Package ‘Signac’

May 30, 2023

Title Analysis of Single-Cell Chromatin Data
Version 1.10.0
Date 2023-05-26
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 et al. (2021) <doi:10.1038/s41592-021-01282-5>.
Depends R (>= 4.0.0), methods
License MIT + file LICENSE
Encoding UTF-8
LazyData true
RoxygenNote 7.2.3
BugReports https://github.com/stuart-lab/signac/issues
LinkingTo Rcpp
Imports GenomInfoDb (> = 1.29.3), GenomicRanges, IRanges, Matrix, Rsamtools, S4Vectors, SeuratObject (> = 4.0.0), data.table, dplyr (> = 1.0.0), future, future.apply, ggplot2, rlang, irlba, pbapply, tidyrr, patchwork, stats, utils, BiocGenerics, stringi, fastmatch, RcppRoll, scales, Rcpp, grid, tidyselect, vctrs
Collate 'RcppExports.R' 'data.R' 'differential_accessibility.R'
   'generics.R' 'dimension_reduction.R' 'footprinting.R'
   'fragments.R' 'genomeinfodb-methods.R' 'granges-methods.R'
   'heatmaps.R' 'iranges-methods.R' 'links.R' 'mito.R' 'motifs.R'
   'objects.R' 'peaks.R' 'preprocessing.R' 'quantification.R'
   'region-enrichment.R' 'utilities.R' 'visualization.R' 'zzz.R'
Suggests Seurat (>= 4.0.6), ggforce, ggrepel, ggseqlogo, testthat (>= 2.1.0), chromVAR, SummarizedExperiment, TFBSTools, motifmatchr, BSgenome, shiny, miniUI, rtracklayer, biovizBase, Biostrings, lsa, qlcMatrix, MASS
NeedsCompilation: yes

Author: Tim Stuart [aut, cre]
        (https://orcid.org/0000-0002-3044-0897),
        Avi Srivastava [aut]
        (https://orcid.org/0000-0001-9798-2079),
        Paul Hoffman [ctb]
        (https://orcid.org/0000-0002-7693-8957),
        Rahul Satija [ctb]
        (https://orcid.org/0000-0001-9448-8833)

Maintainer: Tim Stuart <stuartt@gis.a-star.edu.sg>

Repository: CRAN

Date/Publication: 2023-05-30 08:50:06 UTC

R topics documented:

- Signac-package
- AccessiblePeaks
- AddChromatinModule
- AddMotifs
- AggregateTiles
- AlleleFreq
- Annotation
- AnnotationPlot
- as.ChromatinAssay
- atac_small
- AverageCounts
- BigwigTrack
- BinarizeCounts
- blacklist_ce10
- blacklist_ce11
- blacklist_dm3
- blacklist_dm6
- blacklist_hg19
- blacklist_hg38
- blacklist_hg38_unified
- blacklist_mm10
- CallPeaks
- Cells.Fragment
- Cells<-
- CellsPerGroup
- ChromatinAssay-class
- ClosestFeature
- ClusterClonotypes
- CombineTracks
- ConnectionsToLinks
- ConvertMotifID
- CountFragments
- CountsInRegion
- coverage.ChromatinAssay-method
- CoverageBrowser

R topics documented:
R topics documented:

- CoveragePlot .......................... 32
- CreateChromatinAssay .................. 37
- CreateFragmentObject .................. 38
- CreateMotifMatrix ..................... 39
- CreateMotifObject ..................... 40
- DensityScatter ......................... 41
- DepthCor ............................. 42
- DownsampleFeatures ................... 43
- ExpressionPlot ......................... 43
- Extend ................................ 44
- FeatureMatrix ......................... 45
- FilterCells ............................ 46
- FindClonotypes ......................... 47
- FindMotifs ................................ 48
- findOverlaps-methods .................. 49
- FindTopFeatures ....................... 53
- Footprint ................................ 54
- FractionCountsInRegion .............. 55
- Fragment-class ......................... 56
- FragmentHistogram ..................... 56
- Fragments ................................ 57
- FRiP .................................. 59
- GeneActivity .......................... 59
- GenomeBinMatrix ...................... 61
- GetCellsInRegion ....................... 62
- GetFootprintData ...................... 63
- GetFragmentData ....................... 63
- GetGRangesFromEnsDb .................. 64
- GetIntersectingFeatures .............. 64
- GetLinkedGenes ......................... 65
- GetLinkedPeaks ......................... 66
- GetMotifData ........................... 66
- GetTSSPositions ....................... 67
- granges-methods ....................... 68
- GRangesToString ....................... 69
- head.Fragments ......................... 69
- IdentifyVariants ....................... 70
- InsertionBias .......................... 71
- inter-range-methods ................... 72
- IntersectMatrix ....................... 74
- Jaccard ................................ 75
- LinkPeaks ............................. 75
- LinkPlot ................................ 77
- Links .................................. 78
- LookupGeneCoords ...................... 79
- MatchRegionStats ...................... 79
- Motif-class ............................ 80
- MotifCounts ............................ 81
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 et al. (2021) doi:10.1038/s41592021012825.
Author(s)

Maintainer: Tim Stuart <stuartt@gis.a-star.edu.sg> (ORCID)
Authors:

• Avi Srivastava <asrivastava@nygenome.org> (ORCID)

Other contributors:

• Paul Hoffman <phoffman@nygenome.org> (ORCID) [contributor]
• Rahul Satija <rsatija@nygenome.org> (ORCID) [contributor]

See Also

Useful links:

• [https://github.com/stuart-lab/signac](https://github.com/stuart-lab/signac)
• [https://stuartlab.org/signac](https://stuartlab.org/signac)
• Report bugs at [https://github.com/stuart-lab/signac/issues](https://github.com/stuart-lab/signac/issues)

---

**AccessiblePeaks**

**Accessible peaks**

---

**Description**

Find accessible peaks in a set of cells

**Usage**

```r
AccessiblePeaks(
  object,
  assay = NULL,
  idents = NULL,
  cells = NULL,
  min.cells = 10
)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>A Seurat object</td>
</tr>
<tr>
<td>assay</td>
<td>Name of assay to use</td>
</tr>
<tr>
<td>idents</td>
<td>A set of identity classes to find accessible peaks for</td>
</tr>
<tr>
<td>cells</td>
<td>A vector of cells to find accessible peaks for</td>
</tr>
<tr>
<td>min.cells</td>
<td>Minimum number of cells with the peak accessible (&gt;0 counts) for the peak to be called accessible</td>
</tr>
</tbody>
</table>

**Value**

Returns a vector of peak names
AddChromatinModule  

**Description**

Compute chromVAR deviations for groups of peaks. The goal of this function is similar to that of AddModuleScore except that it is designed for single-cell chromatin data. The chromVAR deviations for each group of peaks will be added to the object metadata.

**Usage**

AddChromatinModule(object, features, genome, assay = NULL, verbose = TRUE, ...)

**Arguments**

- **object**: A Seurat object
- **features**: A named list of features to include in each module. The name of each element in the list will be used to name the modules computed, which will be stored in the object metadata.
- **genome**: A BSgenome object
- **assay**: Name of assay to use. If NULL, use the default assay.
- **verbose**: Display messages
- **...**: Additional arguments passed to RunChromVAR

**Value**

Returns a Seurat object

---

AddMotifs  

**Description**

Construct a Motif object containing DNA sequence motif information and add it to an existing Seurat object or ChromatinAssay. If running on a Seurat object, AddMotifs will also run RegionStats to compute the GC content of each peak and store the results in the feature metadata. PFMs or PWMs are matched to the genome sequence using the matchMotifs function with default parameters to construct a matrix of motif positions in genomic regions.
AggregateTiles

Usage

AddMotifs(object, ...)

## Default S3 method:
AddMotifs(object, genome, pfm, verbose = TRUE, ...)

## S3 method for class 'ChromatinAssay'
AddMotifs(object, genome, pfm, verbose = TRUE, ...)

## S3 method for class 'Assay'
AddMotifs(object, genome, pfm, verbose = TRUE, ...)

## S3 method for class 'Seurat'
AddMotifs(object, genome, pfm, assay = NULL, verbose = TRUE, ...)

Arguments

object A Seurat object or ChromatinAssay object
...
... Additional arguments passed to other methods
genome A BSgenome, DNAStringSet, FaFile, or string stating the genome build recognized by getBSgenome.
pfm A PFMatrixList or PWMatrixList object containing position weight/frequency matrices to use
verbose Display messages
assay Name of assay to use. If NULL, use the default assay

Value

When running on a ChromatinAssay or Seurat object, returns a modified version of the input object. When running on a matrix, returns a Motif object.

See Also

motifmatchr

AggregateTiles

Quantify aggregated genome tiles

Description

Quantifies fragment counts per cell in fixed-size genome bins across the whole genome, then removes bins with less than a desired minimum number of counts in the bin, then merges adjacent tiles into a single region.
AggregateTiles

Usage

AggregateTiles(object, ...) 

## S3 method for class 'Seurat'
AggregateTiles(
  object,
  genome,
  assay = NULL,
  new.assay.name = "tiles",
  min_counts = 5,
  binsize = 5000,
  verbose = TRUE,
  ...
)

## S3 method for class 'ChromatinAssay'
AggregateTiles(
  object,
  genome,
  min_counts = 5,
  binsize = 5000,
  verbose = TRUE,
  ...
)

## Default S3 method:
AggregateTiles(
  object,
  genome,
  cells = NULL,
  min_counts = 5,
  binsize = 5000,
  verbose = TRUE,
  ...
)

Arguments

- **object**: A Seurat object or ChromatinAssay object
- **...**: Additional arguments passed to other methods
- **genome**: 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.
- **assay**: Name of assay to use
- **new.assay.name**: Name of new assay to create containing aggregated genome tiles
- **min_counts**: Minimum number of counts for a tile to be retained prior to aggregation
- **binsize**: Size of the genome bins (tiles) in base pairs
AlleleFreq  

verbose  Display messages  
cells  Cells to include  

Value  
When running on a Seurat object, returns the Seurat object with a new ChromatinAssay added.  
When running on a ChromatinAssay, returns a new ChromatinAssay containing the aggregated genome tiles.  
When running on a fragment file, returns a sparse region x cell matrix.  

AlleleFreq  Compute allele frequencies per cell  

Description  
Collapses allele counts for each strand and normalize by the total number of counts at each nucleotide position.  

Usage  
AlleleFreq(object, ...)  
## Default S3 method:  
AlleleFreq(object, variants, ...)  
## S3 method for class 'Assay'  
AlleleFreq(object, variants, ...)  
## S3 method for class 'Seurat'  
AlleleFreq(object, variants, assay = NULL, new.assay.name = "alleles", ...)  

Arguments  
object  A Seurat object, Assay, or matrix  
...  Arguments passed to other methods  
variants  A character vector of informative variants to keep. For example, c("627G>A", "709G>A", "1045G>A", "1793G>A").  
assay  Name of assay to use  
new.assay.name  Name of new assay to store variant data in  

Value  
Returns a Seurat object with a new assay containing the allele frequencies for the informative variants.
Description
Get the annotation from a ChromatinAssay

Usage
Annotation(object, ...)
Annotation(object, ...) <- value

## S3 method for class 'ChromatinAssay'
Annotation(object, ...)

## S3 method for class 'Seurat'
Annotation(object, ...)

## S3 replacement method for class 'ChromatinAssay'
Annotation(object, ...) <- value

## S3 replacement method for class 'Seurat'
Annotation(object, ...) <- value

Arguments

<table>
<thead>
<tr>
<th>object</th>
<th>A Seurat object or ChromatinAssay object</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>Arguments passed to other methods</td>
</tr>
<tr>
<td>value</td>
<td>A value to set. Can be NULL, to remove the current annotation information, or a GRanges object. If a GRanges object is supplied and the genome information is stored in the assay, the genome of the new annotations must match the genome of the assay.</td>
</tr>
</tbody>
</table>

Value

Returns a GRanges object if the annotation data is present, otherwise returns NULL

Examples

Annotation(atac_small[["peaks"]])

Annotation(atac_small)

genes <- Annotation(atac_small)
AnnotationPlot

Annotation(atac_small["peaks"])) <- genes
genes <- Annotation(atac_small)
Annotation(atac_small) <- genes

---

**Description**

Display gene annotations in a given region of the genome.

**Usage**

```r
AnnotationPlot(
  object, 
  region, 
  assay = NULL, 
  mode = "gene", 
  sep = c("-", "-", "-"),
  extend.upstream = 0,
  extend.downstream = 0
)
```

**Arguments**

- `object`: A Seurat object
- `region`: A genomic region to plot
- `assay`: Name of assay to use. If NULL, use the default assay.
- `mode`: Display mode. Choose either "gene" or "transcript" to determine whether genes or transcripts are plotted.
- `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.
- `extend.upstream`: Number of bases to extend the region upstream.
- `extend.downstream`: Number of bases to extend the region downstream.

**Value**

Returns a ggplot object

**Examples**

```r
AnnotationPlot(object = atac_small, region = c("chr1-29554-39554"))
```
as.ChromatinAssay

Convert objects to a ChromatinAssay

Description

Convert objects to a ChromatinAssay

Usage

as.ChromatinAssay(x, ...)

## S3 method for class 'Assay'
as.ChromatinAssay(
  x,
  ranges = NULL,
  seqinfo = NULL,
  annotation = NULL,
  motifs = NULL,
  fragments = NULL,
  bias = NULL,
  positionEnrichment = NULL,
  sep = c("-", "-"),
  ...
)

Arguments

- **x**: An object to convert to class ChromatinAssay
- **...**: Arguments passed to other methods
- **ranges**: A GRanges object
- **seqinfo**: A Seqinfo object containing basic information about the genome used. Alternatively, the name of a UCSC genome can be provided and the sequence information will be downloaded from UCSC.
- **annotation**: Genomic annotation
- **motifs**: A Motif object
- **fragments**: A list of Fragment objects
- **bias**: Tn5 integration bias matrix
- **positionEnrichment**: A named list of position enrichment matrices.
- **sep**: Characters used to separate the chromosome, start, and end coordinates in the row names of the data matrix
**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)
BigwigTrack (Plot data from BigWig files)

Description
Create coverage tracks, heatmaps, or line plots from bigwig files.

Usage
BigwigTrack(
  region,  
  bigwig, 
  smooth = 200, 
  extend.upstream = 0, 
  extend.downstream = 0, 
  type = "coverage", 
  y_label = "bigWig", 
  bigwig.scale = "common", 
  ymax = NULL, 
  max.downsample = 3000, 
  downsample.rate = 0.1 
)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>region</td>
<td>GRanges object specifying region to plot</td>
</tr>
<tr>
<td>bigwig</td>
<td>List of bigwig file paths. List should be named, and the name of each element in the list of files will be displayed alongside the track in the final plot.</td>
</tr>
<tr>
<td>smooth</td>
<td>Number of bases to smooth data over (rolling mean). If NULL, do not apply smoothing.</td>
</tr>
<tr>
<td>extend.upstream</td>
<td>Number of bases to extend the region upstream.</td>
</tr>
<tr>
<td>extend.downstream</td>
<td>Number of bases to extend the region downstream.</td>
</tr>
<tr>
<td>type</td>
<td>Plot type. Can be one of &quot;line&quot;, &quot;heatmap&quot;, or &quot;coverage&quot;</td>
</tr>
<tr>
<td>y_label</td>
<td>Y-axis label</td>
</tr>
<tr>
<td>bigwig.scale</td>
<td>Scaling to apply to data from different bigwig files. Can be:</td>
</tr>
<tr>
<td></td>
<td>• common: plot each bigwig on a common scale (default)</td>
</tr>
<tr>
<td></td>
<td>• separate: plot each bigwig on a separate scale ranging from zero to the maximum value for that bigwig file within the plotted region</td>
</tr>
<tr>
<td>ymax</td>
<td>Maximum value for Y axis. Can be one of:</td>
</tr>
<tr>
<td></td>
<td>• NULL: set to the highest value among all the tracks (default)</td>
</tr>
</tbody>
</table>
• qXX: clip the maximum value to the XX quantile (for example, q95 will set the maximum value to 95% of the maximum value in the data). This can help remove the effect of extreme values that may otherwise distort the scale.
• numeric: manually define a Y-axis limit

max.downsample
Minimum number of positions kept when downsampling. Downsampling rate is adaptive to the window size, but this parameter will set the minimum possible number of positions to include so that plots do not become too sparse when the window size is small.

downsampling.rate
Fraction of positions to retain when downsampling. Retaining more positions can give a higher-resolution plot but can make the number of points large, resulting in larger file sizes when saving the plot and a longer period of time needed to draw the plot.

Details

Note that this function does not work on windows.

Value

Returns a ggplot object

---

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, ...)

blacklist_ce10

Arguments

- **object**: A Seurat object
- **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

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

blacklist_ce10

Genomic blacklist regions for C. elegans ce10 (0-based)

Description

Genomic blacklist regions for C. elegans ce10 (0-based)

Usage

blacklist_ce10

Format

A GRanges object

Source

https://github.com/Boyle-Lab/Blacklist
doi:10.1038/s4159801945839z
blacklist_ce11  Genomic blacklist regions for C. elegans ce11 (0-based)

Description
Genomic blacklist regions for C. elegans ce11 (0-based)

Usage
blacklist_ce11

Format
A GRanges object

Source
https://github.com/Boyle-Lab/Blacklist
doi:10.1038/s4159801945839z

blacklist_dm3  Genomic blacklist regions for Drosophila dm3 (0-based)

Description
Genomic blacklist regions for Drosophila dm3 (0-based)

Usage
blacklist_dm3

Format
A GRanges object

Source
https://github.com/Boyle-Lab/Blacklist
doi:10.1038/s4159801945839z
blacklist_dm6  Genomic blacklist regions for Drosophila dm6 (0-based)

Description
Genomic blacklist regions for Drosophila dm6 (0-based)

Usage
blacklist_dm6

Format
A GRanges object

Source
https://github.com/Boyle-Lab/Blacklist
doi:10.1038/s4159801945839z

blacklist_hg19  Genomic blacklist regions for Human hg19 (0-based)

Description
Genomic blacklist regions for Human hg19 (0-based)

Usage
blacklist_hg19

Format
A GRanges object

Source
https://github.com/Boyle-Lab/Blacklist
doi:10.1038/s4159801945839z
blacklist_hg38

**Description**
Genomic blacklist regions for Human GRCh38

**Usage**
blacklist_hg38

**Format**
A GRanges object

**Source**
https://github.com/Boyle-Lab/Blacklist
doi:10.1038/s4159801945839z

blacklist_hg38_unified

*Unified genomic blacklist regions for Human GRCh38*

**Description**
Manually curated genomic blacklist regions for the hg38 genome by Anshul Kundaje and Anna Shcherbina. See https://www.encodeproject.org/files/ENCFF356LFX/ for a description of how this blacklist was curated.

**Usage**
blacklist_hg38_unified

**Format**
A GRanges object

**Author(s)**
Anshul Kundaje
Anna Shcherbina

**Source**
https://www.encodeproject.org/files/ENCFF356LFX/
doi:10.1038/s4159801945839z
blacklist_mm10  Genomic blacklist regions for Mouse mm10 (0-based)

Description
Genomic blacklist regions for Mouse mm10 (0-based)

Usage
blacklist_mm10

Format
A GRanges object

Source
https://github.com/Boyle-Lab/Blacklist
doi:10.1038/s4159801945839z

CallPeaks  Call peaks

Description
Call peaks using MACS. Fragment files linked to the specified assay will be used to call peaks. If multiple fragment files are present, all will be used in a single MACS invocation. Returns the .narrowPeak MACS output as a GRanges object.

Usage
CallPeaks(object, ...)

## S3 method for class 'Seurat'
CallPeaks(
  object,
  assay = NULL,
  group.by = NULL,
  idents = NULL,
  macs2.path = NULL,
  broad = FALSE,
  format = "BED",
  outdir = tempdir(),
  fragment.tempdir = tempdir(),
  combine.peaks = TRUE,
CallPeaks

CallPeaks(
  object,
  macs2.path = NULL,
  outdir = tempdir(),
  broad = FALSE,
  format = "BED",
  effective.genome.size = 2.7e+09,
  extsize = 200,
  shift = -extsize/2,
  additional.args = NULL,
  name = "macs2",
  cleanup = TRUE,
  verbose = TRUE,
  ...
)

## S3 method for class 'ChromatinAssay'
CallPeaks(
  object,
  macs2.path = NULL,
  outdir = tempdir(),
  broad = FALSE,
  format = "BED",
  effective.genome.size = 2.7e+09,
  extsize = 200,
  shift = -extsize/2,
  additional.args = NULL,
  name = "macs2",
  cleanup = TRUE,
  verbose = TRUE,
  ...
)

## S3 method for class 'Fragment'
CallPeaks(
  object,
  macs2.path = NULL,
  outdir = tempdir(),
  broad = FALSE,
  format = "BED",
  effective.genome.size = 2.7e+09,
  extsize = 200,
  shift = -extsize/2,
  additional.args = NULL,
  name = "macs2",
  cleanup = TRUE,
  verbose = TRUE,
  ...
)

## Default S3 method:
CallPeaks(
  object,
  macs2.path = NULL,
```r
outdir = tempdir(),
broad = FALSE,
format = "BED",
effective.genome.size = 2.7e+09,
extsize = 200,
shift = -extsize/2,
additional.args = NULL,
name = "macs2",
cleanup = TRUE,
verbose = TRUE,
...
)
Arguments

object A Seurat object, ChromatinAssay object, Fragment object, or the path to fragment file/s.

... Arguments passed to other methods

assay Name of assay to use

group.by Grouping variable to use. If set, peaks will be called independently on each group of cells and then combined. Note that to call peaks using subsets of cells we first split the fragment file/s used, so using a grouping variable will require extra time to split the files and perform multiple MACS peak calls, and will store additional files on-disk that may be large. Note that we store split fragment files in the temp directory (`tempdir`) by default, and if the program is interrupted before completing these temporary files will not be removed. If NULL, peaks are called using all cells together (pseudobulk).

idents List of identities to include if grouping cells (only valid if also setting the `group.by` parameter). If NULL, peaks will be called for all cell identities.

macs2.path Path to MACS program. If NULL, try to find MACS automatically.

broad Call broad peaks (`--broad` parameter for MACS)

format File format to use. Should be either "BED" or "BEDPE" (see MACS documentation).

outdir Path for output files

fragment.tempdir Path to write temporary fragment files. Only used if `group.by` is not NULL.

combine.peaks Controls whether peak calls from different groups of cells are combined using `GenomicRanges::reduce` when calling peaks for different groups of cells (`group.by` parameter). If FALSE, a list of `GenomicRanges` object will be returned. Note that metadata fields such as the p-value, q-value, and fold-change information for each peak will be lost if combining peaks.

effective.genome.size Effective genome size parameter for MACS (`-g`). Default is the human effective genome size (2.7e9).

extsize extsize parameter for MACS. Only relevant if format="BED"
**Cells.Fragment**

- **shift**: shift parameter for MACS. Only relevant if format="BED"
- **additional.args**: Additional arguments passed to MACS. This should be a single character string
- **name**: Name for output MACS files. This will also be placed in the name field in the GRanges output.
- **cleanup**: Remove MACS output files
- **verbose**: Display messages

**Details**

See [https://macs3-project.github.io/MACS/](https://macs3-project.github.io/MACS/) for MACS documentation.

If you call peaks using MACS2 please cite: doi:10.1186/gb200899r137

**Value**

Returns a GRanges object

---

**Cells.Fragment**

*Set and get cell barcode information for a Fragment object*

**Description**

This returns the names of cells in the object that are contained in the fragment file. These cell barcodes may not match the barcodes present in the fragment file. The Fragment object contains an internal mapping of the cell names in the ChromatinAssay object to the cell names in the fragment file, so that cell names can be changed in the assay without needing to change the cell names on disk.

**Usage**

```r
## S3 method for class 'Fragment'
Cells(x, ...)

## S3 replacement method for class 'Fragment'
Cells(x, ...) <- value
```

**Arguments**

- **x**: A Fragment object
- **...**: Arguments passed to other methods
- **value**: A vector of cell names to store in the Fragment object

**Details**

To access the cell names that are stored in the fragment file itself, use `GetFragmentData(object = x, name = "cells").`
Cells<-  

Set and get cell barcode information for a Fragment object

Description

Set and get cell barcode information for a Fragment object

Usage

Cells(x, ...) <- value

Arguments

x  A Seurat object
...
Arguments passed to other methods
value  A character vector of cell barcodes

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)
The ChromatinAssay object is an extended Assay for the storage and analysis of single-cell chromatin data.

**Slots**

- **ranges**: A GRanges object describing the genomic location of features in the object
- **motifs**: A Motif object
- **fragments**: A list of Fragment objects.
- **seqinfo**: A Seqinfo object containing basic information about the genome sequence used.
- **annotation**: A GRanges object containing genomic annotations. This should be a GRanges object with the following columns:
  - tx_id: Transcript ID
  - gene_name: Gene name
  - gene_id: Gene ID
  - gene_biotype: Gene biotype (e.g. "protein_coding", "lincRNA")
  - type: Annotation type (e.g. "exon", "gap")
- **bias**: A vector containing Tn5 integration bias information (frequency of Tn5 integration at different kmers)
- **positionEnrichment**: A named list of matrices containing positional enrichment scores for Tn5 integration (for example, enrichment at the TSS)
- **links**: A GRanges object describing linked genomic positions, such as co-accessible sites or enhancer-gene regulatory relationships. This should be a GRanges object, where the start and end coordinates are the two linked genomic positions, and must contain a "score" metadata column.

**ClosestFeature**

**Description**

Find the closest feature to a given set of genomic regions

**Usage**

ClosestFeature(object, regions, annotation = NULL, ...)
ClusterClonotypes

Find relationships between clonotypes

Description

Perform hierarchical clustering on clonotype data

Usage

ClusterClonotypes(object, assay = NULL, group.by = NULL)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>A Seurat object</td>
</tr>
<tr>
<td>assay</td>
<td>Name of assay to use</td>
</tr>
<tr>
<td>group.by</td>
<td>Grouping variable for cells</td>
</tr>
</tbody>
</table>

Value

Returns a list containing two objects of class `hclust`, one for the cell clustering and one for the feature (allele) clustering.
CombineTracks

Combine genome region plots

Description

This can be used to combine coverage plots, peak region plots, gene annotation plots, and linked element plots. The different tracks are stacked on top of each other and the x-axis combined.

Usage

CombineTracks(plotlist, expression.plot = NULL, heights = NULL, widths = NULL)

Arguments

- **plotlist**: A list of plots to combine. Must be from the same genomic region.
- **expression.plot**: Plot containing gene expression information. If supplied, this will be placed to the left of the coverage tracks and aligned with each track.
- **heights**: Relative heights for each plot. If NULL, the first plot will be 8x the height of the other tracks.
- **widths**: Relative widths for each plot. Only required if adding a gene expression panel. If NULL, main plots will be 8x the width of the gene expression panel.

Value

Returns a patchworked ggplot2 object

Examples

```
p1 <- PeakPlot(atac_small, region = "chr1-29554-39554")
p2 <- AnnotationPlot(atac_small, region = "chr1-29554-39554")
CombineTracks(plotlist = list(p1, p2), heights = c(1, 1))
```

ConnectionsToLinks

Cicero connections to links

Description

Convert the output of Cicero connections to a set of genomic ranges where the start and end coordinates of the range are the midpoints of the linked elements. Only elements on the same chromosome are included in the output.
Usage

ConnectionsToLinks(conns, ccans = NULL, threshold = 0, sep = c("-", "-"))

Arguments

conns  A dataframe containing co-accessible elements. This would usually be the output of run_cicero or assemble_connections. Specifically, this should be a dataframe where the first column contains the genomic coordinates of the first element in the linked pair of elements, with chromosome, start, end coordinates separated by "." characters. The second column should be the second element in the linked pair, formatted in the same way as the first column. A third column should contain the co-accessibility scores.

ccans  This is optional, but if supplied should be a dataframe containing the cis-co-accessibility network (CCAN) information generated by generate_ccans. Specifically, this should be a dataframe containing the name of the peak in the first column, and the CCAN that it belongs to in the second column.

threshold  Threshold for retaining a coaccessible site. Links with a value less than or equal to this threshold will be discarded.

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.

Details

See the Cicero package for more information: https://bioconductor.org/packages/cicero/

Value

Returns a GRanges object

ConvertMotifID  

Convert between motif name and motif ID

Description

Converts from motif name to motif ID or vice versa. To convert common names to IDs, use the name parameter. To convert IDs to common names, use the id parameter.

Usage

ConvertMotifID(object, ...)  

## Default S3 method:  
ConvertMotifID(object, name, id, ...)

## S3 method for class 'Motif'
ConvertMotifID(object, ...)

## S3 method for class 'ChromatinAssay'
ConvertMotifID(object, ...)

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

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

Arguments

object A Seurat, ChromatinAssay, or Motif object
...
Arguments passed to other methods
name A vector of motif names
id A vector of motif IDs. Only one of name and id should be supplied
assay For Seurat object. Name of assay to use. If NULL, use the default assay

Value

Returns a character vector with the same length and order as the input. Any names or IDs that were not found will be stored as NA.

CountFragments Count fragments

Description

Count total fragments per cell barcode present in a fragment file.

Usage

CountFragments(fragments, cells = NULL, max_lines = NULL, verbose = TRUE)

Arguments

fragments Path to a fragment file. If a list of fragment files is provided, the total fragments for each cell barcode across all files will be returned
cells Cells to include. If NULL, include all cells
max_lines Maximum number of lines to read from the fragment file. If NULL, read all lines in the file.
verbose Display messages
CountsInRegion

Value

Returns a data.frame with the following columns:

- CB: the cell barcode
- frequency_count: total number of fragments sequenced for the cell
- mononucleosome: total number of fragments with length between 147 bp and 294 bp
- nucleosome_free: total number of fragments with length <147 bp
- reads_count: total number of reads sequenced for the cell

Examples

```r
fpath <- system.file("extdata", "fragments.tsv.gz", package="Signac")
counts <- CountFragments(fragments = fpath)
```

CountsInRegion | Counts in region

Description

Count reads per cell overlapping a given set of regions

Usage

```r
CountsInRegion(object, assay, regions, ...)
```

Arguments

- `object`: A Seurat object
- `assay`: Name of a chromatin assay in the object to use
- `regions`: A GRanges object
- `...`: Additional arguments passed to `findOverlaps`

Value

Returns a numeric vector

Examples

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

This is the coverage method for ChromatinAssay objects.

Usage

```r
## S4 method for signature 'ChromatinAssay'
coverage(x, shift = 0L, width = NULL, weight = 1L, method = c("auto", "sort", "hash")
)
## S4 method for signature 'Seurat'
coverage(x, shift = 0L, width = NULL, weight = 1L, method = c("auto", "sort", "hash")
)
```

Arguments

- `x`: A ChromatinAssay object
- `shift`: How much each range should be shifted before coverage is computed. See coverage in the IRanges package.
- `width`: Specifies the length of the returned coverage vectors. See coverage in the IRanges package.
- `weight`: Assigns weight to each range in x. See coverage in the IRanges package.
- `method`: See coverage in the IRanges package

Functions

- `coverage(ChromatinAssay)`: method for ChromatinAssay objects
- `coverage(Seurat)`: method for Seurat objects
CoverageBrowser

Genome browser

Description
Interactive version of the CoveragePlot function. Allows altering the genome position interactively. The current view at any time can be saved to a list of ggplot objects using the "Save plot" button, and this list of plots will be returned after ending the browser by pressing the "Done" button.

Usage
CoverageBrowser(object, region, assay = NULL, sep = c("-", ",-"), ...) 

Arguments
- object: A Seurat object
- region: A set of genomic coordinates
- assay: Name of assay to use
- sep: Separators for genomic coordinates if region supplied as a string rather than GRanges object
- ... Parameters passed to CoveragePlot

Value
Returns a list of ggplot objects

CoveragePlot

Plot Tn5 insertion frequency over a region

Description
Plot frequency of Tn5 insertion events for different groups of cells within given regions of the genome. Tracks are normalized using a per-group scaling factor computed as the number of cells in the group multiplied by the mean sequencing depth for that group of cells. This accounts for differences in number of cells and potential differences in sequencing depth between groups.
Usage

CoveragePlot(
  object,
  region,
  features = NULL,
  assay = NULL,
  split.assays = FALSE,
  assay.scale = "common",
  show.bulk = FALSE,
  expression.assay = "RNA",
  expression.slot = "data",
  annotation = TRUE,
  peaks = TRUE,
  peaks.group.by = NULL,
  ranges = NULL,
  ranges.group.by = NULL,
  ranges.title = "Ranges",
  region.highlight = NULL,
  links = TRUE,
  tile = FALSE,
  tile.size = 100,
  tile.cells = 100,
  bigwig = NULL,
  bigwig.type = "coverage",
  bigwig.scale = "common",
  heights = NULL,
  group.by = NULL,
  split.by = NULL,
  window = 100,
  extend.upstream = 0,
  extend.downstream = 0,
  scale.factor = NULL,
  ymax = NULL,
  cells = NULL,
  idents = NULL,
  sep = c("-", "-"),
  max.downsample = 3000,
  downsample.rate = 0.1,
  ...
)

Arguments

object | A Seurat object
region | A set of genomic coordinates to show. Can be a GRanges object, a string encoding a genomic position, a gene name, or a vector of strings describing the genomic coordinates or gene names to plot. If a gene name is supplied, annotations must be present in the assay.
features A vector of features present in another assay to plot alongside accessibility tracks (for example, gene names).

assay Name of the assay to plot. If a list of assays is provided, data from each assay will be shown overlaid on each track. The first assay in the list will define the assay used for gene annotations, links, and peaks (if shown). The order of assays given defines the plotting order.

split.assays When plotting data from multiple assays, display each assay as a separate track. If FALSE, data from different assays are overlaid on a single track with transparency applied.

assay.scale Scaling to apply to data from different assays. Can be:
- common: plot all assays on a common scale (default)
- separate: plot each assay on a separate scale ranging from zero to the maximum value for that assay within the plotted region

show.bulk Include coverage track for all cells combined (pseudo-bulk). Note that this will plot the combined accessibility for all cells included in the plot (rather than all cells in the object).

expression.assay Name of the assay containing expression data to plot alongside accessibility tracks. Only needed if supplying features argument.

expression.slot Name of slot to pull expression data from. Only needed if supplying the features argument.

annotation Display gene annotations. Set to TRUE or FALSE to control whether genes models are displayed, or choose "transcript" to display all transcript isoforms, or "gene" to display gene models only (same as setting TRUE).

peaks Display peaks

peaks.group.by Grouping variable to color peaks by. Must be a variable present in the feature metadata. If NULL, do not color peaks by any variable.

ranges Additional genomic ranges to plot

ranges.group.by Grouping variable to color ranges by. Must be a variable present in the metadata stored in the ranges genomic ranges. If NULL, do not color by any variable.

ranges.title Y-axis title for ranges track. Only relevant if ranges parameter is set.

region.highlight Region to highlight on the plot. Should be a GRanges object containing the coordinates to highlight. By default, regions will be highlighted in grey. To change the color of the highlighting, include a metadata column in the GRanges object named "color" containing the color to use for each region.

links Display links

tile Display per-cell fragment information in sliding windows. If plotting multi-assay data, only the first assay is shown in the tile plot.

tile.size Size of the sliding window for per-cell fragment tile plot

tile.cells Number of cells to display fragment information for in tile plot.
bigwig  List of bigWig file paths to plot data from. Files can be remotely hosted. The name of each element in the list will determine the y-axis label given to the track.

bigwig.type  Type of track to use for bigWig files ("line", "heatmap", or "coverage"). Should either be a single value, or a list of values giving the type for each individual track in the provided list of bigwig files.

bigwig.scale  Same as assay.scale parameter, except for bigWig files when plotted with bigwig.type="coverage"

heights  Relative heights for each track (accessibility, gene annotations, peaks, links).

group.by  Name of one or more metadata columns to group (color) the cells by. Default is the current cell identities

split.by  A metadata variable to split the tracks by. For example, grouping by "celltype" and splitting by "batch" will create separate tracks for each combination of cell-type and batch.

window  Smoothing window size

extend.upstream  Number of bases to extend the region upstream.

extend.downstream  Number of bases to extend the region downstream.

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.

ymax  Maximum value for Y axis. Can be one of:
- NULL: set to the highest value among all the tracks (default)
- qXX: clip the maximum value to the XX quantile (for example, q95 will set the maximum value to 95% of the maximum value in the data). This can help remove the effect of extreme values that may otherwise distort the scale.
- numeric: manually define a Y-axis limit

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.

max.downsample  Minimum number of positions kept when downsampling. Downsampling rate is adaptive to the window size, but this parameter will set the minimum possible number of positions to include so that plots do not become too sparse when the window size is small.

downsample.rate  Fraction of positions to retain when downsampling. Retaining more positions can give a higher-resolution plot but can make the number of points large, resulting in larger file sizes when saving the plot and a longer period of time needed to draw the plot.

...  Additional arguments passed to wrap_plots
Details

Additional information can be layered on the coverage plot by setting several different options in the CoveragePlot function. This includes showing:

- gene annotations
- peak positions
- additional genomic ranges
- additional data stored in a bigWig file, which may be hosted remotely
- gene or protein expression data alongside coverage tracks
- peak-gene links
- the position of individual sequenced fragments as a heatmap
- data for multiple chromatin assays simultaneously
- a pseudobulk for all cells combined

Value

Returns a patchwork object

Examples

```r
fpath <- system.file("extdata", "fragments.tsv.gz", package="Signac")
fragments <- CreateFragmentObject(
  path = fpath,
  cells = colnames(atac_small),
  validate.fragments = FALSE
)
Fragments(atac_small) <- fragments

# Basic coverage plot
CoveragePlot(object = atac_small, region = c("chr1-713500-714500"))

# Show additional ranges
ranges.show <- StringToGRanges("chr1-713750-714000")
CoveragePlot(object = atac_small, region = c("chr1-713500-714500"), ranges = ranges.show)

# Highlight region
CoveragePlot(object = atac_small, region = c("chr1-713500-714500"), region.highlight = ranges.show)

# Change highlight color
ranges.show$color <- "orange"
CoveragePlot(object = atac_small, region = c("chr1-713500-714500"), region.highlight = ranges.show)

# Show expression data
CoveragePlot(object = atac_small, region = c("chr1-713500-714500"), features = "ELK1")
```
CreateChromatinAssay

Create ChromatinAssay object

Description

Create a ChromatinAssay object from a count matrix or normalized data matrix. The expected format of the input matrix is features x cells. A set of genomic ranges must be supplied along with the matrix, with the length of the ranges equal to the number of rows in the matrix. If a set of genomic ranges are not supplied, they will be extracted from the row names of the matrix.

Usage

CreateChromatinAssay(
  counts,
  data,
  min.cells = 0,
  min.features = 0,
  max.cells = NULL,
  ranges = NULL,
  motifs = NULL,
  fragments = NULL,
  genome = NULL,
  annotation = NULL,
  bias = NULL,
  positionEnrichment = NULL,
  sep = c("-", "-"),
  validate.fragments = TRUE,
  verbose = TRUE,
  ...
)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>counts</td>
<td>Unnormalized data (raw counts)</td>
</tr>
<tr>
<td>data</td>
<td>Normalized data; if provided, do not pass counts</td>
</tr>
<tr>
<td>min.cells</td>
<td>Include features detected in at least this many cells. Will subset the counts matrix as well. To reintroduce excluded features, create a new object with a lower cutoff.</td>
</tr>
<tr>
<td>min.features</td>
<td>Include cells where at least this many features are detected.</td>
</tr>
<tr>
<td>max.cells</td>
<td>Include features detected in less than this many cells. Will subset the counts matrix as well. To reintroduce excluded features, create a new object with a higher cutoff. This can be useful for chromatin assays where certain artefactual loci accumulate reads in all cells. A percentage cutoff can also be set using 'q' followed by the percentage of cells, for example 'q90' will discard features detected in 90 percent of cells. If NULL (default), do not apply any maximum value.</td>
</tr>
</tbody>
</table>
ranges A set of GRanges corresponding to the rows of the input matrix
motifs A Motif object (not required)
fragments Path to a tabix-indexed fragments file for the data contained in the input matrix. If multiple fragment files are required, you can add additional Fragment object to the assay after it is created using the CreateFragmentObject and Fragments functions. Alternatively, a list of Fragment objects can be provided.
genome A Seqinfo object containing basic information about the genome used. Alternatively, the name of a UCSC genome can be provided and the sequence information will be downloaded from UCSC.
annotation A set of GRanges containing annotations for the genome used
bias A Tn5 integration bias matrix
positionEnrichment A named list of matrices containing positional signal enrichment information for each cell. Should be a cell x position matrix, centered on an element of interest (for example, TSS sites).
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. Only used if ranges is NULL.
validate.fragments Check that cells in the assay are present in the fragment file.
verbose Display messages
... Additional arguments passed to CreateFragmentObject

CreateFragmentObject Create a Fragment object

Description

Create a Fragment object to store fragment file information. This object stores a 32-bit MD5 hash of the fragment file and the fragment file index so that any changes to the files on-disk can be detected. A check is also performed to ensure that the expected cells are present in the fragment file.

Usage

CreateFragmentObject(
  path,
  cells = NULL,
  validate.fragments = TRUE,
  verbose = TRUE,
  ...
)
Arguments

path A path to the fragment file. The file should contain a tabix index in the same directory.
cells A named character vector containing cell barcodes contained in the fragment file. This does not need to be all cells in the fragment file, but there should be no cells in the vector that are not present in the fragment file. A search of the file will be performed until at least one fragment from each cell is found. If NULL, don’t check for expected cells.

Each element of the vector should be a cell barcode that appears in the fragment file, and the name of each element should be the corresponding cell name in the object.

validate.fragments Check that expected cells are present in the fragment file.
verbose Display messages

Examples

defpath <- system.file("extdata", "fragments.tsv.gz", package="Signac")
cells <- colnames(x = atac_small)
names(x = cells) <- paste0("test_", cells)
frags <- CreateFragmentObject(path = fpath, cells = cells, verbose = FALSE, tolerance = 0.5)

CreateMotifMatrix

Create motif matrix

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("-", "-")
...
CreateMotifObject

Create motif object

Description

Create a Motif-class object.

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

```r
# 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 = granges(atac_small),
  pwm = pwm,
  genome = BSgenome.Hsapiens.UCSC.hg19
)

# End(Not run)
```
Usage

CreateMotifObject(
  data = NULL,
  pwm = NULL,
  motif.names = NULL,
  positions = 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.
- **positions**: A GRangesList object containing exact positions of each motif.
- **meta.data**: A data.frame containing metadata

Value

Returns a Motif object

Examples

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

Description

Create a scatterplot using variables in the object metadata and color cells by the density of points in the x-y space.

Usage

DensityScatter(object, x, y, log_x = FALSE, log_y = FALSE, quantiles = NULL)
**DepthCor**

**Arguments**

- **object**: A Seurat object
- **x**: Name of metadata variable to plot on x axis
- **y**: Name of metadata variable to plot on y axis
- **log_x**: log10 transform x values
- **log_y**: log10 transform y values
- **quantiles**: Vector of quantiles to display for x and y data distribution. Must be integer values between 0 and 100. TRUE can be passed as a shorthand way to set c(5, 10, 90, 95). If FALSE or NULL, no quantile information is displayed

**Value**

Returns a ggplot object

**Description**

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

**Usage**

```r
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**

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

ExpressionPlot

Plot gene expression

Description

Display gene expression values for different groups of cells and different genes. Genes will be arranged on the x-axis and different groups stacked on the y-axis, with expression value distribution for each group shown as a violin plot. This is designed to work alongside a genomic coverage track, and the plot will be able to be aligned with coverage tracks for the same groups of cells.

Usage

```
ExpressionPlot(
  object,
  features,
  assay = NULL,
  group.by = NULL,
  idents = NULL,
  slot = "data"
)
```
Arguments

object A Seurat object
features A list of features to plot
assay Name of the assay storing expression information
group.by A grouping variable to group cells by. If NULL, use the current cell identities
idents A list of identities to include in the plot. If NULL, include all identities
slot Which slot to pull expression data from

Examples

ExpressionPlot(atac_small, features = "TSPAN6", assay = "RNA")

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)
FeatureMatrix

Description

Construct a feature x cell matrix from a genomic fragments file

Usage

FeatureMatrix(
  fragments,
  features,
  cells = NULL,
  process_n = 2000,
  sep = c("-", "-"),
  verbose = TRUE
)

Arguments

fragments
  A list of Fragment objects. Note that if setting the cells parameter, the requested cells should be present in the supplied Fragment objects. However, if the cells information in the fragment object is not set (Cells(fragments) is NULL), then the fragment object will still be searched.

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

process_n
  Number of regions to load into memory at a time, per thread. Processing more regions at once can be faster but uses 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")
fragments <- CreateFragmentObject(fpath)
FeatureMatrix(
  fragments = fragments,
  features = granges(atac_small)
)
FilterCells

Filter cells from fragment file

Description

Remove all fragments that are not from an allowed set of cell barcodes from the fragment file. This will create a new file on disk that only contains fragments from cells specified in the cells argument. The output file is block gzip-compressed and indexed, ready for use with Signac functions.

Usage

FilterCells(
    fragments,
    cells,
    outfile = NULL,
    buffer_length = 256L,
    verbose = TRUE
)

Arguments

fragments  Path to a fragment file
cells      A vector of cells to keep
outfile    Name for output file
buffer_length Size of buffer to be read from the fragment file. This must be longer than the longest line in the file.
verbose    Display messages

Examples

fpath <- system.file("extdata", "fragments.tsv.gz", package="Signac")
tmpf <- tempfile(fileext = ".gz")
FilterCells(
    fragments = fpath,
    cells = head(colnames(atac_small)),
    outfile = tmpf
)
file.remove(tmpf)
FindClonotypes

Description

Identify groups of related cells from allele frequency data. This will cluster the cells based on their allele frequencies, reorder the factor levels for the cluster identities by hierarchical clustering the collapsed (pseudobulk) cluster allele frequencies, and set the variable features for the allele frequency assay to the order of features defined by hierarchical clustering.

Usage

FindClonotypes(
  object,
  assay = NULL,
  features = NULL,
  metric = "cosine",
  resolution = 1,
  k = 10,
  algorithm = 3
)

Arguments

- **object**: A Seurat object
- **assay**: Name of assay to use
- **features**: Features to include when constructing neighbor graph
- **metric**: Distance metric to use
- **resolution**: Clustering resolution to use. See FindClusters
- **k**: Passed to k.param argument in FindNeighbors
- **algorithm**: Community detection algorithm to use. See FindClusters

Value

Returns a Seurat object
Find motifs over-represented 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

```r
FindMotifs(
  object,
  features,
  background = 40000,
  assay = NULL,
  verbose = TRUE,
  p.adjust.method = "BH",
  ...
)
```

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 specifying 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
- `p.adjust.method`: Multiple testing correction method to be applied. Passed to `p.adjust`
- `...`: Arguments passed to `MatchRegionStats`.

Value

Returns a data frame
**Examples**

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

**Description**

The `findOverlaps`, `countOverlaps` methods are available for `ChromatinAssay` objects. This allows finding overlaps between genomic ranges and the ranges stored in the ChromatinAssay.

**Usage**

```r
## S4 method for signature 'Vector,ChromatinAssay'
findOverlaps(
  query,
  subject,
  maxgap = -1L,
  minoverlap = 0L,
  type = c("any", "start", "end", "within", "equal"),
  select = c("all", "first", "last", "arbitrary"),
  ignore.strand = FALSE
)

## S4 method for signature 'ChromatinAssay,Vector'
findOverlaps(
  query,
  subject,
  maxgap = -1L,
  minoverlap = 0L,
  type = c("any", "start", "end", "within", "equal"),
  select = c("all", "first", "last", "arbitrary"),
  ignore.strand = FALSE
)

## S4 method for signature 'ChromatinAssay,ChromatinAssay'
findOverlaps(
  query,
  subject,
  maxgap = -1L,
  minoverlap = 0L,
```
findOverlaps-methods

## S4 method for signature 'Vector,Seurat'
findOverlaps(
  query,
  subject,
  maxgap = -1L,
  minoverlap = 0L,
  type = c("any", "start", "end", "within", "equal"),
  select = c("all", "first", "last", "arbitrary"),
  ignore.strand = FALSE
)

## S4 method for signature 'Seurat,Vector'
findOverlaps(
  query,
  subject,
  maxgap = -1L,
  minoverlap = 0L,
  type = c("any", "start", "end", "within", "equal"),
  select = c("all", "first", "last", "arbitrary"),
  ignore.strand = FALSE
)

## S4 method for signature 'Seurat,Seurat'
findOverlaps(
  query,
  subject,
  maxgap = -1L,
  minoverlap = 0L,
  type = c("any", "start", "end", "within", "equal"),
  select = c("all", "first", "last", "arbitrary"),
  ignore.strand = FALSE
)

## S4 method for signature 'Vector,ChromatinAssay'
countOverlaps(
  query,
  subject,
  maxgap = -1L,
  minoverlap = 0L,
  type = c("any", "start", "end", "within", "equal"),
  ignore.strand = FALSE
)
## S4 method for signature 'ChromatinAssay,Vector'
countOverlaps(
  query,
  subject,
  maxgap = -1L,
  minoverlap = 0L,
  type = c("any", "start", "end", "within", "equal"),
  ignore.strand = FALSE
)

## S4 method for signature 'ChromatinAssay,ChromatinAssay'
countOverlaps(
  query,
  subject,
  maxgap = -1L,
  minoverlap = 0L,
  type = c("any", "start", "end", "within", "equal"),
  ignore.strand = FALSE
)

## S4 method for signature 'Seurat,Vector'
countOverlaps(
  query,
  subject,
  maxgap = -1L,
  minoverlap = 0L,
  type = c("any", "start", "end", "within", "equal"),
  ignore.strand = FALSE
)

## S4 method for signature 'Vector,Seurat'
countOverlaps(
  query,
  subject,
  maxgap = -1L,
  minoverlap = 0L,
  type = c("any", "start", "end", "within", "equal"),
  ignore.strand = FALSE
)

## S4 method for signature 'Seurat,Seurat'
countOverlaps(
  query,
  subject,
  maxgap = -1L,
  minoverlap = 0L,
  type = c("any", "start", "end", "within", "equal"),
  ignore.strand = FALSE
)
Arguments

query, subject  A ChromatinAssay object
maxgap, minoverlap, type, select, ignore.strand

See ?findOverlaps in the GenomicRanges and IRanges packages.

Details

If a ChromatinAssay is set as the default assay in a Seurat object, you can also call findOverlaps directly on the Seurat object.

Value

See findOverlaps

Functions

• findOverlaps(query = ChromatinAssay, subject = Vector): method for ChromatinAssay, Vector
• findOverlaps(query = ChromatinAssay, subject = ChromatinAssay): method for ChromatinAssay, ChromatinAssay
• findOverlaps(query = Vector, subject = Seurat): method for Vector, Seurat
• findOverlaps(query = Seurat, subject = Vector): method for Seurat, Vector
• findOverlaps(query = Seurat, subject = Seurat): method for Seurat, Seurat
• countOverlaps(query = Vector, subject = ChromatinAssay): method for Vector, ChromatinAssay
• countOverlaps(query = ChromatinAssay, subject = Vector): method for ChromatinAssay, Vector
• countOverlaps(query = ChromatinAssay, subject = ChromatinAssay): method for ChromatinAssay, ChromatinAssay
• countOverlaps(query = Seurat, subject = Vector): method for Seurat, Vector
• countOverlaps(query = Vector, subject = Seurat): method for Vector, Seurat
• countOverlaps(query = Seurat, subject = Seurat): method for Seurat, Seurat

See Also

• findOverlaps-methods in the IRanges package.
• findOverlaps-methods in the GenomicRanges package
• ChromatinAssay-class
Description

Find top features for a given assay based on total number of counts for the feature. Can specify a minimum cell count, or a lower percentile bound to determine the set of variable features. Running this function will store the total counts and percentile rank for each feature in the feature metadata for the assay. To only compute the feature metadata, without changing the variable features for the assay, set min.cutoff=NA.

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 minimum number of counts for the feature to be included in the set of VariableFeatures. For example, setting to 10 will include features with >10 total counts in the set of VariableFeatures. If NULL, include all features in VariableFeatures. If NA, VariableFeatures will not be altered, and only the feature metadata will be updated with the total counts and percentile rank for each feature.  
- **verbose** Display messages

Value

Returns a Seurat object
Examples

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

Description

Compute the normalized observed/expected Tn5 insertion frequency for each position surrounding a set of motif instances.

Usage

Footprint(object, ...)

## S3 method for class 'ChromatinAssay'
Footprint(
  object,
  genome,
  motif.name = NULL,
  key = motif.name,
  regions = NULL,
  assay = NULL,
  upstream = 250,
  downstream = 250,
  compute.expected = TRUE,
  in.peaks = FALSE,
  verbose = TRUE,
  ...
)

## S3 method for class 'Seurat'
Footprint(
  object,
  genome,
  regions = NULL,
  motif.name = NULL,
  assay = NULL,
  upstream = 250,
  downstream = 250,
  in.peaks = FALSE,
  verbose = TRUE,
  ...
)
**FractionCountsInRegion**

**Arguments**

- `object` A Seurat or ChromatinAssay object
- `...` Arguments passed to other methods
- `genome` A BSgenome object or any other object supported by `getSeq`. Do `showMethods("getSeq")` to get the list of all supported object types.
- `motif.name` Name of a motif stored in the assay to footprint. If not supplied, must supply a set of regions.
- `key` Key to store positional enrichment information under.
- `regions` A set of genomic ranges containing the motif instances. These should all be the same width.
- `assay` Name of assay to use
- `upstream` Number of bases to extend upstream
- `downstream` Number of bases to extend downstream
- `compute.expected` Find the expected number of insertions at each position given the local DNA sequence context and the insertion bias of Tn5
- `in.peaks` Restrict motifs to those that fall in peaks
- `verbose` Display messages

**Value**

Returns a Seurat object

---

**FractionCountsInRegion**

*Fraction of counts in a genomic region*

**Description**

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

**Usage**

FractionCountsInRegion(object, regions, assay = NULL, ...)

**Arguments**

- `object` A Seurat object
- `regions` A GRanges object containing a set of genomic regions
- `assay` Name of assay to use
- `...` Additional arguments passed to CountsInRegion
Value

Returns a numeric vector

Examples

```r
## Not run:
FractionCountsInRegion(
  object = atac_small,
  assay = 'bins',
  regions = blacklist_hg19
)
## End(Not run)
```

---

### Fragment-class

**The Fragment class**

**Description**

The Fragment class is designed to hold information needed for working with fragment files.

**Slots**

- path: Path to the fragment file on disk. See [https://support.10xgenomics.com/single-cell-atac/software/pipelines/latest/output/fragments](https://support.10xgenomics.com/single-cell-atac/software/pipelines/latest/output/fragments)
- hash: A vector of two md5sums: first element is the md5sum of the fragment file, the second element is the md5sum of the index.
- cells: A named vector of cells where each element is the cell barcode as it appears in the fragment file, and the name of each element is the corresponding cell barcode as stored in the ChromatinAssay object.

---

### FragmentHistogram

**Plot fragment length histogram**

**Description**

Plot the frequency that fragments of different lengths are present for different groups of cells.
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.
... Arguments passed to other functions

Value

Returns a ggplot object

Examples

fpath <- system.file("extdata", "fragments.tsv.gz", package="Signac")
Fragments(atac_small) <- CreateFragmentObject(
    path = fpath,
    cells = colnames(atac_small),
    validate.fragments = FALSE
)
FragmentHistogram(object = atac_small, region = "chr1-10245-780007")

Fragments

Get the Fragment objects

Description

Get the Fragment objects
Fragments

Usage

Fragments(object, ...)

Fragments(object, ...) <- value

## S3 method for class 'ChromatinAssay'
Fragments(object, ...)

## S3 method for class 'Seurat'
Fragments(object, ...)

## S3 replacement method for class 'ChromatinAssay'
Fragments(object, ...) <- value

## S3 replacement method for class 'Seurat'
Fragments(object, ...) <- value

Arguments

object A Seurat object or ChromatinAssay object

... Arguments passed to other methods

value A Fragment object or list of Fragment objects

Value

Returns a list of Fragment objects. If there are no Fragment objects present, returns an empty list.

Examples

Fragments(atac_small["peaks"])
Fragments(atac_small)

fpath <- system.file("extdata", "fragments.tsv.gz", package="Signac")
fragments <- CreateFragmentObject(
  path = fpath,
  cells = colnames(atac_small),
  validate.fragments = FALSE
)
Fragments(atac_small["bins"]) <- fragments

fpath <- system.file("extdata", "fragments.tsv.gz", package="Signac")
fragments <- CreateFragmentObject(
  path = fpath,
  cells = colnames(atac_small),
  validate.fragments = FALSE
)
Fragments(atac_small) <- fragments
FRiP

Calculate fraction of reads in peaks per cell

Description

Calculate fraction of reads in peaks per cell

Usage

FRiP(object, assay, total.fragments, col.name = "FRiP", verbose = TRUE)

Arguments

object
A Seurat object

assay
Name of the assay containing a peak x cell matrix

total.fragments
Name of a metadata column containing the total number of sequenced fragments for each cell. This can be computed using the CountFragments function.

col.name
Name of column in metadata to store the FRiP information.

verbose
Display messages

Value

Returns a Seurat object

Examples

FRiP(object = atac_small, assay = 'peaks', total.fragments = "fragments")

GeneActivity

Create gene activity matrix

Description

Compute counts per cell in gene body and promoter region.
Usage

GeneActivity(
  object,
  assay = NULL,
  features = NULL,
  extend.upstream = 2000,
  extend.downstream = 0,
  biotypes = "protein_coding",
  max.width = 5e+05,
  process_n = 2000,
  gene.id = FALSE,
  verbose = TRUE
)

Arguments

object A Seurat object
assay Name of assay to use. If NULL, use the default assay
features Genes to include. If NULL, use all protein-coding genes in the annotations stored in the object
extend.upstream Number of bases to extend upstream of the TSS
extend.downstream Number of bases to extend downstream of the TTS
biotypes Gene biotypes to include. If NULL, use all biotypes in the gene annotation.
max.width Maximum allowed gene width for a gene to be quantified. Setting this parameter can avoid quantifying extremely long transcripts that can add a relatively long amount of time. If NULL, do not filter genes based on width.
process_n Number of regions to load into memory at a time, per thread. Processing more regions at once can be faster but uses more memory.
gene.id Record gene IDs in output matrix rather than gene name.
verbose Display messages

Value

Returns a sparse matrix

Examples

fpath <- system.file("extdata", "fragments.tsv.gz", package="Signac")
fragments <- CreateFragmentObject(
  path = fpath,
  cells = colnames(atac_small),
  validate.fragments = FALSE
)
Fragments(atac_small) <- fragments
GeneActivity(atac_small)
GenomeBinMatrix

Genome bin matrix

Description

Construct a bin x cell matrix from a fragments file.

Usage

GenomeBinMatrix(
  fragments,
  genome,
  cells = NULL,
  binsize = 5000,
  process_n = 2000,
  sep = c("-", "-"),
  verbose = TRUE
)

Arguments

fragments Path to tabix-indexed fragments file or a list of Fragment objects
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
process_n Number of regions to load into memory at a time, per thread. Processing more regions at once can be faster but uses 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
**GetCellsInRegion**

Get cells in a region

**Description**

Extract cell names containing reads mapped within a given genomic region

**Usage**

```r
GetCellsInRegion(tabix, region, cells = NULL)
```

**Arguments**

- `tabix`: Tabix object
- `region`: A string giving the region to extract from the fragments file
- `cells`: Vector of cells to include in output. If NULL, include all cells

**Value**

Returns a list

**Examples**

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

Get footprinting data

Description

Extract footprint data for a set of transcription factors or metafeatures. This function will pull accessibility data for a given feature (e.g., a TF), and perform background normalization for each identity class. This is the data that's used to create TF footprinting plots with the PlotFootprint function.

Usage

GetFootprintData(
  object, 
  features, 
  assay = NULL, 
  group.by = NULL, 
  idents = NULL 
)

Arguments

object A Seurat object
features A vector of features to extract data for
assay Name of assay to use
group.by A grouping variable
idents Set of identities to group cells by

Value

Returns a matrix

GetFragmentData

Get Fragment object data

Description

Extract data from a Fragment-class object

Usage

GetFragmentData(object, slot = "path")

Arguments

object A Fragment object
slot Information to pull from object (path, hash, cells, prefix, suffix)
GetGRangesFromEnsDb

Extract genomic ranges from EnsDb object

Description
Pulls the transcript information for all chromosomes from an EnsDb object. This wraps crunch and applies the extractor function to all chromosomes present in the EnsDb object.

Usage
GetGRangesFromEnsDb(
  ensdb,
  standard.chromosomes = TRUE,
  biotypes = c("protein_coding", "lincRNA", "rRNA", "processed_transcript"),
  verbose = TRUE
)

Arguments
ensdb An EnsDb object
standard.chromosomes Keep only standard chromosomes
biotypes Biotypes to keep
verbose Display messages

GetIntersectingFeatures

Find intersecting 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 intersect for each object. The order of the row names return corresponds to the intersecting regions, i.e. 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,
  verbose = TRUE
)
GetLinkedGenes

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. Default is 0.

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'
)

GetLinkedGenes

Get genes linked to peaks

Description

Find genes linked to a given set of peaks

Usage

GetLinkedGenes(object, features, assay = NULL, min.abs.score = 0)

Arguments

object
A Seurat object

features
A list of peaks to find linked genes for

assay
Name of assay to use. If NULL, use the default assay

min.abs.score
Minimum absolute value of the link score for a link to be returned

See Also

GetLinkedPeaks
GetLinkedPeaks

Get peaks linked to genes

Description

Find peaks linked to a given set of genes

Usage

GetLinkedPeaks(object, features, assay = NULL, min.abs.score = 0)

Arguments

object A Seurat object
features A list of genes to find linked peaks for
assay Name of assay to use. If NULL, use the default assay
min.abs.score Minimum absolute value of the link score for a link to be returned

See Also

GetLinkedGenes

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 'ChromatinAssay'
GetMotifData(object, slot = "data", ...)

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

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

```r
motif.obj <- SeuratObject::GetAssayData(
  object = atac_small[['Varpeaks']], slot = "motifs"
)
GetMotifData(object = motif.obj)
GetMotifData(object = atac_small)
```

---

GetTSSPositions  
*Find transcriptional start sites*

Description

Get the TSS positions from a set of genomic ranges containing gene positions. Ranges can contain exons, introns, UTRs, etc, rather than the whole transcript. Only protein coding gene biotypes are included in output.

Usage

GetTSSPositions(ranges, biotypes = "protein_coding")

Arguments

- **ranges**: A GRanges object containing gene annotations.
- **biotypes**: Gene biotypes to include. If NULL, use all biotypes in the supplied gene annotation.
granges-methods  

Access genomic ranges for ChromatinAssay objects

Description

Methods for accessing GRanges object information stored in a ChromatinAssay object.

Usage

```r
## S4 method for signature 'ChromatinAssay'
granges(x, use.names = TRUE, use.mcols = FALSE, ...)

## S4 method for signature 'Seurat'
granges(x, use.names = TRUE, use.mcols = FALSE, ...)
```

Arguments

- `x`: A ChromatinAssay object
- `use.names`: Whether the names on the genomic ranges should be propagated to the returned object.
- `use.mcols`: Not supported for ChromatinAssay objects
- `...`: Additional arguments

Value

Returns a GRanges object

Functions

- `granges(Seurat)`: method for Seurat objects

See Also

- `granges` in the GenomicRanges package.
- ChromatinAssay-class

Examples

```
granges(atac_small)
```
**GRangesToString**

**GRanges to String**

**Description**

Convert GRanges object to a vector of strings

**Usage**

```r
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**

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

---

**head.Fragment**

**Return the first rows of a fragment file**

**Description**

Returns the first n rows of a fragment file. This allows the content of a fragment file to be inspected.

**Usage**

```r
## S3 method for class 'Fragment'
head(x, n = 6L, ...)
```

**Arguments**

- `x`: a Fragment object
- `n`: an integer specifying the number of rows to return from the fragment file
- `...`: additional arguments passed to `read.table`

**Value**

The first n rows of a fragment file as a data.frame with the following columns: chrom, start, end, barcode, readCount.
IdentifyVariants

**Description**

Identify mitochondrial variants present in single cells.

**Usage**

IdentifyVariants(object, ...)

```r
## Default S3 method:
IdentifyVariants(
  object,
  refallele,
  stabilize_variance = TRUE,
  low_coverage_threshold = 10,
  verbose = TRUE,
  ...
)
```

```r
## S3 method for class 'Assay'
IdentifyVariants(object, refallele, ...)
```

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

**Arguments**

- **object**
  A Seurat object

- **...**
  Arguments passed to other methods

- **refallele**
  A dataframe containing reference alleles for the mitochondrial genome.

- **stabilize_variance**
  Stabilize variance

- **low_coverage_threshold**
  Low coverage threshold

- **verbose**
  Display messages

- **assay**
  Name of assay to use. If NULL, use the default assay.

**Value**

Returns a dataframe
InsertionBias

Examples

```r
## Not run:
data.dir <- "path/to/data/directory"
mgatk <- ReadMGATK(dir = data.dir)
variant.df <- IdentifyVariants(
  object = mgatk$counts,
  refallele = mgatk$refallele
)

## End(Not run)
```

---

InsertionBias  

*Compute Tn5 insertion bias*

Description

Counts the Tn5 insertion frequency for each DNA hexamer.

Usage

```r
InsertionBias(object, ...)  

## S3 method for class 'ChromatinAssay'
InsertionBias(object, genome, region = "chr1-1-249250621", verbose = TRUE, ...)

## S3 method for class 'Seurat'
InsertionBias(
  object,
  genome,
  assay = NULL,
  region = "chr1-1-249250621",
  verbose = TRUE,
  ...
)
```

Arguments

- `object`  
  A Seurat or ChromatinAssay object

- `...`  
  Additional arguments passed to `StringToGRanges`

- `genome`  
  A BSgenome object or any other object supported by `getSeq`. Do `showMethods("getSeq")` to get the list of all supported object types.

- `region`  
  Genomic region to use when assessing bias.

- `verbose`  
  Display messages

- `assay`  
  Name of assay to use
Value

Returns a Seurat object

Examples

## Not run:
library(BSgenome.Mmusculus.UCSC.mm10)

region.use <- GRanges(
  seqnames = c('chr1', 'chr2'),
  IRanges(start = c(1, 1), end = c(195471971, 182113224))
)

InsertionBias(
  object = atac_small,
  genome = BSgenome.Mmusculus.UCSC.mm10,
  region = region.use
)

## End(Not run)

inter-range-methods  
Inter-range transformations for ChromatinAssay objects

Description

The range, reduce, gaps, disjoin, isDisjoint, disjointBins methods are available for ChromatinAssay objects.

Usage

## S4 method for signature 'ChromatinAssay'
range(x, ..., with.revmap = FALSE, na.rm = FALSE)

## S4 method for signature 'Seurat'
range(x, ..., with.revmap = FALSE, na.rm = FALSE)

## S4 method for signature 'ChromatinAssay'
reduce(x, drop.empty.ranges = FALSE, ...)

## S4 method for signature 'Seurat'
reduce(x, drop.empty.ranges = FALSE, ...)

## S4 method for signature 'ChromatinAssay'
gaps(x, start = NA, end = NA)

## S4 method for signature 'Seurat'
gaps(x, start = NA, end = NA)
## S4 method for signature 'ChromatinAssay'
disjoin(x, ...)

## S4 method for signature 'Seurat'
disjoin(x, ...)

## S4 method for signature 'ChromatinAssay'
isDisjoint(x, ...)

## S4 method for signature 'Seurat'
isDisjoint(x, ...)

## S4 method for signature 'ChromatinAssay'
disjointBins(x, ...)

## S4 method for signature 'Seurat'
disjointBins(x, ...)

### Arguments

- `x`: A `ChromatinAssay` object
- `...`: Additional arguments
- `with.revmap`: See *inter-range-methods* in the *IRanges* packages
- `na.rm`: Ignored
- `drop.empty.ranges`: See ?IRanges{inter-range-methods}
- `start, end`: See ?IRanges{inter-range-methods}

### Functions

- `range(Seurat)`: method for Seurat objects
- `reduce(ChromatinAssay)`: method for ChromatinAssay objects
- `reduce(Seurat)`: method for Seurat objects
- `gaps(ChromatinAssay)`: method for ChromatinAssay objects
- `gaps(Seurat)`: method for Seurat objects
- `disjoin(ChromatinAssay)`: method for ChromatinAssay objects
- `disjoin(Seurat)`: method for Seurat objects
- `isDisjoint(ChromatinAssay)`: method for ChromatinAssay objects
- `isDisjoint(Seurat)`: method for Seurat objects
- `disjointBins(ChromatinAssay)`: method for ChromatinAssay objects
- `disjointBins(Seurat)`: method for Seurat objects
See Also

- inter-range-methods in the IRanges package.
- inter-range-methods in the GenomicRanges package
- ChromatinAssay-class

---

**IntersectMatrix**

**Intersect genomic coordinates with matrix rows**

**Description**

Remove or retain matrix rows that intersect given genomic regions

**Usage**

```r
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**

```r
counts <- matrix(data = rep(0, 12), ncol = 2)
rownames(counts) <- c("chr1-565107-565550","chr1-569174-569639",
  "chr1-713460-714823","chr1-752422-753838",
  "chr1-762186-763359","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 matrices. 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**

\[ \text{Jaccard}(x, y) \]

**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**

\[
\begin{align*}
x & \leftarrow \text{matrix(data = sample(c(0, 1), size = 25, replace = TRUE), ncol = 5)} \\
\text{Jaccard}(x = x, y = x)
\end{align*}
\]

**LinkPeaks**

*Link peaks to genes*

**Description**
Find peaks that are correlated with the expression of nearby genes. For each gene, this function computes the correlation coefficient between the gene expression and accessibility of each peak within a given distance from the gene TSS, and computes an expected correlation coefficient for each peak given the GC content, accessibility, and length of the peak. The expected coefficient values for the peak are then used to compute a z-score and p-value.
Usage

LinkPeaks(
  object,
  peak.assay,  # Name of assay containing peak information
  expression.assay,
  peak.slot = "counts",  # Name of slot to pull chromatin data from
  expression.slot = "data",  # Name of slot to pull expression data from
  method = "pearson",  # Correlation method to use. One of "pearson" or "spearman"
  gene.coords = NULL,  # GRanges object containing coordinates of genes in the expression assay. If
  distance = 5e+05,  # NULL, extract from gene annotations stored in the assay.
  min.distance = NULL,  # Distance threshold for peaks to include in regression model
  min.cells = 10,  # Minimum distance between peak and TSS to include in regression model. If
  genes.use = NULL,  # NULL (default), no minimum distance is used.
  n.sample = 200,  # Minimum number of cells positive for the peak and gene needed to include in
  pvalue_cutoff = 0.05,  # the results.
  score_cutoff = 0.05,  # Genes to test. If NULL, determine from expression assay.
  gene.id = FALSE,  # Number of peaks to sample at random when computing the null distribution.
  verbose = TRUE
)  # Minimum absolute value correlation coefficient for a link to be retained

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>A Seurat object</td>
</tr>
<tr>
<td>peak.assay</td>
<td>Name of assay containing peak information</td>
</tr>
<tr>
<td>expression.assay</td>
<td>Name of assay containing gene expression information</td>
</tr>
<tr>
<td>peak.slot</td>
<td>Name of slot to pull chromatin data from</td>
</tr>
<tr>
<td>expression.slot</td>
<td>Name of slot to pull expression data from</td>
</tr>
<tr>
<td>method</td>
<td>Correlation method to use. One of &quot;pearson&quot; or &quot;spearman&quot;</td>
</tr>
</tbody>
</table>
| gene.coords  | GRanges object containing coordinates of genes in the expression assay. If
               | NULL, extract from gene annotations stored in the assay.                   |
| distance     | Distance threshold for peaks to include in regression model                 |
| min.distance | Minimum distance between peak and TSS to include in regression model. If
               | NULL (default), no minimum distance is used.                                |
| min.cells    | Minimum number of cells positive for the peak and gene needed to include in |
               | the results.                                                                |
| genes.use    | Genes to test. If NULL, determine from expression assay.                    |
| n.sample     | Number of peaks to sample at random when computing the null distribution.   |
| pvalue_cutoff| Minimum p-value required to retain a link. Links with a p-value equal or greater than this value will be removed from the output. |
| score_cutoff | Minimum absolute value correlation coefficient for a link to be retained    |
| gene.id      | Set to TRUE if genes in the expression assay are named using gene IDs rather than gene names. |
| verbose      | Display messages                                                            |
Details

This function was inspired by the method originally described by SHARE-seq (Sai Ma et al. 2020, Cell). Please consider citing the original SHARE-seq work if using this function: doi:10.1016/j.cell.2020.09.056

Value

Returns a Seurat object with the Links information set. This is a granges object accessible via the Links function, with the following information:

- score: the correlation coefficient between the accessibility of the peak and expression of the gene
- zscore: the z-score of the correlation coefficient, computed based on the distribution of correlation coefficients from a set of background peaks
- pvalue: the p-value associated with the z-score for the link
- gene: name of the linked gene
- peak: name of the linked peak

Usage

LinkPlot(
  object,  # A Seurat object
  region,  # A genomic region to plot
  assay = NULL,  # Name of assay to use. If NULL, use the default assay.
  min.cutoff = 0,
  sep = c("-", "-"),
  extend.upstream = 0,
  extend.downstream = 0,
  scale.linewidth = FALSE
)

Arguments

- object  # A Seurat object
- region  # A genomic region to plot
- assay  # Name of assay to use. If NULL, use the default assay.
- min.cutoff  # Minimum absolute score for link to be plotted.

Description

Display links between pairs of genomic elements within a given region of the genome.
Links

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.

extend.upstream
Number of bases to extend the region upstream.

extend.downstream
Number of bases to extend the region downstream.

scale.linewidth
Scale thickness of the line according to link score.

Value
Returns a ggplot object

Description
Get or set the genomic link information for a Seurat object or ChromatinAssay

Usage
Links(object, ...)

Links(object, ...) <- value

## S3 method for class 'ChromatinAssay'
Links(object, ...)

## S3 method for class 'Seurat'
Links(object, ...)

## S3 replacement method for class 'ChromatinAssay'
Links(object, ...) <- value

## S3 replacement method for class 'Seurat'
Links(object, ...) <- value

Arguments

object A Seurat object

... Arguments passed to other methods

value A GRanges object
**LookupGeneCoords**

*Get gene coordinates*

**Description**

Extract the coordinates of the longest transcript for a gene stored in the annotations within an object.

**Usage**

```r
LookupGeneCoords(object, gene, assay = NULL)
```

**Arguments**

- `object` A Seurat object
- `gene` Name of a gene to extract
- `assay` Name of assay to use

**Examples**

```r
LookupGeneCoords(atac_small, gene = "MIR1302-10")
```

**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**

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

- `meta.feature` A dataframe containing DNA sequence information for features to choose from.
- `query.feature` A dataframe containing DNA sequence information for features to match.
- `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.

Details

For each requested feature to match, a density distribution is estimated using the `density` function, and a set of weights for each feature in the dataset estimated based on the density distribution. If multiple features are to be matched (for example, GC content and overall accessibility), a joint density distribution is then computed by multiplying the individual feature weights. A set of features with characteristics matching the query regions is then selected using the `sample` function, with the probability of randomly selecting each feature equal to the joint density distribution weight.

Value

Returns a character vector.

Examples

```r
metafeatures <- SeuratObject::GetAssayData(
  object = atac_small[['peaks']], slot = 'meta.features'
)
query.feature <- metafeatures[1:10, ]
features.choose <- metafeatures[11:nrow(metafeatures), ]
MatchRegionStats(
  meta.feature = features.choose,
  query.feature = query.feature,
  features.match = "percentile",
  n = 10
)
```

Motif-class

The Motif class

Description

The Motif class is designed to store DNA sequence motif information, including motif PWMs or PFM, motif positions, and metadata.
MotifCounts

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
positions  A GRangesList object containing exact positions 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.

MotifCounts  Count fragments surrounding motif sites

Description

Count the number of sequenced DNA fragments in a region surrounding each instance of a given DNA sequence motif.

Usage

MotifCounts(
  object,
  motifs,
  flanking.region = 1000,
  assay = NULL,
  verbose = TRUE,
  ...
)

Arguments

object  A Seurat object
motifs  A list of DNA sequence motif names. One matrix will be generated for each motif
flanking.region  Amount of sequence to include surrounding the motif itself
assay  Name of assay to use. Must be a ChromatinAssay
verbose  Display messages
...

Value

Returns a list of sparse matrices
### MotifPlot

**Plot DNA sequence motif**

#### Description

Plot position weight matrix or position frequency matrix for different DNA sequence motifs.

#### Usage

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

#### Arguments

- `object`: A Seurat object
- `motifs`: A list of motif IDs or motif names 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

```r
motif.obj <- SeuratObject::GetAssayData(atac_small, slot = "motifs")
MotifPlot(atac_small, motifs = head(colnames(motif.obj)))
```

### Motifs

**Get or set a motif information**

#### Description

Get or set the Motif object for a Seurat object or ChromatinAssay.
Usage

Motifs(object, ...)

Motifs(object, ...) <- value

## S3 method for class 'ChromatinAssay'
Motifs(object, ...)

## S3 method for class 'Seurat'
Motifs(object, ...)

## S3 replacement method for class 'ChromatinAssay'
Motifs(object, ...) <- value

## S3 replacement method for class 'Seurat'
Motifs(object, ...) <- value

Arguments

object      A Seurat object
...
Arguments passed to other methods
value       A Motif object

Examples

Motifs(atac_small[["peaks"]])
Motifs(atac_small)
motifs <- Motifs(atac_small)
Motifs(atac_small[["peaks"]]) <- motifs
motifs <- Motifs(atac_small)
Motifs(atac_small) <- motifs

nearest-methods

Find the nearest range neighbors for ChromatinAssay objects

Description

The precede, follow, nearest, distance, distanceToNearest methods are available for ChromatinAssay objects.

Usage

## S4 method for signature 'ANY,ChromatinAssay'
precede(x, subject, select = c("arbitrary", "all"), ignore.strand = FALSE)

## S4 method for signature 'ChromatinAssay,ANY'
precede(x, subject, select = c("arbitrary", "all"), ignore.strand = FALSE)
## S4 method for signature 'ChromatinAssay,ChromatinAssay'
precede(x, subject, select = c("arbitrary", "all"), ignore.strand = FALSE)

## S4 method for signature 'ANY,Seurat'
precede(x, subject, select = c("arbitrary", "all"), ignore.strand = FALSE)

## S4 method for signature 'Seurat,ANY'
precede(x, subject, select = c("arbitrary", "all"), ignore.strand = FALSE)

## S4 method for signature 'Seurat,Seurat'
precede(x, subject, select = c("arbitrary", "all"), ignore.strand = FALSE)

## S4 method for signature 'ANY,ChromatinAssay'
follow(x, subject, select = c("arbitrary", "all"), ignore.strand = FALSE)

## S4 method for signature 'ChromatinAssay,ANY'
follow(x, subject, select = c("arbitrary", "all"), ignore.strand = FALSE)

## S4 method for signature 'ChromatinAssay,ChromatinAssay'
follow(x, subject, select = c("arbitrary", "all"), ignore.strand = FALSE)

## S4 method for signature 'ANY,Seurat'
follow(x, subject, select = c("arbitrary", "all"), ignore.strand = FALSE)

## S4 method for signature 'Seurat,ANY'
follow(x, subject, select = c("arbitrary", "all"), ignore.strand = FALSE)

## S4 method for signature 'Seurat,Seurat'
follow(x, subject, select = c("arbitrary", "all"), ignore.strand = FALSE)

## S4 method for signature 'ANY,ChromatinAssay'
nearest(x, subject, select = c("arbitrary", "all"), ignore.strand = FALSE)

## S4 method for signature 'ChromatinAssay,ANY'
nearest(x, subject, select = c("arbitrary", "all"), ignore.strand = FALSE)

## S4 method for signature 'ChromatinAssay,ChromatinAssay'
nearest(x, subject, select = c("arbitrary", "all"), ignore.strand = FALSE)

## S4 method for signature 'ANY,Seurat'
nearest(x, subject, select = c("arbitrary", "all"), ignore.strand = FALSE)

## S4 method for signature 'Seurat,ANY'
nearest(x, subject, select = c("arbitrary", "all"), ignore.strand = FALSE)

## S4 method for signature 'Seurat,Seurat'
nearest(x, subject, select = c("arbitrary", "all"), ignore.strand = FALSE)
## S4 method for signature 'ANY,ChromatinAssay'
distance(x, y, ignore.strand = FALSE, ...)

## S4 method for signature 'ChromatinAssay,ANY'
distance(x, y, ignore.strand = FALSE, ...)

## S4 method for signature 'ChromatinAssay,ChromatinAssay'
distance(x, y, ignore.strand = FALSE, ...)

## S4 method for signature 'ANY,Seurat'
distance(x, y, ignore.strand = FALSE, ...)

## S4 method for signature 'Seurat,ANY'
distance(x, y, ignore.strand = FALSE, ...)

## S4 method for signature 'Seurat,Seurat'
distance(x, y, ignore.strand = FALSE, ...)

## S4 method for signature 'ANY,ChromatinAssay'
distanceToNearest(x, subject, ignore.strand = FALSE, ...)

## S4 method for signature 'ChromatinAssay,ANY'
distanceToNearest(x, subject, ignore.strand = FALSE, ...)

## S4 method for signature 'ChromatinAssay,ChromatinAssay'
distanceToNearest(x, subject, ignore.strand = FALSE, ...)

## S4 method for signature 'ANY,Seurat'
distanceToNearest(x, subject, ignore.strand = FALSE, ...)

## S4 method for signature 'Seurat,ANY'
distanceToNearest(x, subject, ignore.strand = FALSE, ...)

## S4 method for signature 'Seurat,Seurat'
distanceToNearest(x, subject, ignore.strand = FALSE, ...)

### Arguments

- **x**: A query `ChromatinAssay` object
- **subject**: The subject `GRanges` or `ChromatinAssay` object. If missing, `x` is used as the subject.
- **select**: Logic for handling ties. See `nearest-methods` in the `GenomicRanges` package.
- **ignore.strand**: Logical argument controlling whether strand information should be ignored.
- **y**: For the distance method, a `GRanges` object or a `ChromatinAssay` object
- **...**: Additional arguments for methods
Functions

- `precede(x = ChromatinAssay, subject = ANY): method for ChromatinAssay, ANY`
- `precede(x = ChromatinAssay, subject = ChromatinAssay): method for ChromatinAssay, ChromatinAssay`
- `precede(x = ANY, subject = Seurat): method for ANY, Seurat`
- `precede(x = Seurat, subject = ANY): method for Seurat, ANY`
- `precede(x = Seurat, subject = Seurat): method for Seurat, Seurat`
- `follow(x = ANY, subject = ChromatinAssay): method for ANY, ChromatinAssay`
- `follow(x = ChromatinAssay, subject = ANY): method for ChromatinAssay, ANY`
- `follow(x = ChromatinAssay, subject = ChromatinAssay): method for ChromatinAssay, ChromatinAssay`
- `follow(x = ANY, subject = Seurat): method for ANY, Seurat`
- `follow(x = Seurat, subject = ANY): method for Seurat, ANY`
- `follow(x = Seurat, subject = Seurat): method for Seurat, Seurat`
- `nearest(x = ANY, subject = ChromatinAssay): method for ANY, ChromatinAssay`
- `nearest(x = ChromatinAssay, subject = ANY): method for ChromatinAssay, ANY`
- `nearest(x = ChromatinAssay, subject = ChromatinAssay): method for ChromatinAssay, ChromatinAssay`
- `nearest(x = ANY, subject = Seurat): method for ANY, Seurat`
- `nearest(x = Seurat, subject = ANY): method for Seurat, ANY`
- `nearest(x = Seurat, subject = Seurat): method for Seurat, Seurat`
- `distance(x = ANY, y = ChromatinAssay): method for ANY, ChromatinAssay`
- `distance(x = ChromatinAssay, y = ANY): method for ChromatinAssay, ANY`
- `distance(x = ChromatinAssay, y = ChromatinAssay): method for ChromatinAssay, ChromatinAssay`
- `distance(x = ANY, y = Seurat): method for ANY, Seurat`
- `distance(x = Seurat, y = ANY): method for Seurat, ANY`
- `distance(x = Seurat, y = Seurat): method for Seurat, Seurat`
- `distanceToNearest(x = ANY, subject = ChromatinAssay): method for ANY, ChromatinAssay`
- `distanceToNearest(x = ChromatinAssay, subject = ANY): method for ChromatinAssay, ANY`
- `distanceToNearest(x = ChromatinAssay, subject = ChromatinAssay): method for ChromatinAssay, ChromatinAssay`
- `distanceToNearest(x = ANY, subject = Seurat): method for ANY, Seurat`
- `distanceToNearest(x = Seurat, subject = ANY): method for Seurat, ANY`
- `distanceToNearest(x = Seurat, subject = Seurat): method for Seurat, Seurat`
See Also

- nearest-methods in the IRanges package.
- nearest-methods in the GenomicRanges package
- ChromatinAssay-class

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,
  n = ncol(object) * 5000,
  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.
- **n**: Number of lines to read from the fragment file. If NULL, read all lines. Default scales with the number of cells in the object.
- **verbose**: Display messages
- **...**: Arguments passed to other functions

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

```r
fpath <- system.file("extdata", "fragments.tsv.gz", package="Signac")
Fragments(atac_small) <- CreateFragmentObject(
  path = fpath,
  cells = colnames(atac_small),
  tolerance = 0.5
)
NucleosomeSignal(object = atac_small)
```
PeakPlot

Plot peaks in a genomic region

Description
Display the genomic ranges in a ChromatinAssay object that fall in a given genomic region

Usage
PeakPlot(
  object,
  region,
  assay = NULL,
  peaks = NULL,
  group.by = NULL,
  color = "dimgrey",
  sep = c("-", "-"),
  extend.upstream = 0,
  extend.downstream = 0
)

Arguments

object  A Seurat object
region  A genomic region to plot
assay   Name of assay to use. If NULL, use the default assay.
peaks   A GRanges object containing peak coordinates. If NULL, use coordinates stored
         in the Seurat object.
group.by Name of variable in feature metadata (if using ranges in the Seurat object) or
          genomic ranges metadata (if using supplied ranges) to color ranges by. If NULL,
          do not color by any metadata variable.
color   Color to use. If group.by is not NULL, this can be a custom color scale (see
         examples).
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.
extend.upstream Number of bases to extend the region upstream.
extend.downstream Number of bases to extend the region downstream.

Value
Returns a ggplot object
Examples

# plot peaks in assay
PeakPlot(atac_small, region = "chr1-710000-715000")

# manually set color
PeakPlot(atac_small, region = "chr1-710000-715000", color = "red")

# color by a variable in the feature metadata
PeakPlot(atac_small, region = "chr1-710000-715000", group.by = "count")

PlotFootprint

Plot motif footprinting results

Description

Plot motif footprinting results

Usage

PlotFootprint(
  object,
  features,
  assay = NULL,
  group.by = NULL,
  split.by = NULL,
  idents = NULL,
  label = TRUE,
  repel = TRUE,
  show.expected = TRUE,
  normalization = "subtract",
  label.top = 3,
  label.idents = NULL
)

Arguments

object A Seurat object
features A vector of features to plot
assay Name of assay to use
group.by A grouping variable
split.by A metadata variable to split the plot by. For example, grouping by "celltype" and splitting by "batch" will create separate plots for each celltype and batch.
idents Set of identities to include in the plot
label TRUE/FALSE value to control whether groups are labeled.
repel  Repel labels from each other
show.expected  Plot the expected Tn5 integration frequency below the main footprint plot
normalization  Method to normalize for Tn5 DNA sequence bias. Options are "subtract", "divide", or NULL to perform no bias correction.
label.top  Number of groups to label based on highest accessibility in motif flanking region.
label.idents  Vector of identities to label. If supplied, label.top will be ignored.

ReadMGATK  Read MGATK output

Description

Read output files from MGATK (https://github.com/caleblareau/mgatk).

Usage

ReadMGATK(dir, verbose = TRUE)

Arguments

dir  Path to directory containing MGATK output files
verbose  Display messages

Value

Returns a list containing a sparse matrix (counts) and two dataframes (depth and refallele).
The sparse matrix contains read counts for each base at each position and strand.
The depth dataframe contains the total depth for each cell. The refallele dataframe contains the reference genome allele at each position.

Examples

## Not run:
data.dir <- system.file("extdata", "test_mgatk", package="Signac")
mgatk <- ReadMGATK(dir = data.dir)

## End(Not run)
RegionHeatmap

Description

Plot fragment counts within a set of regions.

Usage

RegionHeatmap(
    object,
    key,
    assay = NULL,
    idents = NULL,
    normalize = TRUE,
    upstream = 3000,
    downstream = 3000,
    max.cutoff = "q95",
    cols = NULL,
    min.counts = 1,
    window = (upstream + downstream)/30,
    order = TRUE,
    nrow = NULL
)

Arguments

object A Seurat object
key Name of key to pull data from. Stores the results from RegionMatrix
assay Name of assay to use. If a list or vector of assay names is given, data will be plotted from each assay. Note that all assays must contain RegionMatrix results with the same key. Sorting will be defined by the first assay in the list
idents Cell identities to include. Note that cells cannot be regrouped, this will require re-running RegionMatrix to generate a new set of matrices
normalize Normalize by number of cells in each group
upstream Number of bases to include upstream of region. If NULL, use all bases that were included in the RegionMatrix function call. Note that this value cannot be larger than the value for upstream given in the original RegionMatrix function call. If NULL, use parameters that were given in the RegionMatrix function call
downstream Number of bases to include downstream of region. See documentation for upstream
max.cutoff Maximum cutoff value. Data above this value will be clipped to the maximum value. A quantile maximum can be specified in the form of "q##" where "##" is the quantile (e.g., "q90" for 90th quantile). If NULL, no cutoff will be set
RegionMatrix

cols Vector of colors to use as the maximum value of the color scale. One color must be supplied for each assay. If NULL, the default ggplot2 colors are used.

min.counts Minimum total counts to display region in plot

window Smoothing window to apply

order Order regions by the total number of fragments in the region across all included identities

nrow Number of rows to use when creating plot. If NULL, chosen automatically by ggplot2

Value

Returns a ggplot2 object

See Also

RegionMatrix

Description

Count fragments within a set of regions for different groups of cells.

Usage

RegionMatrix(object, ...)

## S3 method for class 'Seurat'
RegionMatrix(
  object,
  regions,
  key,
  assay = NULL,
  group.by = NULL,
  idents = NULL,
  upstream = 3000,
  downstream = 3000,
  verbose = TRUE,
  ...
)

## S3 method for class 'ChromatinAssay'
RegionMatrix(
  object,
  regions,
RegionMatrix

key, assay = NULL, group.by = NULL, idents = NULL, upstream = 3000, downstream = 3000, verbose = TRUE, 

)

## Default S3 method:
RegionMatrix(
  object, regions, key, assay = NULL, group.by = NULL, idents = NULL, upstream = 3000, downstream = 3000, verbose = TRUE, ...
)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>A Seurat or ChromatinAssay object</td>
</tr>
<tr>
<td>...</td>
<td>Arguments passed to other methods</td>
</tr>
<tr>
<td>regions</td>
<td>A GRanges object containing the set of genomic ranges to quantify</td>
</tr>
<tr>
<td>key</td>
<td>Name to store resulting matrices under</td>
</tr>
<tr>
<td>assay</td>
<td>Name of assay to use. If NULL, use the default assay</td>
</tr>
<tr>
<td>group.by</td>
<td>Grouping variable to use when aggregating data across cells. If NULL, use the active cell identities</td>
</tr>
<tr>
<td>idents</td>
<td>Cell identities to include. If NULL, include all identities</td>
</tr>
<tr>
<td>upstream</td>
<td>Number of bases to extend regions upstream</td>
</tr>
<tr>
<td>downstream</td>
<td>Number of bases to extend regions downstream</td>
</tr>
<tr>
<td>verbose</td>
<td>Display messages</td>
</tr>
</tbody>
</table>

Value

Returns a Seurat object
RegionPlot

Region plot

Description
Plot fragment counts within a set of regions.

Usage
RegionPlot(
  object,
  key,
  assay = NULL,
  idents = NULL,
  normalize = TRUE,
  upstream = NULL,
  downstream = NULL,
  window = (upstream + downstream)/500,
  nrow = NULL
)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>A Seurat object</td>
</tr>
<tr>
<td>key</td>
<td>Name of key to pull data from. Stores the results from RegionMatrix</td>
</tr>
<tr>
<td>assay</td>
<td>Name of assay to use. If a list or vector of assay names is given, data will be plotted from each assay. Note that all assays must contain RegionMatrix results with the same key. Sorting will be defined by the first assay in the list</td>
</tr>
<tr>
<td>idents</td>
<td>Cell identities to include. Note that cells cannot be regrouped, this will require re-running RegionMatrix to generate a new set of matrices</td>
</tr>
<tr>
<td>normalize</td>
<td>Normalize by number of cells in each group</td>
</tr>
<tr>
<td>upstream</td>
<td>Number of bases to include upstream of region. If NULL, use all bases that were included in the RegionMatrix function call. Note that this value cannot be larger than the value for upstream given in the original RegionMatrix function call. If NULL, use parameters that were given in the RegionMatrix function call</td>
</tr>
<tr>
<td>downstream</td>
<td>Number of bases to include downstream of region. See documentation for upstream</td>
</tr>
<tr>
<td>window</td>
<td>Smoothing window to apply</td>
</tr>
<tr>
<td>nrow</td>
<td>Number of rows to use when creating plot. If NULL, chosen automatically by ggplot2</td>
</tr>
</tbody>
</table>

Value
Returns a ggplot2 object
RegionStats

See Also

RegionMatrix

---

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, verbose = TRUE, ...)

## S3 method for class 'ChromatinAssay'
RegionStats(object, genome, verbose = TRUE, ...)

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

Arguments

object  
A Seurat object, Assay object, or set of genomic ranges

...  
Arguments passed to other methods

genome  
A BSgenome object or any other object supported by getSeq. Do showMethods("getSeq") to get the list of all supported object types.

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
)
RunChromVAR

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

```r
RunChromVAR(object, ...)  
## S3 method for class 'ChromatinAssay'
RunChromVAR(object, genome, motif.matrix = NULL, verbose = TRUE, ...)

## S3 method for class 'Seurat'
RunChromVAR(  
  object,  
  genome,  
  motif.matrix = NULL,  
  assay = NULL,  
  new.assay.name = "chromvar",  
  ...  
)
```

Arguments

- `object`: A Seurat object
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,
...  Additional arguments passed to getBackgroundPeaks

geno
me
A BSgenome object or string stating the genome build recognized by getBSgenome.

motif.matrix
A peak x motif matrix. If NULL, pull the peak x motif matrix from a Motif object stored in the assay.

verbose
Display messages

assay
Name of assay to use

new.assay.name
Name of new assay used to store the chromVAR results. Default is "chromvar".

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

```r
## Not run:
library(BSgenome.Hsapiens.UCSC.hg19)
RunChromVAR(object = atac_small["peaks"], genome = BSgenome.Hsapiens.UCSC.hg19)
## End(Not run)
## Not run:
library(BSgenome.Hsapiens.UCSC.hg19)
RunChromVAR(object = atac_small, genome = BSgenome.Hsapiens.UCSC.hg19)
## End(Not run)
```
RunSVD

scale.embeddings = TRUE,
reduction.key = "LSI_",
scale.max = NULL,
verbose = TRUE,
irlba.work = n * 3,
tol = 1e-05,
...)

## S3 method for class 'Assay'
RunSVD(
  object,
  assay = NULL,
  features = NULL,
  n = 50,
  reduction.key = "LSI_",
  scale.max = NULL,
  verbose = TRUE,
  ...
)

## S3 method for class 'Seurat'
RunSVD(
  object,
  assay = NULL,
  features = NULL,
  n = 50,
  reduction.key = "LSI_",
  reduction.name = "lsi",
  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
**RunTFIDF**

Compute the term-frequency inverse-document-frequency

---

**Description**

Run term frequency inverse document frequency (TF-IDF) normalization on a matrix.

**Usage**

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

```r
# S3 method for class 'Assay'
RunTFIDF(
  object,
  assay = NULL,
  method = 1,
  scale.factor = 10000,
  idf = NULL,
```
### S3 method for class 'Seurat'

RunTFIDF(
  object,
  assay = NULL,
  method = 1,
  scale.factor = 10000,
  idf = NULL,
  verbose = TRUE,
  ...
)

#### Arguments

- **object**: A Seurat object
- **assay**: Name of assay to use
- **method**: Which TF-IDF implementation to use. Choice of:
  - 1: The TF-IDF implementation used by Stuart & Butler et al. 2019 (doi:10.1101/460147). This computes \( \log(TF \times IDF) \).
  - 2: The TF-IDF implementation used by Cusanovich & Hill et al. 2018 (doi:10.1016/j.cell.2018.06.052). This computes \( TF \times (\log(IDF)) \).
  - 3: The log-TF method used by Andrew Hill. This computes \( \log(TF) \times \log(IDF) \).
  - 4: The 10x Genomics method (no TF normalization). This computes \( IDF \).
- **scale.factor**: Which scale factor to use. Default is 10000.
- **idf**: A precomputed IDF vector to use. If NULL, compute based on the input data matrix.
- **verbose**: Print progress

#### Details

Four different TF-IDF methods are implemented. We recommend using method 1 (the default).

#### Value

Returns a Seurat object

#### References

https://en.wikipedia.org/wiki/Latent_semantic_analysis#Latent_semantic_indexing
Examples

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

Description

Methods for accessing and modifying `Seqinfo` object information stored in a `ChromatinAssay` object.

Usage

```r
## S4 method for signature 'ChromatinAssay'
seqinfo(x)

## S4 replacement method for signature 'ChromatinAssay'
seqinfo(x) <- value

## S4 method for signature 'ChromatinAssay'
seqlevels(x)

## S4 replacement method for signature 'ChromatinAssay'
seqlevels(x) <- value

## S4 method for signature 'ChromatinAssay'
seqnames(x)

## S4 replacement method for signature 'ChromatinAssay'
seqnames(x) <- value

## S4 method for signature 'ChromatinAssay'
seqlengths(x)

## S4 replacement method for signature 'ChromatinAssay'
seqlengths(x) <- value

## S4 method for signature 'ChromatinAssay'
genome(x)

## S4 replacement method for signature 'ChromatinAssay'
genome(x) <- value

## S4 method for signature 'ChromatinAssay'
```
isCircular(x)
## S4 replacement method for signature 'ChromatinAssay'
isCircular(x) <- value

## S4 method for signature 'Seurat'
seqinfo(x)
## S4 replacement method for signature 'Seurat'
seqinfo(x) <- value

## S4 method for signature 'Seurat'
seqlevels(x)
## S4 replacement method for signature 'Seurat'
seqlevels(x) <- value

## S4 method for signature 'Seurat'
seqnames(x)
## S4 replacement method for signature 'Seurat'
seqnames(x) <- value

## S4 method for signature 'Seurat'
seqlengths(x)
## S4 replacement method for signature 'Seurat'
seqlengths(x) <- value

## S4 method for signature 'Seurat'
genome(x)
## S4 replacement method for signature 'Seurat'
genome(x) <- value

## S4 method for signature 'Seurat'
isCircular(x)
## S4 replacement method for signature 'Seurat'
isCircular(x) <- value

**Arguments**

- **x**
  
  A ChromatinAssay object

- **value**
  
  A Seqinfo object or name of a UCSC genome to store in the ChromatinAssay

**Functions**

- `seqinfo(ChromatinAssay) <- value`: set method for ChromatinAssay objects
SetMotifData

- seqlevels(ChromatinAssay): get method for ChromatinAssay objects
- seqlevels(ChromatinAssay) <- value: set method for ChromatinAssay objects
- seqnames(ChromatinAssay): get method for ChromatinAssay objects
- seqnames(ChromatinAssay) <- value: set method for ChromatinAssay objects
- seqlengths(ChromatinAssay): get method for ChromatinAssay objects
- seqlengths(ChromatinAssay) <- value: set method for ChromatinAssay objects
- genome(ChromatinAssay): get method for ChromatinAssay objects
- genome(ChromatinAssay) <- value: set method for ChromatinAssay objects
- isCircular(ChromatinAssay): get method for ChromatinAssay objects
- isCircular(ChromatinAssay) <- value: set method for ChromatinAssay objects
- seqinfo(Seurat): get method for Seurat objects
- seqinfo(Seurat) <- value: set method for Seurat objects
- seqlevels(Seurat): get method for Seurat objects
- seqlevels(Seurat) <- value: set method for Seurat objects
- seqnames(Seurat): get method for Seurat objects
- seqnames(Seurat) <- value: set method for Seurat objects
- seqlengths(Seurat): get method for Seurat objects
- seqlengths(Seurat) <- value: set method for Seurat objects
- genome(Seurat): get method for Seurat objects
- genome(Seurat) <- value: set method for Seurat objects
- isCircular(Seurat): get method for Seurat objects
- isCircular(Seurat) <- value: set method for Seurat objects

See Also

- seqinfo in the GenomeInfoDb package.
- ChromatinAssay-class

---

SetMotifData  

Set motif data

Description

Set motif matrix for given assay
Usage

```r
SetMotifData(object, ...)  # S3 method for class 'Motif'
SetMotifData(object, slot, new.data, ...)  # S3 method for class 'ChromatinAssay'
SetMotifData(object, assay = NULL, ...)  # S3 method for class 'Seurat'
```

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

```r
motif.obj <- SeuratObject::GetAssayData(
  object = atac_small[['peaks']], slot = "motifs"
)
SetMotifData(object = motif.obj, slot = 'data', new.data = matrix(1:2))
SetMotifData(
  object = atac_small[['peaks']], slot = 'data', new.data = matrix(1:2)
)
motif.matrix <- GetMotifData(object = atac_small)
SetMotifData(
  object = atac_small, assay = 'peaks', slot = 'data', new.data = motif.matrix
)
```

SplitFragment

_**Split fragment file by cell identities**_

Description

Splits a fragment file into separate files for each group of cells. If splitting multiple fragment files containing common cell types, fragments originating from different files will be appended to the same file for one group of cell identities.
StringToGRanges

Usage

SplitFragments(
  object,
  assay = NULL,
  group.by = NULL,
  idents = NULL,
  outdir = getwd(),
  file.suffix = "",
  append = TRUE,
  buffer_length = 256L,
  verbose = TRUE
)

Arguments

object A Seurat object
assay Name of assay to use
group.by Name of grouping variable to group cells by
idents List of identities to include
outdir Directory to write output files
file.suffix Suffix to add to all file names (before file extension). If splitting multiple fragment files without the append option set to TRUE, an additional numeric suffix will be added to each file (e.g., .1, .2).
append If splitting multiple fragment files, append cells from the same group (e.g., cluster) to the same file. Note that this can cause the output file to be unsorted.
buffer_length Size of buffer to be read from the fragment file. This must be longer than the longest line in the file.
verbose Display messages

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.
...

StringToGRanges String to GRanges

Description

Convert a genomic coordinate string to a GRanges object

Usage

StringToGRanges(regions, sep = c("-", "-"), ...)
**subset.Fragment**

**Value**

Returns a GRanges object

**Examples**

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

---

**subset.Fragment**  
*Subset a Fragment object*

**Description**

Returns a subset of a Fragment-class object.

**Usage**

```r
## S3 method for class 'Fragment'
subset(x, cells = NULL, ...)
```

**Arguments**

- **x**: A Fragment object
- **cells**: Vector of cells to retain
- **...**: Arguments passed to other methods

**Value**

Returns a subsetted Fragment object

**See Also**

subset

**Examples**

```r
fpath <- system.file("extdata", "fragments.tsv.gz", package="Signac")
cells <- colnames(x = atac_small)
names(x = cells) <- paste0("test_", cells)
frags <- CreateFragmentObject(path = fpath, cells = cells, verbose = FALSE, tolerance = 0.5)
subset(frags, head(names(cells)))
```
### Subset a Motif object

#### Description

Returns a subset of a Motif-class object.

#### Usage

```r
## 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

```r
motif.obj <- SeuratObject::GetAssayData(
  object = atac_small[['peaks']], slot = "motifs"
)
subset(x = motif.obj, features = head(rownames(motif.obj), 10))
motif.obj <- SeuratObject::GetAssayData(
  object = atac_small, assay = 'peaks', slot = 'motifs'
)
motif.obj[1:10,1:10]
```
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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mat</td>
<td>A matrix</td>
</tr>
<tr>
<td>min.rows</td>
<td>Minimum number of non-zero elements for the row to be retained</td>
</tr>
<tr>
<td>min.cols</td>
<td>Minimum number of non-zero elements for the column to be retained</td>
</tr>
<tr>
<td>max.row.val</td>
<td>Maximum allowed value in a row for the row to be retained. If NULL, don’t set any limit.</td>
</tr>
<tr>
<td>max.col.val</td>
<td>Maximum allowed value in a column for the column to be retained. If NULL, don’t set any limit.</td>
</tr>
</tbody>
</table>

Value

Returns a matrix

Examples

SubsetMatrix(mat = volcano)
**theme_browser**

*Genome browser theme*

**Description**
Theme applied to panels in the `CoveragePlot` function.

**Usage**
```
theme_browser(..., legend = TRUE, axis.text.y = FALSE)
```

**Arguments**
- `...` Additional arguments
- `legend` Display plot legend
- `axis.text.y` Display y-axis text

**Examples**
```
PeakPlot(atac_small, region = "chr1-710000-715000") + theme_browser()
```

---

**TilePlot**

*Plot integration sites per cell*

**Description**
Plots the presence/absence of Tn5 integration sites for each cell within a genomic region.

**Usage**
```
TilePlot(
  object, 
  region, 
  sep = c("-", ";"), 
  tile.size = 100, 
  tile.cells = 100, 
  extend.upstream = 0, 
  extend.downstream = 0, 
  assay = NULL, 
  cells = NULL, 
  group.by = NULL, 
  order.by = "total", 
  idents = NULL 
)
```
Arguments

- **object**: A Seurat object
- **region**: A set of genomic coordinates to show. Can be a GRanges object, a string encoding a genomic position, a gene name, or a vector of strings describing the genomic coordinates or gene names to plot. If a gene name is supplied, annotations must be present in the assay.
- **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.
- **tile.size**: Size of the sliding window for per-cell fragment tile plot
- **tile.cells**: Number of cells to display fragment information for in tile plot.
- **extend.upstream**: Number of bases to extend the region upstream.
- **extend.downstream**: Number of bases to extend the region downstream.
- **assay**: Name of assay to use
- **cells**: Which cells to plot. Default all cells
- **group.by**: Name of grouping variable to group cells by. If NULL, use the current cell identities
- **order.by**: Option for determining how cells are chosen from each group. Options are "total" or "random". "total" will select the top cells based on total number of fragments in the region, "random" will select randomly.
- **idents**: List of cell identities to include in the plot. If NULL, use all identities.

Value

Returns a `ggplot` object

Examples

```r
fpath <- system.file("extdata", "fragments.tsv.gz", package="Signac")
fragments <- CreateFragmentObject(
    path = fpath,
    cells = colnames(atac_small),
    validate.fragments = FALSE
)
Fragments(atac_small) <- fragments
TilePlot(object = atac_small, region = c("chr1:713500-714500"))
```
TSSEnrichment

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 = NULL,
  n = NULL,
  fast = TRUE,
  assay = NULL,
  cells = NULL,
  process_n = 2000,
  verbose = TRUE
)

Arguments

object A Seurat object

tss.positions A GRanges object containing the TSS positions. If NULL, use the genomic annotations stored in the assay.

n Number of TSS positions to use. This will select the first _n_ TSSs from the set. If NULL, use all TSSs (slower).

fast Just compute the TSS enrichment score, without storing the base-resolution matrix of integration counts at each site. This reduces the memory required to store the object but does not allow plotting the accessibility profile at the TSS.

assay Name of assay to use

cells A vector of cells to include. If NULL (default), use all cells in the object

process_n Number of regions to process at a time if using fast option.

verbose Display messages

Details

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

Value

Returns a Seurat object
Examples

```r
## Not run:
spath <- system.file("extdata", "fragments.tsv.gz", package="Signac")
Fragments(atac_small) <- CreateFragmentObject(
  path = sPath,
  cells = colnames(atac_small),
  tolerance = 0.5
)
TSSEnrichment(object = atac_small)

## End(Not run)
```

---

**TSSPlot**

*Plot signal enrichment around TSSs*

**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**

```r
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
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")

Arguments
object.list A list of Seurat objects or ChromatinAssay 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.

Value
Returns a GRanges object

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

UpdatePath

Update the file path for a Fragment object

Description
Change the path to a fragment file store in a Fragment object. Path must be to the same file that was used to create the fragment object. An MD5 hash will be computed using the new path and compared to the hash stored in the Fragment object to verify that the files are the same.

Usage
UpdatePath(object, new.path, verbose = TRUE)

Arguments
object A Fragment object
new.path Path to the fragment file
verbose Display messages
ValidateCells

Validate cells present in fragment file

Description
Search for a fragment from each cell that should exist in the fragment file. Will iterate through chunks of the fragment file until at least one fragment from each cell barcode requested is found.

Usage
ValidateCells(
  object,
  cells = NULL,
  tolerance = 0.5,
  max.lines = 5e+07,
  verbose = TRUE
)

Arguments

object A Fragment object

cells A character vector containing cell barcodes to search for. If NULL, use the cells stored in the Fragment object.

tolerance Fraction of input cells that can be unseen before returning TRUE. For example, tolerance = 0.01 will return TRUE when 99 have observed fragments in the file. This can be useful if there are cells present that have much fewer total counts, and would require extensive searching before a fragment from those cells are found.

max.lines Maximum number of lines to read in without finding the required number of cells before returning FALSE. Setting this value avoids having to search the whole file if it becomes clear that the expected cells are not present. Setting this value to NULL will enable an exhaustive search of the entire file.

verbose Display messages

ValidateFragments

Validate Fragment object

Description
Verify that the cells listed in the object exist in the fragment file and that the fragment file or index have not changed since creating the fragment object.

Usage
ValidateFragments(object, verbose = TRUE, ...)
ValidateHash

Arguments

object A Fragment object
verbose Display messages
...

Additional parameters passed to ValidateCells

ValidateHash Validate hashes for Fragment object

Description

Validate hashes for Fragment object

Usage

ValidateHash(object, verbose = TRUE)

Arguments

variant object A Fragment object
verbose Display messages

VariantPlot Plot strand concordance vs. VMR

Description

Plot the Pearson correlation between allele frequencies on each strand versus the log10 mean-variance ratio for the allele.

Usage

VariantPlot(
  variants,
  min.cells = 2,
  concordance.threshold = 0.65,
  vmr.threshold = 0.01
)

Arguments

variants A dataframe containing variant information. This should be computed using IdentifyVariants
min.cells Minimum number of high-confidence cells detected with the variant for the variant to be displayed.
concordance.threshold Strand concordance threshold
vmr.threshold Mean-variance ratio threshold
Index

* assay
  Annotation, 10
  as.ChromatinAssay, 12
  Cells<-, 24
  ChromatinAssay-class, 25
  CreateChromatinAssay, 37
  Fragments, 57
  GetFragmentData, 63
  Links, 78
  Motifs, 82
  Signac-package, 4

* coverage
  coverage, ChromatinAssay-method, 31

* datasets
  atac_small, 13
  blacklist_ce10, 16
  blacklist_ce11, 17
  blacklist_dm3, 17
  blacklist_dm6, 18
  blacklist_hg19, 18
  blacklist_hg38, 19
  blacklist_hg38_unified, 19
  blacklist_mm10, 20

* data
  atac_small, 13
  blacklist_ce10, 16
  blacklist_ce11, 17
  blacklist_dm3, 17
  blacklist_dm6, 18
  blacklist_hg19, 18
  blacklist_hg38, 19
  blacklist_hg38_unified, 19
  blacklist_mm10, 20

* dimension_reduction
  Jaccard, 75
  RunSVD, 97

* footprinting
  Footprint, 54
  GetFootprintData, 63
  InsertionBias, 71
  PlotFootprint, 89

* fragments
  Cells.Fragment, 23
  CountFragments, 29
  CreateFragmentObject, 38
  FilterCells, 46
  Fragment-class, 56
  Fragments, 57
  head.Fragment, 69
  SplitFragments, 104
  subset.Fragment, 106
  UpdatePath, 113
  ValidateCells, 114
  ValidateFragments, 114
  ValidateHash, 115

* granges
  granges-methods, 68

* heatmap
  RegionHeatmap, 91
  RegionMatrix, 92
  RegionPlot, 94

* inter_range
  inter-range-methods, 72

* links
  ConnectionsToLinks, 27
  GetLinkedGenes, 65
  GetLinkedPeaks, 66
  LinkPeaks, 75
  LinkPlot, 77
  Links, 78

* mito
  AlleleFreq, 9
  ClusterClonotypes, 26
  FindClonotypes, 47
  IdentifyVariants, 70
  ReadMGATK, 90
  VariantPlot, 115

* motifs
INDEX

117

AddMotifs, 6
ConvertMotifID, 28
CreateMotifMatrix, 39
CreateMotifObject, 40
FindMotifs, 48
GetMotifData, 66
MatchRegionStats, 79
Motif-class, 80
MotifCounts, 81
MotifPlot, 82
Motifs, 82
RegionStats, 95
RunChromVAR, 96
SetMotifData, 103
subset.Motif, 107
* nearest
nearest-methods, 83
* overlaps
findOverlaps-methods, 49
* preprocessing
BinarizeCounts, 15
CreateMotifMatrix, 39
DownsampleFeatures, 43
FindTopFeatures, 53
RunTFIDF, 99
UnifyPeaks, 113
* qc
FragmentHistogram, 56
FRIP, 59
NucleosomeSignal, 87
TSSEnrichment, 111
TSSPlot, 112
* quantification
AggregateTiles, 7
CallPeaks, 20
FeatureMatrix, 45
GenomeBinMatrix, 61
* seqinfo
seqinfo-methods, 101
* utilities
AccessiblePeaks, 5
AddChromatinModule, 6
AddModuleScore, 6
AddMotifs, 6
AggregateTiles, 7
AlleleFreq, 9
Annotation, 10
Annotation<- (Annotation), 10
AnnotationPlot, 11
as.ChromatinAssay, 12
Assay, 25
atic_small, 13
AverageCounts, 13
BigwigTrack, 14
GeneActivity, 59
GetCellsInRegion, 62
GetGRangesFromEnsDb, 64
GetIntersectingFeatures, 64
GetTSSPositions, 67
GRangesToString, 69
IntersectMatrix, 74
LookupGeneCoords, 79
MatchRegionStats, 79
StringToGRanges, 105
SubsetMatrix, 108
UnifyPeaks, 113
* visualization
AnnotationPlot, 11
BigwigTrack, 14
CombineTracks, 27
CoverageBrowser, 32
CoveragePlot, 32
DensityScatter, 41
DepthCor, 42
ExpressionPlot, 43
FragmentHistogram, 56
LinkPlot, 77
MotifPlot, 82
PeakPlot, 88
PlotFootprint, 89
RegionHeatmap, 91
RegionPlot, 94
time_browser, 109
TilePlot, 109
TSSPlot, 112
VariantPlot, 115
[.Motif (subset.Motif), 107

AccessiblePeaks, 5
AddChromatinModule, 6
AddModuleScore, 6
AddMotifs, 6
AggregateTiles, 7
AlleleFreq, 9
Annotation, 10
Annotation<- (Annotation), 10
AnnotationPlot, 11
as.ChromatinAssay, 12
Assay, 25
atic_small, 13
AverageCounts, 13
BigwigTrack, 14
INDEX

BinarizeCounts, 15
blacklist_ce10, 16
blacklist_ce11, 17
blacklist_dm3, 17
blacklist_dm6, 18
blacklist_hg19, 18
blacklist_hg38, 19
blacklist_hg38_unified, 19
blacklist_mm10, 20

CallPeaks, 20
Cells.Fragment, 23
 Cells<-, 24
 Cells<-.Fragment (Cells.Fragment), 23
CellsPerGroup, 24
ChromatinAssay, 9, 12, 23, 31, 37, 49, 52, 68,
  72, 73, 83, 85, 88, 101, 102
ChromatinAssay (ChromatinAssay-class), 25
ChromatinAssay-class, 25, 32, 52, 68, 74,
  87, 103
chromVAR, 96
ClosestFeature, 25
ClusterClonotypes, 26
CombineTracks, 27
ConnectionsToLinks, 27
ConvertMotifID, 28
cor, 42
CountFragments, 29, 59
countOverlaps (findOverlaps-methods), 49
countOverlaps, ChromatinAssay, ChromatinAssay-
  (findOverlaps-methods), 49
countOverlaps, ChromatinAssay, Vector-method
  (findOverlaps-methods), 49
countOverlaps, Seurat, Seurat-method
  (findOverlaps-methods), 49
countOverlaps, Seurat, Vector-method
  (findOverlaps-methods), 49
countOverlaps, Vector, ChromatinAssay-method
  (findOverlaps-methods), 49
countOverlaps, Vector, Seurat-method
  (findOverlaps-methods), 49
CountsInRegion, 30, 55
coverage, 31
 coverage
  (coverage, ChromatinAssay-method), 31
 coverage, ChromatinAssay-method, 31
 coverage, Seurat-method
  (coverage, ChromatinAssay-method), 31
 coverage-methods, 32
CoverageBrowse, 32
CoveragePlot, 32, 32, 109
CreateChromatinAssay, 37
CreateFragmentObject, 38, 38
CreateMotifMatrix, 39
CreateMotifObject, 40
crunch, 64
density, 80
DensityScatter, 41
DepthCor, 42
disjoin, 113
disjoin (inter-range-methods), 72
disjoin, ChromatinAssay-method
  (inter-range-methods), 72
disjoin, Seurat-method
  (inter-range-methods), 72
disjointBins (inter-range-methods), 72
disjointBins, ChromatinAssay-method
  (inter-range-methods), 72
disjointBins, Seurat-method
  (inter-range-methods), 72
distance (nearest-methods), 83
distance, ANY, ChromatinAssay-method
  (nearest-methods), 83
distance, ANY, Seurat-method
  (nearest-methods), 83
distance, ChromatinAssay, ANY-method
  (nearest-methods), 83
distance, ChromatinAssay, ChromatinAssay-
  (nearest-methods), 83
distance, Seurat, ANY-method
  (nearest-methods), 83
distance, Seurat, Seurat-method
  (nearest-methods), 83
distanceToNearest (nearest-methods), 83
distanceToNearest, ANY, ChromatinAssay-method
  (nearest-methods), 83
distanceToNearest, ANY, Seurat-method
  (nearest-methods), 83
distanceToNearest, ChromatinAssay, ANY-method
  (nearest-methods), 83
distanceToNearest, ChromatinAssay, ChromatinAssay-
  (nearest-methods), 83
distanceToNearest, Seurat, ANY-method (nearest-methods), 83
distanceToNearest, Seurat, Seurat-method (nearest-methods), 83
DownsampleFeatures, 43
ExpressionPlot, 43
Extend, 44
FeatureMatrix, 45, 61, 81
FilterCells, 46
FindClonotypes, 47
FindClusters, 47
FindMotifs, 48
FindNeighbors, 47
findOverlaps, 30, 52, 74
findOverlaps(findOverlaps-methods), 49
findOverlaps, ChromatinAssay, ChromatinAssay-method (findOverlaps-methods), 49
findOverlaps, ChromatinAssay, Vector-method (findOverlaps-methods), 49
findOverlaps, Seurat, Seurat-method (findOverlaps-methods), 49
findOverlaps, Seurat, Vector-method (findOverlaps-methods), 49
findOverlaps, Vector, ChromatinAssay-method (findOverlaps-methods), 49
findOverlaps, Vector, Seurat-method (findOverlaps-methods), 49
findOverlaps-methods, 49, 52
FindTopFeatures, 53
follow (nearest-methods), 83
follow, ANY, ChromatinAssay-method (nearest-methods), 83
follow, ANY, Seurat-method (nearest-methods), 83
follow, ChromatinAssay, ANY-method (nearest-methods), 83
follow, ChromatinAssay, ChromatinAssay-method (nearest-methods), 83
follow, Seurat, ANY-method (nearest-methods), 83
follow, Seurat, Seurat-method (nearest-methods), 83
Footprint, 54
FractionCountsInRegion, 55
Fragment, 12, 23, 25, 38, 45, 58, 61, 63, 106, 113–115
Fragment(Fragment-class), 56
Fragment-class, 56
FragmentHistogram, 56
Fragments, 38, 57
Fragments<- (Fragments), 57
FRiP, 59
gaps (inter-range-methods), 72
gaps, ChromatinAssay-method (inter-range-methods), 72
gaps, Seurat-method (inter-range-methods), 72
GeneActivity, 59
genome (seqinfo-methods), 101
genome, ChromatinAssay-method (seqinfo-methods), 101
genome, Seurat-method (seqinfo-methods), 101
GenomeBinMatrix, 61
getBackgroundPeaks, 97
GetCellsInRegion, 62
GetFootprintData, 63
GetFragmentData, 63
GetGRangesFromEnsDb, 64
GetIntersectingFeatures, 64
GetLinkedGenes, 65
GetLinkedPeaks, 66
GetMotifData, 66
GetTSSPositions, 67
ggplot, 11, 32, 42, 57, 78, 82, 88, 110
ggplot2, 112
ggseqlogo, 82
GRanges, 10, 23, 25, 28, 38, 44, 68, 78, 85, 93
granges, 68, 77
granges (granges-methods), 68
granges, ChromatinAssay-method (granges-methods), 68
granges, Seurat-method (granges-methods), 68
granges-methods, 68
GRangesList, 41, 81
GRangesToString, 69
hclust, 26
head.Fragment, 69
IdentifyVariants, 70, 115
InsertionBias, 71
inter-range-methods, 72, 74
IntersectMatrix, 74
IRanges, 73
irlba, 97, 99
isCircular (seqinfo-methods), 101
isCircular, ChromatinAssay-method (seqinfo-methods), 101
isCircular, Seurat-method (seqinfo-methods), 101
isCircular<-, ChromatinAssay-method (seqinfo-methods), 101
isCircular<-, Seurat-method (seqinfo-methods), 101
isDisjoint (inter-range-methods), 72
isDisjoint, ChromatinAssay-method (inter-range-methods), 72
isDisjoint, Seurat-method (inter-range-methods), 72
Jaccard, 75
LinkPeaks, 75
LinkPlot, 77
Links, 77, 78
Links<-(Links), 78
LookupGeneCoords, 79
makeGRangesFromDataFrame, 105
matchMotifs, 6, 40
MatchRegionStats, 48, 79
Motif, 6, 12, 25, 41, 83, 107
Motif (Motif-class), 80
Motif-class, 80
MotifCounts, 81
MotifPlot, 82
Motifs, 82
Motifs<-(Motifs), 82
nearest (nearest-methods), 83
nearest, ANY, ChromatinAssay-method (nearest-methods), 83
nearest, ANY, Seurat-method (nearest-methods), 83
nearest, ChromatinAssay, ANY-method (nearest-methods), 83
nearest, ChromatinAssay, ChromatinAssay-method (nearest-methods), 83
nearest, Seurat, ANY-method (nearest-methods), 83
nearest, Seurat, Seurat-method (nearest-methods), 83
nearest-methods, 83, 87
NucleosomeSignal, 87
p.adjust, 48
patchwork, 36
PeakPlot, 88
PFMatrixList, 40
PlotFootprint, 89
precede (nearest-methods), 83
precede, ANY, ChromatinAssay-method (nearest-methods), 83
precede, ANY, Seurat-method (nearest-methods), 83
precede, ChromatinAssay, ANY-method (nearest-methods), 83
precede, ChromatinAssay, ChromatinAssay-method (nearest-methods), 83
precede, Seurat, ANY-method (nearest-methods), 83
precede, Seurat, Seurat-method (nearest-methods), 83
PWMatrixList, 40
range (inter-range-methods), 72
range, ChromatinAssay-method (inter-range-methods), 72
range, Seurat-method (inter-range-methods), 72
read.table, 69
ReadMGATK, 90
reduce, 113
reduce (inter-range-methods), 72
reduce, ChromatinAssay-method (inter-range-methods), 72
reduce, Seurat-method (inter-range-methods), 72
RegionHeatmap, 91
RegionMatrix, 91, 92, 94
RegionPlot, 94
RegionStats, 6, 48, 95
RunChromVAR, 96
RunSVD, 97
RunTFIDF, 99
sample, 80
Seqinfo, 12, 25, 38, 101, 102
seqinfo, 103
seqinfo (seqinfo-methods), 101
seqinfo, ChromatinAssay-method
  (seqinfo-methods), 101
seqinfo, Seurat-method
  (seqinfo-methods), 101
seqinfo-methods, 101
seqinfo<-, ChromatinAssay-method
  (seqinfo-methods), 101
seqinfo<-, Seurat-method
  (seqinfo-methods), 101
seqlengths, 61
seqlengths (seqinfo-methods), 101
seqlengths, ChromatinAssay-method
  (seqinfo-methods), 101
seqlengths, Seurat-method
  (seqinfo-methods), 101
seqlengths<-, ChromatinAssay-method
  (seqinfo-methods), 101
seqlengths<-, Seurat-method
  (seqinfo-methods), 101
seqlevels (seqinfo-methods), 101
seqlevels, ChromatinAssay-method
  (seqinfo-methods), 101
seqlevels, Seurat-method
  (seqinfo-methods), 101
seqlevels<-, ChromatinAssay-method
  (seqinfo-methods), 101
seqlevels<-, Seurat-method
  (seqinfo-methods), 101
seqnames (seqinfo-methods), 101
seqnames, ChromatinAssay-method
  (seqinfo-methods), 101
seqnames, Seurat-method
  (seqinfo-methods), 101
seqnames<-, ChromatinAssay-method
  (seqinfo-methods), 101
seqnames<-, Seurat-method
  (seqinfo-methods), 101
SetMotifData, 103
Seurat, 9, 11, 16, 42, 43, 47, 52, 53, 55, 59,
  67, 77, 87, 88, 93, 97, 99, 100, 104,
  111
Signac (Signac-package), 4
Signac-package, 4
SplitFragments, 104
StringToGRanges, 26, 71, 105
subset, 106, 107
subset.Fragment, 106
subset.Motif, 107
SubsetMatrix, 108
tempdir, 22
theme_browser, 109
TilePlot, 109
TSSEnrichment, 111, 112
TSSPlot, 112
UnifyPeaks, 113
UpdatePath, 113
ValidateCells, 114, 115
ValidateFragments, 114
ValidateHash, 115
VariableFeatures, 43
VariantPlot, 115
wrap_plots, 35