Package 'Sconify'

November 26, 2024

Type Package

Title A toolkit for performing KNN-based statistics for flow and mass cytometry data

biocViews ImmunoOncology, SingleCell, FlowCytometry, Software, MultipleComparison, Visualization

Version 1.26.0

Author Tyler J Burns

Maintainer Tyler J Burns <burns.tyler@gmail.com>

Description
This package does k-nearest neighbor based statistics and visualizations with flow and mass cytometery data. This gives tSNE maps``fold change" functionality and provides a data quality metric by assessing manifold overlap between fcs files expected to be the same. Other applications using this package include imputation, marker redundancy, and testing the relative information loss of lower dimension embeddings compared to the original manifold.

License Artistic-2.0

Encoding UTF-8

LazyData true

Depends R (>= 3.5)

Imports tibble, dplyr, FNN, flowCore, Rtsne, ggplot2, magrittr, utils, stats, readr

RoxygenNote 6.0.1

Suggests knitr, rmarkdown, testthat

VignetteBuilder knitr

Collate 'k.selection.R' 'process.files.R' 'knn.and.statistics.R' 'post.processing.R' 'data.R'

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

git_branch RELEASE_3_20

git_last_commit 83670d8

git_last_commit_date 2024-10-29

Repository Bioconductor 3.20

Date/Publication 2024-11-25

Contents

`AddTsne` *Add tSNE to your results.*

Description

This function gives the user the option to add t-SNE to the final output, using the same input features used in KNN, eg. surface markers, as input for t-SNE.

Usage

```
AddTsne(dat, input)
```

Arguments

dat	matrix of cells by features, that contain all features needed for tSNE analysis
input	the features to be used as input for tSNE,usually the same for knn generation

Value

result: dat, with tSNE1 and tSNE2 attached

---

bz.gmcsf.final *Bodenmiller-Zunder GM-CSF post-SCONE final data*

Description

The post-SCONE output from Bodenmiller-Zunder dataset pair of fcs files, one untreated and one treated with GM-CSF.We ran this on 10,000 cells and subsampled to 1000 for this vignette.

Usage

```
bz.gmcsf.final
```

Format

A tibble of 1000 cells by 69 features. This includes all the original parameters, the KNN-generated comparisons, differential abundance ("fraction.cond.2), and two t-SNE coordinates.

---

bz.gmcsf.final.norm.scale *Bodenmiller-Zunder GM-CSF post-SCONE final data, that's been quantile normalized and z scored.*

Description

The post-SCONE output from a per-marker quantile normalized and z scored treated with GM-CSF. We ran this on 10,000 cells and sub-sampled to 1000 for this package.

Usage

```
bz.gmcsf.final.norm.scale
```

Format

A tibble of 1000 cells by 69 features. This includes all the original parameters, the KNN-generated comparisons, differential abundance ("fraction.cond.2), and two t-SNE coordinates.

exist — *Random musing*

Description

Seriously random

Usage

```
exist
```

Format

a string

FcsToTibble — *Takes a file as input and returns a data frame of cells by features*

Description

This function is a quick way to take the exprs content of a fcs file, do an asinh transform, and create a tibble data structure that can be further manipulated. Our default transform is asinh, but you just have to change the transform to anything else, and you'll get the raw data. This function is used in the main function process.multiple.files

Usage

```
FcsToTibble(file, transform = "asinh")
```

Arguments

file	the fcs file containing cell infomration
transform	if set to asinh, then asinh transforms with scale arg 5

Value

tibble of info contained within the fcs file

Examples

```
file <- system.file("extdata", "Bendall_et_al_Cell_Sample_C_basal.fcs",
    package = "Sconify")
FcsToTibble(file)
```

Fnn *Compute knn using the fnn algorithm*

Description

This function is a wrapper around FNN package functionality to speed up the KNN process. It uses KD trees as default, along with k set to 100. Selection of K will vary based on the dataset. See k.selection.R.

Usage

```
Fnn(cell.df, input.markers, k = 100)
```

Arguments

cell.df	the cell data frame used as input
input.markers	markers to be used as input for knn
k	the number of nearest neighbors to identify

Value

nn: list of 2, nn.index: index of knn (columns) for each cell (rows) nn.dist: euclidean distance of each k-nearest neighbor

Examples

```
Fnn(wand.combined, input.markers)
```

funct.markers *Functional markers from the Wanderlust dataset.*

Description

These are the markers that will be used in the KNN comparisons, as opposed to the KNN generation.

Usage

```
funct.markers
```

Format

A vector of strings.

GetKnnDe *Get the KNN density estimaion*

Description

Obtain a density estimation derived from the original manifold, avoiding the lossiness of lower dimensional embeddings

Usage

```
GetKnnDe(nn.matrix)
```

Arguments

nn.matrix	A list of 2, where the first is a matrix of nn indices, and the second is a matrix of nn distances

Value

a vector where each element is the KNN-DE for that given cell, ordered by row number, in the original input matrix of cells x features

Examples

```
ex.knn <- Fnn(wand.combined, input.markers, k = 30)
GetKnnDe(ex.knn)
```

GetMarkerNames *Takes in an example file as input and returns all marker names*

Description

This is a quick way to get a list of preferred marker names. This outputs a csv file containing all markers in the dataset in the name format that will be recognized by downstream functions. You manually alter this list to remove and/or categorize the said markers. The file can then be read in (stringsAsFactors = FALSE) to give you the marker list of interest. In particular, name the top of the column as "markers" if you're just altering the list. If you're doing to divide it into static and functional markers, produce two columns, naming them respectively.

Usage

```
GetMarkerNames(file)
```

Arguments

file	the fcs file of interest

Value

the list of markers of interest. This is to be written as a csv

Examples

```
file <- system.file("extdata", "Bendall_et_al_Cell_Sample_C_basal.fcs",
    package = "Sconify")
GetMarkerNames(file)
```

`Impute` *Imputes values for all markers (used as input) for each cell*

Description

This function takes as input the markers to be imputed from a pre-existing KNN computation.

Usage

```
Impute(cells, input.markers, nn)
```

Arguments

`cells`	the input matrix of cells
`input.markers`	the markers the user wants to impute
`nn`	the matrix of k-nearest neighbors (derived perhaps NOT from the "input markers" above)

Value

a data frame of imputed cells for the "input markers" of interest

---

`ImputeTesting` *Impute testing*

Description

Tests the euclidean distance error for imputation using knn and markers of interest

Usage

```
ImputeTesting(k.titration, cells, input.markers, test.markers)
```

Arguments

`k.titration`	a vector integer values of k to be tested
`cells`	a matrix of cells by features used as original input
`input.markers`	markers to be used for the knn calculation
`test.markers`	the markers to be tested for imputation (either surface or scone)

Value

the median imputation error for each value k tested

Examples

```
ImputeTesting(k.titration = c(10, 20),
  cells = wand.combined,
  input.markers = input.markers,
  test.markers = funct.markers)
```

`input.markers` *Input markers for the Wanderlust dataset*

Description

These are the markers that KNN generation will be done on for the Wanderlust dataset. These are mostly surface markers.These are the same markers one would use as input for clustering or t-SNE generation, for exmaple, as they are not expected to change through the duration of the quick IL7 stimulation.

Usage

```
input.markers
```

Format

A vector of strings corresponding to the markers.

`LogTransformQ` *Log transform the q values*

Description

Takes all p values from the data and does a log10 transform for easier visualization.

Usage

```
LogTransformQ(dat, negative)
```

Arguments

`dat`	tibble containing cells x features, with orignal expression, p values, and raw change
`negative`	boolean value to determine whether to multiple transformed p values by -1

Value

result: tibble of cells x features with all p values log10 transformed

MakeHist *make.hist*

Description

Makes a histogram of the data that is inputted

Usage

```
MakeHist(dat, k, column.label, x.label)
```

Arguments

dat	tibble consisting both of original markers and the appended values from scone
k	the binwidth, set to 1/k
column.label	the label in the tibble's columns the function will search for
x.label	the label that the x axis will be labeled as

Value

a histogram of said vector in ggplot2 form

Examples

```
MakeHist(wand.final, 100, "IL7.fraction.cond.2", "fraction IL7")
```

MakeKnnList *Make list of cells by features for each KNN member*

Description

Takes the KNN function output and the cell data, and makes list where each element is a matrix of cells in the KNN and features.

Usage

```
MakeKnnList(cell.data, nn.matrix)
```

Arguments

cell.data	tibble of cells by features
nn.matrix	list of 2. First element is cells x 100 nearest neighbor indices. Second element is cells x 100 nearest neighbor distances

Value

a list where each element is the cell number from the original cell.data tibble and a matrix of cells x feautures for its KNN

Examples

```
ex.knn <- Fnn(wand.combined, input.markers, k = 30)
knn.list <- MakeKnnList(wand.combined, ex.knn)
```

`markers` *Markers for the Wanderlust dataset*

Description

Both the surface and functional markers for the Wanderlust dataset

Usage

```
markers
```

Format

a tibble with two columns, "surface" and "fucntional."

`MeaningOfLife` *Meaning of life*

Description

Just a random musing

Usage

```
MeaningOfLife()
```

Value

A string containing a random musing

Examples

```
MeaningOfLife()
```

MultipleDonorStatistics

Runs a t test on the medians or means of multiple donors for the same condition

Description

This function is for the instance that multiple donors are being compared against each other within the k-nearest neighborhood of interest. The mean value of the markers of interest is calculated across the donors, such that each data point for the subsequent t-test represents a marker for a danor.

Usage

```
MultipleDonorStatistics(basal, stim, stim.name, donors)
```

Arguments

basal	tibble that contains unstim for a knn including donor identity
stim	tibble that contains stim for a knn including donor identity
stim.name	string of the name of the current stim being tested
donors	vector of strings corresponding to the designated names of the donors

Value

result: a named vector of p values (soon to be q values) from the t test done on each marker

ParseMarkers

Parse markers contained in a Sconify-directed marker file

Description

This occurs after the user has modified the markers.csv file to determine which markers are to be used as input for KNN and which markers are to be used for within-knn comparisons

Usage

```
ParseMarkers(marker.file)
```

Arguments

marker.file	modified markers.csv file, now containing two columns. the left column containing KNN input markers, and the right column containing KNN comparison markers

Value

a list of 2 vectors of strings. The first element, labeled "input" is a vector KNN input markers. THe second slemenet, labeled "functional" are the markers to be used in the KNN based comparisons

Examples

```
file <- system.file("extdata", "markers.csv", package = "Sconify")
ParseMarkers(file)
```

`PostProcessing` *Post-processing for scone analysks.*

Description

Performs final processing and transformations on the scone data

Usage

```
PostProcessing(scone.output, cell.data, input, tsne = TRUE,
  log.transform.qvalue = TRUE)
```

Arguments

`scone.output`	tibble of the output of the given scone analysis
`cell.data`	the tibble used as input for the scone.values function
`input`	the input markers used for the knn calculation (to be used for tsne here)
`tsne`	boolean value to indicate whether tSNE is to be done
`log.transform.qvalue`	boolean to indicate whether log transformation of all q values is to be done

Value

result: the concatenated original input data with the scone derived data, with the option of the q values being inverse log10 transformed, and two additional tSNE columns being added to the data (from the Rtsne package)

Examples

```
PostProcessing(wand.scone, wand.combined, input.markers, tsne = FALSE)
```

`ProcessMultipleFiles` *Converts multiple files into a concatenated data frame*

Description

This is tailored to a very specific file format for unstim/stim Files need the following name convention: "xxxx_stim.fcs" Files where you want to name the patients need the following convention: "xxxx__patientID_stim.fcs"

Usage

```
ProcessMultipleFiles(files, transform = "asinh", numcells = 10000,
  norm = FALSE, scale = FALSE, input, name.multiple.donors = FALSE)
```

Arguments

files	a vector of file names (name = "anything_condition.fcs")
transform	set to asinh if you want to do an asinh transform of all markers in the dataset
numcells	desiered number of cells in the matrix, set at 10k
norm	boolean that quantile normalizes the data if true
scale	boolean that converts all values to z scores if true
input	the static markers that will be used downstream in knn computation. These are included here to include the option of per-marker quantile normalization, in the event norm is set to TRUE
name.multiple.donors	boolean indicating whether multiple donors will be distinguished (as a separate "patient" column)

Value

result: a combined file set

Examples

```
file1 <- system.file("extdata", "Bendall_et_al_Cell_Sample_C_basal.fcs",
    package = "Sconify")
file2 <- system.file("extdata", "Bendall_et_al_Cell_Sample_C_IL7.fcs",
    package = "Sconify")
ProcessMultipleFiles(c(file1, file2), input = input.markers)
```

QCorrectionThresholding

Corrects all p values for multiple hypotheses, sets threshold for which change values should be reported

---

Description

Given the number of comparisons we make across k-nearest neighborhoods, which is far more than that of disjoint subsetting, this step is important given that there is an increased likelihood that some statistically significant differences will occur by chance.

Usage

```
QCorrectionThresholding(cells, threshold)
```

Arguments

cells	tibble of change values, p values, and fraction condition 2
threshold	a q value below which the change values will be reported for that cell for that param. If no change is desired, this is set to 1.

Value

inputted p values, adjusted and therefore described as "q values"

`QuantNormalize` *Performs quantile normalization on the data frame (patient) of interest*

Description

Credit goes to: http://davetang.org/muse/2014/07/07/quantile-normalisation-in-r/ for this function

Usage

```
QuantNormalize(df)
```

Arguments

`df`	a data frame with rows as cells and columns as features

Value

a data frame where the columns have been quantile normalized

`QuantNormalizeElements` *Takes a list of tibbles as input, and performs per-column quantile normalization, then outputs the quantile normalized list*

Description

This function performs per-marker quantile normalization on multiple data tibbles. The normalization occurrs marker by marker. The user assumes that the markers are distributed equally across tibbles, as quantile normalization forces these marker distributions to be the same per file

Usage

```
QuantNormalizeElements(dat.list)
```

Arguments

`dat.list`	a list of tibbles

Value

the per-column quantile normalized list

Examples

```
basal <- wand.combined[wand.combined$condition == "basal",][,1:10]
il7 <- wand.combined[wand.combined$condition == "IL7",][,1:10]
QuantNormalizeElements(list(basal, il7))
```

`RunStatistics` *Performs a series of statistical tests on the batch of cells of interest.*

Description

This function performs the statistics across the nearest neighborhoods, and is one of the main workhorses within the scone.values function

Usage

```
RunStatistics(basal, stim, fold = "median", stat.test = "mwu", stim.name)
```

Arguments

`basal`	tibble of cells corresponding to the unstimulated condition
`stim`	a tibble of cells corresponding to the stimulated condition
`fold`	a string that specifies the use of "median" or "mean" when calculating fold change
`stat.test`	a string that specifies Mann-Whitney U test (mwu) or T test (t) for q value calculation
`stim.name`	a string corresponding to the name of the stim being tested compared to basal

Value

result: a named vector corresponding to the results of the "fold change" and mann-whitney u test

`SconeValues` *Master function for per-knn statistics functionality, integrating the other non-exported functions within this script.*

Description

This function is run following the KNN computation and respective cell grouping. The function also contains a progress ticker that allows one to determine how much time left in this function.

Usage

```
SconeValues(nn.matrix, cell.data, scone.markers, unstim, threshold = 0.05,
  fold = "median", stat.test = "mwu", multiple.donor.compare = FALSE)
```

Arguments

`nn.matrix`	a matrix of cell index by nearest neighbor index, with values being cell index of the nth nearest neighbor
`cell.data`	tibble of cells by features
`scone.markers`	vector of all markers to be interrogated via statistical testing
`unstim`	an object (used so far: string, number) specifying the "basal" condition

threshold	a number indicating the p value the raw change should be thresholded by.
fold	a string that specifies the use of "median" or "mean" when calculating fold change
stat.test	string denoting Mann Whitney U test ("mwu") or T test ("t")
multiple.donor.compare	a boolean that indicates whether t test across multiple donors should be done

Value

result: tibble of raw changes and p values for each feature of interest, and fraction of cells with condition 2

Examples

```
ex.nn <- Fnn(wand.combined, input.markers)
SconeValues(ex.nn, wand.combined, funct.markers, "basal")
```

SplitFile *Runs "process.multiple.files" on a single file, splits it randomly, and presends half of it is "unstim" and half of it is "stim"*

Description

This is meant to serve as a control for the basic "unstim" and "stim" pipeline that is generally used within this package. If phosphoproteins are being compared across conditions, for example, then there should be no difference in the case that you split the same file and compare the two halves.

Usage

```
SplitFile(file, transform = "asinh", numcells = 10000, norm = FALSE,
  scale = FALSE, input.markers)
```

Arguments

file	the file we're going to split
transform	if set to asinh, performs asinh transformation on all markers of the dataset
numcells	the number of total cells to be subsampled to, set at 10k for default
norm	boolean of whether data should be quantile normalized
scale	boolean of whether data should be z-scored
input.markers	vector of strings indicating the markers to be used as input

Value

tibble containing original markers and all values calculated by SCONE

Examples

```
file <- system.file("extdata", "Bendall_et_al_Cell_Sample_C_basal.fcs",
    package = "Sconify")
SplitFile(file, input.markers = input.markers)
```

StringToNumbers *Transform strings to numbers.*

Description

Takes a vector of strings and outputs simple numbers. This takes care of the case where conditions are listed as strings (basal, IL7), in which case they are converted to numbers (1, 2)

Usage

```
StringToNumbers(strings)
```

Arguments

strings	vector of strings

Value

strings: same vector with each unique element converted to a number

Examples

```
ex.string <- c("unstim", "unstim", "stim", "stim", "stim")
StringToNumbers(ex.string)
```

SubsampleAndTsne *Subsample data and run tSNE*

Description

A wrapper for Rtsne that takes final SCONE output, and runs tSNE on it after subsampling. This is specifically for SCONE runs that contain large numbers of cells that tSNE would either be too time-consuming or messy for. Regarding the latter, tSNE typically appears less clean in the range of 10^5 cells

Usage

```
SubsampleAndTsne(dat, input, numcells)
```

Arguments

dat	tibble of original input data, and scone-based additions.
input	the markers used in the original knn computation, which are typically surface markers
numcells	the number of cells to be downsampled to

Value

a subsampled tibble that contains tSNE values

Examples

```
SubsampleAndTsne(wand.combined, input.markers, 500)
```

`TsneVis` *Plot a tSNE map colored by a marker of interest*

Description

Wrapper for ggplot2 based plotting of a tSNE map to color by markers from the post-processed file if tSNE was set to TRUE in the post-processing function.

Usage

```
TsneVis(final, marker, label = marker)
```

Arguments

`final`	The tibble of cells by features outputted from the post.processing function. These features encompass both regular markers from the original data and the KNN statistics processed markers
`marker`	String that matches the marker name in the final data object exactly.
`label`	a string that indicates the name of the color label in the ensuing plot. Set to the marker string as default.

Value

A plot of bh-SNE1 x bh-SNE2 colored by the specified marker.

Examples

```
TsneVis(wand.final, "pSTAT5(Nd150)Di.IL7.change", "pSTAT5 change")
```

`wand.combined` *Wanderlust data combined basal and IL7 cells*

Description

A single patient pair of basal and IL7 treated cells from bone marrow gated for B cell precursors.

Usage

```
wand.combined
```

Format

A tibble of 1000 cells by 51 features, including all the input markers, Wanderlust values, and the condition. The first 500 rows are untreated cells and the last 500 rows are IL7 treated.

wand.final — *Post-scone output of the "combiend" Wanderlust data.*

Description

"combined" data taken through KNN generation and comparisons, along with t-SNE map generation.

Usage

```
wand.final
```

Format

A tibble of 1000 cells and 87 feaures, including the input features, the SCONE-generated comparisons, differential abundance, and two t-SNE dimesnions

wand.ideal.k — *A named vector to help the user determine the ideal k for the Wanderlust dataset.*

Description

This is the output of the impute.testing function used on the Wanderlust dataset, which finds the avergae imputation error of all signal markers imputed from KNN of surface markers.

Usage

```
wand.ideal.k
```

Format

A named vector, where the elements are averge imputation error and the names are the values of from a 10,000 cell dataset.

wand.il7 — *Wanderlust IL7 data*

Description

The IL7 treated cells from a single patient in the Wanderlust dataset

Usage

```
wand.il7
```

Format

A tibble of 1000 cells by 51 features. All markers in the dataset, along with pre-calculated Wanderlust value and condition, which is a string that denotes that this is the "IL7" condition for each row. Important when this is concatenated with additional conditions

wand.scone *Wanderlust scone output*

Description

The scone output for the Wanderlust dataset

Usage

```
wand.scone
```

Format

A tibble of 1000 cells by 34 features. These features include the KNN comparisons, KNN density estimation, and differential abundance. Note that this tibble gets concatenated with the original tibble, as well as two t-SNE dimensions in the post.processing() command of the pipeline.

Index