Package: SPICEY (via r-universe)

October 31, 2026

Title Calculates cell type specificity from single cell data

Version 1.3.0

Description SPICEY (SPecificity Index for Coding and Epigenetic activitY) is an R package designed to quantify cell-type specificity in single-cell transcriptomic and epigenomic data, particularly scRNA-seq and scATAC-seq. It introduces two complementary indices: the Gene Expression Tissue Specificity Index (GETSI) and the Regulatory Element Tissue Specificity Index (RETSI), both based on entropy to provide continuous, interpretable measures of specificity. By integrating gene expression and chromatin accessibility, SPICEY enables standardized analysis of cell-type-specific regulatory programs across diverse tissues and conditions.

License Artistic-2.0

Encoding UTF-8

Depends R (>= 4.5.0), utils, stats, grDevices

Imports GenomicRanges, GenomicFeatures, AnnotationDbi, S4Vectors, ggplot2, dplyr, tidyr, tibble, GenomeInfoDb, scales, cowplot

Suggests BiocStyle, knitr, rmarkdown, TxDb.Hsapiens.UCSC.hg38.knownGene, org.Hs.eg.db, testthat (>= 3.0.0)

VignetteBuilder knitr

biocViews Transcriptomics, Epigenetics, SingleCell, DifferentialExpression, DifferentialPeakCalling, GeneRegulation, GeneTarget, GeneExpression, Transcription

RoxygenNote 7.3.2

Roxygen list(markdown = TRUE)

URL https://georginafp.github.io/SPICEY

BugReports https://github.com/georginafp/SPICEY/issues

LazyData false

Config/testthat/edition 3

Config/pak/sysreqs
make libbz2-dev libicu-dev liblzma-dev libpng-dev libxml2-dev libssl-dev xz-utils zlib1g-dev

Repository https://bioc.r-universe.dev

Date/Publication 2026-04-28 13:05:27 UTC

RemoteUrl https://github.com/bioc/SPICEY

RemoteRef HEAD

RemoteSha 8c65e8a22df91698ed97b65989d4a3ea85b59a80

Contents

.add_tss_annotation . . . 2
.parse_input_diff . . . 3
.standardize_peaks . . . 4
annotate_with_coaccessibility . . . 5
annotate_with_nearest . . . 6
atac . . . 8
cicero_links . . . 8
compute_spicey_index . . . 9
entropy_index . . . 10
extract_gene_peak_annotations . . . 11
get_promoters . . . 12
link_spicey . . . 12
plot_heatmap . . . 14
prepare_heatmap_data . . . 14
rna . . . 15
specificity_index . . . 16
SPICEY . . . 16
spicey_heatmap . . . 19

Index **21**

`.add_tss_annotation` *Add TSS annotation to peaks*

Description

Identifies transcription start sites (TSS) overlapping with input peaks and adds logical and gene ID columns to an annotation table.

Usage

```
.add_tss_annotation(
  annotation,
  peaks,
  txdb,
  annot_dbi,
  protein_coding_only,
  verbose
)
```

Arguments

annotation	A data.frame or GRanges containing peak annotations, including a region_id column.
peaks	A GRanges object containing the original peak set.
txdb	TxDb object for genome annotation (required if annotation requested).
annot_dbi	AnnotationDbi object for gene ID mapping (required if annotation requested).
protein_coding_only	Logical; restrict to protein-coding genes (default TRUE).
verbose	Logical; print messages (default TRUE).

Value

A data.frame or GRanges (depending on input) with added columns:

in_TSS Logical, TRUE if the peak overlaps a TSS.

TSS_gene Gene symbol of the overlapping TSS, if any.

.parse_input_diff *Parses input data of various types (e.g., named lists of* GRanges *or* data.frame*, or a* GRangesList*) into a single tidy* data.frame*, with a* cell_type *column.*

Description

Parses input data of various types (e.g., named lists of GRanges or data.frame, or a GRangesList) into a single tidy data.frame, with a cell_type column.

Usage

```
.parse_input_diff(input)
```

Arguments

input	An object representing differential results, such as: • A named list of GRanges objects. • A named list of data.frames. • A GRangesList.

Value

A data.frame combining all elements, with an added cell_type column indicating the source.

.standardize_peaks *Standardizes peak input, ensuring that input peaks is a* GRanges *object, removes alternate scaffolds (*_alt, random, fix, Un*), and assigns region IDs as names.*

Description

Standardizes peak input, ensuring that input peaks is a GRanges object, removes alternate scaffolds (_alt, random, fix, Un), and assigns region IDs as names.

Usage

```
.standardize_peaks(peaks)
```

Arguments

peaks	A data.frame with genomic coordinates and a region_id column, or a GRanges object with a region_id column.

Value

A GRanges object with:

seqnames, start, end Coordinates of the input regions.

region_id Unique identifier of the region (e.g., chr1-5000-5800).

metadata columns Any additional columns present in the input.

annotate_with_coaccessibility *Annotate peaks with co-accessible genes using Cicero links*

Description

Links peaks to genes based on Cicero co-accessibility with promoters or TSSs.

Usage

```
annotate_with_coaccessibility(
  peaks,
  txdb,
  links_df,
  annot_dbi,
  protein_coding_only = TRUE,
  verbose = TRUE,
  add_tss_annotation = FALSE,
  upstream,
  downstream
)
```

Arguments

peaks	A GRanges or data.frame of peaks with at least the following columns: **seqnames** Chromosome name of the regulatory region (e.g., "chr1"). Only for data.frames. **start** Start coordinate of the peak. Only for data.frames. **end** End coordinate of the peak. Only for data.frames. **region_id** Unique identifier of the region (e.g., chr1-5000-5800)
txdb	TxDb object for genome annotation (required if annotation requested).
links_df	A data.frame with Cicero links. Must contain columns: Peak1, Peak2, and coaccess.
annot_dbi	AnnotationDbi object for gene ID mapping (required if annotation requested).
protein_coding_only	Logical; restrict to protein-coding genes (default TRUE).
verbose	Logical; print messages (default TRUE).
add_tss_annotation	Logical; annotate regulatory elements overlapping TSS (default FALSE). If TRUE, use +/- 1bp TSS.
upstream	Single integer value indicating the number of bases upstream from the TSS (transcription start sites) (default "2000kb").
downstream	Single integer values indicating the number of bases downstream from the TSS (transcription start sites) (default "2000kb").

Value

A data.frame with the original metadata columns from peaks, along with an added gene_id column containing the symbol of the co-accessible gene. Peaks with no gene annotation will have NA in the gene_id field.

Examples

```
library(TxDb.Hsapiens.UCSC.hg38.knownGene)
library(org.Hs.eg.db)

data(atac)
data(cicero_links)

peaks <- unique(unlist(atac)[, c("region_id")])
annotation_coacc <- annotate_with_coaccessibility(
  peaks = peaks,
  txdb = TxDb.Hsapiens.UCSC.hg38.knownGene,
  links_df = cicero_links,
  annot_dbi = org.Hs.eg.db,
  protein_coding_only = TRUE,
  verbose = TRUE,
  add_tss_annotation = FALSE,
  upstream = 2000,
  downstream = 2000
)
```

---

annotate_with_nearest *Annotates regulatory elements (e.g., ATAC-seq peaks) to the nearest gene*

Description

based on distance to the transcription start site (TSS), using a TxDb reference and gene annotations from org.*.db packages.

Usage

```
annotate_with_nearest(
  peaks,
  txdb,
  annot_dbi,
  protein_coding_only = TRUE,
  verbose = TRUE,
  add_tss_annotation = FALSE,
  upstream,
  downstream
)
```

Arguments

peaks	A GRanges or data.frame of peaks with at least the following columns: **seqnames** Chromosome name of the regulatory region (e.g., "chr1"). Only for data.frames. **start** Start coordinate of the peak. Only for data.frames. **end** End coordinate of the peak. Only for data.frames. **region_id** Unique identifier of the region (e.g., chr1-5000-5800)
txdb	TxDb object for genome annotation (required if annotation requested).
annot_dbi	AnnotationDbi object for gene ID mapping (required if annotation requested).
protein_coding_only	Logical; restrict to protein-coding genes (default TRUE).
verbose	Logical; print messages (default TRUE).
add_tss_annotation	Logical; annotate regulatory elements overlapping TSS (default FALSE). If TRUE, use +/- 1bp TSS.
upstream	Single integer value indicating the number of bases upstream from the TSS (transcription start sites) (default "2000kb").
downstream	Single integer values indicating the number of bases downstream from the TSS (transcription start sites) (default "2000kb").

Value

A data.frame of peaks annotated to its nearest gene, with columns:

distanceToTSS	Distance to the nearest TSS
gene_id	Identifier of the gene. Must be an official gene symbol (e.g., GAPDH)
annotation	"Promoter" or "Distal" based on distance

Examples

```
library(dplyr)
library(TxDb.Hsapiens.UCSC.hg38.knownGene)
library(org.Hs.eg.db)

data(atac)
peaks <- unique(unlist(atac)[, c("region_id")])

annotation_near <- annotate_with_nearest(
  peaks = peaks,
  txdb = TxDb.Hsapiens.UCSC.hg38.knownGene,
  annot_dbi = org.Hs.eg.db,
  protein_coding_only = TRUE,
  verbose = TRUE,
  add_tss_annotation = FALSE,
  upstream = 2000,
  downstream = 2000
)
```

atac *Example single-cell ATAC-seq differential accessibility data*

Description

A toy example dataset representing single-cell ATAC-seq differential accessibility results. Each row corresponds to a chromatin accessibility peak tested for differential accessibility across one or more cell types. The dataset is formatted as a list of `GRanges` objects or data frames (convertible to `GRanges`), where each element represents differential accessibility statistics for a specific cell type. Multiple rows may exist per peak, each representing results in a different cell type.

Usage

```
data(atac)
```

Format

A data frame or `GRanges`-like object with the following required columns:

region_id Unique identifier of the region (e.g., `chr1-5000-5800`).

avg_log2FC Average log2 fold-change of accessibility for the peak in the specific cell type

p_val_adj Adjusted p-value (e.g., `FDR-corrected`)

cell_type Cell type or cluster label associated with each measurement (e.g., `Acinar`)

Source

Precomputed using `FindMarkers()` (Wilcoxon test, via Presto if available) on control samples from the Human Pancreas Analysis Program (HPAP), using paired snATAC-seq and snRNA-seq data from three non-diabetic human donors.

cicero_links *Example Cicero co-accessibility links*

Description

A toy example dataset of co-accessibility links inferred from single-cell ATAC-seq data using tools such as Cicero or Signac's LinkPeaks(). These links support integrative analysis by associating regulatory elements with putative target genes. The peaks referenced here must exactly match those in the ATAC-seq differential accessibility dataset.

Usage

```
data(cicero_links)
```

Format

A data frame with the following columns:

Peak1 Genomic coordinate or peak identifier for the first peak in the pair (e.g., `chr1-110209621-110211746`)

Peak2 Genomic coordinate or peak identifier for the second peak in the pair (e.g., `chr1-110209621-110211746`)

coaccess Co-accessibility score or correlation value quantifying the linkage

Source

Cicero co-accessibility links were computed from UMAP-reduced snATAC-seq data (HPAP, control donors) using `run_cicero()` with chromosome sizes from hg38. Input data matched the peaks in the provided ATAC dataset.

`compute_spicey_index` *Compute cell type specificity scores from single-cell RNA and/or ATAC data*

Description

Computes:

- GETSI (Gene Expression Tissue Specificity Index) from single-cell RNA-seq differential expression data.
- RETSI (Regulatory Element Tissue Specificity Index) from single-cell ATAC-seq differential accessibility data.

Either RNA or ATAC input must be provided.

Usage

```
compute_spicey_index(diff = NULL, id = NULL)
```

Arguments

`diff` Single `data.frame` with differential results, with required columns:

id Identifier for either genes or regions. Gene IDs must be official gene symbols, while region IDs should follow the usual genomic coordinate format such as `chr-start-end` or `chr:start-end`. The name of this column must match the `id` argument.

avg_log2FC Average log2 fold-change for the gene in that cell type.

p_val_adj Adjusted p-value (e.g., `FDR-corrected`)

cell_type Cell type or cluster label (e.g., `Acinar`)

`id` A character string specifying the name of the column in `diff` that contains the unique identifiers for features (e.g., genes or regions). Must match a column name in `diff`, such as "gene_id" or "region_id".

Value

A data frame with specificity scores for each feature:

score (GETSI or RETSI) Specificity score (weighted log2FC).

norm_entropy Shannon entropy-based specificity measure.

`entropy_index` *Calculate normalized Shannon-entropy of specificity scores*

Description

Computes the normalized entropy of specificity scores (RETSI or GETSI) across cell types. Entropy quantifies how evenly a feature's activity is distributed among cell types, and if normalized, yields scores from 0 to 1, where values close to 1 indicate widespread distribution across cell types, and values near 0 denote dominating distribution towards one cell type.

Usage

```
entropy_index(spec_df, group_col)
```

Arguments

`spec_df`	A data.frame containing the computed specificity scores containing at least the following columns: **cell_type** Cell type or cluster label. **score** Specificity score for each feature in each cell type. `[group_col]` Column containing the feature identifier (e.g., gene_id or region) The **name of this column must match the value passed to the** `group_col` **argument**
`group_col`	A string specifying the name of the column in `da` that identifies each feature, such as `gene_id` for genes or `region` for ATAC peaks.

Value

A data.frame with one row per feature, containing:

group_col Feature identifier.

entropy Raw Shannon entropy computed from specificity scores.

norm_entropy Normalized Shannon entropy score (1 - exp(-entropy)) bounded between 0 and 1, where lower values indicate higher specificity.

`extract_gene_peak_annotations` *Overlap peaks with gene promoters to obtain gene annotations*

Description

Identifies overlaps between a set of peaks and promoter regions, optionally restricted to protein-coding genes.

Usage

```
extract_gene_peak_annotations(
  peaks,
  txdb,
  annot_dbi,
  protein_coding_only = TRUE,
  upstream,
  downstream,
  verbose = FALSE
)
```

Arguments

`peaks`	A `GRanges` or `data.frame` of peaks with at least the following columns: **seqnames** Chromosome name of the regulatory region (e.g., `"chr1"`). Only for data.frames. **start** Start coordinate of the peak. Only for data.frames. **end** End coordinate of the peak. Only for data.frames. **region_id** Unique identifier of the region (e.g., `chr1-5000-5800`)
`txdb`	`TxDb` object for genome annotation (required if annotation requested).
`annot_dbi`	`AnnotationDbi` object for gene ID mapping (required if annotation requested).
`protein_coding_only`	Logical; restrict to protein-coding genes (default TRUE).
`upstream`	Single integer value indicating the number of bases upstream from the TSS (transcription start sites) (default `"2000kb"`).
`downstream`	Single integer values indicating the number of bases downstream from the TSS (transcription start sites) (default `"2000kb"`).
`verbose`	Logical; print messages (default TRUE).

Value

A `GRanges with` with:

seqnames, start, end Coordinates of the peak that overlaps with a gene promoter.

region_id Unique identifier of the region (e.g., `chr1-5000-5800`).

gene_id Identifier of the gene. Must be an official gene symbol (e.g., `GAPDH`)

get_promoters — *Extract promoter regions annotated gene symbols from a TxDb and AnnotationDbi object*

Description

Extract promoter regions annotated gene symbols from a TxDb and AnnotationDbi object

Usage

```
get_promoters(
  txdb,
  annot_dbi,
  upstream,
  downstream,
  protein_coding_only = TRUE
)
```

Arguments

txdb	TxDb object for genome annotation (required if annotation requested).
annot_dbi	AnnotationDbi object for gene ID mapping (required if annotation requested).
upstream	Single integer value indicating the number of bases upstream from the TSS (transcription start sites) (default "2000kb").
downstream	Single integer values indicating the number of bases downstream from the TSS (transcription start sites) (default "2000kb").
protein_coding_only	Logical; restrict to protein-coding genes (default TRUE).

Value

A GRanges object with the chromosomes, start and end positions of defined specie promoter regions together with the official gene symbol stored in the gene_id metadata column.

link_spicey — *Link RETSI regions to GETSI scores using gene-based association methods*

Description

This function connects regulatory regions scored with RETSI

Usage

```
link_spicey(retsi = NULL, getsi = NULL, annotation = NULL)
```

Arguments

retsi	A data.frame containing RETSI scores for chromatin accessibility regions, as returned by `compute_spicey_index()` using single-cell ATAC-seq differential accessibility data. Must include at least the following columns: **region_id** Unique identifier of the region (e.g., `chr1-5000-5800`). **cell_type** Cell type or cluster label (e.g., `Acinar`) **RETSI** RETSI value: cell-type specificity score **norm_entropy** Normalized Shannon entropy of RETSI
getsi	A data.frame containing GETSI scores for genes, as returned by `compute_spicey_index()` using single-cell RNA-seq differential expression data. Must include at least the following columns: **gene_id** Identifier of the gene. Must be an official gene symbol (e.g., `GAPDH`). **cell_type** Cell type or cluster label (e.g., `Acinar`) **GETSI** GETSI value: cell-type specificity score **norm_entropy** Normalized Shannon entropy of GETSI
annotation	(Optional). A data.frame linking `gene_id` to `region_id`. They should have the same names provided in the respective parameters. This can be provided by the user or generated using the functions: `annotate_with_nearest` or `annotate_with_coaccessibility`. It should contain at least the following columns: **region_id** Unique identifier of the region (e.g., `chr1-5000-5800`). The name of this column should match the `region_id` argument. **cell_type** Cell type or cluster label. (e.g., `Acinar`) **gene_id** Identifier of the gene. Must be an official gene symbol (e.g., `GAPDH`). The name of this column should match the `gene_id` argument.

Value

A `data.frame` where each row represents a regulatory element–gene pair linked within a given cell type. The output includes:

region_id Unique identifier of the region (e.g., `chr1-5000-5800`)

gene_id Identifier of the gene. Must be an official gene symbol (e.g., `GAPDH`).

cell_type Cell type or cluster in which the association is observed (e.g., `Acinar`)

RETSI RETSI score: regulatory element specificity in this cell type.

RETSI_entropy Normalized shannon-entropy of RETSI (lower = more specific).

GETSI GETSI score: gene expression specificity in this cell type.

GETSI_entropy Normalized shannon-entropy of GETSI (lower = more specific).

... Any additional columns from the original `retsi` and `getsi` inputs, suffixed with `_ATAC` and `_RNA` respectively (e.g., `avg_log2FC_ATAC`, `p_val_RNA`).

plot_heatmap *Plot a SPICEY score gene-by-cell-type heatmap*

Description

Generates a heatmap using ggplot2 to visualize expression or accessibility SPICEY scores for genes across different cell types. Genes are ordered by their highest-scoring cell type, and then by maximum SPICEY score within that group.

Usage

```
plot_heatmap(df_z, title_text, fill_label)
```

Arguments

df_z	A data frame with SPICEY scored values. Must contain: **gene_id** Identifier of the gene. Must be an official gene symbol (e.g., GAPDH). cell_type Cell type or cluster label (e.g., Acinar) score Numeric. Values of the specificity score (e.g., RETSI, GETSI)
title_text	Character. Title of the heatmap.
fill_label	Character. Legend label for the color scale.

Value

A ggplot2 object representing the heatmap.

See Also

prepare_heatmap_data, spicey_heatmap

prepare_heatmap_data *Prepare data for SPICEY heatmap*

Description

Filters and processes a gene–cell-type matrix for heatmap visualization. Selects the top n genes per cell type, and returns a summary matrix suitable for plotting.

Usage

```
prepare_heatmap_data(df, score_col, top_n)
```

Arguments

df	A data frame with at least: gene_id, cell_type, and a score column.
score_col	Character. Name of the score column to rank.
top_n	Integer. Number of top-ranked genes per cell type.

Value

A data frame with: gene_id, cell_type, and score.

See Also

plot_heatmap, spicey_heatmap

---

rna *Example single-cell RNA-seq differential expression data*

Description

A toy example dataset representing single-cell RNA-seq differential expression results. Each row corresponds to a gene tested across one or more cell types. The dataset is formatted as a list of GRanges objects or data frames (convertible to GRanges), where each element contains differential expression statistics for a specific cell type. Multiple rows may exist per gene, each representing results in a different cell type.

Usage

```
data(rna)
```

Format

A data frame or GRanges-like object with the following required columns:

gene_id Identifier of the gene. Must be an official gene symbol (e.g., GAPDH)

avg_log2FC Average log2 fold-change of expression for the gene in the specific cell type

p_val_adj Adjusted p-value (e.g., FDR-corrected)

cell_type Cell type or cluster label associated with each measurement (e.g., Acinar)

Source

Precomputed using FindMarkers() (Wilcoxon test, via Presto if available) on control samples from the Human Pancreas Analysis Program (HPAP), using paired snATAC-seq and scRNA-seq data from three non-diabetic human donors.

`specificity_index` *Calculate specificity scores for grouped features*

Description

This function computes a specificity index for different features (e.g., genes or regions) based on differential expression/accessibility data. It rescales fold-change values and weights them by significance to quantify how specific a feature's activity is to a particular cell type.

Usage

```
specificity_index(da, group_col)
```

Arguments

`da`	A data.frame containing differential results with at least the following columns: **avg_log2FC** Average log2 fold-change of the feature (gene or region). **cell_type** Cell type or cluster label. (e.g., `Acinar`) `[group_col]` Column containing the feature identifier (e.g., gene_id or region) The **name of this column must match the value passed to the** `group_col` **argument**
`group_col`	A string specifying the name of the column in `da` that identifies each feature, such as `gene_id` for genes or `region` for ATAC peaks.

Value

A data.frame identical to the input but with additional columns:

avg_FC Fold-change converted from log2 scale.

max_FC Maximum fold-change observed within each feature group.

weight Normalized significance weight derived from adjusted p-values.

norm_FC Fold-change normalized by maximum fold-change in the group.

score Specificity score computed as the product of normalized fold-change significantly weighted.

`SPICEY` *SPICEY: Tissue specificity analysis for single-cell data The SPICEY package provides a user-friendly pipeline for quantifying and visualizing tissue specificity specificity from single-cell ATAC-seq and/or single cell RNA-seq datasets, typically processed with tools such as Seurat or Signac. The core outputs of SPICEY are two tissue specific metrics, combined with entropy-based measures.*

Description

Computes tissue-specificity scores from differential accessibility (RETSI) and/or gene expression (GETSI) data obtained from single cell experiments. Supports:

- RETSI calculation from differential accessibility data in different cell types/clusters (scATAC-seq).
- GETSI calculation from differential expression data in different cell types/clusters (scRNA-seq).
- Optional integration of RETSI and GETSI scores by linking gene associations (see `annotate_with_nearest` or `annotate_with_coaccessibility`)

Usage

```
SPICEY(atac = NULL, rna = NULL, annotation = NULL, verbose = TRUE)
```

Arguments

`atac`	Either a single `data.frame` or a named list of `data.frames` or `GRanges` where each element corresponds to a cell type. It should contain differential chromatin accessibility results with required columns: **region_id** Unique identifier of the region (e.g., `chr1-5000-5800`). **avg_log2FC** Average log2 fold-change for accessibility in that cell type. **p_val_adj** Adjusted p-value (e.g., FDR-corrected). **cell_type** Cell type or cluster label. Only necessary when input is a single `data.frame`. If input is a list, it will be generated from list names. Note that the same region may appear multiple times across cell types.
`rna`	Either a single `data.frame` or a named list of `data.frames` or `GRanges` where each element corresponds to a cell type. It should contain differential expression results, with required columns: **gene_id** Identifier of the gene. Must be an official gene symbol (e.g., `GAPDH`). The name of this column should match the `gene_id` argument. **avg_log2FC** Average log2 fold-change for the gene in that cell type. **p_val_adj** Adjusted p-value (e.g., FDR-corrected). **cell_type** Cell type or cluster label. Only necessary when input is a single `data.frame`. If input is a list, it will be generated from list names Note that the same gene may appear multiple times across cell types.
`annotation`	(Optional). A data.frame linking `gene_id` to `region_id`. They should have the same names provided in the respective parameters. This can be provided by the user or generated using the functions: `annotate_with_nearest` or `annotate_with_coaccessibility`. It should contain at least the following columns: **region_id** Unique identifier of the region (e.g., `chr1-5000-5800`). The name of this column should match the `region_id` argument. **cell_type** Cell type or cluster label. (e.g., `Acinar`) **gene_id** Identifier of the gene. Must be an official gene symbol (e.g., `GAPDH`). The name of this column should match the `gene_id` argument.
`verbose`	Logical; print messages (default TRUE).

Value

Depending on inputs, returns RETSI and/or GETSI data frames, optionally linked and annotated.

Examples

```
data(rna)
data(atac)

# Calculate RETSI only
retsi <- SPICEY(atac = atac)

# Calculate GETSI only
getsi <- SPICEY(rna = rna)

# Calculate both
both <- SPICEY(
  rna = rna,
  atac = atac
)

# Integrate RETSI and GETSI with nearest gene
library(TxDb.Hsapiens.UCSC.hg38.knownGene)
library(org.Hs.eg.db)
peaks <- unique(unlist(atac)[, c("region_id")])

annotation_near <- annotate_with_nearest(
  peaks = peaks,
  txdb = TxDb.Hsapiens.UCSC.hg38.knownGene,
  annot_dbi = org.Hs.eg.db,
  protein_coding_only = TRUE,
  verbose = TRUE,
  add_tss_annotation = FALSE,
  upstream = 2000,
  downstream = 2000
)
spicey_near <- SPICEY(
  rna = rna,
  atac = atac,
  annotation = annotation_near
)

# Integrate RETSI and GETSI with coaccessibility
data(cicero_links)

annotation_coacc <- annotate_with_coaccessibility(
  peaks = peaks,
  txdb = TxDb.Hsapiens.UCSC.hg38.knownGene,
  links_df = cicero_links,
  annot_dbi = org.Hs.eg.db,
  protein_coding_only = TRUE,
  verbose = TRUE,
  add_tss_annotation = FALSE,
```

```
  upstream = 2000,
  downstream = 2000
)
spicey_coacc <- SPICEY(
  rna = rna,
  atac = atac,
  annotation = annotation_coacc
)
```

`spicey_heatmap` *SPICEY heatmap for gene specificity across cell types*

Description

Visualizes gene-level specificity scores (RETSI and/or GETSI) across cell types using a SPICEY scored heatmap representation. Depending on the chosen mode, the function can display either RETSI or GETSI scores independently, or compute and visualize a combined SPICEY score (mean score of RETSI and GETSI). If `spicey_measure = "SPICEY"` and `combined_score = TRUE`, RETSI and GETSI scores are scaled, averaged, and shown in a unified heatmap. Otherwise, separate heatmaps are produced for RETSI and GETSI, respectively.

Usage

```
spicey_heatmap(
  df,
  top_n = 5,
  spicey_measure = c("RETSI", "GETSI", "SPICEY"),
  combined_score = FALSE
)
```

Arguments

`df`	A data frame with at least the following columns: **gene_id** Identifier of the gene. Must be an official gene symbol (e.g., `GAPDH`). If you only have ATAC data, link to nearest gene (`annotate_with_nearest`) or using coaccessibility (`annotate_with_coaccessibility`). `cell_type` Cell type or cluster label (e.g., `Acinar`) `RETSI` Numeric. RETSI specificity scores (optional unless used). `GETSI` Numeric. GETSI specificity scores (optional unless used).
`top_n`	Integer. Number of top-ranked genes to include per cell type (default `"5"`)
`spicey_measure`	Character. Score type to visualize. Must be one of the following: `"RETSI"` Only RETSI will be plotted. `"GETSI"` Only GETSI will be plotted. `"SPICEY"` Both RETSI and GETSI are used (requires both columns)
`combined_score`	Logical. Only relevant if `spicey_measure = "SPICEY"`. If `TRUE`, a single heatmap of mean RETSI/GETSI score is generated. If `FALSE`, two heatmaps are produced side by side (RETSI and GETSI).

Value

A ggplot2 object, or a patchwork layout if two heatmaps are returned.

See Also

SPICEY, prepare_heatmap_data, plot_heatmap

Examples

```
library(TxDb.Hsapiens.UCSC.hg38.knownGene)
library(org.Hs.eg.db)

data(rna)
data(atac)
data(cicero_links)

# Obtain annotatin with coaccessibility
peaks <- unique(unlist(atac)[, c("region_id")])
annotation_coacc <- annotate_with_coaccessibility(
  peaks = peaks,
  txdb = TxDb.Hsapiens.UCSC.hg38.knownGene,
  links_df = cicero_links,
  annot_dbi = org.Hs.eg.db,
  protein_coding_only = TRUE,
  verbose = TRUE,
  add_tss_annotation = FALSE,
  upstream = 2000,
  downstream = 2000
)

# Obtain linked SPICEY measures
spicey_coacc <- SPICEY(
  rna = rna,
  atac = atac,
  annotation = annotation_coacc
)

# Make plots
retsi <- spicey_coacc$RETSI |> dplyr::left_join(annotation_coacc, by = c("region_id"))
spicey_heatmap(retsi, spicey_measure = "RETSI")

spicey_heatmap(spicey_coacc$GETSI, spicey_measure = "GETSI")

spicey_heatmap(spicey_coacc$linked, spicey_measure = "SPICEY", combined_score = FALSE)

spicey_heatmap(spicey_coacc$linked, spicey_measure = "SPICEY", combined_score = TRUE)
```

Index

∗ **datasets**
    atac, 8
    cicero_links, 8
    rna, 15
.add_tss_annotation, 2
.parse_input_diff, 3
.standardize_peaks, 4

annotate_with_coaccessibility, 5, *13*, *17*, *19*
annotate_with_nearest, 6, *13*, *17*, *19*
atac, 8

cicero_links, 8
compute_spicey_index, 9

entropy_index, 10
extract_gene_peak_annotations, 11

get_promoters, 12

link_spicey, 12

plot_heatmap, 14, *15*, *20*
prepare_heatmap_data, 14, *14*, *20*

rna, 15

specificity_index, 16
SPICEY, 16, *20*
spicey_heatmap, *14, 15*, 19