Package ‘pipeComp’

March 12, 2024

Type Package
Title pipeComp pipeline benchmarking framework
Version 1.12.0
Depends R (>= 4.1)
Description A simple framework to facilitate the comparison of pipelines involving various steps and parameters.
The `pipelineDefinition` class represents pipelines as, minimally, a set of functions consecutively executed on
the output of the previous one, and optionally accompanied by step-wise evaluation and aggregation functions.
Given such an object, a set of alternative parameters/methods, and benchmark datasets, the `run-Pipeline` function
then proceeds through all combinations arguments, avoiding recomputing the same step twice and compiling evaluations
on the fly to avoid storing potentially large intermediate data.
Imports BiocParallel, S4Vectors, ComplexHeatmap, SingleCellExperiment,
    SummarizedExperiment, Seurat, matrixStats, Matrix, cluster,
aricode, methods, utils, dplyr, grid, scales, scran,
    viridisLite, clue, randomcoloR, ggplot2, cowplot,
    intrinsicDimension, scater, knitr, reshape2, stats, Rtsne,
    uwot, circlize, RColorBrewer
Suggests BiocStyle, rmarkdown
License GPL
Encoding UTF-8
RoxygenNote 7.1.1
VignetteBuilder knitr
URL https://doi.org/10.1186/s13059-020-02136-7
BugReports https://github.com/plger/pipeComp
biocViews GeneExpression, Transcriptomics, Clustering,
    DataRepresentation
git_url https://git.bioconductor.org/packages/pipeComp
R topics documented:

pipeComp-package .............................................. 3
addPipelineStep ................................................ 3
aggregatePipelineResults ..................................... 4
buildCombMatrix ............................................... 5
checkPipelinePackages ........................................ 5
clustMetricsCorr ............................................ 6
colCenterScale ............................................... 6
ctrlgenes ....................................................... 7
dea_evalPlot_curve ........................................... 7
dea_pipeline .................................................... 8
defaultStepAggregation ....................................... 8
evalHeatmap ..................................................... 9
evaluateClustering ............................................ 11
evaluateDEA .................................................... 12
evaluateDimRed ............................................... 12
evaluateNorm ................................................... 13exempleDEAresults .......................................... 14
exempleResults ............................................... 14
distancePoint .................................................. 15
canonicalPlot .................................................. 15
canonicalPlot .................................................. 15
match_evaluate_multiple ..................................... 16
mergePipelineResults ......................................... 17
mockPipeline .................................................... 18
parsePipNames ................................................. 18
PipelineDefinition ........................................... 19
PipelineDefinition-methods ................................. 20
plotElapsed .................................................... 22
readPipelineResults .......................................... 23
runPipeline ..................................................... 23
scrna_describeDatasets ...................................... 25
scrna_evalPlot_filtering .................................... 25
scrna_evalPlot_overall ....................................... 26
scrna_evalPlot_silh .......................................... 28
**Description**

*pipeComp* is a simple framework to facilitate the comparison of pipelines involving various steps and parameters. It was initially developed to benchmark single-cell RNA sequencing pipelines, and contains pre-defined *PipelineDefinitions* and functions to that effect, but could be applied to any context. See `vignette("pipeComp")` for an introduction.

**Author(s)**

Pierre-Luc Germain <pierre-luc.germain@hest.ethz.ch>
Anthony Sonrel <anthony.sonrel@uzh.ch>
Mark D. Robinson <mark.robinson@imls.uzh.ch>

---

**addPipelineStep**

**Description**

Add a step to an existing *PipelineDefinition*

**Usage**

`addPipelineStep(object, name, after = NULL, slots = list())`

**Arguments**

- **object**: A *PipelineDefinition*
- **name**: The name of the step to add
- **after**: The name of the step after which to add the new step. If NULL, will add the step at the beginning of the pipeline.
- **slots**: A optional named list with slots to fill for that step (i.e. ‘functions’, ‘evaluation’, ‘aggregation’, ‘descriptions’ - will be parsed)

**Value**

A *PipelineDefinition*
aggregatePipelineResults

See Also

PipelineDefinition, PipelineDefinition-methods

Examples

```r
pd <- mockPipeline()
pd
pd <- addPipelineStep(pd, name="newstep", after="step1",
slots=list(description="Step that does nothing..."))
```

```r
aggregatePipelineResults
```

Description

Aggregates the evaluation and running times of 'runPipeline' results. Results should be indicated either as a 'path' prefix or as a vector of paths to 'evaluation.rds' files ('resfiles').

Usage

```r
aggregatePipelineResults(res, pipDef = NULL)
```

Arguments

- `res`: A (named) list of results (per dataset), as produced by `readPipelineResults` (or 'mergePipelineResults').
- `pipDef`: An optional `PipelineDefinition` containing the aggregation methods. If omitted, that from the results will be used.

Value

A list with a slot for each step for which there is an aggregation method, or (if no aggregation method available) a list of the `stepIntermediateReturnObjects` of 'runPipeline'

Examples

```r
# we produce mock pipeline results:
pip <- mockPipeline()
datasets <- list(ds1=1:3, ds2=c(5,10,15))
tmpdir1 <- paste0(tempdir(), '/')
res <- runPipeline(datasets, pipelineDef=pip, output.prefix=tmpdir1,
alternatives=list() )
# we read the evaluation files:
res <- readPipelineResults(tmpdir1)
# we aggregate the results (equivalent to the output of `runPipeline`):
res <- aggregatePipelineResults(res)
```
**buildCombMatrix**

Description

Builds a matrix of parameter combinations from a list of alternative values.

Usage

buildCombMatrix(alt, returnIndexMatrix = FALSE)

Arguments

alt
A named list of alternative parameter values
returnIndexMatrix
Logical; whether to return a matrix of indices, rather than a data.frame of factors.

Value

a matrix or data.frame

Examples

buildCombMatrix(list(param1=LETTERS[1:3], param2=1:2))

**checkPipelinePackages**

Description

Checks whether the packages required by a pipeline and its alternative methods are available.

Usage

checkPipelinePackages(alternatives, pipDef = NULL)

Arguments

alternatives
A named list of alternative parameter values
pipDef
An object of class ‘PipelineDefinition’.

Value

Logical.

Examples

checkPipelinePackages(list(argument1="mean"), scrna_pipeline())
clustMetricsCorr  Correlations across clustering evaluation metrics

**Description**
A list of two matrices containing, respectively, the Pearson and Spearman pairwise correlations between various clustering evaluation metrics, computed across a wide range of scRNAseq clustering analyses (see reference).

**Value**
a list.

**References**
See https://doi.org/10.1101/2020.02.02.930578

colCenterScale  colCenterScale

**Description**
Matrix scaling by centering columns separately and then performing variance scaling on the whole matrix, in a NA-robust fashion. With the default arguments, the output will be the number of (matrix-)median absolute deviations from the column-median.

**Usage**
colCenterScale(
  x,  
centerFn = median,  
scaleFn = function(x, na.rm) median(abs(x), na.rm = na.rm)
)

**Arguments**

- **x**  A numeric matrix.
- **centerFn**  The function for calculating centers. Should accept the ‘na.rm’ argument. E.g. ‘centerFn=mean’ or ‘centerFn=median’.
- **scaleFn**  The function for calculating the (matrix-wise) scaling factor. Should accept the ‘na.rm’ argument. Default ‘median(abs(x))’.

**Value**
A scaled matrix of the same dimensions as ‘x’. 
Examples

```r
# random data with column mean differences
d <- cbind(A=rnorm(5, 10, 2), B=rnorm(5, 20, 2), C=rnorm(5,30, 2))
colCenterScale(d)
```

ctrlgenes

**Lists of control genes**

**Description**

Lists of mouse and human control genes (mitochondrial, ribosomal, protein-coding), as ensembl gene ids or official symbols, for computing cell QC.

**Value**

a list.

dea_evalPlot_curve

**dea_evalPlot_curve**

**Description**

dea_evalPlot_curve

**Usage**

```r
dea_evalPlot_curve(
  res,
  scales = "free",
  agg.by = NULL,
  agg.fn = mean,
  xlim = c(NA, NA),
  colourBy = "method",
  shapeBy = NULL,
  pointsize = 4
)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>res</td>
<td>Aggregated results of the DEA pipeline</td>
</tr>
<tr>
<td>scales</td>
<td>Passed to ‘facet_grid’</td>
</tr>
<tr>
<td>agg.by</td>
<td>Aggregate results by these columns (default no aggregation)</td>
</tr>
<tr>
<td>agg.fn</td>
<td>Function for aggregation (default mean)</td>
</tr>
<tr>
<td>xlim</td>
<td>Optional vector of x limits</td>
</tr>
</tbody>
</table>
**defaultStepAggregation**

- **colourBy**: Name of column by which to colour
- **shapeBy**: Name of column determining the shape of the points. If omitted, the shape will indicate whether the nominal FDR is below or equal the real FDR.
- **pointsize**: Size of the points

**Value**

A ggplot.

**Examples**

```r
data("exampleDEAresults", package="pipeComp")
dea_evalPlot_curve(exampleDEAresults, agg.by=c("sva.method"))
```

---

**dea_pipeline**

**Description**

The ‘PipelineDefinition’ for bulk RNAseq differential expression analysis (DEA).

**Usage**

```r
dea_pipeline()
```

**Value**

A ‘PipelineDefinition’ object to be used with ‘runPipeline’.

**Examples**

```r
pip <- dea_pipeline()
pip
```

---

**defaultStepAggregation**

**Description**

defaultStepAggregation

**Usage**

```r
defaultStepAggregation(x)
```
**Arguments**

- **x**: A list of results per dataset, each containing a list (1 element per combination of parameters) of evaluation metrics (coercible to vectors or matrix).

**Value**

- A data.frame.

**Description**

General heatmap representation of aggregated evaluation results. By default, the actual metric values are printed in the cells, and while the coloring is determined by `colCenterScale` (number of matrix-median absolute deviations from the column means). Unless the total number of analyses is small, it is strongly recommended to use the ‘agg.by’ argument to limit the size and improve the readability of the heatmap.

**Usage**

```r
evalHeatmap(
  res,
  step = NULL,
  what,
  what2 = NULL,
  agg.by = NULL,
  agg.fn = mean,
  filterExpr = NULL,
  scale = "colCenterScale",
  value_format = "%.2f",
  reorder_rows = FALSE,
  show_heatmap_legend = FALSE,
  show_column_names = FALSE,
  col = NULL,
  font_factor = 0.9,
  row_split = NULL,
  shortNames = TRUE,
  value_cols = c("black", "white"),
  title = NULL,
  name = NULL,
  anno_legend = TRUE,
  ...
)
```
Arguments

res  Aggregated pipeline results (i.e. the output of `runPipeline` or `aggregateResults`)
step  Name of the step for which to plot the evaluation results. If unspecified, will use the latest step that has evaluation results.
what  What metric to plot.
what2  If the step has more than one benchmark data.frame, which one to use. The function will attempt to guess that automatically based on `what`, and will notify in case of ambiguity.
agg.by  Aggregate results by these columns (default no aggregation)
agg.fn  Function for aggregation (default mean)
filterExpr  An optional filtering expression based on the columns of the target dataframe, (e.g. `filterExpr=param1=="value1"`).
scale  Controls the scaling of the columns for the color mapping. Can either be a logical (TRUE will use NA-safe column z-scores, FALSE will not scale) or a function performing the scaling. The default uses the `colCenterScale` function (per-column centering, but per-matrix variance scaling).
value_format  Format for displaying cells’ values (use `value_format=""` to disable)
reorder_rows  Logical; whether to sort rows (default FALSE). The row names themselves can also be passed to specify an order, or a `ComplexHeatmap`.
show_heatmap_legend  Passed to `Heatmap` (default FALSE)
show_column_names  Passed to `Heatmap` (default FALSE)
col  Colors for the heatmap, or a color-mapping function as produced by `colorRamp2`. If passing a vector of colors and the data is scaled, there should be an odd number of colors. By default, will apply linear mapping (if the data is not scaled) or signed sqrt mapping (if scaled) on the `viridisLite::inferno` palette.
font_factor  A scaling factor applied to fontsizes (default 1)
row_split  Optional column (included in `agg.by`) by which to split the rows. Alternatively, an expression using the columns (retained after aggregation) can be passed.
shortNames  Logical; whether to use short row names (with only the parameter values instead of the parameter name and value pairs), default TRUE.
value_cols  A vector of length 2 indicating the colors of the values (above and below the mean), if printed
title  Plot title
name  Heatmap name (e.g. used for the legend)
anno_legend  Logical; whether to plot the legend for the datasets
...  Passed to `Heatmap`

Value

A Heatmap
evaluateClustering

Examples

data("exampleResults", package="pipeComp")

evalHeatmap( exampleResults, step="clustering", what=c("ARI","MI","min_pr"),
agg.by=c("filt","norm"), row_split = "norm" ) +

evalHeatmap( exampleResults, step="clustering", what="ARI",
agg.by=c("filt","norm"), filterExpr=n_clus==true.nbClusts,
name="ARI at true k", title="ARI at true K" )

Description

Evaluates a clustering using ‘true’ labels. Entries with missing true labels (i.e. NA) are excluded from calculations. If using ‘evaluateClustering’ in a custom pipeline, you might want to use the corresponding ‘pipeComp:::.aggregateClusterEvaluation’ aggregation function.

Usage

evaluateClustering(x, tl = NULL)

Arguments

x
The clustering labels

tl
The true labels

Value

A numeric vector of metrics (see the ‘pipeComp_scRNA’ vignette for details)

Examples

# random data
dat <- data.frame(
  cluster=rep(LETTERS[1:3], each=10),
  x=c(rnorm(20, 0), rnorm(10, 1)),
  y=c(rnorm(10, 1), rnorm(20, 0))
)

# clustering
dat$predicted <- kmeans(dist(dat[,-1]),3)$cluster

# evaluation
evaluateClustering(dat$predicted, dat$cluster)
Description
Evaluates a differential expression analysis (DEA).

Usage
```r
evaluateDEA(dea, truth = NULL, th = c(0.01, 0.05, 0.1))
```

Arguments
- `dea` Expects a data.frame with logFC and FDR, as produced by `edgeR::topTags`, `limma::topTable` or `DESeq2::results`.
- `truth` A data.frame containing the columns `expected.beta` (real logFC) and `isDE` (logical indicating whether there is a difference or not; accepts NA values).
- `th` The significance thresholds for which to compute the metrics.

Value
A list with two slots: `logFC` (vector of metrics on logFC) and `significance` table of significance-related statistics.

Examples
```r
# fake DEA results
dea <- data.frame( row.names=paste0("gene",1:10), logFC=rnorm(10) )
dea$PValue <- dea$FDR <- c(2:8/100, 0.2, 0.5, 1)
truth <- data.frame( row.names=paste0("gene",1:10), expected.beta=rnorm(10),
                    isDE=rep(c(TRUE,FALSE,TRUE,FALSE), c(3,1,2,4)) )
evaluateDEA(dea, truth)
```

description
Gathers evaluation statistics on a reduced space using known cell labels. If using `evaluateDimRed` in a custom pipeline, you will probably want to use `pipeComp:::.aggregateDR` as the corresponding aggregation function.

Usage
```r
evaluateDimRed(x, clusters = NULL, n = c(10, 20, 50), covars)
```
evaluateNorm

Arguments

x
The matrix of the reduced space, with cells as rows and components as columns
clusters
The vector indicating each cell’s cluster.
n
A numeric vector indicating the number of top dimensions at which to gather
statistics (default ‘c(10,20,50)’). Will use all available dimensions if a higher
number is given.
covars
A character vectors containing any additional covariates (column names of ‘col-
Data’) to track during evaluation. If missing, will attempt to use default covari-
ates. To disable, set ‘covars=c()’.

Value

A list with the following components:
* silhouettes: a matrix of the silhouette for each cell-cluster
  pair at each value of ‘n’
* clust.avg.silwidth: a matrix of the cluster average width at each value of
  ‘n’
* R2: the proportion of variance in each component (up to ‘max(n)’) that is explained by the
  clusters (i.e. R-squared of a linear model).

Examples

# random data
library(scater)
sce <- runPCA(logNormCounts(mockSCE(ngenes = 500)))
sce <- addPerCellQC(sce)
# random population labels
sce$cluster <- sample(LETTERS[1:3], ncol(sce), replace=TRUE)
res <- evaluateDimRed(sce, sce$cluster, covars=c("sum","detected"))

# average silhouette widths:
res$clust.avg.silwidth
# adjusted R2 of covariates:
res$covar.adjR2

Description

evaluateNorm

Usage

evaluateNorm(x, clusters = NULL, covars)

Arguments

x
An object of class 'Seurat' or 'SingleCellExperiment' with normalized data
clusters
A vector of true cluster identities. If missing, will attempt to fetch it from the
'phenoid' colData.
covars
Covariates to include, either as data.frame or as colData columns of 'x'
Value

a data.frame.

Examples

```r
# random data
library(scater)
sce <- logNormCounts(mockSCE(ngenes = 500))
sce <- addPerCellQC(sce)
# random population labels
sce$cluster <- sample(LETTERS[1:3], ncol(sce), replace=TRUE)
evaluateNorm(sce, sce$cluster, covars="detected")
```

---

**exampleDEAresults**  
*Example results from the DEA pipeline*

**Description**

Example benchmarking results from a DEA pipeline (see vignette 'pipeComp_dea').

**Value**

a list.

---

**exampleResults**  
*Example pipeline results*

**Description**

Example benchmarking results from a scRNAseq pipeline (see vignette 'pipeComp_scRNA').

**Value**

a list.
farthestPoint

Description
Identifies the point farthest from a line passing through by the first and last points. Used for automation of the elbow method.

Usage
farthestPoint(y, x = NULL)

Arguments
y Monotonically increasing or decreasing values
x Optional x coordinates corresponding to ‘y’ (defaults to seq)

Value
The value of ‘x’ farthest from the diagonal.

Examples
y <- 2^(10:1)
plot(y)
x <- farthestPoint(y)
points(x,y[x],pch=16)

getDimensionality

Description
Returns the estimated intrinsic dimensionality of a dataset.

Usage
getDimensionality(dat, method, maxDims = NULL)

Arguments
dat A Seurat or SCE object with a pca embedding.
method The dimensionality method to use.
maxDims Deprecated and ignored.

Value
An integer.
getQualitativePalette

Description
Returns a qualitative color palette of the given size. If less than 23 colors are required, the colors are based on Paul Tol’s palettes. If more, the ‘randomcolorR’ package is used.

Usage
getQualitativePalette(nbcolors)

Arguments

nbcolors A positive integer indicating the number of colors

Value
A vector of colors

Examples
getQualitativePalette(5)

match_evaluate_multiple

Description
Function to match cluster labels with ‘true’ clusters using the Hungarian algorithm, and return precision, recall, and F1 score. Written by Lukas Weber in August 2016 as part of his cytometry clustering comparison, with just slight modifications on initial handling of input arguments.

Usage
match_evaluate_multiple(clus_algorithm, clus_truth = NULL)

Arguments

clus_algorithm cluster labels from algorithm
clus_truth true cluster labels. If NULL, will attempt to read them from the names of ‘clus_algorithm’ (expecting the format ‘clusterName.cellName’)

Value
A list.
Examples

```r
# random data
data <- data.frame(
  cluster=rep(LETTERS[1:3], each=10),
x=c(rnorm(20, 0), rnorm(10, 1)),
y=c(rnorm(10, 1), rnorm(20, 0))
)
# clustering
data$predicted <- kmeans(dist(data[, -1]), 3)$cluster
# evaluation
match_evaluate_multiple(data$predicted, data$cluster)
```

Description

Merges the (non-aggregated) results of any number of runs of `runPipeline` using the same `PipelineDefinition` (but on different datasets and/or using different parameters). First read the different sets of results using `readPipelineResults`, and pass them to this function.

Usage

```
mergePipelineResults(..., rr = NULL, verbose = TRUE)
```

Arguments

- `...`: Any number of lists of pipeline results, each as produced by `readPipelineResults`
- `rr`: Alternatively, the pipeline results can be passed as a list (in which case `...` is ignored)
- `verbose`: Whether to print processing information

Value

A list of merged pipeline results.

Examples

```
# we produce 2 mock pipeline results:
pip <- mockPipeline()
datasets <- list(ds1=1:3, ds2=c(5, 10, 15))
tmpdir1 <- paste0(tempdir(), '/')
res <- runPipeline(datasets, pip, output.prefix=tmpdir1, alternatives=list())
alternatives <- list(meth1=c('log2', 'sqrt'), meth2='cumsum')
tmpdir2 <- paste0(tempdir(), '/'
res <- runPipeline(datasets, alternatives, pip, output.prefix=tmpdir2)
```

res1 <- readPipelineResults(tmpdir1)
res2 <- readPipelineResults(tmpdir2)
# we merge them:
res <- mergePipelineResults(res1, res2)
# and we aggregate:
res <- aggregatePipelineResults(res)

---

**mockPipeline**  
*mockPipeline*

**Description**  
A mock ‘PipelineDefinition‘ for use in examples.

**Usage**  
mockPipeline()

**Value**  
a ‘PipelineDefinition’

**Examples**  
mockPipeline()

---

**parsePipNames**  
*parsePipNames*

**Description**  
Parses the names of analyses performed through ‘runPipeline‘ to extract a data.frame of parameter values (with decent classes).

**Usage**  
parsePipNames(x, setRowNames = FALSE, addcolumns = NULL)

**Arguments**

- **x**  
The names to parse, or a data.frame with the names to parse as row.names. All names are expected to contain the same parameters.

- **setRowNames**  
Logical; whether to set original names as row.names of the output data.frame (default FALSE)

- **addcolumns**  
An optional data.frame of ‘length(x)’ rows to cbind to the output.
Value

A data.frame

Examples

```r
g <- c("param1=A;param2=5","param1=B;param2=0")
p <- parsePipNames(g)
```

Description

Creates an object of class ‘PipelineDefinition’ containing step functions, as well as optionally step evaluation and aggregation functions.

Usage

```r
PipelineDefinition(
  functions, 
  descriptions = NULL, 
  evaluation = NULL, 
  aggregation = NULL, 
  initiation = identity, 
  defaultArguments = list(), 
  misc = list(), 
  verbose = TRUE 
)
```

Arguments

- `functions`: A list of functions for each step
- `descriptions`: A list of descriptions for each step
- `evaluation`: A list of optional evaluation functions for each step
- `aggregation`: A list of optional aggregation functions for each step
- `initiation`: A function ran when initiating a dataset
- `defaultArguments`: A list of optional default arguments
- `misc`: A list of whatever.
- `verbose`: Whether to output additional warnings (default TRUE).

Value

An object of class ‘PipelineDefinition’, with the slots functions, descriptions, evaluation, aggregation, defaultArguments, and misc.
See Also

`PipelineDefinition-methods, addPipelineStep`. For an example pipeline, see `scrna_pipeline`.

Examples

```r
PipelineDefinition(
  list( step1=function(x, meth1){ get(meth1)(x) },
       step2=function(x, meth2){ get(meth2)(x) } )
)
```

Description

Methods for `PipelineDefinition` class

- get names of PipelineDefinition steps
- set names of PipelineDefinition steps

Usage

```r
## S4 method for signature 'PipelineDefinition'
show(object)

## S4 method for signature 'PipelineDefinition'
names(x)

## S4 replacement method for signature 'PipelineDefinition'
names(x) <- value

## S4 method for signature 'PipelineDefinition'
x$name

## S4 method for signature 'PipelineDefinition'
length(x)

## S4 method for signature 'PipelineDefinition,ANY,ANY,ANY'
x[i]

## S4 method for signature 'PipelineDefinition'
as.list(x)

arguments(object)

## S4 method for signature 'PipelineDefinition'
```
arguments(object)
defaultArguments(object)
defaultArguments(object) <- value

## S4 method for signature 'PipelineDefinition'
defaultArguments(object)

## S4 replacement method for signature 'PipelineDefinition'
defaultArguments(object) <- value

stepFn(object, step = NULL, type)

## S4 method for signature 'PipelineDefinition'
stepFn(object, step = NULL, type)

stepFn(object, step, type) <- value

## S4 replacement method for signature 'PipelineDefinition'
stepFn(object, step, type) <- value

### Arguments

- **object**: An object of class `PipelineDefinition`
- **x**: An object of class `PipelineDefinition`
- **value**: Replacement values
- **name**: The step name
- **i**: The index(es) of the steps
- **step**: The name of the step for which to set or get the function
- **type**: The type of function to set/get, either 'functions', 'evaluation', 'aggregation', 'descriptions', or 'initiation' (will parse partial matches)

### Value

Depends on the method.

### See Also

`PipelineDefinition`, `addPipelineStep`

### Examples

```
pd <- mockPipeline()
length(pd)
names(pd)
pd$step1
pd[2:1]
```
plotElapsed

Description

Plot total elapsed time per run, split per step.

Usage

```r
plotElapsed(
  res,
  steps = names(res$elapsed$stepwise),
  agg.by,
  agg.fn = mean,
  width = 0.9,
  split.datasets = TRUE,
  return.df = FALSE
)
```

Arguments

- `res` Aggregated pipeline results
- `steps` The step(s) to plot (default all)
- `agg.by` The parameters by which to aggregate (set to FALSE to disable aggregation)
- `agg.fn` Aggregation function
- `width` Width of the bar; default 0.9, use 1 to remove the gaps
- `split.datasets` Logical; whether to split the datasets into facets
- `return.df` Logical; whether to return the data.frame instead of plot

Value

A ggplot, or a data.frame if `return.df=TRUE`

Examples

```r
data("exampleResults", package="pipeComp")
plotElapsed(exampleResults, agg.by = "norm")
```
**readPipelineResults**

**Description**

readPipelineResults

**Usage**

readPipelineResults(path = NULL, resfiles = NULL)

**Arguments**

- **path**
  The path (e.g. folder or prefix) to the results. Either ‘path’ or ‘resfiles’ should be given.

- **resfiles**
  A vector of paths to ‘*.evaluation.rds’ files. Either ‘path’ or ‘resfiles’ should be given.

**Value**

A list of results.

**Examples**

```r
# we produce mock pipeline results:
pip <- mockPipeline()
datasets <- list(ds1=1:3, ds2=c(5,10,15))
tmpdir1 <- paste0(tempdir(),'/')
res <- runPipeline(datasets, pipelineDef=pip, output.prefix=tmpdir1,
  alternatives=list() )
# we read the evaluation files:
res <- readPipelineResults(tmpdir1)
```

---

**runPipeline**

**Description**

This function runs a pipeline with combinations of parameter variations on nested steps. The pipeline has to be defined as a list of functions applied consecutively on their respective outputs. See ‘examples’ for more details.
runPipeline(
    datasets,
    alternatives,
    pipelineDef,
    comb = NULL,
    output.prefix = "",
    nthreads = 1,
    saveEndResults = TRUE,
    debug = FALSE,
    skipErrors = TRUE,
    ...
)

Arguments

datasets          A named vector of initial objects or paths to rds files.
alternatives      The (named) list of alternative values for each parameter.
pipelineDef       An object of class PipelineDefinition.
comb              An optional matrix of indexes indicating the combination to run. Each column
                  should correspond to an element of ‘alternatives’, and contain indexes relative
                  to this element. If omitted, all combinations will be performed.
output.prefix     An optional prefix for the output files.
nthreads          Number of threads, default 1. If the memory requirements are very high or the
                  first steps very long to compute, consider setting this as the number of datasets
                  or below.
saveEndResults    Logical; whether to save the output of the last step.
debug            Logical (default FALSE). When enabled, disables multithreading and prints ex-
                  tra information.
skipErrors        Logical. When enabled, ‘runPipeline’ will continue even when an error has been
                  encountered, and report the list of steps/datasets in which errors were encoun-
                  tered.
...                passed to MulticoreParam. Can for instance be used to set ‘makeCluster’ argu-
                  ments, or set ‘threshold="TRACE"’ when debugging in a multithreaded context.

Value

A SimpleList with elapsed time and the results of the evaluation functions defined by the given
‘pipelineDef’.

The results are also stored in the output folder with:

• The clustering results for each dataset (‘endOutputs.rds’ files),
• A SimpleList of elapsed time and evaluations for each dataset (‘evaluation.rds’ files),
• A list of the ‘pipelineDef’, ‘alternatives’, ‘sessionInfo()’ and function call used to produce the
  results (‘runPipelineInfo.rds’ file),
• A copy of the SimpleList returned by the function (‘aggregated.rds‘file).
Examples

```r
pip <- mockPipeline()
datasets <- list(ds1=1:3, ds2=c(5,10,15))
tmpdir1 <- paste0(tempdir(), "/")
res <- runPipeline(datasets, pipelineDef=pip, output.prefix=tmpdir1, 
  alternatives=list() )
# See the `pipeComp_scRNA` vignette for a more complex example
```

Description

Plots descriptive information about the datasets

Usage

```r
scrna_describeDatasets(sces, pt.size = 0.3, ...)
```

Arguments

- `sces`: A character vector of paths to SCE rds files, or a list of SCEs
- `pt.size`: Point size (for reduced dims)
- `...`: Passed to `geom_point()`

Value

A plot_grid output

Description

`scrna_evalPlot_filtering`
Usage

```r
scrna_evalPlot_filtering(
  res,
  steps = c("doublet", "filtering"),
  clustMetric = "mean_F1",
  filterExpr = TRUE,
  atNearestK = FALSE,
  returnTable = FALSE,
  point.size = 2.2,
  ...
)
```

Arguments

- `res`: Aggregated pipeline results (i.e. the output of `runPipeline` or `aggregateResults`)
- `steps`: Steps to include (default 'doublet' and 'filtering'); other steps will be averaged.
- `clustMetric`: Clustering accuracy metric to use (default 'mean_F1"")
- `filterExpr`: An optional filtering expression based on the columns of the clustering evaluation (e.g. 'filterExpr=param1=="value1"' or 'filterExpr=n_clus==true.nbClusts').
- `atNearestK`: Logical; whether to restrict analyses to those giving the smallest deviation from the real number of clusters (default FALSE).
- `returnTable`: Logical; whether to return the data rather than plot.
- `point.size`: Size of the points
- `...`: passed to 'geom_point'

Value

A ggplot, or a data.frame if 'returnTable=TRUE'

Examples

```r
data("exampleResults", package="pipeComp")
scrna_evalPlot_filtering(exampleResults)
```

Description

Plots a multi-level summary heatmap of many analyses of the `scrna_pipeline`.
**Usage**

```r
scrna_evalPlot_overall(
  res,
  agg.by = NULL,
  width = NULL,
  datasets_as_columnNames = TRUE,
  rowAnnoColors = NULL,
  column_names_gp = gpar(fontsize = 10),
  column_title_gp = gpar(fontsize = 12),
  heatmap_legend_param = list(by_row = TRUE, direction = "horizontal", nrow = 1),
  ...
)
```

**Arguments**

- `res`  
  Aggregated pipeline results (i.e. the output of `runPipeline` or `aggregateResults`)
- `agg.by`  
  The parameters by which to aggregate.
- `width`  
  The width of individual heatmap bodies.
- `datasets_as_columnNames`  
  Logical; whether dataset names should be printed below the columns (except for silhouette) rather than using a legend.
- `rowAnnoColors`  
  Optional list of colors for the row annotation variables (passed to `HeatmapAnnotation(col=...)`)  
- `column_names_gp`  
  Passed to each calls to `Heatmap`  
- `column_title_gp`  
  Passed to each calls to `Heatmap`
- `heatmap_legend_param`  
  Passed to each calls to `Heatmap`
- `...`  
  Passed to each calls to `Heatmap`

**Value**

A HeatmapList

**Examples**

```r
library(ComplexHeatmap)
data("exampleResults")
h <- scrna_evalPlot_overall(exampleResults)
draw(h, heatmap_legend_side="bottom")
```
Description

Plot a min/max/mean/median silhouette width heatmap from aggregated evaluation results of the 'scrna_pipeline'.

Usage

```r
scrna_evalPlot_silh(
  res,
  what = c("minSilWidth", "meanSilWidth"),
  step = "dimreduction",
  dims = 1,
  agg.by = NULL,
  agg.fn = mean,
  filterExpr = NULL,
  value_format = ",",
  reorder_rows = FALSE,
  reorder_columns = TRUE,
  show_heatmap_legend = TRUE,
  show_column_names = FALSE,
  col = rev(RColorBrewer::brewer.pal(n = 11, "RdBu")),
  font_factor = 0.9,
  row_split = NULL,
  shortNames = TRUE,
  value_cols = c("white", "black"),
  title = NULL,
  anno_legend = TRUE,
  ...
)
```

Arguments

- **res**: Aggregated pipeline results (i.e. the output of 'runPipeline' or 'aggregateResults')
- **what**: What metric to plot, possible values are "minSilWidth", "meanSilWidth" (default), "medianSilWidth", or "maxSilWidth".
- **step**: Name of the step for which to plot the evaluation results. Defaults to "dimreduction".
- **dims**: If multiple sets of dimensions are available, which one to use (defaults to the first).
- **agg.by**: Aggregate results by these columns (default no aggregation)
- **agg.fn**: Function for aggregation (default mean)
filterExpr    An optional filtering expression based on the columns of the target dataframe, (e.g. ‘filterExpr=param1=="value1"’).
value_format Format for displaying cells’ values (no label by default)
reorder_rows Whether to sort rows (default FALSE). The row names themselves can also be passed to specify an order, or a ‘ComplexHeatmap’.
reorder_columns Whether to sort columns (default TRUE).
show_heatmap_legend Passed to ‘Heatmap’ (default FALSE)
show_column_names Passed to ‘Heatmap’ (default FALSE)
col Colors for the heatmap
font_factor A scaling factor applied to font sizes (default 1)
row_split Optional column (included in ‘agg.by’) by which to split the rows. Alternatively, an expression using the columns (retained after aggregation) can be passed.
shortNames Logical; whether to use short row names (with only the parameter values instead of the parameter name and value pairs), default TRUE.
value_cols A vector of length 2 indicating the colors of the values (above and below the mean), if printed
title Plot title
anno_legend Logical; whether to plot the legend for the datasets
... Passed to ‘Heatmap’

Value
A Heatmap

Examples

data("exampleResults", package="pipeComp")
scrna_evalPlot_silh(exampleResults, agg.by=c("filt","norm"), row_split="norm")

Description
The ‘PipelineDefinition’ for the default scRNAseq clustering pipeline, with steps for doublet identification, filtering, normalization, feature selection, dimensionality reduction, and clustering. Alternative arguments should be character, numeric or logical vectors of length 1 (e.g. the function name for a method, the number of dimensions, etc). The default pipeline has the following steps and arguments:
• doublet: ‘doubletmethod’ (name of the doublet removal function)
• filtering: ‘filt’ (name of the filtering function, or filter string)
• normalization: ‘norm’ (name of the normalization function)
• selection: ‘sel’ (name of the selection function, or variable of rowData on which to select) and ‘selnb’ (number of features to select)
• dimreduction: ‘dr’ (name of the dimensionality reduction function) and ‘maxdim’ (maximum number of components to compute)
• clustering: ‘clustmethod’ (name of the clustering function), ‘dims’ (number of dimensions to use), ‘k’ (number of nearest neighbors to use, if applicable), ‘steps’ (number of steps in the random walk, if applicable), ‘resolution’ (resolution, if applicable), ‘min.size’ (minimum cluster size, if applicable). If using the ‘scrna_alternatives.R’ wrappers, the dimensionality can be automatically estimated by specifying ‘dims = "method_name"’.

Usage

```
scrna_pipeline(saveDimRed = FALSE, pipeClass = c("seurat", "sce"))
```

Arguments

- **saveDimRed**
  Logical; whether to save the dimensionality reduction for each analysis (default FALSE)
- **pipeClass**
  ‘sce’ or ‘seurat’; which object class to use throughout the pipeline. Note that the ‘alternatives’ functions have to be built around the chosen class. Given that, if running the ‘scrna_alternatives’, the class of whole pipeline is determined by the output of the filtering, only this step is affected by this option.

Value

A ‘PipelineDefinition’ object to be used with ‘runPipeline’.

Examples

```
pip <- scrna_pipeline()
pip
```

---

**stableG**

*Lists of stable genes*

**Description**

Genes were simply obtained by querying the respective GO terms

**Value**

a list.
Index

[,PipelineDefinition,ANY,ANY,ANY-method (PipelineDefinition-methods), 20
$,PipelineDefinition-method (PipelineDefinition-methods), 20
addPipelineStep, 3, 20, 21
aggregatePipelineResults, 4
arguments (PipelineDefinition-methods), 20
arguments,PipelineDefinition-method (PipelineDefinition-methods), 20
as.list,PipelineDefinition-method (PipelineDefinition-methods), 20
buildCombMatrix, 5
checkPipelinePackages, 5
clustMetricsCorr, 6
colCenterScale, 6, 9
crlgenes, 7
defa_evalPlot_curve, 7
defa_pipeline, 8
defaultArguments (PipelineDefinition-methods), 20
defaultArguments,PipelineDefinition-method (PipelineDefinition-methods), 20
defaultArguments<-, (PipelineDefinition-methods), 20
defaultArguments<-,PipelineDefinition-method (PipelineDefinition-methods), 20
defaultStepAggregation, 8
evalHeatmap, 9
evaluateClustering, 11
evaluateDEA, 12
evaluateDimRed, 12
evaluateNorm, 13
exampleDEAresults, 14
exampleResults, 14
farthestPoint, 15
getDimensionality, 15
getQualitativePalette, 16
length,PipelineDefinition-method (PipelineDefinition-methods), 20
match_evaluate_multiple, 16
mergePipelineResults, 17
mockPipeline, 18
names,PipelineDefinition-method (PipelineDefinition-methods), 20
names<-,PipelineDefinition-method (PipelineDefinition-methods), 20
parsePipNames, 18
pipeComp (pipeComp-package), 3
pipeComp-package, 3
PipelineDefinition, 3, 4, 17, 19, 20, 21, 24
PipelineDefinition-class (PipelineDefinition), 19
PipelineDefinition-method (PipelineDefinition-methods), 20
PipelineDefinition-methods, 20
plotElapsed, 22
readPipelineResults, 4, 17, 23
runPipeline, 23

scrna_describeDatasets, 25
scrna_evalPlot_filtering, 25
scrna_evalPlot_overall, 26
scrna_evalPlot_silh, 28
scrna_pipeline, 20, 29
show,PipelineDefinition-method
  (PipelineDefinition-methods), 20
stableG, 30
stepFn,(PipelineDefinition-methods), 20
stepFn,PipelineDefinition-method
  (PipelineDefinition-methods), 20
stepFn<-,(PipelineDefinition-methods), 20
stepFn<-,PipelineDefinition-method
  (PipelineDefinition-methods), 20