Package: Spectre (via r-universe)

October 4, 2026

Type Package

Title High-dimensional cytometry and imaging analysis

Version 1.3.0

Date 2025-08-22

Description A computational toolkit in R for the integration, exploration, and analysis of high-dimensional single-cell cytometry and imaging data.

License MIT + file LICENSE

URL https://github.com/ImmuneDynamics/Spectre, http://immunedynamics.io/Spectre/

BugReports https://github.com/ImmuneDynamics/Spectre/issues

Depends R (>= 3.6.0)

Imports colorRamps, data.table, dendsort, factoextra, flowCore, FlowSOM, fs, ggplot2, ggpointdensity, ggpubr, ggrepel, ggthemes, gridExtra, gtools, irlba, lifecycle, parallel, patchwork, pheatmap, RColorBrewer, rstudioapi, rsvd, Rtsne, scales, scattermore, umap, uwot, viridis

Suggests Biobase, caret, CytoNorm, devtools, EnhancedVolcano, exactextractr, FNN, formatR, harmony, HDF5Array, knitr, limma, qs, raster, Rcpp, rhdf5, rmarkdown, Rphenograph, s2, Seurat, SeuratObject, sf, SingleCellExperiment, sp, stars, testthat (>= 3.0.0), tiff, usethis

Remotes saeyslab/CytoNorm, JinmiaoChenLab/Rphenograph

Config/testthat/edition 3

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

VignetteBuilder knitr

Config/pak/sysreqs cmake libfontconfig1-dev libfreetype6-dev libglpk-dev make libicu-dev libpng-dev libuv1-dev libxml2-dev libssl-dev python3 zlib1g-dev

Repository https://agshaas.r-universe.dev

Date/Publication 2025-09-19 00:41:01 UTC

RemoteUrl https://github.com/immunedynamics/spectre

RemoteRef HEAD

RemoteSha 159dc9f6d700b0dbd9fed8677cd94521c661691e

Contents

adjust.files . . . 3
check.files . . . 4
create.dt . . . 4
create.stats . . . 5
create.sumtable . . . 6
demo.clustered . . . 7
do.add.cols . . . 8
do.add.masks . . . 9
do.add.percell . . . 10
do.aggregate . . . 10
do.align . . . 11
do.asinh . . . 12
do.calculate.area . . . 13
do.clip . . . 13
do.combine.cols . . . 14
do.create.outlines . . . 15
do.extract . . . 15
do.extract.cell.dat . . . 16
do.filter . . . 16
do.filter.percell . . . 17
do.label.from.polygon . . . 17
do.list.summary . . . 17
do.list.switch . . . 18
do.logicle . . . 19
do.merge.files . . . 20
do.pull.data . . . 21
do.reorder . . . 21
do.rescale . . . 22
do.stats . . . 22
do.subsample . . . 23
do.zscore . . . 24
fast.colour.plot . . . 24
fast.multi.plot . . . 27
make.autograph . . . 29
make.colour.plot . . . 30
make.multi.plot . . . 32

make.pheatmap . . . 34
make.spatial.plot . . . 36
make.volcano.plot . . . 38
package.check . . . 38
package.install . . . 39
package.load . . . 40
prep.cytonorm . . . 41
read.files . . . 42
read.spatial.files . . . 43
run.align . . . 44
run.cytonorm . . . 45
run.fitsne . . . 46
run.flowsom . . . 48
run.harmony . . . 50
run.knn.classifier . . . 51
run.pca . . . 52
run.phenograph . . . 55
run.rpca . . . 56
run.spatial.analysis . . . 57
run.tsne . . . 58
run.umap . . . 59
spatial-class . . . 61
train.cytonorm . . . 62
train.knn.classifier . . . 63
write.files . . . 64
write.hdf5 . . . 65

Index **67**

`adjust.files` *adjust.files*

Description

adjust.files

Usage

```
adjust.files(dat, colnames.table)
```

Arguments

`dat`	List of data.tables, typically one per 'sample'
`colnames.table`	A data.table of column names. First column contains the current names, second column contains the new names to update to.

`check.files` *check.files*

Description

check.files

Usage

```
check.files(dat)
```

Arguments

`dat`	List of data.tables, typically one per 'sample'

`create.dt` *create.dt - convert a Seurat or SingleCellExperiment to a data.table*

Description

This function converts a Seurat or SingleCellExperiment object into a list containing a data.table, with vectors of gene and dimensionality reduction

Usage

```
create.dt(dat, from)
```

Arguments

`dat`	NO DEFAULT. A Seurat or SingleCellExperiment object.
`from`	DEFAULT = NULL. By default, the class of object will be detected automatically, but this can be overwritten using from. Can be from = 'Seurat' or 'SingleCellExperiment'.

Author(s)

Thomas M Ashhurst, <thomas.ashhurst@sydney.edu.au>

References

https://github.com/ImmuneDynamics/Spectre.

create.stats *create.stats - perform statistical analysis and generate an output data.table*

Description

This function performs statistical analysis and generates an output data.table

Usage

```
create.stats(
  dat,
  use.cols,
  sample.col,
  group.col,
  comparisons,
  variance.test = "kruskal.test",
  pairwise.test = "wilcox.test",
  corrections = "fdr"
)
```

Arguments

dat	NO DEFAULT. A data.table containing summary data (samples vs features)
use.cols	NO DEFAULT. Columns (i.e. features) to use for statistical analysis.
sample.col	NO DEFAULT. The column denoting sample names.
group.col	NO DEFAULT. The column denoting experimental group names.
comparisons	NO DEFAULT. A list of pairwise comparisons between samples. E.g. list(c('Group A', 'Group B'), c('Group B', 'Group C')).
variance.test	DEFAULT = 'kruskal.test'. Currently the only option
pairwise.test	DEFAULT = 'wilcox.test'. Currently the only option
corrections	DEFAULT = 'fdr' (false discovery rate). Currently the only option

Author(s)

Thomas M Ashhurst, <thomas.ashhurst@sydney.edu.au>

References

https://github.com/ImmuneDynamics/Spectre.

`create.sumtable` *create.sumtable - create a data.table 'summarising' cellular data by sample and population/cluster.*

Description

This function summarises cellular data and generates a summary data.table

Usage

```
create.sumtable(dat, sample.col, pop.col, use.cols, annot.cols, counts, func, sep)
```

Arguments

`dat`	NO DEFAULT. A data.table containing cells (rows) vs features/markers (columns). One column must represent sample names, and another must represent populations/clusters.
`sample.col`	NO DEFAULT. Character. Name of the sample column (e.g. "Sample").
`pop.col`	NO DEFAULT. Character. Name of the population/cluster column (e.g. "Population", "Cluster").
`use.cols`	DEFAULT = NULL A character vector indicating the columns to be measured (e.g. cellular columns – c("CD45", "CD3e") etc).
`annot.cols`	DEFAULT = NULL. A character vector indicating the columns to be included as annotation columns (e.g. c("Batch", "Group") etc).
`parent.col`	DEFAULT = NULL. A character entry indicating a column that represents the 'lineage' each population belongs to (e.g. 'CD4 T cells' may belong to the 'T cells' lineage). Use this to also calculate each population as a percentage of lineage.
`counts`	DEFAULT = NULL. If you wish to calculate the actual number of cells per sample, a data.frame or data.table containing the sample names (in column 1) and cell counts per sample (column 2).
`perc.pos`	DEFAULT = NULL. If you wish to calculate the percentage of each population that is 'positive' for a marker, you can provide a data.table containing the mark names (in column 1) and cut off values for positivity (column 2).
`double.pos`	DEDAULT = NULL. List of vectors, each vector containing the names of multiple markers you wish to calculate % positive for (e.g. CD38+HLADR+). Generates 'and' and 'or' combinations.
`func`	DEFAULT = "median". Can be "median" or "mean". Defines the type of function for calculating MFI data.
`sep`	DEFAULT = " – ". Character separation of the measurement type and the population (e.g. MFI of CD4 – T cells)

Author(s)

Thomas M Ashhurst, <thomas.ashhurst@sydney.edu.au>

References

https://github.com/ImmuneDynamics/Spectre.

Examples

```
## Calculate and export results from demonstration data
dat <- Spectre::demo.clustered
counts <- data.frame('Sample' = unique(dat[['Sample']]),
'Counts' = c(rep(100000, 6), rep(1000000, 6)))

sum.dat <- create.sumtable(dat = dat,
                           sample.col = "Sample",
                           pop.col = "Population",
                           use.cols = names(dat)[c(11:19)],
                           counts = counts)
```

`demo.clustered` *demo.clustered - Demo data.table with cells (rows) x markers (columns), with clusters, populations, and UMAP values added*

Description

demo.clustered - Demo data.table with cells (rows) x markers (columns), with clusters, populations, and UMAP values added

Usage

```
demo.clustered
```

Format

Demo data.table with cells (rows) x markers (columns), with clusters, populations, and UMAP values added.

Author(s)

Thomas M Ashhurst, <thomas.ashhurst@sydney.edu.au>

Source

Thomas M Ashhurst.

References

https://github.com/ImmuneDynamics/Spectre.

Examples

```
demo.clustered
```

`do.add.cols` *do.add.cols*

Description

do.add.cols

Usage

```
do.add.cols(dat, base.col, add.dat, add.by, rmv.ext)
```

Arguments

`dat`	NO DEFAULT. A data.table (or data.frame) containing the data to have new values added to.
`base.col`	NO DEFAULT. Name of the column containing values that new values will be matched to.
`add.dat`	NO DEFAULT. A data table of new values to embed as a new columns, with one column containing values used for matching with the target data.table.
`add.by`	NO DEFAULT. Character, name of the column in add.dat that is used for matching to the taret dataset.
`rmv.ext`	DEFAULTS TO FALSE. Logical, can be TRUE or FALSE. Removes a ".csv" or ".fcs" extension from a the 'match.to' vector – especially useful if 'match.to' is a list of sample names that end in .csv or .fcs.
`mem.ctrl`	DEFAULT = TRUE. Runs gc() (garbage collection) after a number of steps to free up memory that hasn't been released quickly enough.

Value

Returns the data.table with new columns embedded.

Author(s)

Thomas M Ashhurst, <thomas.ashhurst@sydney.edu.au>

References

https://github.com/ImmuneDynamics/Spectre.

Examples

```
library(data.table)
dat <- Spectre::demo.clustered[,1:19]

add.dt <- data.table('Files' = unique(dat$FileName),
                     'SampleNumber' = c(1:12))

cell.dat <- do.add.cols(dat = dat,
                        base.col = "FileName",
                        add.dat = add.dt,
                        add.by = "Files")
```

`do.add.masks` *do.add.masks*

Description

do.add.masks

Usage

```
do.add.masks(dat, mask.dir, mask.pattern, mask.label,correct.extent = TRUE,
flip.y = TRUE, value.modifier = 65535, HDF5 = FALSE, array = FALSE)
```

Arguments

`dat`	NO DEFAULT. A list of spatial data objects
`mask.dir`	NO DEFAULT. Directory of mask files.
`mask.pattern`	NO DEFAULT. A character pattern that identifies the mask type (e.g. '_mask')
`mask.label`	NO DEFAULT. What do you want to call the mask
`correct.extent`	DEFAULT = TRUE
`flip.y`	DEFAULT = TRUE
`value.modifier`	DEFAULT = 65535
`HDF5`	DEFAULT = FALSE. Can read in HDF5 mask files if desired (advanced use).
`array`	DEFAULT = FALSE. If the mask TIFF is an array (advanced use).

Author(s)

Thomas M Ashhurst, <thomas.ashhurst@sydney.edu.au>

References

https://github.com/ImmuneDynamics/Spectre.

do.add.percell *do.add.percell*

Description

do.add.percell

Usage

```
do.add.percell(spatial.dat, percell.dat, roi.col, name = "per.cell")
```

Author(s)

Thomas M Ashhurst, <thomas.ashhurst@sydney.edu.au>

References

https://github.com/ImmuneDynamics/Spectre.

do.aggregate *Aggregate data using data.table functions – summarises the mean, median, or sum expression of each marker on each cluster/population*

Description

Aggregate data using data.table functions – summarises the mean, median, or sum expression of each marker on each cluster/population

Usage

```
do.aggregate(dat, use.cols, by, func, dt)
```

Arguments

dat	NO DEFAULT. A data.table (or data.frame) containing the data to be aggregated.
use.cols	NO DEFAULT. The columns to be used for aggregation.
by	NO DEFAULT. The column that contains the cluster/population identities.
func	DEFAULT = 'median'. Character, method of aggregation. Can be 'median', 'mean', or 'sum'.
dt	DEFAULT = TRUE. If TRUE returns data.table with observations and the use.cols, if false returns data.frame with use.cols, and rownames as the observations

Value

Returns a new data.table with values of use.cols aggregated per cluster/population.

Author(s)

Thomas M Ashhurst, <thomas.ashhurst@sydney.edu.au>

References

https://github.com/ImmuneDynamics/Spectre.

Examples

```
exp <- do.aggregate(dat = Spectre::demo.clustered,
                    use.cols = names(demo.clustered)[c(11:19)],
                    by = "Population")

# Typically followed up by making an expression heatmap
Spectre::make.pheatmap(dat = exp,
                       file.name = "Pheatmap.png",
                       plot.title = "Pheatmap",
                       sample.col = "Population",
                       plot.cols = names(exp)[c(2:10)])
```

---

do.align *do.align - Function to align multiple batches of a dataset.*

Description

This function allows you to align multiple batches of a dataset, using either a "Quantile" approach, or using "CytoNorm"

Usage

```
do.align(ref.dat, target.dat, batch.col, align.cols, method, goal,
nQ, Qmin, Qmax, write.ref.fcs, write.target.fcs)

do.align(ref.dat, target.dat, batch.col, align.cols, method = "CytoNorm",
goal = "mean", nQ = 101, Qmin = 0.01, Qmax = 0.99, write.ref.fcs = TRUE,
write.target.fcs = TRUE, mem.ctrl = TRUE)
```

Arguments

ref.dat	NO DEFAULT. If using method = "CytoNorm", this must be a FlowSOM object created by Spectre::do.prep.fsom. If using "Quantiles", then this must be
target.dat	NO DEFAULT. A data.table of data you wish to align
batch.col	NO DEFAULT. Character, column that denotes batches

`align.cols`	NO DEFAULT. Character, a vector of columns that you wish to align.
`method`	DEFAULT = "CytoNorm". Character, can be "CytoNorm" or "Quantile".
`goal`	DEFAULT = "Mean". For method = "CytoNorm". Character, the goal to align to. Can either be "mean", or can be the name of one of the batches (e.g. "Batch1").
`nQ`	DEFAULT = 101. For method = "CytoNorm". Numeric, the number of quanitles to use.
`Qmin`	DEFAULT = 0.01. For method = "Quantiles". Numeric, the minimum valus for the lower threshold.
`Qmax`	DEFAULT = 0.99. For method = "Quantiles". Numeric, the maximum value for the upper threshold.
`write.ref.fcs`	DEFAULT = TRUE. Logical, do you want a quick copy of the reference sample FCS files.
`write.target.fcs`	DEFAULT = TRUE. Logical, do you want a quick copy of the ALIGNED target sample FCS files.
`mem.ctrl`	DEFAULT = TRUE. Runs gc() (garbage collection) after a number of steps to free up memory that hasn't been released quickly enough.

Value

Returns a data.table with selected columns (align.cols) replaced with aligned data.

Author(s)

Thomas M Ashhurst, <thomas.ashhurst@sydney.edu.au>

References

Ashhurst, T. M., et al. (2019). https://www.ncbi.nlm.nih.gov/pubmed/31077106

`do.asinh` *Transform data using ArcSinh transformation*

Description

Function to transform data in selected columns using ArcSinh transformation with a specified co-factor.

Usage

```
do.asinh(dat, use.cols, cofactor=5, append.cf=FALSE,
reduce.noise=FALSE)
```

Arguments

dat	NO DEFAULT. data.table object. Input data.
use.cols	NO DEFAULT. Vector of character column names. The columns to transform. These columns will be transformed and added to the data.table as new columns.
cofactor	DEFAULT = 5. Co-factor to use for arcsinh transformation. Can be list of co-factors that matches the columns in use.cols. If specifying a list of co-factors make sure each column in use.cols is assigned a co-factor (repeat values if necessary).
append.cf	DEFAULT = FALSE. Appends the co-factor used to the end of the name of the transformed columns.
reduce.noise	DEFAULT = FALSE. This is an experimental calculation which should reduce noise from negative values. Use with caution.

Value

A data.table with new columns added, that contain asinh transformed data.

---

do.calculate.area *do.calculate.area*

Description

do.calculate.area

Usage

```
do.calculate.area(dat, region = NULL)
```

Arguments

dat	NO DEFAULT. Spatial data list
region	DEFAULT = NULL. Name of the region mask, if present.

---

do.clip *do.clip - Clips data using a specified lower and upper value*

Description

This function allows you to clip data using specified lower and upper values. E.g. if the upper value is set to 1000, then any values above 1000 will be converted to 1000. No rows are lost, rather the values in those rows are converted.

Usage

```
do.clip(dat, use.cols, min.value = NULL, max.value = NULL, append.name = '_clipped')
```

Arguments

dat	NO DEFAULT. data.table Input sample.
use.cols	NO DEFAULT. Columns to clip
min.value	DEFAULT = NULL. Specify the value for which values below will be converted to. If set to NULL, no clipping will occur.
max.value	DEFAULT = NULL. Specify the value for which values above will be converted to. If set to NULL, no clipping will occur.
append.name	DEFAULT = '_clipped'. Text appended to new column names after clipping.

Value

A data.table with new columns added, that contain the clipped data.

References

https://sydneycytometry.org.au/spectre

See Also

https://sydneycytometry.org.au/spectre for usage instructions and vignettes.

---

do.combine.cols *do.combine.cols - Combine the values of two columns into a single column*

Description

This function allows you to combine the values of two columns, with an option to remove any 'NA' values

Usage

```
do.combine.cols(dat, col1, col2, na.rm)
```

Arguments

dat	NO DEFAULT. Data.table.
col1	NO DEFAULT. Name of the column you want to combine into.
col2	NO DEFAULT. Name of the column you want to combine from. This column will be deleted after combination into col1.
na.rm	DEFAULT = TRUE. If NA values are present in the columns, they will be removed.

Value

Returns a data.table with adjusted columns

Author(s)

Thomas M Ashhurst, <thomas.ashhurst@sydney.edu.au>

`do.create.outlines` *do.create.outlines*

Description

do.create.outlines

Usage

```
do.create.outlines(dat, mask.name, method = 'stars')
```

Arguments

`mask.name`	NO DEFAULT. Name of the mask to create outlines for
`method`	DEFAULT = 'stars'. Can be 'stars' or 'raster'
`spatial.dat`	NO DEFAULT. Spatial data list

`do.extract` *do.extract*

Description

do.extract

Usage

```
do.extract(dat, mask, name = "CellData", fun = "mean")
```

Arguments

`dat`	NO DEFAULT. List of spatial data objects
`mask`	NO DEFAULT. Name of mask to use for cell data extraction
`name`	DEFAULT = 'CellData'. Name of the extracted cell dataset.
`fun`	DEFAULT = 'mean'. Summarisation function.

`do.extract.cell.dat` *do.extract.cell.dat*

Description

do.extract.cell.dat

Usage

```
do.extract.cell.dat(spatial.dat, target.dat)
```

`do.filter` *do.filter - filtering data.table using multiple match values*

Description

This function allows filtering of a data.table using multiple match values – all cells that contain any of the match values will be filtered, and provided in a new data.table.

Usage

```
do.filter(dat, use.col, values)
```

Arguments

`dat`	NO DEFAULT. A data.table
`use.col`	DEFAULT = NULL. The column to use for re-ordering
`values`	DEFAULT = NULL. A vector of values to use for filtering – all cells that contain any of the match values will be filtered, and provided in a new data.table. Up to 20 match values can be provided

Author(s)

Thomas M Ashhurst, <thomas.ashhurst@sydney.edu.au>

References

https://github.com/ImmuneDynamics/Spectre.

do.filter.percell *do.filter.percell*

Description

do.filter.percell

Usage

```
do.filter.percell(spatial.dat, per.cell, to, filter.by,
id.col = "ObjectNumber", x.col = "Location_Center_X", y.col = "Location_Center_Y",
simplify.cp.colname = TRUE, value.modifier = 65535)
```

do.label.from.polygon *do.label.from.polygon*

Description

do.label.from.polygon

Usage

```
do.label.from.polygon(spatial.dat, cell.dat, mask, labels,
name = "Label", id.col = 'ID', roi.col = 'ROI')
```

do.list.summary *Summarise the column names and row numbers for elements of a list*

Description

Summarise the column names and row numbers for elements of a list

Usage

```
do.list.summary(dat)
```

Arguments

dat	NO DEFAULT. A list of data.tables that you wish to create some summary information for

Value

Returns a new list summarising the data.tables in your list.

Author(s)

Thomas M Ashhurst, <thomas.ashhurst@sydney.edu.au>

References

https://sydneycytometry.org.au/spectre.

---

do.list.switch *List switch or inversion*

Description

List switch or inversion

Usage

```
do.list.switch(dat)
```

Arguments

dat	NO DEFAULT. A list of vectors that you wish to switch/invert

Value

Returns a data.table of the switched/inverted li

Author(s)

Thomas M Ashhurst, <thomas.ashhurst@sydney.edu.au>

References

https://sydneycytometry.org.au/spectre.

Examples

```
lst <- list("A" = c(1,2,3), "B" = c(4,5,6), "C" = c(7,8,9))
lst

res <- do.list.switch(lst)
res
```

`do.logicle` *Using logicle to transform data*

Description

The function transform data using logicle transformation. It makes use of flowCore's implementation of logicle transformation, and can either automatically infer the transformation function based on your data, or calculate it based on what you want by looking at the default or overriden `linearisation.width`, `max.scale.val`, `full.transform.width`, `additional.negative.range`. Description of some of the parameter is adapted from flowCore's vignette. For more information on what logicle transformation does, please read the manuscript in the references section.

Usage

```
do.logicle(dat, use.cols, linearisation.width, max.scale.val,
full.transform.width, additional.negative.range)

do.logicle(dat, use.cols, auto.infer.function = TRUE,
linearisation.width = 1.2, max.scale.val = 262144, full.transform.width = 4.5,
additional.negative.range = 0)
```

Arguments

`dat`	NO DEFAULT. data.table. Input data
`use.cols`	NO DEFAULT. Vector of column names to transform
`auto.infer.function`	Default = TRUE. Automatically infer the logical transformation function based on your data. If this is set to FALSE, default or overriden `linearisation.width`, `max.scale.val`, `full.transform.width`, `additional.negative.range` will be used instead to calculate the logical transformation function.
`linearisation.width`	Default = 1.2. Linearisation width in asymptotic decades. This must be > 0 and determines the slope of transformation at zero. We found 1.2 works well for flow data, and 0.5 works well for cytof data. It can be estimated using equation: $(m - log10(max.scale.val/\|r\|))/2$ where r is the most negative value to included in the display. See `max.scale.val` below.
`max.scale.val`	Default = 262144. Maximum scale data value. It can be 10,000 for common 4 decade data or 262144 (the default value) for 18 bit data range. This must be greater than 0
`full.transform.width`	Default = 4.5. The full width of the transformed values in asymptotic decades This must be greater than 0
`additional.negative.range`	Default = 0. Additional negative range to be included in the transformed value in asymptotic decades. Value greater than 0 will bring the prescribed additional range into the transformed values

References

Parks, David R., Mario Roederer, and Wayne A. Moore. "A new “Logicle” display method avoids deceptive effects of logarithmic scaling for low signals and compensated data." Cytometry Part A: The Journal of the International Society for Analytical Cytology 69.6 (2006): 541-551.

Examples

```
library(Spectre)
dat = Spectre::demo.clustered[, 2:9]
use.cols = names(dat)
do.logicle(dat, use.cols = use.cols)
```

`do.merge.files` *do.merge.files - Function to merge a list of data.tables (one data.table per 'sample') into a single large data.table.*

Description

do.merge.files - Function to merge a list of data.tables (one data.table per 'sample') into a single large data.table.

Usage

```
do.merge.files(dat, remove.duplicates)

do.merge.files(dat, remove.duplicates = TRUE)
```

Arguments

`dat`	NO DEFAULT. List of data.tables (or data.frames)
`remove.duplicates`	DEFAULT = TRUE. Do you want to remove duplicates?

Value

Returns a combined data.table.

Author(s)

Thomas M Ashhurst, <thomas.ashhurst@sydney.edu.au>

References

https://sydneycytometry.org.au/spectre.

Examples

```
data.list <- list(
"01_Mock_01" = Spectre::demo.clustered[Sample == '01_Mock_01'],
"02_Mock_02" = Spectre::demo.clustered[Sample == '02_Mock_02']
)
cell.dat <- do.merge.files(dat = data.list, remove.duplicates = TRUE)
head(cell.dat)
```

---

do.pull.data *do.pull.data*

Description

do.pull.data

Usage

```
do.pull.data(spatial.dat, target.dat)
```

Arguments

target.dat	NO DEFAULT. Dataset to pull.
dat	NO DEFAULT. Spatial data list.

---

do.reorder *do.reorder - custom row re-ordering of a data.table*

Description

This function allows for a user-defined re-ordering of data.table rows

Usage

```
do.reorder(dat, use.col, new.order)
```

Arguments

dat	NO DEFAULT. A data.table
use.col	DEFAULT = NULL. The column to use for re-ordering
new.order	DEFAULT = NULL. A vector of row values for the specified column, in the desired order

Author(s)

Thomas M Ashhurst, <thomas.ashhurst@sydney.edu.au>

References

https://github.com/ImmuneDynamics/Spectre.

`do.rescale` *do.rescale - Re-scale data in selected columns between two values, usually 0 and 1.*

Description

This function allows you to re-scale the data in selected columns between two values, usually 0 and 1.

Usage

```
do.rescale(dat, use.cols, new.min = 0, new.max = 1,
append.name = '_rescaled')
```

Arguments

`dat`	NO DEFAULT. data.table Input sample.
`use.cols`	NO DEFAULT. Vector of character column names – these columns will be normalised and added to the data.table as new columns.
`new.min`	DEFAULT = 0. The new minimum value.
`new.max`	DEFAULT = 1. The new maximum value.
`append.name`	DEFAULT = '_rescaled'. Text to be appended to the column names of re-scaled data.

Value

A data.table with new columns added, that contain the re-scaled data.

References

https://sydneycytometry.org.au/spectre

See Also

https://sydneycytometry.org.au/spectre for usage instructions and vignettes.

`do.stats` *Statistical analysis*

Description

Statistical analysis

`do.subsample` *Subsample data*

Description

Method to subsample data. Can subsample by randomly selecting a desired number of cells from all samples (DEFAULT), subsample by specifying the exact number of cells for each sample (specify divide.by), or by subsampling the same number of cells from each sample based on the sample with the lowest count (specify divide.by and min.per). Useful to decrease total cells for generating dimensionality reduction plots (tSNE/UMAP).

Usage

```
do.subsample(dat, targets, samp.col, min.per, seed)
```

Arguments

`dat`	NO DEFAULT. Input dataframe with cells (rows) vs markers (columns).
`targets`	NO DEFAULT. Vector of downsample targets. If divide.by is specified, then must be a vector of subsample targets in the same order as the unique divide.by entries (e.g. unique(dat[divide.by])). Can also provide as a data.table or data.frame where the first column is the unique entries in the divide.by argument (i.e. unique(dat[divide.by])), and the second column should be the targets. In this case, does not have to be in the order they appear in the dataset, but the 'divide.by' argument must be set.
`divide.by`	DEFAULT = NULL. Character. Name of the column that reflects groupings of cells (sample names, group names etc) if you want to subsample by each.
`min.per`	DEFAULT = FALSE. If TRUE, and divide.by is specified, each sample contributes the same amount of data based on sample with lowest count.
`seed`	DEFAULT = 42. Numeric. Seed for reproducibility.

Author(s)

Thomas Ashhurst, <thomas.ashhurst@sydney.edu.au> Felix Marsh-Wakefield, <felix.marsh-wakefield@sydney.edu.

Examples

```
# Subsample 10,000 cells randomly from the total dataset
sub.dat <- Spectre::do.subsample(dat = Spectre::demo.clustered,
                                 targets = 10000)

# Subsample based on the sample with the smallest number of cells
sub.dat.sample <- Spectre::do.subsample(dat = Spectre::demo.clustered,
                                        divide.by = "FileName",
                                        min.per = TRUE)
```

do.zscore *do.zscore - Calculate z.score for a dataset*

Description

do.zscore - Calculate z.score for a dataset

Usage

```
do.zscore(dat, use.cols)
```

Arguments

dat	NO DEFAULT. A data.table (or data.frame) containing the data to be converted to z-scores. Z-score transformed values will be added as new columns.
use.cols	NO DEFAULT. The columns to be used for z-score calculations.
append.name	DEFAULT = '_zscore'. Text to be appended to the end of the new z-score transformed columns.
replace	DEFAULT = FALSE. If FALSE, appends new columns to the data.table. If TRUE, replaces the values in the existing columns with the z-score tranformed values, and does not change the column names.

Value

Returns a new data.table with z-score calculations for each selected column

Examples

```
do.zscore(dat = Spectre::demo.clustered, use.cols = c("NK11", "CD4"))
```

fast.colour.plot *DEPRECATED fast colour plot*

Description

Please use `make.colour.plot(fast = TRUE)` instead.

Usage

```
fast.colour.plot(
  dat,
  x.axis,
  y.axis,
  col.axis = NULL,
  col.type = "continuous",
  add.label = FALSE,
  hex = FALSE,
  hex.bins = 30,
  colours = "spectral",
  col.min.threshold = 0.01,
  col.max.threshold = 0.995,
  align.xy.by = dat,
  align.col.by = dat,
  regression.line = NULL,
  title = col.axis,
  filename = NULL,
  dot.size = 1,
  plot.width = 9,
  plot.height = 7,
  nudge_x = 0.5,
  nudge_y = 0.5,
  square = TRUE,
  legend.loc = "right",
  save.to.disk = TRUE,
  path = getwd(),
  blank.axis = FALSE,
  col.tab = NULL
)
```

Arguments

dat	NO DEFAULT. data.table Input sample.
x.axis	NO DEFAULT. Character. Column for X axis.
y.axis	NO DEFAULT. Character. Column for Y axis.
col.axis	DEFAULT = NULL. If not specified, plot is coloured by density. If you provide a character name of a column (e.g. "BV605.Ly6C", "Group", "FlowSOM_metacluster" etc), then each point will be coloured by the value in that column.
col.type	DEFAULT = "continuous". Can also be "factor".
add.label	DEFAULT = FALSE. Adds labels on the plot at the centroid of each factor. Only works if col.type = "factor".
hex	DEFAULT = FALSE. Whether to split the data into bins and show the average expression of the bin. Currently only works when specifying col.axis, so does not work with density plots.

hex.bins	DEFAULT = 30. Number of bins to split into. Only used if hex is TRUE.
colours	DEFAULT = "spectral". Only used if type = 'colour', ignored if type = 'factor'. Specify a colour scheme. Can be "jet", "spectral", "viridis", "inferno", "magma", or "BuPu". Can also provide a data.table with the first column being the factors, and the second column being the corresponding colour code.
col.min.threshold	DEFAULT = 0.01. Numeric. Define minimum threshold for colour scale. Values below this limit will be coloured as the chosen minimum threshold.
col.max.threshold	DEFAULT = 0.995 Numeric. Define maximum threshold for colour scale. Values above this limit will be coloured as the chosen maximum threshold.
align.xy.by	DEFAULT = dat. data.table Sample to use to determine minimum and maximum X and Y axis values.
align.col.by	DEFAULT = dat. data.table. Sample to use to determine minimum and maximum colour values.
regression.line	DEFAULT = NULL. Regression line to add to show the trend of the data. This will be passed on to ggplot2::geom_smooth function, and accepts either NULL or a character vector, e.g. "lm", "glm", "gam", "loess" or a function.
title	DEFAULT = col.axis. Character. Title for the plot.
filename	DEFAULT = NULL. Character. The name of the file to save the plot to.
dot.size	DEFAULT = 1. Numeric. Size of the dots.
plot.width	DEFAULT = 9. Width of the ggplot when saved to disk.
plot.height	DEFAULT = 7. Height of the ggplot when saved to disk.
nudge_x	DEFAULT = 0.5. When add.label = TRUE, distance the label is shifted from the centroid point on the X axis.
nudge_y	DEFAULT = 0.5. When add.label = TRUE, distance the label is shifted from the centroid point on the Y axis.
square	DEFAULT = TRUE. Ensures the plot is saved as a square. Set to FALSE if you want a plot with different X and Y lengths.
legend.loc	DEFAULT = 'right'. By default plot legends will be on the right hand side. Can specify the legend location to "bottom" if desired, or 'none' to remove it entirely.
save.to.disk	DEFAULT = TRUE. Will save the ggplot to disk. If FALSE, will only show the ggplot.
path	DEFAULT = getwd(). The location to save your ggplot. By default, will save to current working directory. Can be overidden.
blank.axis	DEFAULT = FALSE Logical, do you want a minimalist graph?

Details

Create a FAST dot plot (X vs Y) coloured by a selected continuous (e.g. marker expression) or factorial (e..g. cluster, group) column.

This function allows you to create a coloured XY plot where each cell is coloured by a selected column. Typically used to plot cells on tSNE1/2 or UMAP1/2 coloured by select cellular markers or clusters, samples, groups etc.

Author(s)

Thomas M Ashhurst, <thomas.ashhurst@sydney.edu.au> Givanna Putri

References

https://sydneycytometry.org.au/spectre

See Also

https://sydneycytometry.org.au/spectre for usage instructions and vignettes.

Examples

```
# Load packages
library(Spectre)
cell.dat <- Spectre::demo.umap
cell.dat <- as.data.table(cell.dat)

# Draw plot
Spectre::fast.colour.plot(
  dat = cell.dat,
  x.axis = "UMAP_42_X",
  y.axis = "UMAP_42_Y",
  col.axis = "BV605.Ly6C"
)
```

---

fast.multi.plot *DEPRECATED fast multi plot*

Description

Please use `make.multi.plot(fast = TRUE)` instead.

Usage

```
fast.multi.plot(dat, x.axis, y.axis, plot.by, divide.by, add.density, col.type, figure.title, align.xy
```

Arguments

dat	NO DEFAULT. A data frame containing all the data you wish to plot
x.axis	NO DEFAULT. X axis
y.axis	NO DEFAULT. Y axis
plot.by	NO DEFAULT. A vector of character names for the columns you wish to plot.
divide.by	DEFAULT = NULL. Here you can specify a character name of a column you wish to use to divide up the dataset.

add.density	DEFAULT = FALSE. Can specify to add a density plot at the end the series of colour plots
hex	DEFAULT = FALSE. Whether to split the data into bins and show the average expression of the bin. Currently does not work with density plots, only for those features in the plot.by.
hex.bins	DEFAULT = 30. Number of bins to split into. Only used if hex is TRUE.
col.type	DEFAULT = "continuous". Can also be "factor".
figure.title	DEFAULT = "Multi plot". Also used as the prefix for the saved file name.
global.xy	DEFAULT = TRUE. Defines the limits for the X and Y based on the whole dataset. If FALSE, then each plot X & Y limits scale individually.
global.col	DEFAULT = TRUE. Defines the limits for the colour axis based on the whole dataset. If FALSE, then each plot colour limit scales individually.
colours	DEFAULTS to 'spectral'. What colour scheme do you want to use. Only used if type = 'colour', ignored if type = 'factor'. Can be 'jet', 'spectral', 'viridis', 'inferno', 'magma', or "BuPu".
dot.size	DEFAULT = 1. Numeric. Size of the dots.
col.min.threshold	DEFAULT = 0.01. Numeric. Define minimum threshold for colour scale. Values below this limit will be coloured as the chosen minimum threshold.
col.max.threshold	DEFAULT = 0.995 Numeric. Define maximum threshold for colour scale. Values above this limit will be coloured as the chosen maximum threshold.
path	DEFAULT = getwd() – i.e. the current working directory. Path to the desired output directory
plot.width	DEFAULT = 9.
plot.height	DEFAULT = 7.
blank.axis	DEFAULT = FALSE. Logical. Do you want a minimalist graph?
save.each.plot	DEFAULT = FALSE. Logical. Do you want to save each plot?

Details

Method to create multiple plots for each marker. This function allows you to create a grid of plots, where the cells are plotted by a series of columns, and/or subsetted by a certain factor (e.g. one sample per plot). Makes use of Spectre functions fast.colour.plot and make.density.plot.

Author(s)

Thomas Ashhurst, <thomas.ashhurst@sydney.edu.au> Felix Marsh-Wakefield, <felix.marsh-wakefield@sydney.edu.

Examples

```
# Create grid of plots on demonstration data
Spectre::fast.multi.plot(dat = as.data.table(Spectre::demo.umap),
                         x.axis = "UMAP_42_X",
                         y.axis = "UMAP_42_Y",
                         plot.by = c("BV605.Ly6C", "BUV737.B220", "AF700.CD45"))
```

make.autograph *make.autograph - Creates plots using ggplot2*

Description

This function allows you to generate quantitative plots of your data. Creates scatter plots. If interested in generating other plots (bar graphs) or adding statistics, we recommend reading more on using 'ggplot2' (for more info https://ggplot2.tidyverse.org/). Uses the package 'ggplot2' to generate plots, 'data.table' to manipulate data.

Usage

```
make.autograph(dat, x.axis, y.axis, colour.by, colours,
y.axis.label, grp.order, title, filename, scale, dot.size, width,
height, path)
```

Arguments

dat	NO DEFAULT. data.frame.
x.axis	NO DEFAULT. Character or numeric. Selects column that will specify x-axis.
y.axis	NO DEFAULT. Character or numeric. Selects column that will specify y-axis.
colour.by	DEFAULT = x.axis. Character, column name. Selects column that will determine differentiator of colour in plots.
y.axis.label	NO DEFAULT. Character. Specify y-axis label. Can be "" if one is not desired.
grp.order	DEFAULT = NULL. String of characters or numbers. Specify order of group for x-axis.
colours	DEFAULT = NULL. String of characters. Will determine the colours that will differentiate between colours specified by 'colour.by'.
my_comparisons	DEFAULT = NULL. A list containing vectors – each a pair of groups for comparison.
Variance_test	DEFAULT = NULL. Can be 'wilcox.test' (non-parametric) or 't.test' (parametric)
Pairwise_test	DEFAULT = NULL. Can be 'kruskal.test' (non-parametric) or 'anova' (parametric)
title	DEFAULT = paste0(y.axis.label, " - ", y.axis). Character. Specify title of plot on graph. Default combines 'y.axis.label' and 'y.axis'.
subtitle	DEFAULT = NULL. Plot subtitle.
filename	DEFAULT = paste0(y.axis.label, " - ", y.axis, ".pdf"). Character. Specify name plot will be exported as. Default combines 'y.axis.label' and 'y.axis'. Make sure to end with ".pdf" so file is correctly saved.
violin	DEFAULT = TRUE. Add violin plot underneath the data points.
scale	DEFAULT = "lin". Character. Select y-axis scale. Can also be "sci".

dot.size	DEFAULT = 5. Numeric. Specify size of dots on plot.
width	DEFAULT = 5. Numeric. Specify width of plot.
height	DEFAULT = 5. Numeric. Specify height of plot.
max.y	DEFAULT = 1.4. Multiplier for plot heigh – allowing room for stats.
path	DEFAULT = getwd(). The location to save plots. By default, will save to current working directory. Can be overidden.

Author(s)

Thomas M Ashhurst, <thomas.ashhurst@sydney.edu.au> Felix Marsh-Wakefield, <felix.marsh-wakefield@sydney.ed

Examples

```
dat <- data.frame(Samples = c("Mock_01", "Mock_02", "Mock_03", "WNV_01", "WNV_02", "WNV_03"),
                  Group = c(rep("Mock", 3), rep("WNV", 3)),
                  Tcells = c(20, 40, 30, 60, 70, 80),
                  Bcells = c(90, 95, 70, 20, 15, 30),
                  Batch = c(1,2,1,3,2,1)
                  )

Spectre::make.autograph(dat = dat,
                        x.axis = "Group",
                        y.axis = "Tcells",
                        colour.by = "Batch",
                        colours = c("Black", "Red", "Blue"),
                        y.axis.label = "Proportion"
                        )
```

---

make.colour.plot *Make colour plot*

Description

Create a dot plot (e.g., UMAP or tSNE) of cells, coloured by a continuous variable (e.g., marker expression) or a factor (e.g., cluster, group).

Usage

```
make.colour.plot(dat, x.axis, y.axis)
```

Arguments

dat	A data.table containing the data to plot.
x.axis	Character. Column name for the x-axis.
y.axis	Character. Column name for the y-axis.

col.axis	Character or NULL. Column name for colouring points. If NULL, points are coloured by density.
col.type	Character. "continuous" (default) or "factor". Determines how col.axis is interpreted.
add.label	Logical. If TRUE and col.type = "factor", adds labels at the centroid of each group.
hex	Logical. If TRUE, uses hex binning (only for continuous colour plots).
hex.bins	Integer. Number of hex bins if hex = TRUE.
colours	Character. Colour scheme for continuous plots. Options available are: "jet", all options in RColorBrewer::brewer.pal.info, and all options in viridis pallete. Default is "spectral".
col.min.threshold	Numeric. Minimum quantile for colour scale (continuous).
col.max.threshold	Numeric. Maximum quantile for colour scale (continuous).
align.xy.by	data.table. Data to use for aligning x/y axis limits.
align.col.by	data.table. Data to use for aligning colour scale limits.
regression.line	Character or NULL. If not NULL, adds a regression line ("lm", "loess", etc.).
title	Character or NULL. Plot title. Defaults to col.axis.
filename	Character or NULL. File name for saving the plot. If NULL, a name is generated automatically.
dot.size	Numeric. Size of points.
plot.width	Numeric. Width of saved plot (in inches).
plot.height	Numeric. Height of saved plot (in inches).
nudge_x	nudge_y Numeric. Amount to nudge centroid labels (if add.label = TRUE).
square	Logical. If TRUE, enforces a square aspect ratio.
legend.loc	Character. Legend position: "right" (default), "bottom", "top", "left", or "none".
save.to.disk	Logical. If TRUE (default), saves the plot to disk. If FALSE, only displays the plot.
path	Character. Directory to save the plot.
blank.axis	Logical. If TRUE, produces a minimalist plot with no axis lines or labels.
fast	Logical. If TRUE, uses scattermore for faster plotting of large datasets. Note, this will reduce the resolution of the plot. This only works when col.axis is specified and when hex = FALSE.

Value

A ggplot2 object representing the plot.

Author(s)

Thomas M Ashhurst, <thomas.ashhurst@sydney.edu.au> Givanna Putri

Examples

```
Spectre::make.colour.plot(
    dat = Spectre::demo.clustered,
    x.axis = "UMAP_X",
    y.axis = "UMAP_Y",
    col.axis = "CD4_asinh"
)
```

`make.multi.plot` *Make multiple plots for multiple columns and/or multiple sample/group/clusters*

Description

Method to create multiple plots for each marker. This function allows you to create a grid of plots, where the cells are plotted by a series of columns, and/or subsetted by a certain factor (e.g. one sample per plot). Makes use of Spectre functions make.colour.plot and make.density.plot.

Usage

```
make.multi.plot(dat, x.axis, y.axis, plot.by, divide.by, add.density,
col.type, figure.title, align.xy.by, align.col.by, colours, dot.size,
col.min.threshold, col.max.threshold, path, plot.width, plot.height,
blank.axis, save.each.plot)
```

Arguments

`dat`	NO DEFAULT. A data frame containing all the data you wish to plot
`x.axis`	NO DEFAULT. X axis
`y.axis`	NO DEFAULT. Y axis
`plot.by`	NO DEFAULT. A vector of character names for the columns you wish to plot.
`divide.by`	DEFAULT = NULL. Here you can specify a character name of a column you wish to use to divide up the dataset.
`add.density`	DEFAULT = FALSE. Can specify to add a density plot at the end the series of colour plots
`hex`	DEFAULT = FALSE. Whether to split the data into bins and show the average expression of the bin. Currently does not work with density plots, only for those features in the plot.by.
`hex.bins`	DEFAULT = 30. Number of bins to split into. Only used if hex is TRUE.
`col.type`	DEFAULT = "continuous". Can also be "factor".
`figure.title`	DEFAULT = "Multi plot". Also used as the prefix for the saved file name.
`global.xy`	DEFAULT = TRUE. Defines the limits for the X and Y based on the whole dataset. If FALSE, then each plot X & Y limits scale individually.

`global.col` DEFAULT = TRUE. Defines the limits for the colour axis based on the whole dataset. If FALSE, then each plot colour limit scales individually.

`align.xy.by` DEFAULT = dat. Align X and Y to a dataset. By default it will be based on the total dataset.

`align.col.by` DEFAULT = dat. Align colour to a dataset. By default it will be based on the total dataset.

`colours` DEFAULTS to 'spectral'. What colour scheme do you want to use. Only used if type = 'colour', ignored if type = 'factor'. Can be 'jet', 'spectral', 'viridis', 'inferno', 'magma', or "BuPu".

`dot.size` DEFAULT = 1. Numeric. Size of the dots.

`col.min.threshold` DEFAULT = 0.01. Numeric. Define minimum threshold for colour scale. Values below this limit will be coloured as the chosen minimum threshold.

`col.max.threshold` DEFAULT = 0.995 Numeric. Define maximum threshold for colour scale. Values above this limit will be coloured as the chosen maximum threshold.

`path` DEFAULT = getwd() – i.e. the current working directory. Path to the desired output directory

`plot.width` DEFAULT = 9.

`plot.height` DEFAULT = 7.

`blank.axis` DEFAULT = FALSE. Logical. Do you want a minimalist graph?

`save.each.plot` DEFAULT = FALSE. Logical. Do you want to save each plot?

`add.label` Logical. If TRUE and `col.type = "factor"`, adds labels at the centroid of each group.

`fast` Logical. If TRUE, uses scattermore for faster plotting of large datasets. Note, this will reduce the resolution of the plot. This only works when `col.axis` is specified and when hex = FALSE.

`legend.loc` Character. Legend position: "right" (default), "bottom", "top", "left", or "none".

Author(s)

Thomas Ashhurst, <thomas.ashhurst@sydney.edu.au> Felix Marsh-Wakefield, <felix.marsh-wakefield@sydney.edu.
Givanna Putri

Examples

```
Spectre::make.multi.plot(dat = Spectre::demo.clustered,
                         x.axis = "UMAP_X",
                         y.axis = "UMAP_Y",
                         plot.by = c("Ly6C_asinh", "B220_asinh", "CD45_asinh"))
```

make.pheatmap *make.pheatmap - Create a 'pretty' heatmap (pheatmap)*

Description

This function allows you to create a coloured heatmap for clusters vs markers (coloured by MFI) or clusters vs samples (coloured by MFI or cell counts). Can be set to plot 'fold-change' values (with is.fold = TRUE) that are provided in log2. make.pheatmap is a wrapper around the pheatmap function from pheatmap. Also uses the packages 'RColorBrewer' and 'scales' for colour customisation.

Usage

```
make.pheatmap(dat, sample.col, plot.cols, annot.cols, file.name,
plot.title, transpose, is.fold, fold.range, normalise, dendrograms,
row.sep, col.sep, cell.size, standard.colours, path)
```

Arguments

dat	NO DEFAULT. data.frame. Clusters/populations vs markers (MFI) or Clusters/populations vs samples or (MFI or cell numbers).
sample.col	NO DEFAULT. Character. Specify the name of the column that indicates samples. Will be rows on heatmap.
plot.cols	NO DEFAULT. Character vector. Name of columns you wish to plot on the heatmap. Will be columns on heatmap.
annot.cols	DEFAULT = NULL. Character. Columns which contain values that you do NOT want to plot in the heatmap, e.g. sample name, group name, Live/Dead etc.
file.name	DEFAULT = paste0("Pheatmap by ", sample.col, ".png"). Character. What do you want to call the file, including the extension.
plot.title	DEFAULT = paste0(sample.col, " heatmap"). Character.
transpose	DEFAULT = FALSE. Logical. Do you want to transpose the heatmap.
normalise	DEFAULT = TRUE. Logical. Only applies to standard heatmaps (i.e. when is.fold = FALSE). TRUE to normalise each column between 0 and 1, FALSE to plot the raw values.
is.fold	DEFAULT = FALSE. Logical. TRUE for fold-change heatmap, FALSE for normal values heatmap.
fold.range	DEFAULT = NULL. Numeric vector. For fold-change heatmaps, what is the maxium and minimum values that should be plotted. Example: for a max of 3, and a minimum of -3 would b c(3,-3). Defaults to NULL (which will use the max and min within the dataset).
dendrograms	DEFAULT = "both". Character. Do you want to include dendrograms on columns/rows. Can be "both", "row", "column", or "none.
cutree_rows	DEFAULT = 1 (i.e. no divisions). Divides rows based on the level of dendrogram branches, if dendrograms = 'both' or 'row'.

cutree_cols	DEFAULT = 1 (i.e. no divisions). Divides columns based on the level of dendrogram branches, if dendrograms = 'both' or 'column'
row.sep	DEFAULT = c(). Numeric. Only used if not clustering rows.
col.sep	DEFAULT = c(). Numeric. Only used if not clustering columns
cell.size	DEFAULT = 15. Numeric.
standard.colours	DEFAULT = "BuPu". Character. Can also be "RdYlBu", "YlGnBu", "viridis", "magma", "inferno", "spectral", "Blues", "Reds", "Greys", or "rev(RdBu)".
fold.colours	DEFAULT = "Spectre". Character, a custom blue:black:red fold-change colour scheme. Can also be "RdYlBu", "YlGnBu", "viridis", "magma", "inferno", "spectral", "Blues", "Reds", "Greys", or "rev(RdBu)".
path	DEFAULT = NULL. The location to save plots. By default, will save to current working directory. Can be overidden by entering a sub-directory name (e.g. if you have a subdirectory under your working directory called 'plots', you can enter this here).

Author(s)

Thomas M Ashhurst, <thomas.ashhurst@sydney.edu.au> Felix Marsh-Wakefield, <felix.marsh-wakefield@sydney.ed

References

https://sydneycytometry.org.au/spectre. Helpful examples at https://davetang.org/muse/2018/05/15/making-a-heatmap-in-r-with-the-pheatmap-package/

Examples

```
# Create some sample data
library(data.table)

dat <- Spectre::demo.clustered

## MFI cluster vs marker heatmap

# mock MFI data per sample, population.
demo.exp <- Spectre::demo.clustered[, lapply(.SD, mean), .SDcols = c("CD45_asinh", "Ly6G_asinh", "CD11b_asinh", "B


Spectre::make.pheatmap(dat = demo.exp,
                       file.name = "Expression pheatmap.png",
                       plot.title = "Expression",
                       sample.col = "Population",
                plot.cols = c("CD45_asinh", "Ly6G_asinh", "CD11b_asinh", "B220_asinh", "CD8a_asinh"))

# create mock population count per sample
demo.sum <- Spectre::demo.clustered[, .(count = .N), by = c("Sample", "Population", "Group", "Batch")]
demo.sum <- dcast(demo.sum, Sample+Group+Batch ~ Population, value.var='count')

## Z-scrore of fold-change type heatmap
z.dat <- do.zscore(dat = demo.sum,
```

```
                use.cols = c("CD4 T cells", "CD8 T cells", "Infil Macrophages", "Microglia", "NK cells", "Neutrophils"
                      replace = TRUE)

    Spectre::make.pheatmap(dat = z.dat,
                           file.name = "z-score.png",
                           plot.title = "z-score",
                           sample.col = "Sample",
                      plot.cols = c("CD4 T cells", "CD8 T cells", "Infil Macrophages", "Microglia", "NK cells", "Neutroph
                           annot.cols = c("Group", "Batch"),
                           is.fold = TRUE,
                           fold.range = c(3,-3)
                           )
```

`make.spatial.plot` *make.spatial.plot*

Description

make.spatial.plot

Usage

```
make.spatial.plot(
  dat,
  image.roi,
  image.channel,
  mask.outlines = NULL,
  cell.dat = NULL,
  cell.col = NULL,
  image.y.flip = TRUE,
  image.mask.size = 0.1,
  image.mask.colour = "gold",
  image.min.threshold = 0,
  image.max.threshold = 0.99,
  image.blank = FALSE,
  cell.x = "x",
  cell.y = "y",
  cell.col.type = "numeric",
  cell.colours = "spectral",
  cell.col.min.threshold = 0.01,
  cell.col.max.threshold = 0.995,
  title = paste0(image.roi),
  dot.size = 1,
  dot.alpha = 1,
  align.xy.by = cell.dat,
  align.col.by = cell.dat,
  save.to.disk = TRUE,
```

```
  path = getwd(),
  plot.width = 9,
  plot.height = 7,
  blank.axis = FALSE
)
```

Arguments

dat	NO DEFAULT.A spatial data list
image.roi	NO DEFAULT.The name of the ROI to plot
image.channel	NO DEFAULT.Name of the channel to plot
mask.outlines	DEFAULT = NULL. Name of the mask for outlines
cell.dat	DEFAULT = NULL. Character name for the cellular dataset in the spatial data list, or a data.table object.
cell.col	DEFAULT = NULL. Name of cellular dataset to colour points by
image.y.flip	DEFAULT = TRUE. Flips the Y-axis orientation.
image.mask.size	DEFAULT = 0.1
image.mask.colour	DEFAULT = "gold"
image.min.threshold	DEFAULT = 0.00. Lower threshold for plot signal. Values below this target will be clipped.
image.max.threshold	DEFAULT = 0.99. Upper threshold for plot signal. Values above this target will be clipped.
image.blank	DEFAULT = FALSE. Will blank the plot image.
cell.x	DEFAULT = "x". Column name for the 'x' coordinate in data.table
cell.y	DEFAULT = "y". Column name for the 'y' coordinate in data.table
cell.col.type	DEFAULT= "numeric". Can be 'factor'.
cell.colours	= DEFAULT = "spectral".
cell.col.min.threshold	= DEFAULT = 0.01. Values below this target will be clipped.
cell.col.max.threshold	= DEFAULT = 0.995. Values above this target will be clipped.
title	DEFAULT = paste0(image.roi)
dot.size	DEFAULT = 1
dot.alpha	DEFAULT = 1
align.xy.by	DEFAULT = cell.dat
align.col.by	DEFAULT = cell.dat
save.to.disk	DEFAULT = TRUE
path	DEFAULT = getwd(). Path for saving image files
plot.width	DEFAULT = 9
plot.height	DEFAULT = 7
blank.axis	DEFAULT = FALSE

`make.volcano.plot` *Make an enhanced volcano plot*

Description

Make an enhanced volcano plot

Usage

```
make.volcano.plot(
  dat.p,
  dat.fc,
  vars,
  title,
  subtitle = NULL,
  xlim = c(-6, 8),
  ylim = c(0, 4),
  path = getwd(),
  width = 7,
  height = 8,
  pCutoff = 0.05,
  FCcutoff = 0.26,
  hline = c(0.01, 0.05),
  hlineCol = c("grey75", "grey25"),
  pointSize = 3,
  labSize = 3,
  col = c("black", "black", "blue", "red3"),
  colAlpha = 1,
  ...
)
```

`package.check` *package.check - a function to check the installation of all required packages.*

Description

This function allows you to check to see if all the common use packages dependencies for Spectre are installed. See https://sydneycytometry.org.au/spectre for usage instructions and vignettes.

Usage

```
package.check(type = "general")
```

Arguments

type	DEFAULT = "general". If "general", then checks for the packages required for general Spectre usage. If "spatial", then checks for additional packages required for spatial analysis. If "ML", then checks for additional packages required for machine-learing functionality.

Value

returns an error message if one of the common use packages are not installed. Proceeds in order of package importance, and only the first error message encountered will be returned.

Author(s)

Thomas M Ashhurst, <thomas.ashhurst@sydney.edu.au>

References

https://sydneycytometry.org.au/spectre

Examples

```
package.check()
```

---

package.install *package.install - a function to install packages required for Spectre.*

Description

This function allows you to check to see if all the common use packages dependencies for Spectre are installed. Will only install if the package has not been installed, will not update packages. See https://sydneycytometry.org.au/spectre for usage instructions and vignettes.

Usage

```
package.install(type = "general", update = FALSE)
```

Arguments

type	DEFAULT = "general". If "general", then checks for the packages required for general Spectre usage. If "spatial", then checks for additional packages required for spatial analysis.
update	DEFAULT = FALSE. If FALSE, will only install packages that are not already installed – no updates of packages will be performed. If TRUE, will install and update packages.

Value

returns an error message if one of the common use packages are not installed. Proceeds in order of package importance, and only the first error message encountered will be returned.

Author(s)

Thomas M Ashhurst, <thomas.ashhurst@sydney.edu.au>

References

https://sydneycytometry.org.au/spectre

Examples

```
package.install()
package.install(type = 'spatial')
```

---

package.load *package.load - a function to load (library) all required packages.*

Description

This function allows you to load all of the common use packages dependencies for Spectre. See https://sydneycytometry.org.au/spectre for usage instructions and vignettes.

Usage

```
package.load(type = "general")
```

Arguments

type	DEFAULT = "general". If "general", then loads packages required for general Spectre usage. If "spatial", then loads additional packages required for spatial analysis. If "ML", then loads additional packages required for machine-learing functionality.

Value

loads all the common use package libraries.

Author(s)

Thomas M Ashhurst, <thomas.ashhurst@sydney.edu.au>

References

https://sydneycytometry.org.au/spectre

Examples

```
package.load()
```

`prep.cytonorm` *prep.cytonorm - Prepare reference data into a FlowSOM object*

Description

This function allows you to prepare reference data ahead of performing batch alignment.

Usage

```
prep.cytonorm()
```

Arguments

`dat`	NO DEFAULT. A data.table consisting of the 'refernece' data you will use to train the alignment algorithm
`cellular.cols`	NO DEFAULT. A vector of column names from the data.table that contain 'cellular' markers
`cluster.cols`	NO DEFAULT. A vector of column names from the data.table that contain markers you wish to use for clusteirng
`batch.col`	NO DEFAULT. Name of the column that contains batch names
`sample.col`	DEFAULT = NULL. Name of the column that contains sample names
`dir`	DEFAULT = getwd(). Sets the working directory to operate from. Because this function involves some reading/writing of files, it's best to set this to somewhere static in case the active working directory moves to a subfolder, and then doesn't return because the function runs into an error.
`xdim`	DEFAULT = 5. Size of X-axis of FlowSOM grid.
`ydim`	DEFAULT = 5. Size of Y-axis of FlowSOM grid.
`meta.k`	DEFAULT = 10. Number of metaclusters. If set to 1, will map all cells to a single metacluster
`seed`	DEFAULT = 42. Seed for reproducibility.
`mem.ctrl`	DEFAULT = TRUE. Allows the function to clear held memory on occasion.
`n.cells`	DEFAULT = 10000000. Maxium cells allowed to be read for the prep.cytonorm functionality. Cells read from each file will be the lesser of n.cells / number of files or the total number of cells in the file.

Value

Returns an object which represents the alignment model. In this preparation stage, it contains the FlowSOM object containing the reference data. The 'train.align' function can then be used to calculate the conversions between batches for each metacluser.

Author(s)

Thomas M Ashhurst, <thomas.ashhurst@sydney.edu.au>

References

Ashhurst, T. M., et al. (2019). https://www.ncbi.nlm.nih.gov/pubmed/31077106

Examples

```
# TBC
# align.model <- prep.cytonorm()
```

`read.files` *Read CSV or FCS files into a list.*

Description

This function allows you to read in sample files (.csv or .fcs) into a list, where each file is saved as a data.table object.

Usage

```
read.files(
  file.loc = getwd(),
  file.type = c(".csv", ".fcs"),
  files = NULL,
  nrows = NULL,
  do.embed.file.names = TRUE,
  header = TRUE
)
```

Arguments

`file.loc`	DEFAULT = getwd(). What is the location of your files?
`file.type`	DEFAULT = ".csv". What type of files do you want to read. Can be ".csv" or ".fcs".
`files`	DEFAULT = NULL. A vector of selected file names to import.
`nrows`	DEFAULT = NULL. Can specify a numerical target for the number of cells (rows) to be read from each file. Please note, order is random in FCS files.
`do.embed.file.names`	DEFAULT = TRUE. Do you want to embed each row (cell) of each file with the name name?
`header`	DEFAULT = TRUE. Does the first line of data contain column names? Only works for file.type = ".csv".

Value

Returns a list of data.tables – one per file.

Author(s)

Thomas M Ashhurst, <thomas.ashhurst@sydney.edu.au> Felix Marsh-Wakefield, <felix.marsh-wakefield@sydney.ed
Givanna Putri

References

Ashhurst, T. M., et al. (2019). https://www.ncbi.nlm.nih.gov/pubmed/31077106

Examples

```
# download sample data
download.file(url='https://github.com/ImmuneDynamics/data/blob/main/msCNS.zip?raw=TRUE', destfile = 'msCNS.zip'
unzip(zipfile = 'msCNS.zip')
data.list <- read.files(file.loc = "msCNS/data", file.type = ".csv", do.embed.file.names = TRUE)
```

---

read.spatial.files *Read TIFF files into R and and create spatial data object*

Description

Read TIFF files into R and and create spatial data object

Usage

```
read.spatial.files()
```

Arguments

dir	Directory location of where the ROI files are located
correct.extent	Correct extent so that the minimum is 0,0. Default TRUE
flip.y	Flip the data arrangement for the y-axis. Default TRUE
value.modifier	Data modifier based on image processing. Default 65535
lois	Default = NULL. You can provide a list of only some of the ROI folders to read in.

Value

Returns a list of spatial data objects.

run.align *run.align - Run the alignment model on a target data.table*

Description

This function allows you to prepare reference data ahead of performing batch alignment, using on of the CytofBatchAdjust functions

Usage

```
run.align()
```

Arguments

ref.dat	NO DEFAULT. A data.table consisting of the 'refernece' data you will use to train the alignment algorithm
target.dat	NO DEFAULT.A data.table consisting of the 'target' data you will use to align the data
batch.col	NO DEFAULT. Column name of the data.table that contains batch labels
align.cols	NO DEFAULT. Vector of column names to align.
method	DEFAULT = 'quantile'. Can be 'quantile', 'SD', or a percentile indicated as '50p' (50th percentile) or '95p' (95th percentile) etc.
append.name	DEFAULT = "_aligned". This will be appended to the column names containing the aligned data
dir	DEFAULT = getwd(). Sets the working directory to operate from. Because this function involves some reading/writing of files, it's best to set this to somewhere static in case the active working directory moves to a subfolder, and then doesn't return because the function runs into an error.
mem.ctrl	DEFAULT = TRUE. Allows the function to clear held memory on occasion.

Value

Returns a data.table with aligned data added in new columns.

Author(s)

Thomas M Ashhurst, <thomas.ashhurst@sydney.edu.au>

References

Ashhurst, T. M., et al. (2019). https://www.ncbi.nlm.nih.gov/pubmed/31077106

run.cytonorm *run.cytonorm - Run the alignment model on a target data.table*

Description

This function allows you to prepare reference data ahead of performing batch alignment.

Usage

```
run.align()
```

Arguments

dat	NO DEFAULT. A data.table consisting of the 'refernece' data you will use to train the alignment algorithm
model	NO DEFAULT. A batch alignment conversion model object created by the prep.align() and train.align() functions.
batch.col	NO DEFAULT. Column name of the data.table that contains batch labels
append.name	DEFAULT = "_aligned". This will be appended to the column names containing the aligned data
dir	DEFAULT = getwd(). Sets the working directory to operate from. Because this function involves some reading/writing of files, it's best to set this to somewhere static in case the active working directory moves to a subfolder, and then doesn't return because the function runs into an error.
mem.ctrl	DEFAULT = TRUE. Allows the function to clear held memory on occasion.

Value

Returns a data.table with aligned data added in new columns.

Author(s)

Thomas M Ashhurst, <thomas.ashhurst@sydney.edu.au>

References

Ashhurst, T. M., et al. (2019). https://www.ncbi.nlm.nih.gov/pubmed/31077106

`run.fitsne` *Run FIt-SNE, Fourier Transform TSNE.*

Description

Implementation of FIt-SNE is available from https://github.com/KlugerLab/FIt-SNE. This function uses `fftRtsne` to run FIt-SNE.

Usage

```
run.fitsne(dat, use.cols, seed = 42, fitsne.x.name = "FItSNE_X",
fitsne.y.name = "FItSNE_Y", dims = 2, perplexity = 30, theta = 0.5,
max_iter = 750, fft_not_bh = TRUE, ann_not_vptree = TRUE,
  stop_early_exag_iter = 250, exaggeration_factor = 12.0,
  no_momentum_during_exag = FALSE,start_late_exag_iter = -1,
  late_exag_coeff = 1.0, mom_switch_iter = 250, momentum = 0.5,
  final_momentum = 0.8, learning_rate = 'auto', n_trees = 50,
  search_k = -1, nterms = 3, intervals_per_integer = 1,
  min_num_intervals = 50, K = -1, sigma = -30, initialization = 'pca',
  max_step_norm = 5, load_affinities = NULL, fast_tsne_path = NULL,
  nthreads = 0, perplexity_list = NULL, get_costs = FALSE,  df = 1.0)
```

Arguments

`dat`	NO DEFAULT. Input data.table or data.frame.
`use.cols`	NO DEFAULT. Vector of column names or numbers for clustering.
`seed`	Default = 42. Seed value for reproducibility.
`fitsne.x.name`	Default = "FItSNE_X". Character. Name of FItSNE x-axis.
`fitsne.y.name`	Default = "FItSNE_Y". Character. Name of FItSNE y-axis.
`dims`	Default = 2. Dimensionality of the embedding (reduced data).
`perplexity`	Default = 30. Perplexity is used to determine the bandwidth of the Gaussian kernel in the input space
`theta`	Default = 0.5. For exact t-SNE, set to 0. If non-zero, then will use either Barnes Hut or FIt-SNE based on nbody_algo. If Barnes Hut, then this determines the accuracy of BH approximation.
`max_iter`	Default = 750. Number of iterations of t-SNE to run.
`fft_not_bh`	Default = TRUE. If theta is nonzero, this determines whether to use FIt-SNE or Barnes Hut approximation.
`ann_not_vptree`	Default = TRUE. Use vp-trees (as in bhtsne) or approximate nearest neighbors (default). Set to be TRUE for approximate nearest neighbors.
`stop_early_exag_iter`	Default = 250. When to switch off early exaggeration.
`exaggeration_factor`	Default = 12. Coefficient for early exaggeration (>1).

`no_momentum_during_exag`
: Default = FALSE. Set to 0 to use momentum and other optimization tricks. Can be set to 1 to do plain, vanilla gradient descent (useful for testing large exaggeration coefficients).

`start_late_exag_iter`
: Default = -1. When to start late exaggeration. Set to -1 by default to not use late exaggeration.

`late_exag_coeff`
: Default = 1. Late exaggeration coefficient. Set to 1 by default to not use late exaggeration.

`mom_switch_iter`
: Default = 250. Iteration number to switch from momentum to final_momentum.

`momentum`
: Default = 0.5.Initial value of momentum.

`final_momentum`
: Default = 0.8. Value of momentum to use later in the optimisation.

`learning_rate`
: Default = 'auto'. Set to desired learning rate or 'auto', which sets learning rate to N/exaggeration_factor where N is the sample size, or to 200 if N/exaggeration_factor < 200.

`n_trees`
: Default = 50. When using Annoy, the number of search trees to use.

`search_k`
: Default = -1. When using Annoy, the number of nodes to inspect during search. Default is -1 which translate to 3*perplexity*n_trees (or K*n_trees when using fixed sigma).

`nterms`
: Default = 3. If using FIt-SNE, this is the number of interpolation points per sub-interval.

`intervals_per_integer`
: Default = 1. See min_num_intervals.

`min_num_intervals`
: Default = 50. Let maxloc = ceil(max(max(X))) and minloc = floor(min(min(X))). i.e. the points are in a minloc^no_dims by maxloc^no_dims interval/square. The number of intervals in each dimension is either min_num_intervals or ceil((maxloc - minloc)/intervals_per_integer), whichever is larger. min_num_intervals must be an integer >0, and intervals_per_integer must be >0. Defaults are min_num_intervals=50 and intervals_per_integer = 1.

`K`
: Default = -1. Number of nearest neighbours to get when using fixed sigma.

`sigma`
: Default = -30. Fixed sigma value to use when perplexity==-1.

`initialization`
: Default = 'pca'. pca', 'random', or N x no_dims array to intialize the solution.

`max_step_norm`
: Default = 5. Maximum distance that a point is allowed to move on one iteration. Larger steps are clipped to this value. This prevents possible instabilities during gradient descent. Set to -1 to switch it off.

`load_affinities`
: Default = NULL. If 1, input similarities are loaded from a file and not computed. If 2, input similarities are saved into a file. If 0, affinities are neither saved nor loaded.

`fast_tsne_path`
: Default = NULL. Path to FItSNE executable.

`nthreads`
: Default = 0. Number of threads to use, set to use all available threads by default.

perplexity_list
: Default = NULL. If perplexity==0 then perplexity combination will be used with values taken from perplexity_list.

get_costs
: Default = FALSE. Logical indicating whether the KL-divergence costs computed every 50 iterations should be returned.

df
: Default = 1.0. Positive numeric that controls the degree of freedom of t-distribution. The actual degree of freedom is 2*df-1. The standard t-SNE choice of 1 degree of freedom corresponds to df=1. Large df approximates Gaussian kernel. df<1 corresponds to heavier tails, which can often resolve substructure in the embedding. See Kobak et al. (2019) for details.

Author(s)

Givanna Putri

Examples

```
## Not run:
dat <- Spectre::demo.clustered
dat.sub <- Spectre::do.subsample(dat, 30000)
use.cols <- names(dat)[12:19]
dat.reduced <- run.fitsne(dat = dat.sub, use.cols = use.cols)

## End(Not run)
```

---

run.flowsom *Run FlowSOM*

Description

Method to run the FlowSOM clustering algorithm. This function runs FlowSOM on a data.table with cells (rows) vs markers (columns) with new columns for FlowSOM clusters and metaclusters. Output data will be "flowsom.res.original" (for clusters) and "flowsom.res.meta" (for metaclusters). Uses the R packages "FlowSOM" for clustering, "flowCore" for handling .fcs files, "Biobase" for creating a flow frame, "data.table" for handling data.table format.

Usage

```
run.flowsom(dat, use.cols, xdim=14, ydim=14, meta.k='auto',
max.meta=20, clust.seed=42, meta.seed=42, clust.name="FlowSOM_cluster",
meta.clust.name="FlowSOM_metacluster", mem.ctrl=TRUE, verbose=TRUE)
```

Arguments

dat	NO DEFAULT. data.frame. Input sample.
use.cols	NO DEFAULT. Vector of column names to use for clustering.
xdim	DEFAULT = 14. Numeric. Number of first level clusters across the x-axis. xdim x ydim = total number of first level FlowSOM clusters.
ydim	DEFAULT = 14. Numeric. Number of first level clusters across the y-axis. xdim x ydim = total number of first level FlowSOM clusters.
meta.k	DEFAULT = 'auto'. If set to 'auto', then number of metaclusters will be determined automatically. Alternatively, can specify the desired number of metaclusters to create. If set to zero (0), no metaclusters will be created.
max.meta	DEFAULT = 20. Only used if meta.k is set to 'auto'. This parameter indicates the maximum number of metaclusters FlowSOM will try out when determining the optimal number of metaclusters for the dataset.
clust.seed	DEFAULT = 42 Numeric. Clustering seed for reproducibility.
meta.seed	DEFAULT = 42 Numeric. Metaclustering seed for reproducibility.
clust.name	DEFAULT = "FlowSOM_cluster". Character. Name of the resulting 'cluster' parameter.
meta.clust.name	DEFAULT = "FlowSOM_metacluster". Character. Name of the resulting 'metacluster' parameter.
mem.ctrl	DEFAULT = TRUE. Runs gc() (garbage collection) after a number of steps to free up memory that hasn't been released quickly enough.
verbose	DEFAULT = TRUE. Logical. Whether to print progress messages.

Author(s)

Thomas Ashhurst, <thomas.ashhurst@sydney.edu.au> Felix Marsh-Wakefield, <felix.marsh-wakefield@sydney.edu.

Examples

```
# Run FlowSOM on demonstration dataset
res <- Spectre::run.flowsom(Spectre::demo.clustered,
use.cols = c("NK11_asinh", "CD3_asinh",
"CD45_asinh", "Ly6G_asinh", "CD11b_asinh",
"B220_asinh", "CD8a_asinh", "Ly6C_asinh",
"CD4_asinh"))
```

run.harmony *run.harmony - dun Harmony alignment on a data.table*

Description

This function allows you to run the 'Harmony' data alignment algorithm on single cell or cytometry data stored in a data.table

Usage

```
run.harmony()
```

Arguments

dat	NO DEFAULT. A data.table with all of the data you wish to align
align.cols	NO default. The columns you wish to align. For cytometry data, this can be the markers themselves or principle components. For single-cell seq data, principle components are recommended.
batch.col	NO default. The column that denotes the batch or dataset that each cell belongs to
append.name	DEFAULT = '_aligned'. Text that will be appended to the new columns containing aligned data
do_pca	DEFAULT = TRUE. Whether to perform PCA on input matrix.
npcs	If doing PCA on input matrix, number of PCs to compute.
theta	Diversity clustering penalty parameter. Specify for each variable in vars_use Default theta=2. theta=0 does not encourage any diversity. Larger values of theta result in more diverse clusters.
lambda	Ridge regression penalty parameter. Specify for each variable in vars_use. Default lambda=1. Lambda must be strictly positive. Smaller values result in more aggressive correction.
sigma	Width of soft kmeans clusters. Default sigma=0.1. Sigma scales the distance from a cell to cluster centroids. Larger values of sigma result in cells assigned to more clusters. Smaller values of sigma make soft kmeans cluster approach hard clustering.
nclust	Number of clusters in model. nclust=1 equivalent to simple linear regression.
tau	Protection against overclustering small datasets with large ones. tau is the expected number of cells per cluster.
block.size	What proportion of cells to update during clustering. Between 0 to 1, default 0.05. Larger values may be faster but less accurate
max.iter.harmony	Maximum number of rounds to run Harmony. One round of Harmony involves one clustering and one correction step.

`max.iter.cluster`
: Maximum number of rounds to run clustering at each round of Harmony.

`epsilon.cluster`
: Convergence tolerance for clustering round of Harmony. Set to -Inf to never stop early.

`epsilon.harmony`
: Convergence tolerance for Harmony. Set to -Inf to never stop early.

`plot_convergence`
: Whether to print the convergence plot of the clustering objective function. TRUE to plot, FALSE to suppress. This can be useful for debugging.

`return_object`
: (Advanced Usage) Whether to return the Harmony object or only the corrected PCA embeddings.

`verbose`
: DEFAULT = FALSE. Whether to print progress messages. TRUE to print, FALSE to suppress.

`reference_values`
: (Advanced Usage) Defines reference dataset(s). Cells that have batch variables values matching reference_values will not be moved.

`cluster_prior`
: (Advanced Usage) Provides user defined clusters for cluster initialization. If the number of provided clusters C is less than K, Harmony will initialize K-C clusters with kmeans. C cannot exceed K.

Value

Returns a data.table with aligned data added in new columns.

Author(s)

Thomas M Ashhurst, <thomas.ashhurst@sydney.edu.au>

`run.knn.classifier` *run.knn.classifier - Run KNN classifier*

Description

run.knn.classifier - Run KNN classifier

Usage

```
run.knn.classifier(train.dat, unlabelled.dat, use.cols, label.col, num.neighbours)
```

Arguments

`train.dat`	NO DEFAULT. data.frame. A dataframe containing cells (rows) vs features/markers (columns) to be used to train classifier.
`unlabelled.dat`	NO DEFAULT. data.frame. A dataframe containing cells (rows) vs features/markers (columns) that is the data classified.
`use.cols`	NO DEFAULT. Vector of column names to use for training k-nearest neighbour (KNN) classifier.
`label.col`	NO DEFAULT. Character. Name of the column representing the population name of the cells in dat.
`num.neighbours`	DEFAULTS to 1. Numeric. When using a k-nearest neighbour classifier, then this parameter specifies the number of nearest neighbours. Train a k-NN classifier on a training data, and use it to classify an unlabelled data. Note that for the classifier to work as intended, the unlabelled.data has to be normalised to the range of the train.data. Note make sure that train.data and unlabelled.data has exactly the same features/markers.

Value

The predicted label for the unlabelled data.

Author(s)

Givanna Putri, <ghar1821@uni.sydney.edu.au>

References

https://sydneycytometry.org.au/spectre.

`run.pca` *Run the PCA algorithm (using stats::prcomp)*

Description

Method to run a PCA dimensionality reduction algorithm. A principal component analysis (PCA) is capable of reducing the number of dimensions (i.e. parameters) with minimal effect on the variation of the given dataset. This function will run a PCA calculation (extremely fast) and generate plots (takes time). For individuals (such as samples or patients), a PCA can group them based on their similarities. A PCA is also capable of ranking variables/parameters (such as markers or cell counts) based on their contribution to the variability across a dataset in an extremely fast manner. In cytometry, this can be useful to identify marker(s) that can be used to differentiate between subset(s) of cells. Uses the base R package "stats" for PCA, "factoextra" for PCA and scree plots, "data.table" for saving .csv files, "ggplot2" for saving plots, "gtools" for rearranging data order, 'RColorBrewer' and 'viridis' for colour schemes. More information on PCA plots can be found here http://www.sthda.com/english/articles/31-principal-component-methods-in-r-practical-guide/118-principal-component-analysis-in-r-prcomp-vs-princomp/.

Usage

```
run.pca(dat, use.cols, scale = TRUE, add.pca.col = FALSE,
pca.col.no = 50, pca.lite = FALSE, scree.plot = TRUE, comp.no = 2,
variable.contribution = TRUE, plot.individuals = TRUE,
plot.ind.label = "point", pointsize.ind = 1.5, row.names = NULL,
plot.ind.group = FALSE, group.ind = NULL, colour.group = "viridis",
pointsize.group = 1.5, ellipse.type = "confidence",
ellipse.level = 0.95, mean.point = TRUE,
randomise.order = TRUE, order.seed = 42,
plot.variables = TRUE, colour.var = "solid",
plot.combined = TRUE, repel = FALSE, var.numb = 20, path = getwd())
```

Arguments

dat	NO DEFAULT. data.frame.
use.cols	NO DEFAULT. Vector of numbers, reflecting the columns to use for dimensionality reduction (may not want parameters such as "Time" or "Sample").
scale	DEFAULT = TRUE. A logical value indicating whether the variables should be scaled to have unit variance before the analysis takes place.
add.pca.col	DEFAULT = FALSE. Option to add PC coordinates to input data.
pca.col.no	DEFULAT = 50. Number of PC to be added to input data.
pca.lite	DEFAULT = FALSE. Will stop running the function after PCA coordinates have been added to input data (assuming add.pca.col = TRUE).
scree.plot	DEFAULT = TRUE. Option to create scree plots. Will save generated scree plot. Note this will require the input of an elbow point during run if comp.no = NULL.
comp.no	DEFAULT = 2. Select number of components to be saved. If NULL, user will be asked during run to select number based on scree plot.
variable.contribution	DEFAULT = TRUE. Option to create plot showing the contribution of each variable. Horizontal red line represents the average variable contribution if all variables contributed equally. Requires scree.plot = TRUE.
plot.individuals	DEFAULT = TRUE. Option to create PCA plots on individuals (samples/patients).
plot.ind.label	DEFAULT = "point". Option to add text to PCA plots on individuals as an extra identifier. Use c("point", "text") to include both text and point.
pointsize.ind	DEFAULT = 1.5. Numeric. Size of dots of individuals on PCA plot.
row.names	DEFAULT = NULL. Column (as character) that defines individuals. Will be used to place name on plot.individuals.
plot.ind.group	DEFAULT = FALSE. Option to group individuals with ellipses (which by default show the 95 % confidence interval). Must specify column that groups individuals with group.ind.
group.ind	DEFAULT = NULL. Column (as character) that defines groups of individuals. Works with plot.ind.group which must be set to TRUE.

`colour.group` DEFAULT = "viridis". Colour scheme for each group. Options include "jet", "spectral", "viridis", "inferno", "magma".

`pointsize.group` DEFAULT = 1.5. Numeric. Size of shapes of group individuals on PCA plot.

`ellipse.type` DEFAULT = "confidence". Set type of ellipse. Options include "confidence", "convex", "concentration", "t", "norm", "euclid". See factoextra::fviz for more information.

`ellipse.level` DEFAULT = 0.95. Size of ellipses. By default 95 % (0.95).

`mean.point` DEFAULT = TRUE. Option to plot the mean on PCA plot with different groups.

`randomise.order` DEFAULT = TRUE. Option to randomise plotting order of individuals to control for overlap.

`order.seed` DEFAULT = 42. Set the seed for randomising plotting order of individuals.

`plot.variables` DEFAULT = TRUE. Option to create PCA plots on variables (markers/cell counts).

`colour.var` DEFAULT = "solid". Colour scheme for PCA plot with variables. Options include "solid", "jet", "spectral", "viridis", "inferno", "magma", "BuPu". Note some colours are pale and may not appear clearly on plot.

`plot.combined` DEFAULT = TRUE. Option to create a combined PCA plot with both individuals and variables.

`repel` DEFAULT = FALSE. Option to avoid overlapping text in PCA plots. Can greatly increase plot time if there is a large number of samples.

`var.numb` DEFAULT = 20. Top number of variables to be plotted. Note the greater the number, the longer plots will take.

`path` DEFAULT = getwd(). The location to save plots. By default, will save to current working directory. Can be overidden.

Author(s)

Felix Marsh-Wakefield, <felix.marsh-wakefield@sydney.edu.au>

Examples

```
# Set directory to save files. By default it will save files at get()

# Run PCA on demonstration dataset, adding PC to dataset
dat <- Spectre::demo.clustered

# Run PCA on demonstration dataset
Spectre::run.pca(dat = Spectre::demo.clustered,
                 use.cols = c(11:19),
                 repel = TRUE
                 )

# Compare between groups
## Not run:
Spectre::run.pca(dat = Spectre::demo.clustered,
```

```
                  use.cols = c(11:19),
                  comp.no = NULL,
              plot.ind.label = c("point", "text"), #individual cells will be labelled as numbers
                  plot.ind.group = TRUE,
                  group.ind = "Group",
                  mean.point = FALSE,
                  randomise.order = TRUE
                  )

## End(Not run)

# When prompted, type in "5" and click enter to continue function
# (this selects the elbow point based off the scree plot)

## Possible issues ##
# Remove any NA present
na.omit(dat)

# Remove columns that have zero variance (e.g. if MFI is the same for all
# samples for a marker)
dat <- data.table::as.data.table(dat)
dat <- dat[ , lapply(.SD, function(v) if(data.table::uniqueN(v, na.rm = TRUE) > 1) v)]


# Ellipses are only generated in 'plot.ind.group' when there are at least
# 2 samples per group ('group.ind')
```

`run.phenograph` *run.phenograph - Run phenograph clustering*

Description

This function allows you to perform phenograph clustering on a data.table

Usage

```
run.phenograph()
```

Arguments

`ref.dat`	NO DEFAULT. A data.table consisting of the 'refernece' data you will use to train the alignment algorithm

Value

Returns a data.table with clustering results added

Author(s)

Thomas M Ashhurst, <thomas.ashhurst@sydney.edu.au>

References

Ashhurst, T. M., et al. (2019). https://www.ncbi.nlm.nih.gov/pubmed/31077106

Examples

```
## Not run:
cell.dat <- Spectre::demo.clustered
cell.dat <- Spectre::do.subsample(cell.dat, 5000)

use.cols <- c("NK11_asinh", "CD3_asinh", "CD45_asinh",
"Ly6G_asinh", "CD11b_asinh", "B220_asinh", "CD8a_asinh",
"Ly6C_asinh", "CD4_asinh")

cell.dat <- run.phenograph(dat = cell.dat, use.cols = use.cols)

## End(Not run)
```

`run.rpca` *Run rPCA for batch correction*

Description

Use Seurat rPCA to do batch correction. For more details on rPCA, see https://satijalab.org/seurat/articles/integration_rpca.ht

Usage

```
run.rpca(dat, use.cols, batch.col, reference = NULL, k.anchor = 5)
```

Arguments

`dat`	NO DEFAULT. A data.table
`use.cols`	NO DEFAULT. A vector of character column names to apply batch correction to.
`batch.col`	Character. The column in the data in data.table that identifies which batch each cell belongs to.
`reference`	DEFAULT NULL. Whether to align to batches to a given batch. If yes, then supply this parameter with the name of the batch you want to align the other batches to.
`k.anchor`	DEFAULT 5. Passed to Seurat's FindIntegrationAnchors function. Essentially, it determines the number of neighbors (k) to use when Seurat's FindIntegrationAnchors is picking anchors.

Value

batch corrected data.table

`run.spatial.analysis` *run.spatial.analysis*

Description

run.spatial.analysis

Usage

```
run.spatial.analysis(
  dat,
  sample.col,
  pop.col,
  annot.cols = NULL,
  region.col = NULL,
  area.table = NULL,
  adj.dist = 20,
  x.col = "x",
  y.col = "y",
  distribution = TRUE,
  composition = TRUE,
  counts = TRUE,
  counts.per.area = TRUE,
  distance = TRUE,
  adjacency = TRUE,
  func = "mean"
)
```

Arguments

`dat`	NO DEFAULT. Data.table
`sample.col`	NO DEFAULT. Column that denotes 'samples'
`pop.col`	NO DEFAULT. Column that denotes 'samples'
`annot.cols`	DEFAULT = NULL. Annotation columns.
`region.col`	DEFAULT = NULL. Create a 'total' by default, and add specific ones if requested here
`area.table`	DEFAULT = NULL. Calculate 'total' automatically
`adj.dist`	DEFAULT = 20
`x.col`	DEFAULT = 'x'
`y.col`	DEFAULT = 'y'
`distribution`	DEFAULT = TRUE
`composition`	DEFAULT = TRUE
`counts`	DEFAULT = TRUE

counts.per.area	DEFAULT = TRUE
distance	DEFAULT = TRUE
adjacency	DEFAULT = TRUE
func	DEFAULT = 'mean'

`run.tsne` *Run the tSNE algorithm (using Rtsne::Rtsne())*

Description

Method to run a tSNE dimensionality reduction algorithm. A tSNE (t-distributed stochastic neighbor embedding) plot is a useful means to visualise data. As it is a dimensionality reduction algorithm, some data will be lost. It is good practice to validate any populations (namely through manual gating). Output data will be "tsne.res". Uses the R package "Rtsne" to calculate plots.

Usage

```
run.tsne(dat, use.cols, tsne.x.name, tsne.y.name, tsne.seed,
dims, initial_dims, perplexity, theta, check_duplicates, pca,
max_iter, verbose, is_distance, Y_init, stop_lying_iter,
mom_switch_iter, momentum, final_momentum, eta, exaggeration_factor)
```

Arguments

dat	NO DEFAULT. data.frame.
use.cols	NO DEFAULT. Vector of numbers, reflecting the columns to use for dimensionality reduction.
tsne.x.name	DEFAULT = "tSNE_X". Character. Name of tSNE x-axis.
tsne.y.name	DEFAULT = "tSNE_Y". Character. Name of tSNE y-axis.
tsne.seed	DEFAULT = 42. Numeric. Seed value for reproducibility.
dims	DEFAULT = 2. Number of dimensions for output results, either 2 or 3.
initial_dims	DEFAULT = 50. Number of dimensions retained in initial PCA step.
perplexity	DEFAULT = 30.
theta	DEFAULT = 0.5. Use 0.5 for Barnes-Hut tSNE, 0.0 for exact tSNE (takes longer).
check_duplicates	DEFAULT = FALSE.
pca	DEFAULT = TRUE. Runs PCA prior to tSNE run.
max_iter	DEFAULT = 1000. Maximum number of iterations.
verbose	DEFAULT = TRUE.
is_distance	DEFAULT = FALSE. Experimental, using X as a distance matrix.

`Y_init` DEFAULT = NULL. Recommend NULL for random initialisation.

`stop_lying_iter`
DEFAULT = 250. Number of iterations of early exaggeration.

`mom_switch_iter`
DEFAULT = 250. Number of iterations before increased momentum of spread.

`momentum` DEFAULT = 0.5. Initial momentum of spread.

`final_momentum` DEFAULT = 0.8. Momentum of spread at 'final_momentum'.

`eta` DEFAULT = 200. Learning rate.

`exaggeration_factor`
DEFAULT = 12.0. Factor used during early exaggeration.

Author(s)

Felix Marsh-Wakefield, <felix.marsh-wakefield@sydney.edu.au>

Examples

```
# Run tSNE on a subset of the  demonstration dataset

cell.dat <- do.subsample(Spectre::demo.clustered, 10000) # Subsample the demo dataset to 10000 cells
cell.dat <- Spectre::run.tsne(dat = cell.dat,
                              use.cols = c("NK11_asinh", "CD3_asinh",
                              "CD45_asinh", "Ly6G_asinh", "CD11b_asinh",
                               "B220_asinh", "CD8a_asinh",
                               "Ly6C_asinh", "CD4_asinh"))
```

---

run.umap *Run UMAP*

Description

Function to run a UMAP dimensionality reduction algorithm. UMAP (uniform manifold approximation and projection) plot is a useful means to visualise data. As it is a dimensionality reduction algorithm, some data will be lost. It is good practice to validate any populations (namely through manual gating). For more information on parameter choices, see ?umap::umap.defaults. Uses the R package "umap" to calculate plots and "data.table" to handle data.

Usage

```
run.umap(dat, use.cols, umap.x.name = "UMAP_X",
umap.y.name = "UMAP_Y", umap.seed = 42, neighbours = 15, n_components = 2,
metric = "euclidean", n_epochs = 200, input = "data", init = "spectral",
min_dist = 0.1, set_op_mix_ratio = 1, local_connectivity = 1,
bandwidth = 1, alpha = 1, gamma = 1, negative_sample_rate = 5,
a_gradient = NA, b_gradient = NA, spread = 1, transform_state = 42,
knn.repeats = 1, verbose = TRUE, umap_learn_args = NA, fast = TRUE,
n_threads = "auto", n_sgd_threads = "auto", batch = TRUE)
```

Arguments

dat	NO DEFAULT. Input data.table or data.frame.
use.cols	NO DEFAULT. Vector of column names or numbers for clustering.
umap.x.name	DEFAULT = "UMAP_X". Character. Name of UMAP x-axis.
umap.y.name	DEFAULT = "UMAP_Y". Character. Name of UMAP y-axis.
umap.seed	DEFAULT = 42. Numeric. Seed value for reproducibility.
neighbours	DEFAULT = 15. Numeric. Number of nearest neighbours.
n_components	DEFAULT = 2. Numeric. Number of dimensions for output results.
metric	DEFAULT = "euclidean". Character or function. Determines how distances between data points are computed. Can also be "manhattan".
n_epochs	DEFAULT = 200. Numeric. Number of iterations performed during layout optimisation.
input	DEFAULT = "data". Character. Determines whether primary input argument is a data or distance matrix. Can also be "dist".
init	DEFAULT = "spectral". Character or matrix. Default "spectral" computes an initial embedding using eigenvectors of the connectivity graph matrix. Can also use "random" (creates an initial layout based on random coordinates).
min_dist	DEFAULT = 0.1. Numeric. Determines how close points appear in final layout.
set_op_mix_ratio	DEFAULT = 1. Numeric in range 0,1. Determines who the knn-graph is used to create a fuzzy simplicial graph.
local_connectivity	DEFAULT = 1. Numeric. Used during construction of fuzzy simplicial set.
bandwidth	DEFAULT = 1. Numeric. Used during construction of fuzzy simplicial set.
alpha	DEFAULT = 1. Numeric. Initial value of "learning rate" of layout optimisation.
gamma	DEFAULT = 1. Numeric. Together with alpha, it determines the learning rate of layout optimisation.
negative_sample_rate	DEFAULT = 5. Numeric. Determines how many non-neighbour points are used per point and per iteration during layout optimisation.
a_gradient	DEFAULT = NA. Numeric. Contributes to gradient calculations during layout optimisation. When left at NA, a suitable value will be estimated automatically.
b_gradient	DEFAULT = NA. Numeric. Contributes to gradient calculations during layout optimisation. When left at NA, a suitable value will be estimated automatically.
spread	DEFAULT = 1. Numeric. Used during automatic estimation of a_gradient/b_gradient parameters.
transform_state	DEFAULT = 42. Numeric. Seed for random number generation used during predict().
knn.repeats	DEFAULT = 1. Numeric. Number of times to restart knn search.
verbose	DEFAULT = TRUE. Logical. Determines whether to show progress messages.

umap_learn_args
: DEFAULT = NA. Vector. Vector of arguments to python package umap-learn.

fast
: DEFAULT TRUE Whether to run uwot implementation of UMAP which is much faster.

n_threads
: DEFAULT "auto". Numeric. Number of threads to use (except during stochastic gradient descent). For nearest neighbor search, only applies if `nn_method = "annoy"`. If `n_threads > 1`, then the Annoy index will be temporarily written to disk in the location determined by tempfile. The default "auto" option will automatically set this to the maximum number of threads in the computer - 1.

n_sgd_threads
: DEFAULT "auto". Number of threads to use during stochastic gradient descent. If set to > 1, then be aware that if `batch = FALSE`, results will not be reproducible, even if `set.seed` is called with a fixed seed before running. Set to "auto" to use the same value as n_threads.

batch
: DEFAULT TRUE. If set to TRUE, then embedding coordinates are updated at the end of each epoch rather than during the epoch. In batch mode, results are reproducible with a fixed random seed even with n_sgd_threads > 1, at the cost of a slightly higher memory use. You may also have to modify learning_rate and increase n_epochs, so whether this provides a speed increase over the single-threaded optimization is likely to be dataset and hardware-dependent.

Author(s)

Thomas Ashhurst, <thomas.ashhurst@sydney.edu.au> Felix Marsh-Wakefield, <felix.marsh-wakefield@sydney.edu.

Examples

```
# Run UMAP on a subset of the  demonstration dataset

library(Spectre)
# Subsample the demo dataset to 10000 cells
cell.dat <- do.subsample(demo.clustered, 10000)
cell.dat$UMAP_X <- NULL
cell.dat$UMAP_Y <- NULL

cell.dat <- run.umap(dat = cell.dat,
                               use.cols = c("NK11_asinh", "CD3_asinh",
                               "CD45_asinh", "Ly6G_asinh", "CD11b_asinh",
                               "B220_asinh", "CD8a_asinh", "Ly6C_asinh",
                               "CD4_asinh"))
```

---

spatial-class *spatial object*

Description

spatial object

`train.cytonorm` *train.cytonorm - Prepare reference data into a FlowSOM object*

Description

This function allows you to learn the conversions to use for batch alignment.

Usage

```
train.align()
```

Arguments

`method`	DEFAULT = 'cytonorm'. In future additional methods for batch alignment will be added.
`cytonorm.goal`	DEFAULT = 'mean'. Target values for alignment. Can be 'mean' for the average of all batches, or a specific batch can be selected.
`cytonorm.nQ`	DEFAULT = 101. Number of quantiles.
`dir`	DEFAULT = getwd(). Sets the working directory to operate from. Because this function involves some reading/writing of files, it's best to set this to somewhere static in case the active working directory moves to a subfolder, and then doesn't return because the function runs into an error.
`mem.ctrl`	DEFAULT = TRUE. Allows the function to clear held memory on occasion.
`dat`	NO DEFAULT. A data.table consisting of the 'refernece' data you will use to train the alignment algorithm
`cellular.cols`	NO DEFAULT. A vector of column names from the data.table that contain the markers to be aligned

Value

Returns the alignment model object, with an additional 'conversions' element added.

Author(s)

Thomas M Ashhurst, <thomas.ashhurst@sydney.edu.au>

References

Ashhurst, T. M., et al. (2019). https://www.ncbi.nlm.nih.gov/pubmed/31077106

train.knn.classifier *train.knn.classifier - Train KNN classifier*

Description

train.knn.classifier - Train KNN classifier

Usage

```
train.knn.classifier(dat, use.cols, label.col,min.num.neighbours,
max.num.neighbours, evaluation.metric, num.folds, num.repeats)
```

Arguments

dat	NO DEFAULT. data.frame. A dataframe containing cells (rows) vs features/markers (columns) to be used to train k-nearest neighbour (KNN) classifier
use.cols	NO DEFAULT. Vector of column names to use for training k-nearest neighbour (KNN) classifier.
label.col	NO DEFAULT. Character. Name of the column representing the population name of the cells in dat.
min.num.neighbours	DEFAULTS to 1. Numeric. When using a k-nearest neighbour (KNN) classifier, this parameter specifies the minimum number of nearest neighbours used to train the KNN classifier.
max.num.neighbours	DEFAULTS to 1. Numeric. When using a k-nearest neighbour (KNN) classifier, this parameter specifies the maximum number of nearest neighbours used to train the KNN classifier.
method	DEFAULTS to random. Can either be random which randomly shuffle data and split them into 2 halves, one for training, the other for testing. Or CV (cross validation) where data is split into num.folds complementary portions, and num.folds-1 portions used for training and the remaining for testing.
num.folds	DEFAULTS to 10. Numeric. Number of chunks the training data is to be split into. The classifier will then take 1 chunk for testing the classifier (after it's trained), and use the remaining chunk to train the classifier.
num.repeats	DEFAULTS to 1. Numeric. Number of time the training data will be split into num.folds chunks. If you set this to 3 and num.folds to 10, the classifier will split data into 10 chunks, train classifier on those chunks 10 times, and repeat the entire procedure 3 times (each time differet data will be in each chunk).
seed	DEFAULTS to 42. Seed used when splitting data into training and testing set.

Train a k-nearest neighbour (KNN) classifier. The classifier will be trained on a number of neighbours, starting from min.num.neighbours, and increased gradually by 1 until max.num.neighbours is reached. For each number of neighbour, the accuracy (or other evaluation metric as specified in evaluation.metric) of the classifier will be computed. Data will be normalised to range between 0 and 1

before used in training the classifier. The normalisation method used is Min-MaxScaling method. NOTE: the larger the training data, the longer it takes to train the classifier. Hence please be mindful of the training data size.

Value

The performance of the classifier on different number of neighbours as well as some description on how the training was performed i.e. what sampling process is used (how the data is split for training and testing), the sample sizes (number of data points used for training/testing). Included as well is the recommended number of neighbours for the data and the reasoning behind why that number of neighbours is the best.

Author(s)

Givanna Putri

References

https://sydneycytometry.org.au/spectre.

`write.files` *write.files - Write .csv or .fcs files*

Description

write.files - Write .csv or .fcs files

Usage

```
write.files(dat, file.prefix, divide.by, write.csv, write.fcs)
```

Arguments

`dat`	Dataframe. No default.
`file.prefix`	Character prefix to add before filename.
`divide.by`	Character. Do you want to write the whole dataset (use NULL) or split it by sample/group/cluster etc (etner the name of the column to divide by, e.g. "SampleName"). Default NULL. .
`write.csv`	Defaults to TRUE. Logical. TRUE to write CSV files, FALSE to not.
`write.fcs`	Defaults to FALSE. Logical. TRUE to write FCS files, FALSE to not.
	This function writes CSV files from your data using fwrite. Can also write FCS files.

write.hdf5 *write.hdf5*

Description

write.hdf5

Usage

```
write.hdf5(
  dat,
  channels = NULL,
  merge.channels = NULL,
  q.max = 0.99,
  flip.y = TRUE,
  random.crop.x = NULL,
  random.crop.y = NULL,
  random.crop.x.seed = 42,
  random.crop.y.seed = 21,
  crop.coordinates = NULL,
  print.spatial.plot = TRUE,
  chunk.size = NULL,
  compression = 0,
  plots = FALSE
)
```

Arguments

dat	NO DEFAULT. Spatial data list
channels	DEFAULT = NULL. Channels to include. If NULL, all channels included.
merge.channels	DEFAULT = NULL. Channels to use to create an additional 'merged' channel.
q.max	DEFAULT = 0.99
flip.y	DEFAULT = TRUE
random.crop.x	DEFAULT = NULL
random.crop.y	DEFAULT = NULL
random.crop.x.seed	DEFAULT = 42
random.crop.y.seed	DEFAULT = 21
print.spatial.plot	DEFAULT = TRUE
chunk.size	DEFAULT = NULL
compression	DEFAULT = 0
plots	DEFAULT = FALSE

Author(s)

Thomas M Ashhurst, <thomas.ashhurst@sydney.edu.au>

References

https://github.com/ImmuneDynamics/Spectre.

Index

∗ **datasets**
    demo.clustered, 7
0,1, *60*

adjust.files, 3

check.files, 4
create.dt, 4
create.stats, 5
create.sumtable, 6

demo.clustered, 7
divide.by, *23*
do.add.cols, 8
do.add.masks, 9
do.add.percell, 10
do.aggregate, 10
do.align, 11
do.asinh, 12
do.calculate.area, 13
do.clip, 13
do.combine.cols, 14
do.create.outlines, 15
do.extract, 15
do.extract.cell.dat, 16
do.filter, 16
do.filter.percell, 17
do.label.from.polygon, 17
do.list.summary, 17
do.list.switch, 18
do.logicle, 19
do.merge.files, 20
do.pull.data, 21
do.reorder, 21
do.rescale, 22
do.stats, 22
do.subsample, 23
do.zscore, 24

fast.colour.plot, 24
fast.multi.plot, 27
fftRtsne, *46*

make.autograph, 29
make.colour.plot, 30
make.multi.plot, 32
make.pheatmap, 34
make.spatial.plot, 36
make.volcano.plot, 38
maxloc, *47*
minloc, *47*

package.check, 38
package.install, 39
package.load, 40
prep.cytonorm, 41

read.files, 42
read.spatial.files, 43
run.align, 44
run.cytonorm, 45
run.fitsne, 46
run.flowsom, 48
run.harmony, 50
run.knn.classifier, 51
run.pca, 52
run.phenograph, 55
run.rpca, 56
run.spatial.analysis, 57
run.tsne, 58
run.umap, 59

spatial *(*spatial-class*)*, 61
spatial-class, 61

train.cytonorm, 62
train.knn.classifier, 63

write.files, 64
write.hdf5, 65