Package 'APL'

November 11, 2026

Type Package

Title Association Plots

Version 1.16.0

Description APL is a package developed for computation of Association Plots (AP), a method for visualization and analysis of single cell transcriptomics data. The main focus of APL is the identification of genes characteristic for individual clusters of cells from input data. The package performs correspondence analysis (CA) and allows to identify cluster-specific genes using Association Plots. Additionally, APL computes the cluster-specificity scores for all genes which allows to rank the genes by their specificity for a selected cell cluster of interest.

biocViews StatisticalMethod, DimensionReduction, SingleCell, Sequencing, RNASeq, GeneExpression

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.3.2

VignetteBuilder knitr

Imports Matrix, RSpectra, ggrepel, ggplot2, viridisLite, plotly, SeuratObject, SingleCellExperiment, magrittr, SummarizedExperiment, topGO, methods, stats, utils, org.Hs.eg.db, org.Mm.eg.db, rlang

Depends R (>= 4.4.0)

Suggests BiocStyle, knitr, rmarkdown, scRNAseq, scater, scran, sparseMatrixStats, testthat

Config/testthat/edition 3

Collate 'constructor.R' 'CA.R' 'apl.R' 'convert.R' 'generic_methods.R' 'import_packages.R' 'plot.R' 'utils-pipe.R'

URL https://vingronlab.github.io/APL/

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

git_branch RELEASE_3_23

git_last_commit cb20c67

git_last_commit_date 2026-04-28

Repository Bioconductor 3.23

Date/Publication 2026-11-11

Author Clemens Kohl [cre, aut],
Elzbieta Gralinska [aut],
Martin Vingron [aut]

Maintainer Clemens Kohl <kohl.clemens@gmail.com>

Contents

apl . . . 3
apl_coords . . . 4
apl_ggplot . . . 6
apl_plotly . . . 7
apl_score . . . 8
apl_topGO . . . 10
as.cacomp . . . 12
as.list,cacomp-method . . . 14
cacomp . . . 15
cacomp-class . . . 19
cacomp_names . . . 21
cacomp_slot . . . 21
calc_residuals . . . 22
ca_3Dplot . . . 23
ca_biplot . . . 25
ca_coords . . . 28
check_cacomp . . . 29
clip_residuals . . . 29
comp_ft_residuals . . . 30
comp_NB_residuals . . . 31
comp_std_residuals . . . 32
elbow_method . . . 33
inertia_rows . . . 34
is.empty . . . 34
permutation_cutoff . . . 35
pick_dims . . . 36
plot_enrichment . . . 39
random_direction_cutoff . . . 40
recompute . . . 41
rm_zeros . . . 41
run_APL . . . 42
run_cacomp . . . 49
scree_plot . . . 50
show.cacomp . . . 51
subset_dims . . . 52

var_rows 52
%>% 53

Index **55**

`apl` *Association Plot*

Description

Plot an Association Plot for the chosen columns.

Usage

```
apl(
  caobj,
  type = "ggplot",
  rows_idx = NULL,
  cols_idx = caobj@group,
  row_labs = FALSE,
  col_labs = FALSE,
  show_score = FALSE,
  show_cols = FALSE,
  show_rows = TRUE,
  score_cutoff = 0,
  score_color = "rainbow"
)
```

Arguments

`caobj`	An object of class "cacomp" and "APL" with apl coordinates calculated.
`type`	"ggplot"/"plotly". For a static plot a string "ggplot", for an interactive plot "plotly". Default "ggplot".
`rows_idx`	numeric/character vector. Indices or names of the rows that should be labelled. Default NULL.
`cols_idx`	numeric/character vector. Indices or names of the columns that should be labelled. Default is only to label columns making up the centroid: caobj@group.
`row_labs`	Logical. Whether labels for rows indicated by rows_idx should be labeled with text. Default TRUE.
`col_labs`	Logical. Whether labels for columns indicated by cols_idx shouls be labeled with text. Default FALSE.
`show_score`	Logical. Whether the S-alpha score should be shown in the plot.
`show_cols`	Logical. Whether column points should be plotted.
`show_rows`	Logical. Whether row points should be plotted.
`score_cutoff`	Numeric. Rows (genes) with a score >= score_cutoff will be colored according to their score if show_score = TRUE.
`score_color`	Either "rainbow" or "viridis".

Details

For an interactive plot type="plotly" can be chosen, otherwise a static plot will be returned. The row and column coordinates have to be already calculated by ‘apl_coords()‘.

Value

Either a ggplot or plotly object.

References

Association Plots: Visualizing associations in high-dimensional correspondence analysis biplots
Elzbieta Gralinska, Martin Vingron
bioRxiv 2020.10.23.352096; doi: https://doi.org/10.1101/2020.10.23.352096

Examples

```
set.seed(1234)

# Simulate counts
cnts <- mapply(function(x){rpois(n = 500, lambda = x)},
               x = sample(1:100, 50, replace = TRUE))
rownames(cnts) <- paste0("gene_", 1:nrow(cnts))
colnames(cnts) <- paste0("cell_", 1:ncol(cnts))

# Run correspondence analysis
ca <- cacomp(obj = cnts, princ_coords = 3)

# Calculate APL coordinates for arbitrary group
ca <- apl_coords(ca, group = 1:10)

# plot results
# Note:
# Due to random gene expression & group, no highly
# associated genes are visible.
apl(ca, type = "ggplot")
```

---

apl_coords *Calculate Association Plot coordinates*

Description

Calculates the Association Plot coordinates for either the rows, columns or both (default).

Usage

```
apl_coords(caobj, group, calc_rows = TRUE, calc_cols = TRUE)
```

Arguments

caobj	A "cacomp" object with principal row coordinates and standardized column coordinates calculated.
group	Numeric/Character. Vector of indices or column names of the columns to calculate centroid/x-axis direction.
calc_rows	TRUE/FALSE. Whether apl row coordinates should be calculated. Default TRUE.
calc_cols	TRUE/FALSE. Whether apl column coordinates should be calculated. Default TRUE.

Details

Coordinates (x,y) of row vector $\vec{r}$ are defined as

$$x(\vec{r}) := |\vec{r}| \cos(\phi(\vec{r}))$$

$$y(\vec{r}) := |\vec{r}| \sin(\phi(\vec{r}))$$

The x-direction is determined by calculating the centroid of the columns selected with the indices in "group".

Value

Returns input "cacomp" object and adds components "apl_rows" and/or "apl_cols" for row and column coordinates. In "group" the indices of the columns used to calculate the centroid are saved.

References

Association Plots: Visualizing associations in high-dimensional correspondence analysis biplots
Elzbieta Gralinska, Martin Vingron bioRxiv 2020.10.23.352096; doi: https://doi.org/10.1101/2020.10.23.352096

Examples

```
set.seed(1234)
# Simulate scRNAseq data
cnts <- data.frame(cell_1 = rpois(10, 5),
                   cell_2 = rpois(10, 10),
                   cell_3 = rpois(10, 20),
                   cell_4 = rpois(10, 20))
rownames(cnts) <- paste0("gene_", 1:10)
cnts <- as.matrix(cnts)

# Run correspondence analysis
ca <- cacomp(obj = cnts, princ_coords = 3, dims = 3)
# Calculate APL coordinates
ca <- apl_coords(ca, group = 3:4)
```

apl_ggplot *Plot Association Plot with ggplot*

Description

Uses ggplot to plot an Association Plot

Usage

```
apl_ggplot(
  rows,
  rows_group = NULL,
  cols,
  cols_group = NULL,
  rows_scored = NULL,
  rows_color = "#0066FF",
  rows_high_color = "#FF0000",
  cols_color = "#601A4A",
  cols_high_color = "#EE442F",
  score_color = "rainbow",
  row_labs = FALSE,
  col_labs = FALSE,
  show_score = FALSE,
  show_cols = FALSE,
  show_rows = TRUE
)
```

Arguments

rows	Row APL-coordinates
rows_group	Row AP-coordinates to highlight
cols	Column AP-coordinates
cols_group	Column AP-coordinates for the group to be highlighted.
rows_scored	Row AP-coordinates of rows above a score cutoff.
rows_color	Color for rows
rows_high_color	Color for rows to be highlighted.
cols_color	Column points color.
cols_high_color	Color for column points to be highlighted..
score_color	Color scheme for row points with a score.
row_labs	Logical. Whether labels for rows indicated by rows_idx should be labeled with text. Default TRUE.

col_labs	Logical. Whether labels for columns indicated by cols_idx shouls be labeled with text. Default FALSE.
show_score	Logical. Whether the S-alpha score should be shown in the plot.
show_cols	Logical. Whether column points should be plotted.
show_rows	Logical. Whether row points should be plotted.

Value

ggplot Association Plot

`apl_plotly` *Plot Association Plot with plotly*

Description

Uses plotly to generate an interactive Association Plot

Usage

```
apl_plotly(
  rows,
  rows_group = NULL,
  cols,
  cols_group,
  rows_scored = NULL,
  rows_color = "#0066FF",
  rows_high_color = "#FF0000",
  cols_color = "#601A4A",
  cols_high_color = "#EE442F",
  score_color = "rainbow",
  row_labs = FALSE,
  col_labs = FALSE,
  show_score = FALSE,
  show_cols = FALSE,
  show_rows = TRUE
)
```

Arguments

rows	Row APL-coordinates
rows_group	Row AP-coordinates to highlight
cols	Column AP-coordinates
cols_group	Column AP-coordinates for the group to be highlighted.
rows_scored	Row AP-coordinates of rows above a score cutoff.
rows_color	Color for rows

rows_high_color
: Color for rows to be highlighted.

cols_color
: Column points color.

cols_high_color
: Color for column points to be highlighted.

score_color
: Color scheme for row points with a score.

row_labs
: Logical. Whether labels for rows indicated by rows_idx should be labeled with text. Default TRUE.

col_labs
: Logical. Whether labels for columns indicated by cols_idx shouls be labeled with text. Default FALSE.

show_score
: Logical. Whether the S-alpha score should be shown in the plot.

show_cols
: Logical. Whether column points should be plotted.

show_rows
: Logical. Whether row points should be plotted.

Value

Interactive plotly Association Plot

---

apl_score *Find rows most highly associated with a condition*

Description

Ranks rows by a calculated score which balances the association of the row with the condition and how associated it is with other conditions.

Usage

```
apl_score(
  caobj,
  mat = NULL,
  dims = caobj@dims,
  group = caobj@group,
  reps = 10,
  quant = 0.99,
  python = FALSE,
  store_perm = TRUE,
  method = "permutation"
)
```

Arguments

caobj	A "cacomp" object with principal row coordinates and standardized column coordinates calculated.
mat	A numeric matrix. For sequencing a count matrix, gene expression values with genes in rows and samples/cells in columns. Should contain row and column names.
dims	Integer. Number of CA dimensions to retain. Needs to be the same as in caobj!
group	Vector of indices of the columns to calculate centroid/x-axis direction.
reps	Integer. Number of permutations to perform.
quant	Numeric. Single number between 0 and 1 indicating the quantile used to calculate the cutoff. Default 0.99.
python	DEPRACTED. A logical value indicating whether to use singular-value decomposition from the python package torch.
store_perm	Logical. Whether permuted data should be stored in the CA object. This implementation dramatically speeds up computation compared to ‘svd()‘ in R.
method	Method to calculate the cutoff. Either "random" for random direction method or "permutation" for the permutation method.

Details

The score is calculated by permuting the values of each row to determine the cutoff angle of the 99

$$S_{alpha}(x, y) = x - \frac{y}{\tan \alpha}$$

By default the permutation is repeated 10 times (for random direction min. 300 repetition is recommended!), but for very large matrices this can be reduced. The method "permutation" permutes the columns in each row and calculates AP-coordinates for each such permutation. The cutoff is then taken by the quantile specified by "quan". The "random" method in contrast calculates AP-coordinates for the original data, but by looking into random directions.

If store_perm is TRUE the permuted data is stored in the cacomp object and can be used for future scoring.

Value

Returns the input "cacomp" object with "APL_score" component added. APL_score contains a data frame with ranked rows, their score and their original row number.

References

Association Plots: Visualizing associations in high-dimensional correspondence analysis biplots
Elzbieta Gralinska, Martin Vingron
bioRxiv 2020.10.23.352096; doi: https://doi.org/10.1101/2020.10.23.352096

Examples

```
set.seed(1234)

# Simulate counts
cnts <- mapply(function(x){rpois(n = 500, lambda = x)},
               x = sample(1:20, 50, replace = TRUE))
rownames(cnts) <- paste0("gene_", 1:nrow(cnts))
colnames(cnts) <- paste0("cell_", 1:ncol(cnts))

# Run correspondence analysis.
ca <- cacomp(obj = cnts, princ_coords = 3)

# Calculate APL coordinates:
ca <- apl_coords(ca, group = 1:10)

# Rank genes by S-alpha score
ca <- apl_score(ca, mat = cnts)
```

`apl_topGO` *Run Gene overrepresentation analysis with topGO*

Description

This function uses the Kolmogorov-Smirnov test as implemented by the package topGO to test for overrepresentation in Gene Ontology gene sets.

Usage

```
apl_topGO(
  caobj,
  ontology,
  organism = "hs",
  ngenes = 1000,
  score_cutoff = 0,
  use_coords = FALSE,
  return_plot = FALSE,
  top_res = 15
)
```

Arguments

`caobj`	A "cacomp" object with principal row coordinates and standardized column coordinates calculated.
`ontology`	Character string. Chooses GO sets for 'BP' (biological processes), 'CC' (cell compartment) or 'MF' (molecular function).
`organism`	Character string. Either 'hs' (homo sapiens), 'mm' (mus musculus) or the name of the organism package such as 'org.*.eg.db'.

ngenes	Numeric. Number of top ranked genes to test for overrepresentation.
score_cutoff	numeric. S-alpha score cutoff. Only genes with a score larger will be tested.
use_coords	Logical. Whether the x-coordinates of the row APL coordinates should be used for ranking. Only recommended when no S-alpha score (see apl_score()) can be calculated.
return_plot	Logical. Whether a plot of significant gene sets should be additionally returned.
top_res	Numeric. Number of top scoring genes to plot.

Details

For a chosen group of cells/samples, the top 'ngenes' group specific genes are used for gene overrepresentation analysis. The genes are ranked either by the precomputed APL score, or, if not available by their APL x-coordinates.

Value

A data.frame containing the gene sets with the highest overrepresentation.

References

Adrian Alexa and Jorg Rahnenfuhrer
topGO: Enrichment Analysis for Gene Ontology.
R package version 2.42.0.

Examples

```
library(SeuratObject)
set.seed(1234)
cnts <- SeuratObject::LayerData(pbmc_small, assay = "RNA", layer = "counts")
cnts <- as.matrix(cnts)

# Run CA on example from Seurat

ca <- cacomp(pbmc_small,
             princ_coords = 3,
             return_input = FALSE,
             assay = "RNA",
             slot = "counts")

grp <- which(Idents(pbmc_small) == 2)
ca <- apl_coords(ca, group = grp)
ca <- apl_score(ca,
                mat = cnts)

enr <- apl_topGO(ca,
                 ontology = "BP",
                 organism = "hs")

plot_enrichment(enr)
```

`as.cacomp` *Create cacomp object from Seurat/SingleCellExperiment container*

Description

Converts the values stored in the Seurat/SingleCellExperiment dimensional reduction slot "CA" to a cacomp object. If recompute = TRUE additional parameters are recomputed from the saved values without rerunning SVD (need to specify assay to work).

as.cacomp.cacomp returns input without any calculations.

Recomputes missing values and returns cacomp object from a list. If you have a *complete* cacomp object in list form, use do.call(new_cacomp, obj).

as.cacomp.Seurat: Converts the values stored in the Seurat DimReduc slot "CA" to an cacomp object.

as.cacomp.SingleCellExperiment: Converts the values stored in the SingleCellExperiment reducedDim slot "CA" to a cacomp object.

Usage

```
as.cacomp(obj, ...)

## S4 method for signature 'cacomp'
as.cacomp(obj, ...)

## S4 method for signature 'list'
as.cacomp(obj, ..., mat = NULL)

## S4 method for signature 'Seurat'
as.cacomp(obj, ..., assay = "RNA", slot = "counts")

## S4 method for signature 'SingleCellExperiment'
as.cacomp(obj, ..., assay = "counts")
```

Arguments

`obj`	An object of class "Seurat" or "SingleCellExperiment" with a dim. reduction named "CA" saved. For obj "cacomp" input is returned.
`...`	Further arguments.
`mat`	Original input matrix.
`assay`	Character. The assay from which extract the count matrix, e.g. "RNA" for Seurat objects or "counts"/"logcounts" for SingleCellExperiments.
`slot`	character. Slot of the Seurat assay to use. Default "counts".

Details

By default extracts std_coords_cols, D, prin_coords_rows, top_rows and dims from obj and outputs a cacomp object. If recompute = TRUE the following are additionally recalculated (doesn't run SVD): U, V, std_coords_rows, row_masses, col_masses.

Value

A cacomp object.

Examples

```
#########
# lists #
#########

# Simulate counts
cnts <- mapply(function(x){rpois(n = 500, lambda = x)},
               x = sample(1:100, 50, replace = TRUE))
rownames(cnts) <- paste0("gene_", 1:nrow(cnts))
colnames(cnts) <- paste0("cell_", 1:ncol(cnts))

# Run correspondence analysis
ca <- cacomp(obj = cnts, princ_coords = 3)
ca_list <- as.list(ca)

# Only keep subset of elements for demonstration
ca_list <- ca_list[c("U", "std_coords_rows", "std_coords_cols", "params")]

# convert (incomplete) list to cacomp object.
ca <- as.cacomp(ca_list, mat = cnts)

##########
# Seurat #
##########
library(SeuratObject)
set.seed(1234)

# Simulate counts
cnts <- mapply(function(x){rpois(n = 500, lambda = x)},
               x = sample(1:100, 50, replace = TRUE))
rownames(cnts) <- paste0("gene_", 1:nrow(cnts))
colnames(cnts) <- paste0("cell_", 1:ncol(cnts))

seu <- CreateSeuratObject(counts = cnts)
seu <- cacomp(seu, return_input = TRUE)

ca <- as.cacomp(seu, assay = "RNA", slot = "counts")

########################
# SingleCellExperiment #
########################
library(SingleCellExperiment)
```

```
set.seed(1234)

# Simulate counts
cnts <- mapply(function(x){rpois(n = 500, lambda = x)},
               x = sample(1:100, 50, replace = TRUE))
rownames(cnts) <- paste0("gene_", 1:nrow(cnts))
colnames(cnts) <- paste0("cell_", 1:ncol(cnts))

sce <- SingleCellExperiment(assays=list(counts=cnts))
sce <- cacomp(sce, return_input = TRUE)

ca <- as.cacomp(sce, assay = "counts")
```

`as.list,cacomp-method` *Convert cacomp object to list.*

Description

Convert cacomp object to list.

Usage

```
## S4 method for signature 'cacomp'
as.list(x)
```

Arguments

x	A cacomp object.

Value

A cacomp object.

Examples

```
# Simulate counts
cnts <- mapply(function(x){rpois(n = 500, lambda = x)},
               x = sample(1:100, 50, replace = TRUE))
rownames(cnts) <- paste0("gene_", 1:nrow(cnts))
colnames(cnts) <- paste0("cell_", 1:ncol(cnts))

# Run correspondence analysis
ca <- cacomp(obj = cnts, princ_coords = 3)
ca_list <- as.list(ca)
```

cacomp *Correspondance Analysis*

Description

'cacomp' performs correspondence analysis on a matrix or Seurat/SingleCellExperiment object and returns the transformed data.

'cacomp.seurat' performs correspondence analysis on an assay from a Seurat container and stores the standardized coordinates of the columns (= cells) and the principal coordinates of the rows (= genes) as a DimReduc Object in the Seurat container.

'cacomp.SingleCellExperiment' performs correspondence analysis on an assay from a SingleCellExperiment and stores the standardized coordinates of the columns (= cells) and the principal coordinates of the rows (= genes) as a matrix in the SingleCellExperiment container.

Usage

```
cacomp(
  obj,
  coords = TRUE,
  princ_coords = 3,
  python = FALSE,
  dims = NULL,
  top = 5000,
  inertia = TRUE,
  rm_zeros = TRUE,
  residuals = "pearson",
  cutoff = NULL,
  clip = FALSE,
  ...
)

## S4 method for signature 'matrix'
cacomp(
  obj,
  coords = TRUE,
  princ_coords = 3,
  python = FALSE,
  dims = NULL,
  top = 5000,
  inertia = TRUE,
  rm_zeros = TRUE,
  residuals = "pearson",
  cutoff = NULL,
  clip = FALSE,
  ...
)
```

```
## S4 method for signature 'dgCMatrix'
cacomp(
  obj,
  coords = TRUE,
  princ_coords = 3,
  python = FALSE,
  dims = NULL,
  top = 5000,
  inertia = TRUE,
  rm_zeros = TRUE,
  residuals = "pearson",
  cutoff = NULL,
  clip = FALSE,
  ...
)

## S4 method for signature 'Seurat'
cacomp(
  obj,
  coords = TRUE,
  princ_coords = 3,
  python = FALSE,
  dims = NULL,
  top = 5000,
  inertia = TRUE,
  rm_zeros = TRUE,
  residuals = "pearson",
  cutoff = NULL,
  clip = FALSE,
  ...,
  assay = SeuratObject::DefaultAssay(obj),
  slot = "counts",
  return_input = FALSE
)

## S4 method for signature 'SingleCellExperiment'
cacomp(
  obj,
  coords = TRUE,
  princ_coords = 3,
  python = FALSE,
  dims = NULL,
  top = 5000,
  inertia = TRUE,
  rm_zeros = TRUE,
  residuals = "pearson",
  cutoff = NULL,
```

```
  clip = FALSE,
  ...,
  assay = "counts",
  return_input = FALSE
)
```

Arguments

obj	A numeric matrix or Seurat/SingleCellExperiment object. For sequencing a count matrix, gene expression values with genes in rows and samples/cells in columns. Should contain row and column names.
coords	Logical. Indicates whether CA standard coordinates should be calculated.
princ_coords	Integer. Number indicating whether principal coordinates should be calculated for the rows (=1), columns (=2), both (=3) or none (=0).
python	DEPRACTED. A logical value indicating whether to use singular-value decomposition from the python package torch. This implementation dramatically speeds up computation compared to 'svd()' in R when calculating the full SVD. This parameter only works when dims==NULL or dims==rank(mat), where caculating a full SVD is demanded.
dims	Integer. Number of CA dimensions to retain. If NULL: (0.2 * min(nrow(A), ncol(A)) - 1).
top	Integer. Number of most variable rows to retain. Set NULL to keep all.
inertia	Logical. Whether total, row and column inertias should be calculated and returned.
rm_zeros	Logical. Whether rows & cols containing only 0s should be removed. Keeping zero only rows/cols might lead to unexpected results.
residuals	character string. Specifies which kind of residuals should be calculated. Can be "pearson" (default), "freemantukey" or "NB" for negative-binomial.
cutoff	numeric. Residuals that are larger than cutoff or lower than -cutoff are clipped to cutoff.
clip	logical. Whether residuals should be clipped if they are higher/lower than a specified cutoff
...	Other parameters
assay	Character. The assay from which extract the count matrix for SVD, e.g. "RNA" for Seurat objects or "counts"/"logcounts" for SingleCellExperiments.
slot	character. The slot of the Seurat assay. Default "counts".
return_input	Logical. If TRUE returns the input (SingleCellExperiment/Seurat object) with the CA results saved in the reducedDim/DimReduc slot "CA". Otherwise returns a "cacomp". Default FALSE.

Details

The calculation is performed according to the work of Michael Greenacre. Singular value decomposition can be performed either with the base R function 'svd' or preferably by the faster pytorch implementation (python = TRUE). When working with large matrices, CA coordinates and principal coordinates should only be computed when needed to save computational time.

Value

Returns a named list of class "cacomp" with components U, V and D: The results from the SVD. row_masses and col_masses: Row and columns masses. top_rows: How many of the most variable rows were retained for the analysis. tot_inertia, row_inertia and col_inertia: Only if inertia = TRUE. Total, row and column inertia respectively.

If return_imput = TRUE with Seurat container: Returns input obj of class "Seurat" with a new Dimensional Reduction Object named "CA". Standard coordinates of the cells are saved as embeddings, the principal coordinates of the genes as loadings and the singular values (= square root of principal intertias/eigenvalues) are stored as stdev. To recompute a regular "cacomp" object without rerunning cacomp use ‘as.cacomp()‘.

If return_input =TRUE for SingleCellExperiment input returns a SingleCellExperiment object with a matrix of standardized coordinates of the columns in reducedDim(obj, "CA"). Additionally, the matrix contains the following attributes: "prin_coords_rows": Principal coordinates of the rows. "singval": Singular values. For the explained inertia of each principal axis calculate singval^2. "percInertia": Percent explained inertia of each principal axis. To recompute a regular "cacomp" object from a SingleCellExperiment without rerunning cacomp use ‘as.cacomp()‘.

References

Greenacre, M. Correspondence Analysis in Practice, Third Edition, 2017.

Examples

```
# Simulate scRNAseq data.
cnts <- data.frame(cell_1 = rpois(10, 5),
                   cell_2 = rpois(10, 10),
                   cell_3 = rpois(10, 20))
rownames(cnts) <- paste0("gene_", 1:10)
cnts <- as.matrix(cnts)

# Run correspondence analysis.
ca <- cacomp(obj = cnts, princ_coords = 3, top = 5)

############
# Seurat  #
############
library(SeuratObject)
set.seed(1234)

# Simulate counts
cnts <- mapply(function(x){rpois(n = 500, lambda = x)},
                     x = sample(1:20, 50, replace = TRUE))
rownames(cnts) <- paste0("gene_", 1:nrow(cnts))
colnames(cnts) <- paste0("cell_", 1:ncol(cnts))

# Create Seurat object
seu <- CreateSeuratObject(counts = cnts)

# Run CA and save in dim. reduction slot
seu <- cacomp(seu, return_input = TRUE, assay = "RNA", slot = "counts")
```

```
# Run CA and return cacomp object
ca <- cacomp(seu, return_input = FALSE, assay = "RNA", slot = "counts")

########################
# SingleCellExperiment #
########################
library(SingleCellExperiment)
set.seed(1234)

# Simulate counts
cnts <- mapply(function(x){rpois(n = 500, lambda = x)},
               x = sample(1:20, 50, replace = TRUE))
rownames(cnts) <- paste0("gene_", 1:nrow(cnts))
colnames(cnts) <- paste0("cell_", 1:ncol(cnts))
logcnts <- log2(cnts + 1)

# Create SingleCellExperiment object
sce <- SingleCellExperiment(assays=list(counts=cnts, logcounts=logcnts))

# run CA and save in dim. reduction slot.
sce <- cacomp(sce, return_input = TRUE, assay = "counts") # on counts
sce <- cacomp(sce, return_input = TRUE, assay = "logcounts") # on logcounts

# run CA and return cacomp object.
ca <- cacomp(sce, return_input = FALSE, assay = "counts")
```

`cacomp-class` *An S4 class that contains all elements needed for CA.*

Description

This class contains elements necessary to computer CA coordinates or Association Plot coordinates, as well as other informative data such as row/column inertia, gene-wise APL-scores, etc. ...

Creates new cacomp object.

Usage

```
new_cacomp(...)
```

Arguments

`...`	slot names and objects for new cacomp object.

Value

cacomp object

Slots

`U` class "matrix". Left singular vectors of the original input matrix.

`V` class "matrix". Right singular vectors of the original input matrix.

`D` class "numeric". Singular values of the original inpt matrix.

`std_coords_rows` class "matrix". Standardized CA coordinates of the rows.

`std_coords_cols` class "matrix". Standardized CA coordinates of the columns.

`prin_coords_rows` class "matrix". Principal CA coordinates of the rows.

`prin_coords_cols` class "matrix". Principal CA coordinates of the columns.

`apl_rows` class "matrix". Association Plot coordinates of the rows for the direction defined in slot "group"

`apl_cols` class "matrix". Association Plot coordinates of the columns for the direction defined in slot "group"

`APL_score` class "data.frame". Contains rows sorted by the APL score. Columns: Rowname (gene name in the case of gene expression data), APL score calculated for the direction defined in slot "group", the original row number and the rank of the row as determined by the score.

`dims` class "numeric". Number of dimensions in CA space.

`group` class "numeric". Indices of the chosen columns for APL calculations.

`row_masses` class "numeric". Row masses of the frequency table.

`col_masses` class "numeric". Column masses of the frequency table.

`top_rows` class "numeric". Number of most variable rows chosen.

`tot_inertia` class "numeric". Total inertia in CA space.

`row_inertia` class "numeric". Row-wise inertia in CA space.

`col_inertia` class "numeric". Column-wise inertia in CA space.

`permuted_data` class "list". Storage slot for permuted data.

`params` class "list". List of parameters.

Examples

```
set.seed(1234)

# Simulate counts
cnts <- mapply(function(x){rpois(n = 500, lambda = x)},
               x = sample(1:20, 50, replace = TRUE))
rownames(cnts) <- paste0("gene_", 1:nrow(cnts))
colnames(cnts) <- paste0("cell_", 1:ncol(cnts))

res <-  APL:::comp_std_residuals(mat=cnts)
SVD <- svd(res$S)
names(SVD) <- c("D", "U", "V")
SVD <- SVD[c(2, 1, 3)]

ca <- new_cacomp(U = SVD$U,
                 V = SVD$V,
                 D = SVD$D,
                 row_masses = res$rowm,
                 col_masses = res$colm)
```

cacomp_names *Prints slot names of cacomp object*

Description

Prints slot names of cacomp object

Usage

```
cacomp_names(caobj)
```

Arguments

caobj	a cacomp object

Value

Prints slot names of cacomp object

Examples

```
# Simulate scRNAseq data.
cnts <- data.frame(cell_1 = rpois(10, 5),
                   cell_2 = rpois(10, 10),
                   cell_3 = rpois(10, 20))
rownames(cnts) <- paste0("gene_", 1:10)
cnts <- as.matrix(cnts)

# Run correspondence analysis.
ca <- cacomp(obj = cnts, princ_coords = 3, top = 5)

# show slot names:
cacomp_names(ca)
```

cacomp_slot *Access slots in a cacomp object*

Description

Access slots in a cacomp object

Usage

```
cacomp_slot(caobj, slot)
```

Arguments

caobj	a cacomp object
slot	slot to return

Value

Chosen slot of the cacomp object

Examples

```
# Simulate scRNAseq data.
cnts <- data.frame(cell_1 = rpois(10, 5),
                   cell_2 = rpois(10, 10),
                   cell_3 = rpois(10, 20))
rownames(cnts) <- paste0("gene_", 1:10)
cnts <- as.matrix(cnts)

# Run correspondence analysis.
ca <- cacomp(obj = cnts, princ_coords = 3, top = 5)

# access left singular vectors
cacomp_slot(ca, "U")
```

---

calc_residuals *Calculate residuals for Correspondence analysis*

Description

calc_residuals provides optional residuals as the basis for Correspondence Analysis

Usage

```
calc_residuals(mat, residuals = "pearson", clip = FALSE, cutoff = NULL)
```

Arguments

mat	A numerical matrix or coercible to one by ‘as.matrix()‘. Should have row and column names.
residuals	character string. Specifies which kind of residuals should be calculated. Can be "pearson" (default), "freemantukey" or "NB" for negative-binomial.
clip	logical. Whether residuals should be clipped if they are higher/lower than a specified cutoff
cutoff	numeric. Residuals that are larger than cutoff or lower than -cutoff are clipped to cutoff.

Value

A named list. The elements are:

- "S": standardized residual matrix.
- "tot": grand total of the original matrix.
- "rowm": row masses.
- "colm": column masses.

`ca_3Dplot` *Plot of the first 3D CA projection of the data.*

Description

Plots the first 3 dimensions of the rows and columns in the same plot.

Usage

```
ca_3Dplot(
  obj,
  xdim = 1,
  ydim = 2,
  zdim = 3,
  princ_coords = 1,
  row_labels = NULL,
  col_labels = NULL,
  ...
)

## S4 method for signature 'cacomp'
ca_3Dplot(
  obj,
  xdim = 1,
  ydim = 2,
  zdim = 3,
  princ_coords = 1,
  row_labels = NULL,
  col_labels = NULL,
  ...
)

## S4 method for signature 'Seurat'
ca_3Dplot(
  obj,
  xdim = 1,
  ydim = 2,
  zdim = 3,
```

```
  princ_coords = 1,
  row_labels = NULL,
  col_labels = NULL,
  ...,
  assay = SeuratObject::DefaultAssay(obj),
  slot = "counts"
)

## S4 method for signature 'SingleCellExperiment'
ca_3Dplot(
  obj,
  xdim = 1,
  ydim = 2,
  zdim = 3,
  princ_coords = 1,
  row_labels = NULL,
  col_labels = NULL,
  ...,
  assay = "counts"
)
```

Arguments

obj	An object of class "cacomp", or alternatively an object of class "Seurat" or "SingleCellExperiment" with a dim. reduction named "CA" saved.
xdim	Integer. The dimension for the x-axis. Default 1.
ydim	Integer. The dimension for the y-axis. Default 2.
zdim	Integer. The dimension for the z-axis. Default 3.
princ_coords	Integer. If 1 then principal coordinates are used for the rows, if 2 for the columns. Default 1 (rows).
row_labels	Numeric vector. Indices for the rows for which a label should be added (label should be stored in rownames). Default NULL.
col_labels	Numeric vector. Indices for the columns for which a label should be added (label should be stored in colnames). Default NULL (no columns).
...	Further arguments.
assay	SingleCellExperiment assay to obtain counts from.
slot	Seurat slot from assay to get count matrix from.

Details

Depending on whether ‘princ_coords‘ is set to 1 or 2 either the principal coordinates of either the rows (1) or the columns (2) are chosen. For the other the standardized coordinates are plotted (assymetric biplot). Labels for rows and columns should be stored in the row- and column names respectively.

Value

Plot of class "plotly".

Examples

```
# Simulate counts
cnts <- mapply(function(x){rpois(n = 500, lambda = x)},
               x = sample(1:100, 50, replace = TRUE))
rownames(cnts) <- paste0("gene_", 1:nrow(cnts))
colnames(cnts) <- paste0("cell_", 1:ncol(cnts))

# Run correspondence analysis
ca <- cacomp(obj = cnts, princ_coords = 3)

ca_3Dplot(ca)
```

---

ca_biplot *Plot of 2D CA projection of the data.*

Description

Plots the first 2 dimensions of the rows and columns in the same plot.

Usage

```
ca_biplot(
  obj,
  xdim = 1,
  ydim = 2,
  princ_coords = 1,
  row_labels = NULL,
  col_labels = NULL,
  type = "ggplot",
  col_metadata = NULL,
  row_metadata = NULL,
  show_all = TRUE,
  ...
)

## S4 method for signature 'cacomp'
ca_biplot(
  obj,
  xdim = 1,
  ydim = 2,
  princ_coords = 1,
  row_labels = NULL,
  col_labels = NULL,
```

```
  type = "ggplot",
  col_metadata = NULL,
  row_metadata = NULL,
  show_all = TRUE,
  ...
)

## S4 method for signature 'Seurat'
ca_biplot(
  obj,
  xdim = 1,
  ydim = 2,
  princ_coords = 1,
  row_labels = NULL,
  col_labels = NULL,
  type = "ggplot",
  col_metadata = NULL,
  row_metadata = NULL,
  show_all = TRUE,
  ...,
  assay = SeuratObject::DefaultAssay(obj),
  slot = "counts"
)

## S4 method for signature 'SingleCellExperiment'
ca_biplot(
  obj,
  xdim = 1,
  ydim = 2,
  princ_coords = 1,
  row_labels = NULL,
  col_labels = NULL,
  type = "ggplot",
  col_metadata = NULL,
  row_metadata = NULL,
  show_all = TRUE,
  ...,
  assay = "counts"
)
```

Arguments

obj	An object of class "cacomp" with the relevant standardized and principal coordinates calculated, or alternatively an object of class "Seurat" or "SingleCellExperiment" with a dim. reduction named "CA" saved.
xdim	Integer. The dimension for the x-axis. Default 1.
ydim	Integer. The dimension for the y-axis. Default 2.

princ_coords	Integer. If 1 then principal coordinates are used for the rows, if 2 for the columns. Default 1 (rows).
row_labels	Numeric vector. Indices for the rows for which a label should be added (label should be stored in rownames). Default NULL.
col_labels	Numeric vector. Indices for the columns for which a label should be added (label should be stored in colnames). Default NULL (no columns).
type	String. Type of plot to draw. Either "ggplot" or "plotly". Default "ggplot".
col_metadata	named vector of additional metadata to color points. The names of the elements in col_metadata should correspond to the column names in 'obj'. If NULL columns will be in a single color. Can also specify a metadata column for Seurat/SingleCellExperiment objects.
row_metadata	named vector of additional metadata to color points. The names of the elements in row_metadata should correspond to the row names in 'obj'. If NULL rows will be in a single color. Can also specify a metadata column for Seurat/SingleCellExperiment objects.
show_all	logical. If FALSE cells/genes that are not in col_metadata/ row_metadata are not plotted. If *_metadata is NULL, the cell or genes respectively will still be plotted.
...	Further arguments.
assay	SingleCellExperiment assay for recomputation
slot	Seurat assay slot from which to get matrix.

Details

Choosing type "plotly" will generate an interactive html plot with the package plotly. Type "ggplot" generates a static plot. Depending on whether 'princ_coords' is set to 1 or 2 either the principal coordinates of either the rows (1) or the columns (2) are chosen. For the other the standard coordinates are plotted (assymetric biplot). Labels for rows and columns should be stored in the row and column names respectively.

Value

Plot of class "plotly" or "ggplot".

Examples

```
# Simulate counts
cnts <- mapply(function(x){rpois(n = 500, lambda = x)},
               x = sample(1:100, 50, replace = TRUE))
rownames(cnts) <- paste0("gene_", 1:nrow(cnts))
colnames(cnts) <- paste0("cell_", 1:ncol(cnts))

# Run correspondence analysis
ca <- cacomp(obj = cnts, princ_coords = 3)

ca_biplot(ca)
```

ca_coords *Calculate correspondence analysis row and column coordinates.*

Description

'ca_coords' calculates the standardized and principal coordinates of the rows and columns in CA space.

Usage

```
ca_coords(caobj, dims = NULL, princ_coords = 3, princ_only = FALSE)
```

Arguments

caobj	A "cacomp" object as outputted from 'cacomp()'.
dims	Integer indicating the number of dimensions to use for the calculation of coordinates. All elements of caobj (where applicable) will be reduced to the given number of dimensions. Default NULL (keeps all dimensions).
princ_coords	Integer. Number indicating whether principal coordinates should be calculated for the rows (=1), columns (=2), both (=3) or none (=0). Default 3.
princ_only	Logical, whether only principal coordinates should be calculated. Or, in other words, whether the standardized coordinates are already calculated and stored in 'caobj'. Default 'FALSE'.

Details

Takes a "cacomp" object and calculates standardized and principal coordinates for the visualization of CA results in a biplot or to subsequently calculate coordinates in an Association Plot.

Value

Returns input object with coordinates added. std_coords_rows/std_coords_cols: Standardized coordinates of rows/columns. prin_coords_rows/prin_coords_cols: Principal coordinates of rows/columns.

Examples

```
# Simulate scRNAseq data.
cnts <- data.frame(cell_1 = rpois(10, 5),
                   cell_2 = rpois(10, 10),
                   cell_3 = rpois(10, 20))
rownames(cnts) <- paste0("gene_", 1:10)
cnts <- as.matrix(cnts)

# Run correspondence analysis.
ca <- cacomp(obj = cnts, princ_coords = 1)
ca <- ca_coords(ca, princ_coords = 3)
```

check_cacomp *Check if cacomp object was correctly created.*

Description

Checks if the slots in a cacomp object are of the correct size and whether they are coherent.

Usage

```
check_cacomp(object)
```

Arguments

object	A cacomp object.

Value

TRUE if it is a valid cacomp object. FALSE otherwise.

Examples

```
# Simulate scRNAseq data.
cnts <- data.frame(cell_1 = rpois(10, 5),
                   cell_2 = rpois(10, 10),
                   cell_3 = rpois(10, 20))
rownames(cnts) <- paste0("gene_", 1:10)
cnts <- as.matrix(cnts)

# Run correspondence analysis.
ca <- cacomp(obj = cnts, princ_coords = 3, top = 5)

check_cacomp(ca)
```

clip_residuals *Perform clipping of residuals*

Description

Clips Pearson or negative-binomial residuals above or below a determined value. For Pearson (Poisson) residuals it is set by default for 1, for NB at sqrt(n).

Usage

```
clip_residuals(S, cutoff = sqrt(ncol(S)))
```

Arguments

S	Matrix of residuals.
cutoff	Value above/below which clipping should happen.

Value

Matrix of clipped residuals.

References

Lause, J., Berens, P. & Kobak, D. Analytic Pearson residuals for normalization of single-cell RNA-seq UMI data. Genome Biol 22, 258 (2021). https://doi.org/10.1186/s13059-021-02451-7

---

comp_ft_residuals *Compute Freeman-Tukey residuals*

Description

Computes Freeman-Tukey residuals

Usage

```
comp_ft_residuals(mat)
```

Arguments

mat	A numerical matrix or coercible to one by ‘as.matrix()‘. Should have row and column names.

Value

A named list. The elements are:

- "S": standardized residual matrix.
- "tot": grand total of the original matrix.
- "rowm": row masses.
- "colm": column masses.

`comp_NB_residuals` *Compute Negative-Binomial residuals*

Description

Computes the residuals based on the negative binomial model. By default a theta of 100 is used to capture technical variation.

Usage

```
comp_NB_residuals(mat, theta = 100, clip = FALSE, cutoff = NULL, freq = TRUE)
```

Arguments

`mat`	A numerical matrix or coercible to one by 'as.matrix()'. Should have row and column names.
`theta`	Overdispersion parameter. By default set to 100 as described in Lause and Berens, 2021 (see references).
`clip`	logical. Whether residuals should be clipped if they are higher/lower than a specified cutoff
`cutoff`	numeric. Residuals that are larger than cutoff or lower than -cutoff are clipped to cutoff.
`freq`	logical. Whether a table of frequencies (as used in CA) should be used.

Value

A named list. The elements are:

- "S": standardized residual matrix.
- "tot": grand total of the original matrix.
- "rowm": row masses.
- "colm": column masses.

References

Lause, J., Berens, P. & Kobak, D. Analytic Pearson residuals for normalization of single-cell RNA-seq UMI data. Genome Biol 22, 258 (2021). https://doi.org/10.1186/s13059-021-02451-7

`comp_std_residuals` *Compute Standardized Residuals*

Description

'comp_std_residuals' computes the standardized residual matrix S based on the Poisson model, which is the basis for correspondence analysis and serves as input for singular value decomposition (SVD).

Usage

```
comp_std_residuals(mat, clip = FALSE, cutoff = NULL)
```

Arguments

`mat`	A numerical matrix or coercible to one by 'as.matrix()'. Should have row and column names.
`clip`	logical. Whether residuals should be clipped if they are higher/lower than a specified cutoff
`cutoff`	numeric. Residuals that are larger than cutoff or lower than -cutoff are clipped to cutoff.

Details

Calculates standardized residual matrix S from the proportion matrix P and the expected values E according to $S = \frac{(P-E)}{sqrt(E)}$.

Value

A named list. The elements are:

- "S": standardized residual matrix.
- "tot": grand total of the original matrix.
- "rowm": row masses.
- "colm": column masses.

elbow_method *Runs elbow method*

Description

Helper function for pick_dims() to run the elbow method.

Usage

```
elbow_method(obj, mat, reps, python = FALSE, return_plot = FALSE)
```

Arguments

obj	A "cacomp" object as outputted from ‘cacomp()‘
mat	A numeric matrix. For sequencing a count matrix, gene expression values with genes in rows and samples/cells in columns. Should contain row and column names.
reps	Integer. Number of permutations to perform when choosing "elbow_rule".
python	A logical value indicating whether to use singular value decomposition from the python package torch. This implementation dramatically speeds up computation compared to ‘svd()‘ in R.
return_plot	TRUE/FALSE. Whether a plot should be returned when choosing "elbow_rule".

Value

‘elbow_method‘ (for ‘return_plot=TRUE‘) returns a list with two elements: "dims" contains the number of dimensions and "plot" a ggplot. if ‘return_plot=TRUE‘ it just returns the number of picked dimensions.

References

Ciampi, Antonio, González Marcos, Ana and Castejón Limas, Manuel.
Correspondence analysis and 2-way clustering. (2005), SORT 29(1).

Examples

```
# Get example data from Seurat
library(SeuratObject)
set.seed(2358)
cnts <- as.matrix(SeuratObject::LayerData(pbmc_small,
                                          assay = "RNA",
                                          layer = "data"))
# Run correspondence analysis.
ca <- cacomp(obj = cnts)

# pick dimensions with the elbow rule. Returns list.
pd <- pick_dims(obj = ca,
```

```
                mat = cnts,
                method = "elbow_rule",
                return_plot = TRUE,
                reps = 10)
pd$plot
ca_sub <- subset_dims(ca, dims = pd$dims)
```

`inertia_rows` *Find most variable rows*

Description

Calculates the contributing inertia of each row which is defined as sum of squares of pearson residuals and selects the rows with the largested inertias, e.g. 5,000.

Usage

```
inertia_rows(mat, top = 5000, ...)
```

Arguments

`mat`	A matrix with genes in rows and cells in columns.
`top`	Number of genes to select.
`...`	Further arguments for 'comp_std_residuals'

Value

Returns a matrix, which consists of the top variable rows of mat.

`is.empty` *Helper function to check if object is empty.*

Description

Helper function to check if object is empty.

Usage

```
is.empty(x)
```

Arguments

`x`	object

Value

TRUE if x has length 0 and is not NULL. FALSE otherwise

`permutation_cutoff` *Calculates permuted association plot coordinates*

Description

Calculates matrix of apl coordinates when permuting the original data.

Usage

```
permutation_cutoff(
  caobj,
  mat,
  group = caobj@group,
  dims = caobj@dims,
  reps = 10,
  store_perm = FALSE,
  python = TRUE
)
```

Arguments

`caobj`	A "cacomp" object with principal row coordinates and standardized column coordinates calculated.
`mat`	A numeric matrix. For sequencing a count matrix, gene expression values with genes in rows and samples/cells in columns. Should contain row and column names.
`group`	Vector of indices of the columns to calculate centroid/x-axis direction.
`dims`	Integer. Number of CA dimensions to retain. Needs to be the same as in caobj!
`reps`	Integer. Number of permutations to perform.
`store_perm`	Logical. Whether permuted data should be stored in the CA object. This implementation dramatically speeds up computation compared to ‘svd()‘ in R.
`python`	DEPRACTED. A logical value indicating whether to use singular-value decomposition from the python package torch.

Value

List with permuted apl coordinates ("apl_perm") and, a list of saved ca components ("saved_ca") that allow for quick recomputation of the CA results. For random_direction_cutoff this saved_ca is empty.

pick_dims *Compute statistics to help choose the number of dimensions*

Description

Allow the user to choose from 4 different methods ("avg_inertia", "maj_inertia", "scree_plot" and "elbow_rule") to estimate the number of dimensions that best represent the data.

Usage

```
pick_dims(
  obj,
  mat = NULL,
  method = "scree_plot",
  reps = 3,
  python = FALSE,
  return_plot = FALSE,
  ...
)

## S4 method for signature 'cacomp'
pick_dims(
  obj,
  mat = NULL,
  method = "scree_plot",
  reps = 3,
  python = FALSE,
  return_plot = FALSE,
  ...
)

## S4 method for signature 'Seurat'
pick_dims(
  obj,
  mat = NULL,
  method = "scree_plot",
  reps = 3,
  python = FALSE,
  return_plot = FALSE,
  ...,
  assay = SeuratObject::DefaultAssay(obj),
  slot = "counts"
)

## S4 method for signature 'SingleCellExperiment'
pick_dims(
  obj,
```

```
  mat = NULL,
  method = "scree_plot",
  reps = 3,
  python = FALSE,
  return_plot = FALSE,
  ...,
  assay = "counts"
)
```

Arguments

obj	A "cacomp" object as outputted from `cacomp()`, a "Seurat" object with a "CA" DimReduc object stored, or a "SingleCellExperiment" object with a "CA" dim. reduction stored.
mat	A numeric matrix. For sequencing a count matrix, gene expression values with genes in rows and samples/cells in columns. Should contain row and column names.
method	String. Either "scree_plot", "avg_inertia", "maj_inertia" or "elbow_rule" (see Details section). Default "scree_plot".
reps	Integer. Number of permutations to perform when choosing "elbow_rule". Default 3.
python	DEPRACTED. A logical value indicating whether to use singular value decomposition from the python package torch. This implementation dramatically speeds up computation compared to `svd()` in R.
return_plot	TRUE/FALSE. Whether a plot should be returned when choosing "elbow_rule". Default FALSE.
...	Arguments forwarded to methods.
assay	Character. The assay from which to extract the count matrix for SVD, e.g. "RNA" for Seurat objects or "counts"/"logcounts" for SingleCellExperiments.
slot	Character. Data slot of the Seurat assay. E.g. "data" or "counts". Default "counts".

Details

- "avg_inertia" calculates the number of dimensions in which the inertia is above the average inertia.
- "maj_inertia" calculates the number of dimensions in which cumulatively explain up to 80% of the total inertia.
- "scree_plot" plots a scree plot.
- "elbow_rule" formalization of the commonly used elbow rule. Permutes the rows for each column and reruns `cacomp()` for a total of `reps` times. The number of relevant dimensions is obtained from the point where the line for the explained inertia of the permuted data intersects with the actual data.

Value

For `avg_inertia`, `maj_inertia` and `elbow_rule` (when `return_plot=FALSE`) returns an integer, indicating the suggested number of dimensions to use.

- `scree_plot` returns a ggplot object.
- `elbow_rule` (for `return_plot=TRUE`) returns a list with two elements: "dims" contains the number of dimensions and "plot" a ggplot.

Examples

```
# Simulate counts
cnts <- mapply(function(x){rpois(n = 500, lambda = x)},
               x = sample(1:20, 50, replace = TRUE))
rownames(cnts) <- paste0("gene_", 1:nrow(cnts))
colnames(cnts) <- paste0("cell_", 1:ncol(cnts))

# Run correspondence analysis.
ca <- cacomp(obj = cnts)

# pick dimensions with the elbow rule. Returns list.

set.seed(2358)
pd <- pick_dims(obj = ca,
                mat = cnts,
                method = "elbow_rule",
                return_plot = TRUE,
                reps = 10)
pd$plot
ca_sub <- subset_dims(ca, dims = pd$dims)

# pick dimensions which explain cumulatively >80% of total inertia.
# Returns vector.
pd <- pick_dims(obj = ca,
                method = "maj_inertia")
ca_sub <- subset_dims(ca, dims = pd)

################################
# pick_dims for Seurat objects #
################################
library(SeuratObject)
set.seed(1234)

# Simulate counts
cnts <- mapply(function(x){rpois(n = 500, lambda = x)},
               x = sample(1:20, 50, replace = TRUE))
rownames(cnts) <- paste0("gene_", 1:nrow(cnts))
colnames(cnts) <- paste0("cell_", 1:ncol(cnts))

# Create Seurat object
seu <- CreateSeuratObject(counts = cnts)

# run CA and save in dim. reduction slot.
```

```
seu <- cacomp(seu, return_input = TRUE, assay = "RNA", slot = "counts")

# pick dimensions
pd <- pick_dims(obj = seu,
                method = "maj_inertia",
                assay = "RNA",
                slot = "counts")

#############################################
# pick_dims for SingleCellExperiment objects #
#############################################
library(SingleCellExperiment)
set.seed(1234)

# Simulate counts
cnts <- mapply(function(x){rpois(n = 500, lambda = x)},
               x = sample(1:20, 50, replace = TRUE))
rownames(cnts) <- paste0("gene_", 1:nrow(cnts))
colnames(cnts) <- paste0("cell_", 1:ncol(cnts))

# Create SingleCellExperiment object
sce <- SingleCellExperiment(assays=list(counts=cnts))

# run CA and save in dim. reduction slot.
sce <- cacomp(sce, return_input = TRUE, assay = "counts")

# pick dimensions
pd <- pick_dims(obj = sce,
                method = "maj_inertia",
                assay = "counts")
```

`plot_enrichment` *Generates plot for results from apl_topGO*

Description

Plots the results from the data frame generated via apl_topGO.

Usage

```
plot_enrichment(genenr, ntop = 10)
```

Arguments

`genenr`	data.frame. gene enrichment results table.
`ntop`	numeric. Number of elements to plot.

Value

Returns a ggplot plot.

Examples

```
library(SeuratObject)
set.seed(1234)
cnts <- SeuratObject::LayerData(pbmc_small, assay = "RNA", layer = "counts")
cnts <- as.matrix(cnts)

# Run CA on example from Seurat

ca <- cacomp(pbmc_small,
             princ_coords = 3,
             return_input = FALSE,
             assay = "RNA",
             slot = "counts")

grp <- which(Idents(pbmc_small) == 2)
ca <- apl_coords(ca, group = grp)
ca <- apl_score(ca,
                mat = cnts)

enr <- apl_topGO(ca,
                 ontology = "BP",
                 organism = "hs")

plot_enrichment(enr)
```

---

random_direction_cutoff

Random direction association plot coordinates

Description

Calculates matrix of apl coordinates for random directions

Usage

```
random_direction_cutoff(caobj, dims = caobj@dims, reps = 100)
```

Arguments

caobj	A "cacomp" object with principal row coordinates and standardized column coordinates calculated.
dims	Integer. Number of CA dimensions to retain. Needs to be the same as in caobj!
reps	Integer. Number of permutations to perform.

Value

List with permuted apl coordinates ("apl_perm") and, a list of saved ca components ("saved_ca") that allow for quick recomputation of the CA results. For random_direction_cutoff this saved_ca is empty.

recompute *Recompute missing values of cacomp object.*

Description

The caobj needs to have the std_coords_cols, the prin_coords_rows and D calculated. From this the remainder will be calculated. Future updates might extend this functionality.

Usage

```
recompute(calist, mat, ...)
```

Arguments

calist	A list with std_coords_cols, the prin_coords_rows and D.
mat	A matrix from which the cacomp object is derived from.
...	Further arguments forwarded to cacomp.

Value

A cacomp object with additional calculated row_masses, col_masses, std_coords_rows, U and V.

rm_zeros *removes 0-only rows and columns in a matrix.*

Description

removes 0-only rows and columns in a matrix.

Usage

```
rm_zeros(obj)
```

Arguments

obj	A matrix.

Value

Input matrix with rows & columns consisting of only 0 removed.

run_APL *Compute and plot Association Plot*

Description

Computes singular value decomposition and coordinates for the Association Plot.

runAPL.SingleCellExperiment: Computes singular value decomposition and coordinates for the Association Plot from SingleCellExperiment objects with reducedDim(obj, "CA") slot (optional).

runAPL.Seurat: Computes singular value decomposition and coordinates for the Association Plot from Seurat objects, optionally with a DimReduc Object in the "CA" slot.

Usage

```
run_APL(
  obj,
  group,
  caobj = NULL,
  dims = NULL,
  nrow = 10,
  top = 5000,
  clip = FALSE,
  score = TRUE,
  score_method = "permutation",
  mark_rows = NULL,
  mark_cols = NULL,
  reps = 3,
  python = FALSE,
  row_labs = TRUE,
  col_labs = TRUE,
  type = "plotly",
  show_cols = FALSE,
  show_rows = TRUE,
  score_cutoff = 0,
  score_color = "rainbow",
  pd_method = "elbow_rule",
  pd_reps = 1,
  pd_use = TRUE
)

runAPL(
  obj,
  group,
  caobj = NULL,
  dims = NULL,
  nrow = 10,
  top = 5000,
```

```
  clip = FALSE,
  score = TRUE,
  score_method = "permutation",
  mark_rows = NULL,
  mark_cols = caobj@group,
  reps = 3,
  python = FALSE,
  row_labs = TRUE,
  col_labs = TRUE,
  type = "plotly",
  show_cols = FALSE,
  show_rows = TRUE,
  score_cutoff = 0,
  score_color = "rainbow",
  pd_method = "elbow_rule",
  pd_reps = 1,
  pd_use = TRUE,
  ...
)

## S4 method for signature 'matrix'
runAPL(
  obj,
  group,
  caobj = NULL,
  dims = NULL,
  nrow = 10,
  top = 5000,
  clip = FALSE,
  score = TRUE,
  score_method = "permutation",
  mark_rows = NULL,
  mark_cols = NULL,
  reps = 3,
  python = FALSE,
  row_labs = TRUE,
  col_labs = TRUE,
  type = "plotly",
  show_cols = FALSE,
  show_rows = TRUE,
  score_cutoff = 0,
  score_color = "rainbow",
  pd_method = "elbow_rule",
  pd_reps = 1,
  pd_use = TRUE,
  ...
)
```

```
## S4 method for signature 'SingleCellExperiment'
runAPL(
  obj,
  group,
  caobj = NULL,
  dims = NULL,
  nrow = 10,
  top = 5000,
  clip = FALSE,
  score = TRUE,
  score_method = "permutation",
  mark_rows = NULL,
  mark_cols = NULL,
  reps = 3,
  python = FALSE,
  row_labs = TRUE,
  col_labs = TRUE,
  type = "plotly",
  show_cols = FALSE,
  show_rows = TRUE,
  score_cutoff = 0,
  score_color = "rainbow",
  pd_method = "elbow_rule",
  pd_reps = 1,
  pd_use = TRUE,
  ...,
  assay = "counts"
)

## S4 method for signature 'Seurat'
runAPL(
  obj,
  group,
  caobj = NULL,
  dims = NULL,
  nrow = 10,
  top = 5000,
  clip = FALSE,
  score = TRUE,
  score_method = "permutation",
  mark_rows = NULL,
  mark_cols = NULL,
  reps = 3,
  python = FALSE,
  row_labs = TRUE,
  col_labs = TRUE,
  type = "plotly",
  show_cols = FALSE,
```

```
  show_rows = TRUE,
  score_cutoff = 0,
  score_color = "rainbow",
  pd_method = "elbow_rule",
  pd_reps = 1,
  pd_use = TRUE,
  ...,
  assay = SeuratObject::DefaultAssay(obj),
  slot = "counts"
)

## S4 method for signature 'dgCMatrix'
runAPL(
  obj,
  group,
  caobj = NULL,
  dims = NULL,
  nrow = 10,
  top = 5000,
  clip = FALSE,
  score = TRUE,
  score_method = "permutation",
  mark_rows = NULL,
  mark_cols = NULL,
  reps = 3,
  python = FALSE,
  row_labs = TRUE,
  col_labs = TRUE,
  type = "plotly",
  show_cols = FALSE,
  show_rows = TRUE,
  score_cutoff = 0,
  score_color = "rainbow",
  pd_method = "elbow_rule",
  pd_reps = 1,
  pd_use = TRUE,
  ...
)
```

Arguments

obj	A numeric matrix. For sequencing usually a count matrix, gene expression values with genes in rows and samples/cells in columns. Should contain row and column names.
group	Numeric/Character. Vector of indices or column names of the columns to calculate centroid/x-axis direction.
caobj	A "cacomp" object as outputted from ‘cacomp()‘. If not supplied will be calculated. Default NULL.

dims	Integer. Number of CA dimensions to retain. If NULL: (0.2 * min(nrow(A), ncol(A)) - 1).
nrow	Integer. The top nrow scored row labels will be added to the plot if score = TRUE. Default 10.
top	Integer. Number of most variable rows to retain. Set NULL to keep all.
clip	logical. Whether residuals should be clipped if they are higher/lower than a specified cutoff
score	Logical. Whether rows should be scored and ranked. Ignored when a vector is supplied to mark_rows. Default TRUE.
score_method	Method to calculate the cutoff. Either "random" for random direction method or "permutation" for the permutation method.
mark_rows	Character vector. Names of rows that should be highlighted in the plot. If not NULL, score is ignored. Default NULL.
mark_cols	Character vector. Names of cols that should be highlighted in the plot.
reps	Integer. Number of permutations during scoring. Default 3.
python	DEPRACTED. A logical value indicating whether to use singular-value decomposition from the python package torch. This implementation dramatically speeds up computation compared to ‘svd()‘ in R when calculating the full SVD. This parameter only works when dims==NULL or dims==rank(mat), where caculating a full SVD is demanded.
row_labs	Logical. Whether labels for rows indicated by rows_idx should be labeled with text. Default TRUE.
col_labs	Logical. Whether labels for columns indicated by cols_idx shouls be labeled with text. Default FALSE.
type	"ggplot"/"plotly". For a static plot a string "ggplot", for an interactive plot "plotly". Default "ggplot".
show_cols	Logical. Whether column points should be plotted.
show_rows	Logical. Whether row points should be plotted.
score_cutoff	Numeric. Rows (genes) with a score >= score_cutoff will be colored according to their score if show_score = TRUE.
score_color	Either "rainbow" or "viridis".
pd_method	Which method to use for pick_dims (pick_dims).
pd_reps	Number of repetitions performed when using "elbow_rule" in ‘pick_dims‘. (pick_dims)
pd_use	Whether to use ‘pick_dims‘ (pick_dims) to determine the number of dimensions. Ignored when ‘dims‘ is set by the user.
...	Arguments forwarded to methods.
assay	Character. The assay from which extract the count matrix for SVD, e.g. "RNA" for Seurat objects or "counts"/"logcounts" for SingleCellExperiments.
slot	character. The Seurat assay slot from which to extract the count matrix.

Details

The function is a wrapper that calls ‘cacomp()‘, ‘apl_coords()‘, ‘apl_score()‘ and finally ‘apl()‘ for ease of use. The chosen defaults are most useful for genomics experiments, but for more fine grained control the functions can be also run individually for the same results. If score = FALSE, nrow and reps are ignored. If mark_rows is not NULL score is treated as if FALSE.

Value

Association Plot (plotly object).

References

Association Plots: Visualizing associations in high-dimensional correspondence analysis biplots
Elzbieta Gralinska, Martin Vingron
bioRxiv 2020.10.23.352096; doi: https://doi.org/10.1101/2020.10.23.352096

Examples

```
set.seed(1234)

# Simulate counts
cnts <- mapply(function(x){rpois(n = 500, lambda = x)},
               x = sample(1:100, 50, replace = TRUE))
rownames(cnts) <- paste0("gene_", 1:nrow(cnts))
colnames(cnts) <- paste0("cell_", 1:ncol(cnts))

# (nonsensical) APL
APL:::run_APL(obj = cnts,
       group = 1:10,
       dims = 10,
       top = 500,
       score = TRUE,
       show_cols = TRUE,
       type = "ggplot")
set.seed(1234)

# Simulate counts
cnts <- mapply(function(x){rpois(n = 500, lambda = x)},
               x = sample(1:100, 50, replace = TRUE))
rownames(cnts) <- paste0("gene_", 1:nrow(cnts))
colnames(cnts) <- paste0("cell_", 1:ncol(cnts))

# (nonsensical) APL
runAPL(obj = cnts,
       group = 1:10,
       dims = 10,
       top = 500,
       score = TRUE,
       show_cols = TRUE,
       type = "ggplot")
```

```
########################
# SingleCellExperiment #
########################
library(SingleCellExperiment)
set.seed(1234)

# Simulate counts
cnts <- mapply(function(x){rpois(n = 500, lambda = x)},
               x = sample(1:100, 50, replace = TRUE))
rownames(cnts) <- paste0("gene_", 1:nrow(cnts))
colnames(cnts) <- paste0("cell_", 1:ncol(cnts))

sce <- SingleCellExperiment(assays=list(counts=cnts))

# (nonsensical) APL
runAPL(obj = sce,
       group = 1:10,
       dims = 10,
       top = 500,
       score = TRUE,
       show_cols = TRUE,
       type = "ggplot",
       assay = "counts")

############
# Seurat  #
############
library(SeuratObject)
set.seed(1234)

# Simulate counts
cnts <- mapply(function(x){rpois(n = 500, lambda = x)},
               x = sample(1:100, 50, replace = TRUE))
rownames(cnts) <- paste0("gene_", 1:nrow(cnts))
colnames(cnts) <- paste0("cell_", 1:ncol(cnts))

seu <- CreateSeuratObject(counts = cnts)

# (nonsensical) APL
runAPL(obj = seu,
       group = 1:10,
       dims = 10,
       top = 500,
       score = TRUE,
       show_cols = TRUE,
       type = "ggplot",
       assay = "RNA",
       slot = "counts")
set.seed(1234)

# Simulate counts
cnts <- mapply(function(x){rpois(n = 500, lambda = x)},
```

```
               x = sample(seq(0.01,0.1,by=0.01), 50, replace = TRUE))
rownames(cnts) <- paste0("gene_", 1:nrow(cnts))
colnames(cnts) <- paste0("cell_", 1:ncol(cnts))
cnts <- Matrix::Matrix(cnts)

# (nonsensical) APL
runAPL(obj = cnts,
       group = 1:10,
       dims = 10,
       top = 500,
       score = TRUE,
       show_cols = TRUE,
       type = "ggplot")
```

`run_cacomp` *Internal function for 'cacomp'*

Description

'run_cacomp' performs correspondence analysis on a matrix and returns the transformed data.

Usage

```
run_cacomp(
  obj,
  coords = TRUE,
  princ_coords = 3,
  python = FALSE,
  dims = 100,
  top = 5000,
  inertia = TRUE,
  rm_zeros = TRUE,
  residuals = "pearson",
  cutoff = NULL,
  clip = FALSE,
  ...
)
```

Arguments

`obj`	A numeric matrix or Seurat/SingleCellExperiment object. For sequencing a count matrix, gene expression values with genes in rows and samples/cells in columns. Should contain row and column names.
`coords`	Logical. Indicates whether CA standard coordinates should be calculated.
`princ_coords`	Integer. Number indicating whether principal coordinates should be calculated for the rows (=1), columns (=2), both (=3) or none (=0).

python	DEPRACTED. A logical value indicating whether to use singular-value decomposition from the python package torch. This implementation dramatically speeds up computation compared to ‘svd()‘ in R when calculating the full SVD. This parameter only works when dims==NULL or dims==rank(mat), where caculating a full SVD is demanded.
dims	Integer. Number of CA dimensions to retain. If NULL: (0.2 * min(nrow(A), ncol(A)) - 1).
top	Integer. Number of most variable rows to retain. Set NULL to keep all.
inertia	Logical. Whether total, row and column inertias should be calculated and returned.
rm_zeros	Logical. Whether rows & cols containing only 0s should be removed. Keeping zero only rows/cols might lead to unexpected results.
residuals	character string. Specifies which kind of residuals should be calculated. Can be "pearson" (default), "freemantukey" or "NB" for negative-binomial.
cutoff	numeric. Residuals that are larger than cutoff or lower than -cutoff are clipped to cutoff.
clip	logical. Whether residuals should be clipped if they are higher/lower than a specified cutoff
...	Arguments forwarded to methods.

Details

The calculation is performed according to the work of Michael Greenacre. When working with large matrices, CA coordinates and principal coordinates should only be computed when needed to save computational time.

Value

Returns a named list of class "cacomp" with components U, V and D: The results from the SVD. row_masses and col_masses: Row and columns masses. top_rows: How many of the most variable rows/genes were retained for the analysis. tot_inertia, row_inertia and col_inertia: Only if inertia = TRUE. Total, row and column inertia respectively.

References

Greenacre, M. Correspondence Analysis in Practice, Third Edition, 2017.

---

scree_plot *Scree Plot*

Description

Plots a scree plot.

Usage

```
scree_plot(df)
```

Arguments

df	A data frame with columns "dims" and "inertia".

Value

Returns a ggplot object.

---

show.cacomp *Prints cacomp object*

Description

Provides more user friendly printing of cacomp objects.

Usage

```
show.cacomp(object)

## S4 method for signature 'cacomp'
show(object)
```

Arguments

object	cacomp object to print

Value

prints summary information about cacomp object.

Examples

```
# Simulate scRNAseq data.
cnts <- data.frame(cell_1 = rpois(10, 5),
                   cell_2 = rpois(10, 10),
                   cell_3 = rpois(10, 20))
rownames(cnts) <- paste0("gene_", 1:10)
cnts <- as.matrix(cnts)

# Run correspondence analysis.
ca <- cacomp(obj = cnts, princ_coords = 3, top = 5)

ca
```

subset_dims *Subset dimensions of a caobj*

Description

Subsets the dimensions according to user input.

Usage

```
subset_dims(caobj, dims)
```

Arguments

caobj	A caobj.
dims	Integer. Number of dimensions.

Value

Returns caobj.

Examples

```
# Simulate scRNAseq data.
cnts <- data.frame(cell_1 = rpois(10, 5),
                   cell_2 = rpois(10, 10),
                   cell_3 = rpois(10, 20))
rownames(cnts) <- paste0("gene_", 1:10)
cnts <- as.matrix(cnts)

# Run correspondence analysis.
ca <- cacomp(cnts)
ca <- subset_dims(ca, 2)
```

var_rows *Find most variable rows*

Description

Calculates the variance of the chi-square component matrix and selects the rows with the highest variance, e.g. 5,000.

Usage

```
var_rows(mat, residuals = "pearson", top = 5000, ...)
```

Arguments

mat	A numeric matrix. For sequencing a count matrix, gene expression values with genes in rows and samples/cells in columns. Should contain row and column names.
residuals	character string. Specifies which kind of residuals should be calculated. Can be "pearson" (default), "freemantukey" or "NB" for negative-binomial.
top	Integer. Number of most variable rows to retain. Default 5000.
...	Further arguments for ‘calc_residuals‘.

Value

Returns a matrix, which consists of the top variable rows of mat.

Examples

```
set.seed(1234)

# Simulate counts
cnts <- mapply(function(x){rpois(n = 500, lambda = x)},
               x = sample(1:20, 50, replace = TRUE))
rownames(cnts) <- paste0("gene_", 1:nrow(cnts))
colnames(cnts) <- paste0("cell_", 1:ncol(cnts))

# Choose top 5000 most variable genes
cnts <- var_rows(mat = cnts, top = 5000)
```

%>% *Pipe operator*

Description

See magrittr::%>% for details.

Usage

```
lhs %>% rhs
```

Arguments

lhs	A value or the magrittr placeholder.
rhs	A function call using the magrittr semantics.

Value

magrittr::%>%

Examples

```
x <- 1:100
x %>% head()
```

Index

∗ **internal**
    %>%, 53
%>%, *53*, 53

apl, 3
apl_coords, 4
apl_ggplot, 6
apl_plotly, 7
apl_score, 8
apl_topGO, 10
as.cacomp, 12
as.cacomp,cacomp-method *(*as.cacomp*)*, 12
as.cacomp,list-method *(*as.cacomp*)*, 12
as.cacomp,Seurat-method *(*as.cacomp*)*, 12
as.cacomp,SingleCellExperiment-method *(*as.cacomp*)*, 12
as.list,cacomp-method, 14

ca_3Dplot, 23
ca_3Dplot,cacomp-method *(*ca_3Dplot*)*, 23
ca_3Dplot,Seurat-method *(*ca_3Dplot*)*, 23
ca_3Dplot,SingleCellExperiment-method *(*ca_3Dplot*)*, 23
ca_biplot, 25
ca_biplot,cacomp-method *(*ca_biplot*)*, 25
ca_biplot,Seurat-method *(*ca_biplot*)*, 25
ca_biplot,SingleCellExperiment-method *(*ca_biplot*)*, 25
ca_coords, 28
cacomp, 15
cacomp,dgCMatrix-method *(*cacomp*)*, 15
cacomp,matrix-method *(*cacomp*)*, 15
cacomp,Seurat-method *(*cacomp*)*, 15
cacomp,SingleCellExperiment-method *(*cacomp*)*, 15
cacomp-class, 19
cacomp_names, 21
cacomp_slot, 21
calc_residuals, 22
check_cacomp, 29
clip_residuals, 29
comp_ft_residuals, 30
comp_NB_residuals, 31
comp_std_residuals, 32

elbow_method, 33

inertia_rows, 34
is.empty, 34

new_cacomp *(*cacomp-class*)*, 19

permutation_cutoff, 35
pick_dims, 36, *46*
pick_dims,cacomp-method *(*pick_dims*)*, 36
pick_dims,Seurat-method *(*pick_dims*)*, 36
pick_dims,SingleCellExperiment-method *(*pick_dims*)*, 36
plot_enrichment, 39

random_direction_cutoff, 40
recompute, 41
rm_zeros, 41
run_APL, 42
run_cacomp, 49
runAPL *(*run_APL*)*, 42
runAPL,dgCMatrix-method *(*run_APL*)*, 42
runAPL,matrix-method *(*run_APL*)*, 42
runAPL,Seurat-method *(*run_APL*)*, 42
runAPL,SingleCellExperiment-method *(*run_APL*)*, 42

scree_plot, 50
show,cacomp-method *(*show.cacomp*)*, 51
show.cacomp, 51
subset_dims, 52

var_rows, 52