Package ‘RTNsurvival’

October 16, 2018

Type Package

Title Survival analysis using transcriptional networks inferred by the RTN package

Version 1.4.5

Author Clarice S. Groeneveld, Vinicius S. Chagas, Mauro A. A. Castro

Maintainer Clarice Groeneveld <clari.groeneveld@gmail.com>, Mauro A. A. Castro <mauro.a.castro@gmail.com>

Depends R(>= 3.5), RTN(>= 2.4.5), RTNduals(>= 1.4.4), methods

Imports survival, RColorBrewer, grDevices, graphics, stats, utils, scales

Suggests Fletcher2013b, pheatmap, knitr, rmarkdown, BiocStyle, RUnit, BiocGenerics

Description RTNsurvival is a tool for integrating regulons generated by the RTN package with survival information. For a given regulon, the 2-tailed GSEA approach computes a differential Enrichment Score (dES) for each individual sample, and the dES distribution of all samples is then used to assess the survival statistics for the cohort. There are two main survival analysis workflows: a Cox Proportional Hazards approach used to model regulons as predictors of survival time, and a Kaplan-Meier analysis assessing the stratification of a cohort based on the regulon activity. All plots can be fine-tuned to the user's specifications.

License Artistic-2.0

biocViews NetworkEnrichment, Survival, GeneRegulation, GeneSetEnrichment, NetworkInference, GraphAndNetwork

LazyData TRUE

VignetteBuilder knitr

Encoding UTF-8

RoxygenNote 6.1.0

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

git_branch RELEASE_3_7

git_last_commit 41e23d2

git_last_commit_date 2018-09-04

Date/Publication 2018-10-15

R topics documented:

`RTNsurvival-package` *Performs survival analysis using transcriptional networks inferred by the RTN package.*

Description

This package provides classes and methods to perform survival analysis using transcriptional networks inferred by the RTN package, including Kaplan-Meier and multivariate survival analysis using Cox's regression model.

Details

Package:	RTNsurvival
Type:	Package
Depends:	R (>= 3.3.0), methods
Imports:	RTN, survival, RColorBrewer
Suggests:	Fletcher2013b, knitr, rmarkdown, BiocStyle, RUnit, BiocGenerics
License:	Artistic-2.0
biocViews:	NetworkInference, NetworkEnrichment, GeneRegulation, GeneExpression, GraphAndNetwork, GeneSetEn

Index

TNS-class:	an S4 class for survival survival analysis using RTN transcriptional networks.
tni2tnsPreprocess:	a preprocessing method for objects of class TNS.
tnsGSEA2:	compute regulon activity by calling 'GSEA2' algorithm.

tnsPlotGSEA2:	plot results from the two-tailed GSEA.
tnsKM:	Kaplan-Meier analysis for TNS class objects.
tnsPlotKM:	Kaplan-Meier plots for TNS class objects.
tnsCox:	Cox regression analysis for TNS class objects.
tnsPlotCox:	Cox plots for TNS class objects.
tnsGet:	Get information from slots in a TNS object.
tnsInteraction:	A generic call to 'tnsCoxInteraction' and 'tnsKmInteraction'.
tnsKmInteraction:	Kaplan-Meier analysis for dual regulons.
tnsPlotKmInteraction:	Plot results from Kaplan-Meier analysis for dual regulons.
tnsCoxInteraction:	Cox regression analysis for dual regulons.
tnsPlotCoxInteraction:	Plot results from Cox regression analysis for dual regulons.
tnsPlotGSEA2:	Plot 2-tailed GSEA for a sample from a TNS.
tnsAREA3:	compute regulon activity by calling 'aREA3' algorithm.

Further information is available in the vignettes by typing `vignette("RTNsurvival")`. Documented topics are also available in HTML by typing `help.start()` and selecting the RTNsurvival package from the menu.

Author(s)

Clarice S. Groeneveld, Vinicius S. Chagas, Gordon Robertson, ..., Kerstin Meyer, Mauro A. A. Castro

References

Fletcher M.N.C. et al., *Master regulators of FGFR2 signalling and breast cancer risk.* Nature Communications, 4:2464, 2013.

Castro M.A.A. et al., *Regulators of genetic risk of breast cancer identified by integrative network analysis.* Nature Genetics, 48:12-21, 2016.

`tni2tnsPreprocess,TNI-method`

Preprocessing of TNS class objects

Description

Creates TNS class onbjects for regulons an survival data.

Usage

```
## S4 method for signature 'TNI'
tni2tnsPreprocess(tni, survivalData = NULL,
  regulatoryElements = NULL, time = 1, event = 2, endpoint = NULL,
  pAdjustMethod = "BH", keycovar = NULL, samples = NULL,
  excludeMid = FALSE)
```

Arguments

`tni` A TNI class, already processed with the same samples listed in the survival data.frame.

`survivalData` A named data.frame with samples in rows and survival data in the columns (this does not need to be provided if avaibale in the 'TNI' object).

`regulatoryElements` A character vector specifying which 'TNI' regulatory elements should be evaluated.

`time` A numeric or character value corresponding to the column of the data.frame where the time of last observation is given.

`event` A numeric or character value, corresponding to the columm of the data.frame where the 'event' information is given.

`endpoint` A numeric value. It represents the cut-off point for the 'time', if any.

`pAdjustMethod` A single character value specifying the p-value adjustment method to be used (see 'p.adjust' function for details).

`keycovar` A character vector of 'keycovars' listed in 'survivalData' columns.

`samples` An optional character vector listing samples to be analyzed.

`excludeMid` A logical value. If TRUE, inconclusive dES values is not consired in the survival analysis.

Value

A preprocessed TNS class

See Also

`tni.preprocess` for similar preprocessing.

Examples

```
# load survival data
data(survival.data)

# load TNI-object
data(stni, package = "RTN")

# create a new TNS object
stns <- tni2tnsPreprocess(stni, survivalData = survival.data,
        keycovar = c('Grade','Age'), time = 1, event = 2)
```

TNS-class *TNS: An S4 class for survival analysis using transcriptional networks inferred by the RTN package.*

Description

TNS: An S4 class for survival analysis using transcriptional networks inferred by the RTN package.

Slots

`TNI` a previously computed TNI-class object.

`survivalData` a data frame containing the survival data for all samples. Samples must be in the rows and the survival variables in the columns. Time of last update and event in last update (0 for alive, 1 for deceased).

`para` a list with the parameters used to compute the GSEA2 analysis.

`results` a list with results from TNS methods.

`status` a vector containing the processing status of the TNS object.

Constructor

see `tni2tnsPreprocess` constructor.

`TNS.data` *A pre-processed dataset for demonstration purposes only.*

Description

A minimum dataset used to demonstrate RTNsurvival main features.

Usage

```
data(survival.data)
```

Format

`survival.data` A data.frame with a subset of samples in the Fletcher2013b package.

Details

The dataset consists of data.frame with survival and clinical variables used in the RTNsurvival vignettes. It should be regarded as a toy example for demonstration purposes only, despite being extracted, pre-processed and size-reduced from Fletcher et al. (2013) and Curtis et al. (2012).

Value

a data.frame.

References

Fletcher M.N.C. et al., *Master regulators of FGFR2 signalling and breast cancer risk.* Nature Communications, 4:2464, 2013.

Curtis C. et al., *The genomic and transcriptomic architecture of 2,000 breast tumours reveals novel subgroups.* Nature 486, 7403. 2012.

Examples

```
data(survival.data)
```

tnsAREA3,TNS-method *Compute regulon activity by calling aREA (analytic Rank-based Enrichment Analysis) algorithm*

Description

Uses tni.area3 function to compute regulon activity for TNS class objects.

Usage

```
## S4 method for signature 'TNS'
tnsAREA3(tns, ...)
```

Arguments

tns	A TNS class, which has been preprocessed
...	Additional parameters passed to tni.area3 function.

Value

A TNS class, with added regulon activity scores.

References

Alvarez et al. Functional characterization of somatic mutations in cancer using network-based inference of protein activity. Nature Genetics, 48(8):838-847, 2016.

See Also

tni.area3 for additional details.

Examples

```
# load survival data
data(survival.data)

# load TNI-object
data(stni, package = "RTN")

stns <- tni2tnsPreprocess(stni, survivalData = survival.data,
keycovar = c('Grade','Age'), time = 1, event = 2)

stns <- tnsAREA3(stns)
```

`tnsCox,TNS-method` *Cox regression analysis for TNS class objects*

Description

Run Cox multivariate regression for regulons and other covariates.

Usage

```
## S4 method for signature 'TNS'
tnsCox(tns, regs = NULL, qqkeycovar = FALSE,
  verbose = TRUE)
```

Arguments

`tns`	A TNS object, which must have passed GSEA2 analysis.
`regs`	An optional string vector listing regulons to be tested.
`qqkeycovar`	A logical value. If TRUE, only the samples in the 2nd and 3rd quartils of 'keycovar' are used in the analysis. If FALSE, all samples are used (see `tni2tnsPreprocess`).
`verbose`	A logical value specifying to display detailed messages (when verbose=TRUE) or not (when verbose=FALSE).

Value

Cox hazard models and statistics.

Examples

```
# load survival data
data(survival.data)

# load TNI-object
data(stni, package = "RTN")

stns <- tni2tnsPreprocess(stni, survivalData = survival.data,
keycovar = c('Age','Grade'), time = 1, event = 2)
stns <- tnsGSEA2(stns)
stns <- tnsCox(stns, regs = c('PTTG1','E2F2','FOXM1'))
tnsGet(stns, "coxTable")
```

`tnsCoxInteraction,TNS-method` *Cox regression analysis for dual regulons*

Description

Cox regression analysis for dual regulons, including the interaction term.

Usage

```
## S4 method for signature 'TNS'
tnsCoxInteraction(tns, mbr, stepFilter = TRUE,
  pValueCutoff = 0.05, verbose = TRUE)
```

Arguments

`tns` A TNS object with regulons used to compute the dual regulons.

`mbr` An MBR object computed from the same 'TNI' object used to make the 'TNS' object.

`stepFilter` A single logical value specifying to use a step-filter algorithm, testing dual regulons that have at least one significant predictor in the 'tnsCox' method (when stepFilter=TRUE) or not (when stepFilter=FALSE).

`pValueCutoff` An numeric value. The p-value cutoff applied to the results from the previous steps of the analysis pipeline (when stepFilter=TRUE).

`verbose` A logical value specifying to display detailed messages (when verbose=TRUE) or not (when verbose=FALSE).

Value

Cox hazard models and statistics.

An updated TNS-class object containing Cox regression models for all given duals

See Also

`tni2mbrPreprocess` for all plot parameters

Examples

```
# load survival data
data(survival.data)

# load TNI-object
data(stni, package = "RTN")

# perform survival analysis for regulons
stns <- tni2tnsPreprocess(stni, survivalData = survival.data,
time = 1, event = 2)
stns <- tnsGSEA2(stns)

# infer dual regulons
smbr <- tni2mbrPreprocess(stni)
smbr <- mbrAssociation(smbr)

# run Cox regression for dual regulons
stns <- tnsCoxInteraction(stns, smbr, stepFilter = FALSE)
tnsGet(stns, "coxInteractionTable")
```

`tnsGet,TNS-method` *Get information from slots in a TNS object*

Description

Get information from individual slots in a TNS object and any available results from a previous analysis.

Usage

```
## S4 method for signature 'TNS'
tnsGet(tns, what)
```

Arguments

`tns`	A TNS object.
`what`	A string specifying what should be retrieved from the object. Options: ’status’,’survivalData’, ’regulonActivity’, ’TNI’, ’para’, ’kmTable’, ’kmFit’, ’coxTable’, ’coxFit’, ’kmInteractionTable’, ’kmInteractionFit’, ’coxInteractionTable’, ’coxInteractionFit’, and ’regulatoryElements’.

Value

Content from slots in the TNS object.

Examples

```
# load survival data
data(survival.data)

# load TNI-object
data(stni, package = "RTN")

stns <- tni2tnsPreprocess(stni, survivalData = survival.data,
keycovar = c('Grade','Age'), time = 1, event = 2)
stns <- tnsGSEA2(stns)
regulonActivity <- tnsGet(stns, 'regulonActivity')
```

`tnsGSEA2,TNS-method` *Compute regulon activity using 2-tailed Gene Set Enrichment Analysis*

Description

Works as a wrapper for `tni.gsea2`, performing a 2-tailed GSEA analysis on a TNI class object and integrating the results into the TNS class object.

Usage

```
## S4 method for signature 'TNS'
tnsGSEA2(tns, ...)
```

Arguments

tns	A TNS class, which has been preprocessed
...	Additional parameters passed to tni.gsea2 function.

Value

A TNS class, with added regulon activity scores.

See Also

tni.gsea2 for information on all parameters.

Examples

```
# load survival data
data(survival.data)

# load TNI-object
data(stni, package = "RTN")

stns <- tni2tnsPreprocess(stni, survivalData = survival.data,
keycovar = c('Grade','Age'), time = 1, event = 2)
stns <- tnsGSEA2(stns)

## Not run:

# parallel version with SNOW package!
library(snow)
options(cluster=makeCluster(3, "SOCK"))
stns <- tnsGSEA2(stns)
stopCluster(getOption("cluster"))


## End(Not run)
```

tnsInteraction,TNS-method

Survival analysis for dual regulons

Description

A generic call to 'tnsCoxInteraction' and 'tnsKmInteraction' functions.

Usage

```
## S4 method for signature 'TNS'
tnsInteraction(tns, ..., verbose = TRUE)
```

Arguments

`tns` A TNS object, which must have passed GSEA2 analysis.

`...` Parameters passed to `tnsKmInteraction` and `tnsCoxInteraction` functions.

`verbose` A logical value specifying to display detailed messages (when verbose=TRUE) or not (when verbose=FALSE).

Value

A TNS object evaluated by the ’tnsKmInteraction’ and ’tnsCoxInteraction’ functions.

Examples

```
# load survival data
data(survival.data)

# load TNI-object
data(stni, package = "RTN")

stns <- tni2tnsPreprocess(stni, survivalData = survival.data,
keycovar = c('Grade','Age'), time = 1, event = 2)
stns <- tnsGSEA2(stns)

# infer dual regulons
smbr <- tni2mbrPreprocess(stni)
smbr <- mbrAssociation(smbr)

# survival analysis for dual regulons
# stns <- tnsInteraction(stns, smbr)
```

`tnsKM,TNS-method` *Kaplan-Meier analysis for TNS class objects*

Description

Creates survival curves and tests if there is a difference between curves using ’survfit’ and ’survdiff’ functions, respectivelly.

Usage

```
## S4 method for signature 'TNS'
tnsKM(tns, regs = NULL, nSections = 1,
  verbose = TRUE)
```

Arguments

`tns` A TNS object, which must have passed GSEA2 analysis.

`regs` An optional string vector listing regulons to be tested.

`nSections` A numeric value for sample stratification. The larger the number, the more subdivisions will be created for the Kaplan-Meier analysis.

`verbose` A logical value specifying to display detailed messages (when verbose=TRUE) or not (when verbose=FALSE).

Value

Results from 'survfit' and 'survdiff', including log-rank statistics.

Examples

```
# load survival data
data(survival.data)

# load TNI-object
data(stni, package = "RTN")

stns <- tni2tnsPreprocess(stni, survivalData = survival.data,
        keycovar = c('Grade','Age'), time = 1, event = 2)
stns <- tnsGSEA2(stns)
stns <- tnsKM(stns)
tnsGet(stns, "kmTable")
```

tnsKmInteraction,TNS-method

Kaplan-Meier analysis for dual regulons

Description

Kaplan-Meier analysis for dual regulons, assessing the interaction between regulons.

Usage

```
## S4 method for signature 'TNS'
tnsKmInteraction(tns, mbr, stepFilter = TRUE,
  pValueCutoff = 0.05, verbose = TRUE)
```

Arguments

`tns`	A TNS object, which must have passed GSEA2 analysis.
`mbr`	An MBR object computed from the same 'TNI' object used to make the 'TNS' object.
`stepFilter`	A single logical value specifying to use a step-filter algorithm, testing dual regulons that have at least one significant predictor in the 'tnsKM' method (when stepFilter=TRUE) or not (when stepFilter=FALSE).
`pValueCutoff`	An numeric value. The p-value cutoff applied to the results from the previous steps of the analysis pipeline (when stepFilter=TRUE).
`verbose`	A logical value specifying to display detailed messages (when verbose=TRUE) or not (when verbose=FALSE).

Value

Results from 'survfit' and 'survdiff', including log-rank statistics.

Examples

```
# load survival data
data(survival.data)

# load TNI-object
data(stni, package = "RTN")

stns <- tni2tnsPreprocess(stni, survivalData = survival.data,
keycovar = c('Grade','Age'), time = 1, event = 2)
stns <- tnsGSEA2(stns)

# infer dual regulons
smbr <- tni2mbrPreprocess(stni)
smbr <- mbrAssociation(smbr)

# KM analysis for dual regulons
# stns <- tnsKmInteraction(stns, smbr, stepFilter = FALSE)
# tnsGet(stns, "kmInteractionTable")
```

`tnsPlotCox,TNS-method` *Cox plots for TNS class objects*

Description

Run Cox multivariate regression for regulons and key covariables.

Usage

```
## S4 method for signature 'TNS'
tnsPlotCox(tns, regs = NULL, fname = "coxplot",
  fpath = ".", ylab = "Regulons and other covariates",
  xlab = "Hazard Ratio (95% CI)", width = 5, height = 5,
  xlim = c(0.3, 3), sortregs = TRUE, plotpdf = FALSE)
```

Arguments

`tns`	A TNS object, which must have passed GSEA2 analysis.
`regs`	An optional string vector specifying regulons to make the plot.
`fname`	A string. The name of the PDF file which will contain the plot.
`fpath`	A string. The directory where the file will be saved.
`ylab`	A string. The label of the y-axis, describing what is represented.
`xlab`	A string. The label of the x-axis.
`width`	A numeric value. The width of the plot.
`height`	A numeric value. The height of the plot.
`xlim`	A vector with 2 values indicating lowest and highest HR values.
`sortregs`	A logical value. If TRUE, regulons are sorted from most negatively associated with hazard to most positively associated with hazard.
`plotpdf`	A logical value.

Value

A Cox hazard model plot and statistics.

Examples

```
# load survival data
data(survival.data)

# load TNI-object
data(stni, package = "RTN")

stns <- tni2tnsPreprocess(stni, survivalData = survival.data,
keycovar = c('Age','Grade'), time = 1, event = 2)
stns <- tnsGSEA2(stns)
stns <- tnsCox(stns, regs = c('PTTG1','E2F2','FOXM1'))
tnsPlotCox(stns)
```

tnsPlotCoxInteraction,TNS-method

Plot results from Cox regression analysis for dual regulons

Description

Plot results from Cox regression analysis for dual regulons

Usage

```
## S4 method for signature 'TNS'
tnsPlotCoxInteraction(tns, dualreg, xlim = NULL,
  ylim = NULL, hlim = NULL, hcols = c("#008080ff", "#d45500ff"),
  showdata = TRUE, colorPalette = "bluered",
  fname = "coxInteraction", fpath = ".", width = 4.5, height = 4,
  plotype = "3D", plotpdf = FALSE)
```

Arguments

`tns` A ‘TNS‘ object with regulons used to compute the dual regulon.

`dualreg` A character string with the name of a dual regulon.

`xlim` A numeric vector of length 2, i.e. xlim = c(x1, x2), indicating the limits of the plot for the first member of the dual regulon. If xlim = NULL, it will be derevided from the observed data ranges. Values must be in the range [-2,2].

`ylim` A numeric vector of length 2, i.e. ylim = c(y1, y2), indicating the limits of the plot for the second member of the dual regulon. If ylim = NULL, it will be derevided from the observed data ranges. Values must be in the range [-2,2]. If plotype='2D', ylim represents the two fixed values for the second member of the dual regulon.

`hlim` A numeric vector of length 2, i.e. hlim = c(h1, h2), indicating the limits of the plot for the Hazard Ratio (HR). If hlim = NULL, it will be derevided from the observed data ranges. If plotype='2D', HR is represented in the y-axis.

`hcols`	A vector of length 2 indicating a diverging color scheme for the Hazard Ratio (HR).
`showdata`	A logical value indicating whether to show the original data used to fit linear model.
`colorPalette`	A string, which can be 'red', 'blue', 'redblue', or 'bluered'. Alternatively, it can be a vector of five colors or hex values.
`fname`	A string. The name of the PDF file (when plotpdf=TRUE).
`fpath`	A string. The directory where the file will be saved.
`width`	A numeric value. The width of the plot.
`height`	A numeric value. The height of the plot.
`plotype`	A string indicating '2D' of '3D' plot type. If plotype = '2D', the Hazard Ratio is represented in the y-axis.
`plotpdf`	A logical value.

Value

A Cox hazard model plot and statistics.

A 3D heatmap plot.

See Also

`tnsKM`, `tnsCox`

Examples

```
# load survival data
data(survival.data)

# load TNI-object
data(stni, package = "RTN")

# perform survival analysis for regulons
stns <- tni2tnsPreprocess(stni, survivalData = survival.data, time = 1, event = 2)
stns <- tnsGSEA2(stns, verbose=FALSE)

# infer dual regulons
smbr <- tni2mbrPreprocess(stni)
smbr <- mbrAssociation(smbr)

# run Cox regression for dual regulons
# stns <- tnsCoxInteraction(stns, smbr, stepFilter = FALSE)
# tnsPlotCoxInteraction(stns, dualreg = "FOXM1~PTTG1")
```

`tnsPlotGSEA2,TNS-method`

Plot 2-tailed GSEA for a sample from a TNS

Description

Makes a 2-tailed GSEA plot for a certain phenotype (sample) present in a TNS. A wrapper of `tna.plot.gsea2`

Usage

```
## S4 method for signature 'TNS'
tnsPlotGSEA2(tns, aSample, regs = NULL, refsamp = NULL,
  checklog = FALSE, ntop = NULL, pValueCutoff = 0.05,
  pAdjustMethod = "BH", verbose = TRUE, plotpdf = FALSE, ...)
```

Arguments

`tns` A TNS object

`aSample` A string specifying a given sample number present in the 'survivalData' table.

`regs` An optional string vector specifying regulons to make the plot.

`refsamp` A character vector.

`checklog` A logical value. If TRUE, expression values are transformed into log space.

`ntop` An optional integer value. The number of regulons for which the GSEA2 will be plotted.

`pValueCutoff` An numeric value. The p-value cutoff for the analysis.

`pAdjustMethod` A character. Specifies the adjustment method for the pvalue. See `p.adjust`

`verbose` A logical value specifying to display detailed messages (when verbose=TRUE) or not (when verbose=FALSE).

`plotpdf` A single logical value.

`...` parameters which will be passed to `tna.plot.gsea2`, such as ylimPanels, heightPanels, width, height, ylabPanels, xlab...

Value

A plot containing the 2-tailed GSEA analysis for a phenotype.

See Also

`tna.plot.gsea2` for all plot parameters

Examples

```
# load survival data
data(survival.data)

# load TNI-object
data(stni, package = "RTN")

stns <- tni2tnsPreprocess(stni, survivalData = survival.data,
        keycovar = c('Grade','Age'), time = 1, event = 2)
stns <- tnsGSEA2(stns, verbose=FALSE)
tnsPlotGSEA2(stns, 'MB-5115', regs = 'FOXM1', plotpdf = FALSE)
```

tnsPlotKM,TNS-method *Kaplan-Meier plots for TNS class objects*

Description

Makes a 2 or 3 panel plot for survival analysis. The first panel shows the differential Enrichment score (dES) for all samples, ranked by dES in their sections. The second (optional) panel shows the status of other attributes which may be present in the survival data frame for all samples. The third panel shows a Kaplan-Meier plot computed for the given survival data, with a curve for each section.

Usage

```
## S4 method for signature 'TNS'
tnsPlotKM(tns, regs = NULL, attribs = NULL,
  fname = "survplot", fpath = ".", xlab = "Months",
  ylab = "Survival probability", colorPalette = "bluered",
  plotpdf = FALSE, plotbatch = FALSE, width = 6.3, height = 3.6,
  panelWidths = c(3, 2, 4))
```

Arguments

`tns`	A TNS object, which must have passed GSEA2 analysis.
`regs`	An optional string vector specifying regulons to make the plot.
`attribs`	A numeric vector. Contains the columns of the survival data.frame which will be plotted for the second panel.
`fname`	A string. The name of the file in which the plot will be saved
`fpath`	A string. The path to the directory where the plot will be saved
`xlab`	A string. The label for the x axis on the third panel. This should be the measure of time shown in the survival data frame after the last check-up.
`ylab`	A string. The label for the y axis on the third panel
`colorPalette`	A string, which can be 'red', 'blue', 'redblue', or 'bluered'. Alternatively, it can be colors or hex values.
`plotpdf`	A logical value. If TRUE, the plot is saved as a pdf file. If false, it is plotted in the plotting area.
`plotbatch`	A logical value. If TRUE, plots for all regulons are saved in the same file. If FALSE, each plot for each regulon is saved in a different file.
`width`	A numeric value. Represents the width of the plot.
`height`	A numeric value. Represents the height of the plot.
`panelWidths`	A numeric vector of length=3 specifying the relative width of the internal panels.

Value

A plot, showing a graphical analysis for the 'tnsKM' function.

Examples

```
# load survival data
data(survival.data)

# load TNI-object
data(stni, package = "RTN")

stns <- tni2tnsPreprocess(stni, survivalData = survival.data,
        keycovar = c('Grade','Age'), time = 1, event = 2)
stns <- tnsGSEA2(stns)
stns <- tnsKM(stns)
tnsPlotKM(stns)
```

---

tnsPlotKmInteraction,TNS-method

Plot results from Kaplan-Meier analysis for dual regulons

Description

Plot results from Kaplan-Meier analysis for dual regulons

Usage

```
## S4 method for signature 'TNS'
tnsPlotKmInteraction(tns, dualreg = NULL,
  fname = "kmInteraction", fpath = ".", xlab = "Months",
  ylab = "Survival probability", colorPalette = "bluered", width = 4,
  height = 4, plotpdf = FALSE)
```

Arguments

`tns`	A TNS object, which must have passed GSEA2 analysis.
`dualreg`	A character string with the name of a dual regulon.
`fname`	A string. The name of the file in which the plot will be saved
`fpath`	A string. The path to the directory where the plot will be saved
`xlab`	A string. The label for the x axis on the third panel. This should be the measure of time shown in the survival data.frame after the last check-up.
`ylab`	A string. The label for the y axis on the third panel
`colorPalette`	A string, which can be 'red', 'blue', 'redblue', or 'bluered'. Alternatively, it can be a vector of five colors or hex values.
`width`	A numeric value. Represents the width of the plot.
`height`	A numeric value. Represents the height of the plot.
`plotpdf`	A logical value. If TRUE, the plot is saved as a pdf file. If false, it is plotted in the plotting area.

Value

A plot, showing a graphical analysis for the 'tnsKmInteraction' function.

Examples

```
# load survival data
data(survival.data)

# load TNI-object
data(stni, package = "RTN")

stns <- tni2tnsPreprocess(stni, survivalData = survival.data,
keycovar = c('Grade','Age'), time = 1, event = 2)
stns <- tnsGSEA2(stns)

# infer dual regulons
smbr <- tni2mbrPreprocess(stni)
smbr <- mbrAssociation(smbr)

# KM analysis for dual regulons
# stns <- tnsKmInteraction(stns, smbr, stepFilter=FALSE)
# tnsPlotKmInteraction(stns, dualreg = "FOXM1~PTTG1")
```

`tnsStratification` *Sample stratification for a TNS object*

Description

Internal function, used for sample stratification.

Usage

```
tnsStratification(tns, nSections = 1, center = FALSE)
```

Arguments

`tns`	a TNS object, which must have passed GSEA2 analysis.
`nSections`	A numeric value for the stratification of the sample. The larger the number, the more subdivisions will be created for the Kaplan-Meier analysis.
`center`	a logical value. If TRUE, numbers assigned to each group is centralized on regulon activity scale.

Value

An updated TNS object.

See Also

`tnsKM`

Examples

```
# see tnsKM method.
```

Index