Package 'digitalDLSorteR'

September 7, 2020

Type Package

Title Deconvolution of bulk-RNAseq samples based on Deep Neural Network

Version 0.1.0

Author Author

Maintainer `<yourself@somewhere.net>`

Description digitalDLSorteR is an R package that implements a Deep Learning
based method to enumerate and quantify the cell type composition of bulk
RNA-seq samples. Our method makes use of Deep Neural Network (DNN) models
to adjust any cell type composition starting from single-cell RNA-seq
(scRNA-seq) data.

License What license is it under?

Encoding UTF-8

LazyData true

Depends R (>= 3.4.0)

Imports Matrix,
methods,
tidyr,
SingleCellExperiment,
SummarizedExperiment,
BiocParallel,
splatter,
zinbwave,
DelayedArray,
DelayedMatrixStats,
stats,
HDF5Array,
yardstick,
pbapply,
S4Vectors,
dplyr,
tools,
reshape2,
gtools,
edgeR,
keras,
tensorflow,
ggplot2,

ggpubr,
RColorBrewer

Suggests BiocStyle,
knitr,
rmarkdown,
testthat

RoxygenNote 7.1.1

Roxygen list(markdown = TRUE)

Collate 'AllClasses.R'
'AllGenerics.R'
'data.R'
'digitalDLSorteR.R'
'dnnModel.R'
'evalMetrics.R'
'loadData.R'
'simBulk.R'
'simSingleCell.R'
'utils.R'

VignetteBuilder knitr

R topics documented:

`barErrorPlot` *Generate bar error plot and its dispersion by cell types or by number of different cell types in test bulk samples.*

Description

Generate bar error plot and its dispersion by cell types (`CellType`) or by number of different cell types (`nMix`) in test bulk samples.

Usage

```
barErrorPlot(
  object,
  error,
  by,
  dispersion = "se",
  filter.sc = TRUE,
  title = NULL,
  angle = 90,
  theme = theme_grey()
)
```

Arguments

`object`	`DigitalDLSorter` object with `trained.model` slot containing metrics in `eval.stats.samples` slot.
`error`	MAE or MSE. By default it is used a list of custom colors provided by the package.
`by`	Variable used to display errors.
`dispersion`	Standard error (`'se'`) or standard deviation (`'sd'`). The first by default.
`filter.sc`	Boolean indicating if filter single-cell profiles and only display correlations of results associated with bulk samples (`TRUE` by default).
`title`	Title of the plot.
`angle`	Angle of ticks.
`theme`	ggplot theme.

See Also

`calculateEvalMetrics` `corrExpPredPlot` `distErrorPlot` `blandAltmanLehPlot`

Examples

```
barErrorPlot(
  object = DDLSSmallCompleted,
  error = "MSE",
  by = "CellType"
)

barErrorPlot(
  object = DDLSSmallCompleted,
  error = "MAE",
  by = "nMix"
)
```

`barPlotCellTypes` *Plot a bar plot with deconvoluted cell type proportions.*

Description

This function allows to plot a bar plot with the deconvoluted cell type proportions of a given bulk RNA-seq sample using ggplot2.

Usage

```
barPlotCellTypes(
  data,
  colors,
  simplify = NULL,
  color.line = NA,
  x.label = "Bulk samples",
  rm.x.text = FALSE,
  title = "Results of deconvolution",
  legend.title = "Cell types",
  angle = 90,
  ...
)

## S4 method for signature 'DigitalDLSorter'
barPlotCellTypes(
  data,
  colors = NULL,
  name.data = NULL,
  simplify = NULL,
  color.line = NA,
  x.label = "Bulk samples",
  rm.x.text = FALSE,
  title = "Results of deconvolution",
  legend.title = "Cell types",
  angle = 90
)

## S4 method for signature 'ANY'
barPlotCellTypes(
  data,
  colors,
  color.line = NA,
  x.label = "Bulk samples",
  rm.x.text = FALSE,
  title = "Results of deconvolution",
  legend.title = "Cell types",
  angle = 90
)
```

Arguments

`data`	`DigitalDLSorter` object with `deconv.results` slot or `data.frame`/`matrix` with cell types as columns and samples as rows.
`colors`	Vector with colors that will be used.
`simplify`	Type of simplification performed during deconvolution. It can be `simpli.set` or `simpli.maj` (`NULL` by default). It is only for `DigitalDLSorter` object.
`color.line`	Color of border bars.
`x.label`	Label of x axis.
`rm.x.text`	Logical value indicating if remove x axis ticks (names of samples).
`title`	Title of plot.
`legend.title`	Title of legend plot.
`angle`	Angle of text ticks.
`...`	Other arguments for specific methods.
`name.data`	If a DigitalDLSorter is given, name of the element that stores the results in `deconv.results` slot. If not, forget it.

See Also

`deconvDigitalDLSorter` `deconvDigitalDLSorterObj`

Examples

```
## Using a matrix
## Not run: barPlotCellTypes(deconvResults)
## Using a DigitalDLSorter object
barPlotCellTypes(DDLSSmallCompleted, name.data = "TCGA.breast")
```

---

blandAltmanLehPlot *Generate Bland-Altman agreement plot between predicted and expected cell type proportions from test samples.*

Description

Generate Bland-Altman agreement plot between predicted and expected cell type proportions from test samples. The Bland-Altman agreement plots can be displayed all mixed or split based on cell type (`CellType`) or the number of cell types present in the sample (`nMix`). See `facet.by` argument and examples for more information.

Usage

```
blandAltmanLehPlot(
  object,
  colors,
  color.by,
  facet.by = NULL,
  log.2 = FALSE,
  filter.sc = TRUE,
  density = TRUE,
  color.density = "darkblue",
  size.point = 0.05,
  alpha.point = 1,
  ncol = NULL,
  nrow = NULL,
  title = NULL,
  theme = theme_grey(),
  ...
)
```

Arguments

`object`	`DigitalDLSorter` object with `trained.model` slot containing metrics in `eval.stats.samples` slot.
`colors`	Vector of colors to use. Only vectors with a number of colors equal to or greater than the levels of `color.by` will be accepted. By default it is used a list of custom colors provided by the package.
`color.by`	Variable used to color data. The options are `nMix` and `CellType`.
`facet.by`	Variable used to display data in different panels. If it is `NULL`, the plot is not separated into different panels. The options are `nMix` (by number of different cell types) and `CellType` (by cell type).
`log.2`	If show Bland-Altman agreement plot in log2 space (`FALSE` by default).

`filter.sc`	Boolean indicating if filter single-cell profiles and only display correlations of results associated with bulk samples (`TRUE` by default).
`density`	Boolean indicating if show density lines (`TRUE` by default).
`color.density`	Color of density lines if `density` argument is equal to `TRUE`.
`size.point`	Size of points (0.1 by default).
`alpha.point`	Alpha of points (0.1 by default).
`ncol`	Number of columns if `facet.by` is different than `NULL`.
`nrow`	Number of rows if `facet.by` is different than `NULL`.
`title`	Title of the plot.
`theme`	ggplot theme.
`...`	Additional argument for `facet_wrap` ggplot function if `facet.by` is not equal to `NULL`.

See Also

`calculateEvalMetrics` `corrExpPredPlot` `distErrorPlot` `barErrorPlot`

Examples

```
## Bland-Altman plot by cell type
blandAltmanLehPlot(
  object = DDLSSmallCompleted,
  facet.by = "CellType",
  color.by = "CellType"
)
## Bland-Altman plot of all samples mixed
blandAltmanLehPlot(
  object = DDLSSmallCompleted,
  facet.by = NULL,
  color.by = "CellType",
  alpha.point = 0.3,
  log2 = TRUE
)
```

`breast.chung.generic` *Pre-trained DigitalDLSorter DNN model for deconvolution of TILs present in breast cancer environment (generic version).*

Description

DigitalDLSorter DNN model built and trained with single-cell data from Chung et al., 2017 (GSE75688). This model allows the enumeration and quantification of immune infiltrated cell types in breast cancer environment. This data set consists in single-cell profiles from 11 patients from different tumor etiology and stages (see Torroja and Sanchez-Cabo, 2019 for more details). The analysis and characterization of the cells was carried out by the authors of `digitalDLSorteR` package.

Usage

```
breast.chung.generic
```

Format

A `DigitalDLSorterDNN` object with the following slots:

model Trained DNN model.

training.history Evolution of metrics and loss function during training.

eval.stats Metrics and loss results on test data.

predict.results Predictions of cell types on test data.

cell.types Cell types considered by DNN model.

features Features (genes) considered by model.

Details

The cell types considered in this model are 7. They are the generic groups from cell types considered in specific version: B cells, T CD4+ cells, T CD8+ cells, monocytes, dendritic cells, stromal cells and tumor cells.

The genes considered are 23.260 in SYMBOL notation.

The model consists in 2 hidden layers with 200 neurons per layer trained with 'kullback_leibler_divergence' loss function batch size equal to 128 and a number of epochs equal to 25.

Source

`https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE75688`

References

Chung, W., Eum, H. H., Lee, H. O., Lee, K. M., Lee, H. B., Kim, K. T., et al. (2017). Single-cell RNA-seq enables comprehensive tumour and immune cell profiling in primary breast cancer. Nat. Commun. 8 (1), 15081. doi: `10.1038/ncomms15081`.

Torroja, C. y Sánchez-Cabo, F. (2019). digitalDLSorter: A Deep Learning algorithm to quantify immune cell populations based on scRNA-Seq data. Frontiers in Genetics 10, 978. doi: `10.3389/fgene.2019.00978`

`breast.chung.specific` *Pre-trained DigitalDLSorter DNN model for deconvolution of TILs present in breast cancer environment (specific version).*

Description

DigitalDLSorter DNN model built and trained with single-cell data from Chung et al., 2017 (GSE75688). This model allows the enumeration and quantification of immune infiltrated cell types in breast cancer environment. This data set consists in single-cell profiles from 11 patients from different tumor etiology and stages (see Torroja and Sanchez-Cabo, 2019 for more details). The analysis and characterization of cells was carried out by the authors of `digitalDLSorteR` package.

Usage

```
breast.chung.specific
```

Format

A `DigitalDLSorterDNN` object with the following slots:

model Trained DNN model.

training.history Evolution of metrics and loss function during training.

eval.stats Metrics and loss results on test data.

predict.results Predictions of cell types on test data.

cell.types Cell types considered by DNN model.

features Features (genes) considered by model.

Details

The cell types considered in this model are 13, four of them being the intrinsic molecular subtypes of breast cancer: ER+, HER2+, ER+/HER2+, TNBC, Stromal, Monocyte, TCD4mem (memory CD4+ T cells), BGC (germinal center B cells), Bmem (memory B cells), DC (dendritic cells), Macrophage, TCD8 (CD8+ T cells) and TCD4reg (regulatory CD4+ T cells).

The genes considered are 23.260 in Symbol notation.

The model consists in 2 hidden layers with 200 neurons per layer trained with 'kullback_leibler_divergence' loss function batch size equal to 128 and a number of epochs equal to 25.

Source

`https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE75688`

References

Chung, W., Eum, H. H., Lee, H. O., Lee, K. M., Lee, H. B., Kim, K. T., et al. (2017). Single-cell RNA-seq enables comprehensive tumour and immune cell profiling in primary breast cancer. Nat. Commun. 8 (1), 15081. doi: `10.1038/ncomms15081`.

Torroja, C. y Sánchez-Cabo, F. (2019). digitalDLSorter: A Deep Learning algorithm to quantify immune cell populations based on scRNA-Seq data. Frontiers in Genetics 10, 978. doi: `10.3389/fgene.2019.00978`

`bulk.sim` *Get and set* `bulk.sim` *slot in a* `DigitalDLSorter` *object.*

Description

Get and set `bulk.sim` slot in a `DigitalDLSorter` object.

Usage

```
bulk.sim(object, type.data = "both")

bulk.sim(object, type.data = "both") <- value
```

Arguments

`object`	A `DigitalDLSorter` object.
`type.data`	Element of the list. Can be 'train', 'test' or 'both' (the last by default).
`value`	A `list` with two elements, train and test, each one being a `SummarizedExperiment` object with simulated bulk RNA-Seq samples.

`calculateEvalMetrics` *Calculate evaluation metrics for bulk RNA-seq samples from test data.*

Description

Calculate evaluation metrics for bulk RNA-seq samples from test data in order to know the performance of the model. By default, absolute error (AbsErr), proportional absolute error (ppAbsErr), squared error (SqrErr) and proportional squared error (ppSqrErr) are calculated for each test sample. Moreover, each one of these metrics are aggregated using their mean values by three criteria: each cell type (`CellType`), probability bins of 0.1 (`pBin`), number of different cell types present in the sample `nMix` and a combination of `pBin` and `nMix` (`pBinNMix`). Finally, the process is repeated only for bulk samples, removing single-cell profiles from the evaluation. Evaluation metrics are available in `eval.stats.samples` slot of `DigitalDLSorterDNN` object (`trained.model` of `DigitalDLSorter` object).

Usage

```
calculateEvalMetrics(object, metrics = c("MAE", "MSE"))
```

Arguments

`object`	`DigitalDLSorter` object with `single.cell.final` and `DigitalDLSorterDNN` slots.
`metrics`	Metrics used for evaluating the performance of the model. Mean absolute error (MAE) and mean squared error (MSE) by default.

Value

A `DigitalDLSorter` object with `trained.model` slot containing a `DigitalDLSorterDNN` object with `eval.stats.samples` slot.

See Also

`distErrorPlot` `corrExpPredPlot` `blandAltmanLehPlot` `barErrorPlot`

Examples

```
DDLSSmallCompleted <- calculateEvalMetrics(
  object = DDLSSmallCompleted
)
```

`cell.names` *Get and set* `cell.names` *slot in a* `ProbMatrixCellTypes` *object.*

Description

Get and set `cell.names` slot in a `ProbMatrixCellTypes` object.

Usage

```
cell.names(object)

cell.names(object) <- value
```

Arguments

`object`	A `ProbMatrixCellTypes` object.
`value`	`matrix` object with bulk samples as rows and cells that will be used for simulating these samples as columns (`n.cell` argument)

`cell.types` *Get and set* `cell.types` *slot in a* `DigitalDLSorterDNN` *object.*

Description

Get and set `cell.types` slot in a `DigitalDLSorterDNN` object.

Usage

```
cell.types(object)

cell.types(object) <- value
```

Arguments

`object`	A `DigitalDLSorterDNN` object.
`value`	A `vector` with cell types considered by DNN model.

corrExpPredPlot *Generate correlation plot between predicted and expected cell type proportions from test samples.*

Description

Generate correlation plot between predicted and expected cell type proportions from test samples. The correlation plots can be displayed all mixed or split based on cell type (`CellType`) or the number of cell types present in the sample (`nMix`). See `facet.by` argument and examples for more information. Moreover, a correlation value selected by user is displayed as annotation on the plots. See `corr` argument for details.

Usage

```
corrExpPredPlot(
  object,
  colors,
  facet.by = NULL,
  color.by = "CellType",
  corr = "both",
  filter.sc = TRUE,
  pos.x.label = 0.01,
  pos.y.label = 0.95,
  sep.labels = 0.15,
  size.point = 0.1,
  alpha.point = 1,
  ncol = NULL,
  nrow = NULL,
  title = NULL,
  theme = theme_grey(),
  ...
)
```

Arguments

`object`	`DigitalDLSorter` object with `trained.model` slot containing metrics in `eval.stats.samples` slot.
`colors`	Vector of colors to use. Only vectors with a number of colors equal to or greater than the levels of `color.by` will be accepted. By default it is used a list of custom colors provided by the package.
`facet.by`	Variable used to display data in different panels. If it is `NULL`, the plot is not separated into different panels. The options are `nMix` (by number of different cell types) and `CellType` (by cell type).
`color.by`	Variable used to color data. The options are `nMix` and `CellType`.
`corr`	Correlation value displayed as annotation. The available metrics are Pearson's correlation coefficient (`'pearson'`) and concordance correlation coefficient (`'ccc'`). The argument can be equal to `'pearson'`, `'ccc'` or `'both'` (by default).
`filter.sc`	Boolean indicating if filter single-cell profiles and only display correlations of results associated with bulk samples (`TRUE` by default).

`pos.x.label`	Position on the X axis of the errors annotations. 0.95 by default.
`pos.y.label`	Position on the Y axis of the errors annotations. 0.1 by default.
`sep.labels`	Space separating annotations if `corr` is equal to `'both'`. 0.15 by default.
`size.point`	Size of points (0.1 by default).
`alpha.point`	Alpha of points (0.1 by default).
`ncol`	Number of columns if `facet.by` is different than `NULL`.
`nrow`	Number of rows if `facet.by` is different than `NULL`.
`title`	Title of the plot.
`theme`	ggplot theme.
`...`	Additional argument for `facet_wrap` ggplot function if `facet.by` is not equal to `NULL`.

See Also

`calculateEvalMetrics` `distErrorPlot` `blandAltmanLehPlot` `barErrorPlot`

Examples

```
## correlations by cell type
corrExpPredPlot(
  object = DDLSSmallCompleted,
  facet.by = "CellType",
  color.by = "CellType",
  corr = "both"
)
## correlations of all samples mixed
corrExpPredPlot(
  object = DDLSSmallCompleted,
  facet.by = NULL,
  color.by = "CellType",
  corr = "ccc",
  pos.x.label = 0.2,
  alpha.point = 0.3
)
```

`DDLSChungSmall` `DigitalDLSorter` *'toy' object.*

Description

`DigitalDLSorter` 'toy' object containing a subset from the original data set used for generating `breast.chung.generic` and `breast.chung.specific` models in order to show some examples in vignette and documentation. Moreover, it contains the corresponding `ZinbParams` object in `zinb.params` slot. Data is stored as a `SingleCellExperiment` object with counts in `assay` slot, cells metadata in `colData` slot and genes metadata in `rowData` slot.

Usage

```
DDLSChungSmall
```

Format

An object of class `DigitalDLSorter` of length 1.

Details

For more information about the complete data set, see `breast.chung.generic` or `breast.chung.specific`.

Source

https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE75688

References

Chung, W., Eum, H. H., Lee, H. O., Lee, K. M., Lee, H. B., Kim, K. T., et al. (2017). Single-cell RNA-seq enables comprehensive tumour and immune cell profiling in primary breast cancer. Nat. Commun. 8 (1), 15081. doi: 10.1038/ncomms15081.

Torroja, C. y Sánchez-Cabo, F. (2019). digitalDLSorter: A Deep Learning algorithm to quantify immune cell populations based on scRNA-Seq data. Frontiers in Genetics 10, 978. doi: 10.3389/fgene.2019.00978

`DDLSSmallCompleted` `DigitalDLSorter` *'toy' object (completed version)*

Description

`DigitalDLSorter` 'toy' object containing a subset from the original data set used for generating `breast.chung.generic` and `breast.chung.specific` models in order to show some examples in vignette and documentation. Moreover, it contains the corresponding `ZinbParams` object in `zinb.params` slot. Data is stored as a `SingleCellExperiment` object with counts in `assay` slot, cells metadata in `colData` slot and genes metadata in `rowData` slot. This version contains all steps of pipeline.

Usage

```
DDLSSmallCompleted
```

Format

An object of class `DigitalDLSorter` of length 1.

Details

For more information about the complete data set, see `breast.chung.generic` or `breast.chung.specific`.

Source

https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE75688

References

Chung, W., Eum, H. H., Lee, H. O., Lee, K. M., Lee, H. B., Kim, K. T., et al. (2017). Single-cell RNA-seq enables comprehensive tumour and immune cell profiling in primary breast cancer. Nat. Commun. 8 (1), 15081. doi: `10.1038/ncomms15081`.

Torroja, C. y Sánchez-Cabo, F. (2019). digitalDLSorter: A Deep Learning algorithm to quantify immune cell populations based on scRNA-Seq data. Frontiers in Genetics 10, 978. doi: `10.3389/fgene.2019.00978`

`deconv.data` *Get and set* `deconv.data` *slot in a* `DigitalDLSorter` *object.*

Description

Get and set `deconv.data` slot in a `DigitalDLSorter` object.

Usage

```
deconv.data(object, name.data = NULL)

deconv.data(object, name.data = NULL) <- value
```

Arguments

`object`	A `DigitalDLSorter` object.
`name.data`	Name of the data. If it is `NULL` (by default), all data contained in `deconv.data` slot are returned.
`value`	A `list` whose names are the reference of the data stored.

`deconv.results` *Get and set* `deconv.results` *slot in a* `DigitalDLSorter` *object.*

Description

Get and set `deconv.results` slot in a `DigitalDLSorter` object.

Usage

```
deconv.results(object, name.data = NULL)

deconv.results(object, name.data = NULL) <- value
```

Arguments

`object`	A `DigitalDLSorter` object.
`name.data`	Name of the data. If it is `NULL` (by default), all results contained in `deconv.results` slot are returned.
`value`	A `list` whose names are the reference of the results stored.

deconvDigitalDLSorter

Deconvolute bulk gene expression samples (bulk RNA-Seq) using a pre-trained DigitalDLSorter model.

Description

Deconvolute bulk gene expression samples (RNA-Seq) quantifying the proportion of cell types present in a bulk sample. See in Details the available models. This method uses a pre-trained Deep Neural Network model to enumerate and quantify the cell types present in bulk RNA-Seq samples. For the moment, the available models allow to deconvolute the immune infiltration breast cancer (Chung et al., 2017) at two levels: specific cell types (`'breast.chung.specific'`) and generic cell types (`'breast.chung.generic'`). See `breast.chung.generic` and `breast.chung.specific` documentation for details.

Usage

```
deconvDigitalDLSorter(
  data,
  model = "breast.generic",
  batch.size = 128,
  normalize = TRUE,
  simplify.set = NULL,
  simplify.majority = NULL,
  verbose = TRUE
)
```

Arguments

`data`	A `matrix` or a `data.frame` with bulk gene expression of samples. Rows must be genes in symbol notation and columns must be samples.
`model`	Pre-trained DNN model to use for deconvoluting process. For the moment, the available models are for RNA-Seq samples from breast cancer (`'breast.chung.generic'` and `'breast.chung.specific'`) environment.
`batch.size`	Number of samples loadad in-memory each time of deconvolution process. If unspecified, `batch.size` will default to 128.
`normalize`	Normalize data before deconvolution. `TRUE` by default.
`simplify.set`	List specifying which cell types should be compressed into a new label whose name will be the list name item. See examples for details.
`simplify.majority`	List specifying which cell types should be compressed into the cell type with greater proportions in each sample. Unlike `simplify.set`, it allows to maintain the complexity of the results while compressing the information, because it is not created a new label.
`verbose`	Show informative messages during the execution.

Details

This function is oriented for users that only want to use the method for deconvoluting their bulk RNA-Seq samples. For users that are building their own model from scRNA-seq, see `deconvDigitalDLSorterObj` The former works with base classes, while the last uses `DigitalDLSorter` objects.

For situations where there are cell types exclusive to each other because it does not make sense that they appear together, see arguments `simplify.set` and `simplify.majority`.

Value

A `data.frame` with samples (i) as rows and cell types (j) as columns. Each entry represents the predicted proportion of j cell type in i sample.

References

Chung, W., Eum, H. H., Lee, H. O., Lee, K. M., Lee, H. B., Kim, K. T., et al. (2017). Single-cell RNA-seq enables comprehensive tumour and immune cell profiling in primary breast cancer. Nat. Commun. 8 (1), 15081. doi: 10.1038/ncomms15081.

See Also

`deconvDigitalDLSorterObj`

Examples

```
## to ensure compatibility
tensorflow::tf$compat$v1$disable_eager_execution()
results1 <- deconvDigitalDLSorter(
  data = TCGA.breast.small,
  model = "breast.chung.specific",
  normalize = TRUE
)

## simplify arguments
simplify <- list(Tumor = c("ER+", "HER2+", "ER+/HER2+", "TNBC"),
                 Bcells = c("Bmem", "BGC"))

## in this case,  the item names from list will be the new labels
results2 <- deconvDigitalDLSorter(
  TCGA.breast.small,
  model = "breast.chung.specific",
  normalize = TRUE,
  simplify.set = simplify)

## in this case, the cell type with greatest proportion will be the new label
## the rest of proportion cell types will be added to the greatest
results3 <- deconvDigitalDLSorter(
  TCGA.breast.small,
  model = "breast.chung.specific",
  normalize = TRUE,
  simplify.majority = simplify)
```

deconvDigitalDLSorterObj

Deconvolute bulk gene expression samples (bulk RNA-Seq).

Description

Deconvolute bulk gene expression samples (bulk RNA-Seq) enumerating and quantifying the proportion of cell types present in a bulk sample. This function needs a `DigitalDLSorter` object with a trained DNN model (`trained.model` slot) and bulk samples for deconvoluting in `deconv.data` slot.

Usage

```
deconvDigitalDLSorterObj(
  object,
  name.data,
  batch.size = 128,
  normalize = TRUE,
  simplify.set = NULL,
  simplify.majority = NULL,
  verbose = TRUE
)
```

Arguments

`object`	`DigitalDLSorter` object with `trained.data` and `deconv.data` slots.
`name.data`	Name of the data store in `DigitalDLSorter` object. If it is not provided, the first data set will be used.
`batch.size`	Number of samples per gradient update. If unspecified, `batch.size` will default to 128.
`normalize`	Normalize data before deconvolution. `TRUE` by default.
`simplify.set`	List specifying which cell types should be compressed into a new label whose name will be the list item. See examples for details. The results are stored in a list with normal and simpli.majority results (if provided). The name of the element in the list is `'simpli.set'`.
`simplify.majority`	List specifying which cell types should be compressed into the cell types with greater proportion in each sample. Unlike `simplify.set`, it allows to maintain the complexity of the results while compressing the information, because it is not created a new label. The results are stored in a list with normal and simpli.set results (if provided). The name of the element in the list is `'simpli.majority'`.
`verbose`	Show informative messages during the execution.

Details

This function is oriented for users that have trained a DNN model using their own data. If you want to use a pre-trained model, see `deconvDigitalDLSorter`.

Value

A `data.frame` with samples (i) as rows and cell types (j) as columns. Each entry represents the proportion of j cell type in i sample.

References

Torroja, C. y Sánchez-Cabo, F. (2019). digitalDLSorter: A Deep Learning algorithm to quantify immune cell populations based on scRNA-Seq data. Frontiers in Genetics 10, 978. doi: `10.3389/fgene.2019.00978`

See Also

`trainDigitalDLSorterModel` `DigitalDLSorter`

Examples

```
## to ensure compatibility
## Not run:
tensorflow::tf$compat$v1$disable_eager_execution()
## simplify arguments
simplify <- list(Tumor = c("ER+", "HER2+", "ER+ and HER2+", "TNBC"),
                 Bcells = c("Bmem", "BGC"))

## all results are stored in DigitalDLSorter object
DDLSSmallCompleted <- deconvDigitalDLSorterObj(
  object = DDLSSmallCompleted,
  name.data = "TCGA.breast",
  simplify.set = simplify,
  simplify.majority = simplify
)

## End(Not run)
```

`digitalDLSorteR` *digitalDLSorteR: R package for deconvolution of bulk RNA-Seq samples based on Deep Learning.*

Description

digitalDLSorteR is an R package that implements a Deep Learning based method to enumerate and quantify the cell type composition of bulk RNA-Seq samples. Our method makes use of Deep Neural Network (DNN) models to adjust any cell type composition starting from single-cell RNA-Seq (scRNA-Seq) data.

Details

The rationale of the method consists in a process that starts from scRNA-Seq data and, after a few steps, a Deep Neural Network (DNN) model is trained with simulated bulk RNA-seq samples whose cell composition is known. The trained model is able to deconvolve any bulk RNA-seq sample by determining the proportion of the different cell types present in it. The main advantage of this method is the possibility of building deconvolution models trained with real data which comes from certain biological environments. For example, for quantifying the proportion of tumor infiltrated lymphocytes (TILs) in breast cancer, by following this protocol you can obtain a specific model for this type of samples. This fact overcomes the limitation of other methods, since stromal and immune cells change significantly their profiles depending on the tissue and disease context.

The package can be used in two ways: for deconvolving bulk RNA-seq samples using a pre-trained model provided by us or for building your own models trained from your own scRNA-seq samples. These new models may be published in order to make them available for other users that work with similar data (e.g. neural environment, prostate cancer environment, etc.). For the moment, the available models allows the deconvolution of TILs from breast cancer classified by our team.

`DigitalDLSorter-class` *The DigitalDLSorter Class.*

Description

The DigitalDLSorter object is the core of digitalDLSorteR. This object stores the different intermediate data resulting from running pipeline from real single-cell data to the trained Deep Neural Network, including the data on which to carry out the process of devonvolution. Only it is used in the case of building new deconvolution models. For deconvoluting bulk samples using pre-trained models, see `deconvDigitalDLSorter` function.

Details

This object uses other classes to store the different type of data produced during the process:

- `SingleCellExperiment` class for single-cell RNA-seq data, using sparse matrix from the `Matrix` package (`dgCMatrix` class) to store the matrix of counts.
- `ZinbParams` class with the estimated parameters for the simulation of new single-cell profiles.
- `SummarizedExperiment` class for storing bulk RNA-seq data. In this case, it is possible to load all data in memory or the use of HDF5 files as back-end by `DelayedArray` and `HDF5Array` packages. See `generateBulkSamples` for details.
- `ProbMatrixCellTypes` class for the composition cell matrices built during the process. See `?ProbMatrixCellTypes` for details.
- `DigitalDLSorterDNN` class for storing the trained Deep Neural Network. This step is performed by `keras`. See `DigitalDLSorterDNN` for details.

Slots

`single.cell.real` Real single-cell data stored in a `SingleCellExperiment` object. The counts matrix is stored as a `dgCMatrix` object to optimize the amount of used memory.

`zinb.params` `ZinbParams` object with estimated parameters for the simulation of new single-cell expression profiles.

`single.cell.final` Final single-cell expression profiles used for simulating bulk RNA-seq profiles with known cell composition.

`prob.cell.types` `ProbMatrixCellTypes` class with the cell composition matrix built for the simulation of bulk RNA-seq profiles. The entries determine the proportion of single-cell types that will constitute the simulated bulk samples.

`bulk.sim` A list with two elements: train and test simulated bulk RNA-seq. This data are stored as a `SummarizedExperiment` object. We recommend the use of HDF5 file as a back-end due to the large amount of memory that they occupy.

`final.data` The final data that will be used for training and testing the Deep Neural Network. As in the previous slot, it is a list with two items, train and test. With respect to train counts matrix, it can be the train bulk RNA-seq samples, the train scRNA-seq samples or a combination of both. In the case of test counts matrix, RNA-seq data from bulk and single-cell will be combined. Moreover, data is scaled and shuffled for training.

`trained.model` `DigitalDLSorterDNN` object with the trained model, different metrics obtained during the training and evaluation metrics from the application of the model on test data. After executing `calculateEvalMetrics`, it is alto possible to find the results of the model evaluation.

`deconv.data` Optional slot where is possible to store new bulk samples for its deconvolution. It is a list whose name is the name of the data provided. It is possible to store more than one dataset to make predictions. See `deconvDigitalDLSorterObj` for details.

`deconv.results` Slot where the results from the deconvolution process over `deconv.data` data are stored. It is a list whose name is the name of the data from which they come.

`project` Name of the project.

`version` Version of DigitalDLSorteR this object was built under.

The package can be used in two ways: to build new models of deconvolution from scRNA-seq data or to deconvolute bulk RNA-seq samples using pre-trtained models integrated into the package. If you want to build new models, see `loadRealSCProfiles` or `loadFinalSCProfiles` functions. If yoy want to use pre-trained models, see `deconvDigitalDLSorter` function.

`DigitalDLSorterDNN-class`

The DigitalDLSorterDNN Class.

Description

The DigitalDLSorterDNN object stores the trained Deep Neural Network, the training history of selected metrics and the results of prediction on test data. After executing `calculateEvalMetrics`, it is alto possible to find the results of the model evaluation.

Details

The steps related with Deep Learning are carried out with `keras` package, so the model are stored in a R6 class, system used by the package. If you want to save the object in an rds file, `digitalDLSorteR` provides an `saveRDS` generic that transforms the keras model into a native valid R object. Specifically, the model is converted into a list with the architecture of the network and the weights learned during the training. The is the minimum information to use the model as predictor. If you want to maintain the optimizer state, see `saveTrainedModelAsH5` function. If you want to store an object as rda file, see `preparingToSave` function.

Slots

`model` Trained Deep Neural Network model. This slot can contain a R6 `keras.engine.sequential.Sequenti` object or a list with two elements: the architecture of the model and the resulting weights after training.

`training.history` List with the evolution of the selected metrics during training.

`eval.stats.model` Performance of the model on test data.

`predict.results` Deconvolution results matrix of test data. Columns are cell types, rows are samples and each entry is the proportion of this cell type on this sample.

`cell.types` Vector with the cell types to deconvolute.

`features` Vector with features used during training. These features will be used for the following predictions.

`eval.stats.samples` Performance of the model on each sample of test data in comparison with the known cell proportions.

`distErrorPlot` *Generate box plot or violin plot showing how errors are distributed.*

Description

Generate violin plot or box plot showing how errors are distributed by proportion bins of 0.1. The errors can be displayed all mixed or split based on cell type (`CellType`) or number of cell types present in the sample (`nMix`). See `facet.by` argument and examples for more information.

Usage

```
distErrorPlot(
  object,
  error,
  colors,
  x.by = "pBin",
  facet.by = NULL,
  color.by = "nMix",
  filter.sc = TRUE,
  error.labels = FALSE,
  pos.x.label = 4.6,
  pos.y.label = NULL,
  size.point = 0.1,
  alpha.point = 1,
  type = "violinplot",
  ylimit = NULL,
  nrow = NULL,
  ncol = NULL,
  title = NULL,
  theme = theme_grey(),
  ...
)
```

Arguments

`object`	`DigitalDLSorter` object with `trained.model` slot containing metrics in `eval.stats.samples` slot.
`error`	Which error is going to represent. The available errors are absolute error (`"AbsErr"`), proportional absolute error (`"ppAbsErr"`), squared error (`"SqrErr"`) or proportional squared error (`"ppSqrErr"`).

`colors`	Vector of colors to use. Only vectors with a number of colors equal to or greater than the levels of `color.by` will be accepted. By default it is used a list of custom colors provided by the package.
`x.by`	Variable used for x axis. When `facet.by` is not `NULL`, the best option is `pBin` (probability bin). The options are `nMix` (by number of different cell types), `CellType` (by cell type) and `pBin`.
`facet.by`	Variable used to display data in different panels. If it is `NULL`, the plot is not separated into different panels. The options are `nMix` (by number of different cell types) and `CellType` (by cell type).
`color.by`	Variable used to color data. The options are `nMix` and `CellType`.
`filter.sc`	Boolean indicating if filter single-cell profiles and only display errors associated with bulk samples (`TRUE` by default).
`error.labels`	Boolean indicating if show average error as annotation.
`pos.x.label`	Position on the X axis of the errors annotations.
`pos.y.label`	Position on the Y axis of the errors annotations.
`size.point`	Size of points (0.1 by default).
`alpha.point`	Alpha of points (0.1 by default).
`type`	Type of plot, `'boxplot'` or `'violinplot'`. The last by default.
`ylimit`	Upper limit in y axis if it is needed. `NULL` by default.
`nrow`	Number of rows if `facet.by` is different than `NULL`.
`ncol`	Number of columns if `facet.by` is different than `NULL`.
`title`	Title of the plot.
`theme`	ggplot theme.
`...`	Additional argument for `facet_wrap` ggplot function if `facet.by` is not equal to `NULL`.

See Also

`calculateEvalMetrics` `corrExpPredPlot` `blandAltmanLehPlot` `barErrorPlot`

Examples

```
distErrorPlot(
  object = DDLSSmallCompleted,
  error = "AbsErr",
  facet.by = "CellType",
  color.by = "nMix",
  error.labels = TRUE
)

distErrorPlot(
  object = DDLSSmallCompleted,
  error = "AbsErr",
  x.by = "CellType",
  facet.by = NULL,
  filter.sc = FALSE,
  color.by = "CellType",
  error.labels = TRUE
)
```

`estimateZinbwaveParams` *Estimate parameters for ZINB-WaVE model for simulating new single-cell expression profiles.*

Description

Estimate parameters for the ZINB-WaVE model from a real single-cell data set using ZINB-WaVE model.

Usage

```
estimateZinbwaveParams(
  object,
  cell.ID.column,
  gene.ID.column,
  cell.type.column,
  cell.cov.columns,
  gene.cov.columns,
  set.type = "All",
  threads = 1,
  verbose = TRUE
)
```

Arguments

`object`	`DigitalDLSorter` object with a `single.cell.real` slot.
`cell.ID.column`	Name or number of the column in cells metadata corresponding with cell names in expression matrix.
`gene.ID.column`	Name or number of the column in genes metadata corresponding with the notation used for features/genes.
`cell.type.column`	Name or number of the column in cells metadata corresponding with cell type of each cell.
`cell.cov.columns`	Name or number of columns in cells metadata that will be used as covariates in the model during the estimation.
`gene.cov.columns`	Name or number of columns in genes metadata that will be used as covariates in the model during estimation.
`set.type`	Cell type to evaluate. 'All' by default.
`threads`	Number of threads used for the estimation. For setting the parallel environment `BiocParallel` package is used.
`verbose`	Show informative messages during the execution.

Details

ZINB-WaVE is a flexible model for zero-inflated count data. This function carries out the model fit to real single-cell data modeling Y_{ij} (the count of feature j for sample i) as a random variable following a zero-inflated negative binomial (ZINB) distribution. The estimated parameters will be used for the simulation of new single-cell expression profiles by sampling a negative binomial distribution and introducing dropouts from a binomial distribution. To do this, `DigitalDLSorter` uses `zinbEstimate` function from `splatter` package (Zappia et al., 2017), that is a wrapper around `zinbFit` function from `zinbwave` package (Risso et al., 2018). For more details about the model, see Risso et al., 2018.

Value

A `DigitalDLSorter` object with `zinb.params` slot containing a `ZinbParams` object. This object contains the estimated ZINB parameters from real single-cell data.

References

Risso, D., Perraudeau, F., Gribkova, S. et al. (2018). A general and flexible method for signal extraction from single-cell RNA-seq data. Nat Commun 9, 284. doi: `doi.org/10.1038/s41467-017-02554-5`.

Torroja, C. y Sánchez-Cabo, F. (2019). digitalDLSorter: A Deep Learning algorithm to quantify immune cell populations based on scRNA-Seq data. Frontiers in Genetics 10, 978. doi: `10.3389/fgene.2019.00978`

Zappia, L., Phipson, B. y Oshlack, A. Splatter: simulation of single-cell RNA sequencing data. Genome Biol. 2017; 18: 174.

See Also

`simSingleCellProfiles`

Examples

```
## Not run:
DDLSSmallCompleted <- estimateZinbwaveParams(
  object = DDLSSmallCompleted,
  cell.ID.column = "Cell_ID",
  gene.ID.column = "external_gene_name",
  cell.type.column = "Cell_type",
  cell.cov.columns = c("Patient", "Sample_type"),
  gene.cov.columns = "gene_length",
  verbose = TRUE
)

## End(Not run)
```

`eval.stats.model` *Get and set* `eval.stats.model` *slot in a* `DigitalDLSorterDNN` *object.*

Description

Get and set `eval.stats.model` slot in a `DigitalDLSorterDNN` object.

Usage

```
eval.stats.model(object)

eval.stats.model(object) <- value
```

Arguments

`object`	A `DigitalDLSorterDNN` object.
`value`	A `list` object with the resulting metrics after prediction on test data with DNN model.

`eval.stats.samples` *Get and set* `eval.stats.samples` *slot in a* `DigitalDLSorterDNN` *object.*

Description

Get and set `eval.stats.samples` slot in a `DigitalDLSorterDNN` object.

Usage

```
eval.stats.samples(object, metrics = "All")

eval.stats.samples(object, metrics = "All") <- value
```

Arguments

`object`	A `DigitalDLSorterDNN` object.
`metrics`	Metrics to show (`'All'` by default)
`value`	A `list` with evaluation metrics used for evaluating the performance of the model over each sample from test data.

`exclusive.types` *Get and set* `exclusive.types` *slot in a* `ProbMatrixCellTypes` *object.*

Description

Get and set `exclusive.types` slot in a `ProbMatrixCellTypes` object.

Usage

```
exclusive.types(object)

exclusive.types(object) <- value
```

Arguments

`object`	A `ProbMatrixCellTypes` object.
`value`	Optional slot that contains the exclusive cell types on the experiment if they are provided. NULL by default.

`features` *Get and set* `features` *slot in a* `DigitalDLSorterDNN` *object.*

Description

Get and set `features` slot in a `DigitalDLSorterDNN` object.

Usage

```
features(object)

features(object) <- value
```

Arguments

`object`	A `DigitalDLSorterDNN` object.
`value`	A `vector` with features (genes) considered by DNN model.

`final.data` *Get and set* `final.data` *slot in a* `DigitalDLSorter` *object.*

Description

Get and set `final.data` slot in a `DigitalDLSorter` object.

Usage

```
final.data(object, type.data = "both")

final.data(object, type.data = "both") <- value
```

Arguments

`object`	A `DigitalDLSorter` object.
`type.data`	Element of the list. Can be 'train', 'test' or 'both' (the last by default).
`value`	A `list` with two elements, train and test, each one being a `SummarizedExperiment` object with simulated bulk RNA-Seq samples prepared for training. This samples have been normalized and shuffled.

`generateBulkSamples` *Generate training and test simulated bulk RNA-seq samples.*

Description

Generate training and test bulk profiles using the cell composition matrix built by `generateTrainAndTestBulkPro` function. These samples are generated using the assumption that the expression of gene i in sample j is given by the sum of the cell type specific expression X_{ijk} weighted by the proportions of cell type k in the sample determined by the probability matrix. In practice, as described in Torroja et al., 2019, these profiles are generated by the summation of 100 cells from different cell types determined by cell composition matrix. The number of bulk samples is determined by dimensions of cell composition matrix. See `generateTrainAndTestBulkProbMatrix` for details.

Usage

```
generateBulkSamples(
  object,
  type.data = "both",
  file.backend = NULL,
  threads = 1,
  compression.level = NULL,
  verbose = TRUE
)
```

Arguments

`object`	`DigitalDLSorter` object with `single.cell.final` and `prob.cell.types` slots.
`type.data`	Type of data to generate among 'train', 'test' or 'both' (the last by default).
`file.backend`	Valid file path where to save the HDF5 file used as backend. If it is equal to `NULL` (by default), the data are produced and loaded in memory.
`threads`	Number of threads used during the generation of bulk samples (2 by default).
`compression.level`	The compression level used if file.backend provided (6 by default). It is an integer value between 0 (no compression) and 9 (highest and slowest compression).
`verbose`	Show informative messages during the execution.

Details

`digitalDLSorteR` allows the use of HDF5 files as back-end for the resulting data using `DelayedArray` and `HDF5Array` packages in cases of generating too large bulk expression matrix. This functionality allows you to work without keeping the data loaded in memory, which will be of vital importance during some computationally heavy steps such as neural network training. You must provide a valid file path in `file.backend` argument to store the resulting file with '.h5' extension. The data will be accessible from R without being loaded into memory. This option slightly slows down execution times, since subsequent transformations of data will be carried out by chunks instead of using all data. We recommend this option due to the large size of the simulated matrices.

Value

A `DigitalDLSorter` object with `bulk.sim` slot containing a list with one or two entries (depending on selected `type.data` argument): 'train' and 'test'. Each entry contains a `SummarizedExperiment` object with simulated bulk samples in `assay` slot, sample names in `colData` slot and feature names in `rowData` slot.

References

Pagès H, Hickey wcfP, Lun A (2020). DelayedArray: A unified framework for working transparently with on-disk and in-memory array-like datasets. R package version 0.14.1.

Pagès H (2020). HDF5Array: HDF5 backend for DelayedArray objects. R package version 1.16.1.

See Also

`generateTrainAndTestBulkProbMatrix` `ProbMatrixCellTypes`

Examples

```
## loading all data in memory
DDLSSmallCompleted <- generateBulkSamples(
  DDLSSmallCompleted,
  threads = 2,
  type.data = "both"
)
## Not run:
## using HDF5 as backend
DDLSChung <- generateBulkSamples(
  DDLSChung,
  threads = 2,
  type.data = "both",
  file.backend = "DDLSChung.bulk.sim.h5"
)

## End(Not run)
```

`generateTrainAndTestBulkProbMatrix`
Generate training and test cell composition matrix.

Description

Generate training and test cell composition matrices for the simulation of bulk samples with known cell composition using single-cell expression profiles. The resulting matrix will determine the proportion of the different cell types that will form the simulated bulk samples.

Usage

```
generateTrainAndTestBulkProbMatrix(
  object,
  cell.type.column,
  prob.design,
  proportions.train = c(10, 5, 20, 15, 10, 40),
  proportions.test = c(10, 5, 20, 15, 10, 40),
  train.freq = 2/3,
  n.cells = 100,
  num.bulk.samples = NULL,
  exclusive.types = NULL,
  verbose = TRUE
)
```

Arguments

`object`	`DigitalDLSorter` object with `single.cell.real` and `zinb.params` slots.
`cell.type.column`	Name or number of the column in cells metadata corresponding with the cell type of each cell.
`prob.design`	`data.frame` with the frequency ranges expected for each cell type present in the experiment. This information can be estimated from literature or from the single-cell experiment itself. This `data.frame` must be built by three columns with specific headers: • A cell type column with the same name of the cell type column in cells.metadata. If the name of the column is not the same, function returns an error. Cell types must appear on cells.metadata. • A second column named 'from' with the start frequency for each cell type. • A third column named 'to' with the final frequency for each cell type.

`proportions.train`	Vector of five integer numbers that determine the proportions of bulk samples that will be generated by the methods explained in details in train samples. This vector represents proportions, so they must add 100 and none can be less than 1. By default, a majority of random samples without using predefined ranges will be generated.
`proportions.test`	As `proportions.train` for test samples.
`train.freq`	Proportion of cells used for training set (2/3 by default).
`n.cells`	Number of cells that are aggregated in order to simulate one bulk RNA-seq sample (100 by default).
`num.bulk.samples`	Integer which allows to establish the number of bulk samples that will be generated taking into account training and test data. If it is NULL (by default), approximately 18 more samples will be formed than there are cells in `single.cell.final` slot.
`exclusive.types`	Vector of cell types which allows to establish cell types that biologically do not make sense to be mixed during the generation of bulk samples. Some samples presents this exclusive cell types. If it is equal to NULL (by default), all cell types will be mixed when generating bulk samples.
`verbose`	Show informative messages during the execution.

Details

First of all, simulated single-cell profiles are split into training and test subsets (2/3 for training and 1/3 for test by default). Then, to avoid biases due to the composition of bulk samples, proportions for the mixtures (bulk samples) of cell types ($w_1, ..., w_k$, where k is the number of cell types available in single-cell profiles), are randomly generated using five different approaches:

1. Cell proportions are randomly sampled from a truncated uniform distribution with predefined limits according to a priori knowledge of the abundance of each cell type (see `prob.design` argument). This information ban be inferred from the single cell analysis itself or from the literature.
2. A second set is generated by randomly permuting cell type labels from a distribution generated by the previous method.
3. Cell proportions are randomly sampled as by method 1 without replacement.
4. Using the last method for generating proportions, cell types labels are randomly sampled.
5. Cell proportions are randomly sampled from a Dirichlet distribution.

If you want to see the distribution of cell type proportions generated by each method during the process, you can access them with `showProbPlot` function (see examples).

It is important to note that the number of bulk-samples simulated are determined in this step. You can predefine the number of bulk profiles generated using `num.bulk.samples` argument. By default, the number of bulk samples generated depends on the number of single-cell profiles available: approximately 18 more samples will be formed than there are cells in `single.cell.final`. We recommend set a number of 30000 samples. This number will be split in training and test data: 60% for training and 40% for evaluating the model.

Value

A DigitalDLSorter object with prob.cell.types slot containing a ProbMatrixCellTypes object. For more information about the structure of this class, see ProbMatrixCellTypes. The most important element is the cell composition matrix, which is formed by n rows (being n the number of bulk samples that will be generated) and k columns (being k the number of cell types present in the experiment).

References

Torroja, C. y Sánchez-Cabo, F. (2019). digitalDLSorter: A Deep Learning algorithm to quantify immune cell populations based on scRNA-Seq data. Frontiers in Genetics 10, 978. doi: 10.3389/fgene.2019.00978

See Also

generateBulkSamples ProbMatrixCellTypes

Examples

```
## generate a data.frame with frequency ranges of each cell type
probMatrix <- data.frame(
  Cell_type = c("ER+", "HER2+", "ER+ and HER2+", "TNBC",
                "Stromal", "Monocyte", "Tme", "BGC",
                "Bmem", "DC", "Macrophage", "TCD8", "Treg"),
  from = c(rep(30, 4), 1, rep(0, 8)),
  to = c(rep(70, 4), 50, rep(15, 8))
)
DDLSSmallCompleted <- generateTrainAndTestBulkProbMatrix(
  object = DDLSSmallCompleted,
  cell.type.column = "Cell_type",
  prob.design = probMatrix,
  num.bulk.samples = 200,
  verbose = TRUE
)
```

---

getProbMatrix *Getter function for cell composition matrix.*

Description

Getter function for cell composition matrix. This function allows you access to the cell composition matrix of train or test bulk data.

Usage

```
getProbMatrix(object, type.data)
```

Arguments

object	DigitalDLSorter object with prob.cell.types slot.
type.data	Subset of data to show: train or test.

Value

A `matrix` object.

See Also

`generateTrainAndTestBulkProbMatrix`

`loadDeconvDataFromFile` *Load data to deconvolute from tabulated text file.*

Description

Load data to deconvolute from text file. Accepted formats are tsv and tsv.gz. You must specify the correct extension.

Usage

```
loadDeconvDataFromFile(object, file.path, name.data = NULL)
```

Arguments

`object`	`DigitalDLSorter` object with `trained.model` slot.
`file.path`	File path where data is stored.
`name.data`	Name with which the data is stored in `DigitalDLSorter` object. If `name.data` is not provided, base name of file is used.

See Also

`trainDigitalDLSorterModel` `deconvDigitalDLSorterObj`

`loadDeconvDataFromSummarizedExperiment` *Load data to deconvolute from* `SummarizedExperiment` *object.*

Description

Load data in `DigitalDLSorter` object to deconvolute from `SummarizedExperiment` object.

Usage

```
loadDeconvDataFromSummarizedExperiment(object, se.object, name.data = NULL)
```

Arguments

`object`	`DigitalDLSorter` object with `trained.model` slot.
`se.object`	`SummarizedExperiment` object.
`name.data`	Name with which the data is stored in `DigitalDLSorter` object.

See Also

`trainDigitalDLSorterModel` `deconvDigitalDLSorterObj`

`loadFinalSCProfiles` *Load final real scRNA-Seq data into a* `DigitalDLSorter` *object*

Description

Load scRNA-Seq data into a `DigitalDLSorter` from file stored on disk or from a `SingleCellExperiment` object. Provided data must be composed by three pieces of information:

- Single-cell counts: genes in rows and cells in columns.
- Cells metadata: with annotations (columns) for each cell (rows).
- Genes metadata with annotations (columns) for each gene (rows).

In the case that data is provided from files, `single.cell.real` argument must be a vector of three elements ordered so that the first file corresponds to counts, the second to cells metadata and the last to genes metadata. On the other hand, if data is provided as `SingleCellExperiment`, the object must contains single-cell counts in `assay` slot, cells metadata in `colData` slot and genes metadata in `rowData`.

Usage

```
loadFinalSCProfiles(
  single.cell.final,
  cell.ID.column = 1,
  gene.ID.column = 1,
  min.counts = 0,
  min.cells = 0,
  project = "DigitalDLSorterProject"
)
```

Arguments

`single.cell.final`	If data is provided from files, `single.cell.real` must be a vector with three elements: single-cell counts, cells metadata and genes metadata. If data is provided from a `SingleCellExperiment` object, singlecell counts must be in `assay` slot, cells metadata in `colData` and genes metadata in `rowData`.
`cell.ID.column`	Name or number of the column in cells.metadata corresponding with cell names in expression matrix.
`gene.ID.column`	Name or number of the column in genes.metadata corresponding with the notation used for features/genes.
`min.counts`	Minimum gene counts to filter (0 by default).
`min.cells`	Minimum of cells with more than min.counts (0 by default).
`project`	Name of the project for `DigitaDLSorter` object.

Details

#' The difference with `loadFinalSCProfiles` is that data loaded with this functions will be used for estimating ZINB-WaVE parameters and simulating new single-cell profiles in order to increase the signal of cell types. On the other side, `loadFinalSCProfiles` loads data on `single.cell.final` slot, so this scRNA-seq profiles will be used directly for simulating bulk samples. In this case, data must be enough cells for each cell type and enough cells for simulating bulk profiles.

See Also

`simSingleCellProfiles`

Examples

```
sc.chung.breast <- single.cell.real(DDLSChungSmall)
DDLSChungSmallFinal <- loadFinalSCProfiles(
  single.cell.final = sc.chung.breast,
  cell.ID.column = "Cell_ID",
  gene.ID.column = "external_gene_name",
  min.cells = 0,
  min.counts = 0,
  project = "Chung_example"
)
```

`loadRealSCProfiles` *Load real scRNA-Seq data into a* `DigitalDLSorter` *object for simulating new profiles.*

Description

Load scRNA-Seq data into a `DigitalDLSorter` from file stored on disk or from a `SingleCellExperiment` object. Provided data must be composed by three pieces of information:

Usage

```
loadRealSCProfiles(
  single.cell.real,
  cell.ID.column = 1,
  gene.ID.column = 1,
  min.counts = 0,
  min.cells = 0,
  project = "DigitalDLSorterProject"
)
```

Arguments

`single.cell.real`
: If data is provided from files, `single.cell.real` must be a vector with three elements: single-cell counts, cells metadata and genes metadata. If data is provided from a `SingleCellExperiment` object, single-cell counts must be in `assay` slot, cells metadata in `colData` and genes metadata in `rowData`.

`cell.ID.column`	Name or number of the column in cells metadata corresponding with cell names in expression matrix.
`gene.ID.column`	Name or number of the column in genes metadata corresponding with the names used for features/genes.
`min.counts`	Minimum gene counts to filter (0 by default).
`min.cells`	Minimum of cells with more than min.counts (0 by default).
`project`	Name of the project for `DigitalDLSorter` object.

Details

- Single-cell counts: genes in rows and cells in columns.
- Cells metadata: with annotations (columns) for each cell (rows).
- Genes metadata with annotations (columns) for each gene (rows).

In the case that data is provided from files, `single.cell.real` argument must be a vector of three elements ordered so that the first file corresponds to counts, the second to cells metadata and the last to genes metadata. On the other hand, if data is provided as `SingleCellExperiment`, the object must contains single-cell counts in `assay` slot, cells metadata in `colData` slot and genes metadata in `rowData`.

The difference with `loadFinalSCProfiles` is that data loaded with this functions will be used for estimating ZINB-WaVE parameters and simulating new single-cell profiles in order to increase the signal of cell types. On the other side, `loadFinalSCProfiles` loads data on `single.cell.final` slot, so this scRNA-seq profiles will be used directly for simulating bulk samples. In this case, data must be enough cells for each cell type and enough cells for simulating bulk profiles.

See Also

`simSingleCellProfiles`

Examples

```
sc.chung.breast <- single.cell.real(DDLSChungSmall)
DDLSChungSmall <- loadRealSCProfiles(
  single.cell.real = sc.chung.breast,
  cell.ID.column = "Cell_ID",
  gene.ID.column = "external_gene_name",
  min.cells = 0,
  min.counts = 0,
  project = "Chung_example"
)
```

`loadTrainedModelFromH5` *Load from HDF5 file a trained DigitalDLSorter DNN model.*

Description

Load from HDF5 file a trained DigitalDLSorter DNN model into a `DigitalDLSorter` object. Note that HDF5 file must be a keras valid trained model.

Usage

```
loadTrainedModelFromH5(object, file.path, reset.slot = FALSE)
```

Arguments

`object`	`DigitalDLSorter` object with `trained.model` slot.
`file.path`	Valid file path where model are stored.
`reset.slot`	Remove `trained.slot` if it already exists. A new `DigitalDLSorterDNN` object will be formed, but it will not contain other slots (`FALSE` by default).

See Also

`trainDigitalDLSorterModel` `deconvDigitalDLSorterObj` `saveTrainedModelAsH5`

`model` *Get and set* `model` *slot in a* `DigitalDLSorterDNN` *object.*

Description

Get and set `model` slot in a `DigitalDLSorterDNN` object.

Usage

```
model(object)

model(object) <- value
```

Arguments

`object`	A `DigitalDLSorterDNN` object.
`value`	A `keras.engine.sequential.Sequential` object with a trained DNN model.

`plots` *Get and set* `plots` *slot in a* `ProbMatrixCellTypes` *object.*

Description

Get and set `plots` slot in a `ProbMatrixCellTypes` object.

Usage

```
plots(object)

plots(object) <- value
```

Arguments

`object`	A `ProbMatrixCellTypes` object.
`value`	List of lists with plots showing the distribution of cell proportions generated by each method during the process.

`plotTrainingHistory` *Plot training history of a trained DigitalDLSorter DNN model.*

Description

Plot training history of a trained DigitalDLSorter DNN model.

Usage

```
plotTrainingHistory(
  object,
  title = "History of metrics during training",
  metrics = NULL
)
```

Arguments

`object`	`DigitalDLSorter` object with `trained.model` slot.
`title`	Title of plot.
`metrics`	Which metrics to plot. If it is equal to `NULL` (by default), all metrics available on `DigitalDNNSorter` object will be plotted.

See Also

`trainDigitalDLSorterModel` `deconvDigitalDLSorterObj`

`predict.results` *Get and set* `predict.results` *slot in a* `DigitalDLSorterDNN` *object.*

Description

Get and set `predict.results` slot in a `DigitalDLSorterDNN` object.

Usage

```
predict.results(object)

predict.results(object) <- value
```

Arguments

`object`	A `DigitalDLSorterDNN` object.
`value`	A `matrix` object with prediction results on test data.

`prepareDataForTraining`
Prepare training and test final data for training and evaluation Deep Neural Network model.

Description

Prepare training and test final data for training and evaluating Deep Neural Network model. Expression matrix is normalized by CPMs (counts per million) in log2-space and normalized. Samples are shuffled in order to avoid biases during training. Note that expression matrix is transposed in order to prepare data for training.

Usage

```
prepareDataForTraining(
  object,
  type.data,
  combine = "both",
  file.backend = NULL,
  number.rows = NULL,
  compression.level = NULL,
  verbose = TRUE
)
```

Arguments

`object`	DigitalDLSorter object with `single.cell.final` and `prob.cell.types` slots.
`type.data`	Type of data to generate among 'train', 'test' or 'both' (the last by default).
`combine`	Character determining if combine training data. Can be 'both', 'bulk' or 'single-cell' ('both' by default). Note that test data is always combined.
`file.backend`	A valid file path where to save the HDF5 file used as back-end. If it is equal to `NULL` (by default), the data are loaded in memory.
`number.rows`	HDF5 file is saved by row chunks in order to improve the execution times during training. This is because `trainDigitalDLSorterModel` only access to data by rows (samples). You can provided the number of rows that are stored together in each chunk. Note that the more columns the more RAM is used, although execution times are improved.
`compression.level`	The compression level used if file.backend provided (6 by default). It is an integer value between 0 (no compression) and 9 (highest and slowest compression).
`verbose`	Show informative messages during the execution.

Details

This function allows you to select which kind of data you want to use for training: single-cell profiles, bulk profiles or a combination of both. See `combine` argument for details. We recommend the use of the combination or the bulk profiles, since the results are better. For test data, profiles are combined in any case, but during the evaluation of results you can filter single-cell profiles (see `calculateEvalMetrics`).

`digitalDLSorteR` allows the use of HDF5 files as back-end for the resulting data using `DelayedArray` and `HDF5Array` packages in cases of generating too large expression matrix. This functionality allows you to work without keeping the data loaded in memory, which will be of vital importance during some computationally heavy steps such as neural network training. You must provide a valid file path in `file.backend` argument to store the resulting file with '.h5' extension. The data will be accessible from R without being loaded into memory. This option slightly slows down execution times, since subsequent transformations of data will be carried out by chunks instead of using all data. We recommend this option due to the large size of the simulated matrices.

Value

A `DigitalDLSorter` object with `final.data` slot containing a list with one or two entries (depending on selected `type.data` argument): 'train' and 'test'. Each entry contains a `SummarizedExperiment` object with single-cell and bulk samples combined in `assay` slot, sample names in `rowData` slot and feature names in `colData` slot.

See Also

`generateBulkSamples` `generateTrainAndTestBulkProbMatrix`

Examples

```
## loading all data in memory
DDLSSmallCompleted <- prepareDataForTraining(
  object = DDLSSmallCompleted,
  type.data = "both",
  verbose = TRUE
)
## Not run:
## using HDF5 as backend
DDLSChungSmall <- prepareDataForTraining(
  object = DDLSChungSmall,
  type.data = "both",
  combine = "both",
  file.backend = "DDLSChung.final.data.combined.h5",
  verbose = TRUE
)

## End(Not run)
```

`preparingToSave` *Prepare* `DigitalDLSorter` *object for saving as RDA file.*

Description

Prepare a `DigitalDLSorter` object that has a `DigitalDLSorterDNN` object with trained DNN model. `keras` models are not able to be stored natively as R objects (e.g. RData or RDS files). By saving the structure as JSON character object and weights as list object, it is possible recovering the model and carrying out predictions.

Usage

```
preparingToSave(object)
```

Arguments

`object`	`DigitalDLSorter` object with `trained.data` slot.

Details

With this option, the state of optimizer is not saved, only architecture and weights.

It is possible to save completely the model as HDF5 file with `saveTrainedModelAsH5` function and to load into `DigitalDLSorter` object with `loadTrainedModelFromH5` function.

It is also possible to save a `DigitalDLSorter` object as RDS file with `saveRDS` function without any type of previous preparation.

See Also

`saveRDS` `saveTrainedModelAsH5`

`prob.cell.types` *Get and set* `prob.cell.types` *slot in a* `DigitalDLSorter` *object.*

Description

Get and set `prob.cell.types` slot in a `DigitalDLSorter` object.

Usage

```
prob.cell.types(object, type.data = "both")

prob.cell.types(object, type.data = "both") <- value
```

Arguments

`object`	A `DigitalDLSorter` object.
`type.data`	Element of the list. Can be 'train', 'test' or 'both' (the last by default).
`value`	A list with two elements, train and test, each one with a `ProbMatrixCellTypes` object.

`prob.matrix` *Get and set* `prob.matrix` *slot in a* `ProbMatrixCellTypes` *object.*

Description

Get and set `prob.matrix` slot in a `ProbMatrixCellTypes` object.

Usage

```
prob.matrix(object)

prob.matrix(object) <- value
```

Arguments

`object`	A `ProbMatrixCellTypes` object.
`value`	`matrix` object with cell types as columns and samples as rows.

`ProbMatrixCellTypes-class` *The Class ProbMatrixCellTypes.*

Description

The ProbMatrixCellTypes class is a data storage class that stores information concerning to cell composition matrix used for the simulation of bulk samples. This matrix corresponds with `prob.matrix` slot. The rest of slots are additional information generated during the process and required for subsequent steps.

Details

As described in Torroja and Sanchez-Cabo, 2019, the proportions are built by five different methods in order to avoid biases due to the composition of the bulk samples. In `plots` slot, different representations of these probabilities are stored with the aim of offering a method to monitor the different sets of samples generated during the process. These plots can be displayed with `showProbPlot` function. See documentation for details.

Slots

`prob.matrix` Matrix of cell proportions generated for the simulation of bulk samples. Rows correspond with bulk samples which will be generated (i), columns are the cell types present in single-cell data provided (j) and each entry is the proportion of j cell type in i sample.

`cell.names` Matrix in which names of cells that will compound each bulk samples are stored.

`set.list` List of cells ordered according to the cell type to which they belong.

`set` Vector with the names of cells present in the object.

`exclusive.types` Optional slot that contains the exclusive cell types on the experiment if they are provided. NULL by default.

`plots` List of lists with plots showing the distribution of cell proportions generated by each method during the process.

`type.data` Character with the type of data contained: training or test.

References

Torroja, C. y Sánchez-Cabo, F. (2019). digitalDLSorter: A Deep Learning algorithm to quantify immune cell populations based on scRNA-Seq data. Frontiers in Genetics 10, 978. doi: `10.3389/fgene.2019.00978`

`project` *Get and set* `project` *slot in a* `DigitalDLSorter` *object.*

Description

Get and set `project` slot in a `DigitalDLSorter` object.

Usage

```
project(object)

project(object) <- value
```

Arguments

`object`	A `DigitalDLSorter` object.
`value`	A character indicating the name of the project.

`saveRDS` *Save* `DigitalDLSorter` *object as RDS file.*

Description

Save `DigitalDLSorter` and `DigitalDLSorterDNN` objects as RDS files. We developed this generic with the aim of changing the behavior of the base function and saving the structure and weights of DNN model as R native objects. This is because `keras` models are not able to be stored natively as R objects (e.g. RData or RDS files). By saving the structure as JSON character object and weights as list object, it is possible recovering the model and carrying out predictions. If `trained.model` slot is empty, the function will have the usual behavior.

Usage

```
saveRDS(
  object,
  file,
  ascii = FALSE,
  version = NULL,
  compress = TRUE,
  refhook = NULL
)
```

Details

With this option, the state of optimizer is not saved, only architecture and weights. It is possible to save completely the model as HDF5 file with `saveTrainedModelAsH5` function and to load into `DigitalDLSorter` object with `loadTrainedModelFromH5` function.

Moreover, if you want to save the object as RDA file, it is possible by converting the model to an allowed R object with `preparingToSave` function.

See Also

`saveTrainedModelAsH5` `preparingToSave`

`saveTrainedModelAsH5` *Save on disk trained DigitalDLSorter DNN model as HDF5 file.*

Description

Save on disk the trained model in HDF5 format. Note that this function does not save the `DigitalDLSorterDNN` object, but the trained keras model.

Usage

```
saveTrainedModelAsH5(object, file.path, overwrite = FALSE)
```

Arguments

`object`	`DigitalDLSorter` object with `trained.model` slot.
`file.path`	Valid file path where saving the model.
`overwrite`	Overwrite file if it already exists.

See Also

`trainDigitalDLSorterModel` `loadTrainedModelFromH5`

`set` *Get and set* `set` *slot in a* `ProbMatrixCellTypes` *object.*

Description

Get and set `set` slot in a `ProbMatrixCellTypes` object.

Usage

```
set(object)

set(object) <- value
```

Arguments

`object`	A `ProbMatrixCellTypes` object.
`value`	Vector with the names of cells present in the object.

set.list — *Get and set* set.list *slot in a* ProbMatrixCellTypes *object.*

Description

Get and set set.list slot in a ProbMatrixCellTypes object.

Usage

```
set.list(object)

set.list(object) <- value
```

Arguments

object	A ProbMatrixCellTypes object.
value	List of cells ordered according to the cell type to which they belong.

showProbPlot — *Show distribution plots of cell proportions generated by* generateTrainAndTestBulkProbMatrix.

Description

Show distribution plots of cell proportions generated by the five different methods used by generateTrainAndTestB
These frequencies will determine the proportion of different cell types used during the simulation of bulk RNA-Seq samples. There are 6 subsets of proportions (1 and 2 are generated by the same way) by different approaches that can be shown with three types of visualizations: boxplot, violinplot and linesplot. You also can represent the probabilities based on the number of different cell types present in samples setting type.plot = 'nMix'.

Usage

```
showProbPlot(object, type.data, set, type.plot = "boxplot")
```

Arguments

object	DigitalDLSorter object with prob.cell.types slot with plot slot.
type.data	Subset of data to show: train or test.
set	Integer determining which of the 6 different approaches used for generating the probability matrix show. Note that 1 and 2 follow the same distribution.
type.plot	Character determining which type of visualization show. It can be boxplot, violinplot, linesplot or nmix. See Description for more information.

Details

These plots are only for diagnostic purposes. This is the reason because they are generated without any parameter introduced by the user.

Value

`ggplot` object.

See Also

`generateTrainAndTestBulkProbMatrix`

Examples

```
lapply(1:6, function(x) {
  showProbPlot(DDLSSmallCompleted,
               type.data = "train",
               set = x,
               type.plot = "boxplot")
})
```

`simSingleCellProfiles`

Simulate new single-cell expression profiles using the estimated ZINB parameters.

Description

Simulate single-cell expression profiles by randomly sampling from a negative binomial distribution using ZINB parameters estimated by ZINB-WaVE model and introducing dropouts by sampling from a binomial distribution with ZINB-WaVE model estimated.

Usage

```
simSingleCellProfiles(
  object,
  cell.ID.column,
  cell.type.column,
  n.cells,
  verbose = TRUE
)
```

Arguments

`object`	`DigitalDLSorter` object with `single.cell.real` and `zinb.params` slots.
`cell.ID.column`	Name or number of the column in cells metadata corresponding with cell names in expression matrix.
`cell.type.column`	Name or number of the column in cells metadata corresponding with the cell type of each cell.
`n.cells`	Number of simulated cells generated by cell type (i.e. if you have 10 different cell types on your dataset, if `n.cells = 100`, then 1000 cell profiles will be simulated).
`verbose`	Show informative messages during the execution.

Details

Before this step, see `estimateZinbwaveParams`. As described in Torroja and Sanchez-Cabo, 2019, this function simulates a determined number of transcriptional profiles for each cell type provided by randomly sampling from a negative binomial distribution with μ and θ estimated parameters and introducing dropouts by sampling from a binomial distribution with pi probability. All paramteres are estimated from single-cell real data using `estimateZinbwaveParams` function. It uses the ZINB-WaVE model (Risso et al., 2018). For more details about the model, see `estimateZinbwaveParams`.

Value

A `DigitalDLSorter` object with `single.cell.final` slot containing a `SingleCellExperiment` object with the simulated single-cell profiles.

References

Risso, D., Perraudeau, F., Gribkova, S. et al. (2018). A general and flexible method for signal extraction from single-cell RNA-seq data. Nat Commun 9, 284. doi: `10.1038/s41467-017-02554-5`.

Torroja, C. y Sánchez-Cabo, F. (2019). digitalDLSorter: A Deep Learning algorithm to quantify immune cell populations based on scRNA-Seq data. Frontiers in Genetics 10, 978. doi: `10.3389/fgene.2019.00978`

See Also

`estimateZinbwaveParams`

Examples

```
DDLSSmallCompleted <- simSingleCellProfiles(
  object = DDLSSmallCompleted,
  cell.ID.column = "Cell_ID",
  cell.type.column = "Cell_type",
  n.cells = 10,
  verbose = TRUE
)
```

`single.cell.final` *Get and set* `single.cell.final` *slot in a* `DigitalDLSorter` *object.*

Description

Get and set `single.cell.final` slot in a `DigitalDLSorter` object.

Usage

```
single.cell.final(object)

single.cell.final(object) <- value
```

Arguments

`object`	A `DigitalDLSorter` object.
`value`	A `SingleCellExperiment` object with real and simulated single-cell profiles.

`single.cell.real` *Get and set* `single.cell.real` *slot in a* `DigitalDLSorter` *object.*

Description

Get and set `single.cell.real` slot in a `DigitalDLSorter` object.

Usage

```
single.cell.real(object)

single.cell.real(object) <- value
```

Arguments

`object`	A `DigitalDLSorter` object.
`value`	A `SingleCellExperiment` object with real single-cell profiles.

`TCGA.breast.small` *Breast cancer bulk RNA-Seq samples from TCGA Research Network.*

Description

Subset of Breast cancer bulk RNA-Seq samples from TCGA Research Network. FPKMs were transformed in TPMs and aggregated based on SYMBOL genes.

Usage

```
TCGA.breast.small
```

Format

An object of class `matrix` with 44831 rows and 15 columns.

Source

https://www.cancer.gov/tcga

References

Koboldt, D. C., Fulton, R. S., McLellan, M. D., Schmidt, H., Kalicki-Veizer, J., McMichael, J. F., et al. (2012). Comprehensive molecular portraits of human breast tumours. Nature 490 (7418), 61–70. doi: 10.1038/nature11412

Ciriello, G., Gatza, M. L., Beck, A. H., Wilkerson, M. D., Rhie, S. K., Pastore, A., et al. (2015). Comprehensive molecular portraits of invasive lobular breast cancer. Cell. 163 (2), 506–519. doi: 10.1016/j.cell.2015.09.033

Torroja, C. y Sánchez-Cabo, F. (2019). digitalDLSorter: A Deep Learning algorithm to quantify immune cell populations based on scRNA-Seq data. Frontiers in Genetics 10, 978. doi: 10.3389/fgene.2019.00978

`trainDigitalDLSorterModel`
Train `DigitalDLSorter` *Deep Neural Network model.*

Description

Train `DigitalDLSorter` Deep Neural Network model with data store in `final.data` slot. Moreover, model is evaluated on test data and prediction results are produced.

Usage

```
trainDigitalDLSorterModel(
  object,
  batch.size = 128,
  num.epochs = 20,
  val = FALSE,
  freq.val = 0.1,
  loss = "kullback_leibler_divergence",
  metrics = c("accuracy", "mean_absolute_error", "categorical_accuracy"),
  view.metrics.plot = TRUE,
  verbose = TRUE
)
```

Arguments

`object`	`DigitalDLSorter` object with `final.data` slot.
`batch.size`	Number of samples per gradient update. If unspecified, `batch.size` will default to 128.
`num.epochs`	Number of epochs to train the model.
`val`	Boolean that determines if a validation subset is used during training (`FALSE` by default).
`freq.val`	Number between 0.1 and 0.5 that determines the number of samples from training data that will be used as validation subset.
`loss`	Character indicating loss function selected for training the model (Kullback-Leibler divergence by default). Look at keras documentation to see available loss functions.

metrics	Vector of metrics used to evaluate the performance of the model during training and on test data (c("accuracy","mean_absolute_error","categorical_accuracy by default)
view.metrics.plot	Boolean indicating if show progression plots of loss and metrics during training (TRUE by default). keras for R allows to see the progression of the model during training if you are working on RStudio.
verbose	Boolean indicating if show the progression of the model during training. Besides, it is shown information about the architecture of the model (TRUE by default).

Details

All steps related with Deep Neural Network in digitalDLSorteR package are performed by using keras package, an API in R for keras in Python available from CRAN. We recommend use the guide of installation available on https://keras.rstudio.com/ in order to set a custom configuration (type of back-end used, CPU or GPU, etc.).

Although trainDigitalDLSorterModel allows to select a custom loss function used during training, we recommend using Kullback-Leibler divergence because its better results. If you want to know more details about the architecture of the DNN and its construction, see Torroja and Sanchez-Cabo, 2019.

Value

A DigitalDLSorter object with trained.model slot containing a DigitalDLSorterDNN object. For more information about the structure of this class, see DigitalDLSorterDNN.

References

Torroja, C. y Sánchez-Cabo, F. (2019). digitalDLSorter: A Deep Learning algorithm to quantify immune cell populations based on scRNA-Seq data. Frontiers in Genetics 10, 978. doi: 10.3389/fgene.2019.00978

See Also

plotTrainingHistory deconvDigitalDLSorter deconvDigitalDLSorterObj

Examples

```
## to ensure compatibility
tensorflow::tf$compat$v1$disable_eager_execution()
DDLSSmallCompleted <- trainDigitalDLSorterModel(
  object = DDLSSmallCompleted,
  batch.size = 128,
  num.epochs = 5 ## 20
)
```

`trained.model` *Get and set* `trained.model` *slot in a* `DigitalDLSorter` *object.*

Description

Get and set `trained.model` slot in a `DigitalDLSorter` object.

Usage

```
trained.model(object)

trained.model(object) <- value
```

Arguments

`object`	A `DigitalDLSorter` object.
`value`	A `DigitalDLSorterDNN` object.

`training.history` *Get and set* `training.history` *slot in a* `DigitalDLSorterDNN` *object.*

Description

Get and set `training.history` slot in a `DigitalDLSorterDNN` object.

Usage

```
training.history(object)

training.history(object) <- value
```

Arguments

`object`	A `DigitalDLSorterDNN` object.
`value`	A `keras_training_history` object with training history of DNN model

`zinb.params` *Get and set* `zinb.params` *slot in a* `DigitalDLSorter` *object.*

Description

Get and set `zinb.params` slot in a `DigitalDLSorter` object.

Usage

```
zinb.params(object)

zinb.params(object) <- value
```

Arguments

`object`	A `DigitalDLSorter` object.
`value`	A `ZinbParams` object with ZiNB-WaVE parameters estimated from real single-cell profiles.

Index