Package 'tidytof'

December 31, 2024

Type Package

Title Analyze High-dimensional Cytometry Data Using Tidy Data Principles

Version 1.1.0

Description This package implements an interactive, scientific analysis pipeline for high-dimensional cytometry data built using tidy data principles. It is specifically designed to play well with both the tidyverse and Bioconductor software ecosystems, with functionality for reading/writing data files, data cleaning, preprocessing, clustering, visualization, modeling, and other quality-of-life functions. tidytof implements a ``grammar'' of high-dimensional cytometry data analysis.

License MIT + file LICENSE

Depends R (>= 4.3)

Imports doParallel, dplyr, flowCore, foreach, ggplot2, ggraph, glmnet, methods, parallel, purrr, readr, recipes, rlang, stringr, survival, tidygraph, tidyr, tidyselect, yardstick, Rcpp, tibble, stats, utils, RcppHNSW

Suggests ConsensusClusterPlus, Biobase, broom, covr, diffcyt, emdist, FlowSOM, forcats, ggrepel, HDCytoData, knitr, markdown, philentropy, rmarkdown, Rtsne, statmod, SummarizedExperiment, testthat (>= 3.0.0), lmerTest, lme4, ggridges, spelling, scattermore, preprocessCore, SingleCellExperiment, Seurat, SeuratObject, embed, rsample, BiocGenerics

Config/testthat/edition 3

Encoding UTF-8

LazyData false

RoxygenNote 7.3.1

LinkingTo Rcpp

URL https://keyes-timothy.github.io/tidytof, https://keyes-timothy.github.io/tidytof/

BugReports https://github.com/keyes-timothy/tidytof/issues

VignetteBuilder knitr

Language en-US

biocViews SingleCell, FlowCytometry

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

git_branch devel

git_last_commit 3ca5f4a

git_last_commit_date 2024-10-29

Repository Bioconductor 3.21

Date/Publication 2024-12-30

Author Timothy Keyes [cre] (ORCID: <https://orcid.org/0000-0003-0423-9679>),
Kara Davis [rth, own],
Garry Nolan [rth, own]

Maintainer Timothy Keyes <tkeyes@stanford.edu>

Contents

`as_flowFrame` *Coerce an object into a* `flowFrame`

Description

Coerce an object into a `flowFrame`

Coerce a tof_tbl into a `flowFrame`

Usage

```
as_flowFrame(x, ...)

## S3 method for class 'tof_tbl'
as_flowFrame(x, ...)
```

Arguments

x	A tof_tbl.
...	Unused.

Value

A `flowFrame`

A `flowFrame`. Note that all non-numeric columns in ‘x‘ will be removed.

Examples

```
NULL

NULL
```

as_flowSet *Coerce an object into a flowSet*

Description

Coerce an object into a flowSet

Coerce a tof_tbl into a flowSet

Usage

```
as_flowSet(x, ...)

## S3 method for class 'tof_tbl'
as_flowSet(x, group_cols, ...)
```

Arguments

x	A tof_tbl.
...	Unused.
group_cols	Unquoted names of the columns in ‘x‘ that should be used to group cells into separate flowFrames. Supports tidyselect helpers. Defaults to NULL (all cells are written into a single flowFrame).

Value

A flowSet

A flowSet. Note that all non-numeric columns in ‘x‘ will be removed.

Examples

```
NULL

NULL
```

as_seurat *Coerce an object into a SeuratObject*

Description

Coerce an object into a SeuratObject

Coerce a tof_tbl into a SeuratObject

Usage

```
as_seurat(x, ...)

## S3 method for class 'tof_tbl'
as_seurat(
  x,
  channel_cols = where(tof_is_numeric),
  reduced_dimensions_cols,
  metadata_cols = where(function(.x) !tof_is_numeric(.x)),
  split_reduced_dimensions = FALSE,
  ...
)
```

Arguments

x	A tof_tbl
...	Unused.
channel_cols	Unquoted column names representing columns that contain single-cell protein measurements. Supports tidyselect helpers. If nothing is specified, the default is all numeric columns.
reduced_dimensions_cols	Unquoted column names representing columns that contain dimensionality reduction embeddings, such as tSNE or UMAP embeddings. Supports tidyselect helpers.
metadata_cols	Unquoted column names representing columns that contain metadata about the samples from which each cell was collected. If nothing is specified, the default is all non-numeric columns.
split_reduced_dimensions	A boolean value indicating whether the dimensionality results in x should be split into separate slots in the resulting SingleCellExperiment. If FALSE (the default), the split will not be performed and the reducedDims slot in the result will have a single entry ("tidytof_reduced_dimensions"). If TRUE, the split will be performed and the reducedDims slot in the result will have 1-4 entries depending on which dimensionality reduction results are present in x ("tidytof_pca", "tidytof_tsne", "tidytof_umap", and "tidytof_reduced_dimensions"). Note that "tidytof_reduced_dimensions" will include all dimensionality reduction results that are not named according to tidytof's pca, umap, and tsne conventions.

Value

A SeuratObject

A SeuratObject.

Examples

```
NULL
```

```
NULL
```

`as_SingleCellExperiment` *Coerce an object into a* `SingleCellExperiment`

Description

Coerce an object into a `SingleCellExperiment`

Coerce a tof_tbl into a `SingleCellExperiment`

Usage

```
as_SingleCellExperiment(x, ...)

## S3 method for class 'tof_tbl'
as_SingleCellExperiment(
  x,
  channel_cols = where(tof_is_numeric),
  reduced_dimensions_cols,
  metadata_cols = where(function(.x) !tof_is_numeric(.x)),
  split_reduced_dimensions = FALSE,
  ...
)
```

Arguments

`x`	A tof_tbl
`...`	Unused.
`channel_cols`	Unquoted column names representing columns that contain single-cell protein measurements. Supports tidyselect helpers. If nothing is specified, the default is all numeric columns.
`reduced_dimensions_cols`	Unquoted column names representing columns that contain dimensionality reduction embeddings, such as tSNE or UMAP embeddings. Supports tidyselect helpers.
`metadata_cols`	Unquoted column names representing columns that contain metadata about the samples from which each cell was collected. If nothing is specified, the default is all non-numeric columns.
`split_reduced_dimensions`	A boolean value indicating whether the dimensionality results in x should be split into separate slots in the resulting `SingleCellExperiment`. If FALSE (the default), the split will not be performed and the `reducedDims` slot in the result will have a single entry ("tidytof_reduced_dimensions"). If TRUE, the

split will be performed and the reducedDims slot in the result will have 1-4 entries depending on which dimensionality reduction results are present in x ("tidytof_pca", "tidytof_tsne", "tidytof_umap", and "tidytof_reduced_dimensions"). Note that "tidytof_reduced_dimensions" will include all dimensionality reduction results that are not named according to tidytof's pca, umap, and tsne conventions.

Value

A SingleCellExperiment

A SingleCellExperiment.

Examples

```
NULL
```

```
NULL
```

---

as_tof_tbl *Coerce flowFrames or flowSets into tof_tbl's.*

Description

Coerce flowFrames or flowSets into tof_tbl's.

Usage

```
as_tof_tbl(flow_data, sep = "|")
```

Arguments

flow_data	A flowFrame or flowSet
sep	A string indicating which symbol should be used to separate antigen names and metal names in the columns of the output tof_tbl.

Value

A tof_tbl.

Examples

```
input_file <- dir(tidytof_example_data("aml"), full.names = TRUE)[[1]]

input_flowframe <- flowCore::read.FCS(input_file)

tof_tibble <- as_tof_tbl(input_flowframe)
```

as_tof_tbl.flowSet *Convert an object into a tof_tbl*

Description

Convert an object into a tof_tbl

Usage

```
## S3 method for class 'flowSet'
as_tof_tbl(flow_data, sep = "|")
```

Arguments

flow_data	A FlowSet
sep	A string to use to separate the antigen name and its associated metal in the column names of the output tibble. Defaults to "|".

Value

a ‘tof_tbl‘

cosine_similarity *Find the cosine similarity between two vectors*

Description

Find the cosine similarity between two vectors

Usage

```
cosine_similarity(x, y)
```

Arguments

x	a numeric vector
y	a numeric vector

Value

a scalar value representing the cosine similarity between x and y

`ddpr_data` *CyTOF data from two samples: 5,000 B-cell lineage cells from a healthy patient and 5,000 B-cell lineage cells from a B-cell precursor Acute Lymphoblastic Leukemia (BCP-ALL) patient.*

Description

A dataset containing CyTOF measurements from immune cells originally studied in the following paper:

Good Z, Sarno J, et al. Single-cell developmental classification of B cell precursor acute lymphoblastic leukemia at diagnosis reveals predictors of relapse. Nat Med. 2018 May;24(4):474-483. doi: 10.1038/nm.4505. Epub 2018 Mar 5. PMID: 29505032; PMCID: PMC5953207.

Usage

```
data(ddpr_data)
```

Format

A data frame with 10000 rows and 24 variables:

sample_name name of the sample from which the data was read

cd45 A CyTOF measurement in raw ion counts

cd19 A CyTOF measurement in raw ion counts

cd22 A CyTOF measurement in raw ion counts

cd79b A CyTOF measurement in raw ion counts

cd20 A CyTOF measurement in raw ion counts

cd34 A CyTOF measurement in raw ion counts

cd123 A CyTOF measurement in raw ion counts

cd10 A CyTOF measurement in raw ion counts

cd24 A CyTOF measurement in raw ion counts

cd127 A CyTOF measurement in raw ion counts

cd43 A CyTOF measurement in raw ion counts

cd38 A CyTOF measurement in raw ion counts

cd58 A CyTOF measurement in raw ion counts

psyk A CyTOF measurement in raw ion counts

p4ebp1 A CyTOF measurement in raw ion counts

pstat5 A CyTOF measurement in raw ion counts

pakt A CyTOF measurement in raw ion counts

ps6 A CyTOF measurement in raw ion counts

perk A CyTOF measurement in raw ion counts

pcreb A CyTOF measurement in raw ion counts

Value

A data.frame

Source

https://github.com/kara-davis-lab/DDPR

`ddpr_metadata` *Clinical metadata for each patient sample in Good & Sarno et al. (2018).*

Description

A dataset containing patient-level clinical metadata for samples originally studied in the following paper:

Good Z, Sarno J, et al. Single-cell developmental classification of B cell precursor acute lymphoblastic leukemia at diagnosis reveals predictors of relapse. Nat Med. 2018 May;24(4):474-483. doi: 10.1038/nm.4505. Epub 2018 Mar 5. PMID: 29505032; PMCID: PMC5953207.

Usage

```
data(ddpr_metadata)
```

Format

A data frame with 10000 rows and 12 variables:

patient_id Name of the sample from which the data was read

gender Gender of the patient from which each sample was collected

age_at_diagnosis Age (in years) of the patient from which each sample was collected

wbc_count The diagnostic White Blood Cell (WBC) count of the patient from which each sample was collected

mrd_risk Risk stratification category for each patient using minimal residual disease (MRD) criteria

nci_rome_risk Risk stratification category for each patient using National Cancer Institute (NCI) criteria

relapse_status A string representing whether or not a patient relapsed

time_to_relapse The time (in days) it took each patient to relapse. Patients who did not relapse will have the value of NA

type_of_relapse A string representing the timing of relapse for each patient. "Very early" relapses occurred less than 18 months after diagnosis; "Early" relapses occurred between 18 months and 32 months after diagnosis; "Late" relapses occurred later than 32 months after diagnosis.

ccr The number of documented days of continuous complete remission (CCR) for patients who did not relapse. All patients who relapsed will have a value of NA.

cohort A string representing if each sample was used in the "Training" or "Validation" cohort in the original study

ddpr_risk The risk category ("Low" or "High") assigned to each sample using the original paper's risk-stratification algorithm

Value

A data.frame

Source

Good Z, Sarno J, et al. Single-cell developmental classification of B cell precursor acute lymphoblastic leukemia at diagnosis reveals predictors of relapse. Nat Med. 2018 May;24(4):474-483. doi: 10.1038/nm.4505. Epub 2018 Mar 5. PMID: 29505032; PMCID: PMC5953207. Supplementary Table 1.

`dot` *Find the dot product between two vectors.*

Description

Find the dot product between two vectors.

Usage

```
dot(x, y)
```

Arguments

`x`	A numeric vector.
`y`	A numeric vector.

Value

The dot product between x and y.

get_extension *Find the extension for a file*

Description

Find the extension for a file

Usage

```
get_extension(filename)
```

Arguments

filename	A string representing the name of a file in its local directory

Value

The the file extension of ‘filename‘

l2_normalize *L2 normalize an input vector x to a length of 1*

Description

L2 normalize an input vector x to a length of 1

Usage

```
l2_normalize(x)
```

Arguments

x	a numeric vector

Value

a vector of length length(x) with a magnitude of 1

`magnitude` *Find the magnitude of a vector.*

Description

Find the magnitude of a vector.

Usage

```
magnitude(x)
```

Arguments

`x`	A numeric vector.

Value

A scalar value (the magnitude of x).

`make_flowcore_annotated_data_frame` *Make the AnnotatedDataFrame needed for the flowFrame class*

Description

Make the AnnotatedDataFrame needed for the flowFrame class

Usage

```
make_flowcore_annotated_data_frame(maxes_and_mins)
```

Arguments

`maxes_and_mins`	a data.frame containing information about the max and min values of each channel to be saved in the flowFrame.

Value

An AnnotatedDataFrame.

Examples

```
NULL
```

metal_masterlist *A character vector of metal name patterns supported by tidytof.*

Description

A character vector used by ‘tof_read_fcs‘ and ‘tof_read_data‘ to detect and parse which CyTOF metals correspond to each channel in an input .fcs file.

Usage

```
data(metal_masterlist)
```

Format

A character vector in which each entry is a pattern that tidytof searches for in every CyTOF channel in input .fcs files. These patterns are an amalgamate of example .fcs files sampled from the studies linked below.

Value

A named character vector.

Source

https://github.com/kara-davis-lab/DDPR https://cytobank.org/nolanlab/reports/Levine2015.html https://cytobank.org/nolanlab/reports/Spitzer2015.html https://cytobank.org/nolanlab/reports/Spitzer2017.html https://community.cytobank.org/cytobank/projects/609

new_tof_model *Constructor for a tof_model.*

Description

Constructor for a tof_model.

Usage

```
new_tof_model(
  model,
  recipe,
  penalty,
  mixture,
  model_type = c("linear", "two-class", "multiclass", "survival"),
  outcome_colnames,
  training_data
)
```

Arguments

model	A glmnet model.
recipe	A prepped recipe object.
penalty	A double indicating which lambda value should be used within the glmnet path.
mixture	A double indicating which alpha value was used to fit the glmnet model.
model_type	A string indicating which type of glmnet model is being fit.
outcome_colnames	TO DO
training_data	TO DO

Value

A 'tof_model', an S3 class that includes a trained glmnet model and the recipe used to perform its associated preprocessing.

---

new_tof_tibble *Constructor for a tof_tibble.*

Description

Constructor for a tof_tibble.

Usage

```
new_tof_tibble(x = dplyr::tibble(), panel = dplyr::tibble())
```

Arguments

x	A data.frame or tibble containing single-cell mass cytometry data such that rows are cells and columns are CyTOF measurements.
panel	A data.frame or tibble containing information about the panel for the mass cytometry data in x.

Value

A 'tof_tbl', an tibble extension that tracks a few other attributes that are useful for CyTOF data analysis.

See Also

Other tof_tbl utilities: `tof_get_panel()`, `tof_set_panel()`

phenograph_data *CyTOF data from 6,000 healthy immune cells from a single patient.*

Description

A dataset containing CyTOF measurements from healthy control cells originally studied in the following paper:

Levine JH, Simonds EF, et al. Data-Driven Phenotypic Dissection of AML Reveals Progenitor-like Cells that Correlate with Prognosis. Cell. 2015 Jul 2;162(1):184-97. doi: 10.1016/j.cell.2015.05.047. Epub 2015 Jun 18. PMID: 26095251; PMCID: PMC4508757.

Usage

```
data(phenograph_data)
```

Format

A data frame with 6000 rows and 26 variables:

sample_name Name of the sample from which the data was read

phenograph_cluster Numeric ID of the cluster assignment of each row

cd19 A CyTOF measurement in raw ion counts

cd11b A CyTOF measurement in raw ion counts

cd34 A CyTOF measurement in raw ion counts

cd45 A CyTOF measurement in raw ion counts

cd123 A CyTOF measurement in raw ion counts

cd33 A CyTOF measurement in raw ion counts

cd47 A CyTOF measurement in raw ion counts

cd7 A CyTOF measurement in raw ion counts

cd44 A CyTOF measurement in raw ion counts

cd38 A CyTOF measurement in raw ion counts

cd3 A CyTOF measurement in raw ion counts

cd117 A CyTOF measurement in raw ion counts

cd64 A CyTOF measurement in raw ion counts

cd41 A CyTOF measurement in raw ion counts

pstat3 A CyTOF measurement in raw ion counts

pstat5 A CyTOF measurement in raw ion counts

pampk A CyTOF measurement in raw ion counts

p4ebp1 A CyTOF measurement in raw ion counts

ps6 A CyTOF measurement in raw ion counts

pcreb A CyTOF measurement in raw ion counts

pzap70-syk A CyTOF measurement in raw ion counts

prb A CyTOF measurement in raw ion counts

perk1-2 A CyTOF measurement in raw ion counts

Details

2000 cells from 3 clusters identified in the original paper have been sampled.

Value

A data.frame

Source

https://cytobank.org/nolanlab/reports/Levine2015.html

`reexports` *Objects exported from other packages*

Description

These objects are imported from other packages. Follow the links below to see their documentation.

dplyr %>%

rlang :=, .data

tidyselect all_of, any_of, contains, ends_with, everything, last_col, matches, num_range, starts_with

Value

See documentation in each object's original package.

Examples

```
# See examples in each object's original package
NULL
```

`rev_asinh` *Reverses arcsinh transformation with cofactor 'scale_factor' and a shift of 'shift_factor'.*

Description

Reverses arcsinh transformation with cofactor 'scale_factor' and a shift of 'shift_factor'.

Usage

```
rev_asinh(x, shift_factor, scale_factor)
```

Arguments

`x`	A numeric vector.
`shift_factor`	The scalar value 'a' in the following equation used to transform high-dimensional cytometry raw data ion counts using the hyperbolic arcsinh function: 'new_x <- asinh(a + b * x)'.
`scale_factor`	The scalar value 'b' in the following equation used to transform high-dimensional cytometry raw data ion counts using the hyperbolic arcsinh function: 'new_x <- asinh(a + b * x)'.

Value

A numeric vector after undergoing reverse arcsinh transformation

Examples

```
shift_factor <- 0
scale_factor <- 1 / 5

input_value <- 20
asinh_value <- asinh(shift_factor + input_value * scale_factor)

restored_value <- rev_asinh(asinh_value, shift_factor, scale_factor)
```

`tidytof_example_data` *Get paths to tidytof example data*

Description

tidytof comes bundled with a number of sample .fcs files in its inst/extdata directory. This function makes them easy to access.

Usage

```
tidytof_example_data(dataset_name = NULL)
```

Arguments

dataset_name	Name of the dataset you want to access. If NULL, the names of the datasets (each of which is from a different study) will be listed.

Value

A character vector of file paths where the requested .fcs files are located. If ‘dataset_name‘ is NULL, a character vector of dataset names (that can be used as values for ‘dataset_name‘) is returned instead.

Examples

```
tidytof_example_data()
tidytof_example_data(dataset_name = "phenograph")
```

---

tof_analyze_abundance *Perform Differential Abundance Analysis (DAA) on high-dimensional cytometry data*

Description

This function performs differential abundance analysis on the cell clusters contained within a ‘tof_tbl‘ using one of three methods ("diffcyt", "glmm", and "ttest"). It wraps the members of the ‘tof_analyze_abundance_*‘ function family: tof_analyze_abundance_diffcyt, tof_analyze_abundance_glmm, and tof_analyze_abundance_ttes

Usage

```
tof_analyze_abundance(tof_tibble, method = c("diffcyt", "glmm", "ttest"), ...)
```

Arguments

tof_tibble	A ‘tof_tbl‘ or a ‘tibble‘.
method	A string indicating which statistical method should be used. Valid values include "diffcyt", "glmm", and "ttest".
...	Additional arguments to pass onto the ‘tof_analyze_abundance_*‘ function family member corresponding to the chosen method.

Value

A tibble or nested tibble containing the differential abundance results from the chosen method. See tof_analyze_abundance_diffcyt, tof_analyze_abundance_glmm, and tof_analyze_abundance_ttest for details.

See Also

Other differential abundance analysis functions: `tof_analyze_abundance_diffcyt()`, `tof_analyze_abundance_glmm()`, `tof_analyze_abundance_ttest()`

Examples

```
# For differential discovery examples, please see the package vignettes
NULL
```

`tof_analyze_abundance_diffcyt`

Differential Abundance Analysis (DAA) with diffcyt

Description

This function performs differential abundance analysis on the cell clusters contained within a ‘tof_tbl‘ using one of three methods implemented in the diffcyt package for differential discovery analysis in high-dimensional cytometry data.

Usage

```
tof_analyze_abundance_diffcyt(
  tof_tibble,
  sample_col,
  cluster_col,
  fixed_effect_cols,
  random_effect_cols,
  diffcyt_method = c("glmm", "edgeR", "voom"),
  include_observation_level_random_effects = FALSE,
  min_cells = 3,
  min_samples = 5,
  alpha = 0.05,
  ...
)
```

Arguments

`tof_tibble`	A ‘tof_tbl‘ or a ‘tibble‘.
`sample_col`	An unquoted column name indicating which column in ‘tof_tibble‘ represents the id of the sample from which each cell was collected. ‘sample_col‘ should serve as a unique identifier for each sample collected during data acquisition - all cells with the same value for ‘sample_col‘ will be treated as a part of the same observational unit.

`cluster_col` An unquoted column name indicating which column in 'tof_tibble' stores the cluster ids of the cluster to which each cell belongs. Cluster labels can be produced via any method the user chooses - including manual gating, any of the functions in the 'tof_cluster_*' function family, or any other method.

`fixed_effect_cols` Unquoted column names representing which columns in 'tof_tibble' should be used to model fixed effects during the differential abundance analysis. Generally speaking, fixed effects represent the comparisons of biological interest (often the variables manipulated during experiments), such as treated vs. non-treated, before-treatment vs. after-treatment, or healthy vs. non-healthy.

`random_effect_cols` Optional. Unquoted column names representing which columns in 'tof_tibble' should be used to model random effects during the differential abundance analysis. Generally speaking, random effects should represent variables that a researcher wants to control/account for, but that are not necessarily of biological interest. Example random effect variables might include batch id, patient id (in a paired design), or patient age.

Note that without multiple samples at each level of each of the random effect variables, it can be easy to overfit mixed models. For most high-dimensional cytometry experiments, 2 or fewer (and often 0) random effect variables are appropriate.

`diffcyt_method` A string indicating which diffcyt method should be used for the differential abundance analysis. Valid methods include "glmm" (the default), "edgeR", and "voom".

`include_observation_level_random_effects` A boolean value indicating if "observation-level random effects" (OLREs) should be included as random effect terms in a "glmm" differential abundance model. For details about what OLREs are, see the diffcyt paper. Only the "glmm" method can model observation-level random effects, and all other values will ignore this argument (and throw a warning if it is set to TRUE). Defaults to FALSE.

`min_cells` An integer value used to filter clusters out of the differential abundance analysis. Clusters are not included in the differential abundance testing if they do not have at least 'min_cells' in at least 'min_samples' samples. Defaults to 3.

`min_samples` An integer value used to filter clusters out of the differential abundance analysis. Clusters are not included in the differential abundance testing if they do not have at least 'min_cells' in at least 'min_samples' samples. Defaults to 5.

`alpha` A numeric value between 0 and 1 indicating which significance level should be applied to multiple-comparison adjusted p-values during the differential abundance analysis. Defaults to 0.05.

`...` Optional additional arguments to pass to the under-the-hood diffcyt function being used to perform the differential abundance analysis. See `testDA_GLMM`, `testDA_edgeR`, and `testDA_voom` for details.

Details

The three methods are based on generalized linear mixed models ("glmm"), edgeR ("edgeR"), and voom ("voom"). While both the "glmm" and "voom" methods can model both fixed effects and

random effects, the "edgeR" method can only model fixed effects.

Value

A nested tibble with two columns: ‘tested_effect‘ and ‘daa_results‘.

The first column, ‘tested_effect‘ is a character vector indicating which term in the differential abundance model was used for significance testing. The values in this row are obtained by pasting together the column names for each fixed effect variable and each of its values. For example, a fixed effect column named ‘fixed_effect‘ with levels "a", "b", and "c" have two terms in ‘tested_effect‘: "fixed_effectb" and "fixed_effectc" (note that level "a" of fixed_effect is set as the reference level during dummy coding). These values correspond to the terms in the differential abundance model that represent the difference in cluster abundances between samples with fixed_effect = "b" and fixed_effect = "a" and between samples with fixed_effect = "c" and fixed_effect = "a", respectively. In addition, the first row in ‘tested_effect‘ will always represent the "omnibus" test, or the test that there were significant differences between *any* levels of *any* fixed effect variable in the model.

The second column, ‘daa_results‘ is a list of tibbles in which each entry gives the differential abundance results for each tested_effect. Within each entry of ‘daa_results‘, you will find several columns including the following: * ‘p_val‘, the p-value associated with each tested effect in each input cluster * ‘p_adj‘, the multiple-comparison adjusted p-value (using the `p.adjust` function) * Other values associated with the underlying method used to perform the differential abundance analysis (such as the log-fold change of cluster abundance between the levels being compared). For details, see `glmFit`, `voom`, `topTable`, and `testDA_GLMM`.

See Also

Other differential abundance analysis functions: `tof_analyze_abundance()`, `tof_analyze_abundance_glmm()`, `tof_analyze_abundance_ttest()`

Examples

```
# For differential discovery examples, please see the package vignettes
NULL
```

`tof_analyze_abundance_glmm` *Differential Abundance Analysis (DAA) with generalized linear mixed-models (GLMMs)*

Description

This function performs differential abundance analysis on the cell clusters contained within a ‘tof_tbl‘ using generalized linear mixed-models. Users specify which columns represent sample, cluster, fixed effect, and random effect information, and a (mixed) binomial regression model is fit using either `glmer` or `glm`.

Usage

```
tof_analyze_abundance_glmm(
  tof_tibble,
  sample_col,
  cluster_col,
  fixed_effect_cols,
  random_effect_cols,
  min_cells = 3,
  min_samples = 5,
  alpha = 0.05
)
```

Arguments

tof_tibble	A 'tof_tbl' or a 'tibble'.
sample_col	An unquoted column name indicating which column in 'tof_tibble' represents the id of the sample from which each cell was collected. 'sample_col' should serve as a unique identifier for each sample collected during data acquisition - all cells with the same value for 'sample_col' will be treated as a part of the same observational unit.
cluster_col	An unquoted column name indicating which column in 'tof_tibble' stores the cluster ids of the cluster to which each cell belongs. Cluster labels can be produced via any method the user chooses - including manual gating, any of the functions in the 'tof_cluster_*' function family, or any other method.
fixed_effect_cols	Unquoted column names representing which columns in 'tof_tibble' should be used to model fixed effects during the differential abundance analysis. Supports tidyselect helpers. Generally speaking, fixed effects should represent the comparisons of biological interest (often the the variables manipulated during experiments), such as treated vs. non-treated, before-treatment vs. after-treatment, or healthy vs. non-healthy.
random_effect_cols	Unquoted column names representing which columns in 'tof_tibble' should be used to model random effects during the differential abundance analysis. Supports tidyselection. Generally speaking, random effects should represent variables that a researcher wants to control/account for, but that are not necessarily of biological interest. Example random effect variables might include batch id, patient id (in a paired design), or patient age. Note that without many samples at each level of each of the random effect variables, it can be easy to overfit mixed models. For most high-dimensional cytometry experiments, 2 or fewer (and often 0) random effect variables are appropriate.
min_cells	An integer value used to filter clusters out of the differential abundance analysis. Clusters are not included in the differential abundance testing if they do not have at least 'min_cells' in at least 'min_samples' samples. Defaults to 3.

`min_samples` An integer value used to filter clusters out of the differential abundance analysis. Clusters are not included in the differential abundance testing if they do not have at least ‘min_cells‘ in at least ‘min_samples‘ samples. Defaults to 5.

`alpha` A numeric value between 0 and 1 indicating which significance level should be applied to multiple-comparison adjusted p-values during the differential abundance analysis. Defaults to 0.05.

Value

A nested tibble with two columns: ‘tested_effect‘ and ‘daa_results‘.

The first column, ‘tested_effect‘, is a character vector indicating which term in the differential abundance model was used for significance testing. The values in this row are obtained by pasting together the column names for each fixed effect variable and each of its values. For example, a fixed effect column named fixed_effect with levels "a", "b", and "c" have two terms in ‘tested_effect‘: "fixed_effectb" and "fixed_effectc" (note that level "a" of fixed_effect is set as the reference level during dummy coding). These values correspond to the terms in the differential abundance model that represent the difference in cluster abundances between samples with fixed_effect = "b" and fixed_effect = "a" and between samples with fixed_effect = "c" and fixed_effect = "a", respectively. In addition, note that the first row in ‘tested_effect‘ will always represent the "omnibus" test, or the test that there were significant differences between any levels of any fixed effect variable in the model.

The second column, ‘daa_results‘, is a list of tibbles in which each entry gives the differential abundance results for each tested_effect. Within each entry of ‘daa_results‘, you will find ‘p_value‘, the p-value associated with each tested effect in each input cluster; ‘p_adj‘, the multiple-comparison adjusted p-value (using the `p.adjust` function), and other values associated with the underlying method used to perform the differential abundance analysis (such as the log-fold change of cluster abundance between the levels being compared).

See Also

Other differential abundance analysis functions: `tof_analyze_abundance()`, `tof_analyze_abundance_diffcyt()`, `tof_analyze_abundance_ttest()`

Examples

```
# For differential discovery examples, please see the package vignettes
NULL
```

`tof_analyze_abundance_ttest`

Differential Abundance Analysis (DAA) with t-tests

Description

This function performs differential abundance analysis on the cell clusters contained within a ‘tof_tbl‘ using simple t-tests. Users specify which columns represent sample, cluster, and effect information, and either a paired or unpaired t-test (one per cluster) is used to detect significant differences between sample types.

Usage

```
tof_analyze_abundance_ttest(
  tof_tibble,
  cluster_col,
  effect_col,
  group_cols,
  test_type = c("unpaired", "paired"),
  min_cells = 3,
  min_samples = 5,
  alpha = 0.05,
  quiet = FALSE
)
```

Arguments

tof_tibble	A ‘tof_tbl‘ or a ‘tibble‘.
cluster_col	An unquoted column name indicating which column in ‘tof_tibble‘ stores the cluster ids of the cluster to which each cell belongs. Cluster labels can be produced via any method the user chooses - including manual gating, any of the functions in the ‘tof_cluster_*‘ function family, or any other method.
effect_col	Unquoted column name representing which column in ‘tof_tibble‘ should be used to break samples into groups for the t-test. Should only have 2 unique values.
group_cols	Unquoted names of the columns other than ‘effect_col‘ that should be used to group cells into independent observations. Fills a similar role to ‘sample_col‘ in other ‘tof_analyze_abundance_*‘ functions. For example, if an experiment involves analyzing samples taken from multiple patients at two timepoints (with ‘effect_col = timepoint‘), then group_cols should be the name of the column representing patient IDs.
test_type	A string indicating whether the t-test should be "unpaired" (the default) or "paired".
min_cells	An integer value used to filter clusters out of the differential abundance analysis. Clusters are not included in the differential abundance testing if they do not have at least ‘min_cells‘ in at least ‘min_samples‘ samples. Defaults to 3.
min_samples	An integer value used to filter clusters out of the differential abundance analysis. Clusters are not included in the differential abundance testing if they do not have at least ‘min_cells‘ in at least ‘min_samples‘ samples. Defaults to 5.
alpha	A numeric value between 0 and 1 indicating which significance level should be applied to multiple-comparison adjusted p-values during the differential abundance analysis. Defaults to 0.05.

quiet A boolean value indicating whether warnings should be printed. Defaults to ‘TRUE‘.

Value

A tibble with 7 columns:

{cluster_col} The name/ID of the cluster being tested. Each entry in this column will match a unique value in the input {cluster_col}.

t The t-statistic computed for each cluster.

df The degrees of freedom used for the t-test for each cluster.

p_val The (unadjusted) p-value for the t-test for each cluster.

p_adj The `p.adjust`-adjusted p-value for the t-test for each cluster.

significant A character vector that will be "*" for clusters for which p_adj < alpha and "" otherwise.

mean_diff For an unpaired t-test, the difference between the average proportions of each cluster in the two levels of ‘effect_col‘. For a paired t-test, the average difference between the proportions of each cluster in the two levels of ‘effect_col‘ within a given patient.

mean_fc For an unpaired t-test, the ratio between the average proportions of each cluster in the two levels of ‘effect_col‘. For a paired t-test, the average ratio between the proportions of each cluster in the two levels of ‘effect_col‘ within a given patient. 0.001 is added to the denominator of the ratio to avoid divide-by-zero errors.

The "levels" attribute of the result indicates the order in which the different levels of the ‘effect_col‘ were considered. The ‘mean_diff‘ value for each row of the output is computed by subtracting the second level from the first level, and the ‘mean_fc‘ value for each row is computed by dividing the first level by the second level.

See Also

Other differential abundance analysis functions: `tof_analyze_abundance()`, `tof_analyze_abundance_diffcyt()`, `tof_analyze_abundance_glmm()`

Examples

```
# For differential discovery examples, please see the package vignettes
NULL
```

`tof_analyze_expression` *Perform Differential Expression Analysis (DEA) on high-dimensional cytometry data*

Description

This function performs differential expression analysis on the cell clusters contained within a ‘tof_tbl‘ using one of three methods ("diffcyt", "glmm", and "ttest"). It wraps the members of the ‘tof_analyze_expression_*‘ function family: tof_analyze_expression_diffcyt, tof_analyze_expression_lmm, and tof_analyze_expression_tt

Usage

```
tof_analyze_expression(tof_tibble, method = c("diffcyt", "glmm", "ttest"), ...)
```

Arguments

tof_tibble	A ‘tof_tbl‘ or a ‘tibble‘.
method	A string indicating which statistical method should be used. Valid values include "diffcyt", "lmm", and "ttest".
...	Additional arguments to pass onto the ‘tof_analyze_expression_*‘ function family member corresponding to the chosen method.

Value

A tibble or nested tibble containing the differential abundance results from the chosen method. See tof_analyze_expression_diffcyt, tof_analyze_expression_lmm, and tof_analyze_expression_ttest for details.

See Also

Other differential expression analysis functions: tof_analyze_expression_diffcyt(), tof_analyze_expression_lmm() tof_analyze_expression_ttest()

Examples

```
# For differential discovery examples, please see the package vignettes
NULL
```

---

tof_analyze_expression_diffcyt

Differential Expression Analysis (DEA) with diffcyt

Description

This function performs differential expression analysis on the cell clusters contained within a ‘tof_tbl‘ using one of two methods implemented in the diffcyt package for differential discovery analysis in high-dimensional cytometry data.

Usage

```
tof_analyze_expression_diffcyt(
  tof_tibble,
  sample_col,
  cluster_col,
  marker_cols = where(tof_is_numeric),
  fixed_effect_cols,
  random_effect_cols,
  diffcyt_method = c("lmm", "limma"),
  include_observation_level_random_effects = FALSE,
  min_cells = 3,
  min_samples = 5,
  alpha = 0.05,
  ...
)
```

Arguments

tof_tibble	A ‘tof_tbl‘ or a ‘tibble‘.
sample_col	An unquoted column name indicating which column in ‘tof_tibble‘ represents the id of the sample from which each cell was collected. ‘sample_col‘ should serve as a unique identifier for each sample collected during data acquisition - all cells with the same value for ‘sample_col‘ will be treated as a part of the same observational unit.
cluster_col	An unquoted column name indicating which column in ‘tof_tibble‘ stores the cluster ids of the cluster to which each cell belongs. Cluster labels can be produced via any method the user chooses - including manual gating, any of the functions in the ‘tof_cluster_*‘ function family, or any other method.
marker_cols	Unquoted column names representing which columns in ‘tof_tibble‘ (i.e. which high-dimensional cytometry protein measurements) should be tested for differential expression between levels of the ‘fixed_effect_cols‘. Defaults to all numeric (integer or double) columns. Supports tidyselect helpers.
fixed_effect_cols	Unquoted column names representing which columns in ‘tof_tibble‘ should be used to model fixed effects during the differential expression analysis. Generally speaking, fixed effects represent the comparisons of biological interest (often the the variables manipulated during experiments), such as treated vs. non-treated, before-treatment vs. after-treatment, or healthy vs. non-healthy.
random_effect_cols	Unquoted column names representing which columns in ‘tof_tibble‘ should be used to model random effects during the differential expression analysis. Generally speaking, random effects represent variables that a researcher wants to control/account for, but that are not necessarily of biological interest. Example random effect variables might include batch id, patient id (in a paired design), or patient age. Note that without many samples at each level of each of the random effect variables, it can be easy to overfit mixed models. For most high-dimensional cytom-

etry experiments, 2 or fewer (and often 0) random effect variables are appropriate.

`diffcyt_method` A string indicating which diffcyt method should be used for the differential expression analysis. Valid methods include "lmm" (the default) and "limma".

`include_observation_level_random_effects`
A boolean value indicating if "observation-level random effects" (OLREs) should be included as random effect terms in a "lmm" differential expression model. For details about what OLREs are, see the diffcyt paper. Defaults to FALSE.

`min_cells` An integer value used to filter clusters out of the differential expression analysis. Clusters are not included in the differential expression testing if they do not have at least ‘min_cells‘ in at least ‘min_samples‘ samples. Defaults to 3.

`min_samples` An integer value used to filter clusters out of the differential expression analysis. Clusters are not included in the differential expression testing if they do not have at least ‘min_cells‘ in at least ‘min_samples‘ samples. Defaults to 5.

`alpha` A numeric value between 0 and 1 indicating which significance level should be applied to multiple-comparison adjusted p-values during the differential abundance analysis. Defaults to 0.05.

`...` Optional additional arguments to pass to the under-the-hood diffcyt function being used to perform the differential expression analysis. See `testDS_LMM` and `testDS_limma` for details.

Details

The two methods are based on linear mixed models ("lmm") and limma ("limma"). Both the "lmm" and "limma" methods can model both fixed effects and random effects.

Value

A nested tibble with two columns: ‘tested_effect‘ and ‘dea_results‘.

The first column, ‘tested_effect‘ is a character vector indicating which term in the differential expression model was used for significance testing. The values in this row are obtained by pasting together the column names for each fixed effect variable and each of its values. For example, a fixed effect column named fixed_effect with levels "a", "b", and "c" have two terms in ‘tested_effect‘: "fixed_effectb" and "fixed_effectc" (note that level "a" of fixed_effect is set as the reference level during dummy coding). These values correspond to the terms in the differential expression model that represent the difference in cluster median expression values of each marker between samples with fixed_effect = "b" and fixed_effect = "a" and between samples with fixed_effect = "c" and fixed_effect = "a", respectively. In addition, note that the first row in ‘tested_effect‘ will always represent the "omnibus" test, or the test that there are significant differences between *any* levels of *any* fixed effect variable in the model.

The second column, ‘dea_results‘ is a list of tibbles in which each entry gives the differential expression results for each tested_effect. Within each entry of ‘dea_results‘, you will find ‘p_val‘, the p-value associated with each tested effect in each input cluster/marker pair; ‘p_adj‘, the multiple-comparison adjusted p-value (using the `p.adjust` function), and other values associated with the underlying method used to perform the differential expression analysis (such as the log-fold change of clusters’ median marker expression values between the conditions being compared). Each tibble in ‘dea_results‘ will also have two columns representing the cluster and marker corresponding to the p-value in each row.

See Also

Other differential expression analysis functions: `tof_analyze_expression()`, `tof_analyze_expression_lmm()`, `tof_analyze_expression_ttest()`

Examples

```
# For differential discovery examples, please see the package vignettes
NULL
```

`tof_analyze_expression_lmm` *Differential Expression Analysis (DEA) with linear mixed-models (LMMs)*

Description

This function performs differential expression analysis on the cell clusters contained within a ‘tof_tbl‘ using linear mixed-models. Users specify which columns represent sample, cluster, marker, fixed effect, and random effect information, and a (mixed) linear regression model is fit using either `lmer` or `glm`.

Usage

```
tof_analyze_expression_lmm(
  tof_tibble,
  sample_col,
  cluster_col,
  marker_cols = where(tof_is_numeric),
  fixed_effect_cols,
  random_effect_cols,
  central_tendency_function = median,
  min_cells = 3,
  min_samples = 5,
  alpha = 0.05
)
```

Arguments

`tof_tibble`	A ‘tof_tbl‘ or a ‘tibble‘.
`sample_col`	An unquoted column name indicating which column in ‘tof_tibble‘ represents the id of the sample from which each cell was collected. ‘sample_col‘ should serve as a unique identifier for each sample collected during data acquisition - all cells with the same value for ‘sample_col‘ will be treated as a part of the same observational unit.

`cluster_col` An unquoted column name indicating which column in ‘tof_tibble‘ stores the cluster ids of the cluster to which each cell belongs. Cluster labels can be produced via any method the user chooses - including manual gating, any of the functions in the ‘tof_cluster_*‘ function family, or any other method.

`marker_cols` Unquoted column names representing which columns in ‘tof_tibble‘ (i.e. which high-dimensional cytometry protein measurements) should be included in the differential discovery analysis. Defaults to all numeric (integer or double) columns. Supports tidyselection.

`fixed_effect_cols`
Unquoted column names representing which columns in ‘tof_tibble‘ should be used to model fixed effects during the differential expression analysis. Supports tidyselection.

Generally speaking, fixed effects should represent the comparisons of biological interest (often the the variables manipulated during experiments), such as treated vs. non-treated, before-treatment vs. after-treatment, or healthy vs. non-healthy.

`random_effect_cols`
Optional. Unquoted column names representing which columns in ‘tof_tibble‘ should be used to model random effects during the differential expression analysis. Supports tidyselection.

Generally speaking, random effects should represent variables that a researcher wants to control/account for, but that are not necessarily of biological interest. Example random effect variables might include batch id, patient id (in a paired design), or patient age. Most analyses will not include random effects.

`central_tendency_function`
The function that will be used to calculate the measurement of central tendency for each cluster/marker pair (to be used as the dependent variable in the linear model). Defaults to `median`.

`min_cells` An integer value used to filter clusters out of the differential expression analysis. Clusters are not included in the differential expression testing if they do not have at least ‘min_cells‘ in at least ‘min_samples‘ samples. Defaults to 3.

`min_samples` An integer value used to filter clusters out of the differential expression analysis. Clusters are not included in the differential expression testing if they do not have at least ‘min_cells‘ in at least ‘min_samples‘ samples. Defaults to 5.

`alpha` A numeric value between 0 and 1 indicating which significance level should be applied to multiple-comparison adjusted p-values during the differential abundance analysis. Defaults to 0.05.

Details

Specifically, one linear model is fit for each cluster/marker pair. For each cluster/marker pair, a user-supplied measurement of central tendency (‘central_tendency_function‘), such as mean or median, is calculated across all cells in the cluster on a sample-by-sample basis. Then, this central tendency value is used as the dependent variable in a linear model with ‘fixed_effect_cols‘ as fixed effects predictors and ‘random_effect_cols‘ as random effects predictors. Once all models (one per each cluster/marker pair) are fit, p-values for each coefficient in each model are multiple-comparisons adjusted using the `p.adjust` function.

Value

A nested tibble with two columns: ‘tested_effect‘ and ‘dea_results‘.

The first column, ‘tested_effect‘ is a character vector indicating which term in the differential expression model was used for significance testing. The values in this row are obtained by pasting together the column names for each fixed effect variable and each of its values. For example, a fixed effect column named fixed_effect with levels "a", "b", and "c" have two terms in ‘tested_effect‘: "fixed_effectb" and "fixed_effectc" (note that level "a" of fixed_effect is set as the reference level during dummy coding). These values correspond to the terms in the differential expression model that represent the difference in cluster median expression values of each marker between samples with fixed_effect = "b" and fixed_effect = "a" and between samples with fixed_effect = "c" and fixed_effect = "a", respectively. In addition, note that the first row in ‘tested_effect‘ will always represent the "omnibus" test, or the test that there were significant differences between any levels of any fixed effect variable in the model.

The second column, ‘dea_results‘ is a list of tibbles in which each entry gives the differential expression results for each tested_effect. Within each entry of ‘daa_results‘, you will find ‘p_val‘, the p-value associated with each tested effect in each input cluster/marker pair; ‘p_adj‘, the multiple-comparison adjusted p-value (using the `p.adjust` function), and other values associated with the underlying method used to perform the differential expression analysis (such as the log-fold change of clusters’ median marker expression values between the levels being compared).

See Also

Other differential expression analysis functions: `tof_analyze_expression()`, `tof_analyze_expression_diffcyt()`, `tof_analyze_expression_ttest()`

Examples

```
# For differential discovery examples, please see the package vignettes
NULL
```

---

tof_analyze_expression_ttest

Differential Expression Analysis (DEA) with t-tests

Description

This function performs differential expression analysis on the cell clusters contained within a ‘tof_tbl‘ using simple t-tests. Specifically, either an unpaired or paired t-test will compare samples’ marker expression distributions (between two conditions) within each cluster using a user-specified summary function (i.e. mean or median). One t-test is conducted per cluster/marker pair and significant differences between sample types are detected after multiple-hypothesis correction.

Usage

```
tof_analyze_expression_ttest(
  tof_tibble,
  cluster_col,
  marker_cols = where(tof_is_numeric),
  effect_col,
  group_cols,
  test_type = c("unpaired", "paired"),
  summary_function = mean,
  min_cells = 3,
  min_samples = 5,
  alpha = 0.05,
  quiet = FALSE
)
```

Arguments

tof_tibble	A ‘tof_tbl‘ or a ‘tibble‘.
cluster_col	An unquoted column name indicating which column in ‘tof_tibble‘ stores the cluster ids of the cluster to which each cell belongs. Cluster labels can be produced via any method the user chooses - including manual gating, any of the functions in the ‘tof_cluster_*‘ function family, or any other method.
marker_cols	Unquoted column names representing which columns in ‘tof_tibble‘ (i.e. which high-dimensional cytometry protein measurements) should be tested for differential expression between levels of the ‘effect_col‘. Defaults to all numeric (integer or double) columns. Supports tidyselect helpers.
effect_col	Unquoted column name representing which column in ‘tof_tibble‘ should be used to break samples into groups for the t-test. Should only have 2 unique values.
group_cols	Unquoted names of the columns other than ‘effect_col‘ that should be used to group cells into independent observations. Fills a similar role to ‘sample_col‘ in other ‘tof_analyze_abundance_*‘ functions. For example, if an experiment involves analyzing samples taken from multiple patients at two timepoints (with ‘effect_col = timepoint‘), then group_cols should be the name of the column representing patient IDs.
test_type	A string indicating whether the t-test should be "unpaired" (the default) or "paired".
summary_function	The vector-valued function that should be used to summarize the distribution of each marker in each cluster (within each sample, as grouped by ‘group_cols‘). Defaults to ‘mean‘.
min_cells	An integer value used to filter clusters out of the differential abundance analysis. Clusters are not included in the differential abundance testing if they do not have at least ‘min_cells‘ in at least ‘min_samples‘ samples. Defaults to 3.
min_samples	An integer value used to filter clusters out of the differential abundance analysis. Clusters are not included in the differential abundance testing if they do not have at least ‘min_cells‘ in at least ‘min_samples‘ samples. Defaults to 5.

`alpha` A numeric value between 0 and 1 indicating which significance level should be applied to multiple-comparison adjusted p-values during the differential abundance analysis. Defaults to 0.05.

`quiet` A boolean value indicating whether warnings should be printed. Defaults to 'TRUE'.

Value

A tibble with 7 columns:

{cluster_col} The name/ID of the cluster in the cluster/marker pair being tested. Each entry in this column will match a unique value in the input {cluster_col}.

marker The name of the marker in the cluster/marker pair being tested.

t The t-statistic computed for each cluster.

df The degrees of freedom used for the t-test for each cluster.

p_val The (unadjusted) p-value for the t-test for each cluster.

p_adj The `p.adjust`-adjusted p-value for the t-test for each cluster.

significant A character vector that will be "*" for clusters for which p_adj < alpha and "" otherwise.

mean_diff For an unpaired t-test, the difference between the average proportions of each cluster in the two levels of 'effect_col'. For a paired t-test, the average difference between the proportions of each cluster in the two levels of 'effect_col' within a given patient.

mean_fc For an unpaired t-test, the ratio between the average proportions of each cluster in the two levels of 'effect_col'. For a paired t-test, the average ratio between the proportions of each cluster in the two levels of 'effect_col' within a given patient. 0.001 is added to the denominator of the ratio to avoid divide-by-zero errors.

The "levels" attribute of the result indicates the order in which the different levels of the 'effect_col' were considered. The 'mean_diff' value for each row of the output is computed subtracting the second level from the first level, and the 'mean_fc' value for each row is computed by dividing the first level by the second level.

See Also

Other differential expression analysis functions: `tof_analyze_expression()`, `tof_analyze_expression_diffcyt()`, `tof_analyze_expression_lmm()`

Examples

```
# For differential discovery examples, please see the package vignettes
NULL
```

`tof_annotate_clusters` *Manually annotate tidytof-computed clusters using user-specified labels*

Description

This function adds an additional column to a ‘tibble‘ or ‘tof_tbl‘ to allow users to incorporate manual cell type labels for clusters identified using unsupervised algorithms.

Usage

```
tof_annotate_clusters(tof_tibble, cluster_col, annotations)
```

Arguments

`tof_tibble`	‘tof_tbl‘ or ‘tibble‘.
`cluster_col`	An unquoted column name indicating which column in ‘tof_tibble‘ contains the ids of the unsupervised cluster to which each cell belongs. Cluster labels can be produced via any method the user chooses - including manual gating, any of the functions in the ‘tof_cluster_*‘ function family, or any other method.
`annotations`	A data structure indicating how to annotate each cluster id in ‘cluster_col‘. ‘annotations‘ can be provided as a data.frame with two columns (the first should have the same name as ‘cluster_col‘ and contain each unique cluster id; the second can have any name and should contain a character vector indicating which manual annotation should be matched with each cluster id in the first column). ‘annotations‘ can also be provided as a named character vector; in this case, each entry in ‘annotations‘ should be a unique cluster id, and the names for each entry should be the corresponding manual cluster annotation. See below for examples.

Value

A ‘tof_tbl‘ with the same number of rows as ‘tof_tibble‘ and one additional column containing the manual cluster annotations for each cell (as a character vector). If ‘annotations‘ was provided as a data.frame, the new column will have the same name as the column containing the cluster annotations in ‘annotations‘. If ‘annotations‘ was provided as a named character vector, the new column will be named ‘{cluster_col}_annotation‘.

Examples

```
sim_data <-
    dplyr::tibble(
        cd45 = rnorm(n = 1000),
        cd38 = c(rnorm(n = 500), rnorm(n = 500, mean = 2)),
        cd34 = c(rnorm(n = 500), rnorm(n = 500, mean = 4)),
        cd19 = rnorm(n = 1000),
        cluster_id = c(rep("a", 500), rep("b", 500))
```

```
    )

# using named character vector
sim_data |>
    tof_annotate_clusters(
        cluster_col = cluster_id,
        annotations = c("macrophage" = "a", "dendritic cell" = "b")
    )

# using two-column data.frame
annotation_data_frame <-
    data.frame(
        cluster_id = c("a", "b"),
        cluster_annotation = c("macrophage", "dendritic cell")
    )

sim_data |>
    tof_annotate_clusters(
        cluster_col = cluster_id,
        annotations = annotation_data_frame
    )
```

`tof_apply_classifier` *Perform developmental clustering on CyTOF data using a pre-fit classifier*

Description

Perform developmental clustering on CyTOF data using a pre-fit classifier

Usage

```
tof_apply_classifier(
  cancer_tibble = NULL,
  classifier_fit = NULL,
  distance_function = c("mahalanobis", "cosine", "pearson"),
  num_cores = 1,
  parallel_vars
)
```

Arguments

`cancer_tibble`	A ‘tibble‘ or ‘tof_tibble‘ containing cells to be classified into their nearest healthy subpopulation (generally cancer cells).
`classifier_fit`	A nested ‘tibble‘ produced by ‘tof_build_classifier‘ in which each row represents a healthy cell subpopulation into which the cells in ‘cancer_tibble‘ should be classified using minimum distance.

distance_function
: A string indicating which distance function should be used to perform the classification. Options are "mahalanobis" (the default), "cosine", and "pearson".

num_cores
: An integer indicating the number of CPU cores used to parallelize the classification. Defaults to 1 (a single core).

parallel_vars
: Unquoted column names indicating which columns in ‘cancer_tibble‘ to use for breaking up the data in order to parallelize the classification. Defaults to NULL. Supports tidyselect helpers.

Value

A tibble with ‘nrow(cancer_tibble)‘ rows and ‘nrow(classifier_fit) + 1‘ columns. Each row represents a cell from ‘cancer_tibble‘, and ‘nrow(classifier_fit)‘ of the columns represent the distance between the cell and each of the healthy subpopulations’ cluster centroids. The final column represents the cluster id of the healthy subpopulation with the minimum distance to the cell represented by that row.

Examples

```
NULL
```

---

tof_assess_channels *Detect low-expression (i.e. potentially failed) channels in high-dimensional cytometry data*

Description

Detect low-expression (i.e. potentially failed) channels in high-dimensional cytometry data

Usage

```
tof_assess_channels(
  tof_tibble,
  channel_cols = where(tof_is_numeric),
  negative_threshold = asinh(10/5),
  negative_proportion_flag = 0.95
)
```

Arguments

tof_tibble
: A ‘tof_tbl‘ or ‘tibble‘.

channel_cols
: A vector of unquoted column names representing columns that contain single-cell protein measurements. Supports tidyselect helpers. If nothing is specified, the default is to analyze all numeric columns.

negative_threshold
: A scalar indicating the threshold below which a measurement should be considered negative. Defaults to the hyperbolic arcsine transformation of 10 counts.

negative_proportion_flag
: A scalar between 0 and 1 indicating the proportion of cells in tof_tibble that need to be below ‘negative_threshold‘ for a given marker in order for that marker to be flagged. Defaults to 0.95.

Value

A tibble 3 columns and a number of rows equal to the number of columns in ‘tof_tibble‘ chosen by ‘channel_cols‘. The three columns are "channel", a character vector of channel names, "negative_proportion", a numeric vector with values between 0 and 1 indicating how many cells in ‘tof_tibble‘ below ‘negative_threshold‘ for each channel, and ‘flagged_channel‘, a boolean vector indicating whether or not a channel has been flagged as potentially failed (TRUE means that the channel had a large number of cells below ‘negative_threshold‘).

Examples

```
# simulate some data
sim_data <-
    data.frame(
        cd4 = rnorm(n = 100, mean = 5, sd = 0.5),
        cd8 = rnorm(n = 100, mean = 0, sd = 0.1),
        cd33 = rnorm(n = 100, mean = 10, sd = 0.1)
    )

tof_assess_channels(tof_tibble = sim_data)

tof_assess_channels(tof_tibble = sim_data, channel_cols = c(cd4, cd8))

tof_assess_channels(tof_tibble = sim_data, negative_threshold = 2)
```

---

tof_assess_clusters_distance

Assess a clustering result by calculating the z-score of each cell's mahalanobis distance to its cluster centroid and flagging outliers.

Description

This function evaluates the result of a clustering procedure by comparing the mahalanobis distance between each cell and the centroid of the cluster to which it was assigned among all cells in a given cluster. All cells with a mahalanobis-distance z-score above a user-specified threshold are flagged as potentially anomalous. Note that the z-score is calculated using a modified formula to minimize the effect of outliers (Z = x - median(x) / mad(x)).

Usage

```
tof_assess_clusters_distance(
  tof_tibble,
  cluster_col,
  marker_cols = where(tof_is_numeric),
  z_threshold = 3,
  augment = FALSE
)
```

Arguments

tof_tibble	A ‘tof_tbl‘ or ‘tibble‘.
cluster_col	An unquoted column name indicating which column in ‘tof_tibble‘ stores the cluster ids for the cluster to which each cell belongs. Cluster labels can be produced via any method the user chooses - including manual gating, any of the functions in the ‘tof_cluster_*‘ function family, or any other method.
marker_cols	Unquoted column names indicating which column in ‘tof_tibble‘ should be interpreted as markers to be used in the mahalanobis distance calculation. Defaults to all numeric columns. Supports tidyselection.
z_threshold	A scalar indicating the distance z-score threshold above which a cell should be considered anomalous. Defaults to 3.
augment	A boolean value indicating if the output should column-bind the computed flags for each cell (see below) as new columns in ‘tof_tibble‘ (TRUE) or if a tibble including only the computed flags should be returned (FALSE, the default).

Value

If augment = FALSE (the default), a tibble with 3 columns: ".mahalanobis_distance" (the mahalanobis distance from each cell to the centroid of tits assigned cluster), "z_score" (the modified z-score of each cell's mahalanobis distance relative to all other cells in the dataset), and "flagged_cell" (a boolean indicating whether or not each cell was flagged as having a z-score above z_threshold). If augment = TRUE, the same 3 columns will be column-bound to tof_tibble, and the resulting tibble will be returned.

Examples

```
# simulate data
sim_data_inner <-
    dplyr::tibble(
        cd45 = c(rnorm(n = 600), rnorm(n = 500, mean = -4)),
        cd38 =
            c(
                rnorm(n = 100, sd = 0.5),
                rnorm(n = 500, mean = -3),
                rnorm(n = 500, mean = 8)
            ),
        cd34 =
            c(
```

```
                rnorm(n = 100, sd = 0.2, mean = -10),
                rnorm(n = 500, mean = 4),
                rnorm(n = 500, mean = 60)
            ),
        cd19 = c(rnorm(n = 100, sd = 0.3, mean = 10), rnorm(n = 1000)),
        cluster_id = c(rep("a", 100), rep("b", 500), rep("c", 500)),
        dataset = "inner"
    )

sim_data_outer <-
    dplyr::tibble(
        cd45 = c(rnorm(n = 10), rnorm(50, mean = 3), rnorm(n = 50, mean = -12)),
        cd38 =
            c(
                rnorm(n = 10, sd = 0.5),
                rnorm(n = 50, mean = -10),
                rnorm(n = 50, mean = 10)
            ),
        cd34 =
            c(
                rnorm(n = 10, sd = 0.2, mean = -15),
                rnorm(n = 50, mean = 15),
                rnorm(n = 50, mean = 70)
            ),
        cd19 = c(rnorm(n = 10, sd = 0.3, mean = 19), rnorm(n = 100)),
        cluster_id = c(rep("a", 10), rep("b", 50), rep("c", 50)),
        dataset = "outer"
    )

sim_data <- rbind(sim_data_inner, sim_data_outer)

# detect anomalous cells (in this case, the "outer" dataset contains small
# clusters that get lumped into the larger clusters in the "inner" dataset)
z_result <-
    sim_data |>
    tof_assess_clusters_distance(cluster_col = cluster_id, z_threshold = 2.5)
```

`tof_assess_clusters_entropy` *Assess a clustering result by calculating the shannon entropy of each cell's mahalanobis distance to all cluster centroids and flagging outliers.*

Description

This function evaluates the result of a clustering procedure by calculating the mahalanobis distance between each cell and the centroids of all clusters in the dataset and finding the shannon entropy of the resulting vector of distances. All cells with an entropy threshold above a user-specified threshold are flagged as potentially anomalous. Entropy is minimized (to 0) when a cell is close to one (or a

small number) of clusters, but far from the rest of them. If a cell is close to multiple cluster centroids (i.e. has an ambiguous phenotype), its entropy will be large.

Usage

```
tof_assess_clusters_entropy(
  tof_tibble,
  cluster_col,
  marker_cols = where(tof_is_numeric),
  entropy_threshold,
  entropy_quantile = 0.9,
  num_closest_clusters,
  augment = FALSE
)
```

Arguments

tof_tibble	A 'tof_tbl' or 'tibble'.
cluster_col	An unquoted column name indicating which column in 'tof_tibble' stores the cluster ids for the cluster to which each cell belongs. Cluster labels can be produced via any method the user chooses - including manual gating, any of the functions in the 'tof_cluster_*' function family, or any other method.
marker_cols	Unquoted column names indicating which column in 'tof_tibble' should be interpreted as markers to be used in the mahalanobis distance calculation. Defaults to all numeric columns. Supports tidyselection.
entropy_threshold	A scalar indicating the entropy threshold above which a cell should be considered anomalous. If unspecified, a threshold will be computed using 'entropy_quantile' (see below). (Note: Entropy is often between 0 and 1, but can be larger with many classes/clusters).
entropy_quantile	A scalar between 0 and 1 indicating the entropy quantile above which a cell should be considered anomalous. Defaults to 0.9, which means that cells with an entropy above the 90th percentile will be flagged. Ignored if entropy_threshold is specified directly.
num_closest_clusters	An integer indicating how many of a cell's closest cluster centroids should have their mahalanobis distance included in the entropy calculation. Playing with this argument will allow you to ignore distances to clusters that are far away from each cell (and thus may distort the result, as many distant centroids with large distances can artificially inflate a cells' entropy value; that being said, this is rarely an issue empirically). Defaults to all clusters in tof_tibble.
augment	A boolean value indicating if the output should column-bind the computed flags for each cell (see below) as new columns in 'tof_tibble' (TRUE) or if a tibble including only the computed flags should be returned (FALSE, the default).

Value

If augment = FALSE (the default), a tibble with 2 + NUM_CLUSTERS columns. where NUM_CLUSTERS is the number of unique clusters in cluster_col. Two of the columns will be "entropy" (the entropy value for each cell) and "flagged_cell" (a boolean value indicating if each cell had an entropy value above entropy_threshold). The other NUM_CLUSTERS columns will contain the mahalanobis distances from each cell to each of the clusters in cluster_col (named ".mahalanobis_{cluster_name}"). If augment = TRUE, the same 2 + NUM_CLUSTERS columns will be column-bound to tof_tibble, and the resulting tibble will be returned.

Examples

```
# simulate data
sim_data <-
    dplyr::tibble(
     cd45 = c(rnorm(n = 1000, sd = 1.5), rnorm(n = 1000, mean = 2), rnorm(n = 1000, mean = -2)),
     cd38 = c(rnorm(n = 1000, sd = 1.5), rnorm(n = 1000, mean = 2), rnorm(n = 1000, mean = -2)),
     cd34 = c(rnorm(n = 1000, sd = 1.5), rnorm(n = 1000, mean = 2), rnorm(n = 1000, mean = -2)),
     cd19 = c(rnorm(n = 1000, sd = 1.5), rnorm(n = 1000, mean = 2), rnorm(n = 1000, mean = -2)),
        cluster_id = c(rep("a", 1000), rep("b", 1000), rep("c", 1000))
    )

# imagine a "reference" dataset in which "cluster a" isn't present
sim_data_reference <-
    sim_data |>
    dplyr::filter(cluster_id %in% c("b", "c"))

# if we cluster into the reference dataset, we will force all cells in
# cluster a into a population where they don't fit very well
sim_data <-
    sim_data |>
    tof_cluster(
        healthy_tibble = sim_data_reference,
        healthy_label_col = cluster_id,
        method = "ddpr"
    )

# we can evaluate the clustering quality by calculating by the entropy of the
# mahalanobis distance vector for each cell to all cluster centroids
entropy_result <-
    sim_data |>
    tof_assess_clusters_entropy(
        cluster_col = .mahalanobis_cluster,
        marker_cols = starts_with("cd"),
        entropy_quantile = 0.8,
        augment = TRUE
    )

# most cells in "cluster a" are flagged, and few cells in the other clusters are
flagged_cluster_proportions <-
    entropy_result |>
    dplyr::group_by(cluster_id) |>
```

```
    dplyr::summarize(
        prop_flagged = mean(flagged_cell)
    )
```

`tof_assess_clusters_knn` *Assess a clustering result by calculating a cell's cluster assignment to that of its K nearest neighbors.*

Description

This function evaluates the result of a clustering procedure by finding the cell's K nearest neighbors, determining which cluster the majority of them are assigned to, and checking if this matches the cell's own cluster assignment. If the cluster assignment of the majority of a cell's nearest neighbors does not match with the cell's own cluster assignment, the cell is flagged as potentially anomalous.

Usage

```
tof_assess_clusters_knn(
  tof_tibble,
  cluster_col,
  marker_cols = where(tof_is_numeric),
  num_neighbors = min(10, nrow(tof_tibble)),
  distance_function = c("euclidean", "cosine", "l2", "ip"),
  augment = FALSE
)
```

Arguments

`tof_tibble`	A 'tof_tbl' or 'tibble'.
`cluster_col`	An unquoted column name indicating which column in 'tof_tibble' stores the cluster ids for the cluster to which each cell belongs. Cluster labels can be produced via any method the user chooses - including manual gating, any of the functions in the 'tof_cluster_*' function family, or any other method.
`marker_cols`	Unquoted column names indicating which column in 'tof_tibble' should be interpreted as markers to be used in the mahalanobis distance calculation. Defaults to all numeric columns. Supports tidyselection.
`num_neighbors`	An integer indicating how many neighbors should be found during the nearest neighbor calculation.
`distance_function`	A string indicating which distance function should be used to perform the k nearest neighbor calculation. Options are "euclidean" (the default) and "cosine".
`augment`	A boolean value indicating if the output should column-bind the computed flags for each cell (see below) as new columns in 'tof_tibble' (TRUE) or if a tibble including only the computed flags should be returned (FALSE, the default).

Value

If augment = FALSE (the default), a tibble with 2 columns: ".knn_cluster" (a character vector indicating which cluster received the majority vote of each cell's k nearest neighbors) and "flagged_cell" (a boolean value indicating if the cell's cluster assignment matched the majority vote (TRUE) or not (FALSE)). If augment = TRUE, the same 2 columns will be column-bound to tof_tibble, and the resulting tibble will be returned.

Examples

```
sim_data <-
    dplyr::tibble(
     cd45 = c(rnorm(n = 1000, sd = 1.5), rnorm(n = 1000, mean = 2), rnorm(n = 1000, mean = -2)),
     cd38 = c(rnorm(n = 1000, sd = 1.5), rnorm(n = 1000, mean = 2), rnorm(n = 1000, mean = -2)),
     cd34 = c(rnorm(n = 1000, sd = 1.5), rnorm(n = 1000, mean = 2), rnorm(n = 1000, mean = -2)),
     cd19 = c(rnorm(n = 1000, sd = 1.5), rnorm(n = 1000, mean = 2), rnorm(n = 1000, mean = -2)),
        cluster_id = c(rep("a", 1000), rep("b", 1000), rep("c", 1000))
    )

knn_result <-
    sim_data |>
    tof_assess_clusters_knn(
        cluster_col = cluster_id,
        num_neighbors = 10
    )
```

---

tof_assess_flow_rate *Detect flow rate abnormalities in high-dimensional cytometry data*

Description

This function performs a simplified version of flowAI's statistical test to detect time periods with abnormal flow rates over the course of a flow cytometry experiment. Briefly, the relative flow rates for each timestep throughout data acquisition are calculated (see tof_calculate_flow_rate), and outlier timepoints with particularly high or low flow rates (i.e. those beyond extreme values of the t-distribution across timesteps) are flagged.

Usage

```
tof_assess_flow_rate(
  tof_tibble,
  time_col,
  group_cols,
  num_timesteps = nrow(tof_tibble)/1000,
  alpha_threshold = 0.01,
  visualize = FALSE,
  ...,
  augment = FALSE
)
```

Arguments

tof_tibble	A ‘tof_tbl‘ or ‘tibble‘.
time_col	An unquoted column name indicating which column in ‘tof_tibble‘ contains the time at which each cell was collected.
group_cols	Optional. Unquoted column names indicating which columns should be used to group cells before analysis. Flow rate calculation is then performed independently within each group. Supports tidyselect helpers.
num_timesteps	The number of bins into which ‘time_col‘ should be split. to define "timesteps" of the data collection process. The number of cells analyzed by the cytometer will be counted in each bin separately and will represent the relative average flow rate for that timestep in data collection.
alpha_threshold	A scalar between 0 and 1 indicating the two-tailed significance level at which to draw outlier thresholds in the t-distribution with ‘num_timesteps‘ - 1 degrees of freedom. Defaults to 0.01.
visualize	A boolean value indicating if a plot should be generated to visualize each timestep’s relative flow rate (by group) instead of returning the tibble directly. Defaults to FALSE.
...	Optional additional arguments to pass to facet_wrap. Ignored if visualize = FALSE.
augment	A boolean value indicating if the output should column-bind the computed flags for each cell (see below) as new columns in ‘tof_tibble‘ (TRUE) or if a tibble including only the computed flags should be returned (FALSE, the default).

Value

A tibble with the same number of rows as ‘tof_tibble‘. If augment = FALSE (the default), it will have 3 columns: "{time_col}" (the same column as ‘time_col‘), "timestep" (the numeric timestep to which each cell was assigned based on its value for ‘time_col‘), and "flagged_window" (a boolean vector indicating if each cell was collecting during a timestep flagged for having a high or low flow rate). If augment = TRUE, these 3 columns will be column-bound to ‘tof_tibble‘ to return an augmented version of the input dataset. (Note that in this case, time_col will not be duplicated). If visualize = TRUE, then a ggplot object is returned instead of a tibble.

Examples

```
set.seed(1000L)
sim_data <-
    data.frame(
        cd4 = rnorm(n = 1000, mean = 5, sd = 0.5),
        cd8 = rnorm(n = 1000, mean = 0, sd = 0.1),
        cd33 = rnorm(n = 1000, mean = 10, sd = 0.1),
        file_name = c(rep("a", times = 500), rep("b", times = 500)),
        time =
            c(
                sample(1:100, size = 200, replace = TRUE),
                sample(100:400, size = 300, replace = TRUE),
```

```
                sample(1:150, size = 400, replace = TRUE),
                sample(1:500, size = 100, replace = TRUE)
            )
    )

sim_data |>
    tof_assess_flow_rate(
        time_col = time,
        num_timesteps = 20,
        visualize = TRUE
    )

sim_data |>
    tof_assess_flow_rate(
        time_col = time,
        group_cols = file_name,
        num_timesteps = 20,
        visualize = TRUE
    )
```

`tof_assess_flow_rate_tibble` *Detect flow rate abnormalities in high-dimensional cytometry data (stored in a single data.frame)*

Description

This function performs a simplified version of flowAI's statistical test to detect time periods with abnormal flow rates over the course of a flow cytometry experiment. Briefly, the relative flow rates for each timestep throughout data acquisition are calculated (see tof_calculate_flow_rate), and outlier timepoints with particularly high or low flow rates (i.e. those beyond extreme values of the t-distribution across timesteps) are flagged.

Usage

```
tof_assess_flow_rate_tibble(
  tof_tibble,
  time_col,
  num_timesteps = nrow(tof_tibble)/1000,
  alpha_threshold = 0.01,
  augment = FALSE
)
```

Arguments

`tof_tibble`	A 'tof_tbl' or 'tibble'.
`time_col`	An unquoted column name indicating which column in 'tof_tibble' contains the time at which each cell was collected.

num_timesteps
: The number of bins into which ‘time_col‘ should be split. to define "timesteps" of the data collection process. The number of cells analyzed by the cytometer will be counted in each bin separately and will represent the relative average flow rate for that timestep in data collection.

alpha_threshold
: A scalar between 0 and 1 indicating the two-tailed significance level at which to draw outlier thresholds in the t-distribution with ‘num_timesteps‘ - 1 degrees of freedom. Defaults to 0.01.

augment
: A boolean value indicating if the output should column-bind the computed flags for each cell (see below) as new columns in ‘tof_tibble‘ (TRUE) or if a tibble including only the computed flags should be returned (FALSE, the default).

Value

A tibble with the same number of rows as ‘tof_tibble‘. If augment = FALSE (the default), it will have 3 columns: "{time_col}" (the same column as ‘time_col‘), "timestep" (the numeric timestep to which each cell was assigned based on its value for ‘time_col‘), and "flagged_window" (a boolean vector indicating if each cell was collecting during a timestep flagged for having a high or low flow rate). If augment = TRUE, these 3 columns will be column-bound to ‘tof_tibble‘ to return an augmented version of the input dataset. (Note that in this case, time_col will not be duplicated).

Examples

```
set.seed(1000L)
sim_data <-
    data.frame(
        cd4 = rnorm(n = 1000, mean = 5, sd = 0.5),
        cd8 = rnorm(n = 1000, mean = 0, sd = 0.1),
        cd33 = rnorm(n = 1000, mean = 10, sd = 0.1),
        time =
            c(
                sample(1:100, size = 200, replace = TRUE),
                sample(100:400, size = 300, replace = TRUE),
                sample(1:150, size = 400, replace = TRUE),
                sample(1:500, size = 100, replace = TRUE)
            )
    )

sim_data |>
    tof_assess_flow_rate(
        time_col = time,
        num_timesteps = 20,
        visualize = TRUE
    )
```

`tof_assess_model` *Assess a trained elastic net model*

Description

This function assesses a trained 'tof_model''s performance on new data by computing model type-specific performance measurements. If new data isn't provided, performance metrics for the training data will be provided.

Usage

```
tof_assess_model(tof_model, new_data)
```

Arguments

`tof_model`	A 'tof_model' trained using `tof_train_model`
`new_data`	A tibble of new observations that should be used to evaluate the 'tof_model''s performance. If new_data isn't provided, model evaluation will will be performed using the training data used to fit the model. Alternatively, the string "tuning" can be provided to access the model's performance metrics during the (resampled) model tuning process.

Value

A list of performance metrics whose components depend on the model type:

"model_metrics" A tibble with two columns ("metric" and "value") containing standard performance metrics for each model type. For linear models, the "mse" (the mean squared error of the predictions) and "mae" (the mean absolute error of the predictions). For two-class models, "roc_auc" (the area under the Receiver-Operating Curve for the classification), "misclassification error" (the proportion of misclassified observations), "binomial_deviance" (see `deviance.glmnet`), "mse" (the mean squared error of the logit function), and "mae" (the mean absolute error of the logit function). For multiclass models, "roc_auc" (the area under the Receiver-Operating Curve for the classification using the Hand-Till generalization of the ROC AUC for multiclass models in `roc_auc`), "misclassification error" (the proportion of misclassified observations), "multinomial_deviance" (see `deviance.glmnet`), and "mse" and "mae" as above. For survival models, "concordance_index" (Harrel's C index; see `deviance.glmnet`) and "partial_likelihood_deviance" (see `deviance.glmnet`).

"roc_curve" Reported only for "two-class" and "multiclass" models. For both, a tibble is provided reporting the true-positive rate (tpr) and false-positive rate (fpr) at each threshold for classification for use in plotting a receiver-operating curve. For "multiclass" models, the ".level" column allows for separating the values in roc_curve such that one ROC can be plotted for each class.

"confusion_matrix" Reported only for "two-class" and "multiclass" models. For both, a tibble is provided reporting the "confusion matrix" of the classification in long-format.

"survival_curves" Reported only for "survival" models. A tibble indicating each patient's probability of survival (1 - probability(event)) at each timepoint in the dataset and whether each sample was placed in the "high" or "low" risk group according to its predicted relative risk (and the tof_model's optimal relative_risk cutoff in the training dataset).

See Also

Other modeling functions: tof_create_grid(), tof_predict(), tof_split_data(), tof_train_model()

Examples

```
feature_tibble <-
    dplyr::tibble(
        sample = as.character(1:100),
        cd45 = runif(n = 100),
        pstat5 = runif(n = 100),
        cd34 = runif(n = 100),
        outcome = (3 * cd45) + (4 * pstat5) + rnorm(100)
    )

new_tibble <-
    dplyr::tibble(
        sample = as.character(1:20),
        cd45 = runif(n = 20),
        pstat5 = runif(n = 20),
        cd34 = runif(n = 20),
        outcome = (3 * cd45) + (4 * pstat5) + rnorm(20)
    )

split_data <- tof_split_data(feature_tibble, split_method = "simple")

# train a regression model
regression_model <-
    tof_train_model(
        split_data = split_data,
        predictor_cols = c(cd45, pstat5, cd34),
        response_col = outcome,
        model_type = "linear"
    )

# assess the model on new data
tof_assess_model(tof_model = regression_model, new_data = new_tibble)
```

---

tof_assess_model_new_data

Compute a trained elastic net model's performance metrics using new_data.

Description

Compute a trained elastic net model's performance metrics using new_data.

Usage

```
tof_assess_model_new_data(tof_model, new_data)
```

Arguments

tof_model	A 'tof_model' trained using tof_train_model
new_data	A tibble of new observations that should be used to evaluate the 'tof_model''s performance.

Value

A list of performance metrics whose components depend on the model type.

`tof_assess_model_tuning` *Access a trained elastic net model's performance metrics using its tuning data.*

Description

Access a trained elastic net model's performance metrics using its tuning data.

Usage

```
tof_assess_model_tuning(tof_model)
```

Arguments

tof_model	A 'tof_model' trained using tof_train_model

Value

A list of performance metrics whose components depend on the model type.

`tof_batch_correct` *Perform groupwise linear rescaling of high-dimensional cytometry measurements*

Description

This function performs quantile normalization on high-dimensional cytometry data in tidy format using either linear rescaling or quantile normalization. Each channel specified by ‘channel_cols‘ is batch corrected, and ‘group_cols‘ can be used to break cells into groups for which the batch correction should be performed separately.

Usage

```
tof_batch_correct(
  tof_tibble,
  channel_cols,
  group_cols,
  augment = TRUE,
  method = c("rescale", "quantile")
)
```

Arguments

`tof_tibble`	A ‘tof_tbl‘ or a ‘tibble‘.
`channel_cols`	Unquoted column names representing columns that contain single-cell protein measurements. Supports tidyselect helpers.
`group_cols`	Optional. Unquoted column names indicating which columns should be used to group cells before batch correction. Batch correction is then performed independently within each group. Supports tidyselect helpers.
`augment`	A boolean value indicating if the output should replace the ‘channel_cols‘ in ‘tof_tibble‘ with the new, batch corrected columns (TRUE, the default) or if it should only return the batch-corrected columns (FALSE) with all other columns omitted.
`method`	A string indicating which batch correction method should be used. Valid options are "rescale" for linear scaling (the default) and "quantile" for quantile normalization using `normalize.quantiles`.

Value

If augment = TRUE, a tibble with the same number of rows and columns as tof_tibble, with the columns specified by ‘channel_cols‘ batch-corrected. If augment = FALSE, a tibble containing only the batch-corrected ‘channel_cols‘.

Examples

```
NULL
```

tof_batch_correct_quantile *Batch-correct a tibble of high-dimensional cytometry data using quantile normalization.*

Description

This function performs quantile normalization on high-dimensional cytometry data in tidy format using `normalize.quantiles`. Optionally, groups can be specified and normalized separately.

Usage

```
tof_batch_correct_quantile(
  tof_tibble,
  channel_cols,
  group_cols,
  augment = TRUE
)
```

Arguments

`tof_tibble`	A ‘tof_tbl‘ or a ‘tibble‘.
`channel_cols`	Unquoted column names representing columns that contain single-cell protein measurements. Supports tidyselect helpers.
`group_cols`	Optional. Unquoted column names indicating which columns should be used to group cells before batch correction. Batch correction is then performed independently within each group. Supports tidyselect helpers.
`augment`	A boolean value indicating if the output should replace the ‘channel_cols‘ in ‘tof_tibble‘ with the new, batch corrected columns (TRUE, the default) or if it should only return the batch-corrected columns (FALSE) with all other columns omitted.

Value

If augment = TRUE, a tibble with the same number of rows and columns as tof_tibble, with the columns specified by ‘channel_cols‘ batch-corrected. If augment = FALSE, a tibble containing only the batch-corrected ‘channel_cols‘.

Examples

```
NULL
```

`tof_batch_correct_quantile_tibble` *Batch-correct a tibble of high-dimensional cytometry data using quantile normalization.*

Description

This function performs quantile normalization on high-dimensional cytometry data in tidy format using `normalize.quantiles`.

Usage

```
tof_batch_correct_quantile_tibble(tof_tibble, channel_cols, augment = TRUE)
```

Arguments

`tof_tibble`	A 'tof_tbl' or a 'tibble'.
`channel_cols`	Unquoted column names representing columns that contain single-cell protein measurements. Supports tidyselect helpers.
`augment`	A boolean value indicating if the output should replace the 'channel_cols' in 'tof_tibble' with the new, batch corrected columns (TRUE, the default) or if it should only return the batch-corrected columns (FALSE) with all other columns omitted.

Value

If augment = TRUE, a tibble with the same number of rows and columns as tof_tibble, with the columns specified by 'channel_cols' batch-corrected. If augment = FALSE, a tibble containing only the batch-corrected 'channel_cols'.

Examples

```
NULL
```

`tof_batch_correct_rescale` *Perform groupwise linear rescaling of high-dimensional cytometry measurements*

Description

This function performs quantile normalization on high-dimensional cytometry data in tidy format using linear rescaling. Each channel specified by 'channel_cols' is rescaled such that the maximum value is 1 and the minimum value is 0. 'group_cols' specifies the columns that should be used to break cells into groups in which the rescaling should be performed separately.

Usage

```
tof_batch_correct_rescale(tof_tibble, channel_cols, group_cols, augment = TRUE)
```

Arguments

tof_tibble	A ‘tof_tbl‘ or a ‘tibble‘.
channel_cols	Unquoted column names representing columns that contain single-cell protein measurements. Supports tidyselect helpers.
group_cols	Optional. Unquoted column names indicating which columns should be used to group cells before batch correction. Batch correction is then performed independently within each group. Supports tidyselect helpers.
augment	A boolean value indicating if the output should replace the ‘channel_cols‘ in ‘tof_tibble‘ with the new, batch corrected columns (TRUE, the default) or if it should only return the batch-corrected columns (FALSE) with all other columns omitted.

Value

If augment = TRUE, a tibble with the same number of rows and columns as tof_tibble, with the columns specified by ‘channel_cols‘ batch-corrected. If augment = FALSE, a tibble containing only the batch-corrected ‘channel_cols‘.

Examples

```
NULL
```

---

tof_build_classifier *Calculate centroids and covariance matrices for each cell subpopulation in healthy CyTOF data.*

Description

This function takes a ‘tibble‘ or ‘tof_tibble‘ storing healthy cell measurements in each of its rows and a vector (‘healthy_cell_labels‘) representing the cell subpopulation to which each cell belongs. It uses these values to calculate several values required to perform "developmental classification" as described in this paper.

Usage

```
tof_build_classifier(
  healthy_tibble = NULL,
  healthy_cell_labels = NULL,
  classifier_markers = where(tof_is_numeric),
  verbose = FALSE
)
```

Arguments

`healthy_tibble`	A ‘tibble‘ or ‘tof_tibble‘ containing cells from only healthy control samples (i.e. not disease samples).
`healthy_cell_labels`	A character or integer vector of length ‘nrow(healthy_tibble)‘. Each entry in this vector should represent the cell subpopulation label (or cluster id) for the corresponding row in ‘healthy_tibble‘.
`classifier_markers`	Unquoted column names indicating which columns in ‘healthy_tibble‘ to use in the developmental classification. Defaults to all numeric columns in ‘healthy_tibble‘. Supports tidyselect helpers.
`verbose`	A boolean value indicating if updates should be printed to the console during classification. Defaults to FALSE.

Value

A tibble with three columns: **population** (id of the healthy cell population), **centroid** (the centroid vector for that cell population), and **covariance_matrix** (the covariance matrix for that cell population)

`tof_calculate_flow_rate` *Calculate the relative flow rates of different timepoints throughout a flow or mass cytometry run.*

Description

Calculate the relative flow rates of different timepoints throughout a flow or mass cytometry run.

Usage

```
tof_calculate_flow_rate(
  tof_tibble,
  time_col,
  num_timesteps = nrow(tof_tibble)/1000
)
```

Arguments

`tof_tibble`	A ‘tof_tbl‘ or ‘tibble‘.
`time_col`	An unquoted column name indicating which column in ‘tof_tibble‘ contains the time at which each cell was collected.
`num_timesteps`	The number of bins into which ‘time_col‘ should be split. to define "timesteps" of the data collection process. The number of cells analyzed by the cytometer will be counted in each bin separately and will represent the relative average flow rate for that timestep in data collection.

Value

A tibble with 3 columns and num_timesteps rows. Each row will represent a single timestep (and an error will be thrown if ‘num_timesteps‘ is larger than the number of rows in ‘tof_tibble‘). The three columns are as follows: "timestep", a numeric vector indicating which timestep is represented by a given row; "time_window", a factor showing the interval in ‘time_col‘ over which "timestep" is defined; and "num_cells", the number of cells that were collected during each timestep.

Examples

```
# simulate some data
sim_data <-
    data.frame(
        cd4 = rnorm(n = 100, mean = 5, sd = 0.5),
        cd8 = rnorm(n = 100, mean = 0, sd = 0.1),
        cd33 = rnorm(n = 100, mean = 10, sd = 0.1),
        time = sample(1:300, size = 100)
    )

tof_calculate_flow_rate(tof_tibble = sim_data, time_col = time, num_timesteps = 20L)
```

`tof_check_model_args` *Check argument specifications for a glmnet model.*

Description

Check argument specifications for a glmnet model.

Usage

```
tof_check_model_args(
  split_data,
  model_type = c("linear", "two-class", "multiclass", "survival"),
  best_model_type = c("best", "best with sparsity"),
  response_col,
  time_col,
  event_col
)
```

Arguments

`split_data`	An ‘rsplit‘ or ‘rset‘ object from the `rsample` package containing the sample-level data to use for modeling. Alternatively, an unsplit tbl_df can be provided, though this is not recommended.
`model_type`	A string indicating which kind of elastic net model to build. If a continuous response is being predicted, use "linear" for linear regression; if a categorical response with only 2 classes is being predicted, use "two-class" for logistic regression; if a categorical response with more than 2 levels is being predicted,

use "multiclass" for multinomial regression; and if a time-to-event outcome is being predicted, use "survival" for Cox regression.

`best_model_type`

Currently unused.

`response_col` Unquoted column name indicating which column in the data contained in ‘split_data‘ should be used as the outcome in a "two-class", "multiclass", or "linear" elastic net model. Must be a factor for "two-class" and "multiclass" models and must be a numeric for "linear" models. Ignored if ‘model_type‘ is "survival".

`time_col` Unquoted column name indicating which column in the data contained in ‘split_data‘ represents the time-to-event outcome in a "survival" elastic net model. Must be numeric. Ignored if ‘model_type‘ is "two-class", "multiclass", or "linear".

`event_col` Unquoted column name indicating which column in the data contained in ‘split_data‘ represents the time-to-event outcome in a "survival" elastic net model. Must be a binary column - all values should be either 0 or 1 (with 1 indicating the adverse event) or FALSE and TRUE (with TRUE indicating the adverse event). Ignored if ‘model_type‘ is "two-class", "multiclass", or "linear".

Value

A tibble. If arguments are specified correctly, this tibble can be used to create a recipe for preprocessing.

`tof_classify_cells` *Classify each cell (i.e. each row) in a matrix of cancer cells into its most similar healthy developmental subpopulation.*

Description

This function uses a specified distance metric to classify each cell in a data.frame or matrix (‘cancer_data‘) into one of ‘nrow(classifier_fit)‘ subpopulations based on minimum distance, as described in this paper.

Usage

```
tof_classify_cells(
  classifier_fit,
  cancer_data,
  distance_function = c("mahalanobis", "cosine", "pearson")
)
```

Arguments

`classifier_fit` A tibble produced by `tof_build_classifier`.

`cancer_data` A matrix in which each row corresponds to a cell and each column corresponds to a measured CyTOF antigen.

distance_function
: A string indicating which of three distance functions should be used to calculate the distances between each row of ‘cancer_data‘ and the healthy developmental subpopulations corresponding to each row of ‘classifier_fit‘.

Value

A data.frame in which each column represents the distance between a cell in the input data and each healthy subpopulation cells are being classified into.

---

tof_clean_metric_names

Rename glmnet's default model evaluation metrics to make them more interpretable

---

Description

Rename glmnet's default model evaluation metrics to make them more interpretable

Usage

```
tof_clean_metric_names(metric_tibble, model_type)
```

Arguments

metric_tibble	A tibble in which each column represents a glmnet model evaluation metric with its default name.
model_type	A string indicating which type of glmnet model was trained.

Value

A tibble in which each column represents a glmnet model evaluation metric with its "cleaned" name.

---

tof_cluster

Cluster high-dimensional cytometry data.

Description

This function is a wrapper around tidytof's tof_cluster_* function family. It performs clustering on high-dimensional cytometry data using a user-specified method (of 5 choices) and each method's corresponding input parameters.

Usage

```
tof_cluster(
  tof_tibble,
  cluster_cols = where(tof_is_numeric),
  group_cols = NULL,
  ...,
  augment = TRUE,
  method
)
```

Arguments

tof_tibble	A 'tof_tbl' or 'tibble'.
cluster_cols	Unquoted column names indicating which columns in 'tof_tibble' to use in computing the clusters. Defaults to all numeric columns in 'tof_tibble'. Supports tidyselect helpers.
group_cols	Optional. Unquoted column names indicating which columns should be used to group cells before clustering. Clustering is then performed on each group independently. Supports tidyselect helpers.
...	Additional arguments to pass to the 'tof_cluster_*' function family member corresponding to the chosen method.
augment	A boolean value indicating if the output should column-bind the cluster ids of each cell as a new column in 'tof_tibble' (TRUE, the default) or if a single-column tibble including only the cluster ids should be returned (FALSE).
method	A string indicating which clustering methods should be used. Valid values include "flowsom", "phenograph", "kmeans", "ddpr", and "xshift".

Value

A 'tof_tbl' or 'tibble' If augment = FALSE, it will have a single column encoding the cluster ids for each cell in 'tof_tibble'. If augment = TRUE, it will have ncol(tof_tibble) + 1 columns: each of the (unaltered) columns in 'tof_tibble' plus an additional column encoding the cluster ids.

See Also

Other clustering functions: tof_cluster_ddpr(), tof_cluster_flowsom(), tof_cluster_kmeans(), tof_cluster_phenograph()

Examples

```
sim_data <-
    dplyr::tibble(
        cd45 = rnorm(n = 500),
        cd38 = rnorm(n = 500),
        cd34 = rnorm(n = 500),
        cd19 = rnorm(n = 500)
    )
```

```
tof_cluster(tof_tibble = sim_data, method = "kmeans")
tof_cluster(tof_tibble = sim_data, method = "phenograph")
```

`tof_cluster_ddpr` *Perform developmental clustering on high-dimensional cytometry data.*

Description

This function performs distance-based clustering on high-dimensional cytometry data by sorting cancer cells (passed into the function as ‘tof_tibble‘) into their most phenotypically similar healthy cell subpopulation (passed into the function using ‘healthy_tibble‘). For details about the algorithm used to perform the clustering, see this paper.

Usage

```
tof_cluster_ddpr(
  tof_tibble,
  healthy_tibble,
  healthy_label_col,
  cluster_cols = where(tof_is_numeric),
  distance_function = c("mahalanobis", "cosine", "pearson"),
  num_cores = 1L,
  parallel_cols,
  return_distances = FALSE,
  verbose = FALSE
)
```

Arguments

`tof_tibble`	A ‘tibble‘ or ‘tof_tbl‘ containing cells to be classified into their nearest healthy subpopulation (generally cancer cells).
`healthy_tibble`	A ‘tibble‘ or ‘tof_tibble‘ containing cells from only healthy control samples (i.e. not disease samples).
`healthy_label_col`	An unquoted column name indicating which column in ‘healthy_tibble‘ contains the subpopulation label (or cluster id) for each cell in ‘healthy_tibble‘.
`cluster_cols`	Unquoted column names indicating which columns in ‘tof_tibble‘ to use in computing the DDPR clusters. Defaults to all numeric columns in ‘tof_tibble‘. Supports tidyselect helpers.
`distance_function`	A string indicating which distance function should be used to perform the classification. Options are "mahalanobis" (the default), "cosine", and "pearson".
`num_cores`	An integer indicating the number of CPU cores used to parallelize the classification. Defaults to 1 (a single core).

`parallel_cols` Optional. Unquoted column names indicating which columns in ‘tof_tibble‘ to use for breaking up the data in order to parallelize the classification using ‘foreach‘ on a ‘doParallel‘ backend. Supports tidyselect helpers.

`return_distances` A boolean value indicating whether or not the returned result should include only one column, the cluster ids corresponding to each row of ‘tof_tibble‘ (return_distances = FALSE, the default), or if the returned result should include additional columns representing the distance between each row of ‘tof_tibble‘ and each of the healthy subpopulation centroids (return_distances = TRUE).

`verbose` A boolean value indicating whether progress updates should be printed during developmental classification. Default is FALSE.

Value

If ‘return_distances = FALSE‘, a tibble with one column named ‘.{distance_function}_cluster‘, a character vector of length ‘nrow(tof_tibble)‘ indicating the id of the developmental cluster to which each cell (i.e. each row) in ‘tof_tibble‘ was assigned.

If ‘return_distances = TRUE‘, a tibble with ‘nrow(tof_tibble)‘ rows and ‘nrow(classifier_fit) + 1‘ columns. Each row represents a cell from ‘tof_tibble‘, and ‘nrow(classifier_fit)‘ of the columns represent the distance between the cell and each of the healthy subpopulations’ cluster centroids. The final column represents the cluster id of the healthy subpopulation with the minimum distance to the cell represented by that row.

If ‘return_distances = FALSE‘, a tibble with one column named ‘.{distance_function}_cluster‘. This column will contain an integer vector of length ‘nrow(tof_tibble)‘ indicating the id of the developmental cluster to which each cell (i.e. each row) in ‘tof_tibble‘ was assigned.

See Also

Other clustering functions: `tof_cluster()`, `tof_cluster_flowsom()`, `tof_cluster_kmeans()`, `tof_cluster_phenograph()`

Examples

```
sim_data <-
    dplyr::tibble(
        cd45 = rnorm(n = 1000),
        cd38 = rnorm(n = 1000),
        cd34 = rnorm(n = 1000),
        cd19 = rnorm(n = 1000)
    )

healthy_data <-
    dplyr::tibble(
        cd45 = rnorm(n = 200),
        cd38 = rnorm(n = 200),
        cd34 = rnorm(n = 200),
        cd19 = rnorm(n = 200),
        cluster_id = c(rep("a", times = 100), rep("b", times = 100))
    )
```

```
tof_cluster_ddpr(
    tof_tibble = sim_data,
    healthy_tibble = healthy_data,
    healthy_label_col = cluster_id
)
```

`tof_cluster_flowsom` *Perform FlowSOM clustering on high-dimensional cytometry data*

Description

This function performs FlowSOM clustering on high-dimensional cytometry data using a user-specified selection of input variables/high-dimensional cytometry measurements. It is mostly a convenient wrapper around `SOM` and `MetaClustering`.

Usage

```
tof_cluster_flowsom(
  tof_tibble = NULL,
  cluster_cols = where(tof_is_numeric),
  som_xdim = 10,
  som_ydim = 10,
  som_distance_function = c("euclidean", "manhattan", "chebyshev", "cosine"),
  perform_metaclustering = TRUE,
  num_metaclusters = 20,
  ...
)
```

Arguments

`tof_tibble`	A ‘tof_tbl‘ or ‘tibble‘.
`cluster_cols`	Unquoted column names indicating which columns in ‘tof_tibble‘ to use in computing the flowSOM clusters. Defaults to all numeric columns in ‘tof_tibble‘. Supports tidyselect helpers.
`som_xdim`	The width of the grid used by the self-organizing map. The total number of clusters returned by FlowSOM will be som_xdim * som_ydim, so adjust this value to affect the final number of clusters. Defaults to 10.
`som_ydim`	The height of the grid used by the self-organizing map. The total number of clusters returned by FlowSOM will be som_xdim * som_ydim, so adjust this value to affect the final number of clusters. Defaults to 10.
`som_distance_function`	The distance function used during self-organizing map calculations. Options are "euclidean" (the default), "manhattan", "chebyshev", and "cosine".

perform_metaclustering
: A boolean value indicating if metaclustering should be performed on the initial clustering result returned by FlowSOM. Defaults to TRUE.

num_metaclusters
: An integer indicating the maximum number of metaclusters that should be returned after metaclustering. Defaults to 20.

...
: Optional additional parameters that can be passed to the BuildSOM function.

Details

For additional details about the FlowSOM algorithm, see this paper.

Value

A tibble with one column named '.flowsom_cluster' or '.flowsom_metacluster' depending on the value of 'perform_metaclustering'. The column will contain an integer vector of length 'nrow(tof_tibble)' indicating the id of the flowSOM cluster to which each cell (i.e. each row) in 'tof_tibble' was assigned.

See Also

Other clustering functions: tof_cluster(), tof_cluster_ddpr(), tof_cluster_kmeans(), tof_cluster_phenograph()

Examples

```
sim_data <-
    dplyr::tibble(
        cd45 = rnorm(n = 200),
        cd38 = rnorm(n = 200),
        cd34 = rnorm(n = 200),
        cd19 = rnorm(n = 200)
    )

tof_cluster_flowsom(tof_tibble = sim_data, cluster_cols = c(cd45, cd19))
```

---

tof_cluster_grouped *Cluster (grouped) high-dimensional cytometry data.*

Description

This function is a wrapper around tidytof's tof_cluster_* function family and provides a low-level API for clustering grouped data frames. It is a subroutine of tof_cluster and shouldn't be called directly by users.

Usage

```
tof_cluster_grouped(tof_tibble, group_cols, ..., augment = TRUE, method)
```

Arguments

tof_tibble	A ‘tof_tbl‘ or ‘tibble‘.
group_cols	An unquoted column name indicating which columns should be used to group cells before clustering. Clustering is then performed on each group independently.
...	Additional arguments to pass to the ‘tof_cluster_*‘ function family member corresponding to the chosen method.
augment	A boolean value indicating if the output should column-bind the cluster ids of each cell as a new column in ‘tof_tibble‘ (TRUE, the default) or if a single-column tibble including only the cluster ids should be returned (FALSE).
method	A string indicating which clustering methods should be used. Valid values include "flowsom", "phenograph", "kmeans", "ddpr", and "xshift".

Value

A ‘tof_tbl‘ or ‘tibble‘ If augment = FALSE, it will have a single column encoding the cluster ids for each cell in ‘tof_tibble‘. If augment = TRUE, it will have ncol(tof_tibble) + 1 columns: each of the (unaltered) columns in ‘tof_tibble‘ plus an additional column encoding the cluster ids.

---

tof_cluster_kmeans *Perform k-means clustering on high-dimensional cytometry data.*

Description

This function performs k-means clustering on high-dimensional cytometry data using a user-specified selection of input variables/high-dimensional cytometry measurements. It is mostly a convenient wrapper around kmeans.

Usage

```
tof_cluster_kmeans(
  tof_tibble,
  cluster_cols = where(tof_is_numeric),
  num_clusters = 20,
  ...
)
```

Arguments

tof_tibble	A ‘tof_tibble‘.
cluster_cols	Unquoted column names indicating which columns in ‘tof_tibble‘ to use in computing the k-means clusters. Defaults to all numeric columns in ‘tof_tibble‘. Supports tidyselect helpers.
num_clusters	An integer indicating the maximum number of clusters that should be returned. Defaults to 20.
...	Optional additional arguments that can be passed to kmeans.

Value

A tibble with one column named '.kmeans_cluster'. This column will contain an integer vector of length 'nrow(tof_tibble)' indicating the id of the k-means cluster to which each cell (i.e. each row) in 'tof_tibble' was assigned.

See Also

Other clustering functions: `tof_cluster()`, `tof_cluster_ddpr()`, `tof_cluster_flowsom()`, `tof_cluster_phenograph(`

Examples

```
sim_data <-
    dplyr::tibble(
        cd45 = rnorm(n = 1000),
        cd38 = rnorm(n = 1000),
        cd34 = rnorm(n = 1000),
        cd19 = rnorm(n = 1000)
    )
tof_cluster_kmeans(tof_tibble = sim_data)
tof_cluster_kmeans(tof_tibble = sim_data, cluster_cols = c(cd45, cd19))
```

---

tof_cluster_phenograph

Perform PhenoGraph clustering on high-dimensional cytometry data.

Description

This function performs PhenoGraph clustering on high-dimensional cytometry data using a user-specified selection of input variables/high-dimensional cytometry measurements.

Usage

```
tof_cluster_phenograph(
  tof_tibble,
  cluster_cols = where(tof_is_numeric),
  num_neighbors = 30,
  distance_function = c("euclidean", "cosine"),
  ...
)
```

Arguments

`tof_tibble`	A 'tof_tbl' or 'tibble'.
`cluster_cols`	Unquoted column names indicating which columns in 'tof_tibble' to use in computing the PhenoGraph clusters. Defaults to all numeric columns in 'tof_tibble'. Supports tidyselect helpers.

num_neighbors An integer indicating the number of neighbors to use when constructing PhenoGraph's k-nearest-neighbor graph. Smaller values emphasize local graph structure; larger values emphasize global graph structure (and will add time to the computation). Defaults to 30.

distance_function A string indicating which distance function to use for the nearest-neighbor calculation. Options include "euclidean" (the default) and "cosine" distances.

... Optional additional parameters that can be passed to tof_find_knn.

Details

For additional details about the Phenograph algorithm, see this paper.

Value

A tibble with one column named '.phenograph_cluster'. This column will contain an integer vector of length 'nrow(tof_tibble)' indicating the id of the PhenoGraph cluster to which each cell (i.e. each row) in 'tof_tibble' was assigned.

See Also

Other clustering functions: tof_cluster(), tof_cluster_ddpr(), tof_cluster_flowsom(), tof_cluster_kmeans()

Examples

```
sim_data <-
    dplyr::tibble(
        cd45 = rnorm(n = 1000),
        cd38 = rnorm(n = 1000),
        cd34 = rnorm(n = 1000),
        cd19 = rnorm(n = 1000)
    )
tof_cluster_phenograph(tof_tibble = sim_data)
tof_cluster_phenograph(tof_tibble = sim_data, cluster_cols = c(cd45, cd19))
```

---

tof_cluster_tibble *Cluster (ungrouped) high-dimensional cytometry data.*

Description

This function is a wrapper around tidytof's tof_cluster_* function family and provides a low-level API for clustering ungrouped data frames. It is a subroutine of tof_cluster and shouldn't be called directly by users.

Usage

```
tof_cluster_tibble(tof_tibble, ..., augment = TRUE, method)
```

Arguments

tof_tibble	A ‘tof_tbl‘ or ‘tibble‘.
...	Additional arguments to pass to the ‘tof_cluster_*‘ function family member corresponding to the chosen method.
augment	A boolean value indicating if the output should column-bind the cluster ids of each cell as a new column in ‘tof_tibble‘ (TRUE, the default) or if a single-column tibble including only the cluster ids should be returned (FALSE).
method	A string indicating which clustering methods should be used. Valid values include "flowsom", "phenograph", "kmeans", "ddpr", and "xshift".

Value

A ‘tof_tbl‘ or ‘tibble‘ If augment = FALSE, it will have a single column encoding the cluster ids for each cell in ‘tof_tibble‘. If augment = TRUE, it will have ncol(tof_tibble) + 1 columns: each of the (unaltered) columns in ‘tof_tibble‘ plus an additional column encoding the cluster ids.

---

tof_compute_km_curve *Compute a Kaplan-Meier curve from sample-level survival data*

Description

Compute a Kaplan-Meier curve from sample-level survival data

Usage

```
tof_compute_km_curve(survival_curves)
```

Arguments

survival_curves	A tibble from which the Kaplan-Meier curve will be computed. Each row must represent an observation and must have two columns named "time_to_event" and "event".

Value

A tibble with 3 columns: time_to_event, survival_probability, and is_censored (whether or not an event was censored at that timepoint).

tof_cosine_dist — *A function for finding the cosine distance between each of the rows of a numeric matrix and a numeric vector.*

Description

A function for finding the cosine distance between each of the rows of a numeric matrix and a numeric vector.

Usage

```
tof_cosine_dist(matrix, vector)
```

Arguments

matrix	A numeric matrix.
vector	A numeric vector.

Value

A numeric vector of distances of length ‘nrow(matrix)‘ in which the ith entry represents the cosine distance between the ith row of ‘matrix‘ and ‘vector‘.

Examples

```
NULL
```

tof_create_grid — *Create an elastic net hyperparameter search grid of a specified size*

Description

This function creates a regular hyperparameter search grid (in the form of a tibble) specifying the search space for the two hyperparameters of a generalized linear model using the glmnet package: the regularization penalty term and the lasso/ridge regression mixture term.

Usage

```
tof_create_grid(
  penalty_values,
  mixture_values,
  num_penalty_values = 5,
  num_mixture_values = 5
)
```

Arguments

penalty_values	A numeric vector of the unique elastic net penalty values ("lambda") to include in the hyperparameter grid. If unspecified, a regular grid with ‘num_penalty_values‘ between 10^(-10) and 10^(0) will be used.
mixture_values	A numeric vector of all elastic net mixture values ("alpha") to include in the hyperparameter grid. If unspecified, a regular grid with ‘num_mixture_values‘ between 0 and 1 will be used.
num_penalty_values	Optional. If ‘penalty_values‘ is not supplied, ‘num_penalty_values‘ (an integer) can be given to specify how many equally-spaced penalty values between 10^(-10) and 1 should be included in the hyperparameter grid. If this method is used, the regular grid will always be returned. Defaults to 5.
num_mixture_values	Optional. If ‘mixture_values‘ is not supplied, ‘num_mixture_values‘ (an integer) can be given to specify how many equally-spaced penalty values between 0 (ridge regression) and 1 (lasso) should be included in the hyperparameter grid. If this method is used, the regular grid will always be returned. Defaults to 5.

Value

A tibble with two numeric columns: ‘penalty‘ and ‘mixture‘.

See Also

Other modeling functions: `tof_assess_model()`, `tof_predict()`, `tof_split_data()`, `tof_train_model()`

Examples

```
tof_create_grid()

tof_create_grid(num_penalty_values = 10, num_mixture_values = 5)

tof_create_grid(penalty_values = c(0.01, 0.1, 0.5))
```

---

tof_create_recipe *Create a recipe for preprocessing sample-level cytometry data for an elastic net model*

Description

Create a recipe for preprocessing sample-level cytometry data for an elastic net model

Usage

```
tof_create_recipe(
  feature_tibble,
  predictor_cols,
  outcome_cols,
  standardize_predictors = TRUE,
  remove_zv_predictors = FALSE,
  impute_missing_predictors = FALSE
)
```

Arguments

feature_tibble	A tibble in which each row represents a sample- or patient- level observation, such as those produced by tof_extract_features.
predictor_cols	Unquoted column names indicating which columns in the data contained in 'feature_tibble' should be used as predictors in the elastic net model. Supports tidyselect helpers.
outcome_cols	Unquoted column names indicating which columns in 'feature_tibble' should be used as outcome variables in the elastic net model. Supports tidyselect helpers.
standardize_predictors	A logical value indicating if numeric predictor columns should be standardized (centered and scaled) before model fitting. Defaults to TRUE.
remove_zv_predictors	A logical value indicating if predictor columns with near-zero variance should be removed before model fitting using step_nzv. Defaults to FALSE.
impute_missing_predictors	A logical value indicating if predictor columns should have missing values imputed using k-nearest neighbors before model fitting (see step_impute_knn). Imputation is performed using an observation's 5 nearest-neighbors. Defaults to FALSE.

Value

A recipe object.

---

tof_downsample *Downsample high-dimensional cytometry data.*

Description

This function downsamples the number of cells in a 'tof_tbl' using the one of three methods (randomly sampling a constant number of cells, randomly sampling a proportion of cells, or performing density-dependent downsampling per the algorithm in Qiu et al., (2011)).

Usage

```
tof_downsample(
  tof_tibble,
  group_cols = NULL,
  ...,
  method = c("constant", "prop", "density")
)
```

Arguments

tof_tibble	A ‘tof_tbl‘ or a ‘tibble‘.
group_cols	Unquoted names of the columns in ‘tof_tibble‘ that should be used to define groups within which the downsampling will be performed. Supports tidyselect helpers. Defaults to ‘NULL‘ (no grouping).
...	Additional arguments to pass to the ‘tof_downsample_*‘ function family member corresponding to the chosen method.
method	A string indicating which downsampling method to use: "constant" (the default), "prop", or "density".

Value

A downsampled ‘tof_tbl‘ with the same number of columns as the input ‘tof_tibble‘, but fewer rows. The number of rows in the result will depend on the chosen downsampling method.

See Also

Other downsampling functions: tof_downsample_constant(), tof_downsample_density(), tof_downsample_prop()

Examples

```
sim_data <-
    dplyr::tibble(
        cd45 = rnorm(n = 1000),
        cd38 = rnorm(n = 1000),
        cd34 = rnorm(n = 1000),
        cd19 = rnorm(n = 1000),
        cluster_id = sample(letters, size = 1000, replace = TRUE)
    )

# sample 200 cells from the input data
tof_downsample(
    tof_tibble = sim_data,
    num_cells = 200L,
    method = "constant"
)

# sample 10% of all cells from the input data
tof_downsample(
    tof_tibble = sim_data,
    prop_cells = 0.1,
```

```
    method = "prop"
)

# sample ~10% of cells from the input data using density dependence
tof_downsample(
    tof_tibble = sim_data,
    target_prop_cells = 0.1,
    method = "density"
)
```

`tof_downsample_constant` *Downsample high-dimensional cytometry data by randomly selecting a constant number of cells per group.*

Description

This function downsamples the number of cells in a ‘tof_tbl‘ by randomly selecting ‘num_cells‘ cells from each unique combination of values in ‘group_cols‘.

Usage

```
tof_downsample_constant(tof_tibble, group_cols = NULL, num_cells)
```

Arguments

`tof_tibble`	A ‘tof_tbl‘ or a ‘tibble‘.
`group_cols`	Unquoted names of the columns in ‘tof_tibble‘ that should be used to define groups from which ‘num_cells‘ will be downsampled. Supports tidyselect helpers. Defaults to ‘NULL‘ (no grouping).
`num_cells`	An integer number of cells that should be sampled from each group defined by ‘group_cols‘.

Value

A ‘tof_tbl‘ with the same number of columns as the input ‘tof_tibble‘, but fewer rows. Specifically, the number of rows will be ‘num_cells‘ multiplied by the number of unique combinations of the values in ‘group_cols‘. If any group has fewer than ‘num_cells‘ number of cells, all cells from that group will be kept.

See Also

Other downsampling functions: `tof_downsample()`, `tof_downsample_density()`, `tof_downsample_prop()`

Examples

```
sim_data <-
    dplyr::tibble(
        cd45 = rnorm(n = 1000),
        cd38 = rnorm(n = 1000),
        cd34 = rnorm(n = 1000),
        cd19 = rnorm(n = 1000),
        cluster_id = sample(letters, size = 1000, replace = TRUE)
    )

# sample 500 cells from the input data
tof_downsample_constant(
    tof_tibble = sim_data,
    num_cells = 500L
)

# sample 20 cells per cluster from the input data
tof_downsample_constant(
    tof_tibble = sim_data,
    group_cols = cluster_id,
    num_cells = 20L
)
```

`tof_downsample_density` *Downsample high-dimensional cytometry data by randomly selecting a proportion of the cells in each group.*

Description

This function downsamples the number of cells in a ‘tof_tbl‘ using the density-dependent downsampling algorithm described in Qiu et al., (2011).

Usage

```
tof_downsample_density(
  tof_tibble,
  group_cols = NULL,
  density_cols = where(tof_is_numeric),
  target_num_cells,
  target_prop_cells,
  target_percentile = 0.03,
  outlier_percentile = 0.01,
  distance_function = c("euclidean", "cosine", "l2", "ip"),
  density_estimation_method = c("mean_distance", "sum_distance", "spade"),
  ...
)
```

Arguments

tof_tibble	A 'tof_tbl' or a 'tibble'.
group_cols	Unquoted names of the columns in 'tof_tibble' that should be used to define groups within which the downsampling will be performed. Supports tidyselect helpers. Defaults to 'NULL' (no grouping).
density_cols	Unquoted names of the columns in 'tof_tibble' to use in the density estimation for each cell. Defaults to all numeric columns in 'tof_tibble'.
target_num_cells	An approximate constant number of cells (between 0 and 1) that should be sampled from each group defined by 'group_cols'. Slightly more or fewer cells may be returned due to how the density calculation is performed.
target_prop_cells	An approximate proportion of cells (between 0 and 1) that should be sampled from each group defined by 'group_cols'. Slightly more or fewer cells may be returned due to how the density calculation is performed. Ignored if 'target_num_cells' is specified.
target_percentile	The local density percentile (i.e. a value between 0 and 1) to which the downsampling procedure should adjust all cells. In short, the algorithm will continue to remove cells from the input 'tof_tibble' until the local densities of all remaining cells is equal to 'target_percentile'. Lower values will result in more cells being removed. See Qiu et al., (2011) for details. Defaults to 0.1 (the 10th percentile of local densities). Ignored if either 'target_num_cells' or 'target_prop_cells' are specified.
outlier_percentile	The local density percentile (i.e. a value between 0 and 1) below which cells should be considered outliers (and discarded). Cells with a local density below 'outlier_percentile' will never be selected during the downsampling procedure. Defaults to 0.01 (cells below the 1st local density percentile will be removed).
distance_function	A string indicating which distance function to use for the cell-to-cell distance calculations. Options include "euclidean" (the default) and "cosine" distances.
density_estimation_method	A string indicating which algorithm should be used to calculate the local density estimate for each cell. Options include k-nearest neighbor density estimation using the mean distance to a cell's k-nearest neighbors ("mean_distance"; the default), k-nearest neighbor density estimation using the summed distance to a cell's k nearest neighbors ("sum_distance") and counting the number of neighboring cells within a spherical radius around each cell as described in Qiu et al., 2011 ("spade"). While "spade" often produces the best results, it is slower than knn-density estimation methods.
...	Optional additional arguments to pass to tof_knn_density or tof_spade_density.

Value

A 'tof_tbl' with the same number of columns as the input 'tof_tibble', but fewer rows. The number of rows will depend on the chosen value of 'target_percentile', with fewer cells selected with lower

values of ‘target_percentile‘.

See Also

Other downsampling functions: `tof_downsample()`, `tof_downsample_constant()`, `tof_downsample_prop()`

Examples

```
sim_data <-
    dplyr::tibble(
        cd45 = rnorm(n = 1000),
        cd38 = rnorm(n = 1000),
        cd34 = rnorm(n = 1000),
        cd19 = rnorm(n = 1000)
    )

tof_downsample_density(
    tof_tibble = sim_data,
    density_cols = c(cd45, cd34, cd38),
    target_prop_cells = 0.5,
    density_estimation_method = "spade"
)

tof_downsample_density(
    tof_tibble = sim_data,
    density_cols = c(cd45, cd34, cd38),
    target_num_cells = 200L,
    density_estimation_method = "spade"
)

tof_downsample_density(
    tof_tibble = sim_data,
    density_cols = c(cd45, cd34, cd38),
    target_num_cells = 200L,
    density_estimation_method = "mean_distance"
)
```

`tof_downsample_prop` *Downsample high-dimensional cytometry data by randomly selecting a proportion of the cells in each group.*

Description

This function downsamples the number of cells in a ‘tof_tbl‘ by randomly selecting a ‘prop_cells‘ proportion of the total number of cells with each unique combination of values in ‘group_cols‘.

Usage

```
tof_downsample_prop(tof_tibble, group_cols = NULL, prop_cells)
```

Arguments

tof_tibble	A ‘tof_tbl‘ or a ‘tibble‘.
group_cols	Unquoted names of the columns in ‘tof_tibble‘ that should be used to define groups from which ‘prop_cells‘ will be downsampled. Supports tidyselect helpers. Defaults to ‘NULL‘ (no grouping).
prop_cells	A proportion of cells (between 0 and 1) that should be sampled from each group defined by ‘group_cols‘.

Value

A ‘tof_tbl‘ with the same number of columns as the input ‘tof_tibble‘, but fewer rows. Specifically, the number of rows should be ‘prop_cells‘ times the number of rows in the input ‘tof_tibble‘.

See Also

Other downsampling functions: `tof_downsample()`, `tof_downsample_constant()`, `tof_downsample_density()`

Examples

```
sim_data <-
    dplyr::tibble(
        cd45 = rnorm(n = 1000),
        cd38 = rnorm(n = 1000),
        cd34 = rnorm(n = 1000),
        cd19 = rnorm(n = 1000),
        cluster_id = sample(letters, size = 1000, replace = TRUE)
    )

# sample 10% of all cells from the input data
tof_downsample_prop(
    tof_tibble = sim_data,
    prop_cells = 0.1
)

# sample 10% of all cells from each cluster in the input data
tof_downsample_prop(
    tof_tibble = sim_data,
    group_cols = cluster_id,
    prop_cells = 0.1
)
```

---

tof_estimate_density *Estimate the local densities for all cells in a high-dimensional cytometry dataset.*

Description

This function is a wrapper around tidytof's tof_*_density() function family. It performs local density estimation on high-dimensional cytometry data using a user-specified method (of 3 choices) and each method's corresponding input parameters.

Usage

```
tof_estimate_density(
  tof_tibble,
  distance_cols = where(tof_is_numeric),
  distance_function = c("euclidean", "cosine", "l2", "ip"),
  normalize = TRUE,
  ...,
  augment = TRUE,
  method = c("mean_distance", "sum_distance", "spade")
)
```

Arguments

tof_tibble	A 'tof_tbl' or a 'tibble'.
distance_cols	Unquoted names of the columns in 'tof_tibble' to use in calculating cell-to-cell distances during the local density estimation for each cell. Defaults to all numeric columns in 'tof_tibble'.
distance_function	A string indicating which distance function to use for calculating cell-to-cell distances during local density estimation. Options include "euclidean" (the default) and "cosine".
normalize	A boolean value indicating if the vector of local density estimates should be normalized to values between 0 and 1. Defaults to TRUE.
...	Additional arguments to pass to the 'tof_*_density()' function family member corresponding to the chosen 'method'.
augment	A boolean value indicating if the output should column-bind the local density estimates of each cell as a new column in 'tof_tibble' (TRUE; the default) or if a single-column tibble including only the local density estimates should be returned (FALSE).
method	A string indicating which local density estimation method should be used. Valid values include "mean_distance", "sum_distance", and "spade".

Value

A 'tof_tbl' or 'tibble' If augment = FALSE, it will have a single column encoding the local density estimates for each cell in 'tof_tibble'. If augment = TRUE, it will have ncol(tof_tibble) + 1 columns: each of the (unaltered) columns in 'tof_tibble' plus an additional column encoding the local density estimates.

See Also

Other local density estimation functions: tof_knn_density(), tof_spade_density()

Examples

```
sim_data <-
    dplyr::tibble(
        cd45 = rnorm(n = 1000),
        cd38 = rnorm(n = 1000),
        cd34 = rnorm(n = 1000),
        cd19 = rnorm(n = 1000)
    )

# perform the density estimation
tof_estimate_density(tof_tibble = sim_data, method = "spade")

# perform the density estimation with a smaller search radius around
# each cell
tof_estimate_density(
    tof_tibble = sim_data,
    alpha_multiplier = 2,
    method = "spade"
)
```

---

tof_extract_central_tendency

Extract the central tendencies of CyTOF markers in each cluster in a 'tof_tibble'.

Description

This feature extraction function calculates a user-specified measurement of central tendency (i.e. median or mode) of the cells in each cluster in a 'tof_tibble' across a user-specified selection of CyTOF markers. These calculations can be done either overall (across all cells in the dataset) or after breaking down the cells into subgroups using 'group_cols'.

Usage

```
tof_extract_central_tendency(
  tof_tibble,
  cluster_col,
  group_cols = NULL,
  marker_cols = where(tof_is_numeric),
  stimulation_col = NULL,
  central_tendency_function = stats::median,
  format = c("wide", "long")
)
```

Arguments

tof_tibble	A ‘tof_tibble‘ or a ‘tibble‘ in which each row represents a single cell and each column represents a CyTOF measurement or a piece of metadata (i.e. cluster id, patient id, etc.) about each cell.
cluster_col	An unquoted column name indicating which column in ‘tof_tibble‘ stores the cluster ids of the cluster to which each cell belongs. Cluster labels can be produced via any method the user chooses - including manual gating, any of the functions in the ‘tof_cluster_*‘ function family, or any other method.
group_cols	Unquoted column names representing which columns in ‘tof_tibble‘ should be used to break the rows of ‘tof_tibble‘ into subgroups for the feature extraction calculation. Defaults to NULL (i.e. performing the extraction without subgroups).
marker_cols	Unquoted column names representing which columns in ‘tof_tibble‘ (i.e. which CyTOF protein measurements) should be included in the feature extraction calculation. Defaults to all numeric (integer or double) columns. Supports tidyselection.
stimulation_col	Optional. An unquoted column name that indicates which column in ‘tof_tibble‘ contains information about which stimulation condition each cell was exposed to during data acquisition. If provided, the feature extraction will be further broken down into subgroups by stimulation condition (and features from each stimulation condition will be included as their own features in wide format).
central_tendency_function	The function that will be used to calculate the measurement of central tendency for each cluster (to be used as the dependent variable in the linear model). Defaults to median.
format	A string indicating if the data should be returned in "wide" format (the default; each cluster feature is given its own column) or in "long" format (each cluster feature is provided as its own row).

Value

A tibble.

If format == "wide", the tibble will have 1 row for each combination of the grouping variables provided in ‘group_cols‘ and one column for each grouping variable, one column for each extracted feature (the central tendency of a given marker in a given cluster). The names of each column containing cluster features is obtained using the following pattern: "{marker_id}@{cluster_id}_ct".

If format == "long", the tibble will have 1 row for each combination of the grouping variables in ‘group_cols‘, each cluster id (i.e. level) in ‘cluster_col‘, and each marker in ‘marker_cols‘. It will have one column for each grouping variable, one column for the cluster ids, one column for the CyTOF channel names, and one column (‘value‘) containing the features.

See Also

Other feature extraction functions: tof_extract_emd(), tof_extract_features(), tof_extract_jsd(), tof_extract_proportion(), tof_extract_threshold()

Examples

```
sim_data <-
    dplyr::tibble(
        cd45 = rnorm(n = 1000),
        cd38 = rnorm(n = 1000),
        cd34 = rnorm(n = 1000),
        cd19 = rnorm(n = 1000),
        cluster_id = sample(letters, size = 1000, replace = TRUE),
        patient = sample(c("kirby", "mario"), size = 1000, replace = TRUE),
        stim = sample(c("basal", "stim"), size = 1000, replace = TRUE)
    )

# extract proportion of each cluster in each patient in wide format
tof_extract_central_tendency(
    tof_tibble = sim_data,
    cluster_col = cluster_id,
    group_cols = patient
)

# extract proportion of each cluster in each patient in long format
tof_extract_central_tendency(
    tof_tibble = sim_data,
    cluster_col = cluster_id,
    group_cols = patient,
    format = "long"
)
```

---

tof_extract_emd *Extract aggregated features from CyTOF data using earth-mover's distance (EMD)*

Description

This feature extraction function calculates the earth-mover's distance (EMD) between the stimulated and unstimulated ("basal") experimental conditions of samples in a CyTOF experiment. This calculation is performed across a user-specified selection of CyTOF antigens and can be performed either overall (across all cells in the dataset) or after breaking down the cells into subgroups using 'group_cols'.

Usage

```
tof_extract_emd(
  tof_tibble,
  cluster_col,
  group_cols = NULL,
  marker_cols = where(tof_is_numeric),
  emd_col,
  reference_level,
```

```
  format = c("wide", "long"),
  num_bins = 100
)
```

Arguments

tof_tibble	A ‘tof_tbl‘ or a ‘tibble‘.
cluster_col	An unquoted column name indicating which column in ‘tof_tibble‘ stores the cluster ids of the cluster to which each cell belongs. Cluster labels can be produced via any method the user chooses - including manual gating, any of the functions in the ‘tof_cluster_*‘ function family, or any other method.
group_cols	Unquoted column names representing which columns in ‘tof_tibble‘ should be used to break the rows of ‘tof_tibble‘ into subgroups for the feature extraction calculation. Defaults to NULL (i.e. performing the extraction without subgroups).
marker_cols	Unquoted column names representing which columns in ‘tof_tibble‘ (i.e. which CyTOF protein measurements) should be included in the earth-mover’s distance calculation. Defaults to all numeric (integer or double) columns. Supports tidyselect helpers.
emd_col	An unquoted column name that indicates which column in ‘tof_tibble‘ should be used to group cells into different distributions to be compared with one another during the EMD calculation. For example, if you want to compare marker expression distributions across stimulation conditions, ‘emd_col‘ should be the column in ‘tof_tibble‘ containing information about which stimulation condition each cell was exposed to during data acquisition. If provided, the feature extraction will be further broken down into subgroups by stimulation condition (and features from each stimulation condition will be included as their own features in wide format).
reference_level	A string indicating what the value in ‘emd_col‘ corresponds to the "reference" value to which all other values in ‘emd_col‘ should be compared. For example, if ‘emd_col‘ represents the stimulation condition for a cell, reference_level might take the value of "basal" or "unstimulated" if you want to compare each stimulation to the basal state.
format	A string indicating if the data should be returned in "wide" format (the default; each cluster feature is given its own column) or in "long" format (each cluster feature is provided as its own row).
num_bins	Optional. The number of bins to use in dividing one-dimensional marker distributions into discrete segments for the EMD calculation. Defaults to 100.

Value

A tibble.

If format == "wide", the tibble will have 1 row for each combination of the grouping variables provided in ‘group_cols‘ and one column for each grouping variable, one column for each extracted feature (the EMD between the distribution of a given marker in a given cluster in the basal

condition and the distribution of that marker in a given cluster in a stimulated condition). The names of each column containing cluster features is obtained using the following pattern: "{stimulation_id}_{marker_id}@{cluster_id}_emd".

If format == "long", the tibble will have 1 row for each combination of the grouping variables in ‘group_cols‘, each cluster id (i.e. level) in ‘cluster_col‘, and each marker in ‘marker_cols‘. It will have one column for each grouping variable, one column for the cluster ids, one column for the CyTOF channel names, and one column (‘value‘) containing the features.

See Also

Other feature extraction functions: tof_extract_central_tendency(), tof_extract_features(), tof_extract_jsd(), tof_extract_proportion(), tof_extract_threshold()

Examples

```
sim_data <-
    dplyr::tibble(
        cd45 = rnorm(n = 1000),
        cd38 = rnorm(n = 1000),
        cd34 = rnorm(n = 1000),
        cd19 = rnorm(n = 1000),
        cluster_id = sample(letters, size = 1000, replace = TRUE),
        patient = sample(c("kirby", "mario"), size = 1000, replace = TRUE),
        stim = sample(c("basal", "stim"), size = 1000, replace = TRUE)
    )

# extract emd of each cluster in each patient (using the "basal" stim
# condition as a reference) in wide format
tof_extract_emd(
    tof_tibble = sim_data,
    cluster_col = cluster_id,
    group_cols = patient,
    emd_col = stim,
    reference_level = "basal"
)

# extract emd of each cluster (using the "basal" stim
# condition as a reference) in long format
tof_extract_emd(
    tof_tibble = sim_data,
    cluster_col = cluster_id,
    emd_col = stim,
    reference_level = "basal",
    format = "long"
)
```

`tof_extract_features` *Extract aggregated, sample-level features from CyTOF data.*

Description

This function wraps other members of the ‘tof_extract_*‘ function family to extract sample-level features from both lineage (i.e. cell surface antigen) CyTOF channels assumed to be stable across stimulation conditions and signaling CyTOF channels assumed to change across stimulation conditions. Features are extracted for each cluster within each independent sample (as defined with the ‘group_cols‘ argument).

Usage

```
tof_extract_features(
  tof_tibble,
  cluster_col,
  group_cols = NULL,
  stimulation_col = NULL,
  lineage_cols,
  signaling_cols,
  central_tendency_function = stats::median,
  signaling_method = c("threshold", "emd", "jsd", "central tendency"),
  basal_level = NULL,
  ...
)
```

Arguments

`tof_tibble`	A ‘tof_tbl‘ or a ‘tibble‘.
`cluster_col`	An unquoted column name indicating which column in ‘tof_tibble‘ stores the cluster ids of the cluster to which each cell belongs. Cluster labels can be produced via any method the user chooses - including manual gating, any of the functions in the ‘tof_cluster_*‘ function family, or any other method.
`group_cols`	Unquoted column names representing which columns in ‘tof_tibble‘ should be used to break the rows of ‘tof_tibble‘ into subgroups for the feature extraction calculation. Defaults to NULL (i.e. performing the extraction without subgroups).
`stimulation_col`	Optional. An unquoted column name that indicates which column in ‘tof_tibble‘ contains information about which stimulation condition each cell was exposed to during data acquisition. If provided, the feature extraction will be further broken down into subgroups by stimulation condition (and features from each stimulation condition will be included as their own features in wide format).
`lineage_cols`	Unquoted column names representing which columns in ‘tof_tibble‘ (i.e. which CyTOF protein measurements) should be considered lineage markers in the feature extraction calculation. Supports tidyselect helpers.

`signaling_cols` Unquoted column names representing which columns in ‘tof_tibble‘ (i.e. which CyTOF protein measurements) should be considered signaling markers in the feature extraction calculation. Supports tidyselect helpers.

`central_tendency_function` The function that will be used to calculate the measurement of central tendency for each cluster (to be used as the dependent variable in the linear model). Defaults to `median`.

`signaling_method` A string indicating which feature extraction method to use for signaling markers (as identified by the ‘signaling_cols‘ argument). Options are "threshold" (the default), "emd", "jsd", and "central tendency".

`basal_level` A string indicating what the value in ‘stimulation_col‘ corresponds to the basal stimulation condition (i.e. "basal" or "unstimulated").

`...` Optional additional arguments to be passed to tof_extract_threshold, `tof_extract_emd`, or `tof_extract_jsd`.

Details

Lineage channels are specified using the ‘lineage_cols‘ argument, and their extracted features will be measurements of central tendency (as computed by the user-supplied ‘central_tendency_function‘).

Signaling channels are specified using the ‘signaling_cols‘ argument, and their extracted features will depend on the user’s chosen ‘signaling_method‘. If ‘signaling method‘ == "threshold" (the default), `tof_extract_threshold` will be used to calculate the proportion of cells in each cluster with signaling marker expression over ‘threshold‘ in each stimulation condition. If ‘signaling_method‘ == "emd" or ‘signaling_method‘ == "jsd", `tof_extract_emd` or `tof_extract_jsd` will be used to calculate the earth-mover’s distance (EMD) or Jensen-Shannon Distance (JSD), respectively, between the basal condition and each of the stimulated conditions in each cluster for each sample. Finally, if none of these options are chosen, `tof_extract_central_tendency` will be used to calculate measurements of central tendency.

In addition, `tof_extract_proportion` will be used to extract the proportion of cells in each cluster will be computed for each sample.

These calculations can be performed either overall (across all cells in the dataset) or after breaking down the cells into subgroups using ‘group_cols‘.

Value

A tibble.

The output tibble will have 1 row for each combination of the grouping variables provided in ‘group_cols‘ (thus, each row will represent what is considered a single "sample" based on the grouping provided). It will have one column for each grouping variable and one column for each extracted feature ("wide" format).

See Also

Other feature extraction functions: `tof_extract_central_tendency()`, `tof_extract_emd()`, `tof_extract_jsd()`, `tof_extract_proportion()`, `tof_extract_threshold()`

Examples

```
sim_data <-
    dplyr::tibble(
        cd45 = rnorm(n = 1000),
        cd38 = rnorm(n = 1000),
        cd34 = rnorm(n = 1000),
        cd19 = rnorm(n = 1000),
        cluster_id = sample(letters, size = 1000, replace = TRUE),
        patient = sample(c("kirby", "mario"), size = 1000, replace = TRUE),
        stim = sample(c("basal", "stim"), size = 1000, replace = TRUE)
    )

# extract the following features from each cluster in each
# patient/stimulation:
#    - proportion of each cluster
#    - central tendency (median) of cd45 and cd38 in each cluster
#    - the proportion of cells in each cluster with cd34 expression over
#      the default threshold (asinh(10 / 5))
tof_extract_features(
    tof_tibble = sim_data,
    cluster_col = cluster_id,
    group_cols = patient,
    lineage_cols = c(cd45, cd38),
    signaling_cols = cd34,
    stimulation_col = stim
)

# extract the following features from each cluster in each
# patient/stimulation:
#    - proportion of each cluster
#    - central tendency (mean) of cd45 and cd38 in each cluster
#    - the earth mover's distance between each cluster's cd34 histogram in
#      the "basal" and "stim" conditions
tof_extract_features(
    tof_tibble = sim_data,
    cluster_col = cluster_id,
    group_cols = patient,
    lineage_cols = c(cd45, cd38),
    signaling_cols = cd34,
    central_tendency_function = mean,
    stimulation_col = stim,
    signaling_method = "emd",
    basal_level = "basal"
)
```

`tof_extract_jsd` *Extract aggregated features from CyTOF data using the Jensen-Shannon Distance (JSD)*

Description

This feature extraction function calculates the Jensen-Shannon Distance (JSD) between the stimulated and unstimulated ("basal") experimental conditions of samples in a CyTOF experiment. This calculation is performed across a user-specified selection of CyTOF antigens and can be performed either overall (across all cells in the dataset) or after breaking down the cells into subgroups using ‘group_cols‘.

Usage

```
tof_extract_jsd(
  tof_tibble,
  cluster_col,
  group_cols = NULL,
  marker_cols = where(tof_is_numeric),
  jsd_col,
  reference_level,
  format = c("wide", "long"),
  num_bins = 100
)
```

Arguments

tof_tibble	A ‘tof_tbl‘ or a ‘tibble‘.
cluster_col	An unquoted column name indicating which column in ‘tof_tibble‘ stores the cluster ids of the cluster to which each cell belongs. Cluster labels can be produced via any method the user chooses - including manual gating, any of the functions in the ‘tof_cluster_*‘ function family, or any other method.
group_cols	Unquoted column names representing which columns in ‘tof_tibble‘ should be used to break the rows of ‘tof_tibble‘ into subgroups for the feature extraction calculation. Defaults to NULL (i.e. performing the extraction without subgroups).
marker_cols	Unquoted column names representing which columns in ‘tof_tibble‘ (i.e. which CyTOF protein measurements) should be included in the feature extraction calculation. Defaults to all numeric (integer or double) columns. Supports tidyselect helpers.
jsd_col	An unquoted column name that indicates which column in ‘tof_tibble‘ contains information about which stimulation condition each cell was exposed to during data acquisition. If provided, the feature extraction will be further broken down into subgroups by stimulation condition (and features from each stimulation condition will be included as their own features in wide format).
reference_level	A string indicating what the value in ‘jsd_col‘ corresponds to the basal stimulation condition (i.e. "basal" or "unstimulated").
format	A string indicating if the data should be returned in "wide" format (the default; each cluster feature is given its own column) or in "long" format (each cluster feature is provided as its own row).

num_bins Optional. The number of bins to use in dividing one-dimensional marker distributions into discrete segments for the JSD calculation. Defaults to 100.

Value

A tibble.

If format == "wide", the tibble will have 1 row for each combination of the grouping variables provided in ‘group_cols‘ and one column for each grouping variable, one column for each extracted feature (the JSD between the distribution of a given marker in a given cluster in the basal condition and the distribution of that marker in the same cluster in a stimulated condition). The names of each column containing cluster features is obtained using the following pattern: "{stimulation_id}_{marker_id}@{cluster_id}_jsd".

If format == "long", the tibble will have 1 row for each combination of the grouping variables in ‘group_cols‘, each cluster id (i.e. level) in ‘cluster_col‘, and each marker in ‘marker_cols‘. It will have one column for each grouping variable, one column for the cluster ids, one column for the CyTOF channel names, and one column (‘value‘) containing the features.

See Also

Other feature extraction functions: tof_extract_central_tendency(), tof_extract_emd(), tof_extract_features(), tof_extract_proportion(), tof_extract_threshold()

Examples

```
sim_data <-
    dplyr::tibble(
        cd45 = rnorm(n = 1000),
        cd38 = rnorm(n = 1000),
        cd34 = rnorm(n = 1000),
        cd19 = rnorm(n = 1000),
        cluster_id = sample(letters, size = 1000, replace = TRUE),
        patient = sample(c("kirby", "mario"), size = 1000, replace = TRUE),
        stim = sample(c("basal", "stim"), size = 1000, replace = TRUE)
    )

# extract jsd of each cluster in each patient (using the "basal" stim
# condition as a reference) in wide format
tof_extract_jsd(
    tof_tibble = sim_data,
    cluster_col = cluster_id,
    group_cols = patient,
    jsd_col = stim,
    reference_level = "basal"
)

# extract jsd of each cluster (using the "basal" stim
# condition as a reference) in long format
tof_extract_jsd(
    tof_tibble = sim_data,
    cluster_col = cluster_id,
    jsd_col = stim,
```

```
    reference_level = "basal",
    format = "long"
)
```

`tof_extract_proportion` *Extract the proportion of cells in each cluster in a 'tof_tibble'.*

Description

This feature extraction function allows you to calculate the proportion of cells in each cluster in a 'tof_tibble' - either overall or when broken down into subgroups using 'group_cols'.

Usage

```
tof_extract_proportion(
  tof_tibble,
  cluster_col,
  group_cols = NULL,
  format = c("wide", "long")
)
```

Arguments

`tof_tibble`	A 'tof_tbl' or a 'tibble'.
`cluster_col`	An unquoted column name indicating which column in 'tof_tibble' stores the cluster ids of the cluster to which each cell belongs. Cluster labels can be produced via any method the user chooses - including manual gating, any of the functions in the 'tof_cluster_*' function family, or any other method.
`group_cols`	Unquoted column names representing which columns in 'tof_tibble' should be used to break the rows of 'tof_tibble' into subgroups for the feature extraction calculation. Defaults to NULL (i.e. performing the extraction without subgroups).
`format`	A string indicating if the data should be returned in "wide" format (the default; each cluster proportion is given its own column) or in "long" format (each cluster proportion is provided as its own row).

Value

A tibble.

If format == "wide", the tibble will have 1 row for each combination of the grouping variables provided in 'group_cols' and one column for each grouping variable as well as one column for the proportion of cells in each cluster. The names of each column containing cluster proportions is obtained using the following pattern: "prop@{cluster_id}".

If format == "long", the tibble will have 1 row for each combination of the grouping variables in ‘group_cols‘ and each cluster id (i.e. level) in ‘cluster_col‘. It will have one column for each grouping variable, one column for the cluster ids, and one column (‘prop‘) containing the cluster proportions.

See Also

Other feature extraction functions: `tof_extract_central_tendency()`, `tof_extract_emd()`, `tof_extract_features()`, `tof_extract_jsd()`, `tof_extract_threshold()`

Examples

```
sim_data <-
    dplyr::tibble(
        cd45 = rnorm(n = 1000),
        cd38 = rnorm(n = 1000),
        cd34 = rnorm(n = 1000),
        cd19 = rnorm(n = 1000),
        cluster_id = sample(letters, size = 1000, replace = TRUE),
        patient = sample(c("kirby", "mario"), size = 1000, replace = TRUE),
        stim = sample(c("basal", "stim"), size = 1000, replace = TRUE)
    )

# extract proportion of each cluster in each patient in wide format
tof_extract_proportion(
    tof_tibble = sim_data,
    cluster_col = cluster_id,
    group_cols = patient
)

# extract proportion of each cluster in each patient in long format
tof_extract_proportion(
    tof_tibble = sim_data,
    cluster_col = cluster_id,
    group_cols = patient,
    format = "long"
)
```

---

tof_extract_threshold *Extract aggregated features from CyTOF data using a binary threshold*

Description

This feature extraction function calculates the proportion of cells in a given cluster that have a CyTOF antigen expression over a user-specified threshold across a user-specified selection of CyTOF markers. These calculations can be done either overall (across all cells in the dataset) or after breaking down the cells into subgroups using ‘group_cols‘.

Usage

```
tof_extract_threshold(
  tof_tibble,
  cluster_col,
  group_cols = NULL,
  marker_cols = where(tof_is_numeric),
  stimulation_col = NULL,
  threshold = asinh(10/5),
  format = c("wide", "long")
)
```

Arguments

tof_tibble	A ‘tof_tbl‘ or a ‘tibble‘.
cluster_col	An unquoted column name indicating which column in ‘tof_tibble‘ stores the cluster ids of the cluster to which each cell belongs. Cluster labels can be produced via any method the user chooses - including manual gating, any of the functions in the ‘tof_cluster_*‘ function family, or any other method.
group_cols	Unquoted column names representing which columns in ‘tof_tibble‘ should be used to break the rows of ‘tof_tibble‘ into subgroups for the feature extraction calculation. Defaults to NULL (i.e. performing the extraction without subgroups).
marker_cols	Unquoted column names representing which columns in ‘tof_tibble‘ (i.e. which CyTOF protein measurements) should be included in the feature extraction calculation. Defaults to all numeric (integer or double) columns. Supports tidyselect helpers.
stimulation_col	Optional. An unquoted column name that indicates which column in ‘tof_tibble‘ contains information about which stimulation condition each cell was exposed to during data acquisition. If provided, the feature extraction will be further broken down into subgroups by stimulation condition (and features from each stimulation condition will be included as their own features in wide format).
threshold	A double or integer of length 1 indicating what threshold should be used.
format	A string indicating if the data should be returned in "wide" format (the default; each cluster feature is given its own column) or in "long" format (each cluster feature is provided as its own row).

Value

A tibble.

If format == "wide", the tibble will have 1 row for each combination of the grouping variables provided in ‘group_cols‘ and one column for each grouping variable, one column for each extracted feature (the proportion of cells in a given cluster over with marker expression values over ‘threshold‘). The names of each column containing cluster features is obtained using the following pattern: "{marker_id}@{cluster_id}_threshold".

If format == "long", the tibble will have 1 row for each combination of the grouping variables in ‘group_cols‘, each cluster id (i.e. level) in ‘cluster_col‘, and each marker in ‘marker_cols‘. It will have one column for each grouping variable, one column for the cluster ids, one column for the CyTOF channel names, and one column (‘value‘) containing the features.

See Also

Other feature extraction functions: `tof_extract_central_tendency()`, `tof_extract_emd()`, `tof_extract_features()`, `tof_extract_jsd()`, `tof_extract_proportion()`

Examples

```
sim_data <-
    dplyr::tibble(
        cd45 = rnorm(n = 1000),
        cd38 = rnorm(n = 1000),
        cd34 = rnorm(n = 1000),
        cd19 = rnorm(n = 1000),
        cluster_id = sample(letters, size = 1000, replace = TRUE),
        patient = sample(c("kirby", "mario"), size = 1000, replace = TRUE),
        stim = sample(c("basal", "stim"), size = 1000, replace = TRUE)
    )

# extract proportion of each cluster in each patient in wide format
tof_extract_threshold(
    tof_tibble = sim_data,
    cluster_col = cluster_id,
    group_cols = patient
)

# extract proportion of each cluster in each patient in long format
tof_extract_threshold(
    tof_tibble = sim_data,
    cluster_col = cluster_id,
    group_cols = patient,
    format = "long"
)
```

---

tof_find_best *Find the optimal hyperparameters for an elastic net model from candidate performance metrics*

Description

Find the optimal hyperparameters for an elastic net model from candidate performance metrics

Usage

```
tof_find_best(performance_metrics, model_type, optimization_metric)
```

Arguments

performance_metrics	A tibble of performance metrics for an elastic net model (in wide format)
model_type	A string indicating which type of glmnet model was trained.
optimization_metric	A string indicating which performance metric should be used to select the optimal model.

Value

A tibble with 3 columns: "mixture", "penalty", and a column containing the chosen optimization metric. If the returned tibble has more than 1 column, it means that more than 1 mixture/penalty combination yielded the optimal result (i.e. the tuning procedure resulted in a tie).

---

tof_find_cv_predictions

Calculate and store the predicted outcomes for each validation set observation during model tuning

---

Description

Calculate and store the predicted outcomes for each validation set observation during model tuning

Usage

```
tof_find_cv_predictions(
  split_data,
  prepped_recipe,
  lambda,
  alpha,
  model_type,
  outcome_colnames
)
```

Arguments

split_data	An ‘rsplit‘ object from the rsample package. Alternatively, an unsplit tbl_df can be provided, though this is not recommended.
prepped_recipe	A trained recipe
lambda	A single numeric value indicating which penalty (lambda) value should be used to make the predictions
alpha	A single numeric value indicating which mixture (alpha) value should be used to make the predictions

model_type A string indicating which kind of elastic net model to build. If a continuous response is being predicted, use "linear" for linear regression; if a categorical response with only 2 classes is being predicted, use "two-class" for logistic regression; if a categorical response with more than 2 levels is being predicted, use "multiclass" for multinomial regression; and if a time-to-event outcome is being predicted, use "survival" for Cox regression.

outcome_colnames Quoted column names indicating which columns in the data being fit represent the outcome variables (with all others assumed to be predictors).

Value

A tibble containing the predicted and true values for the outcome for each of the validation observations in 'split_data'.

---

tof_find_emd *Find the earth-mover's distance between two numeric vectors*

Description

Find the earth-mover's distance between two numeric vectors

Usage

```
tof_find_emd(vec_1, vec_2, num_bins = 100)
```

Arguments

vec_1 A numeric vector.

vec_2 A numeric vector.

num_bins An integer number of bins to use when performing kernel density estimation on the two vectors. Defaults to 100.

Value

A double (of length 1) representing the EMD between the two vectors.

tof_find_jsd — *Find the Jensen-Shannon Divergence (JSD) between two numeric vectors*

Description

Find the Jensen-Shannon Divergence (JSD) between two numeric vectors

Usage

```
tof_find_jsd(vec_1, vec_2, num_bins = 100)
```

Arguments

vec_1	A numeric vector.
vec_2	A numeric vector.
num_bins	An integer number of bins to use when binning across the two vectors' combined range. Defaults to 100.

Value

A double (of length 1) representing the JSD between the two vectors.

tof_find_knn — *Find the k-nearest neighbors of each cell in a high-dimensional cytometry dataset.*

Description

Find the k-nearest neighbors of each cell in a high-dimensional cytometry dataset.

Usage

```
tof_find_knn(
  .data,
  k = min(10, nrow(.data)),
  distance_function = c("euclidean", "cosine", "l2", "ip"),
  .query,
  ...
)
```

Arguments

.data	A ‘tof_tibble‘ or ‘tibble‘ in which each row represents a cell and each column represents a high-dimensional cytometry measurement.
k	An integer indicating the number of nearest neighbors to return for each cell.
distance_function	A string indicating which distance function to use for the nearest-neighbor calculation. Options include "euclidean" (the default) and "cosine" distances.
.query	A set of cells to be queried against .data (i.e. a set of cells for which to find nearest neighbors within .data). Defaults to .data itself, i.e. finding nearest neighbors for all cells in .data.
...	Optional additional arguments to pass to hnsw_knn

Value

A list with two elements: "neighbor_ids" and "neighbor_distances," both of which are n by k matrices (in which n is the number of cells in the input ‘.data‘. The [i,j]-th entry of "neighbor_ids" represents the row index for the j-th nearest neighbor of the cell in the i-th row of ‘.data‘. The [i,j]-th entry of "neighbor_distances" represents the distance between those two cells according to ‘distance_function‘.

Examples

```
sim_data <-
    dplyr::tibble(
        cd45 = rnorm(n = 1000),
        cd38 = rnorm(n = 1000),
        cd34 = rnorm(n = 1000),
        cd19 = rnorm(n = 1000)
    )

# Find the 10 nearest neighbors of each cell in the dataset
tof_find_knn(
    .data = sim_data,
    k = 10,
    distance_function = "euclidean"
)

# Find the 10 approximate nearest neighbors
tof_find_knn(
    .data = sim_data,
    k = 10,
    distance_function = "euclidean",
)
```

`tof_find_log_rank_threshold` *Compute the log-rank test p-value for the difference between the two survival curves obtained by splitting a dataset into a "low" and "high" risk group using all possible relative-risk thresholds.*

Description

Compute the log-rank test p-value for the difference between the two survival curves obtained by splitting a dataset into a "low" and "high" risk group using all possible relative-risk thresholds.

Usage

```
tof_find_log_rank_threshold(input_data, relative_risk_col, time_col, event_col)
```

Arguments

`input_data`	A tbl_df or data.frame in which each observation is a row.
`relative_risk_col`	An unquote column name indicating which column contains the relative-risk estimates for each observation.
`time_col`	An unquoted column name indicating which column contains the true time-to-event information for each observation.
`event_col`	An unquoted column name indicating which column contains the outcome (event or censorship). Must be a binary column - all values should be either 0 or 1 (with 1 indicating the adverse event and 0 indicating censorship) or FALSE and TRUE (with TRUE indicating the adverse event and FALSE indicating censorship).

Value

A tibble with 3 columns: "candidate_thresholds" (the relative-risk threshold used for the log-rank test), "log_rank_p_val" (the p-values of the log-rank tests) and "is_best" (a logical value indicating which candidate threshold gave the optimal, i.e. smallest, p-value).

`tof_find_panel_info` *Use tidytof's opinionated heuristic for extracted a high-dimensional cytometry panel's metal-antigen pairs from a flowFrame (read from a .fcs file.)*

Description

Using the character vectors obtained from the ‘name‘ and ‘desc‘ columns of the parameters of the data of a flowFrame, figure out the high-dimensional cytometry panel used to collect the data and return it as a tidy tibble.

Usage

```
tof_find_panel_info(input_flowFrame)
```

Arguments

input_flowFrame	a raw flowFrame (just read from an .fcs file) from which a high-dimensional cytometry panel should be extracted

Value

A tibble with 2 columns (‘metals‘ and ‘antigens‘) that correspond to the metals and antigens of the high-dimensional cytometry panel used during data acquisition.

---

tof_fit_split *Fit a glmnet model and calculate performance metrics using a single rsplit object*

Description

This function trains a glmnet model on the training set of an rsplit object, then calculates performance metrics of that model on the validation/holdout set at all combinations of the mixture and penalty hyperparameters provided in a hyperparameter grid.

Usage

```
tof_fit_split(
  split_data,
  prepped_recipe,
  hyperparameter_grid,
  model_type,
  outcome_colnames
)
```

Arguments

split_data	An ‘rsplit‘ object from the rsample package. Alternatively, an unsplit tbl_df can be provided, though this is not recommended.
prepped_recipe	A trained recipe
hyperparameter_grid	A tibble containing the hyperparameter values to tune. Can be created using tof_create_grid
model_type	A string representing the type of glmnet model being fit.
outcome_colnames	Quoted column names indicating which columns in the data being fit represent the outcome variables (with all others assumed to be predictors).

Value

A tibble with the same number of rows as the input hyperparameter grid. Each row represents a combination of mixture and penalty, and each column contains a performance metric for the fitted glmnet model on ‘split_data‘’s holdout set. The specific performance metrics depend on the type of model being fit:

"linear" mean-squared error (‘mse‘) and mean absolute error (‘mae‘)

"two-class" binomial deviance (‘binomial_deviance‘); misclassification error rate ‘misclassification_error‘; the area under the receiver-operating curve (‘roc_auc‘); and ‘mse‘ and ‘mse‘ as above

"multiclass" multinomial deviance (‘multinomial_deviance‘); misclassification error rate ‘misclassification_error‘; the area under the receiver-operating curve (‘roc_auc‘) computed using the Hand-Till method in `roc_auc`; and ‘mse‘ and ‘mse‘ as above

"survival" the negative log2-transformed partial likelihood (‘neg_log_partial_likelihood‘) and Harrel’s concordance index (often simply called "C"; ‘concordance_index‘)

References

Harrel Jr, F. E. and Lee, K. L. and Mark, D. B. (1996) Tutorial in biostatistics: multivariable prognostic models: issues in developing models, evaluating assumptions and adequacy, and measuring and reducing error, Statistics in Medicine, 15, pages 361–387.

`tof_generate_palette` *Generate a color palette using tidytof.*

Description

This function generates a color palette based on the color palette of the author’s favorite pokemon.

Usage

```
tof_generate_palette(num_colors)
```

Arguments

`num_colors`	An integer specifying the number of colors you’d like to generate.

Value

A character vector of hex codes specifying the colors in the palette.

Examples

```
tof_generate_palette(num_colors = 5L)
```

`tof_get_model_mixture` *Get a 'tof_model''s optimal mixture (alpha) value*

Description

Get a 'tof_model''s optimal mixture (alpha) value

Usage

```
tof_get_model_mixture(tof_model)
```

Arguments

`tof_model`	A tof_model

Value

A numeric value

Examples

```
feature_tibble <-
    dplyr::tibble(
        sample = as.character(1:100),
        cd45 = runif(n = 100),
        pstat5 = runif(n = 100),
        cd34 = runif(n = 100),
        outcome = (3 * cd45) + (4 * pstat5) + rnorm(100),
        class =
            as.factor(
                dplyr::if_else(outcome > median(outcome), "class1", "class2")
            ),
        multiclass =
            as.factor(
                c(rep("class1", 30), rep("class2", 30), rep("class3", 40))
            ),
        event = c(rep(0, times = 30), rep(1, times = 70)),
        time_to_event = rnorm(n = 100, mean = 10, sd = 2)
    )

split_data <- tof_split_data(feature_tibble, split_method = "simple")

# train a regression model
regression_model <-
    tof_train_model(
        split_data = split_data,
        predictor_cols = c(cd45, pstat5, cd34),
        response_col = outcome,
        model_type = "linear"
    )
```

```
tof_get_model_mixture(regression_model)
```

`tof_get_model_outcomes` *Get a 'tof_model''s outcome variable name(s)*

Description

Get a 'tof_model''s outcome variable name(s)

Usage

```
tof_get_model_outcomes(tof_model)
```

Arguments

`tof_model`	A tof_model

Value

A character vector

Examples

```
feature_tibble <-
    dplyr::tibble(
        sample = as.character(1:100),
        cd45 = runif(n = 100),
        pstat5 = runif(n = 100),
        cd34 = runif(n = 100),
        outcome = (3 * cd45) + (4 * pstat5) + rnorm(100),
        class =
            as.factor(
                dplyr::if_else(outcome > median(outcome), "class1", "class2")
            ),
        multiclass =
            as.factor(
                c(rep("class1", 30), rep("class2", 30), rep("class3", 40))
            ),
        event = c(rep(0, times = 30), rep(1, times = 70)),
        time_to_event = rnorm(n = 100, mean = 10, sd = 2)
    )

split_data <- tof_split_data(feature_tibble, split_method = "simple")

# train a regression model
regression_model <-
    tof_train_model(
```

```
        split_data = split_data,
        predictor_cols = c(cd45, pstat5, cd34),
        response_col = outcome,
        model_type = "linear"
    )

tof_get_model_outcomes(regression_model)
```

`tof_get_model_penalty` *Get a 'tof_model''s optimal penalty (lambda) value*

Description

Get a 'tof_model''s optimal penalty (lambda) value

Usage

```
tof_get_model_penalty(tof_model)
```

Arguments

`tof_model`	A tof_model

Value

A numeric value

Examples

```
feature_tibble <-
    dplyr::tibble(
        sample = as.character(1:100),
        cd45 = runif(n = 100),
        pstat5 = runif(n = 100),
        cd34 = runif(n = 100),
        outcome = (3 * cd45) + (4 * pstat5) + rnorm(100),
        class =
            as.factor(
                dplyr::if_else(outcome > median(outcome), "class1", "class2")
            ),
        multiclass =
            as.factor(
                c(rep("class1", 30), rep("class2", 30), rep("class3", 40))
            ),
        event = c(rep(0, times = 30), rep(1, times = 70)),
        time_to_event = rnorm(n = 100, mean = 10, sd = 2)
    )

split_data <- tof_split_data(feature_tibble, split_method = "simple")
```

```
# train a regression model
regression_model <-
    tof_train_model(
        split_data = split_data,
        predictor_cols = c(cd45, pstat5, cd34),
        response_col = outcome,
        model_type = "linear"
    )

tof_get_model_penalty(regression_model)
```

`tof_get_model_training_data` *Get a 'tof_model''s training data*

Description

Get a 'tof_model''s training data

Usage

```
tof_get_model_training_data(tof_model)
```

Arguments

`tof_model`	A tof_model

Value

A tibble of (non-preprocessed) training data used to fit the model

Examples

```
feature_tibble <-
    dplyr::tibble(
        sample = as.character(1:100),
        cd45 = runif(n = 100),
        pstat5 = runif(n = 100),
        cd34 = runif(n = 100),
        outcome = (3 * cd45) + (4 * pstat5) + rnorm(100),
        class =
            as.factor(
                dplyr::if_else(outcome > median(outcome), "class1", "class2")
            ),
        multiclass =
            as.factor(
                c(rep("class1", 30), rep("class2", 30), rep("class3", 40))
            ),
```

```
        event = c(rep(0, times = 30), rep(1, times = 70)),
        time_to_event = rnorm(n = 100, mean = 10, sd = 2)
    )

split_data <- tof_split_data(feature_tibble, split_method = "simple")

# train a regression model
regression_model <-
    tof_train_model(
        split_data = split_data,
        predictor_cols = c(cd45, pstat5, cd34),
        response_col = outcome,
        model_type = "linear"
    )

tof_get_model_training_data(regression_model)
```

`tof_get_model_type` *Get a 'tof_model''s model type*

Description

Get a 'tof_model''s model type

Usage

```
tof_get_model_type(tof_model)
```

Arguments

`tof_model`	A tof_model

Value

A string

Examples

```
feature_tibble <-
    dplyr::tibble(
        sample = as.character(1:100),
        cd45 = runif(n = 100),
        pstat5 = runif(n = 100),
        cd34 = runif(n = 100),
        outcome = (3 * cd45) + (4 * pstat5) + rnorm(100),
        class =
            as.factor(
                dplyr::if_else(outcome > median(outcome), "class1", "class2")
            ),
```

```
        multiclass =
            as.factor(
                c(rep("class1", 30), rep("class2", 30), rep("class3", 40))
            ),
        event = c(rep(0, times = 30), rep(1, times = 70)),
        time_to_event = rnorm(n = 100, mean = 10, sd = 2)
    )

split_data <- tof_split_data(feature_tibble, split_method = "simple")

# train a regression model
regression_model <-
    tof_train_model(
        split_data = split_data,
        predictor_cols = c(cd45, pstat5, cd34),
        response_col = outcome,
        model_type = "linear"
    )

tof_get_model_type(regression_model)
```

`tof_get_model_x` *Get a 'tof_model''s processed predictor matrix (for glmnet)*

Description

Get a 'tof_model''s processed predictor matrix (for glmnet)

Usage

```
tof_get_model_x(tof_model)
```

Arguments

`tof_model`	A tof_model

Value

An x value formatted for glmnet

Examples

```
feature_tibble <-
    dplyr::tibble(
        sample = as.character(1:100),
        cd45 = runif(n = 100),
        pstat5 = runif(n = 100),
        cd34 = runif(n = 100),
        outcome = (3 * cd45) + (4 * pstat5) + rnorm(100),
```

```
        class =
            as.factor(
                dplyr::if_else(outcome > median(outcome), "class1", "class2")
            ),
        multiclass =
            as.factor(
                c(rep("class1", 30), rep("class2", 30), rep("class3", 40))
            ),
        event = c(rep(0, times = 30), rep(1, times = 70)),
        time_to_event = rnorm(n = 100, mean = 10, sd = 2)
    )

split_data <- tof_split_data(feature_tibble, split_method = "simple")

# train a regression model
regression_model <-
    tof_train_model(
        split_data = split_data,
        predictor_cols = c(cd45, pstat5, cd34),
        response_col = outcome,
        model_type = "linear"
    )

tof_get_model_x(regression_model)
```

`tof_get_model_y` *Get a 'tof_model''s processed outcome variable matrix (for glmnet)*

Description

Get a 'tof_model''s processed outcome variable matrix (for glmnet)

Usage

```
tof_get_model_y(tof_model)
```

Arguments

`tof_model`	A tof_model

Value

A y value formatted for glmnet

Examples

```
feature_tibble <-
    dplyr::tibble(
        sample = as.character(1:100),
        cd45 = runif(n = 100),
        pstat5 = runif(n = 100),
        cd34 = runif(n = 100),
        outcome = (3 * cd45) + (4 * pstat5) + rnorm(100),
        class =
            as.factor(
                dplyr::if_else(outcome > median(outcome), "class1", "class2")
            ),
        multiclass =
            as.factor(
                c(rep("class1", 30), rep("class2", 30), rep("class3", 40))
            ),
        event = c(rep(0, times = 30), rep(1, times = 70)),
        time_to_event = rnorm(n = 100, mean = 10, sd = 2)
    )

split_data <- tof_split_data(feature_tibble, split_method = "simple")

# train a regression model
regression_model <-
    tof_train_model(
        split_data = split_data,
        predictor_cols = c(cd45, pstat5, cd34),
        response_col = outcome,
        model_type = "linear"
    )

tof_get_model_y(regression_model)
```

`tof_get_panel` *Get panel information from a tof_tibble*

Description

Get panel information from a tof_tibble

Usage

```
tof_get_panel(tof_tibble)
```

Arguments

`tof_tibble`	A 'tof_tbl'.

Value

A tibble containing information about the CyTOF panel that was used during data acquisition for the data contained in ‘tof_tibble‘.

See Also

Other tof_tbl utilities: `new_tof_tibble()`, `tof_set_panel()`

Examples

```
input_file <- dir(tidytof_example_data("aml"), full.names = TRUE)[[1]]
tof_tibble <- tof_read_data(input_file)
tof_get_panel(tof_tibble)
```

---

tof_is_numeric *Find if a vector is numeric*

Description

This function takes an input vector ‘.vec‘ and checks if it is either an integer or a double (i.e. is the type of vector that might encode high-dimensional cytometry measurements).

Usage

```
tof_is_numeric(.vec)
```

Arguments

`.vec`	A vector.

Value

A boolean value indicating if .vec is of type integer or double.

tof_knn_density *Estimate cells' local densities using K-nearest-neighbor density estimation*

Description

This function uses the distances between a cell and each of its K nearest neighbors to estimate local density of each cell in a 'tof_tbl' or 'tibble' containing high-dimensional cytometry data.

Usage

```
tof_knn_density(
  tof_tibble,
  distance_cols = where(tof_is_numeric),
  num_neighbors = min(15L, nrow(tof_tibble)),
  distance_function = c("euclidean", "cosine", "l2", "ip"),
  estimation_method = c("mean_distance", "sum_distance"),
  normalize = TRUE,
  ...
)
```

Arguments

tof_tibble	A 'tof_tbl' or a 'tibble'.
distance_cols	Unquoted names of the columns in 'tof_tibble' to use in calculating cell-to-cell distances during the local density estimation for each cell. Defaults to all numeric columns in 'tof_tibble'.
num_neighbors	An integer indicating the number of nearest neighbors to use in estimating the local density of each cell. Defaults to the minimum of 15 and the number of rows in 'tof_tibble'.
distance_function	A string indicating which distance function to use for calculating cell-to-cell distances during local density estimation. Options include "euclidean" (the default) and "cosine".
estimation_method	A string indicating how the relative density for each cell should be calculated from the distances between it and each of its k nearest neighbors. Options are "mean_distance" (the default; estimates the relative density for a cell's neighborhood by taking the negative average of the distances to its nearest neighbors) and "sum_distance" (estimates the relative density for a cell's neighborhood by taking the negative sum of the distances to its nearest neighbors).
normalize	A boolean value indicating if the vector of local density estimates should be normalized to values between 0 and 1. Defaults to TRUE.
...	Additional optional arguments to pass to tof_find_knn.

Value

A tibble with a single column named ".knn_density" containing the local density estimates for each input cell in ‘tof_tibble‘.

See Also

Other local density estimation functions: `tof_estimate_density()`, `tof_spade_density()`

`tof_log_rank_test` *Compute the log-rank test p-value for the difference between the two survival curves obtained by splitting a dataset into a "low" and "high" risk group using a given relative-risk threshold.*

Description

Compute the log-rank test p-value for the difference between the two survival curves obtained by splitting a dataset into a "low" and "high" risk group using a given relative-risk threshold.

Usage

```
tof_log_rank_test(
  input_data,
  relative_risk_col,
  time_col,
  event_col,
  threshold
)
```

Arguments

`input_data`	A tbl_df or data.frame in which each observation is a row.
`relative_risk_col`	An unquote column name indicating which column contains the relative-risk estimates for each observation.
`time_col`	An unquoted column name indicating which column contains the true time-to-event information for each observation.
`event_col`	An unquoted column name indicating which column contains the outcome (event or censorship). Must be a binary column - all values should be either 0 or 1 (with 1 indicating the adverse event and 0 indicating censorship) or FALSE and TRUE (with TRUE indicating the adverse event and FALSE indicating censorship).
`threshold`	A numeric value indicating the relative-risk threshold that should be used to split observations into low- and high-risk groups.

Value

A numeric value <1, the p-value of the log-rank test.

Examples

```
NULL
```

`tof_make_knn_graph` *Title*

Description

Title

Usage

```
tof_make_knn_graph(
  tof_tibble,
  knn_cols,
  num_neighbors,
  distance_function = c("euclidean", "cosine"),
  graph_type = c("weighted", "unweighted"),
  ...
)
```

Arguments

`tof_tibble`	A tibble or tof_tbl.
`knn_cols`	Unquoted column names indicating which columns in tof_tibble should be used for the KNN calculation.
`num_neighbors`	An integer number of neighbors to find for each cell (not including itself).
`distance_function`	A string indicating which distance function to use for the nearest-neighbor calculation. Options include "euclidean" (the default) and "cosine" distances.
`graph_type`	A string indicating if the graph's edges should have weights ("weighted"; the default) or not ("unweighted").
`...`	Optional additional arguments to pass to `tof_find_knn`

Value

A `tbl_graph`.

Examples

```
NULL
```

`tof_make_roc_curve` *Compute a receiver-operating curve (ROC) for a two-class or multi-class dataset*

Description

Compute a receiver-operating curve (ROC) for a two-class or multiclass dataset

Usage

```
tof_make_roc_curve(input_data, truth_col, prob_cols)
```

Arguments

`input_data`	A tof_tbl, tbl_df, or data.frame in which each row is an observation.
`truth_col`	An unquoted column name indicating which column in ‘input_data‘ contains the true class labels for each observation. Must be a factor.
`prob_cols`	Unquoted column names indicating which columns in ‘input_data‘ contain the probability estimates for each class in ‘truth_col‘. These columns must be specified in the same order as the factor levels in ‘truth_col‘.

Value

A tibble that can be used to plot the ROC for a classification task. For each candidate probability threshold, the following are reported: specificity, sensitivity, true-positive rate (tpr), and false-positive rate (fpr).

Examples

```
feature_tibble <-
    dplyr::tibble(
        sample = as.character(1:100),
        cd45 = runif(n = 100),
        pstat5 = runif(n = 100),
        cd34 = runif(n = 100),
        outcome = (3 * cd45) + (4 * pstat5) + rnorm(100),
        class =
            as.factor(
                dplyr::if_else(outcome > median(outcome), "class1", "class2")
            )
    )

split_data <- tof_split_data(feature_tibble, split_method = "simple")

# train a logistic regression classifier
log_model <-
    tof_train_model(
        split_data = split_data,
```

```
        predictor_cols = c(cd45, pstat5, cd34),
        response_col = class,
        model_type = "two-class"
    )

# make predictions
predictions <-
    tof_predict(
        log_model,
        new_data = feature_tibble,
        prediction_type = "response"
    )
prediction_tibble <-
    dplyr::tibble(
        truth = feature_tibble$class,
        prediction = predictions$.pred
    )

# make ROC curve
tof_make_roc_curve(
    input_data = prediction_tibble,
    truth_col = truth,
    prob_cols = prediction
)
```

`tof_metacluster` *Metacluster clustered CyTOF data.*

Description

This function is a wrapper around tidytof's tof_metacluster_* function family. It performs metaclustering on CyTOF data using a user-specified method (of 5 choices) and each method's corresponding input parameters.

Usage

```
tof_metacluster(
  tof_tibble,
  cluster_col,
  metacluster_cols = where(tof_is_numeric),
  central_tendency_function = stats::median,
  ...,
  augment = TRUE,
  method = c("consensus", "hierarchical", "kmeans", "phenograph", "flowsom")
)
```

Arguments

tof_tibble	A ‘tof_tbl‘ or ‘tibble‘.
cluster_col	An unquoted column name indicating which column in ‘tof_tibble‘ stores the cluster ids for the cluster to which each cell belongs. Cluster labels can be produced via any method the user chooses - including manual gating, any of the functions in the ‘tof_cluster_*‘ function family, or any other method.
metacluster_cols	Unquoted column names indicating which columns in ‘tof_tibble‘ to use in computing the metaclusters. Defaults to all numeric columns in ‘tof_tibble‘. Supports tidyselect helpers.
central_tendency_function	The function that should be used to calculate the measurement of central tendency for each cluster before metaclustering. This function will be used to compute a summary statistic for each input cluster in ‘cluster_col‘ across all columns specified by ‘metacluster_cols‘, and the resulting vector (one for each cluster) will be used as the input for metaclustering. Defaults to median.
...	Additional arguments to pass to the ‘tof_metacluster_*‘ function family member corresponding to the chosen ‘method‘.
augment	A boolean value indicating if the output should column-bind the metacluster ids of each cell as a new column in ‘tof_tibble‘ (TRUE; the default) or if a single-column tibble including only the metacluster ids should be returned (FALSE).
method	A string indicating which clustering method should be used. Valid values include "consensus", "hierarchical", "kmeans", "phenograph", and "flowsom".

Value

A ‘tof_tbl‘ or ‘tibble‘ If augment = FALSE, it will have a single column encoding the metacluster ids for each cell in ‘tof_tibble‘. If augment = TRUE, it will have ncol(tof_tibble) + 1 columns: each of the (unaltered) columns in ‘tof_tibble‘ plus an additional column encoding the metacluster ids.

See Also

Other metaclustering functions: tof_metacluster_consensus(), tof_metacluster_flowsom(), tof_metacluster_hierarchical(), tof_metacluster_kmeans(), tof_metacluster_phenograph()

Examples

```
sim_data <-
    dplyr::tibble(
        cd45 = rnorm(n = 1000),
        cd38 = rnorm(n = 1000),
        cd34 = rnorm(n = 1000),
        cd19 = rnorm(n = 1000),
        cluster_id = sample(letters, size = 1000, replace = TRUE)
    )

tof_metacluster(
    tof_tibble = sim_data,
```

```
    cluster_col = cluster_id,
    clustering_algorithm = "consensus",
    method = "flowsom"
)

tof_metacluster(
    tof_tibble = sim_data,
    cluster_col = cluster_id,
    method = "phenograph"
)
```

`tof_metacluster_consensus` *Metacluster clustered CyTOF data using consensus clustering*

Description

This function performs consensus metaclustering on a ‘tof_tbl‘ containing CyTOF data using a user-specified selection of input variables/CyTOF measurements and the number of desired metaclusters. See `ConsensusClusterPlus` for additional details.

Usage

```
tof_metacluster_consensus(
  tof_tibble,
  cluster_col,
  metacluster_cols = where(tof_is_numeric),
  central_tendency_function = stats::median,
  num_metaclusters = 10L,
  proportion_clusters = 0.9,
  proportion_features = 1,
  num_reps = 20L,
  clustering_algorithm = c("hierarchical", "pam", "kmeans"),
 distance_function = c("euclidean", "minkowski", "pearson", "spearman", "maximum",
    "binary", "canberra"),
  ...
)
```

Arguments

`tof_tibble`	A ‘tof_tbl‘ or ‘tibble‘.
`cluster_col`	An unquoted column name indicating which column in ‘tof_tibble‘ stores the cluster ids for the cluster to which each cell belongs. Cluster labels can be produced via any method the user chooses - including manual gating, any of the functions in the ‘tof_cluster_*‘ function family, or any other method.

`metacluster_cols`
: Unquoted column names indicating which columns in 'tof_tibble' to use in computing the metaclusters. Defaults to all numeric columns in 'tof_tibble'. Supports tidyselect helpers.

`central_tendency_function`
: The function that should be used to calculate the measurement of central tendency for each cluster before metaclustering. This function will be used to compute a summary statistic for each input cluster in 'cluster_col' across all columns specified by 'metacluster_cols', and the resulting vector (one for each cluster) will be used as the input for metaclustering. Defaults to `median`.

`num_metaclusters`
: An integer indicating the number of clusters that should be returned. Defaults to 10.

`proportion_clusters`
: A numeric value between 0 and 1 indicating the proportion of clusters to subsample (from the total number of clusters in 'cluster_col') during each iteration of the consensus clustering. Defaults to 0.9

`proportion_features`
: A numeric value between 0 and 1 indicating the proportion of features (i.e. the proportion of columns specified by 'metacluster_cols') to subsample during each iteration of the consensus clustering. Defaults to 1 (all features are included).

`num_reps`
: An integer indicating how many subsampled replicates to run during consensus clustering. Defaults to 20.

`clustering_algorithm`
: A string indicating which clustering algorithm `ConsensusClusterPlus` should use to metacluster the subsampled clusters during each resampling. Options are "hierarchical" (the default), "pam" (partitioning around medoids), and "kmeans".

`distance_function`
: A string indicating which distance function should be used to compute the distances between clusters during consensus clustering. Options are "euclidean" (the default), "manhattan", "minkowski", "pearson", "spearman", "maximum", "binary", and "canberra". See `ConsensusClusterPlus`.

`...`
: Optional additional arguments to pass to `ConsensusClusterPlus`.

Value

A tibble with a single column ('.consensus_metacluster') and the same number of rows as the input 'tof_tibble'. Each entry in the column indicates the metacluster label assigned to the same row in 'tof_tibble'.

See Also

Other metaclustering functions: `tof_metacluster()`, `tof_metacluster_flowsom()`, `tof_metacluster_hierarchical()`
`tof_metacluster_kmeans()`, `tof_metacluster_phenograph()`

Examples

```
sim_data <-
    dplyr::tibble(
        cd45 = rnorm(n = 1000),
        cd38 = rnorm(n = 1000),
        cd34 = rnorm(n = 1000),
        cd19 = rnorm(n = 1000),
        cluster_id = sample(letters, size = 1000, replace = TRUE)
    )

tof_metacluster_consensus(tof_tibble = sim_data, cluster_col = cluster_id)
```

---

tof_metacluster_flowsom

Metacluster clustered CyTOF data using FlowSOM's built-in metaclustering algorithm

---

Description

This function performs metaclustering on a 'tof_tbl' containing CyTOF data using a user-specified selection of input variables/CyTOF measurements and the number of desired metaclusters. It takes advantage of the FlowSOM package's built-in functionality for automatically detecting the number of metaclusters and can use several strategies as adapted by the FlowSOM team: consensus metaclustering, hierarchical metaclustering, k-means metaclustering, or metaclustering using the FlowSOM algorithm itself. See `MetaClustering` for additional details.

Usage

```
tof_metacluster_flowsom(
  tof_tibble,
  cluster_col,
  metacluster_cols = where(tof_is_numeric),
  central_tendency_function = stats::median,
  num_metaclusters = 10L,
  clustering_algorithm = c("consensus", "hierarchical", "kmeans", "som"),
  ...
)
```

Arguments

`tof_tibble`	A 'tof_tbl' or 'tibble'.
`cluster_col`	An unquoted column name indicating which column in 'tof_tibble' stores the cluster ids for the cluster to which each cell belongs. Cluster labels can be produced via any method the user chooses - including manual gating, any of the functions in the 'tof_cluster_*' function family, or any other method.

`metacluster_cols`
: Unquoted column names indicating which columns in 'tof_tibble' to use in computing the metaclusters. Defaults to all numeric columns in 'tof_tibble'. Supports tidyselect helpers.

`central_tendency_function`
: The function that should be used to calculate the measurement of central tendency for each cluster before metaclustering. This function will be used to compute a summary statistic for each input cluster in 'cluster_col' across all columns specified by 'metacluster_cols', and the resulting vector (one for each cluster) will be used as the input for metaclustering. Defaults to `median`.

`num_metaclusters`
: An integer indicating the maximum number of clusters that should be returned. Defaults to 10. Note that for this function, the output may provide a small number of metaclusters than requested. This is because `MetaClustering` uses the "Elbow method" to automatically detect the optimal number of metaclusters.

`clustering_algorithm`
: A string indicating which clustering algorithm `MetaClustering` should use to perform the metaclustering. Options are "consensus" (the default), "hierarchical", "kmeans", and "som" (i.e. self-organizing map; the FlowSOM algorithm itself).

`...`
: Optional additional arguments to pass to `MetaClustering`.

Value

A tibble with a single column ('.flowsom_metacluster') and the same number of rows as the input 'tof_tibble'. Each entry in the column indicates the metacluster label assigned to the same row in 'tof_tibble'.

See Also

Other metaclustering functions: `tof_metacluster()`, `tof_metacluster_consensus()`, `tof_metacluster_hierarchical`
`tof_metacluster_kmeans()`, `tof_metacluster_phenograph()`

Examples

```
sim_data <-
    dplyr::tibble(
        cd45 = rnorm(n = 1000),
        cd38 = rnorm(n = 1000),
        cd34 = rnorm(n = 1000),
        cd19 = rnorm(n = 1000),
        cluster_id = sample(letters, size = 1000, replace = TRUE)
    )

tof_metacluster_flowsom(
    tof_tibble = sim_data,
    cluster_col = cluster_id,
    clustering_algorithm = "consensus"
)
```

```
tof_metacluster_flowsom(
    tof_tibble = sim_data,
    cluster_col = cluster_id,
    clustering_algorithm = "som"
)
```

`tof_metacluster_hierarchical` *Metacluster clustered CyTOF data using hierarchical agglomerative clustering*

Description

This function performs hierarchical metaclustering on a ‘tof_tbl‘ containing CyTOF data using a user-specified selection of input variables/CyTOF measurements and the number of desired metaclusters. See `hclust`.

Usage

```
tof_metacluster_hierarchical(
  tof_tibble,
  cluster_col,
  metacluster_cols = where(tof_is_numeric),
  central_tendency_function = stats::median,
  num_metaclusters = 10L,
 distance_function = c("euclidean", "manhattan", "minkowski", "maximum", "canberra",
    "binary"),
 agglomeration_method = c("complete", "single", "average", "median", "centroid",
    "ward.D", "ward.D2", "mcquitty")
)
```

Arguments

`tof_tibble`	A ‘tof_tbl‘ or ‘tibble‘.
`cluster_col`	An unquoted column name indicating which column in ‘tof_tibble‘ stores the cluster ids for the cluster to which each cell belongs. Cluster labels can be produced via any method the user chooses - including manual gating, any of the functions in the ‘tof_cluster_*‘ function family, or any other method.
`metacluster_cols`	Unquoted column names indicating which columns in ‘tof_tibble‘ to use in computing the metaclusters. Defaults to all numeric columns in ‘tof_tibble‘. Supports tidyselect helpers.
`central_tendency_function`	The function that should be used to calculate the measurement of central tendency for each cluster before metaclustering. This function will be used to compute a summary statistic for each input cluster in ‘cluster_col‘ across all columns

specified by ‘metacluster_cols‘, and the resulting vector (one for each cluster) will be used as the input for metaclustering. Defaults to median.

num_metaclusters
: An integer indicating the number of clusters that should be returned. Defaults to 10.

distance_function
: A string indicating which distance function should be used to compute the distances between clusters during the hierarchical metaclustering. Options are "euclidean" (the default), "manhattan", "minkowski", "maximum", "canberra", and "binary". See dist for additional details.

agglomeration_method
: A string indicating which agglomeration algorithm should be used during hierarchical cluster combination. Options are "complete" (the default), "single", "average", "median", "centroid", "ward.D", "ward.D2", and "mcquitty". See hclust for details.

Value

A tibble with a single column (‘.hierarchical_metacluster‘) and the same number of rows as the input ‘tof_tibble‘. Each entry in the column indicates the metacluster label assigned to the same row in ‘tof_tibble‘.

See Also

Other metaclustering functions: tof_metacluster(), tof_metacluster_consensus(), tof_metacluster_flowsom(), tof_metacluster_kmeans(), tof_metacluster_phenograph()

Examples

```
sim_data <-
    dplyr::tibble(
        cd45 = rnorm(n = 1000),
        cd38 = rnorm(n = 1000),
        cd34 = rnorm(n = 1000),
        cd19 = rnorm(n = 1000),
        cluster_id = sample(letters, size = 1000, replace = TRUE)
    )

tof_metacluster_hierarchical(tof_tibble = sim_data, cluster_col = cluster_id)
```

---

tof_metacluster_kmeans

Metacluster clustered CyTOF data using k-means clustering

Description

This function performs k-means metaclustering on a ‘tof_tbl‘ containing CyTOF data using a user-specified selection of input variables/CyTOF measurements and the number of desired metaclusters. See hclust.

Usage

```
tof_metacluster_kmeans(
  tof_tibble,
  cluster_col,
  metacluster_cols = where(tof_is_numeric),
  central_tendency_function = stats::median,
  num_metaclusters = 10L,
  ...
)
```

Arguments

tof_tibble	A ‘tof_tbl‘ or ‘tibble‘.
cluster_col	An unquoted column name indicating which column in ‘tof_tibble‘ stores the cluster ids for the cluster to which each cell belongs. Cluster labels can be produced via any method the user chooses - including manual gating, any of the functions in the ‘tof_cluster_*‘ function family, or any other method.
metacluster_cols	Unquoted column names indicating which columns in ‘tof_tibble‘ to use in computing the metaclusters. Defaults to all numeric columns in ‘tof_tibble‘. Supports tidyselect helpers.
central_tendency_function	The function that should be used to calculate the measurement of central tendency for each cluster before metaclustering. This function will be used to compute a summary statistic for each input cluster in ‘cluster_col‘ across all columns specified by ‘metacluster_cols‘, and the resulting vector (one for each cluster) will be used as the input for metaclustering. Defaults to median.
num_metaclusters	An integer indicating the number of clusters that should be returned. Defaults to 10.
...	Optional additional method specifications to pass to tof_cluster_kmeans.

Value

A tibble with a single column (‘.kmeans_metacluster‘) and the same number of rows as the input ‘tof_tibble‘. Each entry in the column indicates the metacluster label assigned to the same row in ‘tof_tibble‘.

See Also

Other metaclustering functions: tof_metacluster(), tof_metacluster_consensus(), tof_metacluster_flowsom(), tof_metacluster_hierarchical(), tof_metacluster_phenograph()

Examples

```
sim_data <-
    dplyr::tibble(
        cd45 = rnorm(n = 1000),
        cd38 = rnorm(n = 1000),
        cd34 = rnorm(n = 1000),
        cd19 = rnorm(n = 1000),
        cluster_id = sample(letters, size = 1000, replace = TRUE)
    )

tof_metacluster_kmeans(tof_tibble = sim_data, cluster_col = cluster_id)
```

---

tof_metacluster_phenograph

Metacluster clustered CyTOF data using PhenoGraph clustering

Description

This function performs PhenoGraph metaclustering on a ‘tof_tbl‘ containing CyTOF data using a user-specified selection of input variables/CyTOF measurements. The number of metaclusters is automatically detected by the PhenoGraph algorithm. See tof_cluster_phenograph.

Usage

```
tof_metacluster_phenograph(
  tof_tibble,
  cluster_col,
  metacluster_cols = where(tof_is_numeric),
  central_tendency_function = stats::median,
  num_neighbors = 5L,
  ...
)
```

Arguments

tof_tibble	A ‘tof_tbl‘ or ‘tibble‘.
cluster_col	An unquoted column name indicating which column in ‘tof_tibble‘ stores the cluster ids for the cluster to which each cell belongs. Cluster labels can be produced via any method the user chooses - including manual gating, any of the functions in the ‘tof_cluster_*‘ function family, or any other method.
metacluster_cols	Unquoted column names indicating which columns in ‘tof_tibble‘ to use in computing the metaclusters. Defaults to all numeric columns in ‘tof_tibble‘. Supports tidyselect helpers.

central_tendency_function
: The function that should be used to calculate the measurement of central tendency for each cluster before metaclustering. This function will be used to compute a summary statistic for each input cluster in 'cluster_col' across all columns specified by 'metacluster_cols', and the resulting vector (one for each cluster) will be used as the input for metaclustering. Defaults to median.

num_neighbors
: An integer indicating the number of neighbors to use when constructing PhenoGraph's k-nearest-neighbor graph. Smaller values emphasize local graph structure; larger values emphasize global graph structure (and will add time to the computation). Defaults to 5.

...
: Optional additional method specifications to pass to tof_cluster_phenograph.

Value

A tibble with a single column ('.phenograph_metacluster') and the same number of rows as the input 'tof_tibble'. Each entry in the column indicates the metacluster label assigned to the same row in 'tof_tibble'.

See Also

Other metaclustering functions: tof_metacluster(), tof_metacluster_consensus(), tof_metacluster_flowsom(), tof_metacluster_hierarchical(), tof_metacluster_kmeans()

Examples

```
sim_data <-
    dplyr::tibble(
        cd45 = rnorm(n = 1000),
        cd38 = rnorm(n = 1000),
        cd34 = rnorm(n = 1000),
        cd19 = rnorm(n = 1000),
        cluster_id = sample(letters, size = 1000, replace = TRUE)
    )

tof_metacluster_phenograph(tof_tibble = sim_data, cluster_col = cluster_id)
```

---

tof_plot_cells_density

Plot marker expression density plots

Description

This function plots marker expression density plots for a user-specified column in a tof_tbl. Optionally, cells can be grouped to plot multiple vertically-arranged density plots

Usage

```
tof_plot_cells_density(
  tof_tibble,
  marker_col,
  group_col,
  num_points = 512,
  theme = ggplot2::theme_bw(),
  use_ggridges = FALSE,
  scale = 1,
  ...
)
```

Arguments

tof_tibble	A 'tof_tbl' or a 'tibble'.
marker_col	An unquoted column name representing which column in 'tof_tibble' (i.e. which CyTOF protein measurement) should be included in the feature extraction calculation.
group_col	Unquoted column names representing which column in 'tof_tibble' should be used to break the rows of 'tof_tibble' into subgroups to be plotted as separate histograms. Defaults to plotting without subgroups.
num_points	The number of points along the full range of 'marker_col' at which the density should be calculated
theme	The ggplot2 theme for the plot. Defaults to theme_bw
use_ggridges	A boolean value indicting if geom_ridgeline should be used to plot overlain histograms. Defaults to FALSE. If TRUE, the ggridges package must be installed.
scale	Use to set the 'scale' argument in geom_ridgeline, which controls how far apart (vertically) density plots are arranged along the y-axis. Defaults to 1.
...	Additional optional arguments to send to geom_ridgeline.

Value

A ggplot object

Examples

```
sim_data <-
    dplyr::tibble(
        cd45 = rnorm(n = 1000),
        cd38 = rnorm(n = 1000),
        cd34 = rnorm(n = 1000),
        cd19 = rnorm(n = 1000),
        cluster_id = sample(c("a", "b"), size = 1000, replace = TRUE)
    )

density_plot <-
```

```
tof_plot_cells_density(
    tof_tibble = sim_data,
    marker_col = cd45,
    group_col = cluster_id
)
```

`tof_plot_cells_embedding` *Plot scatterplots of single-cell data using low-dimensional feature embeddings*

Description

This function makes scatterplots using single-cell data embedded in a low-dimensional space (such as that generated by `tof_reduce_dimensions`, with each point colored using a user-specified variable.

Usage

```
tof_plot_cells_embedding(
  tof_tibble,
  embedding_cols,
  color_col,
  facet_cols,
  compute_embedding_cols = where(tof_is_numeric),
  embedding_method = c("pca", "tsne", "umap"),
  embedding_args = list(),
  theme = ggplot2::theme_bw(),
  ...,
  method = c("ggplot2", "scattermore")
)
```

Arguments

`tof_tibble`	A ‘tof_tbl‘ or a ‘tibble‘.
`embedding_cols`	Unquoted column names indicating which columns in ‘tof_tibble‘ should be used as the x and y axes of the scatterplot. Supports tidyselect helpers. Must select exactly 2 columns. If not provided, a feature embedding can be computed from scratch using the method provided using the ‘embedding_method‘ argument and the `tof_reduce_dimensions` arguments passed to ‘embedding_args‘.
`color_col`	An unquoted column name specifying which column in ‘tof_tibble‘ should be used to color each point in the scatterplot.
`facet_cols`	An unquoted column name specifying which column in ‘tof_tibble‘ should be used to break the scatterplot into facets using `facet_wrap`.

compute_embedding_cols
: Unquoted column names indicating which columns in 'tof_tibble' to use for computing the embeddings with the method specified by 'embedding_method'. Defaults to all numeric columns in 'tof_tibble'. Supports tidyselect helpers.

embedding_method
: A string indicating which method should be used for the feature embedding (if 'embedding_cols' are not provided). Options (which are passed to tof_reduce_dimensions) are "pca" (the default), "tsne", and "umap".

embedding_args
: Optional additional arguments to pass to tof_reduce_dimensions. For example, for 'method = "tsne"', these might include 'num_comp', 'perplexity', and 'theta'.

theme
: A ggplot2 theme to apply to the scatterplot. Defaults to theme_bw.

...
: Optional additional arguments to pass to tof_plot_cells_scatter.

method
: A string indicating which plotting engine should be used. Valid values include "ggplot2" (the default) and "scattermore" (recommended if more than 100K cells are being plotted). Note that method = "scattermore" requires the scattermore package to be installed.

Value

A ggplot object.

See Also

Other visualization functions: tof_plot_cells_layout(), tof_plot_cells_scatter()

Examples

```
sim_data <-
    dplyr::tibble(
        cd45 = rnorm(n = 1000),
        cd38 = c(rnorm(n = 500), rnorm(n = 500, mean = 2)),
        cd34 = c(rnorm(n = 500), rnorm(n = 500, mean = 4)),
        cd19 = rnorm(n = 1000),
        cluster_id = c(rep("a", 500), rep("b", 500))
    )

# embed with pca
pca_plot <-
    tof_plot_cells_embedding(
        tof_tibble = sim_data,
        color_col = cd38,
        embedding_method = "pca",
        compute_embedding_cols = starts_with("cd")
    )

# embed with tsne
tsne_plot <-
    tof_plot_cells_embedding(
        tof_tibble = sim_data,
```

```
    color_col = cluster_id,
    embedding_method = "tsne",
    compute_embedding_cols = starts_with("cd")
)
```

`tof_plot_cells_layout` *Plot force-directed layouts of single-cell data*

Description

This function makes force-directed layouts using single-cell data embedded in a 2-dimensional space representing a k-nearest-neighbor graph constructed using cell-to-cell similarities. Each node in the force-directed layout represents a single cell colored using a user-specified variable.

Usage

```
tof_plot_cells_layout(
  tof_tibble,
  knn_cols = where(tof_is_numeric),
  color_col,
  facet_cols,
  num_neighbors = 5,
  graph_type = c("weighted", "unweighted"),
  graph_layout = "fr",
  distance_function = c("euclidean", "cosine"),
  edge_alpha = 0.25,
  node_size = 2,
  theme = ggplot2::theme_void(),
  ...
)
```

Arguments

`tof_tibble`	A ‘tof_tbl‘ or a ‘tibble‘.
`knn_cols`	Unquoted column names indicating which columns in ‘tof_tibble‘ should be used to compute the cell-to-cell distances used to construct the k-nearest-neighbor graph. Supports tidyselect helpers. Defaults to all numeric columns.
`color_col`	Unquoted column name indicating which column in ‘tof_tibble‘ should be used to color the nodes in the force-directed layout.
`facet_cols`	Unquoted column names indicating which columns in ‘tof_tibble‘ should be used to separate nodes into different force-directed layouts.
`num_neighbors`	An integer specifying how many neighbors should be used to construct the k-nearest neighbor graph.
`graph_type`	A string specifying if the k-nearest neighbor graph should be "weighted" (the default) or "unweighted".

graph_layout	A string specifying which algorithm should be used to compute the force-directed layout. Passed to ggraph. Defaults to "fr", the Fruchterman-Reingold algorithm. Other examples include "nicely", "gem", "kk", and many others. See layout_tbl_graph_igraph for other examples.
distance_function	A string indicating which distance function to use in computing the cell-to-cell distances. Valid options include "euclidean" (the default) and "cosine".
edge_alpha	A numeric value between 0 and 1 specifying the transparency of the edges drawn in the force-directed layout. Defaults to 0.25.
node_size	A numeric value specifying the size of the nodes in the force-directed layout. Defaults to 2.
theme	A ggplot2 theme to apply to the force-directed layout. Defaults to theme_void
...	hnsw_knn

Value

A ggraph/ggplot object.

See Also

Other visualization functions: tof_plot_cells_embedding(), tof_plot_cells_scatter()

Examples

```
sim_data <-
    dplyr::tibble(
        cd45 = rnorm(n = 1000),
        cd38 = c(rnorm(n = 500), rnorm(n = 500, mean = 2)),
        cd34 = c(rnorm(n = 500), rnorm(n = 500, mean = 4)),
        cd19 = rnorm(n = 1000),
        cluster_id = c(rep("a", 500), rep("b", 500))
    )

# make a layout colored by a marker
layout_cd38 <-
    tof_plot_cells_layout(
        tof_tibble = sim_data,
        color_col = cd38
    )

# make a layout colored by cluster id
layout_cluster <-
    tof_plot_cells_layout(
        tof_tibble = sim_data,
        color_col = cluster_id,
    )
```

`tof_plot_cells_scatter` *Plot scatterplots of single-cell data.*

Description

This function makes scatterplots of single-cell data using user-specified x- and y-axes. Additionally, each point in the scatterplot can be colored using a user-specified variable.

Usage

```
tof_plot_cells_scatter(
  tof_tibble,
  x_col,
  y_col,
  color_col,
  facet_cols,
  theme = ggplot2::theme_bw(),
  ...,
  method = c("ggplot2", "scattermore")
)
```

Arguments

`tof_tibble`	A ‘tof_tbl‘ or a ‘tibble‘.
`x_col`	An unquoted column name specifying which column in ‘tof_tibble‘ should be used as the x-axis.
`y_col`	An unquoted column name specifying which column in ‘tof_tibble‘ should be used as the y-axis.
`color_col`	An unquoted column name specifying which column in ‘tof_tibble‘ should be used to color each point in the scatterplot.
`facet_cols`	An unquoted column name specifying which column in ‘tof_tibble‘ should be used to break the scatterplot into facets using `facet_wrap`.
`theme`	A ggplot2 theme to apply to the scatterplot. Defaults to `theme_bw`.
`...`	Optional additional arguments to pass to `geom_point` if `method = "ggplot2"` or `geom_scattermore` if `method = "scattermore"`.
`method`	A string indicating which plotting engine should be used. Valid values include "ggplot2" (the default) and "scattermore" (recommended if more than 100K cells are being plotted). Note that `method = "scattermore"` requires the scattermore package to be installed.

Value

A ggplot object.

See Also

Other visualization functions: tof_plot_cells_embedding(), tof_plot_cells_layout()

Examples

```
sim_data <-
    dplyr::tibble(
        cd45 = rnorm(n = 1000),
        cd38 = c(rnorm(n = 500), rnorm(n = 500, mean = 2)),
        cd34 = c(rnorm(n = 500), rnorm(n = 500, mean = 4)),
        cd19 = rnorm(n = 1000),
        cluster_id = c(rep("a", 500), rep("b", 500))
    )
```

---

tof_plot_clusters_heatmap

Make a heatmap summarizing cluster marker expression patterns in CyTOF data

---

Description

This function makes a heatmap of cluster-to-cluster marker expression patterns in single-cell data. Markers are plotted along the horizontal (x-) axis of the heatmap and cluster IDs are plotted along the vertical (y-) axis of the heatmap.

Usage

```
tof_plot_clusters_heatmap(
  tof_tibble,
  cluster_col,
  marker_cols = where(tof_is_numeric),
  central_tendency_function = stats::median,
  scale_markerwise = FALSE,
  scale_clusterwise = FALSE,
  cluster_markers = TRUE,
  cluster_clusters = TRUE,
  line_width = 0.25,
  theme = ggplot2::theme_minimal()
)
```

Arguments

tof_tibble	A ‘tof_tbl‘ or a ‘tibble‘.
cluster_col	An unquoted column name indicating which column in ‘tof_tibble‘ stores the cluster ids for the cluster to which each cell belongs. Cluster labels can be produced via any method the user chooses - including manual gating, any of the functions in the ‘tof_cluster_*‘ function family, or any other method.

`marker_cols` Unquoted column names indicating which column in 'tof_tibble' should be interpreted as markers to be plotted along the x-axis of the heatmap. Supports tidyselect helpers.

`central_tendency_function`
A function to use for computing the measure of central tendency that will be aggregated from each cluster in cluster_col. Defaults to the median.

`scale_markerwise`
A boolean value indicating if the heatmap should rescale the columns of the heatmap such that the maximum value for each marker is 1 and the minimum value is 0. Defaults to FALSE.

`scale_clusterwise`
A boolean value indicating if the heatmap should rescale the rows of the heatmap such that the maximum value for each cluster is 1 and the minimum value is 0. Defaults to FALSE.

`cluster_markers`
A boolean value indicating if the heatmap should order its columns (i.e. markers) using hierarchical clustering. Defaults to TRUE.

`cluster_clusters`
A boolean value indicating if the heatmap should order its rows (i.e. clusters) using hierarchical clustering. Defaults to TRUE.

`line_width` A numeric value indicating how thick the lines separating the tiles of the heatmap should be. Defaults to 0.25.

`theme` A ggplot2 theme to apply to the heatmap. Defaults to `theme_minimal`

Value

A ggplot object.

Examples

```
sim_data <-
    dplyr::tibble(
        cd45 = rnorm(n = 1000),
        cd38 = rnorm(n = 1000),
        cd34 = rnorm(n = 1000),
        cd19 = rnorm(n = 1000),
        cluster_id = sample(letters, size = 1000, replace = TRUE)
    )

heatmap <-
    tof_plot_clusters_heatmap(
        tof_tibble = sim_data,
        cluster_col = cluster_id
    )
```

`tof_plot_clusters_mst` *Visualize clusters in CyTOF data using a minimum spanning tree (MST).*

Description

This function plots a minimum-spanning tree using clustered single-cell data in order to summarize cluster-level characteristics. Each node in the MST represents a single cluster colored using a user-specified variable (either continuous or discrete).

Usage

```
tof_plot_clusters_mst(
  tof_tibble,
  cluster_col,
  knn_cols = where(tof_is_numeric),
  color_col,
  num_neighbors = 5L,
  graph_type = c("unweighted", "weighted"),
  graph_layout = "nicely",
  central_tendency_function = stats::median,
  distance_function = c("euclidean", "cosine"),
  edge_alpha = 0.4,
  node_size = "cluster_size",
  theme = ggplot2::theme_void(),
  ...
)
```

Arguments

`tof_tibble`	A ‘tof_tbl‘ or a ‘tibble‘.
`cluster_col`	An unquoted column name indicating which column in ‘tof_tibble‘ stores the cluster ids for the cluster to which each cell belongs. Cluster labels can be produced via any method the user chooses - including manual gating, any of the functions in the ‘tof_cluster_*‘ function family, or any other method.
`knn_cols`	Unquoted column names indicating which columns in ‘tof_tibble‘ should be used to compute the cluster-to-cluster distances used to construct the k-nearest-neighbor graph. Supports tidyselect helpers. Defaults to all numeric columns.
`color_col`	Unquoted column name indicating which column in ‘tof_tibble‘ should be used to color the nodes in the MST.
`num_neighbors`	An integer specifying how many neighbors should be used to construct the k-nearest neighbor graph.
`graph_type`	A string specifying if the k-nearest neighbor graph should be "weighted" (the default) or "unweighted".

graph_layout
: This argument specifies a layout for the MST in one of two ways. Option 1: Provide a string specifying which algorithm should be used to compute the force-directed layout. Passed to ggraph. Defaults to "nicely", which tries to automatically select a visually-appealing layout. Other examples include "fr", "gem", "kk", and many others. See layout_tbl_graph_igraph for other examples. Option 2: Provide a ggraph object previously generated with this function. The layout used to plot this ggraph object will then be used as a template for the new plot. Using this option, number of clusters (and their labels) must be identical to the template. This option is useful if you want to make multiple plots of the same tof_tibble colored by different protein markers, for example.

central_tendency_function
: A function to use for computing the measure of central tendency that will be aggregated from each cluster in cluster_col. Defaults to the median.

distance_function
: A string indicating which distance function to use in computing the cluster-to-clusters distances in constructing the MST. Valid options include "euclidean" (the default) and "cosine".

edge_alpha
: A numeric value between 0 and 1 specifying the transparency of the edges drawn in the force-directed layout. Defaults to 0.25.

node_size
: Either a numeric value specifying the size of the nodes in the MST or the string "cluster_size", in which case the size of the node representing each cluster will be scaled according to the number of cells in that cluster (the default).

theme
: A ggplot2 theme to apply to the force-directed layout. Defaults to theme_void

...
: Optional additional arguments to hnsw_knn

Value

A ggraph/ggplot object.

Examples

```
sim_data <-
    dplyr::tibble(
        cd45 = rnorm(n = 1000),
        cd38 = rnorm(n = 1000),
        cd34 = rnorm(n = 1000),
        cd19 = rnorm(n = 1000),
        cluster_id = sample(letters, size = 1000, replace = TRUE)
    )

# make a layout colored by a marker
layout_cd38 <-
    tof_plot_clusters_mst(
        tof_tibble = sim_data,
        cluster_col = cluster_id,
        color_col = cd38
    )

# use the same layout as the plot above to color the same
```

```
# tree using a different marker
layout_cd45 <-
    tof_plot_clusters_mst(
        tof_tibble = sim_data,
        cluster_col = cluster_id,
        color_col = cd45,
        graph_layout = layout_cd38
    )
```

`tof_plot_clusters_volcano` *Create a volcano plot from differential expression analysis results*

Description

This function makes a volcano plot using the results of a differential expression analysis (DEA) produced by one of the 'tof_dea_*' verbs. Each point in the volcano plot represents a single cluster-marker pair, colored by significance level and the direction of the marker expression difference.

Usage

```
tof_plot_clusters_volcano(
  dea_result,
  num_top_pairs = 10L,
  alpha = 0.05,
  point_size = 2,
  label_size = 3,
  nudge_x = 0,
  nudge_y = 0.25,
  increase_color = "#207394",
  decrease_color = "#cd5241",
  insignificant_color = "#cdcdcd",
  use_ggrepel = FALSE,
  theme = ggplot2::theme_bw()
)
```

Arguments

`dea_result`	A tibble containing the differential expression analysis (DEA) results produced by one of the members of the 'tof_dea_*' function family.
`num_top_pairs`	An integer representing the number of most significant cluster-marker pairs that should be labeled in the volcano plot.
`alpha`	A numeric value between 0 and 1 representing the significance level below which a p-value should be considered statistically significant. Defaults to 0.05.
`point_size`	A numeric value specifying the size of the points in the volcano plot.
`label_size`	A numeric value specifying the size of the text labeling cluster-marker pairs.

nudge_x	A numeric value specifying how far cluster-marker pair labels should be adjusted to the left (if 'nudge_x' is negative) or to the right (if 'nudge_x' is positive) to avoid overlap with the plotted points. Passed to geom_text, and ignored if 'use_ggrepel' = TRUE. Defaults to 0.
nudge_y	A numeric value specifying how far cluster-marker pair labels should be adjusted downwards (if 'nudge_y' is negative) or upwards (if 'nudge_y' is positive) to avoid overlap with the plotted points. Passed to geom_text, and ignored if 'use_ggrepel' = TRUE. Defaults to 0.25.
increase_color	A hex code specifying which fill color should be used for points corresponding to cluster-marker pairs where significant increases were detected.
decrease_color	A hex code specifying which fill color should be used for points corresponding to cluster-marker pairs where significant decreases were detected.
insignificant_color	A hex code specifying which fill color should be used for points corresponding to cluster-marker pairs where no significant differences were detected.
use_ggrepel	A boolean value indicting if geom_text_repel should be used to plot labels for cluster-marker pairs. Defaults to FALSE. If TRUE, the ggrepel package must be installed.
theme	A ggplot2 theme to apply to the volcano plot. Defaults to theme_bw

Value

A ggplot object.

Examples

```
# create a mock differential expression analysis result
sim_dea_result <-
    dplyr::tibble(
        cluster_id = rep(letters, 2),
        marker = rep(c("cd45", "cd34"), times = length(letters)),
        p_adj = runif(n = 2 * length(letters), min = 0, max = 0.5),
        mean_fc = runif(n = 2 * length(letters), min = 0.01, max = 10),
        significant = dplyr::if_else(p_adj < 0.05, "*", "")
    )

attr(sim_dea_result, which = "dea_method") <- "t_unpaired"

# create the volcano plot
volcano <- tof_plot_clusters_volcano(dea_result = sim_dea_result)
```

---

tof_plot_heatmap *Make a heatmap summarizing group marker expression patterns in high-dimensional cytometry data*

Description

This function makes a heatmap of group-to-group marker expression patterns in single-cell data. Markers are plotted along the horizontal (x-) axis of the heatmap and groups are plotted along the vertical (y-) axis of the heatmap.

Usage

```
tof_plot_heatmap(
  tof_tibble,
  y_col,
  marker_cols = where(tof_is_numeric),
  central_tendency_function = stats::median,
  scale_markerwise = FALSE,
  scale_ywise = FALSE,
  cluster_markers = TRUE,
  cluster_groups = TRUE,
  line_width = 0.25,
  theme = ggplot2::theme_minimal()
)
```

Arguments

tof_tibble	A ‘tof_tbl‘ or a ‘tibble‘.
y_col	An unquoted column name indicating which column in ‘tof_tibble‘ stores the ids for the group to which each cell belongs.
marker_cols	Unquoted column names indicating which column in ‘tof_tibble‘ should be interpreted as markers to be plotted along the x-axis of the heatmap. Supports tidyselect helpers.
central_tendency_function	A function to use for computing the measure of central tendency that will be aggregated from each cluster in cluster_col. Defaults to the median.
scale_markerwise	A boolean value indicating if the heatmap should rescale the columns of the heatmap such that the maximum value for each marker is 1 and the minimum value is 0. Defaults to FALSE.
scale_ywise	A boolean value indicating if the heatmap should rescale the rows of the heatmap such that the maximum value for each group is 1 and the minimum value is 0. Defaults to FALSE.
cluster_markers	A boolean value indicating if the heatmap should order its columns (i.e. markers) using hierarchical clustering. Defaults to TRUE.
cluster_groups	A boolean value indicating if the heatmap should order its rows (i.e. groups) using hierarchical clustering. Defaults to TRUE.
line_width	A numeric value indicating how thick the lines separating the tiles of the heatmap should be. Defaults to 0.25.
theme	A ggplot2 theme to apply to the heatmap. Defaults to theme_minimal

Value

A ggplot object.

`tof_plot_model` *Plot the results of a glmnet model fit on sample-level data.*

Description

Plot the results of a glmnet model fit on sample-level data.

Usage

```
tof_plot_model(tof_model, new_data, theme = ggplot2::theme_bw())
```

Arguments

`tof_model`	A 'tof_model' trained using `tof_train_model`
`new_data`	A tibble of new observations for which a plot should be made. If new_data isn't provided, the plot will be made using the training data used to fit the model. Alternatively, the string "tuning_data" can be provided, and the plot will be generated using the predictions generated during model tuning.
`theme`	A ggplot2 theme to apply to the plot Defaults to `theme_bw`

Value

A ggplot object. If the 'tof_model' is a linear model, a scatterplot of the predicted outcome vs. the true outcome will be returned. If the 'tof_model' is a two-class model, an ROC curve will be returned. If the 'tof_model' is a multiclass model, a one-versus-all ROC curve will be returned for each class. If 'tof_model' is a survival model, a Kaplan-Meier curve will be returned.

Examples

```
feature_tibble <-
    dplyr::tibble(
        sample = as.character(1:100),
        cd45 = runif(n = 100),
        pstat5 = runif(n = 100),
        cd34 = runif(n = 100),
        outcome = (3 * cd45) + (4 * pstat5) + rnorm(100),
        class =
            as.factor(
                dplyr::if_else(outcome > median(outcome), "class1", "class2")
            )
    )

new_tibble <-
    dplyr::tibble(
        sample = as.character(1:20),
```

```
        cd45 = runif(n = 20),
        pstat5 = runif(n = 20),
        cd34 = runif(n = 20),
        outcome = (3 * cd45) + (4 * pstat5) + rnorm(20),
        class =
            as.factor(
                dplyr::if_else(outcome > median(outcome), "class1", "class2")
            )
    )

split_data <- tof_split_data(feature_tibble, split_method = "simple")

# train a regression model
regression_model <-
    tof_train_model(
        split_data = split_data,
        predictor_cols = c(cd45, pstat5, cd34),
        response_col = outcome,
        model_type = "linear"
    )

# make the plot
plot_1 <- tof_plot_model(tof_model = regression_model, new_data = new_tibble)

# train a logistic regression classifier
logistic_model <-
    tof_train_model(
        split_data = split_data,
        predictor_cols = c(cd45, pstat5, cd34),
        response_col = class,
        model_type = "two-class"
    )

# make the plot

plot_2 <- tof_plot_model(tof_model = logistic_model, new_data = new_tibble)
```

`tof_plot_model_linear` *Plot the results of a linear glmnet model fit on sample-level data.*

Description

Plot the results of a linear glmnet model fit on sample-level data.

Usage

```
tof_plot_model_linear(tof_model, new_data, theme = ggplot2::theme_bw())
```

Arguments

tof_model	A 'tof_model' trained using tof_train_model
new_data	A tibble of new observations for which a plot should be made. If new_data isn't provided, the plot will be made using the training data used to fit the model. Alternatively, the string "tuning_data" can be provided, and the plot will be generated using the predictions generated during model tuning.
theme	A ggplot2 theme to apply to the plot Defaults to theme_bw

Value

A ggplot object. Specifically, a scatterplot of the predicted outcome vs. the true outcome will be returned.

---

tof_plot_model_logistic

Plot the results of a two-class glmnet model fit on sample-level data.

Description

Plot the results of a two-class glmnet model fit on sample-level data.

Usage

```
tof_plot_model_logistic(tof_model, new_data, theme = ggplot2::theme_bw())
```

Arguments

tof_model	A 'tof_model' trained using tof_train_model
new_data	A tibble of new observations for which a plot should be made. If new_data isn't provided, the plot will be made using the training data used to fit the model. Alternatively, the string "tuning_data" can be provided, and the plot will be generated using the predictions generated during model tuning.
theme	A ggplot2 theme to apply to the plot. Defaults to theme_bw

Value

A ggplot object. Specifically, an ROC curve..

tof_plot_model_multinomial

Plot the results of a multiclass glmnet model fit on sample-level data.

Description

Plot the results of a multiclass glmnet model fit on sample-level data.

Usage

```
tof_plot_model_multinomial(tof_model, new_data, theme = ggplot2::theme_bw())
```

Arguments

tof_model	A 'tof_model' trained using tof_train_model
new_data	A tibble of new observations for which a plot should be made. If new_data isn't provided, the plot will be made using the training data used to fit the model. Alternatively, the string "tuning_data" can be provided, and the plot will be generated using the predictions generated during model tuning.
theme	A ggplot2 theme to apply to the plot. Defaults to theme_bw.

Value

A ggplot object. Specifically, a one-versus-all ROC curve (one for each class).

tof_plot_model_survival

Plot the results of a survival glmnet model fit on sample-level data.

Description

Plot the results of a survival glmnet model fit on sample-level data.

Usage

```
tof_plot_model_survival(
  tof_model,
  new_data,
  censor_size = 2.5,
  theme = ggplot2::theme_bw()
)
```

Arguments

tof_model	A ‘tof_model‘ trained using tof_train_model
new_data	A tibble of new observations for which a plot should be made. If new_data isn’t provided, the plot will be made using the training data used to fit the model. Alternatively, the string "tuning_data" can be provided, and the plot will be generated using the predictions generated during model tuning.
censor_size	A numeric value indicating how large to plot the tick marks representing censored values in the Kaplan-Meier curve.
theme	A ggplot2 theme to apply to the plot. Defaults to theme_bw

Value

A ggplot object. Specifically, a Kaplan-Meier curve.

---

tof_plot_sample_features

Make a heatmap summarizing sample marker expression patterns in CyTOF data

---

Description

This function makes a heatmap of sample-to-sample marker expression patterns in single-cell data. Markers are plotted along the horizontal (x-) axis of the heatmap and sample IDs are plotted along the vertical (y-) axis of the heatmap.

Usage

```
tof_plot_sample_features(
  feature_tibble,
  sample_col,
  feature_cols = where(tof_is_numeric),
  scale_featurewise = FALSE,
  scale_samplewise = FALSE,
  line_width = 0.25,
  theme = ggplot2::theme_minimal()
)
```

Arguments

feature_tibble	A tbl_df or data.frame of aggregated sample-level features, such as that generated by tof_extract_features.
sample_col	An unquoted column name indicating which column in ‘tof_tibble‘ stores the IDs for each sample. If no sample IDs are present, a numeric ID will be assigned to each row of ‘feature_tibble‘ based on its row index.

feature_cols — Unquoted column names indicating which column in 'feature_tibble' should be interpreted as features to be plotted along the x-axis of the heatmap. Supports tidyselect helpers.

scale_featurewise — A boolean value indicating if the heatmap should rescale the columns of the heatmap such that the maximum value for each marker is 1 and the minimum value is 0. Defaults to FALSE.

scale_samplewise — A boolean value indicating if the heatmap should rescale the rows of the heatmap such that the maximum value for each sample is 1 and the minimum value is 0. Defaults to FALSE.

line_width — A numeric value indicating how thick the lines separating the tiles of the heatmap should be. Defaults to 0.25.

theme — A ggplot2 theme to apply to the heatmap. Defaults to theme_minimal

Value

A ggplot object.

Examples

```
# simulate single-cell data
sim_data <-
    dplyr::tibble(
        cd45 = rnorm(n = 1000),
        cd38 = rnorm(n = 1000),
        cd34 = rnorm(n = 1000),
        cd19 = rnorm(n = 1000),
        cluster_id = sample(letters, size = 1000, replace = TRUE),
        sample_id = sample(paste0("sample", 1:5), size = 1000, replace = TRUE)
    )

# extract cluster proportions in each simulated patient
feature_data <-
    tof_extract_proportion(
        tof_tibble = sim_data,
        cluster_col = cluster_id,
        group_cols = sample_id
    )

# plot the heatmap
heatmap <- tof_plot_sample_features(feature_tibble = feature_data)
```

---

tof_plot_sample_heatmap

Make a heatmap summarizing sample marker expression patterns in CyTOF data

---

Description

This function makes a heatmap of sample-to-sample marker expression patterns in single-cell data. Markers are plotted along the horizontal (x-) axis of the heatmap and sample IDs are plotted along the vertical (y-) axis of the heatmap.

Usage

```
tof_plot_sample_heatmap(
  tof_tibble,
  sample_col,
  marker_cols = where(tof_is_numeric),
  central_tendency_function = stats::median,
  scale_markerwise = FALSE,
  scale_samplewise = FALSE,
  line_width = 0.25,
  theme = ggplot2::theme_minimal()
)
```

Arguments

tof_tibble	A ‘tof_tbl‘ or a ‘tibble‘.
sample_col	An unquoted column name indicating which column in ‘tof_tibble‘ stores the ids for the sample to which each cell belongs.
marker_cols	Unquoted column names indicating which column in ‘tof_tibble‘ should be interpreted as markers to be plotted along the x-axis of the heatmap. Supports tidyselect helpers.
central_tendency_function	A function to use for computing the measure of central tendency that will be aggregated from each sample in cluster_col. Defaults to the median.
scale_markerwise	A boolean value indicating if the heatmap should rescale the columns of the heatmap such that the maximum value for each marker is 1 and the minimum value is 0. Defaults to FALSE.
scale_samplewise	A boolean value indicating if the heatmap should rescale the rows of the heatmap such that the maximum value for each sample is 1 and the minimum value is 0. Defaults to FALSE.
line_width	A numeric value indicating how thick the lines separating the tiles of the heatmap should be. Defaults to 0.25.
theme	A ggplot2 theme to apply to the heatmap. Defaults to theme_minimal

Value

A ggplot object.

Examples

```
sim_data <-
    dplyr::tibble(
        cd45 = rnorm(n = 1000),
        cd38 = rnorm(n = 1000),
        cd34 = rnorm(n = 1000),
        cd19 = rnorm(n = 1000),
        sample_id = sample(paste0("sample", 1:5), size = 1000, replace = TRUE)
    )

heatmap <-
    tof_plot_sample_heatmap(
        tof_tibble = sim_data,
        sample_col = sample_id
    )
```

`tof_postprocess` *Post-process transformed CyTOF data.*

Description

This function transforms a ‘tof_tibble‘ of transformed ion counts from a mass cytometer back into something that looks more like an .fcs file that Fluidigm software generates.

Usage

```
tof_postprocess(
  tof_tibble = NULL,
  channel_cols = where(tof_is_numeric),
  redo_noise = FALSE,
  transform_fun = function(x) rev_asinh(x, shift_factor = 0, scale_factor = 0.2)
)
```

Arguments

`tof_tibble`	A ‘tof_tibble‘ or a ‘tibble‘.
`channel_cols`	A vector of non-quoted column names indicating which columns in ‘tof_tibble‘ contain protein measurements. Supports tidyselect helpers. If nothing is specified, the default is to transform all numeric columns.
`redo_noise`	A boolean value indicating whether to add uniform noise that to each CyTOF measurement for aesthetic and visualization purposes. See this paper. Defaults to FALSE
`transform_fun`	A vectorized function to apply to each column specified by ‘channel_cols‘ for post-processing. Defaults to `rev_asinh` transformation (with a cofactor of 5).

Value

A ‘tof_tbl‘ with identical dimensions to the input ‘tof_tibble‘, with all columns specified in channel_cols transformed using ‘transform_fun‘ (with noise added or not removed depending on ‘redo_noise‘).

See Also

[tof_preprocess()]

Examples

```
# read in an example .fcs file from tidytof's internal datasets
input_file <- dir(tidytof_example_data("aml"), full.names = TRUE)[[1]]
tof_tibble <- tof_read_data(input_file)

# preprocess all numeric columns with default behavior
# arcsinh transformation with a cofactor of 5
preprocessed_tof_tibble <- tof_preprocess(tof_tibble)

# postprocess all numeric columns to reverse the preprocessing
tof_postprocess(tof_tibble)
```

---

tof_predict *Use a trained elastic net model to predict fitted values from new data*

Description

This function uses a trained ‘tof_model‘ to make predictions on new data.

Usage

```
tof_predict(
  tof_model,
  new_data,
  prediction_type = c("response", "class", "link", "survival curve")
)
```

Arguments

tof_model	A ‘tof_model‘ trained using tof_train_model
new_data	A tibble of new observations for which predictions should be made. If new_data isn't provided, predictions will be made for the training data used to fit the model.
prediction_type	A string indicating which type of prediction should be provided by the model:

"response" (the default) For "linear" models, the predicted response for each observation. For "two-class" and "multiclass" models, the fitted probabilities of each class for each observation. For "survival" models, the fitted relative-risk for each observation.

"class" Only applies to "two-class" and "multiclass" models. For both, the class label corresponding to the class with the maximum fitted probability.

"link" The linear predictions of the model (the output of the link function for each model family.)

"survival curve" Only applies to "survival" models. Returns a tibble indicating each patient's probability of survival (1 - probability(event)) at each timepoint in the dataset. Obtained using the `survfit` function.

Value

A `tibble` with a single column (‘.pred‘) containing the predictions or, for multiclass models with ‘prediction_type‘ == "response", a tibble with one column for each class. Each row in the output corresponds to a row in ‘new_data‘ (or, if ‘new_data‘ is not provided, to a row in the ‘tof_model‘’s training data). In the latter case, be sure to check ‘tof_model$training_data‘ to confirm the order of observations, as the resampling procedure can change their ordering relative to the original input data.

See Also

Other modeling functions: `tof_assess_model()`, `tof_create_grid()`, `tof_split_data()`, `tof_train_model()`

Examples

```
feature_tibble <-
    dplyr::tibble(
        sample = as.character(1:100),
        cd45 = runif(n = 100),
        pstat5 = runif(n = 100),
        cd34 = runif(n = 100),
        outcome = (3 * cd45) + (4 * pstat5) + rnorm(100)
    )

new_tibble <-
    dplyr::tibble(
        sample = as.character(1:20),
        cd45 = runif(n = 20),
        pstat5 = runif(n = 20),
        cd34 = runif(n = 20),
        outcome = (3 * cd45) + (4 * pstat5) + rnorm(20)
    )

split_data <- tof_split_data(feature_tibble, split_method = "simple")

# train a regression model
regression_model <-
    tof_train_model(
        split_data = split_data,
```

```
        predictor_cols = c(cd45, pstat5, cd34),
        response_col = outcome,
        model_type = "linear"
    )

# apply the model to new data
tof_predict(tof_model = regression_model, new_data = new_tibble)
```

`tof_preprocess` *Preprocess raw high-dimensional cytometry data.*

Description

This function transforms a ‘tof_tbl‘ of raw ion counts, reads, or fluorescence intensity units directly measured on a cytometer using a user-provided function. It can be used to perform standard preprocessing steps (i.e. arcsinh transformation) before cytometry data analysis.

Usage

```
tof_preprocess(
  tof_tibble = NULL,
  channel_cols = where(tof_is_numeric),
  undo_noise = FALSE,
  transform_fun = function(x) asinh(x/5)
)
```

Arguments

`tof_tibble`	A ‘tof_tbl‘ or a ‘tibble‘.
`channel_cols`	Unquoted column names representing columns that contain single-cell protein measurements. Supports tidyselect helpers. If nothing is specified, the default is to transform all numeric columns.
`undo_noise`	A boolean value indicating whether to remove the uniform noise that Fluidigm software adds to CyTOF measurements for aesthetic and visualization purposes. See this paper. Defaults to FALSE.
`transform_fun`	A vectorized function to apply to each protein value for variance stabilization. Defaults to `asinh` transformation (with a co-factor of 5).

Value

A ‘tof_tbl‘ with identical dimensions to the input ‘tof_tibble‘, with all columns specified in channel_cols transformed using ‘transform_fun‘ (with noise removed or not removed depending on ‘undo_noise‘).

See Also

[tof_postprocess()]

Examples

```
# read in an example .fcs file from tidytof's internal datasets
input_file <- dir(tidytof_example_data("aml"), full.names = TRUE)[[1]]
tof_tibble <- tof_read_data(input_file)

# preprocess all numeric columns with default behavior
# arcsinh transformation with a cofactor of 5
tof_preprocess(tof_tibble)

# preprocess all numeric columns using the log base 10 tranformation
tof_preprocess(tof_tibble, transform_fun = log10)
```

`tof_prep_recipe` *Train a recipe or list of recipes for preprocessing sample-level cytometry data*

Description

Train a recipe or list of recipes for preprocessing sample-level cytometry data

Usage

```
tof_prep_recipe(split_data, unprepped_recipe)
```

Arguments

`split_data`	An ‘rsplit‘ or ‘rset‘ object from the `rsample` package containing the sample-level data to use for modeling. The easiest way to generate this is to use `tof_split_data`. Alternatively, an unsplit tbl_df, though this is not recommended.
`unprepped_recipe`	A `recipe` object (if ‘split_data‘ is an ‘rsplit‘ object or a ‘tbl_df‘) or list of recipes (if ‘split_data‘ is an ‘rset‘ object).

Value

If split_data is an "rsplit" or "tbl_df" object, will return a single prepped recipe. If split_data is an "rset" object, will return a list of prepped recipes specific for each fold of the resampling procedure.

`tof_read_csv` *Read high-dimensional cytometry data from a .csv file into a tidy tibble.*

Description

Read high-dimensional cytometry data from a .csv file into a tidy tibble.

Usage

```
tof_read_csv(file_path = NULL, panel_info = dplyr::tibble())
```

Arguments

`file_path`	A file path to a single .csv file.
`panel_info`	Optional. A tibble or data.frame containing information about the panel used during high-dimensional cytometry data acquisition. Two columns are required: "metals" and "antigens".

Value

A ‘tof_tbl‘ in which each row represents a single cell and each column represents a high-dimensional cytometry antigen channel.

A ‘tof_tbl‘ is an S3 class that extends the "tibble" class by storing one additional attribute: "panel" (a tibble storing information about the panel used during data acquisition). Because panel information isn’t obvious from data read as a .csv file, this information must be provided manually from the user (unlike in ‘tof_read_fcs‘).

`tof_read_data` *Read data from an .fcs/.csv file or a directory of .fcs/.csv files.*

Description

Read data from an .fcs/.csv file or a directory of .fcs/.csv files.

Usage

```
tof_read_data(path = NULL, sep = "|", panel_info = dplyr::tibble())
```

Arguments

path	A file path to a single file or to a directory of files. The only valid file types are .fcs files or .csv files containing high-dimensional cytometry data.
sep	Optional. A string to use to separate the antigen name and its associated metal in the column names of the output tibble. Defaults to "|". Only used if the input file is an .fcs file.
panel_info	Optional. A tibble or data.frame containing information about the panel used during high-dimensional cytometry data acquisition. Two columns are required: "metals" and "antigens". Only used if the input file is a .csv file.

Value

An [c by m+1] tibble in which each row represents a single cell (of c total in the dataset) and each column represents a high-dimensional cytometry measurement (of m total in the dataset). If more than one .fcs is read at once, the last column of the tibble (‘file_name‘) will represent the file name of the .fcs file from which each cell was read.

See Also

Other input/output functions: tof_write_csv(), tof_write_data(), tof_write_fcs()

Examples

```
input_file <- dir(tidytof_example_data("aml"), full.names = TRUE)[[1]]
tof_read_data(input_file)
```

---

tof_read_fcs *Read high-dimensional cytometry data from an .fcs file into a tidy tibble.*

Description

This function reads high-dimensional cytometry data from a single .fcs file into a tidy data structure called a ‘tof_tbl‘ ("tof_tibble"). tof_tibbles are identical to normal tibbles except for an additional attribute ("panel") that stores information about the high-dimensional cytometry panel used during data acquisition.

Usage

```
tof_read_fcs(file_path = NULL, sep = "|")
```

Arguments

file_path	A file path to a single .fcs file.
sep	A string to use to separate the antigen name and its associated metal in the column names of the output tibble. Defaults to "|".

Value

a ‘tof_tbl‘ in which each row represents a single cell and each column represents a high-dimensional cytometry antigen channel.

A ‘tof_tbl‘ is an S3 class that extends the "tibble" class by storing one additional attribute: "panel" (a tibble storing information about the panel used during data acquisition).

`tof_read_file` *Read high-dimensional cytometry data from a single .fcs or .csv file into a tidy tibble.*

Description

Read high-dimensional cytometry data from a single .fcs or .csv file into a tidy tibble.

Usage

```
tof_read_file(file_path = NULL, sep = "|", panel_info = dplyr::tibble())
```

Arguments

`file_path`	A file path to a single .fcs or .csv file.
`sep`	A string to use to separate the antigen name and its associated metal in the column names of the output tibble. Defaults to "|". Only used if the input file is an .fcs file.
`panel_info`	Optional. A tibble or data.frame containing information about the panel used during high-dimensional cytometry data acquisition. Two columns are required: "metals" and "antigens". Only used if the input file is a .csv file.

Value

A ‘tof_tbl‘ in which each row represents a single cell and each column represents a high-dimensional cytometry antigen channel.

A ‘tof_tbl‘ is an S3 class that extends the "tibble" class by storing one additional attribute: "panel" (a tibble storing information about the panel used during data acquisition). Because panel information isn’t obvious from data read as a .csv file, this information must be provided manually by the user.

`tof_reduce_dimensions` *Apply dimensionality reduction to a single-cell dataset.*

Description

This function is a wrapper around tidytof's tof_reduce_* function family. It performs dimensionality reduction on single-cell data using a user-specified method (of 3 choices) and each method's corresponding input parameters

Usage

```
tof_reduce_dimensions(
  tof_tibble,
  ...,
  augment = TRUE,
  method = c("pca", "tsne", "umap")
)
```

Arguments

`tof_tibble`	A 'tof_tbl' or 'tibble'.
`...`	Arguments to be passed to the tof_reduce_* function corresponding to the embedding method. See `tof_reduce_pca`, `tof_reduce_tsne`, and `tof_reduce_umap`.
`augment`	A boolean value indicating if the output should column-bind the dimensionality-reduced embedding vectors of each cell as a new column in 'tof_tibble' (TRUE, the default) or if a tibble including only the low-dimensionality embeddings should be returned (FALSE).
`method`	A method of dimensionality reduction. Currently, PCA, tSNE, and UMAP embedding are supported.

Value

A tibble with the same number of rows as 'tof_tibble', each representing a single cell. Each of the 'num_comp' columns represents each cell's embedding in the calculated embedding space.

See Also

Other dimensionality reduction functions: `tof_reduce_pca()`, `tof_reduce_tsne()`, `tof_reduce_umap()`

Examples

```
# simulate single-cell data
sim_data <-
    dplyr::tibble(
        cd45 = rnorm(n = 100),
        cd38 = rnorm(n = 100),
        cd34 = rnorm(n = 100),
```

```
        cd19 = rnorm(n = 100)
    )

# calculate pca
tof_reduce_dimensions(tof_tibble = sim_data, method = "pca")

# calculate tsne
tof_reduce_dimensions(tof_tibble = sim_data, method = "tsne")

# calculate umap
tof_reduce_dimensions(tof_tibble = sim_data, method = "umap")
```

`tof_reduce_pca` *Perform principal component analysis on single-cell data*

Description

This function calculates principal components using single-cell data from a ‘tof_tibble‘.

Usage

```
tof_reduce_pca(
  tof_tibble,
  pca_cols = where(tof_is_numeric),
  num_comp = 5,
  threshold = NA,
  center = TRUE,
  scale = TRUE,
  return_recipe = FALSE
)
```

Arguments

`tof_tibble`	A ‘tof_tbl‘ or ‘tibble‘.
`pca_cols`	Unquoted column names indicating which columns in ‘tof_tibble‘ to use for computing the principal components. Defaults to all numeric columns. Supports tidyselect helpers.
`num_comp`	The number of PCA components to calculate. Defaults to 5. See `step_pca`.
`threshold`	A double between 0 and 1 representing the fraction of total variance that should be covered by the components returned in the output. See `step_pca`.
`center`	A boolean value indicating if each column should be centered to mean 0 before PCA analysis. Defaults to TRUE.
`scale`	A boolean value indicating if each column should be scaled to standard deviation = 1 before PCA analysis. Defaults to TRUE.

return_recipe A boolean value indicating if instead of the UMAP result, a prepped recipe object containing the PCA embedding should be returned. Set this option to TRUE if you want to create the PCA embedding using one dataset but also want to project new observations onto the same embedding space later.

Value

A tibble with the same number of rows as 'tof_tibble', each representing a single cell. Each of the 'num_comp' columns represents each cell's embedding in the calculated principal component space.

See Also

Other dimensionality reduction functions: tof_reduce_dimensions(), tof_reduce_tsne(), tof_reduce_umap()

Examples

```
# simulate single-cell data
sim_data <-
    dplyr::tibble(
        cd45 = rnorm(n = 200),
        cd38 = rnorm(n = 200),
        cd34 = rnorm(n = 200),
        cd19 = rnorm(n = 200)
    )
new_data <-
    dplyr::tibble(
        cd45 = rnorm(n = 50),
        cd38 = rnorm(n = 50),
        cd34 = rnorm(n = 50),
        cd19 = rnorm(n = 50)
    )

# calculate pca
tof_reduce_pca(tof_tibble = sim_data, num_comp = 2)

# return recipe instead of embeddings
pca_recipe <- tof_reduce_pca(tof_tibble = sim_data, return_recipe = TRUE)

# apply recipe to new data
recipes::bake(pca_recipe, new_data = new_data)
```

---

tof_reduce_tsne *Perform t-distributed stochastic neighborhood embedding on single-cell data*

Description

This function calculates a tSNE embedding using single-cell data from a 'tof_tibble'.

Usage

```
tof_reduce_tsne(
  tof_tibble,
  tsne_cols = where(tof_is_numeric),
  num_comp = 2,
  perplexity = 30,
  theta = 0.5,
  max_iterations = 1000,
  verbose = FALSE,
  ...
)
```

Arguments

tof_tibble	A 'tof_tbl' or 'tibble'.
tsne_cols	Unquoted column names indicating which columns in 'tof_tibble' to use in computing the tSNE embedding. Defaults to all numeric columns in 'tof_tibble'. Supports tidyselect helpers.
num_comp	The number of tSNE components to calculate for the embedding. Defaults to 2.
perplexity	A positive numeric value that represents represents the rough balance between the input data's local and global structure emphasized in the embedding. Smaller values emphasize local structure; larger values emphasize global structure. The recommended range is generally 5-50. Defaults to 30.
theta	A numeric value representing the speed/accuracy tradeoff for the embedding. Set to 0 for the exact tSNE; increase for a faster approximation. Defaults to 0.5
max_iterations	An integer number of iterations to use during embedding calculation. Defaults to 1000.
verbose	A boolean value indicating whether progress updates should be printed during embedding calculation. Default is FALSE.
...	Additional arguments to pass to Rtsne.

Value

A tibble with the same number of rows as 'tof_tibble', each representing a single cell. Each of the 'num_comp' columns represents each cell's embedding in the calculated tSNE space.

See Also

Other dimensionality reduction functions: tof_reduce_dimensions(), tof_reduce_pca(), tof_reduce_umap()

Examples

```
# simulate single-cell data
sim_data <-
    dplyr::tibble(
        cd45 = rnorm(n = 200),
        cd38 = rnorm(n = 200),
```

```
        cd34 = rnorm(n = 200),
        cd19 = rnorm(n = 200)
    )

# calculate tsne
tof_reduce_tsne(tof_tibble = sim_data)

# calculate tsne with only 2 columns
tof_reduce_tsne(tof_tibble = sim_data, tsne_cols = c(cd34, cd38))
```

`tof_reduce_umap` *Apply uniform manifold approximation and projection (UMAP) to single-cell data*

Description

This function calculates a UMAP embedding from single-cell data in a ‘tof_tibble‘.

Usage

```
tof_reduce_umap(
  tof_tibble,
  umap_cols = where(tof_is_numeric),
  num_comp = 2,
  neighbors = 5,
  min_dist = 0.01,
  learn_rate = 1,
  epochs = NULL,
  verbose = FALSE,
  n_threads = 1,
  return_recipe = FALSE,
  ...
)
```

Arguments

`tof_tibble`	A ‘tof_tbl‘ or ‘tibble‘.
`umap_cols`	Unquoted column names indicating which columns in ‘tof_tibble‘ to use in computing the UMAP embedding. Defaults to all numeric columns in ‘tof_tibble‘. Supports tidyselect helpers.
`num_comp`	An integer for the number of UMAP components.
`neighbors`	An integer for the number of nearest neighbors used to construct the target simplicial set.
`min_dist`	The effective minimum distance between embedded points.
`learn_rate`	Positive number of the learning rate for the optimization process.

epochs	Number of iterations for the neighbor optimization. See umap for details.
verbose	A boolean indicating if run details should be logged to the console. Defaults to FALSE.
n_threads	Number of threads to use during UMAP calculation. Defaults to 1.
return_recipe	A boolean value indicating if instead of the UMAP result, a prepped recipe object containing the UMAP embedding should be returned. Set this option to TRUE if you want to create the UMAP embedding using one dataset but also want to project new observations onto the same embedding space later.
...	Optional. Other options to be passed as arguments to umap.

Value

A tibble with the same number of rows as 'tof_tibble', each representing a single cell. Each of the 'num_comp' columns represents each cell's embedding in the calculated UMAP space.

See Also

Other dimensionality reduction functions: tof_reduce_dimensions(), tof_reduce_pca(), tof_reduce_tsne()

Examples

```
# simulate single-cell data
sim_data <-
    dplyr::tibble(
        cd45 = rnorm(n = 200),
        cd38 = rnorm(n = 200),
        cd34 = rnorm(n = 200),
        cd19 = rnorm(n = 200)
    )
new_data <-
    dplyr::tibble(
        cd45 = rnorm(n = 50),
        cd38 = rnorm(n = 50),
        cd34 = rnorm(n = 50),
        cd19 = rnorm(n = 50)
    )

# calculate umap
tof_reduce_umap(tof_tibble = sim_data)

# calculate umap with only 2 columns
tof_reduce_tsne(tof_tibble = sim_data, umap_cols = c(cd34, cd38))

# return recipe
umap_recipe <- tof_reduce_umap(tof_tibble = sim_data, return_recipe = TRUE)

# apply recipe to new data
recipes::bake(umap_recipe, new_data = new_data)
```

tof_set_panel *Set panel information from a tof_tibble*

Description

Set panel information from a tof_tibble

Usage

```
tof_set_panel(tof_tibble, panel)
```

Arguments

tof_tibble	A ‘tof_tbl‘.
panel	A tibble containing two columns (‘metals‘ and ‘antigens‘) representing the information about a panel

Value

A ‘tof_tibble‘ containing information about the CyTOF panel that was used during data acquisition for the data contained in the input ‘tof_tibble‘. Two columns are required: "metals" and "antigens".

See Also

Other tof_tbl utilities: new_tof_tibble(), tof_get_panel()

Examples

```
# get current panel from an .fcs file
input_file <- dir(tidytof_example_data("aml"), full.names = TRUE)[[1]]
tof_tibble <- tof_read_data(input_file)
current_panel <- tof_get_panel(tof_tibble)

# create a new panel (remove empty channels)
new_panel <- dplyr::filter(current_panel, antigens != "empty")
tof_set_panel(tof_tibble = tof_tibble, panel = new_panel)
```

`tof_spade_density` *Estimate cells' local densities as done in Spanning-tree Progression Analysis of Density-normalized Events (SPADE)*

Description

This function uses the algorithm described in Qiu et al., (2011) to estimate the local density of each cell in a 'tof_tbl' or 'tibble' containing high-dimensional cytometry data. Briefly, this algorithm involves counting the number of neighboring cells within a sphere of radius alpha surrounding each cell. Here, we do so using the `nn2` function.

Usage

```
tof_spade_density(
  tof_tibble,
  distance_cols = where(tof_is_numeric),
  distance_function = c("euclidean", "cosine", "l2", "ip"),
  num_alpha_cells = 2000L,
  alpha_multiplier = 5,
  max_neighbors = round(0.01 * nrow(tof_tibble)),
  normalize = TRUE,
  ...
)
```

Arguments

`tof_tibble`	A 'tof_tbl' or a 'tibble'.
`distance_cols`	Unquoted names of the columns in 'tof_tibble' to use in calculating cell-to-cell distances during the local density estimation for each cell. Defaults to all numeric columns in 'tof_tibble'.
`distance_function`	A string indicating which distance function to use for calculating cell-to-cell distances during local density estimation. Options include "euclidean" (the default) and "cosine".
`num_alpha_cells`	An integer indicating how many cells from 'tof_tibble' should be randomly sampled from 'tof_tibble' in order to estimate 'alpha', the radius of the sphere constructed around each cell during local density estimation. Alpha is calculated by taking the median nearest-neighbor distance from the 'num_alpha_cells' randomly-sampled cells and multiplying it by 'alpha_multiplier'. Defaults to 2000.
`alpha_multiplier`	An numeric value indicating the multiplier that should be used when calculating 'alpha', the radius of the sphere constructed around each cell during local density estimation. Alpha is calculated by taking the median nearest-neighbor distance from the 'num_alpha_cells' cells randomly-sampled from 'tof_tibble' and multiplying it by 'alpha_multiplier'. Defaults to 5.

max_neighbors	An integer indicating the maximum number of neighbors that can be counted within the sphere surrounding any given cell. Implemented to reduce the density estimation procedure's speed and memory requirements. Defaults to 1% of the number of rows in 'tof_tibble'.
normalize	A boolean value indicating if the vector of local density estimates should be normalized to values between 0 and 1. Defaults to TRUE.
...	Additional optional arguments to pass to tof_find_knn.

Value

A tibble with a single column named ".spade_density" containing the local density estimates for each input cell in 'tof_tibble'.

See Also

Other local density estimation functions: tof_estimate_density(), tof_knn_density()

Examples

```
sim_data <-
    dplyr::tibble(
        cd45 = rnorm(n = 1000),
        cd38 = rnorm(n = 1000),
        cd34 = rnorm(n = 1000),
        cd19 = rnorm(n = 1000)
    )

# perform the density estimation
tof_spade_density(tof_tibble = sim_data)

# perform the density estimation using cosine distance
tof_spade_density(
    tof_tibble = sim_data,
    distance_function = "cosine",
    alpha_multiplier = 2
)

# perform the density estimation with a smaller search radius around
# each cell
tof_spade_density(
    tof_tibble = sim_data,
    alpha_multiplier = 2
)
```

tof_split_data *Split high-dimensional cytometry data into a training and test set*

Description

Split high-dimensional cytometry data into a training and test set

Usage

```
tof_split_data(
  feature_tibble,
  split_method = c("k-fold", "bootstrap", "simple"),
  split_col,
  simple_prop = 3/4,
  num_cv_folds = 10,
  num_cv_repeats = 1L,
  num_bootstraps = 10,
  strata = NULL,
  ...
)
```

Arguments

feature_tibble	A tibble in which each row represents a sample- or patient- level observation, such as those produced by tof_extract_features.
split_method	Either a string or a logical vector specifying how to perform the split. If a string, valid options include k-fold cross validation ("k-fold"; the default), bootstrapping ("bootstrap"), or a single binary split ("simple"). If a logical vector, it should contain one entry for each row in ‘feature_tibble‘ indicating if that row should be included in the training set (TRUE) or excluded for the validation/test set (FALSE). Ignored entirely if ‘split_col‘ is specified.
split_col	The unquoted column name of the logical column in ‘feature_tibble‘ indicating if each row should be included in the training set (TRUE) or excluded for the validation/test set (FALSE).
simple_prop	A numeric value between 0 and 1 indicating what proportion of the data should be used for training. Defaults to 3/4. Ignored if split_method is not "simple".
num_cv_folds	An integer indicating how many cross-validation folds should be used. Defaults to 10. Ignored if split_method is not "k-fold".
num_cv_repeats	An integer indicating how many independent cross-validation replicates should be used (i.e. how many num_cv_fold splits should be performed). Defaults to 1. Ignored if split_method is not "k-fold".
num_bootstraps	An integer indicating how many independent bootstrap replicates should be used. Defaults to 25. Ignored if split_method is not "bootstrap".
strata	An unquoted column name representing the column in feature_tibble that should be used to stratify the data splitting. Defaults to NULL (no stratification).

... Optional additional arguments to pass to vfold_cv for k-fold cross validation, bootstraps for bootstrapping, or initial_split for simple splitting.

Value

If for k-fold cross validation and bootstrapping, an "rset" object; for simple splitting, an "rsplit" object. For details, see rsample.

See Also

Other modeling functions: tof_assess_model(), tof_create_grid(), tof_predict(), tof_train_model()

Examples

```
feature_tibble <-
    dplyr::tibble(
        sample = as.character(1:100),
        cd45 = runif(n = 100),
        pstat5 = runif(n = 100),
        cd34 = runif(n = 100),
        outcome = (3 * cd45) + (4 * pstat5) + rnorm(100),
        class =
            as.factor(
                dplyr::if_else(outcome > median(outcome), "class1", "class2")
            ),
        multiclass =
            as.factor(
                c(rep("class1", 30), rep("class2", 30), rep("class3", 40))
            ),
        event = c(rep(0, times = 50), rep(1, times = 50)),
        time_to_event = rnorm(n = 100, mean = 10, sd = 2)
    )

# split the dataset into 10 CV folds
tof_split_data(
    feature_tibble = feature_tibble,
    split_method = "k-fold"
)

# split the dataset into 10 bootstrap resamplings
tof_split_data(
    feature_tibble = feature_tibble,
    split_method = "bootstrap"
)

# split the dataset into a single training/test set
# stratified by the "class" column
tof_split_data(
    feature_tibble = feature_tibble,
    split_method = "simple",
    strata = class
)
```

tof_split_tidytof_reduced_dimensions

Split the dimensionality reduction data that tidytof combines during `SingleCellExperiment` *conversion*

Description

Split the dimensionality reduction data that tidytof combines during `SingleCellExperiment` conversion

Usage

```
tof_split_tidytof_reduced_dimensions(sce)
```

Arguments

`sce`	A `SingleCellExperiment` with an entry named "tidytof_reduced_dimensions" in its `reducedDims` slot.

Value

A `SingleCellExperiment` with separate entries named "tidytof_pca", "tidytof_umap", and "tidytof_tsne" in its `reducedDims` slots (one for each of the dimensionality reduction methods for which tidytof has native support).

Examples

```
NULL
```

tof_train_model

Train an elastic net model to predict sample-level phenomena using high-dimensional cytometry data.

Description

This function uses a training set/test set paradigm to tune and fit an elastic net model using a variety of user-specified details. Tuning can be performed using either a simple training vs. test set split, k-fold cross-validation, or bootstrapping, and multiple preprocessing options are available.

Usage

```
tof_train_model(
  split_data,
  unsplit_data,
  predictor_cols,
  response_col = NULL,
  time_col = NULL,
  event_col = NULL,
  model_type = c("linear", "two-class", "multiclass", "survival"),
  hyperparameter_grid = tof_create_grid(),
  standardize_predictors = TRUE,
  remove_zv_predictors = FALSE,
  impute_missing_predictors = FALSE,
  optimization_metric = "tidytof_default",
  best_model_type = c("best", "best with sparsity"),
  num_cores = 1
)
```

Arguments

split_data	An ‘rsplit‘ or ‘rset‘ object from the rsample package containing the sample-level data to use for modeling. The easiest way to generate this is to use tof_split_data.
unsplit_data	A tibble containing sample-level data to use for modeling without resampling. While using a resampling method is advised, this argument provides an interface to fit a model without using cross-validation or bootstrap resampling. Ignored if split_data is provided.
predictor_cols	Unquoted column names indicating which columns in the data contained in ‘split_data‘ should be used as predictors in the elastic net model. Supports tidyselect helpers.
response_col	Unquoted column name indicating which column in the data contained in ‘split_data‘ should be used as the outcome in a "two-class", "multiclass", or "linear" elastic net model. Must be a factor for "two-class" and "multiclass" models and must be a numeric for "linear" models. Ignored if ‘model_type‘ is "survival".
time_col	Unquoted column name indicating which column in the data contained in ‘split_data‘ represents the time-to-event outcome in a "survival" elastic net model. Must be numeric. Ignored if ‘model_type‘ is "two-class", "multiclass", or "linear".
event_col	Unquoted column name indicating which column in the data contained in ‘split_data‘ represents the time-to-event outcome in a "survival" elastic net model. Must be a binary column - all values should be either 0 or 1 (with 1 indicating the adverse event) or FALSE and TRUE (with TRUE indicating the adverse event). Ignored if ‘model_type‘ is "two-class", "multiclass", or "linear".
model_type	A string indicating which kind of elastic net model to build. If a continuous response is being predicted, use "linear" for linear regression; if a categorical response with only 2 classes is being predicted, use "two-class" for logistic regression; if a categorical response with more than 2 levels is being predicted, use "multiclass" for multinomial regression; and if a time-to-event outcome is being predicted, use "survival" for Cox regression.

hyperparameter_grid
: A hyperparameter grid indicating which values of the elastic net penalty (lambda) and the elastic net mixture (alpha) hyperparamters should be used during model tuning. Generate this grid using tof_create_grid.

standardize_predictors
: A logical value indicating if numeric predictor columns should be standardized (centered and scaled) before model fitting, as is standard practice during elastic net regularization. Defaults to TRUE.

remove_zv_predictors
: A logical value indicating if predictor columns with near-zero variance should be removed before model fitting using step_nzv. Defaults to FALSE.

impute_missing_predictors
: A logical value indicating if predictor columns should have missing values imputed using k-nearest neighbors before model fitting (see step_impute_knn). Imputation is performed using an observation's 5 nearest-neighbors. Defaults to FALSE.

optimization_metric
: A string indicating which optimization metric should be used for hyperparameter selection during model tuning. Valid values depend on the model_type.

  - For "linear" models, choices are "mse" (the mean squared error of the predictions; the default) and "mae" (the mean absolute error of the predictions).
  - For "two-class" models, choices are "roc_auc" (the area under the Receiver-Operating Curve for the classification; the default), "misclassification error" (the proportion of misclassified observations), "binomial_deviance" (see deviance.glmnet), "mse" (the mean squared error of the logit function), and "mae" (the mean absolute error of the logit function).
  - For "multiclass" models, choices are "roc_auc" (the area under the Receiver-Operating Curve for the classification using the Hand-Till generalization of the ROC AUC for multiclass models in roc_auc; the default), "misclassification error" (the proportion of misclassified observations), "multinomial_deviance" (see deviance.glmnet), and "mse" and "mae" as above.
  - For "survival" models, choices are "concordance_index" (Harrel's C index; see deviance.glmnet) and "partial_likelihood_deviance" (see deviance.glmnet).

best_model_type
: Currently unused.

num_cores
: Integer indicating how many cores should be used for parallel processing when fitting multiple models. Defaults to 1. Overhead to separate models across multiple cores can be high, so significant speedup is unlikely to be observed unless many large models are being fit.

Value

A 'tof_model', an S3 class that includes the elastic net model with the best performance (assessed via cross-validation, bootstrapping, or simple splitting depending on 'split_data') across all tested hyperparameter value combinations. 'tof_models' store the following information:

model The final elastic net ("glmnet") model, which is chosen by selecting the elastic net hyperparameters with the best 'optimization_metric' performance on the validation sets of each resample used to train the model (on average)

recipe The recipe used for data preprocessing

mixture The optimal mixture hyperparameter (alpha) for the glmnet model

penalty The optimal penalty hyperparameter (lambda) for the glmnet model

model_type A string indicating which type of glmnet model was fit

outcome_colnames A character vector representing the names of the columns in the training data modeled as outcome variables

training_data A tibble containing the (not preprocessed) data used to train the model

tuning_metrics A tibble containing the validation set performance metrics (and model predictions) during for each resample fold during model tuning.

log_rank_thresholds For survival models only, a tibble containing information about the relative-risk thresholds that can be used to split the training data into 2 risk groups (low- and high-risk) based on the final model's predictions. For each relative-risk threshold, the log-rank test p-value and an indicator of which threshold gives the most significant separation is provided.

best_log_rank_threshold For survival models only, a numeric value representing the relative-risk threshold that yields the most significant log-rank test when separating the training data into low- and high-risk groups.

See Also

Other modeling functions: tof_assess_model(), tof_create_grid(), tof_predict(), tof_split_data()

Examples

```
feature_tibble <-
    dplyr::tibble(
        sample = as.character(1:100),
        cd45 = runif(n = 100),
        pstat5 = runif(n = 100),
        cd34 = runif(n = 100),
        outcome = (3 * cd45) + (4 * pstat5) + rnorm(100),
        class =
            as.factor(
                dplyr::if_else(outcome > median(outcome), "class1", "class2")
            ),
        multiclass =
            as.factor(
                c(rep("class1", 30), rep("class2", 30), rep("class3", 40))
            ),
        event = c(rep(0, times = 30), rep(1, times = 70)),
        time_to_event = rnorm(n = 100, mean = 10, sd = 2)
    )

split_data <- tof_split_data(feature_tibble, split_method = "simple")

# train a regression model
```

```
tof_train_model(
    split_data = split_data,
    predictor_cols = c(cd45, pstat5, cd34),
    response_col = outcome,
    model_type = "linear"
)

# train a logistic regression classifier
tof_train_model(
    split_data = split_data,
    predictor_cols = c(cd45, pstat5, cd34),
    response_col = class,
    model_type = "two-class"
)

# train a cox regression survival model
tof_train_model(
    split_data = split_data,
    predictor_cols = c(cd45, pstat5, cd34),
    time_col = time_to_event,
    event_col = event,
    model_type = "survival"
)
```

`tof_transform` *Transform raw high-dimensional cytometry data.*

Description

This function transforms a ‘tof_tbl‘ of raw ion counts, reads, or fluorescence intensity units directly measured on a cytometer using a user-provided function.

Usage

```
tof_transform(
  tof_tibble = NULL,
  channel_cols = where(tof_is_numeric),
  transform_fun
)
```

Arguments

`tof_tibble`	A ‘tof_tbl‘ or a ‘tibble‘.
`channel_cols`	Unquoted column names representing columns that contain single-cell protein measurements. Supports tidyselect helpers. If nothing is specified, the default is to transform all numeric columns.
`transform_fun`	A vectorized function to apply to each protein value for variance stabilization.

Value

A ‘tof_tbl‘ with identical dimensions to the input ‘tof_tibble‘, with all columns specified in channel_cols transformed using ‘transform_fun‘.

Examples

```
# read in an example .fcs file from tidytof's internal datasets
input_file <- dir(tidytof_example_data("aml"), full.names = TRUE)[[1]]
tof_tibble <- tof_read_data(input_file)

# preprocess all numeric columns with default behavior
# arcsinh transformation with a cofactor of 5
tof_preprocess(tof_tibble)

# preprocess all numeric columns using the log base 10 tranformation
tof_preprocess(tof_tibble, transform_fun = log10)
```

---

tof_tune_glmnet *Tune an elastic net model's hyperparameters over multiple resamples*

Description

Tune an elastic net model's hyperparameters over multiple resamples

Usage

```
tof_tune_glmnet(
  split_data,
  prepped_recipe,
  hyperparameter_grid,
  model_type,
  outcome_cols,
  optimization_metric = "tidytof_default",
  num_cores = 1
)
```

Arguments

split_data	An ‘rsplit‘ or ‘rset‘ object from the rsample package. The easiest way to generate this is to use tof_split_data. Alternatively, an unsplit tbl_df can be provided, though this is not recommended.
prepped_recipe	Either a single recipe object (if ‘split_data‘ is an ‘rsplit‘ object or a ‘tbl_df‘) or list of recipes (if ‘split_data‘ is an ‘rset‘ object) such that each entry in the list corresponds to a resample in ‘split_data‘.

`hyperparameter_grid`
: A hyperparameter grid indicating which values of the elastic net penalty (lambda) and the elastic net mixture (alpha) hyperparameters should be used during model tuning. Generate this grid using `tof_create_grid`.

`model_type`
: A string indicating which kind of elastic net model to build. If a continuous response is being predicted, use "linear" for linear regression; if a categorical response with only 2 classes is being predicted, use "two-class" for logistic regression; if a categorical response with more than 2 levels is being predicted, use "multiclass" for multinomial regression; and if a time-to-event outcome is being predicted, use "survival" for Cox regression.

`outcome_cols`
: Unquoted column name(s) indicating which column(s) in the data contained in 'split_data' should be used as the outcome in the elastic net model. For survival models, two columns should be selected; for all others, only one column should be selected.

`optimization_metric`
: A string indicating which optimization metric should be used for hyperparameter selection during model tuning. Valid values depend on the model_type.

`num_cores`
: Integer indicating how many cores should be used for parallel processing when fitting multiple models. Defaults to 1. Overhead to separate models across multiple cores can be high, so significant speedup is unlikely to be observed unless many large models are being fit.

Value

A tibble containing a summary of the model's performance in each resampling iteration across all hyperparameter combinations. Will contain 3 columns: "splits" (a list-col containing each resampling iteration's 'rsplit' object), "id" (the name of the resampling iteration), and "performance_metrics" (a list-col containing the performance metrics for each resampling iteration. Each row of "performance_metrics" is a tibble with the columns "mixture" and "penalty" and several additional columns containing the performance metrics of the model for each mixture/penalty combination). See `tof_fit_split` for additional details.

`tof_upsample` *Upsample cells into the closest cluster in a reference dataset*

Description

This function performs distance-based upsampling on CyTOF data by sorting single cells (passed into the function as 'tof_tibble') into their most phenotypically similar cell subpopulation in a reference dataset (passed into the function as 'reference_tibble'). It does so by calculating the distance (either mahalanobis, cosine, or pearson) between each cell in 'tof_tibble' and the centroid of each cluster in 'reference_tibble', then sorting cells into the cluster corresponding to their closest centroid.

Usage

```
tof_upsample(
  tof_tibble,
  reference_tibble,
  reference_cluster_col,
  upsample_cols = where(tof_is_numeric),
  ...,
  augment = TRUE,
  method = c("distance", "neighbor")
)
```

Arguments

tof_tibble	A ‘tibble‘ or ‘tof_tbl‘ containing cells to be upsampled into their nearest reference subpopulation.
reference_tibble	A ‘tibble‘ or ‘tof_tibble‘ containing cells that have already been clustered or manually gated into subpopulations.
reference_cluster_col	An unquoted column name indicating which column in ‘reference_tibble‘ contains the subpopulation label (or cluster id) for each cell in ‘reference_tibble‘.
upsample_cols	Unquoted column names indicating which columns in ‘tof_tibble‘ to use in computing the distances used for upsampling. Defaults to all numeric columns in ‘tof_tibble‘. Supports tidyselect helpers.
...	Additional arguments to pass to the ‘tof_upsample_*‘ function family member corresponding to the chosen method.
augment	A boolean value indicating if the output should column-bind the cluster ids of each cell as a new column in ‘tof_tibble‘ (TRUE, the default) or if a single-column tibble including only the cluster ids should be returned (FALSE).
method	A string indicating which clustering methods should be used. Valid values include "distance" (default) and "neighbor".

Value

A ‘tof_tbl‘ or ‘tibble‘ If augment = FALSE, it will have a single column encoding the upsampled cluster ids for each cell in ‘tof_tibble‘. If augment = TRUE, it will have ncol(tof_tibble) + 1 columns: each of the (unaltered) columns in ‘tof_tibble‘ plus an additional column encoding the cluster ids.

Examples

```
# simulate single-cell data (and reference data with clusters to upsample
# into
sim_data <-
    dplyr::tibble(
        cd45 = rnorm(n = 1000),
        cd38 = rnorm(n = 1000),
```

```
        cd34 = rnorm(n = 1000),
        cd19 = rnorm(n = 1000)
    )
reference_data <-
    dplyr::tibble(
        cd45 = rnorm(n = 200),
        cd38 = rnorm(n = 200),
        cd34 = rnorm(n = 200),
        cd19 = rnorm(n = 200),
        cluster_id = c(rep("a", times = 100), rep("b", times = 100))
    )

# upsample using distance to cluster centroids
tof_upsample(
    tof_tibble = sim_data,
    reference_tibble = reference_data,
    reference_cluster_col = cluster_id,
    method = "distance"
)

# upsample using distance to nearest neighbor
tof_upsample(
    tof_tibble = sim_data,
    reference_tibble = reference_data,
    reference_cluster_col = cluster_id,
    method = "neighbor"
)
```

`tof_upsample_distance` *Upsample cells into the closest cluster in a reference dataset*

Description

This function performs distance-based upsampling on CyTOF data by sorting single cells (passed into the function as 'tof_tibble') into their most phenotypically similar cell subpopulation in a reference dataset (passed into the function as 'reference_tibble'). It does so by calculating the distance (either mahalanobis, cosine, or pearson) between each cell in 'tof_tibble' and the centroid of each cluster in 'reference_tibble', then sorting cells into the cluster corresponding to their closest centroid.

Usage

```
tof_upsample_distance(
  tof_tibble,
  reference_tibble,
  reference_cluster_col,
  upsample_cols = where(tof_is_numeric),
  parallel_cols,
```

```
  distance_function = c("mahalanobis", "cosine", "pearson"),
  num_cores = 1L,
  return_distances = FALSE
)
```

Arguments

tof_tibble	A ‘tibble‘ or ‘tof_tbl‘ containing cells to be upsampled into their nearest reference subpopulation.
reference_tibble	A ‘tibble‘ or ‘tof_tibble‘ containing cells that have already been clustered or manually gated into subpopulations.
reference_cluster_col	An unquoted column name indicating which column in ‘reference_tibble‘ contains the subpopulation label (or cluster id) for each cell in ‘reference_tibble‘.
upsample_cols	Unquoted column names indicating which columns in ‘tof_tibble‘ to use in computing the distances used for upsampling. Defaults to all numeric columns in ‘tof_tibble‘. Supports tidyselect helpers.
parallel_cols	Optional. Unquoted column names indicating which columns in ‘tof_tibble‘ to use for breaking up the data in order to parallelize the upsampling using ‘foreach‘ on a ‘doParallel‘ backend. Supports tidyselect helpers.
distance_function	A string indicating which distance function should be used to perform the upsampling. Options are "mahalanobis" (the default), "cosine", and "pearson".
num_cores	An integer indicating the number of CPU cores used to parallelize the classification. Defaults to 1 (a single core).
return_distances	A boolean value indicating whether or not the returned result should include only one column, the cluster ids corresponding to each row of ‘tof_tibble‘ (return_distances = FALSE, the default), or if the returned result should include additional columns representing the distance between each row of ‘tof_tibble‘ and each of the reference subpopulation centroids (return_distances = TRUE).

Value

If ‘return_distances = FALSE‘, a tibble with one column named ‘.upsample_cluster‘, a character vector of length ‘nrow(tof_tibble)‘ indicating the id of the reference cluster to which each cell (i.e. each row) in ‘tof_tibble‘ was assigned.

If ‘return_distances = TRUE‘, a tibble with ‘nrow(tof_tibble)‘ rows and num_clusters + 1 columns, where num_clusters is the number of clusters in ‘reference_tibble‘. Each row represents a cell from ‘tof_tibble‘, and num_clusters of the columns represent the distance between the cell and each of the reference subpopulations’ cluster centroids. The final column represents the cluster id of the reference subpopulation with the minimum distance to the cell represented by that row.

Examples

```
# simulate single-cell data (and reference data with clusters to upsample
```

```
# into
sim_data <-
    dplyr::tibble(
        cd45 = rnorm(n = 1000),
        cd38 = rnorm(n = 1000),
        cd34 = rnorm(n = 1000),
        cd19 = rnorm(n = 1000)
    )

reference_data <-
    dplyr::tibble(
        cd45 = rnorm(n = 200),
        cd38 = rnorm(n = 200),
        cd34 = rnorm(n = 200),
        cd19 = rnorm(n = 200),
        cluster_id = c(rep("a", times = 100), rep("b", times = 100))
    )

# upsample using mahalanobis distance
tof_upsample_distance(
    tof_tibble = sim_data,
    reference_tibble = reference_data,
    reference_cluster_col = cluster_id
)

# upsample using cosine distance
tof_upsample_distance(
    tof_tibble = sim_data,
    reference_tibble = reference_data,
    reference_cluster_col = cluster_id,
    distance_function = "cosine"
)
```

`tof_upsample_neighbor` *Upsample cells into the cluster of their nearest neighbor a reference dataset*

Description

This function performs upsampling on CyTOF data by sorting single cells (passed into the function as 'tof_tibble') into their most phenotypically similar cell subpopulation in a reference dataset (passed into the function as 'reference_tibble'). It does so by finding each cell in 'tof_tibble''s nearest neighbor in 'reference_tibble' and assigning it to the cluster to which its nearest neighbor belongs. The nearest neighbor calculation can be performed with either euclidean or cosine distance.

Usage

```
tof_upsample_neighbor(
```

```
  tof_tibble,
  reference_tibble,
  reference_cluster_col,
  upsample_cols = where(tof_is_numeric),
  num_neighbors = 1L,
  distance_function = c("euclidean", "cosine", "l2", "ip")
)
```

Arguments

tof_tibble	A ‘tibble‘ or ‘tof_tbl‘ containing cells to be upsampled into their nearest reference subpopulation.
reference_tibble	A ‘tibble‘ or ‘tof_tibble‘ containing cells that have already been clustered or manually gated into subpopulations.
reference_cluster_col	An unquoted column name indicating which column in ‘reference_tibble‘ contains the subpopulation label (or cluster id) for each cell in ‘reference_tibble‘.
upsample_cols	Unquoted column names indicating which columns in ‘tof_tibble‘ to use in computing the distances used for upsampling. Defaults to all numeric columns in ‘tof_tibble‘. Supports tidyselect helpers.
num_neighbors	An integer indicating how many neighbors should be used in the nearest neighbor calculation. Clusters are assigned based on majority vote.
distance_function	A string indicating which distance function should be used to perform the upsampling. Options are "euclidean" (the default) and "cosine".

Value

A tibble with one column named ‘.upsample_cluster‘, a character vector of length ‘nrow(tof_tibble)‘ indicating the id of the reference cluster to which each cell (i.e. each row) in ‘tof_tibble‘ was assigned.

Examples

```
# simulate single-cell data (and reference data with clusters to upsample
# into
sim_data <-
    dplyr::tibble(
        cd45 = rnorm(n = 1000),
        cd38 = rnorm(n = 1000),
        cd34 = rnorm(n = 1000),
        cd19 = rnorm(n = 1000)
    )

reference_data <-
    dplyr::tibble(
        cd45 = rnorm(n = 200),
        cd38 = rnorm(n = 200),
```

```
        cd34 = rnorm(n = 200),
        cd19 = rnorm(n = 200),
        cluster_id = c(rep("a", times = 100), rep("b", times = 100))
    )

# upsample using euclidean distance
tof_upsample_neighbor(
    tof_tibble = sim_data,
    reference_tibble = reference_data,
    reference_cluster_col = cluster_id
)

# upsample using cosine distance
tof_upsample_neighbor(
    tof_tibble = sim_data,
    reference_tibble = reference_data,
    reference_cluster_col = cluster_id,
    distance_function = "cosine"
)
```

`tof_write_csv` *Write a series of .csv files from a tof_tbl*

Description

This function takes a given ‘tof_tbl‘ and writes the single-cell data it contains into .csv files within the directory located at ‘out_path‘. The ‘group_cols‘ argument specifies how the rows of the ‘tof_tbl‘ (each cell) should be broken into separate .csv files

Usage

```
tof_write_csv(tof_tibble, group_cols, out_path, sep = "_", file_name)
```

Arguments

`tof_tibble`	A ‘tof_tbl‘ or a ‘tibble‘.
`group_cols`	Optional. Unquoted names of the columns in ‘tof_tibble‘ that should be used to group cells into separate files. Supports tidyselect helpers. Defaults to NULL (all cells are written into a single file).
`out_path`	A system path indicating the directory where the output .csv files should be saved. If the directory doesn’t exist, it will be created.
`sep`	Delimiter that should be used between each of the values of ‘group_cols‘ to create the output .csv file names. Defaults to "_".
`file_name`	If ‘group_cols‘ isn’t specified, the name (without an extension) that should be used for the saved .csv file.

Value

This function does not return anything. Instead, it has the side-effect of saving .csv files to ‘out_path‘.

See Also

Other input/output functions: `tof_read_data()`, `tof_write_data()`, `tof_write_fcs()`

`tof_write_data` *Write high-dimensional cytometry data to a file or to a directory of files*

Description

Write data (in the form of a ‘tof_tbl‘) into either a .csv or an .fcs file for storage.

Usage

```
tof_write_data(
  tof_tibble = NULL,
  group_cols,
  out_path = NULL,
  format = c("fcs", "csv"),
  sep = "_",
  file_name
)
```

Arguments

`tof_tibble`	A ‘tof_tbl‘ or a ‘tibble‘.
`group_cols`	Optional. Unquoted names of the columns in ‘tof_tibble‘ that should be used to group cells into separate files. Supports tidyselect helpers. Defaults to no grouping (all cells are written into a single file).
`out_path`	Path to the directory where output files should be saved.
`format`	format for the files being written. Currently supports .csv and .fcs files
`sep`	Delimiter that should be used between each of the values of ‘group_cols‘ to create the output .csv/.fcs file names. Defaults to "_".
`file_name`	If ‘group_cols‘ isn’t specified, the name (without an extension) that should be used for the saved file.

Value

This function does not explicitly return any values. Instead, it writes .csv and/or .fcs files to the specified ‘out_path‘.

See Also

Other input/output functions: `tof_read_data()`, `tof_write_csv()`, `tof_write_fcs()`

Examples

```
NULL
```

`tof_write_fcs` *Write a series of .fcs files from a tof_tbl*

Description

This function takes a given ‘tof_tbl‘ and writes the single-cell data it contains into .fcs files within the directory located at ‘out_path‘. The ‘group_cols‘ argument specifies how the rows of the ‘tof_tbl‘ (each cell) should be broken into separate .fcs files

Usage

```
tof_write_fcs(tof_tibble, group_cols, out_path, sep = "_", file_name)
```

Arguments

`tof_tibble`	A ‘tof_tbl‘ or a ‘tibble‘.
`group_cols`	Unquoted names of the columns in ‘tof_tibble‘ that should be used to group cells into separate files. Supports tidyselect helpers. Defaults to NULL (all cells are written into a single file).
`out_path`	A system path indicating the directory where the output .csv files should be saved. If the directory doesn’t exist, it will be created.
`sep`	Delimiter that should be used between each of the values of ‘group_cols‘ to create the output .fcs file names. Defaults to "_".
`file_name`	If ‘group_cols‘ isn’t specified, the name (without an extension) that should be used for the saved .csv file.

Value

This function does not return anything. Instead, it has the side-effect of saving .fcs files to ‘out_path‘.

See Also

Other input/output functions: `tof_read_data()`, `tof_write_csv()`, `tof_write_data()`

Examples

```
NULL
```

`where` *Select variables with a function*

Description

This is a copy of `where`, a selection helper that selects the variables for which a predicate function returns TRUE. See `language` for more details about tidyselection.

Usage

```
where(fn)
```

Arguments

`fn`	A function that returns TRUE or FALSE (technically, a predicate function). Can also be a purrr-like formula.

Details

This help file was replicated verbatim from `tidyselect-package`.

Value

A predicate that can be used to select columns from a data.frame.

References

Lionel Henry and Hadley Wickham (2021). tidyselect: Select from a Set of Strings. R package version 1.1.1. https://CRAN.R-project.org/package=tidyselect

Examples

```
NULL
```

Index