Package 'scCNAutils'

March 23, 2019

Title Functions to analyze copy number aberrations in single-cell data

Version 0.0.0.9000

Description Functions to analyze copy number aberrations in single-cell data. A bunch of scripts and workflows to read and analyze scRNA-seq data and look at CNA-oriented signal.

Depends R (¿= 3.4.4)

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Imports Matrix,
dplyr,
magrittr,
tidyr,
rlang,
parallel,
ggplot2,
data.table,
Rtsne,
FNN,
igraph,
RcppHMM,
GenomicRanges,
IRanges,
shiny,
rbokeh,
scales,
ggrepel,
Rcpp

Suggests testthat

RoxygenNote 6.1.1

LinkingTo Rcpp

R topics documented:

scCNAutils-package *scCNAutils*

Description

A cleaned up version of the functions I used to analyze copy number aberrations in single-cell data. Still in development.

Author(s)

Maintainer: Jean Monlong <jean.monlong@gmail.com>

annotate_cna *Annotate CNAs with gene information*

Description

Add columns with the names of the genes in the region, total and expressed genes only.

Usage

```
annotate_cna(cna_df, gene_info)
```

Arguments

cna_df	a data.frame with CNA calls. E.g. data.frames created by call_cna_* functions.
gene_info	a data.frame with gene information created by the gene_info function

Details

The subset of "expressed" genes is made out of genes with non-zero expression in at least 10 expression higher than 0.5.

Value

the input data.frame with two new columns with all/expressed genes.

Author(s)

Jean Monlong

auto_cna_call *Automated pipeline to call CNA*

Description

Automated pipeline to call CNA using metacells.

Usage

```
auto_cna_call(ge_df, comm_df, nb_metacells = 10, metacell_size = 3,
  multisamps = TRUE, trans_prob = 0.1, baseline_cells = NULL,
  baseline_communities = NULL, prefix = "scCNAutils_out",
  nb_cores = 1, chrs = c(1:22, "X", "Y"), bin_mean_exp = 3,
  z_wins_th = 3, smooth_wsize = 3)
```

Arguments

ge_df	normalized gene expression of all cells (e.g. output from norm_ge.
comm_df	a data.frame with community information, output from find_communities.
nb_metacells	the number of metacells per comunity.
metacell_size	the number of cells in a metacell.
multisamps	use the multi-sample version of the HMM segmentation? Default is TRUE. See details.
trans_prob	the transition probability for the HMM.
baseline_cells	cells to use as baseline.
baseline_communities	communities to use as baseline. Used if baseline.cells is NULL.
prefix	the prefix to use for the files created by this function (e.g. graphs).
nb_cores	the number of processors to use.
chrs	the chromosome names to keep. NULL to include all the chromosomes.
bin_mean_exp	the desired minimum mean expression in the bin.
z_wins_th	the threshold to winsorize Z-score. Default is 3
smooth_wsize	the window size for smoothing. Default is 3.

Details

Once the metacells are created there are two ways to call CNA. First, if multisamps=FALSE, to call CNA on each metacell and merge the result per community, keeping the information about how many metacell support the CNA. Second, if multisamps=TRUE (default), to run the HMM on all the metacells for a community. The multi-sample approach should be more robust.

The transition probability (trans_prob) is going to affect the HMM segmentation. Smaller values will create longer segments. One approach, often advocated by HMM aficionados, is to try different values and use the ones that gives the best results, for example based on the QC graphs (TODO). Another approach is to use a loose transition probability and then filter short segments ('length' column or 'pass.filter' column).

Value

a data.frame with CNAs

Author(s)

Jean Monlong

`auto_cna_signal` *Automated pipeline to compute CNA signal from scRNA expression*

Description

Goes from reading raw gene counts to CNA-level signal, tSNE and community detection.

Usage

```
auto_cna_signal(data, genes_coord, prefix = "scCNAutils_out",
  nb_cores = 1, pause_after_qc = FALSE, use_cache = TRUE,
  sample_names = NULL, info_df = NULL, max_mito_prop = 0.2,
  min_total_exp = 0, cells_sel = NULL, chrs = c(1:22, "X", "Y"),
  cell_cycle = NULL, bin_mean_exp = 3, rm_cv_quant = NULL,
  z_wins_th = 3, smooth_wsize = 3, cc_sd_th = 3, nb_pcs = 10,
  comm_k = 100, viz = c("tsne", "umap", "both"), tsne.seed = 999,
  rcpp = FALSE)
```

Arguments

`data`	a data.frame with gene expression or the path to the folder with the 'matrix.mtx', 'genes.tsv' and 'barcodes.tsv' files. A list if multiple samples.
`genes_coord`	either a file name or a data.frame with coordinates and gene names.
`prefix`	the prefix to use for the files created by this function (e.g. graphs).
`nb_cores`	the number of processors to use.
`pause_after_qc`	pause after the QC to pick custom QC thresholds.
`use_cache`	should intermediate files used and avoid redoing steps?
`sample_names`	the names of each sample. If NULL, tries to use data's names.
`info_df`	a data.frame with information about cells.
`max_mito_prop`	the maximum proportion of mitochondrial RNA.
`min_total_exp`	the minimum total cell expression
`cells_sel`	consider only these cells. Other cells filtered no matter what.
`chrs`	the chromosome names to keep. NULL to include all the chromosomes.
`cell_cycle`	if non-null, either a file or data.frame to compute cell cycle scores. See details.

bin_mean_exp	the desired minimum mean expression in the bin.
rm_cv_quant	the quantile threshold to remove CV outlier. Default NULL (i.e. not used).
z_wins_th	the threshold to winsorize Z-score. Default is 3
smooth_wsize	the window size for smoothing. Default is 3.
cc_sd_th	the number of SD used for the thresholds when defining cycling cells.
nb_pcs	the number of PCs used in the community detection or tSNE.
comm_k	the number of nearest neighbor for the KNN graph. Default 100.
viz	which method to use for visualization ('tsne', 'umap' or 'both'). Default is 'tsne'.
tsne.seed	the seed for the tSNE.
rcpp	use Rcpp function. Default is FALSE. More memory-efficient and faster when running on one core.

Value

a data.frame with QC, community and tSNE for each cell.

Author(s)

Jean Monlong

`binGenesC` *Bin genes*

Description

Bin genes

Usage

```
binGenesC(ge, bins)
```

Arguments

ge	an ordered gene expression matrix
bins	a vector with bin ids

bin_genes *Merge consecutive genes into expressed bins*

Description

Merge consecutive genes into expressed bins

Usage

```
bin_genes(ge_df, mean_exp = 3, nb_cores = 1, rcpp = FALSE)
```

Arguments

ge_df	the input gene expression with coordinate columns (chr, start, end) and then one column per cell.
mean_exp	the desired minimum mean expression in the bin.
nb_cores	the number of processors to use.
rcpp	use Rcpp function. Default is FALSE. More memory-efficient and faster when running on one core.

Value

a data.frame with bin expression.

Author(s)

Jean Monlong

call_cna *Call CNA*

Description

Calls CNA using a HMM approach.

Usage

```
call_cna(z_df, trans_prob = 1e-04, nb_cores = 1, mc_info = NULL)
```

Arguments

z_df	the Z-scores, from zscore.
trans_prob	the transition probability for the HMM.
nb_cores	the number of processor to use.
mc_info	the information about the metacells, if relevant. Default is NULL.

Value

a data.frame with the CNA calls.

Author(s)

Jean Monlong

`call_cna_multisamps` *Call CNA*

Description

Calls CNA using a HMM approach considering multiple samples at the same time.

Usage

```
call_cna_multisamps(z_df, mc_info, trans_prob = 1e-04, nb_cores = 1)
```

Arguments

`z_df`	the Z-scores, from `zscore`.
`mc_info`	the information about the metacells, if relevant. Default is NULL.
`trans_prob`	the transition probability for the HMM.
`nb_cores`	the number of processor to use.

Value

a data.frame with the CNA calls.

Author(s)

Jean Monlong

`convert_to_coord` *Convert gene symbols to coordinates*

Description

Convert the 'symbol' column (gene names) into three columns with gene coordinates 'chr', 'start' and 'end'.

Usage

```
convert_to_coord(ge_df, genes_coord, chrs = c(1:22, "X", "Y"),
  rm_dup = TRUE)
```

Arguments

ge_df	the data.frame with gene expression and one column 'symbol' with gene names.
genes_coord	either a file name or a data.frame with coordinates and gene names.
chrs	the chromosome names to keep. NULL to include all the chromosomes.
rm_dup	remove duplicated coordinates? Default is TRUE.

Details

If *genes_coord* is a filename, the file is expected to be a tab-delimited file with four columns: 'chr', 'start', 'end', 'symbol'. The order of the columns is not important.

The gene names in column 'symbol' should match the gene names in the input *ge_df*.

Value

a data.frame with columns 'chr', 'start', 'end' columns with genes coordinates (and still one column per barcode).

Author(s)

Jean Monlong

---

define_cycling_cells *Define cycling cells*

Description

Using cell cycle scores, identify cells that are cycling.

Usage

```
define_cycling_cells(qc_df, sd_th = 3)
```

Arguments

qc_df	the output data.frame from qc_cells (ran with a non-null *cell_cycle* parameter)
sd_th	the number of SD used for the thresholds.

Value

a list with

cells.noc	a vector with the names of non-cycling cells
graphs	a list of ggplot2 graphs

Author(s)

Jean Monlong

`find_communities` *Community detection*

Description

Build a KNN graph and run Louvain algorithm for community detection.

Usage

```
find_communities(pca_o, nb_pcs = 10, k = 100, gamma = 1, nreps = 1,
  nb_cores = 1)
```

Arguments

`pca_o`	the output of `run_pca`
`nb_pcs`	the number of PCs to use. Default 10.
`k`	the number of nearest neighbor for the KNN graph. Default 100.
`gamma`	a vector of gamma. Default 1.
`nreps`	the number of repetition for each gamma, Default 1.
`nb_cores`	the number of processors to use. Default is 1.

Value

a list with:

`comm`	a data.frame with two columns: 'cell' and 'community'.
`comm.all`	a matrix with communities for each gamma
`gamma`	the list of input gamma corresponding to each comm.all column.
`best.gamma`	the gamma resulting on the highest ARI mean
`ari.df`	data.frame with ARI stats for each gamma

Author(s)

Jean Monlong

gene_info *Gene information*

Description

Computes information at the gene level that can be used to annotate the results later (e.g. CNA calls).

Usage

```
gene_info(ge_df, genes_coord, subset_cells = 10000)
```

Arguments

ge_df	a data.frame with gene expression information across cells. E.g. after norm.ge.
genes_coord	either a file name or a data.frame with coordinates and gene names.
subset_cells	the maximum number of cells to use.

Details

If the input data.frame has a symbol column, 'genes_coord' is used to add the coordinates, later used to overlap with region/CNA calls. If the input data.frame has coordinates, 'genes_coord' is used to retrieve the gene name. In both cases a row in the input data.frame must be a gene, not a bin.

It's better to run this function after normalization (norm_ge but before binning (bin_genes).

Value

a data.frame with summary stats for each gene.

Author(s)

Jean Monlong

make_metacells *Make metacells*

Description

Randomly select cells in each community and merge them to create metacells with higher resolution.

Usage

```
make_metacells(ge_df, comm_df, nb_metacells = 10, metacell_size = 3,
  baseline_cells = NULL, nb_cores = 1)
```

Arguments

ge_df	normalized gene expression of all cells (e.g. output from norm_ge.
comm_df	a data.frame with community information, output from find_communities.
nb_metacells	the number of metacells per comunity.
metacell_size	the number of cells in a metacell.
baseline_cells	the cells to use for baseline communities.
nb_cores	the number of processor to use.

Value

a list with

ge	a data.frame with coordinates and gene expression for each metacell.
info	information about which metacell correspond to which community.
mc_cells	information about which cells were used for each metacell.

Author(s)

Jean Monlong

---

merge_samples *Merge expression of multiple samples*

Description

The expression of multiple samples are merged. New cell names are produced as SAMPLE_CELL.

Usage

```
merge_samples(ge_list, sample_names = NULL)
```

Arguments

ge_list	a list of ge_df (e.g. read from read_mtx).
sample_names	the names of each sample. If NULL, tries to use ge_list's names.

Value

a list with

ge	the merged gene expression data.frame
info	a data.frame with new and original cell names, and corresponding sample name

Author(s)

Jean Monlong

norm_ge *Normalize gene expression*

Description

The expression of each cell is normalized to account for depth differences.

Usage

```
norm_ge(ge_df, method = c("tmm", "total"), nb_cores = 1,
  rcpp = FALSE)
```

Arguments

ge_df	the input gene expression
method	the normalization method
nb_cores	the number of processors to use.
rcpp	use Rcpp function. Default is FALSE. More memory-efficient and faster when running on one core.

Value

a data.frame with the normalized expression.

Author(s)

Jean Monlong

plot_aneuploidy *Aneuploidy graph*

Description

Graphs showing the median expression in each chromosome for each community.

Usage

```
plot_aneuploidy(ge_df, comm_df = NULL, baseline_cells = NULL,
  baseline_communities = NULL, max_cells = 100, chrs_order = c(1:22,
  "X", "Y"))
```

Arguments

ge_df	a data.frame with gene expression (better if binned and normalized).
comm_df	a data.frame with community information for each cell.
baseline_cells	cells to use as baseline.
baseline_communities	the communities to use as baseline.
max_cells	the maximum number of cells to consider in the boxplot of each community. Default: 100.
chrs_order	order of the chromosomes in the graph.

Value

a list of ggplot2 object, one for each chromosome.

Author(s)

Jean Monlong

---

plot_cna *Heatmap of CNA*

Description

Heatmap of CNA

Usage

```
plot_cna(cna, chrs_order = c(1:22, "X", "Y"))
```

Arguments

cna	CNAs from call_cna.
chrs_order	order of the chromosomes in the graph.

Value

a ggplot2 graph

Author(s)

Jean Monlong

plot_communities *Community graphs*

Description

Graphs about the communities found by find_communities. For example the size of the communities or the distribution of QC metrics in each community.

Usage

```
plot_communities(comm_df, qc_df = NULL, info_df = NULL)
```

Arguments

comm_df	the output data.frame from find_communities
qc_df	a data.frame with QC metrics (output from qc_cells). Default is NULL (i.e. not used)
info_df	a data.frame with sample merge info (output from merge_samples). Default is NULL (i.e. not used)

Details

If the QC data.frame is provided, the distribution of QC metrics is shown to investigate if some communities are batch effects.

If multiple samples were merged (merge_samples), the proportion of cells from each sample of origin can be shown if the info_df data.frame is provided.

If qc_df and/or info_df are null but their columns present in comm_df, their corresponding graphs will be generated. Hence a merged version of comm_df, qc_df and info_df works (e.g. output of auto_cna_signal.

Value

a list of ggplot2 graphs.

Author(s)

Jean Monlong

Examples

```
## Not run:
ggp.l = plot_communities(comm_df, qc_df)

## Print first graph
ggpl.l[[1]]

## Customize ggplot
ggpl.l[[1]] + ggtitle('First graph about communities')
```

```
## End(Not run)
```

`plot_heatmap` *Heatmap of the CNA scores*

Description

Heatmap of the CNA scores

Usage

```
plot_heatmap(z_df, cells_df = NULL, nb_subsamp = 1000,
  hc.method = "ward.D", z_win = 3)
```

Arguments

`z_df`	the data.frame with Z-scores.
`cells_df`	a data.frame with cell information.
`nb_subsamp`	the number of cells to sub-sample. Default is 1000.
`hc.method`	the hierarchical clustering method to order cells.
`z_win`	threshold to winsorized Z scores in the color scale.

Value

a list of ggplot2 objects.

Author(s)

Jean Monlong

`plot_qc_cells` *QC graphs*

Description

QC graphs

Usage

```
plot_qc_cells(qc_df, info_df = NULL)
```

Arguments

`qc_df`	the output data.frame from `qc_cells`
`info_df`	a data.frame with sample merge info (output from `merge_samples`). Default is NULL (i.e. not used)

Value

a list of ggplots

Author(s)

Jean Monlong

Examples

```
## Not run:
ggp.l = plot_qc_cells(qc_df)

## Print first graph
ggpl.l[[1]]

## Customize ggplot
ggpl.l[[1]] + ggtitle('First QC graph')

## End(Not run)
```

---

plot_tsne *tSNE graphs*

Description

tSNE graphs colored according to QC metrics or sample labels.

Usage

```
plot_tsne(tsne_df, qc_df = NULL, comm_df = NULL, info_df = NULL)
```

Arguments

tsne_df	the output data.frame from run_tsne (columns: cell, tsne1, tsne2)
qc_df	a data.frame with QC metrics (output from qc_cells). Default is NULL (i.e. not used)
comm_df	a data.frame with communities (output from find_communities). Default is NULL (i.e. not used)
info_df	a data.frame with sample merge info (output from merge_samples).

Details

If the QC data.frame is provided, the distribution of QC metrics is shown to investigate if some communities are batch effects.

If multiple samples were merged (merge_samples), the points can be colored by sample of origin by providing the info_df data.frame.

If any qc_df/comm_df/info_df are null but their columns present in tsne_df, their corresponding graphs will be generated. Hence a merged version of tsne_df, comm_df, qc_df and info_df works (e.g. output of auto_cna_signal.

Value

a list of ggplot objects

Author(s)

Jean Monlong

Examples

```
## Not run:
ggp.l = plot_tsne(tsne_df, qc_df, comm_df)

## Print first graph
ggpl.l[[1]]

## Customize ggplot
ggpl.l[[1]] + ggtitle('First tSNE graph')

## End(Not run)
```

---

plot_umap *UMAP graphs*

Description

UMAP graphs colored according to QC metrics or sample labels.

Usage

```
plot_umap(umap_df, qc_df = NULL, comm_df = NULL, info_df = NULL)
```

Arguments

umap_df	the output data.frame from run_umap (columns: cell, umap1, umap2)
qc_df	a data.frame with QC metrics (output from qc_cells). Default is NULL (i.e. not used)
comm_df	a data.frame with communities (output from find_communities). Default is NULL (i.e. not used)
info_df	a data.frame with sample merge info (output from merge_samples).

Details

If the QC data.frame is provided, the distribution of QC metrics is shown to investigate if some communities are batch effects.

If multiple samples were merged (merge_samples), the points can be colored by sample of origin by providing the info_df data.frame.

If any qc_df/comm_df/info_df are null but their columns present in umap_df, their corresponding graphs will be generated. Hence a merged version of umap_df, comm_df, qc_df and info_df works (e.g. output of auto_cna_signal.

Value

a list of ggplot objects

Author(s)

Jean Monlong

Examples

```
## Not run:
ggp.l = plot_umap(umap_df, qc_df, comm_df)

## Print first graph
ggpl.l[[1]]

## Customize ggplot
ggpl.l[[1]] + ggtitle('First umap graph')

## End(Not run)
```

---

qc_cells *Compute quality control metrics for each cell*

Description

From raw gene expression, a few QC metrics are computed.

Usage

```
qc_cells(ge_df, cell_cycle = NULL)
```

Arguments

ge_df	the input gene expression with a 'symbol' column and then one column per cell.
cell_cycle	if non-null, either a file or data.frame to compute cell cycle scores. See details.

Details

If cell_cycle is provided it should be a data.frame (or a tsv file) with two columns: 'symbol' with gene names, and 'phase' with the cell cycle phase (e.g. either 'G1.S' or 'G2.M').

Value

a data.frame with qc metrics per cell.

Author(s)

Jean Monlong

qc_filter *Filter cells based on QC results*

Description

Filter cells based on QC results

Usage

```
qc_filter(ge_df, qc_df, max_mito_prop = 0.2, min_total_exp = 0,
  cells_sel = NULL)
```

Arguments

ge_df	the input gene expression with a 'symbol' column and then one column per cell.
qc_df	the output data.frame from qc_cells
max_mito_prop	the maximum proportion of mitochondrial RNA.
min_total_exp	the minimum total cell expression
cells_sel	consider only these cells. Other cells filtered no matter what.

Value

ge_df with only the cells that passed the filters

Author(s)

Jean Monlong

read_mtx *Read a trio of genes, barcodes and mtx files.*

Description

Read a trio of genes, barcodes and mtx files.

Usage

```
read_mtx(mtx_file = "matrix.mtx", genes_file = "genes.tsv",
  barcodes_file = "barcodes.tsv", path = ".", rm_dup = TRUE,
  genes_col = 2)
```

Arguments

mtx_file	the path to the mtx file
genes_file	the path to the genes file.
barcodes_file	the path to the barcodes file
path	the path to the folder containing the files
rm_dup	remove duplicated gene names? Default is TRUE.
genes_col	the column to use in genes_file. Default is 2.

Value

a data.frame with a 'symbol' column with gene names and one column per barcode.

Author(s)

Jean Monlong

---

rebin_cov *Re-bin coverage data*

Description

The new bins are overlapped with the regions in cov_df to compute a weight. The coverage in the new bin is the weighted sum of the coverage in overlapping regions of cov_df.

Usage

```
rebin_cov(cov_df, bins)
```

Arguments

cov_df	a data.frame with coverage information.
bins	a data.frame or GRanges object with the new bins.

Details

Coordinates in data.frame are expected to be defined by columns names 'chr', 'start' and 'end'.

Value

a data.frame with new bins

Author(s)

Jean Monlong

rm_cv_outliers *Remove outliers based on the coefficient of variation*

Description

Compute the coefficient of variation for each gene/bin and remove the ones with the highest values, either based on a quantile or SD-based threshold. The genes/bins that satisfy both quantile and SD-based thresholds are removed.

Usage

```
rm_cv_outliers(ge_df, ol_quant_th = 0.99, ol_sd_th = 5)
```

Arguments

ge_df	the expression data.frame.
ol_quant_th	the quantile threshold. Default is 0.99 (removes the top 1% with highest values).
ol_sd_th	the SD-based treshold. Default is 5.

Value

a subset of the ge_df data.frame

Author(s)

Jean Monlong

run_louvain *Python wrapper to run Louvain*

Description

Louvain on an igraph object.

Usage

```
run_louvain(graph, gamma = 1, nreps = 1, nb_cores = 1)
```

Arguments

graph	a igraph object
gamma	a vector of gamma. Default 1.
nreps	the number of repetition for each gamma, Default 1.
nb_cores	the number of processors to use. Default is 1.

Details

This functions depends on Python and louvain being installed. Make sure igraph, louvain and numpy are installed. For example with something like: 'pip install python-igraph louvain numpy'.

Value

a list with

comm	a data.frame with the community for each gamma
gamma	the input gammas corresponding to the columns of comm

Author(s)

Jean Monlong

---

run_pca *Run PCA*

Description

PCA analysis, eventually using a subset of core cells for the PC construction.

Usage

```
run_pca(z_df, core_cells = NULL, out_pcs = 100)
```

Arguments

z_df	a data.frame with z-scores for each cell
core_cells	if non-NULL, a vector with the names of the cells to use as core cells. See details. Default is NULL.
out_pcs	the number of top PCs to report. Default is 100.

Details

Cells in core_cells are used to build the principal components to which all cells are then projected to. Usually used to reduce the effect of cell cycle in the PCA, by using only cells that don't cycle (see qc_cells) as *core_cells*.

The graph (*sdev.graph*) shows the standard deviation for the top 50 PCs. To show more/less PCs, add `xlim(1,N)` to the *sdev.graph*. See examples.

Value

a list with

x	the PC matrix
sdev	the standard deviations of the PCs
sdev.graph	a ggplot graph of the sdev

Author(s)

Jean Monlong

Examples

```
## Not run:
pca.o = run_pca(z)

## Zoom in to the top 20 PCs
pca.o$sdev.graph + xlim(1,20)

## End(Not run)
```

---

run_tsne *Run tSNE*

Description

tSNE from PCA results.

Usage

```
run_tsne(pca_o, nb_pcs = 10, nb_it = 1000, tsne_init = NULL,
  seed = 999)
```

Arguments

pca_o	the output of run_pca
nb_pcs	the number of PCs to use. Default 10.
nb_it	the number of iterations. Default 1000.
tsne_init	previous tSNE results to use as starting point. Not used is NULL (default).
seed	the seed for the random generator.

Value

a data.frame with columns: cell, tsne1, tsne2

Author(s)

Jean Monlong

run_umap *Run UMAP*

Description

UMAP on the PCA results.

Usage

```
run_umap(pca_o, nb_pcs = 10, nb_neighbors = 5)
```

Arguments

pca_o	the output of run_pca
nb_pcs	the number of PCs to use. Default 10.
nb_neighbors	the number of neighbors. Default 5.

Details

This functions depends on Python and UMAP being installed. Make sure umap-mearn, sklearn, numpy, scipy and pandas are installed. For example with something like: 'pip install sklearn numpy scipy pandas umap-learn'.

Value

a data.frame with columns: cell, umap1, umap2

Author(s)

Jean Monlong

smoothMovingC *Smooth signal*

Description

Smooth signal

Usage

```
smoothMovingC(ge, winsize)
```

Arguments

ge	an ordered gene expression matrix
winsize	the size of the sliding window (assumed to be odd).

`smooth_movingw` *Moving-window smoothing*

Description

The expression/score of a gene/bin is replaced by a summary of bins around. For example the median across 3 bins.

Usage

```
smooth_movingw(df, wsize = 3, nb_cores = 1, FUN = stats::median,
  rcpp = FALSE)
```

Arguments

`df`	the input data.frame with coordinate columns (chr, start, end) and then one column per cell
`wsize`	the window size. Default is 3.
`nb_cores`	the number of processors to use.
`FUN`	the function to apply to each window. Default is median.
`rcpp`	use Rcpp function. Default is FALSE. More memory-efficient and faster when running on one core.

Value

a data.frame with smoothed signal.

Author(s)

Jean Monlong

`tmmNormC` *Compute the normalization factor*

Description

Compute the normalization factor

Usage

```
tmmNormC(ge, cont)
```

Arguments

`ge`	gene expression amtrix
`cont`	index of the control sample. R index so starting at 1.

tsne_browser *Shiny application to visualize tSNE results*

Description

Interactive application to visualize the tSNE results: zoom, hover information, different colors.

Usage

```
tsne_browser(cells_df, nb_points = 5000, plot_dim = 800)
```

Arguments

cells_df	the data.frame with tSNE and other information for each cell
nb_points	the default number of points to show. See details.
plot_dim	the dimension of the plot in pixels.

Details

Drawing thousands of points in a web-browser can be demanding. To reduce the number of points (cells) to draw, close-by cells are merged into bigger points. The merging is done separately for different samples/communities to be able to color them if necessary. The user can decide how many points to draw with the 'nb_points' parameter or directly within the application. In practice, increase the number of points until the app gets too slow.

Value

opens a Shiny app in a web-browser.

Author(s)

Jean Monlong

umap_browser *Shiny application to visualize UMAP results*

Description

Interactive application to visualize the UMAP results: zoom, hover information, different colors.

Usage

```
umap_browser(cells_df, nb_points = 5000, plot_dim = 800)
```

Arguments

cells_df	the data.frame with UMAP and other information for each cell
nb_points	the default number of points to show. See details.
plot_dim	the dimension of the plot in pixels.

Details

Drawing thousands of points in a web-browser can be demanding. To reduce the number of points (cells) to draw, close-by cells are merged into bigger points. The merging is done separately for different samples/communities to be able to color them if necessary. The user can decide how many points to draw with the 'nb_points' parameter or directly within the application. In practice, increase the number of points until the app gets too slow.

Value

opens a Shiny app in a web-browser.

Author(s)

Jean Monlong

---

winsor *Winsorize*

Description

Convenience function to winsorize a vector.

Usage

```
winsor(x, u = NULL, l = NULL, uq = NULL)
```

Arguments

x	input vector
u	upper limit
l	lower limit
uq	the quantile for the upper limit. Used is u is NULL.

Value

winsorized vector

Author(s)

Jean Monlong

`zscore` *Compute Z-score*

Description

Transform gene expression into a scaled score, either using all cells or a subset of cells as baseline.

Usage

```
zscore(ge_df, wins_th = 3, method = c("z", "norm"), normals = NULL)
```

Arguments

`ge_df`	the input expression data.frame
`wins_th`	the threshold to winsorize Z-score. Default is 3
`method`	the normalization method. Either 'z' or norm'.
`normals`	the cells to use as normals. If NULL (default) all cells are used as normals

Value

a data.frame with Z-scores.

Author(s)

Jean Monlong

Index