Package 'cicero'

November 11, 2026

Type Package

Title Predict cis-co-accessibility from single-cell chromatin accessibility data

Version 1.30.0

Description Cicero computes putative cis-regulatory maps from single-cell chromatin accessibility data. It also extends monocle 2 for use in chromatin accessibility data.

Depends R (>= 3.5.0), monocle, Gviz (>= 1.22.3)

License MIT + file LICENSE

Encoding UTF-8

Imports assertthat (>= 0.2.0), Biobase (>= 2.37.2), BiocGenerics (>= 0.23.0), data.table (>= 1.10.4), dplyr (>= 0.7.4), FNN (>= 1.1), GenomicRanges (>= 1.30.3), ggplot2 (>= 2.2.1), glasso (>= 1.8), grDevices, igraph (>= 1.1.0), IRanges (>= 2.10.5), Matrix (>= 1.2-12), methods, parallel, plyr (>= 1.8.4), reshape2 (>= 1.4.3), S4Vectors (>= 0.14.7), stats, stringi, stringr (>= 1.2.0), tibble (>= 1.4.2), tidyr, VGAM (>= 1.0-5), utils

RoxygenNote 7.2.3

Suggests AnnotationDbi (>= 1.38.2), knitr, markdown, rmarkdown, rtracklayer (>= 1.36.6), testthat, vdiffr (>= 0.2.3), covr

VignetteBuilder knitr

biocViews Sequencing, Clustering, CellBasedAssays, ImmunoOncology, GeneRegulation, GeneTarget, Epigenetics, ATACSeq, SingleCell

LazyData true

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

git_branch RELEASE_3_23

git_last_commit d9adb19

git_last_commit_date 2026-04-28

Repository Bioconductor 3.23

Date/Publication 2026-11-11

Author Hannah Pliner [aut, cre],
Cole Trapnell [aut]

Maintainer Hannah Pliner <hpliner@uw.edu>

Contents

cicero-package 2
aggregate_by_cell_bin 3
aggregate_nearby_peaks 3
annotate_cds_by_site 4
assemble_connections 5
build_gene_activity_matrix 6
cell_data 8
cicero_data 8
compare_connections 9
df_for_coords 9
estimate_distance_parameter 10
find_overlapping_ccans 13
find_overlapping_coordinates 13
generate_ccans 14
generate_cicero_models 15
gene_annotation_sample 17
human.hg19.genome 18
make_atac_cds 18
make_cicero_cds 19
make_sparse_matrix 21
normalize_gene_activities 21
plot_accessibility_in_pseudotime 22
plot_connections 23
ranges_for_coords 26
run_cicero 27

Index **29**

`cicero-package` *cicero*

Description

Cicero computes putative cis-regulatory maps from single-cell chromatin accessibility data. It also extends monocle 2 for use in chromatin accessibility data.

Author(s)

Maintainer: Hannah Pliner <hpliner@uw.edu>

Authors:

- Cole Trapnell <coletrap@uw.edu>

aggregate_by_cell_bin *Aggregate count CDS by groups of cells*

Description

Aggregates a CDS based on an indicator column in the pData table

Usage

```
aggregate_by_cell_bin(cds, group_col)
```

Arguments

cds	A CDS object to be aggregated
group_col	The name of the column in the pData table that indicates the cells assignment to its aggregate bin.

Details

This function takes an input CDS object and collapses cells based on a column in the pData table by summing the values within the cell group.

Value

A count cds aggregated by group_col

Examples

```
data("cicero_data")
#input_cds <- make_atac_cds(cicero_data, binarize = TRUE)
#pData(input_cds)$cell_subtype <- rep(1:10, times=20)
#binned_input_lin <-aggregate_by_cell_bin(input_cds, "cell_subtype")
```

aggregate_nearby_peaks *Make an aggregate count cds by collapsing nearby peaks*

Description

Make an aggregate count cds by collapsing nearby peaks

Usage

```
aggregate_nearby_peaks(cds, distance = 1000)
```

Arguments

cds	A CellDataSet (CDS) object. For example, output of make_atac_cds
distance	The distance within which peaks should be collapsed

Details

This function takes an input CDS object and collapses features within a given distance by summing the values for the collapsed features. Ranges of features are determined by their feature name, so the feature names must be in the form "chr1:1039013-2309023".

Value

A CDS object with aggregated peaks.

Examples

```
data("cicero_data")
input_cds <- make_atac_cds(cicero_data, binarize = TRUE)
agg_cds <- aggregate_nearby_peaks(input_cds, distance = 10000)
```

---

annotate_cds_by_site *Add feature data columns to fData*

Description

Annotate the sites of your CDS with feature data based on coordinate overlap.

Usage

```
annotate_cds_by_site(
  cds,
  feature_data,
  verbose = FALSE,
  maxgap = 0,
  all = FALSE,
  header = FALSE
)
```

Arguments

cds	A CDS object.
feature_data	Data frame, or a character path to a file of feature data. If a path, the file should be tab separated. Default assumes no header, if your file has a header, set header = FALSE. For either a data frame or a path, the file should be in bed-like format, with the first 3 columns containing chromosome, start and stop respectively. The remaining columns will be added to the fData table as feature data.

verbose	Logical, should progress messages be printed?
maxgap	The maximum number of base pairs allowed between the peak and the feature for the feature and peak to be considered overlapping. Default = 0 (overlapping). Details in findOverlaps-methods. If maxgap is set to "nearest" then the nearest feature will be assigned regardless of distance.
all	Logical, should all overlapping intervals be reported? If all is FALSE, the largest overlap is reported.
header	Logical, if reading a file, is there a header?

Details

annotate_cds_by_site will add columns to the fData table of a CDS object based on the overlap of peaks with features in a data frame or file. An "overlap" column will be added, along with any columns beyond the three required columns in the feature data. The "overlap" column is the number of base pairs overlapping the fData site. When maxgap is used, the true overlap is still calculated (overlap will be 0 if the two features only overlap because of maxgap) NA means that there was no overlapping feature. If a peak overlaps multiple data intervals and all is FALSE, the largest overlapping interval will be chosen (in a tie, the first entry is taken), otherwise all intervals will be chosen and annotations will be collapsed using a comma as a separator.

Value

A CDS object with updated fData table.

Examples

```
data("cicero_data")
input_cds <- make_atac_cds(cicero_data, binarize = TRUE)
feat <- data.frame(chr = c("chr18", "chr18", "chr18", "chr18"),
                   bp1 = c(10000, 10800, 50000, 100000),
                   bp2 = c(10700, 11000, 60000, 110000),
                   type = c("Acetylated", "Methylated", "Acetylated",
                   "Methylated"))
input_cds <- annotate_cds_by_site(input_cds, feat)
```

---

assemble_connections *Combine and reconcile cicero models*

Description

Function which takes the output of generate_cicero_models and assembles the connections into a data frame with cicero co-accessibility scores.

Usage

```
assemble_connections(cicero_model_list, silent = FALSE)
```

Arguments

cicero_model_list	A list of cicero output objects, generally, the output of generate_cicero_models.
silent	Logical, should the function run silently?

Details

This function combines glasso models computed on overlapping windows of the genome. Pairs of sites whose regularized correlation was calculated twice are first checked for qualitative concordance (both zero, positive or negative). If they not concordant, NA is returned. If they are concordant the mean is returned.

Value

A data frame of connections with their cicero co-accessibility scores.

See Also

generate_cicero_models

Examples

```
data("cicero_data")
data("human.hg19.genome")
sample_genome <- subset(human.hg19.genome, V1 == "chr18")
sample_genome$V2[1] <- 100000
input_cds <- make_atac_cds(cicero_data, binarize = TRUE)
input_cds <- reduceDimension(input_cds, max_components = 2, num_dim=6,
                             reduction_method = 'tSNE',
                             norm_method = "none")
tsne_coords <- t(reducedDimA(input_cds))
row.names(tsne_coords) <- row.names(pData(input_cds))
cicero_cds <- make_cicero_cds(input_cds, reduced_coordinates = tsne_coords)
model_output <- generate_cicero_models(cicero_cds,
                                       distance_parameter = 0.3,
                                       genomic_coords = sample_genome)
cicero_cons <- assemble_connections(model_output)
```

---

build_gene_activity_matrix

Calculate initial Cicero gene activity matrix

Description

This function calculates the initial Cicero gene activity matrix. After this function, the activity matrix should be normalized with any comparison matrices using the function normalize_gene_activities.

Usage

```
build_gene_activity_matrix(
  input_cds,
  cicero_cons_info,
  site_weights = NULL,
  dist_thresh = 250000,
  coaccess_cutoff = 0.25
)
```

Arguments

input_cds	Binary sci-ATAC-seq input CDS. The input CDS must have a column in the fData table called "gene" which is the gene name if the site is a promoter, and NA if the site is distal.
cicero_cons_info	Cicero connections table, generally the output of run_cicero. This table is a data frame with three required columns named "Peak1", "Peak2", and "coaccess". Peak1 and Peak2 contain coordinates for the two compared elements, and coaccess contains their Cicero co-accessibility score.
site_weights	NULL or an individual weight for each site in input_cds.
dist_thresh	The maximum distance in base pairs between pairs of sites to include in the gene activity calculation.
coaccess_cutoff	The minimum Cicero co-accessibility score that should be considered connected.

Value

Unnormalized gene activity matrix.

Examples

```
data("cicero_data")
data("human.hg19.genome")
sample_genome <- subset(human.hg19.genome, V1 == "chr18")
sample_genome$V2[1] <- 100000
input_cds <- make_atac_cds(cicero_data, binarize = TRUE)
input_cds <- detectGenes(input_cds)
input_cds <- reduceDimension(input_cds, max_components = 2, num_dim=6,
                             reduction_method = 'tSNE',
                             norm_method = "none")
tsne_coords <- t(reducedDimA(input_cds))
row.names(tsne_coords) <- row.names(pData(input_cds))
cicero_cds <- make_cicero_cds(input_cds,
                              reduced_coordinates = tsne_coords)
cons <- run_cicero(cicero_cds, sample_genome, sample_num=2)

data(gene_annotation_sample)
gene_annotation_sub <- gene_annotation_sample[,c(1:3, 8)]
names(gene_annotation_sub)[4] <- "gene"
```

```
input_cds <- annotate_cds_by_site(input_cds, gene_annotation_sub)
num_genes <- pData(input_cds)$num_genes_expressed
names(num_genes) <- row.names(pData(input_cds))
unnorm_ga <- build_gene_activity_matrix(input_cds, cons)
```

`cell_data` *Metadata for example cells in cicero_data*

Description

Metadata information for cicero_data

Usage

```
data(cell_data)
```

Format

A data frame with 200 rows and 2 variables:

timepoint Time at cell collection

cell Cell barcode

`cicero_data` *Example single-cell chromatin accessibility data*

Description

A dataset containing a subset of a single-cell ATAC-seq dataset collected on Human Skeletal Muscle Myoblasts. Only includes data from chromosome 18.

Usage

```
data(cicero_data)
```

Format

A data frame with 35137 rows and 3 variables:

Peak Peak information

Cell Cell ID

Count Reads per cell per peak

compare_connections *Compare Cicero connections to other datasets*

Description

Compare two sets of connections and return a vector of logicals for whether connections in one are present in the other.

Usage

```
compare_connections(conns1, conns2, maxgap = 0)
```

Arguments

conns1	A data frame of Cicero connections, like those output from assemble_connections. The first two columns must be the coordinates of peaks that are connected.
conns2	A data frame of connections to be searched for overlap. The first two columns must be coordinates of genomic sites that are connected.
maxgap	The number of base pairs between peaks allowed to be called overlapping. See findOverlaps-methods in the IRanges package for further description.

Value

A vector of logicals of whether the Cicero pair is present in the alternate dataset.

Examples

```
## Not run:
cons$in_dataset <- compare_connections(conns, alt_data)

## End(Not run)
```

df_for_coords *Construct a data frame of coordinate info from coordinate strings*

Description

Construct a data frame of coordinate info from coordinate strings

Usage

```
df_for_coords(coord_strings)
```

Arguments

coord_strings	A list of coordinate strings (each like "chr1:500000-1000000")

Details

Coordinate strings consist of three pieces of information: chromosome, start, and stop. These pieces of information can be separated by the characters ":", "_", or "-". Commas will be removed, not used as separators (ex: "chr18:8,575,097-8,839,855" is ok).

Value

data.frame with three columns, chromosome, starting base pair and ending base pair

Examples

```
df_for_coords(c("chr1:2,039-30,239", "chrX:28884:101293"))
```

---

estimate_distance_parameter

Calculate distance penalty parameter

Description

Function to calculate distance penalty parameter (`distance_parameter`) for random genomic windows. Used to choose `distance_parameter` to pass to `generate_cicero_models`.

Usage

```
estimate_distance_parameter(
  cds,
  window = 5e+05,
  maxit = 100,
  s = 0.75,
  sample_num = 100,
  distance_constraint = 250000,
  distance_parameter_convergence = 1e-22,
  max_elements = 200,
  genomic_coords = cicero::human.hg19.genome,
  max_sample_windows = 500
)
```

Arguments

cds	A cicero CDS object generated using make_cicero_cds.
window	Size of the genomic window to query, in base pairs.
maxit	Maximum number of iterations for distance_parameter estimation.
s	Power law value. See details for more information.
sample_num	Number of random windows to calculate distance_parameter for.
distance_constraint	Maximum distance of expected connections. Must be smaller than window.
distance_parameter_convergence	Convergence step size for distance_parameter calculation.
max_elements	Maximum number of elements per window allowed. Prevents very large models from slowing performance.
genomic_coords	Either a data frame or a path (character) to a file with chromosome lengths. The file should have two columns, the first is the chromosome name (ex. "chr1") and the second is the chromosome length in base pairs. See data(human.hg19.genome) for an example. If a file, should be tab-separated and without header.
max_sample_windows	Maximum number of random windows to screen to find sample_num windows for distance calculation. Default 500.

Details

The purpose of this function is to calculate the distance scaling parameter used to adjust the distance-based penalty function used in Cicero's model calculation. The scaling parameter, in combination with the power law value s determines the distance-based penalty.

This function chooses random windows of the genome and calculates a distance_parameter. The function returns a vector of values calculated on these random windows. We recommend using the mean value of this vector moving forward with Cicero analysis.

The function works by finding the minimum distance scaling parameter such that no more than 5 distance_constraint have non-zero entries after graphical lasso regularization and such that fewer than 80 nonzero.

If the chosen random window has fewer than 2 or greater than max_elements sites, the window is skipped. In addition, the random window will be skipped if there are insufficient long-range comparisons (see below) to be made. The max_elements parameter exist to prevent very dense windows from slowing the calculation. If you expect that your data may regularly have this many sites in a window, you will need to raise this parameter.

Calculating the distance_parameter in a sample window requires peaks in that window that are at a distance greater than the distance_constraint parameter. If there are not enough examples at high distance have been found, the function will return the warning "Warning: could not calculate sample_num distance_parameters - see documentation details".When looking for sample_num example windows, the function will search max_sample_windows windows. By default this is set at 500, which should be well beyond the 100 windows that need to be found. However, in very sparse datasets, increasing max_sample_windows may help avoid the above warning. Increasing max_sample_windows may slow performance in sparse datasets. If you are still not able

to get enough example windows, even with a large max_sample_windows paramter, this may mean your window parameter needs to be larger or your distance_constraint parameter needs to be smaller. A less likely possibility is that your max_elements parameter needs to be larger. This would occur if your data is particularly dense.

The parameter s is a constant that captures the power-law distribution of contact frequencies between different locations in the genome as a function of their linear distance. For a complete discussion of the various polymer models of DNA packed into the nucleus and of justifiable values for s, we refer readers to (Dekker et al., 2013) for a discussion of justifiable values for s. We use a value of 0.75 by default in Cicero, which corresponds to the "tension globule" polymer model of DNA (Sanborn et al., 2015). This parameter must be the same as the s parameter for generate_cicero_models.

Further details are available in the publication that accompanies this package. Run citation("cicero") for publication details.

Value

A list of results of length sample_num. List members are numeric distance_parameter values.

References

- Dekker, J., Marti-Renom, M.A., and Mirny, L.A. (2013). Exploring the three-dimensional organization of genomes: interpreting chromatin interaction data. Nat. Rev. Genet. 14, 390–403.
- Sanborn, A.L., Rao, S.S.P., Huang, S.-C., Durand, N.C., Huntley, M.H., Jewett, A.I., Bochkov, I.D., Chinnappan, D., Cutkosky, A., Li, J., et al. (2015). Chromatin extrusion explains key features of loop and domain formation in wild-type and engineered genomes. Proc. Natl. Acad. Sci. U. S. A. 112, E6456–E6465.

See Also

generate_cicero_models

Examples

```
data("cicero_data")
data("human.hg19.genome")
sample_genome <- subset(human.hg19.genome, V1 == "chr18")
sample_genome$V2[1] <- 100000
input_cds <- make_atac_cds(cicero_data, binarize = TRUE)
input_cds <- reduceDimension(input_cds, max_components = 2, num_dim=6,
                             reduction_method = 'tSNE',
                             norm_method = "none")
tsne_coords <- t(reducedDimA(input_cds))
row.names(tsne_coords) <- row.names(pData(input_cds))
cicero_cds <- make_cicero_cds(input_cds, reduced_coordinates = tsne_coords)
distance_parameters <- estimate_distance_parameter(cicero_cds,
                                                   sample_num=5,
                                                   genomic_coords = sample_genome)
```

find_overlapping_ccans

Find CCANs that overlap each other in genomic coordinates

Description

Find CCANs that overlap each other in genomic coordinates

Usage

```
find_overlapping_ccans(ccan_assignments, min_overlap = 1)
```

Arguments

ccan_assignments	A data frame where the first column is the peak and the second is the CCAN assignment. For example, output of `generate_ccans`.
min_overlap	The minimum base pair overlap to count as overlapping.

Value

A data frame with two columns, CCAN1 and CCAN2. CCANs in this list are overlapping. The data frame is reciprocal (if CCAN 2 overlaps CCAN 1, there will be two rows, 1,2 and 2,1).

Examples

```
ccan_df <- data.frame(peak = c("chr18_1408345_1408845", "chr18_1779830_1780330",
                               "chr18_1929095_1929595", "chr18_1954501_1954727",
                               "chr18_2049865_2050884", "chr18_2083726_2084102",
                               "chr18_2087935_2088622", "chr18_2104705_2105551",
                               "chr18_2108641_2108907"),
                      CCAN = c(1,2,2,2,3,3,3,3,2))
olap_ccans <- find_overlapping_ccans(ccan_df)
```

find_overlapping_coordinates

Find peaks that overlap a specific genomic location

Description

Find peaks that overlap a specific genomic location

Usage

```
find_overlapping_coordinates(coord_list, coord, maxgap = 0)
```

Arguments

coord_list	A list of coordinates to be searched for overlap in the form chr_100_2000.
coord	The coordinates that you want to find in the form chr1_100_2000.
maxgap	The maximum distance in base pairs between coord and the coord_list that should count as overlapping. Default is 0.

Value

A character vector of the peaks that overlap coord.

Examples

```
test_coords <- c("chr18_10025_10225", "chr18_10603_11103",
                 "chr18_11604_13986",
                 "chr18_157883_158536", "chr18_217477_218555",
                 "chr18_245734_246234")
find_overlapping_coordinates(test_coords, "chr18:10,100-1246234")
```

---

generate_ccans *Generate cis-co-accessibility networks (CCANs)*

Description

Post process cicero co-accessibility scores to extract modules of sites that are co-accessible.

Usage

```
generate_ccans(
  connections_df,
  coaccess_cutoff_override = NULL,
  tolerance_digits = 2
)
```

Arguments

connections_df	Data frame of connections with columns: Peak1, Peak2, coaccess. Generally, the output of run_cicero or assemble_connections
coaccess_cutoff_override	Numeric, co-accessibility score threshold to impose. Overrides automatic calculation.
tolerance_digits	The number of digits to calculate cutoff to. Default is 2 (0.01 tolerance)

Details

CCANs are calculated by first specifying a minimum co-accessibility score and then using the Louvain community detection algorithm on the subgraph induced by excluding edges below this score. For this function, either the user can specify the minimum co-accessibility using `coaccess_cutoff_override`, or the cutoff can be calculated automatically by optimizing for CCAN number. The cutoff calculation can be slow, so users may wish to use the `coaccess_cutoff_override` after initially calculating the cutoff to speed future runs.

Value

Data frame with two columns - Peak and CCAN. CCAN column indicates CCAN assignment. Peaks not included in a CCAN are not returned.

Examples

```
## Not run:
  data("cicero_data")
  set.seed(18)
  data("human.hg19.genome")
  sample_genome <- subset(human.hg19.genome, V1 == "chr18")
  sample_genome$V2[1] <- 100000
  input_cds <- make_atac_cds(cicero_data, binarize = TRUE)
  input_cds <- reduceDimension(input_cds, max_components = 2, num_dim=6,
                               reduction_method = 'tSNE',
                               norm_method = "none")
  tsne_coords <- t(reducedDimA(input_cds))
  row.names(tsne_coords) <- row.names(pData(input_cds))
  cicero_cds <- make_cicero_cds(input_cds, reduced_coordinates = tsne_coords)
  cicero_cons <- run_cicero(cicero_cds, sample_genome, sample_num = 2)
  ccan_assigns <- generate_ccans(cicero_cons)

## End(Not run)
```

`generate_cicero_models` *Generate cicero models*

Description

Function to generate graphical lasso models on all sites in a CDS object within overlapping genomic windows.

Usage

```
generate_cicero_models(
  cds,
  distance_parameter,
```

```
  s = 0.75,
  window = 5e+05,
  max_elements = 200,
  genomic_coords = cicero::human.hg19.genome
)
```

Arguments

cds	A cicero CDS object generated using make_cicero_cds.
distance_parameter	Distance based penalty parameter value. Generally, the mean of the calculated distance_parameter values from estimate_distance_parameter.
s	Power law value. See details.
window	Size of the genomic window to query, in base pairs.
max_elements	Maximum number of elements per window allowed. Prevents very large models from slowing performance.
genomic_coords	Either a data frame or a path (character) to a file with chromosome lengths. The file should have two columns, the first is the chromosome name (ex. "chr1") and the second is the chromosome length in base pairs. See data(human.hg19.genome) for an example. If a file, should be tab-separated and without header.

Details

The purpose of this function is to compute the raw covariances between each pair of sites within overlapping windows of the genome. Within each window, the function then estimates a regularized correlation matrix using the graphical LASSO (Friedman et al., 2008), penalizing pairs of distant sites more than proximal sites. The scaling parameter, distance_parameter, in combination with the power law value s determines the distance-based penalty.

The parameter s is a constant that captures the power-law distribution of contact frequencies between different locations in the genome as a function of their linear distance. For a complete discussion of the various polymer models of DNA packed into the nucleus and of justifiable values for s, we refer readers to (Dekker et al., 2013) for a discussion of justifiable values for s. We use a value of 0.75 by default in Cicero, which corresponds to the "tension globule" polymer model of DNA (Sanborn et al., 2015). This parameter must be the same as the s parameter for estimate_distance_parameter.

Further details are available in the publication that accompanies this package. Run citation("cicero") for publication details.

Value

A list of results for each window. Either a glasso object, or a character description of why the window was skipped. This list can be directly input into assemble_connections to create a reconciled list of cicero co-accessibility scores.

References

- Dekker, J., Marti-Renom, M.A., and Mirny, L.A. (2013). Exploring the three-dimensional organization of genomes: interpreting chromatin interaction data. Nat. Rev. Genet. 14, 390–403.
- Friedman, J., Hastie, T., and Tibshirani, R. (2008). Sparse inverse covariance estimation with the graphical lasso. Biostatistics 9, 432–441.
- Sanborn, A.L., Rao, S.S.P., Huang, S.-C., Durand, N.C., Huntley, M.H., Jewett, A.I., Bochkov, I.D., Chinnappan, D., Cutkosky, A., Li, J., et al. (2015). Chromatin extrusion explains key features of loop and domain formation in wild-type and engineered genomes. Proc. Natl. Acad. Sci. U. S. A. 112, E6456–E6465.

See Also

estimate_distance_parameter

Examples

```
data("cicero_data")
data("human.hg19.genome")
sample_genome <- subset(human.hg19.genome, V1 == "chr18")
sample_genome$V2[1] <- 100000
input_cds <- make_atac_cds(cicero_data, binarize = TRUE)
input_cds <- reduceDimension(input_cds, max_components = 2, num_dim=6,
                             reduction_method = 'tSNE',
                             norm_method = "none")
tsne_coords <- t(reducedDimA(input_cds))
row.names(tsne_coords) <- row.names(pData(input_cds))
cicero_cds <- make_cicero_cds(input_cds, reduced_coordinates = tsne_coords)
model_output <- generate_cicero_models(cicero_cds,
                                       distance_parameter = 0.3,
                                       genomic_coords = sample_genome)
```

---

gene_annotation_sample

Example gene annotation information

Description

Gencode gene annotation data from chromosome 18 of the human genome (hg19).

Usage

```
data(gene_annotation_sample)
```

Format

A data frame with 15129 rows and 8 variables:

chromosome Chromosome

start Exon starting base

end Exon ending base

strand Exon mapping direction

feature Feature type

gene Gene ID

transcript Transcript ID

symbol Gene symbol

---

human.hg19.genome *Chromosome lengths from human genome hg19*

Description

A list of the chromosomes in hg19 and their lengths in base pairs.

Usage

```
data(human.hg19.genome)
```

Format

A data frame with 93 rows and 2 variables:

V1 Chromosome

V2 Chromosome length, base pairs

---

make_atac_cds *Make ATAC CDS object*

Description

This function takes as input a data frame or a path to a file in a sparse matrix format and returns a properly formatted `CellDataSet` (CDS) object.

Usage

```
make_atac_cds(input, binarize = FALSE)
```

Arguments

input	Either a data frame or a path to input data. If a file, it should be a tab-delimited text file with three columns and no header. For either a file or a data frame, the first column is the peak coordinates in the form "chr10_100013372_100013596", the second column is the cell name, and the third column is an integer that represents the number of reads from that cell overlapping that peak. Zero values do not need to be included (sparse matrix format).
binarize	Logical. Should the count matrix be converted to binary?

Value

A CDS object containing your ATAC data in proper format.

Examples

```
data("cicero_data")
input_cds <- make_atac_cds(cicero_data, binarize = TRUE)
```

---

make_cicero_cds *Create cicero input CDS*

Description

Function to generate an aggregated input CDS for cicero. `run_cicero` takes as input an aggregated cicero CDS object. This function will generate the CDS given an input CDS (perhaps generated by `make_atac_cds`) and a value for k, which is the number of cells to be aggregated per bin. The default value for k is 50.

Usage

```
make_cicero_cds(
  cds,
  reduced_coordinates,
  k = 50,
  summary_stats = NULL,
  size_factor_normalize = TRUE,
  silent = FALSE,
  return_agg_info = FALSE
)
```

Arguments

cds	Input CDS object.

`reduced_coordinates`
: A data frame with columns representing the coordinates of each cell in reduced dimension space (generally 2-3 dimensions). `row.names(reduced_coordinates)` should match the cell names in the CDS object. If dimension reduction was done using monocle, tSNE coordinates can be accessed by `t(reducedDimA(cds))`, and DDRTree coordinates can be accessed by `t(reducedDimS(cds))`.

`k`
: Number of cells to aggregate per bin.

`summary_stats`
: Which numeric `pData(cds)` columns you would like summarized (mean) by bin in the resulting CDS object.

`size_factor_normalize`
: Logical, should accessibility values be normalized by size factor?

`silent`
: Logical, should warning and info messages be printed?

`return_agg_info`
: Logical, should a list of the assignments of cells to aggregated bins be output? When `TRUE`, this function returns a list of two items, first, the aggregated CDS object and second, a data.frame with the binning information.

Details

Aggregation of similar cells is done using a k-nearest-neighbors graph and a randomized "bagging" procedure. Details are available in the publication that accompanies this package. Run `citation("cicero")` for publication details. KNN is calculated using `knn.index`

Value

Aggregated CDS object. If return_agg_info is `TRUE`, a list of the aggregated CDS object and a data.frame of aggregation info.

Examples

```
## Not run:
  data("cicero_data")

  input_cds <- make_atac_cds(cicero_data, binarize = TRUE)
  input_cds <- reduceDimension(input_cds, max_components = 2, num_dim=6,
                               reduction_method = 'tSNE',
                               norm_method = "none")
  tsne_coords <- t(reducedDimA(input_cds))
  row.names(tsne_coords) <- row.names(pData(input_cds))
  cicero_cds <- make_cicero_cds(input_cds, reduced_coordinates = tsne_coords)

## End(Not run)
```

make_sparse_matrix *Make a symmetric square sparse matrix from data frame*

Description

Convert a data frame into a square sparse matrix (all versus all)

Usage

```
make_sparse_matrix(data, i.name = "Peak1", j.name = "Peak2", x.name = "value")
```

Arguments

data	data frame
i.name	name of i column
j.name	name of j column
x.name	name of value column

Value

sparse matrix

normalize_gene_activities *Normalize gene activities*

Description

Normalize the output of build_gene_activity_matrix. Input is either one or multiple gene activity matrices. Any gene activities to be compared amongst each other should be normalized together.

Usage

```
normalize_gene_activities(activity_matrices, cell_num_genes)
```

Arguments

activity_matrices	A gene activity matrix, output from build_gene_activity_matrix, or a list of gene activity matrices to be normalized together.
cell_num_genes	A named vector of the total number of accessible sites per cell. Names should correspond to the cell names in the activity matrices. These values can be found in the "num_genes_expressed" column of the pData table of the CDS used to calculate the gene activity matrix.

Value

Normalized activity matrix or matrices.

Examples

```
data("cicero_data")
data("human.hg19.genome")
sample_genome <- subset(human.hg19.genome, V1 == "chr18")
sample_genome$V2[1] <- 100000
input_cds <- make_atac_cds(cicero_data, binarize = TRUE)
input_cds <- detectGenes(input_cds)
input_cds <- reduceDimension(input_cds, max_components = 2, num_dim=6,
                             reduction_method = 'tSNE',
                             norm_method = "none")
tsne_coords <- t(reducedDimA(input_cds))
row.names(tsne_coords) <- row.names(pData(input_cds))
cicero_cds <- make_cicero_cds(input_cds,
                              reduced_coordinates = tsne_coords)
cons <- run_cicero(cicero_cds, sample_genome, sample_num=2)

data(gene_annotation_sample)
gene_annotation_sub <- gene_annotation_sample[,c(1:3, 8)]
names(gene_annotation_sub)[4] <- "gene"
input_cds <- annotate_cds_by_site(input_cds, gene_annotation_sub)
num_genes <- pData(input_cds)$num_genes_expressed
names(num_genes) <- row.names(pData(input_cds))
unnorm_ga <- build_gene_activity_matrix(input_cds, cons)
cicero_gene_activities <- normalize_gene_activities(unnorm_ga, num_genes)
```

---

plot_accessibility_in_pseudotime

Plot accessibility by pseudotime

Description

Make a barplot of chromatin accessibility across pseudotime

Usage

```
plot_accessibility_in_pseudotime(cds_subset, breaks = 10)
```

Arguments

cds_subset	Subset of the CDS object you want to plot. The CDS must have a column in the pData table called "Pseudotime".
breaks	Number of breaks along pseudotime. Controls the coarseness of the plot.

Details

This function plots each site in the CDS subset by cell pseudotime as a barplot. Cells are divided into bins by pseudotime (number determined by `breaks`) and the percent of cells in each bin that are accessible is represented by bar height. In addition, the black line represents the pseudotime-dependent average accessibility from a smoothed binomial regression.

Value

ggplot object

Examples

```
## Not run:
plot_accessibility_in_pseudotime(input_cds_lin[c("chr18_38156577_38158261",
                                                 "chr18_48373358_48374180",
                                                 "chr18_60457956_60459080")])

## End(Not run)
```

`plot_connections` *Plot connections*

Description

Plotting function for Cicero connections. Uses `plotTracks` as its basis

Usage

```
plot_connections(
  connection_df,
  chr,
  minbp,
  maxbp,
  coaccess_cutoff = 0,
  peak_color = "#B4656F",
  connection_color = "#7F7CAF",
  connection_color_legend = TRUE,
  alpha_by_coaccess = FALSE,
  connection_width = 2,
  connection_ymax = NULL,
  gene_model = NULL,
  gene_model_color = "#81D2C7",
  gene_model_shape = c("smallArrow", "box"),
  comparison_track = NULL,
  comparison_coaccess_cutoff = 0,
  comparison_peak_color = "#B4656F",
```

```
  comparison_connection_color = "#7F7CAF",
  comparison_connection_color_legend = TRUE,
  comparison_connection_width = 2,
  comparison_ymax = NULL,
  collapseTranscripts = FALSE,
  include_axis_track = TRUE,
  return_as_list = FALSE,
  viewpoint = NULL,
  comparison_viewpoint = TRUE,
  viewpoint_color = "#F0544F",
  viewpoint_fill = "#EFD8D7",
  viewpoint_alpha = 0.5
)
```

Arguments

connection_df	Data frame of connections, which must include the columns 'Peak1', 'Peak2', and 'coaccess'. Generally, the output of run_cicero or assemble_connections.
chr	The chromosome of the region you would like to plot in the form 'chr10'.
minbp	The base pair coordinate of the start of the region to be plotted.
maxbp	The base pair coordinate of the end of the region to be plotted.
coaccess_cutoff	The minimum cicero co-accessibility score you would like to be plotted. Default is 0.
peak_color	Color for peak annotations - a single color, the name of a column containing color values that correspond to Peak1, or the name of column containing a character or factor to base peak colors on.
connection_color	Color for connection lines. A single color, the name of a column containing color values, or the name of a column containing a character or factor to base connection colors on.
connection_color_legend	Logical, should connection color legend be shown?
alpha_by_coaccess	Logical, should the transparency of connection lines be scaled based on co-accessibility score?
connection_width	Width of connection lines.
connection_ymax	Connection y-axis height. If NULL, chosen automatically.
gene_model	Either NULL or a data.frame. The data.frame should be in a form compatible with the Gviz function GeneRegionTrack-class (cannot have NA as column names).
gene_model_color	Color for gene annotations.

gene_model_shape
: Shape for gene models, passed to GeneRegionTrack-class. Options described at GeneRegionTrack-class.

comparison_track
: Either NULL or a data frame. If a data frame, a second track of connections will be plotted based on this data. This data frame has the same requirements as connection_df (Peak1, Peak2 and coaccess columns).

comparison_coaccess_cutoff
: The minimum cicero co-accessibility score you would like to be plotted for the comparison dataset. Default = 0.

comparison_peak_color
: Color for comparison peak annotations - a single color, the name of a column containing color values that correspond to Peak1, or the name of a column containing a character or factor to base peak colors on.

comparison_connection_color
: Color for comparison connection lines. A single color, the name of a column containing color values, or the name of a column containing a character or factor to base connection colors on.

comparison_connection_color_legend
: Logical, should comparison connection color legend be shown?

comparison_connection_width
: Width of comparison connection lines.

comparison_ymax
: Connection y-axis height for comparison track. If NULL, chosen automatically.

collapseTranscripts
: Logical or character scalar. Can be one in gene, longest, shortest or meta. Variable is passed to the GeneRegionTrack-class function of Gviz. Determines whether and how to collapse related transcripts. See Gviz documentation for details.

include_axis_track
: Logical, should a genomic axis be plotted?

return_as_list
: Logical, if TRUE, the function will not plot, but will return the plot components as a list. Allows user to add/customize Gviz components and plot them separately using plotTracks.

viewpoint
: NULL or Coordinates in form "chr1_10000_10020". Use viewpoint if you would like to plot cicero connections "4C-seq style". Only connections originating in the viewpoint will be shown. Ideal for comparisons with 4C-seq data. If comparison_viewpoint is TRUE, any comparison track will be subsetted as well.

comparison_viewpoint
: Logical, should viewpoint apply to comparison track as well?

viewpoint_color
: Color for the highlight border.

viewpoint_fill
: Color for the highlight fill.

viewpoint_alpha
: Alpha value for the highlight fill.

Value

A gene region plot, or list of components if return_as_list is TRUE.

Examples

```
cicero_cons <- data.frame(
           Peak1 = c("chr18_10034652_10034983", "chr18_10034652_10034983",
                     "chr18_10034652_10034983", "chr18_10034652_10034983",
                     "chr18_10087586_10087901", "chr18_10120685_10127115",
                     "chr18_10097718_10097934", "chr18_10087586_10087901",
                     "chr18_10154818_10155215", "chr18_10238762_10238983",
                     "chr18_10198959_10199183", "chr18_10250985_10251585"),
           Peak2 = c("chr18_10097718_10097934", "chr18_10087586_10087901",
                     "chr18_10154818_10155215", "chr18_10238762_10238983",
                     "chr18_10198959_10199183", "chr18_10250985_10251585",
                     "chr18_10034652_10034983", "chr18_10034652_10034983",
                     "chr18_10034652_10034983", "chr18_10034652_10034983",
                     "chr18_10087586_10087901", "chr18_10120685_10127115"),
           coaccess = c(0.0051121787, 0.0016698617, 0.0006570246,
                        0.0013466927, 0.0737935011, 0.3264019452,
                        0.0051121787, 0.0016698617, 0.0006570246,
                        0.0013466927, 0.0737935011, 0.3264019452))
plot_connections(cicero_cons, chr = "chr18",
                 minbp = 10034652,
                 maxbp = 10251585,
                 peak_color = "purple")
```

---

ranges_for_coords *Construct GRanges objects from coordinate strings*

Description

Construct GRanges objects from coordinate strings

Usage

```
ranges_for_coords(coord_strings, meta_data_df = NULL, with_names = FALSE)
```

Arguments

coord_strings	A list of coordinate strings (in the form "chr1:500000-1000000")
meta_data_df	A data frame with any meta data columns you want included with the ranges. Must be in the same order as coord_strings.
with_names	logical - should meta data include coordinate string (field coord_string)?

Details

Coordinate strings consist of three pieces of information: chromosome, start, and stop. These pieces of information can be separated by the characters ":", "_", or "-". Commas will be removed, not used as separators (ex: "chr18:8,575,097-8,839,855" is ok).

Value

GRanges object of the input strings

See Also

GRanges-class

Examples

```
ran1 <- ranges_for_coords("chr1:2039-30239", with_names = TRUE)
ran2 <- ranges_for_coords(c("chr1:2049-203902", "chrX:489249-1389389"),
                          meta_data_df = data.frame(dat = c("1", "X")))
ran3 <- ranges_for_coords(c("chr1:2049-203902", "chrX:489249-1389389"),
                          with_names = TRUE,
                          meta_data_df = data.frame(dat = c("1", "X"),
                                                    stringsAsFactors = FALSE))
```

---

run_cicero *Run Cicero*

Description

A wrapper function that runs the primary functions of the Cicero pipeline with default parameters. Runs estimate_distance_parameter, generate_cicero_models and assemble_connections. See the manual pages of these functions for details about their function and parameter options. Defaults in this function are designed for mammalian data, those with non-mammalian data should read about parameters in the above functions.

Usage

```
run_cicero(
  cds,
  genomic_coords,
  window = 5e+05,
  silent = FALSE,
  sample_num = 100
)
```

Arguments

cds	Cicero CDS object, created using make_cicero_cds
genomic_coords	Either a data frame or a path (character) to a file with chromosome lengths. The file should have two columns, the first is the chromosome name (ex. "chr1") and the second is the chromosome length in base pairs. See data(human.hg19.genome) for an example. If a file, should be tab-separated and without header.
window	Size of the genomic window to query, in base pairs.
silent	Whether to print progress messages
sample_num	How many sample genomic windows to use to generate distance_parameter parameter. Default: 100.

Value

A table of co-accessibility scores

Examples

```
data("cicero_data")
data("human.hg19.genome")
sample_genome <- subset(human.hg19.genome, V1 == "chr18")
sample_genome$V2[1] <- 100000
input_cds <- make_atac_cds(cicero_data, binarize = TRUE)
input_cds <- reduceDimension(input_cds, max_components = 2, num_dim=6,
                             reduction_method = 'tSNE',
                             norm_method = "none")
tsne_coords <- t(reducedDimA(input_cds))
row.names(tsne_coords) <- row.names(pData(input_cds))
cicero_cds <- make_cicero_cds(input_cds, reduced_coordinates = tsne_coords)
cons <- run_cicero(cicero_cds, sample_genome, sample_num = 2)
```

Index

∗ **datasets**
    cell_data, 8
    cicero_data, 8
    gene_annotation_sample, 17
    human.hg19.genome, 18

aggregate_by_cell_bin, 3
aggregate_nearby_peaks, 3
annotate_cds_by_site, 4
assemble_connections, 5, *14*, *16*, *27*

build_gene_activity_matrix, 6, *21*

cell_data, 8
cicero *(*cicero-package*)*, 2
cicero-package, 2
cicero_data, 8
compare_connections, 9

df_for_coords, 9

estimate_distance_parameter, 10, *16, 17*, *27*

find_overlapping_ccans, 13
find_overlapping_coordinates, 13

gene_annotation_sample, 17
generate_ccans, 14
generate_cicero_models, *5, 6*, *10*, *12*, 15, *27*

human.hg19.genome, 18

knn.index, *20*

make_atac_cds, *4*, 18
make_cicero_cds, *11*, *16*, 19, *28*
make_sparse_matrix, 21

normalize_gene_activities, *6*, 21

plot_accessibility_in_pseudotime, 22
plot_connections, 23
plotTracks, *23*, *25*

ranges_for_coords, 26
run_cicero, *7*, *14*, 27