

Package ‘coxphw’

January 2, 2012

Type Package

Title Weighted Cox regression

Version 2.11

Date 2011-10-25

Author R by Meinhard Ploner, Fortran by Georg Heinze

Maintainer <georg.heinze@meduniwien.ac.at>

Depends R (>= 2.13.1), survival

Description Weighted estimation for Cox regression

License GPL

Repository CRAN

Date/Publication 2011-10-26 17:59:29

R topics documented:

coxphw	2
fp.power	7
plotfp	9
plotw	11
pow2	13
pow3	14
powM0.5	15
powM1	16
powM2	17
PT	18
RI	19
Rlog	20
Rpow2	21
Rpow3	22
RpowM0.5	23

RpowM1	24
RpowM2	25
Rsqrt	26

Index	27
--------------	-----------

coxphw	<i>Weighted estimation in Cox regression</i>
--------	--

Description

This package implements weighted estimation in Cox regression, as proposed by Schemper, Wakounig and Heinze (2009). Weighted estimation provides unbiased average hazard ratio estimates in case of non-proportional hazards. An experimental feature allows the estimation of nonlinear effects using fractional polynomials. This feature can also be used to estimate the interaction of a covariate with a nonlinear function of survival time.

Usage

```
coxphw( formula=attr(data, "formula"), data=sys.parent(), breslow=NA,
prentice=NA, taroneware=NA, id=NULL, robust=FALSE,
jack=FALSE, normalize=TRUE, scale.weights=1, offset=NULL,
alpha=0.05, alpha.fp=c(0.2, 0.05, 0.05), fp.iter=10, fp.max=2, maxit=200,
maxhs=5, xconv=1e-4, gconv=1e-4, maxstep=1, x=TRUE, y=TRUE,
censcorr=FALSE, trunc.weights=1, round.times.to=0.00001, add.constant=0,
print=TRUE, sorted=FALSE, AHR=TRUE, ARE=FALSE, PH=FALSE, AHR.norobust=FALSE, pc=TRUE, pc.time=TRUE, . . .
```

Arguments

formula	a formula object, with the response on the left of the operator, and the model terms on the right. The response must be a survival object as returned by the 'Surv' function. formula may include fp()-terms (see section on fractional polynomials).
data	a data.frame in which to interpret the variables named in the formula argument.
prentice	a righthand formula (e. g., ~A+B) with the terms which should be estimated using Prentice (survival function) weights, or use prentice=TRUE to apply these weights to all model terms
breslow	a righthand formula with the terms which should be estimated using Breslow (N at risk) weights, or TRUE
taroneware	a righthand formula with the terms which should be estimated using Tarone-Ware (square root of N at risk) weights, or TRUE. Note that specifications in prentice overrule breslow, and breslow overrules taroneware.
alpha.fp	A vector of length 2 which specifies p-values to include an fp()-term: alpha.fp[1] defines the threshold p-value for keeping at least the linear term in the model, alpha.fp[2] defines the threshold p-value for the nonlinear term. Default is c(0.20, 0.05, 0.05).

fp.iter	Number of iterations for big fp-loop, default=10. This is important if several fp()-terms are included in the model as inclusion or exclusion of one term based on the threshold p-value may increase or decrease the significance of others.
fp.max	Highest power of fp() terms (applies to all fp-terms). Select 1 or 2 (default=2).
normalize	if TRUE, weights are normalized such that their sum is equal to the number of events. May speed up or enable convergence if for some variables no weighting is used.
alpha	the significance level ($1-\alpha$ = the confidence level), 0.05 as default.
maxit	maximum number of iterations (default value is 200)
maxhs	maximum number of step-halvings per iterations (default value is 5). The increments of the parameter vector in one Newton-Rhaphson iteration step are halved, unless the new likelihood is greater than the old one, maximally doing maxhs halvings.
xconv	specifies the maximum allowed change in standardized parameter estimates to declare convergence. Default value is 0.0001.
gconv	specifies the maximum allowed change in score function to declare convergence. Default value is 0.0001.
maxstep	specifies the maximum change of (standardized) parameter values allowed in one iteration. Default value is 1.
x	requests copying explanatory variables into output object
y	requests copying survival information into output object
id	a vector of patient identification numbers, must be integers starting from 1. These IDs are used for computing the robust covariance matrix. If id=NA (the default) the program assumes that each line of the data set refers to a distinct individual.
robust	set to TRUE if the robust variance estimate should be computed. This robust estimate is computed from $V=D'D$ where D is the matrix of dfbeta residuals which are computed by $D'D = A^{-1}U'UA^{-1}$ with A denoting the weighted sum of contributions to the second derivative of the log likelihood and U denoting the nxp matrix of score residuals.
jack	set to TRUE if the variance should be based on a complete jackknife. Each individual (as identified by distinct values of the variable specified in id) is left out in turn. The resulting matrix of dfbeta residuals D is then used to compute the variance matrix: $V=D'D$.
offset	specifies a variable which is included in the model but its parameter estimate is fixed at 1.
scale.weights	specifies a scaling factor for the weights
trunc.weights	specifies a quantile at which the weights are to be truncated. Can be used to increase the precision of the estimates, particularly if censcorr=TRUE is used. Default=1 (no truncation). Recommended value is 0.95 for mild truncation.
censcorr	If set to TRUE, the weights are multiplied by the inverse marginal Kaplan-Meier estimator with reverse status indicator (default=F).
AHR	If set to TRUE, applies a template for average hazard ratio estimation (Schemper, Wakounig and Heinze, 2009): it sets prentice=TRUE, censcorr=TRUE, robust=TRUE

AHR.norobust	If set to TRUE, applies a template for average hazard ratio estimation but with the Lin-Sasieni covariance: it sets prentice=TRUE, censcorr=TRUE, robust=FALSE
ARE	If set to TRUE, applies a template for average regression effect estimation (Xu and O'Quigley, 2000): it sets prentice=NA, breslow=NA, taroneware=NA, censcorr=TRUE, robust=TRUE
PH	If set to TRUE, applies a template for proportional hazards regression (similar to coxph): it sets prentice=NA, breslow=NA, taroneware=NA, censcorr=FALSE, robust=TRUE
round.times.to	rounds survival times to the nearest number that is a multiple of round.times.to. May improve numerical stability if very low survival times are used (mostly in simulations).
add.constant	this number will be added to stop times, and add.constant/2 will be added to start times
pc	if set to TRUE, will transform model matrix to speed up convergence during fp mode. default=TRUE.
pc.time	if set to TRUE, will transform the time dependent covariates (interactions of covariates with functions of time) to speed up convergence. default=TRUE.
print	requests echoing of intermediate results (particularly, for fp estimation)
sorted	if set to TRUE, the data set will not be sorted prior to passing it to FORTRAN. May speed up computations.
...	additional arguments, e.g., km can be used as synonym of prentice, or N for breslow

Details

If Cox's proportional hazards regression model is used in the presence of non-proportional hazards, i.e., with underlying time-dependent hazard ratios of prognostic factors, the average relative risk for such a factor is under- or overestimated and testing power for the corresponding regression parameter is reduced. In such a situation weighted estimation of this parameter provides a parsimonious alternative to more elaborate modelling of time-dependent effects. Weighted estimation in Cox regression (WCR) extends the tests by Breslow and Prentice to a multi-covariate situation as does the Cox model to Mantel's logrank test. WCR can also be seen as a robust alternative to the standard Cox estimator, reducing the influence of outlying survival times on parameter estimates. Schemper (1992) first demonstrated the suitability of WCR for estimating average hazard ratios when hazards are non-proportional and Sasieni (1993) extensively investigated the favorable properties of WCR.

Weighted estimation assigns weights to risk sets, according to the survival function estimates (Prentice weights), the number of subjects at risk (Breslow weights), or according to their square roots (Tarone-Ware weights). These weights are applied to the summands of the score function. The final estimate is the vector of parameter values which equates the score function to 0. Since there is one score function corresponding to each parameter of the model, weights may be applied to some but not necessarily to all parameters of a model.

Breslow's tie-handling method is used by the program, other methods to handle ties are currently not available.

By default, the program estimates the covariance matrix using Lin (1991) and Sasieni (1993) sandwich estimate $A^{-1}BA^{-1}$ with $-A$ and $-B$ denoting the sum of contributions to the second derivative of the log likelihood, weighted by $w(t_j)$ and $w(t_j)^2$, respectively. This estimate is independent

from the scaling of the weights and reduces to the inverse of the information matrix in case of no weighting.

As an experimental feature, the RA-2 algorithm of Royston and Altman (1994) (cf. also Royston and Sauerbrei, 2008) to select *fractional polynomial* transformations for continuous variables was implemented. This allows to obtain for flexible nonlinear estimation. The method can also be applied to survival time in a time by covariate interaction. For some examples see the documentation of the `plotfp` function.

Value

<code>coefficients</code>	the parameter estimates
<code>alpha</code>	the significance level = 1 - confidence level
<code>var</code>	the estimated covariance matrix
<code>df</code>	the degrees of freedom
<code>loglik</code>	the null and maximized (penalized) log likelihood
<code>method.ties</code>	the ties handling method
<code>iter</code>	the number of iterations needed to converge
<code>n</code>	the number of observations
<code>y</code>	the response
<code>formula</code>	the model formula
<code>means</code>	the means of the covariates
<code>linear.predictors</code>	the linear predictors
<code>method</code>	the estimation method (usually weighted estimation)
<code>method.ci</code>	the confidence interval estimation method (Profile Likelihood or Wald)
<code>ci.lower</code>	the lower confidence limits
<code>ci.upper</code>	the upper confidence limits
<code>prob</code>	the p-values
<code>call</code>	the function call
<code>cov.ls</code>	the covariance matrix computed by the Lin-Sasieni method (the default method)
<code>cov.lw</code>	the covariance matrix computed by the Lin-Wei method (robust covariance, only computed if <code>robust==T</code>)
<code>cov.j</code>	the covariance matrix computed by the jackknife method (only computed if <code>jack==T</code>)
<code>cov.method</code>	the method used to compute the (displayed) covariance matrix and the standard errors. This method is either "jack" if <code>jack==T</code> , or "Lin-Wei" if <code>robust==T</code> and <code>jack==F</code> , or "Lin-Sasieni" if <code>robust==F</code> and <code>jack==F</code> .
<code>w.matrix</code>	A matrix with 4 columns and rows according to the number of uncensored failure times. The first column contains the failure times, the remaining columns (labeled <code>w.raw</code> , <code>w.obskm</code> , and <code>w</code>) contain the raw weights, the weights according to the inverse of the Kaplan-Meier estimates with reverse status indicator and the normalized product of both.
<code>dataline</code>	The first dataline of the input data set (needed for <code>plotfp</code> function)

Note

A SAS macro WCM with similar functionality (but without fp options) is offered for download at <http://www.meduniwien.ac.at/cemsiis/kb/>

Author(s)

Georg Heinze and Meinhard Ploner

References

Lin D and Wei L (1989). The robust inference for the Cox proportional hazards model. *Journal of the American Statistical Association* 84, 1074-1078.

Lin D (1991). Goodness-of-fit analysis for the Cox regression model based on a class of parameter estimators. *Journal of the American Statistical Association* 86, 725-728.

Royston P and Altman D (1994). Regression Using Fractional Polynomials of Continuous Covariates: Parsimonious Parametric Modelling. *Applied Statistics* 43, 429-467.

Royston P and Sauerbrei W (2008). *Multivariable Model-building. A pragmatic approach to regression analysis based on fractional polynomials for modelling continuous variables*. Wiley, Chichester, UK.

Sasieni P (1993). Maximum Weighted Partial Likelihood Estimators for the Cox Model. *Journal of the American Statistical Association* 88, 144-152.

Schemper M (1992). Cox analysis of survival data with non-proportional hazard functions. *The Statistician* 41, 455-465.

Schemper M, Wakounig S and Heinze G (2009). The estimation of average hazard ratios by weighted Cox regression. *Statistics in Medicine* 28, 2473-2489.

Xu R and O'Quigley J (2000). Estimating average regression effect under non-proportional hazards. *Biostatistics* 1, 423-439.

See Also

coxph

Examples

```
# gastric cancer data set
gastric <-
  structure(list(patnr = as.integer(c(46, 1, 2, 3, 4, 5, 47, 6,
    7, 8, 9, 48, 10, 11, 49, 12, 13, 14, 50, 15, 16, 17, 18, 19,
    20, 51, 21, 22, 52, 23, 53, 54, 55, 24, 25, 56, 57, 58, 59, 60,
    61, 62, 63, 64, 26, 65, 27, 66, 28, 29, 67, 68, 69, 70, 30, 71,
    31, 72, 32, 73, 33, 34, 74, 75, 76, 77, 78, 35, 79, 36, 80, 81,
    82, 37, 38, 39, 83, 84, 40, 85, 41, 86, 87, 88, 42, 43, 44, 89,
    90, 45)),
    treat = as.integer(c(0, 1, 1, 1, 1, 1, 0, 1, 1, 1,
    1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0,
    0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0,
    0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0,
    1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1))),
```


Description

Provides fractional polynomials as accessible function

Usage

```
fp.power(z, a, b=NULL)
```

Arguments

z	a scalar or vector of positive numerical values
a	first power
b	optional second power

Details

The function returns fp(a) of z (and optionally fp(b) of z).

Value

a matrix with one or two columns (if a second power b was specified), and number of rows equal to the length of z. The columns are sorted by descending power.

Author(s)

Georg Heinze

See Also

coxph

Examples

```
z<-c(1,4,6)
fp.power(z,1)
fp.power(z,0.5)
fp.power(z,0.5,0.5)
fp.power(z,0,2)
```

plotfp *Compute and plot nonlinear effect estimated by fractional polynomials*

Description

This function enables visualizing a nonlinear effect estimated by the fractional polynomial functionality of `coxphw`. It plots the log relative hazard versus values of a continuous covariable.

Usage

```
plotfp(obj, plot.x = NA, ref = NA, plot = TRUE, treatment=NULL, ref.type="value", variable=NULL, xlab=NA)
```

Arguments

<code>obj</code>	an output object of <code>coxphw</code> , which includes <code>fp()</code> terms
<code>plot.x</code>	The data values for the continuous variable (e. g., <code>plot.x=30:70</code>)
<code>ref</code>	A reference value. The log relative hazard at this value will be 0.
<code>plot</code>	If TRUE, requests a plot. Otherwise, only an output object with x values and log relative hazards will be computed.
<code>treatment</code>	variable which is in interaction with fp variable (use "")
<code>ref.type</code>	"value" for simple fp term (no interaction), "interaction.time" for interaction for treatment with <code>fp(time)</code> , "interaction.treat" for interaction of treatment with <code>fp(variable)</code> , or any value to specify the level of the treatment variable for which the fp of "variable" should be plotted
<code>variable</code>	name of fp variable (use "")
<code>xlab</code>	label for x-axis of plot, uses variable specified in "variable" as default
<code>ylab</code>	label for y-axis of plot, uses "log relative hazard" as default
<code>...</code>	further parameters, to be used for plots (e.g., scaling of axes)

Details

After having fit an fp model, this function can be used to depict the nonlinearity. It supports simple non-linear effects as well as interaction effects of continuous variables with binary covariates (see examples section).

Value

`obj` A vector with log relative hazard values.

Author(s)

Georg Heinze and Meinhard Ploner

See Also

`coxphw`

Examples

```

#Example 1
set.seed(30091)
n<-300
x<-1:n
true.func<-function(x) 3*(x/100)^{2}-log(x/100)-3*x/100
x<-round(rnorm(x)*10+40,0)
#x<-round(runif(x)*60+10,0)
time<-rexp(n,rate=1)/exp(true.func(x))
event<-rep(1,n)
fuptime<-runif(n,0,309000)
event<-(time<fuptime)+0
time[event==0]<-fuptime[event==0]
my.data<-data.frame(x,time,event)

fit<-coxphw(data=my.data, Surv(time,event)~fp(x),alpha.fp=c(1,0.05,0.05), AHR=TRUE, maxit=200)
fit1<-coxphw(data=my.data, Surv(time,event)~fp(x),alpha.fp=c(1,0.05,0.05), PH=TRUE)

# estimated function
y<-plotfp(fit, plot.x=10:70, ref=40, variable="x", ref.type="value")
y1<-plotfp(fit1, plot.x=10:70, ref=40, variable="x", ref.type="value")
# true function
x<-10:70
lines(x, true.func(x)-true.func(40))
lines(x, y1, lty=2)
legend(x=40, y=(max(true.func(x)-true.func(40))+min(true.func(x)-true.func(40)))/2, lty=c(NA,2,1),pch=c(1,NA,NA))

#Example 2
n<-200
x<-1:n
true.func<-function(x) 2.5*log(x)-2
#x<-round(rnorm(x)*10+40,0)
x<-round(runif(x)*60+10,0)

time<-round(100000*rexp(n,rate=1)/exp(true.func(x)),1)
event<-rep(1,n)
my.data<-data.frame(x,time,event)

# select best fp(1) model, without testing in RA2 algorithm
fit<-coxphw(data=my.data, Surv(time,event)~fp(x),alpha.fp=c(1,1,0.05), fp.max=1, AHR=TRUE)

# estimated function
y<-plotfp(fit,plot.x=10:70, ref=40, ref.type="value", variable="x", type="p")
# true function
x<-10:70
lines(x, true.func(x)-true.func(40))
legend(x=40, y=(max(true.func(x)-true.func(40))+min(true.func(x)-true.func(40)))/2, lty=c(NA,1),pch=c(1,NA),leg

#interaction of continuous variable with binary "treatment" variable

n<-200

```

```

trt<-rbinom(n,1,0.5)
x<-1:n
true.func<-function(x) 2.5*log(x)-2
#x<-round(rnorm(x)*10+40,0)
x<-round(runif(x)*60+10,0)

time<-100*rexp(n,rate=1)/exp(true.func(x)/4*trt-(true.func(x)/4)^2*(trt==0))
event<-rep(1,n)
my.data<-data.frame(x, trt, time, event)
fit<-coxphw(data=my.data, Surv(time,event)~fp(x)*trt,alpha.fp=c(1,1,0.05), AHR=TRUE)

# plots the interaction of trt with x (the effect of trt dependent on the values of x):
y<-plotfp(fit,variable="x",treatment="trt",ref.type="interaction.treat",plot.x=10:70)

# plots the effect of x in subjects with trt=0:
y0<-plotfp(fit,variable="x",treatment="trt",ref.type=0,plot.x=10:70, ref=40)

# plots the effect of x in subjects with trt=1:
y1<-plotfp(fit,variable="x",treatment="trt",ref.type=1,plot.x=10:70, ref=40)

#interaction of binary "treatment" variable with function of time

time<-100*rexp(n,rate=1)/exp((true.func(x)/10)^2/2000*trt+trt)
event<-rep(1,n)
my.data<-data.frame(x, trt, time, event)

fit<-coxphw(data=my.data, Surv(time,event)~fp(time)*trt+x, alpha.fp=c(1,1,0.05), fp.max=2, AHR=TRUE)
y<-plotfp(fit,variable="time", treatment="trt", plot.x=seq(1,100,1), ref.type="interaction.time", type="1")

fit2<-coxphw(data=my.data, Surv(time,event)~fp(time)*trt+x, alpha.fp=c(1,1,0.05), fp.max=2, PH=TRUE)
y2<-plotfp(fit2,variable="time", treatment="trt", plot.x=seq(1,100,1), ref.type="interaction.time", type="1")

plot(x=1:100, xlab="time", y=y, ylab="log relative hazard", pch=1, ylim=c(min(y,y2),max(y,y2)))
lines(x=1:100, y=y2, lty=1)
legend(x=60, y=max(y,y2), lty=c(NA,1), pch=c(1,NA), legend=c("AHR","PH"))

```

plotw

Plot weights of weighted estimation in Cox regression

Description

This function plots the weights used in a weighted Cox regression analysis against time.

Usage

```
plotw(x, rank=FALSE, log=FALSE, ...)
```



```

      1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0,
      0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0))) ,
.Names = c("patnr",
  "treat", "time", "status"), class = "data.frame",
row.names = c("1",
  "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12", "13",
  "14", "15", "16", "17", "18", "19", "20", "21", "22", "23", "24",
  "25", "26", "27", "28", "29", "30", "31", "32", "33", "34", "35",
  "36", "37", "38", "39", "40", "41", "42", "43", "44", "45", "46",
  "47", "48", "49", "50", "51", "52", "53", "54", "55", "56", "57",
  "58", "59", "60", "61", "62", "63", "64", "65", "66", "67", "68",
  "69", "70", "71", "72", "73", "74", "75", "76", "77", "78", "79",
  "80", "81", "82", "83", "84", "85", "86", "87", "88", "89", "90"
))

```

```
# weighted estimation of average hazard ratio
```

```
fit2<-coxphw(data=gastric, Surv(time,status)~treat, AHR=TRUE)
plotw(fit2)
```

```
# estimation of average regression effect by inverse probability of censoring weights; truncate weights at 95th perc
```

```
fit2<-coxphw(data=gastric, Surv(time,status)~treat, ARE=TRUE, trunc.weights=0.95)
plotw(fit2)
```

pow2

Provides 'power to the 2' as accessible function

Description

Provides 'power to the 2' as accessible function

Usage

```
pow2(z)
```

Arguments

z a scalar or vector of numerical values

Details

The function returns z^2 .

Value z^2 **Author(s)**

Georg Heinze

See Also

coxph

Examples

```
a<-c(1,4,6)
pow2(a)

# returns: [1] 1 16 36
```

pow3

Provides 'power to the 3' as accessible function

Description

Provides 'power to the 3' as accessible function

Usage $\text{pow3}(z)$ **Arguments**

z a scalar or vector of numerical values

Details

The function returns z^3 .

Value z^3 **Author(s)**

Georg Heinze

See Also

coxph

Examples

```
a<-c(1,4,6)
pow3(a)

# returns: [1] 1 64 216
```

powM0.5

Provides 'power to the minus 0.5' as accessible function

Description

Provides 'power to the minus 0.5' as accessible function

Usage

```
powM0.5(z)
```

Arguments

z a scalar or vector of numerical values

Details

The function returns $z^{-0.5}$.

Value

$z^{-0.5}$

Author(s)

Georg Heinze

See Also

coxph

Examples

```
a<-c(1,4,6)
powM0.5(a)

# returns: [1] 1.0000000 0.5000000 0.4082483
```

powM1	<i>Provides 'power to the minus 1' as accessible function</i>
-------	---

Description

Provides 'power to the minus 1' as accessible function

Usage

```
powM1(z)
```

Arguments

`z` a scalar or vector of numerical values

Details

The function returns z^{-1} .

Value

z^{-1}

Author(s)

Georg Heinze

See Also

coxph

Examples

```
a<-c(1,4,6)
powM1(a)

# returns: [1] 1.0000000 0.2500000 0.1666667
```

powM2	<i>Provides 'power to the minus 2' as accessible function</i>
-------	---

Description

Provides 'power to the minus 2' as accessible function

Usage

```
powM2(z)
```

Arguments

`z` a scalar or vector of numerical values

Details

The function returns z^{-2} .

Value

z^{-2}

Author(s)

Georg Heinze

See Also

coxph

Examples

```
a<-c(1,4,6)
powM2(a)
a
# returns: [1] 1.00000000 0.06250000 0.02777778
```

PT	<i>Pretransformation function</i>
----	-----------------------------------

Description

Provides automatic pretransformation of variables (to well-scaled and nonzero values)

Usage

PT(z)

Arguments

z a vector of numerical values

Details

The function transforms a variable by shifting to positive values, and dividing by scaling factor (a power of 10) such that the SD is approximately equal to 1.

Value

(a+shift)/scale

Author(s)

Georg Heinze

See Also

coxph

Examples

```
a<-c(-6, -1, 4, 6)
PT(a)
# returns: [1] 0.2 0.7 1.2 1.4
```

RI *Provides 'repeated identity' as accessible function*

Description

Provides 'repeated identity' as accessible function

Usage

RI(z)

Arguments

z a scalar or vector of numerical values

Details

The function returns $z * \log(z)$.

Value

$z * \log(z)$

Author(s)

Georg Heinze

See Also

coxph

Examples

```
a<-c(1,4,6)
RI(a)

# returns: [1] 0.000000 5.545177 10.750557
```

Rlog*Provides 'repeated logarithm' as accessible function*

Description

Provides 'repeated logarithm' as accessible function

Usage

```
Rlog(z)
```

Arguments

`z` a scalar or vector of numerical values

Details

The function returns $\log(z) * \log(z)$.

Value

$\log(z) * \log(z)$

Author(s)

Georg Heinze

See Also

`coxph`

Examples

```
a<-c(1,4,6)
Rlog(a)

# returns: [1] 0.000000 1.921812 3.210402
```

Rpow2

Provides 'repeated power to the 2' as accessible function

Description

Provides 'repeated power to the 2' as accessible function

Usage

```
Rpow2(z)
```

Arguments

z a scalar or vector of numerical values

Details

The function returns $z^2 * \log(z)$.

Value

$z^2 * \log(z)$

Author(s)

Georg Heinze

See Also

coxph

Examples

```
a<-c(1,4,6)
Rpow2(a)

# returns: [1] 0.00000 22.18071 64.50334
```

Rpow3

Provides 'repeated power to the 3' as accessible function

Description

Provides 'repeated power to the 3' as accessible function

Usage

```
Rpow3(z)
```

Arguments

`z` a scalar or vector of numerical values

Details

The function returns $z^3 * \log(z)$.

Value

$z^3 * \log(z)$

Author(s)

Georg Heinze

See Also

coxph

Examples

```
a<-c(1,4,6)
Rpow3(a)

# returns: [1] 0.00000 88.72284 387.02005
```

`RpowM0.5`*Provides 'repeated power to the minus 0.5' as accessible function*

Description

Provides 'repeated power to the minus 0.5' as accessible function

Usage

```
RpowM0.5(z)
```

Arguments

`z` a scalar or vector of numerical values

Details

The function returns $z^{(-0.5)} * \log(z)$.

Value

$z^{(-0.5)} * \log(z)$

Author(s)

Georg Heinze

See Also

`coxph`

Examples

```
a<-c(1,4,6)
RpowM0.5(a)

# returns: [1] 0.0000000 0.6931472 0.7314827
```

RpowM1

Provides 'repeated power to the minus 1' as accessible function

Description

Provides 'repeated power to the minus 1' as accessible function

Usage

```
RpowM1(z)
```

Arguments

`z` a scalar or vector of numerical values

Details

The function returns $z^{(-1)} * \log(z)$.

Value

$z^{(-1)} * \log(z)$

Author(s)

Georg Heinze

See Also

coxph

Examples

```
a<-c(1,4,6)
RpowM1(a)

# returns: [1] 0.0000000 0.3465736 0.2986266
```

RpowM2

Provides 'repeated power to the minus 2' as accessible function

Description

Provides 'repeated power to the minus 2' as accessible function

Usage

```
RpowM2(z)
```

Arguments

`z` a scalar or vector of numerical values

Details

The function returns $z^{-2} * \log(z)$.

Value

$z^{-2} * \log(z)$

Author(s)

Georg Heinze

See Also

coxph

Examples

```
a<-c(1,4,6)
RpowM2(a)

# returns: [1] 0.0000000 0.0866434 0.0497711
```

Rsqr

Provides 'repeated square root' as accessible function

Description

Provides 'repeated square root' as accessible function

Usage

Rsqr(z)

Arguments

z a scalar or vector of numerical values

Details

The function returns $\sqrt{z} * \log(z)$.

Value

$\sqrt{z} * \log(z)$

Author(s)

Georg Heinze

See Also

coxph

Examples

```
a<-c(1,4,6)
Rsqr(a)
# returns: [1] 0.000000 2.772589 4.388896
```

Index

*Topic **survival**

- coxphw, 2
- fp.power, 8
- plotfp, 9
- plotw, 11
- pow2, 13
- pow3, 14
- powM0.5, 15
- powM1, 16
- powM2, 17
- PT, 18
- RI, 19
- Rlog, 20
- Rpow2, 21
- Rpow3, 22
- RpowM0.5, 23
- RpowM1, 24
- RpowM2, 25
- Rsqrt, 26

coxphw, 2

fp.power, 7

plotfp, 9
plotw, 11
pow2, 13
pow3, 14
powM0.5, 15
powM1, 16
powM2, 17
PT, 18

RI, 19
Rlog, 20
Rpow2, 21
Rpow3, 22
RpowM0.5, 23
RpowM1, 24
RpowM2, 25
Rsqrt, 26