Package ‘ATACCoGAPS’

May 13, 2024

Title Analysis Tools for scATACseq Data with CoGAPS

Version 1.6.0

Description Provides tools for running the CoGAPS algorithm (Fertig et al, 2010) on single-cell ATAC sequencing data and analysis of the results. Can be used to perform analyses at the level of genes, motifs, TFs, or pathways. Additionally provides tools for transfer learning and data integration with single-cell RNA sequencing data.

License Artistic-2.0

Depends R (>= 4.2.0), CoGAPS (>= 3.5.13)

Imports gtools, GenomicRanges, projectR, TFBSTools, GeneOverlap, msigdbr, tidyverse, gplots, motifmatcher, chromVAR, GenomicFeatures, IRanges, fgsea, rGREAT, JASPAR2016, Homo.sapiens, Mus.musculus, BSgenome.Hsapiens.UCSC.hg19, BSgenome.Mmusculus.UCSC.mm10, stringr, dplyr

biocViews Software, ResearchField, Epigenetics, SingleCell, Transcription, Bayesian, Clustering, DimensionReduction

BiocType Software

RoxygenNote 7.2.0

Encoding UTF-8

Suggests knitr, viridis

BugReports https://github.com/FertigLab/ATACCoGAPS/issues

VignetteBuilder knitr

git_url https://git.bioconductor.org/packages/ATACCoGAPS
git_branch RELEASE_3_19
git_last_commit d9555fb
git_last_commit_date 2024-04-30

Repository Bioconductor 3.19

Date/Publication 2024-05-13

Author Rossin Erbe [aut, cre] (<https://orcid.org/0000-0002-3423-2104>)

Maintainer Rossin Erbe <rerbe1@jhm.edu>
applyGREAT

Find Enrichment of GO Terms in PatternMarker Peaks using GREAT

Description

Use the rGREAT package to find enrichment of GO terms or genes for the peaks found to be most pattern differentiating using the PatternMarker statistic.

Usage

applyGREAT(
  cogapsResult,
  granges,
  genome,
  scoreThreshold = NULL,
  GREATCategory = "GO"
)
**ATACTransferLearning**

**Description**

Wrapper function for projectR which finds overlaps between the peaks of the atac data CoGAPS was run on and maps them to new data set the user wishes to project learned patterns into.

**Usage**

```r
ATACTransferLearning(
  newData,
  CoGAPSResult, originalPeaks, originalGranges, newGranges
)
```

**Arguments**

- `cogapsResult`: result object from CoGAPS
- `granges`: GRanges object corresponding to the peaks of the scATAC-seq data CoGAPS was applied to
- `genome`: UCSC genome designation for input to the submitGreatJob function from the rGREAT package (e.g. "hg19")
- `scoreThreshold`: threshold of PatternMarker score to take peaks for analysis, higher values return more peaks. Defaults to use all PatternMarker genes with value NULL
- `GREATCategory`: input to the category argument of the rGREAT getEnrichmentTables function. Usually "GO" or "Genes"

**Value**

list containing enrichment results for each pattern

**Examples**

```r
data("schepCogapsResult")
data(schepGranges)

GOenrichment <- applyGREAT(cogapsResult = schepCogapsResult,
granges = schepGranges, genome = "hg19")
```
Arguments

- `newData` the ATAC data to project into
- `CoGAPSResult` result from CoGAPS run on original ATAC data
- `originalPeaks` peaks from the ATAC data Cogaps was run on
- `originalGranges` granges of the peaks for the data set Cogaps was run on
- `newGranges` granges of the peaks for the new data set

Value

A matrix of the projected patterns in the input data as well as p-values for each element of that matrix.

---

`cgapsPlot` **Plot Individual CoGAPS Patterns**

Description

Function to plot each pattern of the pattern matrix from a cogapsResult and color by cell classifier information to identify which patterns identify which cell classes.

Usage

```r
cgapsPlot(
  cgaps_result,  # CoGAPSResult object from a CoGAPS run or the pattern matrix (matrix must be set equal to TRUE in the latter case)
  sample.classifier,  # factor of sample classifications for all cells for the data to be plotted by (e.g. celltypes)
  cols = NULL,  # vector of colors to be used for the cell classes; should have the same number of colors as levels of the sample.classifier factor. If left null a list of colors is produced
  sort = TRUE,  # TRUE if samples will be sorted according to sample.classifier prior to plotting
  patterns = NULL,  # numerical vector of patterns to be plotted; if null all patterns are plotted
  ...  # other arguments
)
```

Arguments

- `cgaps_result` CoGAPSResult object from a CoGAPS run or the pattern matrix (matrix must be set equal to TRUE in the latter case)
- `sample.classifier` factor of sample classifications for all cells for the data to be plotted by (e.g. celltypes)
- `cols` vector of colors to be used for the cell classes; should have the same number of colors as levels of the sample.classifier factor. If left null a list of colors is produced
- `sort` TRUE if samples will be sorted according to sample.classifier prior to plotting
- `patterns` numerical vector of patterns to be plotted; if null all patterns are plotted
Function to filter a set of scATACseq data by sparsity and return a subset of filtered data, as well as list of the remaining cells and peaks.

**Usage**

```r
dataSubsetBySparsity(
  data, 
  cell_list, 
  peak_list, 
  cell_cut = 0.99, 
  peak_cut = 0.99 
)
```

**Arguments**

- `data`: matrix of read counts peaks x cells
- `cell_list`: list of cell names/identifiers for the data
- `peak_list`: list of peaks from the data
- `cell_cut`: threshold of sparsity to filter at (e.g. 0.99 filters all cells with more than 99 percent zero values)
- `peak_cut`: threshold of sparsity to filter at for peaks

**Value**

nested list containing the subset data, a list of peaks, and list of cells
Examples

data("subsetSchepData")
data("schepPeaks")
data("schepCellTypes")

outData = dataSubsetBySparsity(subsetSchepData, schepCellTypes, schepPeaks)

---

exampleMotifList  
**Example list of motifs for examples**

---

Description

PWMMatrixList used for examples with functions based on DNA motifs. Each entry contains the motif ID and the probability of each nucleotide at each position, as a matrix.

Usage

exampleMotifList

Format

PWMMatrixList of length 100

---

foldAccessibility  
**Estimate fold Accessibility of a Gene Relative to Average**

---

Description

Compares the accessibility of peaks overlapping with a gene, as returned by the geneAccessibility function to the average accessibility of peaks within a given cell population. Meant to provide a rough estimate of how accessible a gene is with values higher than 1 providing evidence of differential accessibility (and thus implying possible transcription), with values lower than 1 indicating the opposite.

Usage

foldAccessibility(peaksAccessibility, cellTypeList, cellType, binaryMatrix)

Arguments

- **peaksAccessibility**
  - the binarized accessibility of a set of peaks; one value returned from the geneAccessibility function
- **cellTypeList**
  - list of celltypes grouping cells in the data
- **cellType**
  - the particular cell type of interest from within cellTypeList
- **binaryMatrix**
  - binarized scATAC data matrix
geneAccessibility

**Value**

Fold accessibility value as compared to average peaks for a given cell type

**Examples**

```r
data("subsetSchepData")
data(schepCellTypes)
library(Homo.sapiens)
geneList <- c("TAL1", "IRF1")
data(schepGranges)
binarizedData <- (subsetSchepData > 0) + 0
accessiblePeaks <- geneAccessibility(geneList = geneList, peakGranges = schepGranges,
atacData = subsetSchepData, genome = Homo.sapiens)
foldAccessibility(peaksAccessibility = accessiblePeaks$TAL1, cellTypeList = schepCellTypes,
cellType = "K562 Erythroleukemia", binaryMatrix = binarizedData)
```

**Description**

The accessibility of a particular set of interest genes is checked by testing overlap of peaks with the genes and gene promoters and then returning the binarized accessibility data for those peaks

**Usage**

```r
geneAccessibility(geneList, peakGranges, atacData, genome)
```

**Arguments**

- `geneList`: vector of HGNC gene symbols to find overlapping peaks for in the data
- `peakGranges`: a GRanges object corresponding to the peaks in the atacData matrix, in the same order as the rows of the atacData matrix
- `atacData`: a single-cell ATAC-seq count matrix peaks by cells
- `genome`: TxDb object to produce gene GRanges from

**Value**

List of matrices corresponding to the accessible peaks overlapping with each gene across all cells in the data
Examples

```r
genePatternMatch
```

Description

Function to take as input CoGAPS results for ATAC-seq data and find genes within the most "pattern-defining" regions (as identified by cut thresholded pattern Marker statistic from the CoGAPS package), as well as the nearest gene and the nearest gene following the region. Note: a TxDb object for the genome of interest must be loaded prior to running this function.

Usage

genePatternMatch(cogapsResult, generanges, genome, scoreThreshold = NULL)

Arguments

cogapsResult: the CogapsResult object produced by a CoGAPS run

generanges: GRanges object corresponding to the genomic regions identified as peaks for the ATAC-seq data that CoGAPS was run on

genome: A TxDb object for the genome of interest, it must be loaded prior to calling this function

scoreThreshold: threshold for the most pattern defining peaks as per the PatternMarker statistic from the CoGAPS package. Default is NULL, returning all PatternMarker peaks. Useful to reduce computational time, as top results are reasonably robust to using more stringent thresholds

Value

double nested list containing lists of the genes in, nearest, and following the peaks matched each pattern

Examples

```r
data("schepCogapsResult")
data(schepGranges)
library(Homo.sapiens)

genes = genePatternMatch(cogapsResult = schepCogapsResult, generanges = schepGranges, genome = Homo.sapiens)```
Description

Use the output from geneAccessibility function to plot a heatmap of the accessible peaks for a particular gene.

Usage

heatmapGeneAccessibility(
  genePeaks, 
  celltypes, 
  colColors = NULL, 
  order = TRUE, 
  ...
)

Arguments

genePeaks The peaks corresponding to a singular gene; one element of the list output by geneAccessibility()
celltypes List or factor of celltypes corresponding to the cells in the scATAC-seq data set the peaks were found in
colColors A vector of colors to color the celltypes by, if NULL a random vector of colors is generated
order should the data be ordered by the celltype classifier? TRUE by default
...
additional arguments to the heatmap.2 function from the gplots package

Value

A plot of the peaks overlapping with a particular gene of interest

Examples

library(Homo.sapiens)
geneList <- c("TAL1", "EGR1")
data(schepGranges)
data("subsetSchepData")
data(schepCellTypes)
accessiblePeaks <- geneAccessibility(geneList = geneList,
  peakGranges = schepGranges, atacData = subsetSchepData, genome = Homo.sapiens)
heatmapGeneAccessibility(genePeaks = accessiblePeaks$EGR1, celltypes = schepCellTypes)
heatmapPatternMarkers

Create Heatmap of PatternMarker Peaks

Description

Function to make a heatmap of the accessibility of the most differentially accessible regions as discovered by CoGAPS.

Usage

heatmapPatternMarkers(
  cgaps_result,
  atac_data,
  celltypes,
  numregions = 50,
  colColors = NULL,
  rowColors = NULL,
  patterns = NULL,
  order = TRUE,
  ...
)

Arguments

cgaps_result  CogapsResult object from CoGAPS run
atac_data     a numeric matrix of the ATAC data input to CoGAPS
celltypes     a list or factor of celltypes corresponding to the positions of those cells in the atac_data matrix
numregions    number of chromosomal regions/peaks to plot for each CoGAPS pattern. Default is 50. Plotting very large numbers of regions can cause significant slowdown in runtime
colColors      column-wise colors for distinguishing celltypes. If NULL, will be generated randomly
rowColors      row-wise colors for distinguishing patterns. If NULL will be generated randomly
patterns       which patterns should be plotted, if NULL all will be plotted
order          option whether to sort the data by celltype before plotting. TRUE by default
...            additional arguments to the heatmap.2 function

Value

heatmap of the accessibility for numregions for each pattern
**heatmapPatternMatrix**

**Note**

If you get the error: "Error in plot.new() : figure margins too large" while using this function in RStudio just make the plotting pane in RStudio larger and run the code again; this error only means the legend is being cut off in any case, the main plot will still appear correctly.

**Examples**

```r
data("schepCogapsResult")
data(schepCellTypes)
data("subsetSchepData")

heatmapPatternMarkers(schepCogapsResult, atac_data = subsetSchepData,
celltypes = schepCellTypes, numregions = 50)
```

**Description**

Selects the patternMatrix (patterns by cells) from the CoGAPSResult and plots the data as a heatmap. Intended to visualize the celltypes distinguished by the patterns found by CoGAPS.

**Usage**

```r
heatmapPatternMatrix(
  cgaps_result, sample.classifier, cellCols = NULL, sort = TRUE, patterns = NULL, matrix = FALSE, rowColors = NULL, ...)
```

**Arguments**

- `cgaps_result` CoGAPSResult object from a CoGAPS run or the pattern matrix (matrix must be set equal to TRUE in the latter case)
- `sample.classifier` factor of sample classifications for all cells for the data to be plotted by (e.g. celltypes)
- `cellCols` vector of colors to be used for the cell classes; should have the same number of colors as levels of the sample.classifier factor. If left null a list of colors is produced
- `sort` TRUE if samples will be sorted according to sample.classifier prior to plotting
motifPatternMatch

Patterns numerical vector of patterns to be plotted; if NULL all patterns are plotted
matrix if false cgaps_result is interpreted as a CoGAPSResult object, if true it is interpreted as the pattern matrix being input directly
rowColors vector of colors to plot along patterns, if NULL generated automatically
... additional arguments to the heatmap.2 function

Value

Heatmap of patternMatrix with color labels for samples

Examples

data("schepCogapsResult")
data(schepCellTypes)
heatmapPatternMatrix(schepCogapsResult, sample.classifier = schepCellTypes)

motifPatternMatch  
Find Motifs and TFs from PatternMarker Peaks

Description

Function that takes CoGAPS result and list of DNA motifs as input and returns motifs which match to the most pattern-defining peaks for each pattern.

Usage

motifPatternMatch(
cogapsResult, generanges, motiflist, genome, scoreThreshold = NULL, motifsPerRegion = 1
)

getTFs(motifList, tfData)

findRegulatoryNetworks(TFs, networks)

getTFDescriptions(TFs)
**Arguments**

- `cogapsResult` the result object from a CoGAPS run
- `generanges` GRanges objects corresponding to the genomic regions which form the rows of the ATAC-seq data that CoGAPS was run on
- `motiflist` a PWMlist of motifs to search the regions for
- `genome` the ucsc genome version to use e.g. "hg19", "mm10"
- `scoreThreshold` threshold for the most pattern defining peaks as per the PatternMarker statistic from the CoGAPS package. By default is NULL, in which case all Pattern defining peaks will be used for motif matching. Used to reduce compute time, as results are quite robust across thresholds
- `motifsPerRegion` number of top motifs to return from each peak
- `motifList` list produced by the motifPatternMatch function
- `tfData` dataframe of motifs and TFs from cisBP database
- `TFs` object of TF info returned from the getTFs function
- `networks` a list of regulatory networks of genes corresponding to TFs; we include human-RegNets and mouseRegNets, downloaded from the TTTrust database (Han et al Nucleic Acid Res. 2018)

**Value**

- `motifPatternMatch`: nested list of the top motif for each region for x number of regions for each pattern
- `getTFs`: list containing list of dataframes of tfData subset to matched TFs and list of how many times each TF was matched to a motif/peak
- `findRegulatoryNetworks`: list of TFs for which we have annotations and the corresponding gene networks for each pattern
- `getTFDescriptions`: list of functional annotations for all TFs in each pattern

**Functions**

- `getTFs`: Match motifs to TFs based on the list of motifs returned by motifPatternMatch
- `findRegulatoryNetworks`: function to match TFs identified by getTFs function to a list of regulatory networks of genes known for those TFs
- `getTFDescriptions`: function to match functional annotation to a list of TFs from the getTFs function

**Examples**

```r
data(exampleMotifList)
data(schepGranges)
data(schepCogapsResult)

motifsByPattern = motifPatternMatch(schepCogapsResult, schepGranges, exampleMotifList, "hg19")```
motifSummarization

Map Peaks to DNA motifs in scATAC-seq Data

Description

Provides functionality to summarize scATAC-seq data by motifs from peak summary. Uses motifmatchr to prepare data for CoGAPS run using motif summarization.

Usage

motifSummarization(
  motifList,  
  scATACData,  
  granges,  
  genome,  
  cellNames,  
  pCutoff = 5e-09
)

Arguments

motifList    PWMatrixList object of motifs (from the TFBS tools package)
scATACData   matrix of scATACseq data, peaks (rows) by cells (columns)
pathwayMatch

DESCRIPTION

Takes the result of the genePatternMatch function and finds significantly enriched pathways for each pattern.

USAGE

pathwayMatch(gene_list, pathways, p_threshold = 0.05, pAdjustMethod = "BH")

ARGUMENTS

gene_list  Result from the genePatternMatch function, a list of genes for each pattern
pathways  List of pathways to perform gene enrichment on. Recommended to download using msigdb (see examples)
p_threshold  significance level to use in enrichment analysis
pAdjustMethod  multiple testing correction method to apply using the p.adjust options (e.g. "BH")

VALUE

List of gene overlap objects, pathways with significant overlap and pathway names for each pattern.
Examples

data(schepCogapsResult)
data(schepGranges)
library(Homo.sapiens)

genes <- genePatternMatch(cogapsResult = schepCogapsResult,
    generanges = schepGranges, genome = Homo.sapiens)

library(dplyr)
pathways = msigdbr::msigdb(species = "Homo sapiens", category ="H") %>%
dplyr::select(gs_name, gene_symbol) %>% as.data.frame()

matchedPathways = pathwayMatch(genes, pathways, p_threshold = 0.001)

---

patternMarkerCellClassifier

*Match cells to patterns*

Description

Use the patternMarker statistic to determine which cells belong to each pattern in the data

Usage

`patternMarkerCellClassifier(cgapsResult)`

Arguments

cgapsResult a CoGAPSResult object

Value

list containing a prediction matrix and vector classifying cells to patterns

Examples

data("schepCogapsResult")
pClass <- patternMarkerCellClassifier(schepCogapsResult)
peaksToGRanges

List of peaks to GRanges

Description

Wrapper function for makeGrangesFromDataFrame() from the GenomicRanges package to build GRanges objects from character list of chromosomal regions because this is a common format to receive peak information.

Usage

peaksToGRanges(region_list, sep)

Arguments

region_list

character list or vector of chromosomal regions/peaks in form chromosome-umber(sep)start(sep)end eg. Chr1-345678-398744

sep

separator between information pieces of string (conventionally ".-" or ".")

Value

GRanges corresponding to input list of region information

Note

If region_list is a dataframe you should use the GenomicRanges function makeGRangesFromDataFrame which this function applies

Examples

data(schepPeaks)

schepGranges = peaksToGRanges(schepPeaks, sep = "-")

RNAseqTFValidation

Validate TF Findings with RNA-seq CoGAPS

Description

Use results from CoGAPS run on matched RNA-seq data to verify TF activity suggested by motif matching analysis of ATAC CoGAPS output. Uses the fgsea package to find enrichment of Pattern-Marker genes among genes regulated by identified candidate TFs.
schepCellTypes

Usage

```r
RNAseqTFValidation(
  TFGenes,
  RNACoGAPSResult,
  ATACPatternSet,
  RNAPatternSet,
  matrix = FALSE
)
```

Arguments

- **TFGenes** genes regulated by the TFs as returned by `simpleMotifTFMatch()` or `findRegulatoryNetworks()`.
- **RNACoGAPSResult** CoGAPSResult object from matched RNA-seq data, or, if matrix = TRUE, a matrix containing patternMarker gene ranks. Must contain gene names.
- **ATACPatternSet** vector of patterns found by CoGAPS in the ATAC data to match against patterns found in RNA.
- **RNAPatternSet** vector of patterns found by CoGAPS in RNA to match against those found in ATAC.
- **matrix** TRUE if inputting matrix of PatternMarker genes, FALSE if inputting CoGAPS result object. FALSE by default.

Value

Result matrices from the `fgsea` function for each pattern comparison.

Examples

```r
## Not run:
gseaList = RNAseqTFValidation(TFMatchResult$RegulatoryNetworks, RNACoGAPS, c(1,3), c(2,7), matrix = FALSE)
## End(Not run)
```

schepCellTypes

Cell types corresponding to subsetSchepData

Description


Usage

```r
schepCellTypes
```
**schepCogapsResult**

**Format**
Factor of length 600 with 12 levels

**Source**
10.1038/nmeth.4401

---

**schepCogapsResult**  
*CogapsResult from the subsetSchepData object*

**Description**
Output from applying the CoGAPS algorithm to the subsetSchepData object.

**Usage**
schepCogapsResult

---

**schepGranges**  
*GRanges corresponding to subsetSchepData*

**Description**
GRanges in the order of the peaks of the subsetSchepData object from the Schep et al, 2017, Nature Methods paper.

**Usage**
schepGranges

---

**Format**
GRanges of length 5036

**Source**
10.1038/nmeth.4401
simpleMotifTFMatch

- **schepPeaks**
  - *Peaks corresponding to subsetSchepData*

**Description**

Character vector of peaks in the order of the peaks of the subsetSchepData object from the Schep et al, 2017, Nature Methods paper.

**Usage**

schepPeaks

**Format**

Character vector of length 5036

**Source**

[10.1038/nmeth.4401](https://10.1038/nmeth.4401)

---

**simpleMotifTFMatch**

*Motif/TF Matching in a Single Function*

**Description**

If the user does not have a specific set of motifs, transcription factors, or regulatory networks that they want to match against, simply uses the core motifs from the JASPAR database to find motifs and TFs in the most Pattern differentiating peaks, as well as regulatory networks from TTrust database corresponding to the identified TFs. This is used to provide transcription factors with functional annotation which may suggest plausible unknown regulatory mechanisms operating in the cell types of interest within the data.

**Usage**

```r
simpleMotifTFMatch(
  cogapsResult, 
  generanges, 
  organism, 
  genome, 
  scoreThreshold = NULL, 
  motifsPerRegion = 1
)
```
Arguments

cogapsResult  result object from CoGAPS run
generanges  GRanges object corresponding to peaks in ATACseq data CoGAPS was run on
organism  organism name (e.g. "Homo sapiens")
genome  genome version to use (e.g. hg19, mm10)
scoreThreshold  threshold for the most pattern defining peaks as per the PatternMarker statistic from the CoGAPS package. By default is NULL, in which case all Pattern defining peaks will be used for motif matching. Used to reduce compute time, as results are quite robust across thresholds
motifsPerRegion  number of motifs to attempt to find within each peak

Value

list containing list of matched motifs, list of transcription factors, regulatory gene networks known for those TFs, functional annotations, summary showing how many times each TF was matched to a peak, and the downloaded set of motifs for the user to save for reproducibility

Examples

data("schepCogapsResult")
data(schepGranges)

motifResults = simpleMotifTFMatch(cogapsResult = schepCogapsResult,
generanges = schepGranges, organism = "Homo sapiens",
genome = "hg19", motifsPerRegion = 1)


Description

Subset from the Schep et al data, used for examples in this package.

Usage

subsetSchepData

Format

A matrix with 5036 peaks and 600 cells in the order of the schepPeaks, schepCellTypes, and schepGranges data objects.

Source

10.1038/nmeth.4401
tfData  

**List of human TFs and motifs from cisBP database**

### Description

Information on human TFs and their corresponding DNA motifs

### Usage

```
tfData
```

### Format

Data frame with 95,413 rows and 28 columns.

### Source

[http://cisbp.ccbr.utoronto.ca/](http://cisbp.ccbr.utoronto.ca/)
## Index

### *datasets*

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>exampleMotifList</td>
<td>6</td>
</tr>
<tr>
<td>schepCellTypes</td>
<td>18</td>
</tr>
<tr>
<td>schepCogapsResult</td>
<td>19</td>
</tr>
<tr>
<td>schepGranges</td>
<td>19</td>
</tr>
<tr>
<td>schepPeaks</td>
<td>20</td>
</tr>
<tr>
<td>subsetSchepData</td>
<td>21</td>
</tr>
<tr>
<td>tfData</td>
<td>22</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>applyGREAT</td>
<td>2</td>
</tr>
<tr>
<td>ATACTransferLearning</td>
<td>3</td>
</tr>
<tr>
<td>cgapsPlot</td>
<td>4</td>
</tr>
<tr>
<td>dataSubsetBySparsity</td>
<td>5</td>
</tr>
<tr>
<td>exampleMotifList</td>
<td>6</td>
</tr>
</tbody>
</table>

### findRegulatoryNetworks

- (motifPatternMatch), 12

### foldAccessibility

- 6

### geneAccessibility

- 7

### genePatternMatch

- 8

### getTFDescriptions

- (motifPatternMatch), 12

### getTFs

- (motifPatternMatch), 12

### heatmapGeneAccessibility

- 9

### heatmapPatternMarkers

- 10

### heatmapPatternMatrix

- 11

### motifPatternMatch

- 12

### motifSummarization

- 14

### pathwayMatch

- 15

### patternMarkerCellClassifier

- 16

### peaksToGRanges

- 17

### RNAseqTFValidation

- 17

### schepCellTypes

- 18

### schepCogapsResult

- 19

### schepGranges

- 19

### schepPeaks

- 20

### simpleMotifTFMatch

- 20

### subsetSchepData

- 21

### tfData

- 22