Package 'Signac'

April 16, 2020

Title Analysis of Single-Cell Chromatin Data

Version 0.2.5

Date 2020-04-11

Description A framework for the analysis and exploration of single-cell chromatin data. The 'Signac' package contains functions for quantifying single-cell chromatin data, computing per-cell quality control metrics, dimension reduction and normalization, visualization, and DNA sequence motif analysis. Reference: Stuart and Butler et al. (2019) <doi:10.1016/j.cell.2019.05.031>.

Depends R (>= 3.4.0), methods

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.1.0

URL https://github.com/timoast/signac, https://satijalab.org/signac

BugReports https://github.com/timoast/signac/issues

Imports GenomeInfoDb, GenomicRanges, IRanges, Matrix, Rsamtools, S4Vectors, Seurat, data.table, dplyr, future, future.apply, ggplot2, ggseqlogo, irlba, pbapply, tidyr, zoo, patchwork, gggenes, grid, stats, utils, BiocGenerics

Collate 'data.R' 'generics.R' 'dimension_reduction.R' 'motifs.R' 'objects.R' 'preprocessing.R' 'utilities.R' 'visualization.R'

Suggests testthat (>= 2.1.0), chromVAR, SummarizedExperiment, TFBSTools, motifmatchr, Biostrings, BSgenome

NeedsCompilation no

Author Tim Stuart [aut, cre] (<https://orcid.org/0000-0002-3044-0897>), Paul Hoffman [ctb] (<https://orcid.org/0000-0002-7693-8957>), Rahul Satija [ctb] (<https://orcid.org/0000-0001-9448-8833>)

Maintainer Tim Stuart <tstuart@nygenome.org>

Repository CRAN

Date/Publication 2020-04-16 09:10:02 UTC

R topics documented:

`AddMotifObject` *Add a Motif object to a Seurat object*

Description

Add a Motif object to a Seurat object

Usage

```
AddMotifObject(object, ...)

## S3 method for class 'Assay'
AddMotifObject(object, motif.object, verbose = TRUE, ...)

## S3 method for class 'Seurat'
AddMotifObject(object, motif.object, assay = NULL, ...)
```

Arguments

`object`	A Seurat object
`...`	Additional arguments
`motif.object`	An object of class Motif
`verbose`	Display messages
`assay`	Name of assay to store motif object in

Value

Returns a `Seurat` object

Examples

```
obj <- GetMotifObject(atac_small[['peaks']])
atac_small[['peaks']] <- AddMotifObject(
object = atac_small[['peaks']], motif.object = obj
)
obj <- GetMotifObject(object = atac_small)
atac_small[['peaks']] <- AddMotifObject(
object = atac_small, motif.object = obj
)
```

---

atac_small *A small example scATAC-seq dataset*

Description

A subsetted version of 10x Genomics 10k human (hg19) PBMC scATAC-seq dataset

Usage

```
atac_small
```

Format

A Seurat object with the following assays

peaks A peak x cell dataset

bins A 5 kb genome bin x cell dataset

RNA A gene x cell dataset

Source

https://support.10xgenomics.com/single-cell-atac/datasets/1.1.0/atac_v1_pbmc_10k

---

AverageCounts *Average Counts*

Description

Compute the mean counts per group of cells for a given assay

Usage

```
AverageCounts(object, assay = NULL, group.by = NULL, verbose = TRUE)
```

Arguments

object	A Seurat object
assay	Name of assay to use. Default is the active assay
group.by	Grouping variable to use. Default is the active identities
verbose	Display messages

Value

Returns a dataframe

Examples

```
AverageCounts(atac_small)
```

BinarizeCounts *Binarize counts*

Description

Set counts >1 to 1 in a count matrix

Usage

```
BinarizeCounts(object, ...)

## Default S3 method:
BinarizeCounts(object, assay = NULL, verbose = TRUE, ...)

## S3 method for class 'Assay'
BinarizeCounts(object, assay = NULL, verbose = TRUE, ...)

## S3 method for class 'Seurat'
BinarizeCounts(object, assay = NULL, verbose = TRUE, ...)
```

Arguments

object	A Seurat object
...	Arguments passed to other methods
assay	Name of assay to use. Can be a list of assays, and binarization will be applied to each.
verbose	Display messages

Value

Returns a Seurat object

Examples

```
x <- matrix(data = sample(0:3, size = 25, replace = TRUE), ncol = 5)
BinarizeCounts(x)
BinarizeCounts(atac_small[['peaks']])
BinarizeCounts(atac_small)
```

---

blacklist_ce10 *Genomic blacklist regions for C. elegans ce10*

Description

Genomic blacklist regions for C. elegans ce10

Usage

```
blacklist_ce10
```

Format

A GRanges object

Source

https://github.com/Boyle-Lab/Blacklist

https://doi.org/10.1038/s41598-019-45839-z

---

blacklist_ce11 *Genomic blacklist regions for C. elegans ce11*

Description

Genomic blacklist regions for C. elegans ce11

Usage

```
blacklist_ce11
```

Format

A GRanges object

Source

https://github.com/Boyle-Lab/Blacklist

https://doi.org/10.1038/s41598-019-45839-z

blacklist_dm3 *Genomic blacklist regions for Drosophila dm3*

Description

Genomic blacklist regions for Drosophila dm3

Usage

```
blacklist_dm3
```

Format

A GRanges object

Source

https://github.com/Boyle-Lab/Blacklist

https://doi.org/10.1038/s41598-019-45839-z

blacklist_dm6 *Genomic blacklist regions for Drosophila dm6*

Description

Genomic blacklist regions for Drosophila dm6

Usage

```
blacklist_dm6
```

Format

A GRanges object

Source

https://github.com/Boyle-Lab/Blacklist

https://doi.org/10.1038/s41598-019-45839-z

blacklist_hg19 *Genomic blacklist regions for Human hg19*

Description

Genomic blacklist regions for Human hg19

Usage

```
blacklist_hg19
```

Format

A GRanges object

Source

https://github.com/Boyle-Lab/Blacklist

https://doi.org/10.1038/s41598-019-45839-z

blacklist_hg38 *Genomic blacklist regions for Human GRCh38*

Description

Genomic blacklist regions for Human GRCh38

Usage

```
blacklist_hg38
```

Format

A GRanges object

Source

https://github.com/Boyle-Lab/Blacklist

https://doi.org/10.1038/s41598-019-45839-z

blacklist_mm10 *Genomic blacklist regions for Mouse mm10*

Description

Genomic blacklist regions for Mouse mm10

Usage

```
blacklist_mm10
```

Format

A GRanges object

Source

https://github.com/Boyle-Lab/Blacklist

https://doi.org/10.1038/s41598-019-45839-z

CellsPerGroup *Cells per group*

Description

Count the number of cells in each group

Usage

```
CellsPerGroup(object, group.by = NULL)
```

Arguments

object	A Seurat object
group.by	A grouping variable. Default is the active identities

Value

Returns a vector

Examples

```
CellsPerGroup(atac_small)
```

ClosestFeature *Closest Feature*

Description

Find the closest feature to a given set of genomic regions

Usage

```
ClosestFeature(regions, annotation, ...)
```

Arguments

regions	A set of genomic regions to query
annotation	A GRanges object containing annotation information.
...	Additional arguments passed to StringToGRanges

Value

Returns a dataframe with the name of each region, the closest feature in the annotation, and the distance to the feature.

Examples

```
ClosestFeature(
  regions = head(rownames(atac_small)),
  annotation = StringToGRanges(
  head(rownames(atac_small)),
  sep = c(':', '-')),
  sep = c(":", "-")
)
```

CountsInRegion *CountsInRegion*

Description

Count reads per cell overlapping a given set of regions

Usage

```
CountsInRegion(object, assay, regions, sep = c("-", "-"), ...)
```

Arguments

object	A Seurat object
assay	Name of assay in the object to use
regions	A GRanges object
sep	Separator to use when extracting genomic coordinates from the Seurat object
...	Additional arguments passed to findOverlaps

Value

Returns a numeric vector

Examples

```
CountsInRegion(
  object = atac_small,
  assay = 'bins',
  regions = blacklist_hg19
)
```

CreateMotifMatrix *CreateMotifMatrix*

Description

Create a motif x feature matrix from a set of genomic ranges, the genome, and a set of position weight matrices.

Usage

```
CreateMotifMatrix(
  features,
  pwm,
  genome,
  score = FALSE,
  use.counts = FALSE,
  sep = c("-", "-"),
  ...
)
```

Arguments

features	A GRanges object containing a set of genomic features
pwm	A PFMatrixList or PWMatrixList object containing position weight/frequency matrices to use
genome	Any object compatible with the genome argument in matchMotifs
score	Record the motif match score, rather than presence/absence (default FALSE)
use.counts	Record motif counts per region. If FALSE (default), record presence/absence of motif. Only applicable if score=FALSE.
sep	A length-2 character vector containing the separators to be used when constructing matrix rownames from the GRanges
...	Additional arguments passed to matchMotifs

Details

Requires that motifmatchr is installed https://www.bioconductor.org/packages/motifmatchr/.

Value

Returns a sparse matrix

Examples

```
## Not run:
library(JASPAR2018)
library(TFBSTools)
library(BSgenome.Hsapiens.UCSC.hg19)

pwm <- getMatrixSet(
  x = JASPAR2018,
  opts = list(species = 9606, all_versions = FALSE)
)
motif.matrix <- CreateMotifMatrix(
  features = StringToGRanges(rownames(atac_small), sep = c(":", "-")),
  pwm = pwm,
  genome = BSgenome.Hsapiens.UCSC.hg19,
  sep = c(":", "-")
)

## End(Not run)
```

CreateMotifObject *CreateMotifObject*

Description

Create an object of class Motif

Usage

```
CreateMotifObject(
  data = NULL,
  pwm = NULL,
  motif.names = NULL,
  meta.data = NULL
)
```

Arguments

data	A motif x region matrix
pwm	A named list of position weight matrices or position frequency matrices matching the motif names in data. Can be of class PFMatrixList.
motif.names	A named list of motif names. List element names must match the names given in pwm. If NULL, use the names from the list of position weight or position frequency matrices. This can be used to set a alternative common name for the motif. If a PFMatrixList is passed to pwm, it will pull the motif name from the PFMatrixList.
meta.data	A data.frame containing metadata

Value

Returns a Motif object

Examples

```
motif.matrix <- matrix(
  data = sample(c(0,1),
    size = 100,
    replace = TRUE),
  ncol = 5
)
motif <- CreateMotifObject(data = motif.matrix)
```

CutMatrix *Generate matrix of integration sites*

Description

Generates a cell-by-position matrix of Tn5 integration sites centered on a given region (usually a DNA sequence motif). This matrix can be used for downstream footprinting analysis.

Usage

```
CutMatrix(
  object,
  region,
  tabix.file = NULL,
  assay = NULL,
  cells = NULL,
  verbose = TRUE
)
```

Arguments

object	A Seurat object
region	A GRanges object containing the region of interest
tabix.file	A TabixFile object. If NULL, the file specified in fragment.path will be opened and closed after the function completes. If iterating over many regions, providing an open TabixFile is much faster as it avoids opening and closing the connection each time.
assay	Name of the assay to use
cells	Which cells to include in the matrix. If NULL (default), use all cells in the object
verbose	Display messages

Value

Returns a sparse matrix

Examples

```
fpath <- system.file("extdata", "fragments.tsv.gz", package="Signac")
atac_small <- SetFragments(atac_small, file = fpath)
CutMatrix(
 object = atac_small,
 region = StringToGRanges("chr1-10245-762629")
)
```

DepthCor *Sequencing depth correlation*

Description

Compute the correlation between total counts and each reduced dimension component.

Usage

```
DepthCor(object, assay = NULL, reduction = "lsi", n = 10, ...)
```

Arguments

object	A Seurat object
assay	Name of assay to use for sequencing depth. If NULL, use the default assay.
reduction	Name of a dimension reduction stored in the input object
n	Number of components to use. If NULL, use all components.
...	Additional arguments passed to cor

Value

Returns a ggplot object

Examples

```
DepthCor(object = atac_small)
```

DownsampleFeatures *DownsampleFeatures*

Description

Randomly downsample features and assign to VariableFeatures for the object. This will select n features at random.

Usage

```
DownsampleFeatures(object, assay = NULL, n = 20000, verbose = TRUE)
```

Arguments

object	A Seurat object
assay	Name of assay to use. Default is the active assay.
n	Number of features to retain (default 20000).
verbose	Display messages

Value

Returns a Seurat object with VariableFeatures set to the randomly sampled features.

Examples

```
DownsampleFeatures(atac_small, n = 10)
```

Extend *Extend*

Description

Resize GenomicRanges upstream and or downstream. From https://support.bioconductor.org/p/78652/

Usage

```
Extend(x, upstream = 0, downstream = 0, from.midpoint = FALSE)
```

Arguments

x	A range
upstream	Length to extend upstream
downstream	Length to extend downstream
from.midpoint	Count bases from region midpoint, rather than the 5' or 3' end for upstream and downstream respectively.

Value

Returns a GRanges object

Examples

```
Extend(x = blacklist_hg19, upstream = 100, downstream = 100)
```

ExtractCell *ExtractCell*

Description

Extract cell barcode from list of tab delimited character vectors (output of scanTabix)

Usage

```
ExtractCell(x)
```

Arguments

x	List of character vectors

Value

Returns a string

Examples

```
ExtractCell(x = "chr1\t1\t10\tatcg\t1")
```

FeatureMatrix *FeatureMatrix*

Description

Construct a feature x cell matrix from a genomic fragments file

Usage

```
FeatureMatrix(
  fragments,
  features,
  cells = NULL,
  chunk = 50,
  sep = c("-", "-"),
  verbose = TRUE
)
```

Arguments

fragments	Path to tabix-indexed fragments file
features	A GRanges object containing a set of genomic intervals. These will form the rows of the matrix, with each entry recording the number of unique reads falling in the genomic region for each cell.
cells	Vector of cells to include. If NULL, include all cells found in the fragments file
chunk	Number of chunks to use when processing the fragments file. Fewer chunks may enable faster processing, but will use more memory.
sep	Vector of separators to use for genomic string. First element is used to separate chromosome and coordinates, second separator is used to separate start and end coordinates.
verbose	Display messages

Value

Returns a sparse matrix

Examples

```
fpath <- system.file("extdata", "fragments.tsv.gz", package="Signac")
FeatureMatrix(
  fragments = fpath,
  features = StringToGRanges(rownames(atac_small), sep = c(":", "-"))
)
```

FilterFragments *FilterFragments*

Description

Remove cells from a fragments file that are not present in a given list of cells. Note that this reads the whole fragments file into memory, so may require a lot of memory depending on the size of the fragments file.

Usage

```
FilterFragments(
  fragment.path,
  cells,
  output.path,
  assume.sorted = FALSE,
  compress = TRUE,
  index = TRUE,
  verbose = TRUE,
  ...
)
```

Arguments

fragment.path	Path to a tabix-indexed fragments file
cells	A vector of cells to retain
output.path	Name and path for output tabix file. A tabix index file will also be created in the same location, with the .tbi file extension.
assume.sorted	Assume sorted input and don't sort the filtered file. Can save a lot of time, but indexing will fail if assumption is wrong.
compress	Compress filtered fragments using bgzip (default TRUE)
index	Index the filtered tabix file (default TRUE)
verbose	Display messages
...	Additional arguments passed to fread

Value

None

Examples

```
fpath <- system.file("extdata", "fragments.tsv.gz", package="Signac")
output.path = file.path(tempdir(), "filtered.tsv")

FilterFragments(
```

```
  fragment.path = fpath,
  cells = colnames(atac_small),
  output.path = output.path
)
```

FindMotifs *FindMotifs*

Description

Find motifs overrepresented in a given set of genomic features. Computes the number of features containing the motif (observed) and compares this to the total number of features containing the motif (background) using the hypergeometric test.

Usage

```
FindMotifs(
  object,
  features,
  background = 40000,
  assay = NULL,
  verbose = TRUE,
  ...
)
```

Arguments

object	A Seurat object
features	A vector of features to test for enrichments over background
background	Either a vector of features to use as the background set, or a number specify the number of features to randomly select as a background set. If a number is provided, regions will be selected to match the sequence characteristics of the query features. To match the sequence characteristics, these characteristics must be stored in the feature metadata for the assay. This can be added using the RegionStats function. If NULL, use all features in the assay.
assay	Which assay to use. Default is the active assay
verbose	Display messages
...	Arguments passed to MatchRegionStats.

Value

Returns a data frame

Examples

```
de.motif <- head(rownames(atac_small))
bg.peaks <- tail(rownames(atac_small))
FindMotifs(
object = atac_small,
features = de.motif,
background = bg.peaks
)
```

`FindTopFeatures` *Find most frequently observed features*

Description

Find top binary features for a given assay based on total number of cells containing feature. Can specify a minumum cell count, or a lower percentile bound.

Usage

```
FindTopFeatures(object, ...)

## Default S3 method:
FindTopFeatures(object, assay = NULL, min.cutoff = "q5", verbose = TRUE, ...)

## S3 method for class 'Assay'
FindTopFeatures(object, assay = NULL, min.cutoff = "q5", verbose = TRUE, ...)

## S3 method for class 'Seurat'
FindTopFeatures(object, assay = NULL, min.cutoff = "q5", verbose = TRUE, ...)
```

Arguments

`object`	A Seurat object
`...`	Arguments passed to other methods
`assay`	Name of assay to use
`min.cutoff`	Cutoff for feature to be included in the VariableFeatures for the object. This can be a percentile specified as 'q' followed by the minimum percentile, for example 'q5' to set the top 95% most common features as the VariableFeatures for the object. Alternatively, this can be an integer specifying the minumum number of cells containing the feature for the feature to be included in the set of VariableFeatures. For example, setting to 10 will include features in >10 cells in the set of VariableFeatures. If NULL, include all features in VariableFeatures.
`verbose`	Display messages

Value

Returns a `Seurat` object

Examples

```
FindTopFeatures(object = atac_small[['peaks']][])
FindTopFeatures(object = atac_small[['peaks']])
FindTopFeatures(atac_small)
```

`FractionCountsInRegion` *FractionCountsInRegion*

Description

Find the fraction of counts per cell that overlap a given set of genomic ranges

Usage

```
FractionCountsInRegion(object, assay, regions, sep = c("-", "-"), ...)
```

Arguments

`object`	A Seurat object
`assay`	Name of assay to use
`regions`	A GRanges object containing a set of genomic regions
`sep`	The separator used to separate genomic coordinate information in the assay feature names
`...`	Additional arguments passed to `CountsInRegion`

Value

Returns a numeric vector

Examples

```
FractionCountsInRegion(
  object = atac_small,
  assay = 'bins',
  regions = blacklist_hg19
)
```

`FragmentHistogram` *Plot fragment length histogram*

Description

Plot fragment length histogram

Usage

```
FragmentHistogram(
  object,
  assay = NULL,
  region = "chr1-1-2000000",
  group.by = NULL,
  cells = NULL,
  log.scale = FALSE,
  ...
)
```

Arguments

`object`	A Seurat object
`assay`	Which assay to use. Default is the active assay.
`region`	Genomic range to use. Default is fist two megabases of chromosome 1. Can be a GRanges object, a string, or a vector of strings.
`group.by`	Name of one or more metadata columns to group (color) the cells by. Default is the current cell identities
`cells`	Which cells to plot. Default all cells
`log.scale`	Display Y-axis on log scale. Default is FALSE.
`...`	Additional arguments passed to `GetReadsInRegion`

Value

Returns a `ggplot` object

Examples

```
fpath <- system.file("extdata", "fragments.tsv.gz", package="Signac")
atac_small <- SetFragments(atac_small, file = fpath)
FragmentHistogram(object = atac_small, region = "chr1-10245-780007")
```

FRiP *Calculate fraction of reads in peaks per cell*

Description

Calculate fraction of reads in peaks per cell

Usage

```
FRiP(object, peak.assay, bin.assay, chromosome = "chr1", verbose = TRUE)
```

Arguments

object	A Seurat object
peak.assay	Name of the assay containing a peak x cell matrix
bin.assay	Name of the assay containing a bin x cell matrix
chromosome	Which chromosome to use. Default is chromosome 1 ('chr1'). If NULL, use the whole genome.
verbose	Display messages

Value

Returns a Seurat object

Examples

```
FRiP(object = atac_small, peak.assay = 'peaks', bin.assay = 'bins')
```

GenomeBinMatrix *GenomeBinMatrix*

Description

Construct a bin x cell matrix from a fragments file.

Usage

```
GenomeBinMatrix(
  fragments,
  genome,
  cells = NULL,
  binsize = 5000,
  chunk = 50,
  sep = c("-", "-"),
  verbose = TRUE
)
```

Arguments

fragments	Path to tabix-indexed fragments file
genome	A vector of chromosome sizes for the genome. This is used to construct the genome bin coordinates. The can be obtained by calling seqlengths on a BSgenome-class object.
cells	Vector of cells to include. If NULL, include all cells found in the fragments file
binsize	Size of the genome bins to use
chunk	Number of chunks to use when processing the fragments file. Fewer chunks may enable faster processing, but will use more memory.
sep	Vector of separators to use for genomic string. First element is used to separate chromosome and coordinates, second separator is used to separate start and end coordinates.
verbose	Display messages

Details

This function bins the genome and calls FeatureMatrix to construct a bin x cell matrix.

Value

Returns a sparse matrix

Examples

```
gn <- 780007
names(gn) <- 'chr1'
fpath <- system.file("extdata", "fragments.tsv.gz", package="Signac")
GenomeBinMatrix(
  fragments = fpath,
  genome = gn,
  binsize = 1000,
  chunk = 1
)
```

GetCellsInRegion *GetCellsInRegion*

Description

Extract cell names containing reads mapped within a given genomic region

Usage

```
GetCellsInRegion(tabix, region, sep = c("-", "-"), cells = NULL)
```

Arguments

tabix	Tabix object
region	A string giving the region to extract from the fragments file
sep	Vector of separators to use for genomic string. First element is used to separate chromosome and coordinates, second separator is used to separate start and end coordinates.
cells	Vector of cells to include in output. If NULL, include all cells

Value

Returns a list

Examples

```
fpath <- system.file("extdata", "fragments.tsv.gz", package="Signac")
GetCellsInRegion(tabix = fpath, region = "chr1-10245-762629")
```

GetFragments *GetFragments*

Description

Retrieve path to fragments file from assay object, and checks that the file exists and is indexed before returning the file path.

Usage

```
GetFragments(object, assay = NULL)
```

Arguments

object	A Seurat object
assay	Name of the assay use to store the fragments file path

Value

Returns the path to a fragments file stored in the Assay if present

Examples

```
fpath <- system.file("extdata", "fragments.tsv.gz", package="Signac")
atac_small <- SetFragments(object = atac_small, file = fpath)
GetFragments(object = atac_small)
```

`GetIntersectingFeatures` *Find interesecting regions between two objects*

Description

Intersects the regions stored in the rownames of two objects and returns a vector containing the names of rows that interesect for each object. The order of the row names return corresponds to the intersecting regions, ie the nth feature of the first vector will intersect the nth feature in the second vector. A distance parameter can be given, in which case features within the given distance will be called as intersecting.

Usage

```
GetIntersectingFeatures(
  object.1,
  object.2,
  assay.1 = NULL,
  assay.2 = NULL,
  distance = 0,
  sep.1 = c("-", "-"),
  sep.2 = c("-", "-"),
  verbose = TRUE
)
```

Arguments

`object.1`	The first Seurat object
`object.2`	The second Seurat object
`assay.1`	Name of the assay to use in the first object. If NULL, use the default assay
`assay.2`	Name of the assay to use in the second object. If NULL, use the default assay
`distance`	Maximum distance between regions allowed for an intersection to be recorded.
`sep.1`	Genomic coordinate separators to use for the first object
`sep.2`	Genomic coordinate separators to use for the second object
`verbose`	Display messages

Value

Returns a list of two character vectors containing the row names in each object that overlap each other.

Examples

```
GetIntersectingFeatures(
  object.1 = atac_small,
  object.2 = atac_small,
  assay.1 = 'peaks',
  assay.2 = 'bins',
  sep.1 = c(":", "-"),
  sep.2 = c("-", "-")
)
```

`GetMotifData` *Retrieve a motif matrix*

Description

Get motif matrix for given assay

Usage

```
GetMotifData(object, ...)

## S3 method for class 'Motif'
GetMotifData(object, slot = "data", ...)

## S3 method for class 'Assay'
GetMotifData(object, slot = "data", ...)

## S3 method for class 'Seurat'
GetMotifData(object, assay = NULL, slot = "data", ...)
```

Arguments

`object`	A Seurat object
`...`	Arguments passed to other methods
`slot`	Information to pull from object (data, pwm, meta.data)
`assay`	Which assay to use. Default is the current active assay

Value

Returns a `Seurat` object

Examples

```
motif.obj <- GetMotifObject(object = atac_small[['peaks']])
GetMotifData(object = motif.obj)
GetMotifData(object = atac_small[['peaks']])
GetMotifData(object = atac_small)
```

GetMotifObject *Retrieve a Motif object*

Description

Get motif object from given assay

Usage

```
GetMotifObject(object, ...)

## S3 method for class 'Assay'
GetMotifObject(object, ...)

## S3 method for class 'Seurat'
GetMotifObject(object, assay = NULL, ...)
```

Arguments

object	A Seurat object
...	Arguments passed to other methods
assay	Which assay to use. Default is the current active assay

Value

Returns a Motif object

Examples

```
GetMotifObject(object = atac_small[['peaks']])
GetMotifObject(object = atac_small)
```

GetReadsInRegion *GetReadsInRegion*

Description

Extract reads for each cell within a given genomic region or set of regions

Usage

```
GetReadsInRegion(
  object,
  region,
  assay = NULL,
  tabix.file = NULL,
  group.by = NULL,
  cells = NULL,
  verbose = TRUE,
  ...
)
```

Arguments

object	A Seurat object
region	A genomic region, specified as a string in the format 'chr:start-end'. Can be a vector of regions.
assay	Name of assay to use
tabix.file	A TabixFile object. If NULL, the file specified in fragment.path will be opened and closed after the function completes. If iterating over many regions, providing an open TabixFile is much faster as it avoids opening and closing the connection each time.
group.by	Cell grouping information to add
cells	Cells to include. Default is all cells present in the object.
verbose	Display messages
...	Additional arguments passed to StringToGRanges

Value

Returns a data frame

Examples

```
fpath <- system.file("extdata", "fragments.tsv.gz", package="Signac")
atac_small <- SetFragments(object = atac_small, file = fpath)
region <- StringToGRanges(regions = "chr1-10245-762629")
GetReadsInRegion(object = atac_small, region = region)
```

GRangesToString *GRanges to String*

Description

Convert GRanges object to a vector of strings

Usage

```
GRangesToString(grange, sep = c("-", "-"))
```

Arguments

grange	A GRanges object
sep	Vector of separators to use for genomic string. First element is used to separate chromosome and coordinates, second separator is used to separate start and end coordinates.

Value

Returns a character vector

Examples

```
GRangesToString(grange = blacklist_hg19)
```

IntersectMatrix *Intersect genomic coordinates with matrix rows*

Description

Remove or retain matrix rows that intersect given genomic regions

Usage

```
IntersectMatrix(
  matrix,
  regions,
  invert = FALSE,
  sep = c("-", "-"),
  verbose = TRUE,
  ...
)
```

Arguments

matrix	A matrix with genomic regions in the rows
regions	A set of genomic regions to intersect with regions in the matrix. Either a vector of strings encoding the genomic coordinates, or a GRanges object.
invert	Discard rows intersecting the genomic regions supplied, rather than retain.
sep	A length-2 character vector containing the separators to be used for extracting genomic coordinates from a string. The first element will be used to separate the chromosome name from coordinates, and the second element used to separate start and end coordinates.
verbose	Display messages
...	Additional arguments passed to findOverlaps

Value

Returns a sparse matrix

Examples

```
counts <- matrix(data = rep(0, 12), ncol = 2)
rownames(counts) <- c("chr1-565107-565550","chr1-569174-569639",
"chr1-713460-714823","chr1-752422-753038",
"chr1-762106-763359","chr1-779589-780271")
IntersectMatrix(matrix = counts, regions = blacklist_hg19)
```

Jaccard *Calculate the Jaccard index between two matrices*

Description

Finds the Jaccard similarity between rows of the two matricies. Note that the matrices must be binary, and any rows with zero total counts will result in an NaN entry that could cause problems in downstream analyses.

Usage

```
Jaccard(x, y)
```

Arguments

x	The first matrix
y	The second matrix

Details

This will calculate the raw Jaccard index, without normalizing for the expected similarity between cells due to differences in sequencing depth.

Value

Returns a matrix

Examples

```
x <- matrix(data = sample(c(0, 1), size = 25, replace = TRUE), ncol = 5)
Jaccard(x = x, y = x)
```

MatchRegionStats *Match DNA sequence characteristics*

Description

Return a vector if genomic regions that match the distribution of a set of query regions for any given set of characteristics, specified in the input `meta.feature` dataframe.

Usage

```
MatchRegionStats(
  meta.feature,
  regions,
  features.match = c("GC.percent"),
  n = 10000,
  verbose = TRUE,
  ...
)
```

Arguments

`meta.feature`	A dataframe containing DNA sequence information
`regions`	Set of query regions. Must be present in rownames.
`features.match`	Which features of the query to match when selecting a set of regions. A vector of column names present in the feature metadata can be supplied to match multiple characteristics at once. Default is GC content.
`n`	Number of regions to select, with characteristics matching the query
`verbose`	Display messages
`...`	Arguments passed to other functions

Value

Returns a character vector

Examples

```
metafeatures <- Seurat::GetAssayData(
  object = atac_small[['peaks']], slot = 'meta.features'
)
MatchRegionStats(
  meta.feature = metafeatures,
  regions = head(rownames(metafeatures), 10),
  features.match = "percentile",
  n = 10
)
```

`MergeWithRegions` *Region-aware object merging*

Description

This will find intersecting regions in both objects and rename the overlapping features with the region coordinates of the first object (by default; this can be changed with the regions.use parameter).

Usage

```
MergeWithRegions(
  object.1,
  object.2,
  assay.1 = NULL,
  assay.2 = NULL,
  sep.1 = c("-", "-"),
  sep.2 = c("-", "-"),
  regions.use = 1,
  distance = 0,
  new.assay.name = "peaks",
  project = "SeuratProject",
  verbose = TRUE,
  ...
)
```

Arguments

`object.1`	The first Seurat object
`object.2`	The second Seurat object
`assay.1`	Name of the assay to use in the first object. If NULL, use the default assay
`assay.2`	Name of the assay to use in the second object. If NULL, use the default assay
`sep.1`	Genomic coordinate separators to use for the first object
`sep.2`	Genomic coordinate separators to use for the second object
`regions.use`	Which regions to use when naming regions in the merged object. Options are: • 1: Use the region coordinates from the first object • 2: Use the region coordinates from the second object
`distance`	Maximum distance between regions allowed for an intersection to be recorded. Default is 0.
`new.assay.name`	Name for the merged assay. Default is 'peaks'
`project`	Project name for the new object
`verbose`	Display messages
`...`	Additional arguments passed to `CreateAssayObject`

Details

This allows a merged object to be constructed with common feature names.

Value

Returns a Seurat object

Examples

```
MergeWithRegions(
  object.1 = atac_small,
  object.2 = atac_small,
  assay.1 = 'peaks',
  assay.2 = 'bins',
  sep.1 = c(":","-"),
  sep.2 = c("-","-")
)
```

Motif-class *The Motif class*

Description

The Motif class stores DNA motif information

Slots

`data` A sparse, binary, feature x motif matrix. Columns correspond to motif IDs, rows correspond to genomic features (peaks or bins). Entries in the matrix should be 1 if the genomic feature contains the motif, and 0 otherwise.

`pwm` A named list of position weight matrices

`motif.names` A list containing the name of each motif

`meta.data` A dataframe for storage of additional information related to each motif. This could include the names of proteins that bind the motif.

MotifPlot *MotifPlot*

Description

Plot motifs

Usage

```
MotifPlot(object, motifs, assay = NULL, use.names = TRUE, ...)
```

Arguments

object	A Seurat object
motifs	A list of motifs to plot
assay	Name of the assay to use
use.names	Use motif names stored in the motif object
...	Additional parameters passed to ggseqlogo

Value

Returns a ggplot object

Examples

```
motif.obj <- GetMotifObject(atac_small)
MotifPlot(atac_small, motifs = head(colnames(motif.obj)))
```

NucleosomeSignal *NucleosomeSignal*

Description

Calculate the strength of the nucleosome signal per cell. Computes the ratio of fragments between 147 bp and 294 bp (mononucleosome) to fragments < 147 bp (nucleosome-free)

Usage

```
NucleosomeSignal(
  object,
  assay = NULL,
  region = "chr1-1-249250621",
  min.threshold = 147,
  max.threshold = 294,
  verbose = TRUE,
  ...
)
```

Arguments

object	A Seurat object
assay	Name of assay to use. Only required if a fragment path is not provided. If NULL, use the active assay.
region	Which region to use. Can be a GRanges region, a string, or a vector of strings. Default is human chromosome 1.
min.threshold	Lower bound for the mononucleosome size. Default is 147
max.threshold	Upper bound for the mononucleosome size. Default is 294
verbose	Display messages
...	Additional arguments passed to GetReadsInRegion

Value

Returns a Seurat object with added metadata for the ratio of mononucleosomal to nucleosome-free fragments per cell, and the percentile rank of each ratio.

Examples

```
fpath <- system.file("extdata", "fragments.tsv.gz", package="Signac")
atac_small <- SetFragments(object = atac_small, file = fpath)
NucleosomeSignal(object = atac_small)
```

RegionStats *Compute base composition information for genomic ranges*

Description

Compute the GC content, region lengths, and dinucleotide base frequencies for regions in the assay and add to the feature metadata.

Usage

```
RegionStats(object, ...)

## Default S3 method:
RegionStats(object, genome, sep = c("-", "-"), verbose = TRUE, ...)

## S3 method for class 'Assay'
RegionStats(object, genome, sep = c("-", "-"), verbose = TRUE, ...)

## S3 method for class 'Seurat'
RegionStats(
  object,
  genome,
  assay = NULL,
  sep = c("-", "-"),
  verbose = TRUE,
  ...
)
```

Arguments

object	A Seurat object, Assay object, or set of genomic ranges
...	Arguments passed to other methods
genome	A BSgenome object
sep	A length-2 character vector containing the separators to be used when constructing genomic coordinates from the regions. The first element is used to separate the chromosome from the genomic coordinates, and the second element used to separate the start and end coordinates.
verbose	Display messages
assay	Name of assay to use

Value

Returns a dataframe

Examples

```
## Not run:
library(BSgenome.Hsapiens.UCSC.hg19)
RegionStats(
object = rownames(atac_small),
genome = BSgenome.Hsapiens.UCSC.hg19, sep = c(":", "-")
)

## End(Not run)
## Not run:
library(BSgenome.Hsapiens.UCSC.hg19)
RegionStats(
```

```
object = atac_small[['peaks']],
genome = BSgenome.Hsapiens.UCSC.hg19, sep = c(":", "-")
)

## End(Not run)
## Not run:
library(BSgenome.Hsapiens.UCSC.hg19)
RegionStats(
  object = atac_small,
  assay = 'bins',
  genome = BSgenome.Hsapiens.UCSC.hg19
)

## End(Not run)
```

RunChromVAR *Run chromVAR*

Description

Wrapper to run chromVAR on an assay with a motif object present. Will return a new Seurat assay with the motif activities (the deviations in chromatin accessibility across the set of regions) as a new assay.

Usage

```
RunChromVAR(
  object,
  genome,
  new.assay.name = "chromvar",
  motif.matrix = NULL,
  assay = NULL,
  sep = c(":", "-"),
  verbose = TRUE,
  ...
)
```

Arguments

object	A Seurat object
genome	A BSgenome object
new.assay.name	Name of new assay used to store the chromVAR results. Default is "chromvar".
motif.matrix	A peak x motif matrix. If NULL, pull the peak x motif matrix from a Motif object stored in the assay.
assay	Name of assay to use
sep	A length-2 character vector containing the separators passed to StringToGRanges.
verbose	Display messages
...	Additional arguments passed to getBackgroundPeaks

Details

See the chromVAR documentation for more information: https://greenleaflab.github.io/chromVAR/index.html

See the chromVAR paper: https://www.nature.com/articles/nmeth.4401

Value

Returns a Seurat object with a new assay

Examples

```
## Not run:
library(BSgenome.Hsapiens.UCSC.hg19)
RunChromVAR(object = atac_small, genome = BSgenome.Hsapiens.UCSC.hg19)

## End(Not run)
```

RunSVD *Run singular value decomposition*

Description

Run partial singular value decomposition using irlba

Usage

```
RunSVD(object, ...)

## Default S3 method:
RunSVD(
  object,
  assay = NULL,
  n = 50,
  scale.embeddings = TRUE,
  reduction.key = "SVD_",
  scale.max = NULL,
  verbose = TRUE,
  irlba.work = n + 50,
  ...
)

## S3 method for class 'Assay'
RunSVD(
  object,
  assay = NULL,
  features = NULL,
  n = 50,
```

```
  reduction.key = "SVD_",
  scale.max = NULL,
  verbose = TRUE,
  ...
)

## S3 method for class 'Seurat'
RunSVD(
  object,
  assay = NULL,
  features = NULL,
  n = 50,
  reduction.key = "SVD_",
  reduction.name = "svd",
  scale.max = NULL,
  verbose = TRUE,
  ...
)
```

Arguments

object	A Seurat object
...	Arguments passed to other methods
assay	Which assay to use. If NULL, use the default assay
n	Number of singular values to compute
scale.embeddings	Scale cell embeddings within each component to mean 0 and SD 1 (default TRUE).
reduction.key	Key for dimension reduction object
scale.max	Clipping value for cell embeddings. Default (NULL) is no clipping.
verbose	Print messages
irlba.work	work parameter for irlba. Working subspace dimension, larger values can speed convergence at the cost of more memory use.
features	Which features to use. If NULL, use variable features
reduction.name	Name for stored dimension reduction object. Default 'svd'

Value

Returns a Seurat object

Examples

```
x <- matrix(data = rnorm(100), ncol = 10)
RunSVD(x)
RunSVD(atac_small[['peaks']])
RunSVD(atac_small)
```

RunTFIDF *Compute the term-frequency inverse-document-frequency*

Description

Run term frequency inverse document frequency (TF-IDF) normalization

Usage

```
RunTFIDF(object, ...)

## Default S3 method:
RunTFIDF(
  object,
  assay = NULL,
  method = 1,
  scale.factor = 10000,
  verbose = TRUE,
  ...
)

## S3 method for class 'Assay'
RunTFIDF(
  object,
  assay = NULL,
  method = 1,
  scale.factor = 10000,
  verbose = TRUE,
  ...
)

## S3 method for class 'Seurat'
RunTFIDF(
  object,
  assay = NULL,
  method = 1,
  scale.factor = 10000,
  verbose = TRUE,
  ...
)
```

Arguments

object	A Seurat object
...	Arguments passed to other methods
assay	Name of assay to use

method	Which TF-IDF implementation to use. Choice of: • 1: The LSI implementation used by Stuart & Butler et al. 2019 (https://doi.org/10.1101/460147). • 2: The standard LSI implementation used by Cusanovich & Hill et al. 2018 (https://doi.org/10.1016/j.cell.2018.06.052). • 3: The log-TF method • 4: The 10x Genomics method (no TF normalization)
scale.factor	Which scale factor to use. Default is 10000.
verbose	Print progress

Value

Returns a Seurat object

Examples

```
mat <- matrix(data = rbinom(n = 25, size = 5, prob = 0.2), nrow = 5)
RunTFIDF(object = mat)
RunTFIDF(atac_small[['peaks']])
RunTFIDF(object = atac_small)
```

SetFragments *Set the fragments file path for creating plots*

Description

Give path of indexed fragments file that goes with data in the object. Checks for a valid path and an index file with the same name (.tbi) at the same path. Stores the path under the tools slot for access by visualization functions. One fragments file can be stored for each assay.

Usage

```
SetFragments(object, file, assay = NULL)
```

Arguments

object	A Seurat object
file	Path to indexed fragment file. See https://support.10xgenomics.com/single-cell-atac/software/pipelines/latest/output/fragments
assay	Assay used to generate the fragments. If NULL, use the active assay.

Value

Returns a Seurat object

Examples

```
fpath <- system.file("extdata", "fragments.tsv.gz", package="Signac")
SetFragments(object = atac_small, file = fpath)
```

SetMotifData *Set motif data*

Description

Set motif matrix for given assay

Usage

```
SetMotifData(object, ...)

## S3 method for class 'Motif'
SetMotifData(object, slot, new.data, ...)

## S3 method for class 'Assay'
SetMotifData(object, slot, new.data, ...)

## S3 method for class 'Seurat'
SetMotifData(object, assay = NULL, ...)
```

Arguments

object	A Seurat object
...	Arguments passed to other methods
slot	Name of slot to use
new.data	motif matrix to add. Should be matrix or sparse matrix class
assay	Name of assay whose data should be set

Value

Returns a Seurat object

Examples

```
motif.obj <- GetMotifObject(object = atac_small)
SetMotifData(object = motif.obj, slot = 'data', new.data = matrix())
SetMotifData(
object = atac_small[['peaks']], slot = 'data', new.data = matrix()
)
motif.matrix <- GetMotifData(object = atac_small)
SetMotifData(
object = atac_small, assay = 'peaks', slot = 'data', new.data = motif.matrix
)
```

SingleCoveragePlot *Plot Tn5 insertion sites over a region*

Description

Plot fragment coverage (frequence of Tn5 insertion) within given regions for groups of cells.

Usage

```
SingleCoveragePlot(
  object,
  region,
  annotation = NULL,
  peaks = NULL,
  assay = NULL,
  fragment.path = NULL,
  group.by = NULL,
  window = 100,
  downsample = 0.1,
  height.tracks = 10,
  extend.upstream = 0,
  extend.downstream = 0,
  ymax = NULL,
  scale.factor = NULL,
  cells = NULL,
  idents = NULL,
  sep = c("-", "-")
)

CoveragePlot(
  object,
  region,
  annotation = NULL,
  peaks = NULL,
  assay = NULL,
  fragment.path = NULL,
  group.by = NULL,
  window = 100,
  downsample = 0.1,
  height.tracks = 10,
  extend.upstream = 0,
  extend.downstream = 0,
  scale.factor = NULL,
  ymax = NULL,
  cells = NULL,
  idents = NULL,
  sep = c("-", "-"),
```

```
  ...
)
```

Arguments

object	A Seurat object
region	A set of genomic coordinates to show. Can be a GRanges object, a string, or a vector of strings describing the genomic coordinates to plot.
annotation	An Ensembl based annotation package
peaks	A GRanges object containing peak coordinates
assay	Name of the assay to plot
fragment.path	Path to an index fragment file. If NULL, will look for a path stored for the requested assay using the SetFragments function
group.by	Name of one or more metadata columns to group (color) the cells by. Default is the current cell identities
window	Smoothing window size
downsample	Fraction of positions to retain in the plot.
height.tracks	Height of the accessibility tracks relative to the height of the gene annotation track.
extend.upstream	Number of bases to extend the region upstream.
extend.downstream	Number of bases to extend the region downstream.
ymax	Maximum value for Y axis. If NULL (default) set to the highest value among all the tracks.
scale.factor	Scaling factor for track height. If NULL (default), use the median group scaling factor determined by total number of fragments sequences in each group.
cells	Which cells to plot. Default all cells
idents	Which identities to include in the plot. Default is all identities.
sep	Separators to use for strings encoding genomic coordinates. First element is used to separate the chromosome from the coordinates, second element is used to separate the start from end coordinate.
...	Additional arguments passed to wrap_plots

Details

Thanks to Andrew Hill for providing an early version of this function http://andrewjohnhill.com/blog/2019/04/12/streamlining-scatac-seq-visualization-and-analysis/

Value

Returns a ggplot object

Examples

```
fpath <- system.file("extdata", "fragments.tsv.gz", package="Signac")
atac_small <- SetFragments(atac_small, file = fpath)
CoveragePlot(object = atac_small, region = c("chr1-713500-714500"))
```

`StringToGRanges` *String to GRanges*

Description

Convert a genomic coordinate string to a GRanges object

Usage

```
StringToGRanges(regions, sep = c("-", "-"), ...)
```

Arguments

`regions`	Vector of genomic region strings
`sep`	Vector of separators to use for genomic string. First element is used to separate chromosome and coordinates, second separator is used to separate start and end coordinates.
`...`	Additional arguments passed to `makeGRangesFromDataFrame`

Value

Returns a GRanges object

Examples

```
regions <- c('chr1-1-10', 'chr2-12-3121')
StringToGRanges(regions = regions)
```

`subset.Motif` *Return a subset of a Motif object*

Description

Return a subset of a Motif object

Usage

```
## S3 method for class 'Motif'
subset(x, features = NULL, motifs = NULL, ...)

## S3 method for class 'Motif'
x[i, j, ...]
```

Arguments

`x`	A Motif object
`features`	Which features to retain
`motifs`	Which motifs to retain
`...`	Arguments passed to other methods
`i`	Which columns to retain
`j`	Which rows to retain

Value

Returns a subsetted `Motif` object

See Also

`subset`

Examples

```
motif.obj <- GetMotifObject(object = atac_small)
subset(x = motif.obj, features = head(rownames(motif.obj), 10))
motif.obj <- GetMotifObject(atac_small)
motif.obj[1:10,1:5]
```

SubsetMatrix *Subset matrix rows and columns*

Description

Subset the rows and columns of a matrix by removing rows and columns with less than the specified number of non-zero elements.

Usage

```
SubsetMatrix(
  mat,
  min.rows = 1,
  min.cols = 1,
  max.row.val = 10,
  max.col.val = NULL
)
```

Arguments

mat	A matrix
min.rows	Minimum number of non-zero elements for the row to be retained
min.cols	Minimum number of non-zero elements for the column to be retained
max.row.val	Maximum allowed value in a row for the row to be retained. If NULL, don't set any limit.
max.col.val	Maximum allowed value in a column for the column to be retained. If NULL, don't set any limit.

Value

Returns a matrix

Examples

```
SubsetMatrix(mat = volcano)
```

TSSEnrichment *Compute TSS enrichment score per cell*

Description

Compute the transcription start site (TSS) enrichment score for each cell, as defined by ENCODE: https://www.encodeproject.org/data-standards/terms/.

Usage

```
TSSEnrichment(
  object,
  tss.positions,
  assay = NULL,
  cells = NULL,
  verbose = TRUE
)
```

Arguments

object	A Seurat object
tss.positions	A GRanges object containing the TSS positions
assay	Name of assay to use
cells	A vector of cells to include. If NULL (default), use all cells in the object
verbose	Display messages

Details

The computed score will be added to the object metadata as "TSS.enrichment".

Value

Returns a Seurat object

Examples

```
## Not run:
library(EnsDb.Hsapiens.v75)
gene.ranges <- genes(EnsDb.Hsapiens.v75)
gene.ranges <- gene.ranges[gene.ranges$gene_biotype == 'protein_coding', ]
tss.ranges <- GRanges(
  seqnames = seqnames(gene.ranges),
  ranges = IRanges(start = start(gene.ranges), width = 2),
  strand = strand(gene.ranges)
)
seqlevelsStyle(tss.ranges) <- 'UCSC'
tss.ranges <- keepStandardChromosomes(tss.ranges, pruning.mode = 'coarse')
```

```
fpath <- system.file("extdata", "fragments.tsv.gz", package="Signac")
atac_small <- SetFragments(object = atac_small, file = fpath)
TSSEnrichment(object = atac_small, tss.positions = tss.ranges[1:100])

## End(Not run)
```

TSSPlot *Plot the enrichment around TSS*

Description

Plot the normalized TSS enrichment score at each position relative to the TSS. Requires that TSSEnrichment has already been run on the assay.

Usage

```
TSSPlot(object, assay = NULL, group.by = NULL, idents = NULL)
```

Arguments

object	A Seurat object
assay	Name of the assay to use. Should have the TSS enrichment information for each cell already computed by running TSSEnrichment
group.by	Set of identities to group cells by
idents	Set of identities to include in the plot

Value

Returns a ggplot2 object

Examples

```
## Not run:
# create granges object with TSS positions
library(EnsDb.Hsapiens.v75)
gene.ranges <- genes(EnsDb.Hsapiens.v75)
gene.ranges <- gene.ranges[gene.ranges$gene_biotype == 'protein_coding', ]
tss.ranges <- GRanges(
  seqnames = seqnames(gene.ranges),
  ranges = IRanges(start = start(gene.ranges), width = 2),
  strand = strand(gene.ranges)
)
seqlevelsStyle(tss.ranges) <- 'UCSC'
tss.ranges <- keepStandardChromosomes(tss.ranges, pruning.mode = 'coarse')

# to save time use the first 2000 TSSs
atac_small <- TSSEnrichment(
object = atac_small,
```

```
tss.positions = tss.ranges[1:2000]
)
TSSPlot(atac_small)

## End(Not run)
```

`UnifyPeaks` *Unify genomic ranges*

Description

Create a unified set of non-overlapping genomic ranges from multiple Seurat objects containing single-cell chromatin data.

Usage

```
UnifyPeaks(object.list, mode = "reduce", sep = c(":", "-"))
```

Arguments

`object.list`	A list of Seurat objects
`mode`	Function to use when combining genomic ranges. Can be "reduce" (default) or "disjoin". See `reduce` and `disjoin` for more information on these functions.
`sep`	Separators to use to extract genomic ranges from object row names. To specify different separators for different objects, pass a list of length equal to the length of `object.list`.

Value

Returns a GRanges object

Examples

```
UnifyPeaks(object.list = list(atac_small, atac_small))
```

Index