then the variable is considered continuous.

Details

plotPropensity uses the thresh argument to determine whether to plot propensity scores as continuous (line plot) or catetgorical (bar plot). These settings can also be overwritten manually.

Value

Returns a plot of propensity score distributions among matched sets.

See Also

plotCovariate() to plot covariate distributions.

Examples

```
## Matched example dataset
set.seed(123)
mdf <- makeExampleMatchedDataSet(matched = TRUE)

## Visualize propensity scores
plotPropensity(mdf)
plotPropensity(mdf,
               sets = c('focal', 'matched', 'pool'))
plotPropensity(mdf,
               sets = c('focal', 'matched', 'pool'),
               type = 'ridges')
plotPropensity(mdf,
               sets = c('focal', 'matched', 'pool'),
               type = 'jitter')
```

---

plotSegment *Plot genome segmentation*

Description

Plot genome segmentation

Usage

```
plotSegment(
  seg,
  exclude = NULL,
  type = c("ranges", "barplot", "boxplot"),
  region = NULL
)
```

Arguments

seg	the segmentation GRanges returned by segmentDensity function
exclude	GRanges of excluded region
type	the type of plot returned. Choices are segmentation plot included ranges information, barplot showing segmentation states' distribution across chromosome, or a box plot indicating average density within each states. Default is all plots are displayed. The y axis "density" represent square root of overlap counts within segment length. If a user defined segmentation GRanges is given, the y axis is default to be 1 for all states in segmentation plot.
region	GRanges of stricted region that want to be plotted.

Value

A ggplot set by type argument

Examples

```
example("segmentDensity")
plotSegment(seg, exclude, type = "ranges")
plotSegment(seg, exclude, type = "barplot")
plotSegment(seg, exclude, type = "boxplot")
```

---

pool *Get pool set from a Matched object*

Description

Get pool set from a Matched object

Usage

```
pool(x, ...)

## S4 method for signature 'MDF_OR_MGR_OR_MGI'
pool(x, ...)
```

Arguments

x	A MatchedDataFrame, MatchedGRanges, or MatchedGInteractions object.
...	Additional options.

Value

An object of the same class as x representing the pool set.

Examples

```
set.seed(123)
x <- makeExampleMatchedDataSet(matched = TRUE)
pool(x)
```

`reduceSegment` *Combine nearby regions with same state*

Description

Combine nearby regions with same state

Usage

```
reduceSegment(x, col = "state")
```

Arguments

`x`	the input GRanges
`col`	the name of the column for the segment states

Value

a GRanges with metadata column `state` giving the segmentation state

Examples

```
n <- 10000
library(GenomicRanges)
gr <- GRanges("chr1", IRanges(round(
  c(runif(n/4,1,991), runif(n/4,1001,3991),
    runif(n/4,4001,4991), runif(n/4,7001,9991))),
  width=10), seqlengths=c(chr1=10000))
gr$name <- rep(1:4,each=10)
gr <- sort(gr)
seg <- reduceSegment(gr, col="name")
```

`segmentDensity` *Genome segmentation based on feature density*

Description

This function allows for various methods (see `type`) of segmenting based on the density of features `x`.

Usage

```
segmentDensity(x, n, L_s = 1e+06, exclude = NULL, type = c("cbs", "hmm"))
```

Arguments

`x`	the input GRanges, e.g. genes
`n`	the number of states
`L_s`	segment length
`exclude`	GRanges of excluded region
`type`	the type of segmentation, either Circular Binary Segmentation `"cbs"` (which will use 'DNAcopy' to segment) or Hidden Markov Model `"hmm"` (which will use 'RcppHMM'). These packages are not imported by nullranges, but must be installed by the user

Value

a GRanges with metadata columns containing:

- state - segmentation state
- counts - average number of genes

References

Circular binary segmentation (CBS):

Olshen, A. B., E. S. Venkatraman, R. Lucito, and M. Wigler. 2004. "Circular binary segmentation for the analysis of array-based DNA copy number data." Biostatistics 5 (4): 557–72.

Hidden Markov Model from RcppHMM:

Roberto A. Cardenas-Ovando, Julieta Noguez, and Claudia Rangel-Escareno. "Rcpp Hidden Markov Model." CRAN R package.

Examples

```
n <- 10000
library(GenomicRanges)
gr <- GRanges("chr1", IRanges(round(
  c(runif(n/4,1,991), runif(n/4,1001,3991),
    runif(n/4,4001,4991), runif(n/4,7001,9991))),
  width=10), seqlengths=c(chr1=10000))
gr <- sort(gr)
exclude <- GRanges("chr1", IRanges(5001,6000), seqlengths=c(chr1=10000))
seg <- segmentDensity(gr, n=3, L_s=100, exclude=exclude, type="cbs")
```

---

unmatched *Get unmatched set from a Matched object*

Description

Get unmatched set from a Matched object

Usage

```
unmatched(x, ...)

## S4 method for signature 'MDF_OR_MGR_OR_MGI'
unmatched(x, ...)
```

Arguments

x	A MatchedDataFrame, MatchedGRanges, or MatchedGInteractions object.
...	Additional options.

Value

An object of the same class as x representing the unmatched set.

Examples

```
set.seed(123)
x <- makeExampleMatchedDataSet(matched = TRUE)
unmatched(x)
```

withReplacement *Get replace method*

Description

Determine if a Matched object was created with or without replacement.

Usage

```
withReplacement(x, ...)

## S4 method for signature 'Matched'
withReplacement(x, ...)
```

Arguments

x	Matched object.
...	Additional arguments.

Value

TRUE/FALSE indicating whether matching was done with or without replacement.

Examples

```
set.seed(123)
mdf <- makeExampleMatchedDataSet(matched = TRUE)
withReplacement(mdf)
```

Index

∗ **Matched plotting**
    plotCovariate, 25
∗ **internal**
    names.MatchedOverview, 23
[[,MatchedOverview,ANY,ANY-method *(names.MatchedOverview)*, 23
$,MatchedOverview-method *(names.MatchedOverview)*, 23

BootRanges *(BootRanges-class)*, 4
bootRanges, 3
BootRanges-class, 4

combnCov, 5
combnCov,character-method *(combnCov)*, 5
covariates, 5, *10*, *13*, *15*, *17*
covariates,Matched-method *(covariates)*, 5

focal, 6, *13*, *15*, *17*
focal,MDF_OR_MGR_OR_MGI-method *(focal)*, 6

GenomicRanges::GRanges, *17*

indices, 7, *10*, *13*, *15*, *17*
indices,Matched-method *(indices)*, 7
InteractionSet::GInteractions, *15*

makeExampleMatchedDataSet, 8
makeExampleMatchedDataSet,character_OR_missing,logical_OR_missing,character_OR_missing,logical_OR_missi *(makeExampleMatchedDataSet)*, 8
Matched *(Matched-class)*, 10
matched, 9, *13*, *15*, *17*
matched,MDF_OR_MGR_OR_MGI-method *(matched)*, 9
Matched-class, 10
matchedData, *10*, 11, *13*, *15*, *17*
matchedData,Matched-method *(matchedData)*, 11
MatchedDataFrame *(MatchedDataFrame-class)*, 12
MatchedDataFrame-class, 12
MatchedGInteractions *(MatchedGInteractions-class)*, 14
MatchedGInteractions-class, 14
MatchedGRanges *(MatchedGRanges-class)*, 16
MatchedGRanges-class, 16
matchitToMatched, 18
matchitToMatched,matchit-method *(matchitToMatched)*, 18
matchRanges, 19
matchRanges,DF_OR_df_OR_dt,DF_OR_df_OR_dt,formula,characte *(matchRanges)*, 19
matchRanges,GInteractions,GInteractions,formula,character_ *(matchRanges)*, 19
matchRanges,GRanges,GRanges,formula,character_OR_missing,l *(matchRanges)*, 19
method, *10*, *13*, *15*, *17*, 22
method,Matched-method *(method)*, 22

names.MatchedOverview, 23

oneRegionSegment, 24
overview, 24
overview,Matched,numeric_OR_missing-method *(overview)*, 24

plotCovariate, 25
plotCovariate(), *28*
plotCovariate,Matched,character_OR_missing,character_OR_mi *(plotCovariate)*, 25
plotPropensity, 27
plotPropensity(), *26*
plotPropensity,Matched,character_OR_missing,character_OR_m *(plotPropensity)*, 27
plotSegment, 28
pool, *13*, *15*, *17*, 29

pool,MDF_OR_MGR_OR_MGI-method *(*pool*)*, 29

reduceSegment, 30

S4Vectors::DataFrame, *13*
segmentDensity, 31
show,MatchedOverview-method
*(*names.MatchedOverview*)*, 23

unmatched, *13*, *15*, *17*, 32
unmatched,MDF_OR_MGR_OR_MGI-method
*(*unmatched*)*, 32

withReplacement, *10*, *13*, *15*, *17*, 33
withReplacement,Matched-method
*(*withReplacement*)*, 33