Package: valorate (via r-universe)

August 28, 2024

Version 1.0-1

Date 2016-10-08

Title Velocity and Accuracy of the LOg-RAnk TEst

Author Victor Trevino [aut, cre]

Maintainer Victor Trevino <vtrevino@itesm.mx>

Depends R (>= 3.1.0), methods

Imports survival, graphics, utils, stats

Description The algorithm implemented in this package was designed to quickly estimates the distribution of the log-rank especially for heavy unbalanced groups. VALORATE estimates the null distribution and the p-value of the log-rank test based on a recent formulation. For a given number of alterations that define the size of survival groups, the estimation involves a weighted sum of distributions that are conditional on a co-occurrence term where mutations and events are both present. The estimation of conditional distributions is quite fast allowing the analysis of large datasets in few minutes <http://bioinformatica.mty.itesm.mx/valorate>.

License GPL (>= 2)

URL http://bioinformatica.mty.itesm.mx/valorate

NeedsCompilation yes

Date/Publication 2016-10-09 23:23:03

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

RemoteUrl https://github.com/cran/valorate

RemoteRef HEAD

RemoteSha bd40ce2d0f20e5e4e9063b540c88a34206e361cb

Contents

`new.valorate` *CREATE A VALORATE OBJECT*

Description

Creates a new valorate object from the survival information and basic parameters. `new.valorate`.

Usage

```
new.valorate(time, status, censored, rank, sampling.size=max(10000, 2e+05/events),
 min.sampling.size=1000, tails=2, sampling.ties=30,
 weights.method=c("logrank", "Wilcoxon", "Tarone-Ware", "Peto",
  "Flemington-Harrington", "Trevino", "user")[1],
 weights.parameters=list(p=1, q=1, t=3), weights,
    verbose=FALSE, save.sampling=TRUE, method="C",
  estimate.distribution.parameters=c("empirical","gaussian","beta","weibull")[1])
```

Arguments

`time`	character or numeric vector representing the survival time. If character it could be "76+" representing 76 units of time and censored. In this case, the censored parameter should not be provided.
`status`	numeric or logical vector representing the status (1 for event and 0 for censoring). This should be specified in the same order than the time argument and only if time was specified and does not include the censoring "+" indicator within.
`censored`	numeric or logical vector representing the censoring status (1 for censored and 0 for event). This should be specified in the same order than the time argument and only if time was specified and does not include the censoring "+" indicator within.
`rank`	a numeric or logical vector representing time-ordered subjects and whether they are events (1 or TRUE) or censoring observations (0 or FALSE). If rank is provided, time and status, censored should not. This argument is basically the 'c' vector of the log-rank formulation in the VALORATE publication.
`sampling.size`	a numeric value representing the length of random samples of the survival group vector (basically the 'x' vector, see the publication and references) that will be used to estimate the log-rank distribution. See the details section. The default is max(10000,200000/events).

`min.sampling.size`
: a numeric value representing the minimum number of random samples of the survival group vector. See the details section. The default is 1000.

`tails`
: a numeric value indicating whether the p-values generated will represent a 1 tail or two-tails (the default is two-tails).

`sampling.ties`
: a numeric value indicating the number of permutations of tie positions used for the estimation of the log-rank distribution. The default is 30.

`weights.method`
: a character specifying the type of log-rank test (see books in references). It can be "logrank" (default), "Wilcoxon", "Tarone-Ware", "Peto", "Flemington-Harrington", "Trevino",and "user". In case of "user", the 'weights' parameter should also be specified

`weights.parameters`
: a list of values for 'weights.method'. "Flemington-Harrington" uses a p and q parameters. "Trevino" uses a t parameter. The default is list(p=1,q=1,t=3).

`weights`
: a numeric vector having order according to the 'time' or 'rank' parameter only for "user" weights.method

`verbose`
: a logical value indicating whether estimation should show messages of partial calculations. The default is FALSE.

`save.sampling`
: a logical value indicating whether all sampling will be saved within the VALORATE object. The default is TRUE. This can be used to avoid saving all sampling and save memory. See details about memory usage.

`method`
: a character value either "R" or "C" that specify the implemented method of calculation. Both should generate same values but "C" is by far faster (default="C"). This can be used in cases where C calculations does not work for any reason or to compare methods and algorithms.

`estimate.distribution.parameters`
: a character vector containing subsets of "empirical","gaussian","beta",and "weibull". The default is "empirical". This has not been extensively explored but attempts to fit the observed log-rank distribution using sums of other distributions whose parameters are estimated after sampling. The "empirical" means nothing basically whereas "gaussian" for example means the estimation of mean and standard deviation of the observed conditional log-ranks. The results of these estimations can be viewd by valorate.plot.empirical or within the corresponding variables of the @subpop environment. This is experimental and is not intended for most users and applictions.

Details

The values in time and censored arguments do not need to be sorted by time but it is assumed that further specification of survival groups (e. g. calls to valorate.survdiff) will be provided in the corresponding order than that of this parameter. This function generates a VALORATE object prepared to run further analyses on the estimation of log-rank distributions for the specified population. See the 'value' section for details.

It is critical for computation time and memory the handling of the sampling. This can be managed by the sampling.size and min.sampling.size.

To save memory, save.sampling can be set to FALSE. In this case, valorate will estimate a ~1000 breaks histogram to store each conditional log-rank distribution. However, this strategy will lead in loosing resolution and therefore precision in the estimation of p-values. Thus, save.sampling=FALSE is not recommended for most applications.

Value

A valorate object.

`s` numeric vector representing the subjects ordered in time. This is basically the same than 'rank' argument if specified. So, this is the 'c' vector of the log-rank formulation in the VALORATE publication.

`n` the total number of subjects. It should be equal to the length of s.

`events` the total number of events observed. It should be equal to the sum of s.

`parameters` this is a list of the parameters specified.

`sampling.size` the 'total' number of sampling used to estimate the log-rank distribution. The computation time and memory depends largely in this value. Many numeric vectors will be created whose sum of their length will be approximately this value. This may be a copy of the original argument.

`min.sampling.size` the minimum number of sampling that will be used to estimate a conditional log-rank distribution (conditional to 'k' co-occurrences, see publication and references below). This may be a copy of the original argument.

`wcensored` a numeric vector denoting the positions of the 's' vector having censored subjects.

`wevents` a numeric vector denoting the positions of the 's' vector having events.

`order` the index positions of the time/censoring values needed to sort the subjects by time. This will be used in valorate.survdiff to re-accomodate the data specified.

`subpop` an environment of currently estimated log-rank distributions. The names for each item is given by 'subpop#' where '#' is the number of subjects in the survival group of interest (basically the n1 value that is equal to the sum of 1's within the 'x' vector). Each 'subpop' contains a list of values needed for the estimations and many are further indexed by the value of co-occurrences 'k', including 'sampling' which stores all log-rank conditional samplings, 'emp.hist' a tiny 'histogram' version of the estimated conditional distribution, 'combinations' the number of combinations of each co-occurrence, and 'k.density' its corresponding density or weights. See the tutorial within references for details.

`ties` a list of vectos having the positions of ties.

`sampling.ties` this is a copy of the original argument.

`tiesame` equal to ties.

`tiesame.pos` all positions having ties.

`tiesame.sampling` the actual value of samplings done to ties. 1 means no additional samplings.

`verbose` this is a copy of the original argument.

save.sampling	this is a copy of the original argument.
time	this is a copy of the original argument.
tails	this is a copy of the original argument.
weights.method	this is a copy of the original argument.
weights	this is a copy of the original argument.
method	this is a copy of the original argument.
samplings	this is depracated and has been moved to each subpop.

Author(s)

Victor Trevino <vtrevino@itesm.mx>

References

Trevino et al. 2016 http://bioinformatica.mty.itesm.mx/valorateR

David G. Kleinbaum and Mitchel Klein (2005). *Survival Analysis: A Self-Learning Text*. Second Edition. New York: Springer.

David Collett (2004). *Modelling survival data in medical research Collett David*. Second Edition. Chapman & Hall-CRC.

See Also

valorate.survdiff. valorate.plot.empirical.

Examples

```
## Create a random population of 100 subjects
## having 20 events
subjects <- numeric(100)
subjects[sample(100,20)] <- 1
vo <- new.valorate(rank=subjects, sampling.size=100000)

## print the structure of properties
str(vo)

## print slots
slotNames(vo)
```

prepare.n1 *ESTIMATES THE LOG-RANK DISTRIBUTION AND STORE IT WITHIN A VALORATE OBJECT*

Description

This method estimates the log-rank distribution for a minor size group equal to n1.

Usage

```
prepare.n1(vro, n1)
```

Arguments

vro	the valorate object.
n1	a numerical value (perhaps integer) of the size of the survival group.

Details

This method actually estimates the log-rank distribution for a minor size group equal to n1. This method is internally called by all functions to first check and/or compute the log-rank distribution of n1. It is not intended to be used by final users unless it is intended to prepare the VALORATE object before p-value calculations (perhaps in separated threds or jobs and saving/restoring the object).

Value

The updated valorate object. This is a S4 method operating an object, so the valorate object specified in argument will be updated with the estimations of the distribution of the log-rank for n1 subjects.

Author(s)

Victor Trevino <vtrevino@itesm.mx>

References

Trevino et al. 2016 http://bioinformatica.mty.itesm.mx/valorateR

See Also

new.valorate. valorate.survdiff.

Examples

```
## Create a random population of 100 subjects
## having 20 events
subjects <- numeric(100)
subjects[sample(100,20)] <- 1
vo <- new.valorate(rank=subjects, sampling.size=100000)
# and with verbose
vo <- new.valorate(rank=subjects, sampling.size=100000, verbose=TRUE)

## print the initial subpop
ls(vo@subpop) # should be character(0)

## calculate
prepare.n1(vo, 10) # should show messages of partial calculations P(L|k)

## print the current subpop
ls(vo@subpop) #should show now: [1] "subpop10"

names(vo@subpop[["subpop10"]]) #should show the internal names of the estimated subpop
```

`valorate.p.value` *ESTIMATES THE P-VALUE OF THE LOG-RANK TEST*

Description

Estimates the p-value using specific approximations to the log-rank.

Usage

```
valorate.p.value.sampling(vro, vrsubo, lrv, z)
valorate.p.value.chisq(vro, vrsubo, lrv, z)
valorate.p.value.normal(vro, vrsubo, lrv, z)
valorate.p.value.gaussian(vro, vrsubo, lrv, z)
valorate.p.value.weibull(vro, vrsubo, lrv, z)
valorate.p.value.beta(vro, vrsubo, lrv, z)
valorate.p.value.all(vro, vrsubo, lrv, z)
```

Arguments

`vro`	the valorate object.
`vrsubo`	the subpop list object (see prepare.n1) or a numeric value representing n1 used to obtain the subpop.
`lrv`	if provided, the log-rank value. It is needed for .sampling, .gaussian, .weibull, .beta, .normal, and .all .
`z`	if provided, the log-rank value in z-score (divided by the approximated standard deviation). It is needed for .normal, .chisq, optionally to .all if normal and chisq are required.

Details

This family of functions estimates the p-value of the log-rank test using specific approximations. The intended 'user' function in VALORATE is valorate.p.value.sampling, which is the function that is described in the publications. The rest of the functions are complementary for comparison with the classical approximations (chisq and normal) and for experimental purposes fitting each conditional log-rank distribution sampled (conditioned on k co-occurrences) with the specified distribution (gaussian, weibull, and beta). The function valorate.p.value.all is just a proxy to all calculations in the same function.

Value

the estimated p-value (times tails).

Author(s)

Victor Trevino <vtrevino@itesm.mx>

References

Trevino et al. 2016 http://bioinformatica.mty.itesm.mx/valorateR

See Also

new.valorate. valorate.survdiff. valorate.plot.empirical.

Examples

```
## Create a random population of 100 subjects
## having 20 events
subjects <- numeric(100)
subjects[sample(100,20)] <- 1
vo <- new.valorate(rank=subjects, sampling.size=100000, verbose=TRUE)

groups <- numeric(100)
groups[sample(100,4)] <- 1  # only 4 subjects are within the 'mutated' group
pvr <- valorate.survdiff(vo, groups)
print(pvr)

# the same than the value of pvr
valorate.p.value.sampling(vo, vo@subpop[["subpop4"]], attributes(pvr)[[1]]["LR"])

# the same than the value of pvr
valorate.p.value.sampling(vo, 4, attributes(pvr)[[1]]["LR"])

#classical approximations:
valorate.p.value.normal(vo, 4, attributes(pvr)[[1]]["LR"], attributes(pvr)[[1]]["Z"])
valorate.p.value.chisq(vo, 4, attributes(pvr)[[1]]["LR"], attributes(pvr)[[1]]["Z"])

# approximations of the conditional log-rank sampled density
valorate.p.value.gaussian(vo, 4, attributes(pvr)[[1]]["LR"])
valorate.p.value.beta(vo, 4, attributes(pvr)[[1]]["LR"])
```

```
valorate.p.value.weibull(vo, 4, attributes(pvr)[[1]]["LR"])

# all above can be get by:
valorate.p.value.all(vo, 4, attributes(pvr)[[1]]["LR"], attributes(pvr)[[1]]["Z"])

# Estimate a p-value a given log-rank
prepare.n1(vo, 50)
valorate.p.value.all(vo, 50, 0, 0) # 0 log-rank, 0 z-score
```

`valorate.plot.empirical` *PLOT THE SAMPLED (EMPIRICAL) LOG-RANK DISTRIBUTION*

Description

Plots the estimated density of the log-rank distribution.

Usage

```
valorate.plot.empirical(vro, n1, vstat, type, log, add, include, xlab, ylab,
 main, samp, smooth, legends, shades, transparency, lwd, xlim,
 minL=NA, minR=NA, ...)
```

Arguments

`vro`	the valorate object.
`n1`	the size of the 'mutated' or interested survival group. It can be also the numerical/logical 'x' vector as in valorate.survdiff in which case n1 and vstat are estimated.
`vstat`	log-rank statistic. If provided, a vertical mark on this value is shown and both sides of the density are filled to highlight areas.
`type`	typical plot parameter: "p"=points, "l"=lines (default), "o"=overlap.
`log`	typical plot parameter : specify which axis are shown in logarithm base 10.
`add`	typical plot parameter : specify whether the plot is new or added to an existing one.
`include`	specify which other estimations are included. This is experimental. Default "none", possibles: "none","gaussian","beta","weibull","all".
`xlab`	typical plot parameter: how the x axis is labelled, the default is "valorate LR".
`ylab`	typical plot parameter: how the y axis is labelled, the default is "density".
`main`	typical plot parameter. The default is an expression depending on the parameters.
`samp`	a numeric value specifying the length of sampling when using other density functions. It is related to the include argument.

smooth	the strength of density smoothing for display purposes. The default is 10.
legends	logical value that defines whether the legends for each curve should be displayed. The default is FALSE.
shades	define de colors used to shade the empirical distribution when the either 'vstat' argument is used or n1 represent the 'x' vector. The default is c(6,8).
transparency	defines the transparency in shades. The default is 0.25.
lwd	typical plot parameter: width of the lines. Default is 2.
xlim	typical plot parameter.
minL, minR	limits to the estimated empirical density.
...	arguments passed to plot.

Details

Plots the estimated density of the log-rank distribution.

Value

An invisible data frame of the density estimation.

Author(s)

Victor Trevino <vtrevino@itesm.mx>

References

Trevino et al. 2016 http://bioinformatica.mty.itesm.mx/valorateR

See Also

new.valorate. valorate.p.value. valorate.plot.empirical.

Examples

```
## Create a random population of 100 subjects
## having 20 events
subjects <- numeric(100)
subjects[sample(100,20)] <- 1
vo <- new.valorate(rank=subjects, sampling.size=100000, verbose=TRUE)

groups <- numeric(100)
groups[sample(100,4)] <- 1  # only 4 subjects are within the 'mutated' group
pvr <- valorate.survdiff(vo, groups)
print(pvr)

# Crude density
## Not run: valorate.plot.empirical(vo, 4)

# Similar but marking the statistic
# returned by groups and shading
```

```
## Not run: valorate.plot.empirical(vo, groups)

# Plot density and check symmetry
## Not run: valorate.plot.empirical(vo, 4, 0)

# Now should be almost symmetric
## Not run: valorate.plot.empirical(vo, 50, 0)

# Crude density plus gaussian, weibull, and beta estimations
## Not run: valorate.plot.empirical(vo, 4, include="all", legends=TRUE)
```

`valorate.plot.kaplan` *PLOT KAPLAN-MEIER CURVES*

Description

Plots the Kaplan-Meier Curves from two groups.

Usage

```
valorate.plot.kaplan(vro, clusters, p=valorate.survdiff(vro, clusters),
  main, short.names=TRUE, draw.all=FALSE, mark="|", mark.cex=0.75,
  margins=TRUE, col=2:3, col.all="skyblue")
```

Arguments

`vro`	the valorate object.
`clusters`	a numerical or logical vector representing the two survival groups encoded in 1/TRUE for those 'mutated' (in the group of interest) or 1/FALSE for those who not. Basically this value is the 'x' vector in the VALORATE re-formulation. See references.
`p`	the estimated p-value of the log-rank test. The default is valorate.survdiff(vro, clusters).
`main`	typical plot parameter. The default is an expression depending on the parameters.
`short.names`	if TRUE (default) use 'LR' instead or 'Log-Rank' and 'HR' instead or 'Hazard-Ratio' in legends.
`draw.all`	if TRUE, the plot includes also the survival curve of all subjects before stratification.
`mark`	character to mark censoring. The default is "|".
`mark.cex`	the character expansion. The default is 0.75
`margins`	if TRUE (default) set the margins properly.
`col`	specifies the colors for survival curves. The default is 2:3 (red for cluster=0, green for cluster=1).
`col.all`	specifies the color when draw.all is TRUE. The default is "skyblue".

Details

Plots the estimated Kaplan-Meier survival curves from data.

Value

Nothing.

Author(s)

Victor Trevino <vtrevino@itesm.mx>

References

Trevino et al. 2016 http://bioinformatica.mty.itesm.mx/valorateR

See Also

new.valorate. valorate.survdiff.

Examples

```
## Create a random population of 100 subjects
## having 20 events
subjects <- numeric(100)
subjects[sample(100,20)] <- 1
vo <- new.valorate(rank=subjects, sampling.size=100000, verbose=TRUE)

groups <- numeric(100)
groups[sample(100,20)] <- 1  # 20 to likely see some difference
pvr <- valorate.survdiff(vo, groups)
print(pvr)

## Not run: valorate.plot.kaplan(vo, groups, main="Two Curves")

## Not run: valorate.plot.kaplan(vo, groups, draw.all=TRUE,
    main="Three Curves (Including All Data)")
## End(Not run)
```

---

valorate.plot.sampling.densities

PLOT CO-OCCURRENCE DENSITIES FORMING A LOG-RANK DISTRIBUTION

Description

Plots the densities of each co-occurrence that shapes the final log-rank distribution for a n1 group size.

Usage

```
valorate.plot.sampling.densities(vro, n1,
  type, log, xlim, ylim, ncol, main,
  rug, add, w.sum, sampling,
  weighted,legends.cex, weights.cex,
  weights.pos, w.sum.lwd=3,
  y.limit=1e-13,
  ...)

valorate.plot.sampling.densities.figure(vro, n1, type, log,
  xlim, ylim, main, rug, rug.size,
  sub, w.sum, sampling, ncol,
  y.limit=1e-13, ...)
```

Arguments

vro	the valorate object.
n1	the size of the 'mutated' or interested survival group. It can be also the numerical/logical 'x' vector as in valorate.survdiff in which case n1 and vstat are estimated.
type	typical plot parameter: "p"=points, "l"=lines (default), "o"=overlap.
log	typical plot parameter : specify which axis are shown in logarithm base 10.
xlim	typical plot parameter.
ylim	typical plot parameter.
ncol	number of columns for legends.
main	typical plot parameter. The default is an expression depending on the parameters.
rug	if FALSE removes the drawing of rugs.
add	if FALSE assumes plots are added to existing one. Not valid for all functions.
w.sum	if FALSE removes the drawing of the weighted sum distribution (the final log-rank distibution).
sampling	if TRUE includes the drawing of a crude-histogram version of the overall distribution (the final log-rank distibution).
weighted	if TRUE the densities of each co-occurrence is weighted to its overall contribution (proportion of combinations). Nice! to explain the overall distribution.
legends.cex	the character expansion for legends.
weights.cex	the character expansion for legends of the weights
weights.pos	the position of the weights "middle"=above each curve, "left", or "right".
w.sum.lwd	the line width of the weighted sum line.
...	parameters forwarded to plot.
rug.size	determine the size of rugs made in valorate.plot.sampling.densities.figure
sub	specifies the character to include in each plot.
y.limit	specifies the lowest density value. The default is 1e-13. This is useful for log="y".

Details

Plots the densities of each co-occurrence that shapes the final log-rank distribution for a n1 group size. `valorate.plot.sampling.densities` plots all co-occurrences in the same figure whereas `valorate.plot.sampling.densities.figure` plots all in separated figures.

Value

Nothing.

Author(s)

Victor Trevino <vtrevino@itesm.mx>

References

Trevino et al. 2016 http://bioinformatica.mty.itesm.mx/valorateR

See Also

new.valorate. valorate.survdiff.

Examples

```
## Create a random population of 100 subjects
## having 20 events
subjects <- numeric(100)
subjects[sample(100,20)] <- 1
vo <- new.valorate(rank=subjects, sampling.size=100000, verbose=TRUE)

## Not run: valorate.plot.sampling.densities(vo, 5)
## Not run: valorate.plot.sampling.densities(vo, 10)
## Not run: valorate.plot.sampling.densities(vo, 20, weighted=TRUE)

## Not run: valorate.plot.sampling.densities.figure(vo, 5)
```

---

valorate.plot.subpop *PLOT ALL ESTIMATED LOG-RANK DISTRIBUTIONS*

Description

Plots all log-rank distributions estimated with the same object (different values of n1). This family of plots is commonly used to compare the estimated distributions.

Usage

```
valorate.plot.subpop.empirical(vro, which,
  type, log, xlim, smooth, legends,
  density, ylim, ...)

valorate.plot.subpop.empirical.to.0(vro, which,
  type, log, xlim, smooth, legends, density, ylim, ...)

valorate.plot.subpop.empirical.scaled(vro, which,
  type, log, xlim,
  smooth, legends, density, ylim,
  scale.point, ...)
```

Arguments

vro	the valorate object.
which	The values of n1 that will be shown. NULL to plot them all.
type	typical plot parameter: "p"=points, "l"=lines (default), "o"=overlap.
log	typical plot parameter : specify which axis are shown in logarithm base 10.
xlim	typical plot parameter.
ylim	typical plot parameter.
smooth	the strength of density smoothing for display purposes. The default is 10.
legends	the number of columns in legends. 0 to omit legends.
density	indicates whether all curves should represent density (default to TRUE). FALSE to scale to maximum.
...	arguments passed to plot.
scale.point	a double between 0 and 0.5 (exclusive) that determines the two points in quantiles in which all densities will be 'equalized'. The quantiles are scale.point and 1-scale.point. 0.5 should be avoided.

Details

`valorate.plot.subpop.empirical` plots all log-rank distributions estimated with the same object (different values of n1) in raw densities and scales. `valorate.plot.subpop.empirical.to.0` is similar to `valorate.plot.subpop.empirical` but shift distributions to 0 and scale horizontal axis to similar limits. `valorate.plot.subpop.empirical.scaled` is similar to `valorate.plot.subpop.empirical` but scales the distributions to have the same scale.point(s) (in x) for all distributions. It also shifts all distribution to zero. This helps to compare the tendencies of the overall distributions.

Value

Nothing.

Author(s)

Victor Trevino <vtrevino@itesm.mx>

References

Trevino et al. 2016 http://bioinformatica.mty.itesm.mx/valorateR

See Also

new.valorate. valorate.survdiff.

Examples

```
## Create a random population of 100 subjects
## having 20 events
subjects <- numeric(100)
subjects[sample(100,20)] <- 1
vo <- new.valorate(rank=subjects, sampling.size=100000, verbose=TRUE)

for (i in c(5,10,20,30,40,50)) {
  groups <- numeric(100)
  groups[sample(100,i)] <- 1
  valorate.survdiff(vo, groups)
}

## Not run: valorate.plot.subpop.empirical(vo)
## Not run: valorate.plot.subpop.empirical.to.0(vo)
```

---

valorate.risk *ESTIMATES RISK*

Description

Estimates the risk (hazard ratio), and confidence interval of a 'mutated' group.

Usage

```
valorate.risk(vro, clusters)
```

Arguments

vro	the valorate object.
clusters	a numerical or logical vector representing the two survival groups encoded in 1/TRUE for those 'mutated' (in the group of interest) or 1/FALSE for those who not. Basically this value is the 'x' vector in the VALORATE re-formulation. See references.

Details

A coxph model depending on clusters is run to establish the risk/hazard ratio.

Value

A number representing the relative risk. The confidence interval, p-value, and the coxph model are included as attributes.

Author(s)

Victor Trevino <vtrevino@itesm.mx>

References

Trevino et al. 2016 http://bioinformatica.mty.itesm.mx/valorateR

See Also

new.valorate. valorate.survdiff. coxph (survival package).

Examples

```
## Create a random population of 100 subjects
## having 20 events
subjects <- numeric(100)
subjects[sample(100,20)] <- 1
vo <- new.valorate(rank=subjects, sampling.size=100000, verbose=TRUE)

groups <- numeric(100)
groups[sample(100,20)] <- 1  # 20 to likely see some difference
pvr <- valorate.survdiff(vo, groups)
print(pvr)

valorate.risk(vo, groups)
```

---

valorate.survdiff *ESTIMATES THE P-VALUE AND STATISTICS OF THE LOG-RANK TEST*

Description

Estimates the p-value using the VALORATE calculation.

Usage

```
valorate.survdiff(vro, clusters, p.func)
```

Arguments

vro	the valorate object.
clusters	a numerical or logical vector representing the two survival groups encoded in 1/TRUE for those 'mutated' (in the group of interest) or 1/FALSE for those who not. Basically this value is the 'x' vector in the VALORATE re-formulation. See references.
p.func	the function that provides the estimation. The default is valorate.p.value.sampling. See valorate.p.value.

Details

The main function to estimate the p-value of the difference of two survival curves under the VALORATE algorithm. Because the definition of the survival group as '1' or '0' is arbitrary, the actual calculation is performed over the less frequent group. From clusters and s=sum(clusters), n1 is determined as the min(s, length(clusters)-s). Then a call for prepare.n1 is performed and finally the p-value estimations. The statistics are added as attributes.

Value

the estimated p-value.

Author(s)

Victor Trevino <vtrevino@itesm.mx>

References

Trevino et al. 2016 http://bioinformatica.mty.itesm.mx/valorateR

See Also

new.valorate. valorate.p.value. valorate.plot.empirical.

Examples

```
## Create a random population of 100 subjects
## having 20 events
subjects <- numeric(100)
subjects[sample(100,20)] <- 1
vo <- new.valorate(rank=subjects, sampling.size=100000, verbose=TRUE)

groups <- numeric(100)
groups[sample(100,4)] <- 1  # only 4 subjects are within the 'mutated' group
pvr <- valorate.survdiff(vo, groups)
print(pvr)
```

Index