Package 'survival'

January 16, 2026

Title Survival Analysis

Priority recommended

Version 3.8-6

Date 2026-01-09

Depends R (>= 3.5.0)

Imports graphics, Matrix, methods, splines, stats, utils

LazyData Yes

LazyDataCompression xz

ByteCompile Yes

Description Contains the core survival analysis routines, including
definition of Surv objects,
Kaplan-Meier and Aalen-Johansen (multi-state) curves, Cox models,
and parametric accelerated failure time models.

License LGPL (>= 2)

URL https://github.com/therneau/survival

NeedsCompilation yes

Author Terry M Therneau [aut, cre],
Thomas Lumley [ctb, trl] (original S->R port and R maintainer until 2009),
Atkinson Elizabeth [ctb],
Crowson Cynthia [ctb]

Maintainer Terry M Therneau <therneau.terry@mayo.edu>

Repository CRAN

Date/Publication 2026-01-16 09:00:02 UTC

Contents

`aareg` *Aalen's additive regression model for censored data*

Description

Returns an object of class `"aareg"` that represents an Aalen model.

Usage

```
aareg(formula, data, weights, subset, na.action,
   qrtol=1e-07, nmin, dfbeta=FALSE, taper=1,
   test = c('aalen', 'variance', 'nrisk'), cluster,
    model=FALSE, x=FALSE, y=FALSE)
```

Arguments

formula	a formula object, with the response on the left of a '~' operator and the terms, separated by + operators, on the right. The response must be a Surv object. Due to a particular computational approach that is used, the model MUST include an intercept term. If "-1" is used in the model formula the program will ignore it.
data	data frame in which to interpret the variables named in the formula, subset, and weights arguments. This may also be a single number to handle some speci al cases – see below for details. If data is missing, the variables in the model formula should be in the search path.
weights	vector of observation weights. If supplied, the fitting algorithm minimizes the sum of the weights multiplied by the squared residuals (see below for additional technical details). The length of weights must be the same as the number of observations. The weights must be nonnegative and it i s recommended that they be strictly positive, since zero weights are ambiguous. To exclude particular observations from the model, use the subset argument instead of zero weights.
subset	expression specifying which subset of observations should be used in the fit. Th is can be a logical vector (which is replicated to have length equal to the numb er of observations), a numeric vector indicating the observation numbers to be included, or a character vector of the observation names that should be included. All observations are included by default.
na.action	a function to filter missing data. This is applied to the model.fr ame after any subset argument has be en applied. The default is na.fail, which returns a n error if any missing values are found. An alternative is na.excl ude, which deletes observations that contain one or more missing values.
qrtol	tolerance for detection of singularity in the QR decomposition
nmin	minimum number of observations for an estimate; defaults to 3 times the number of covariates. This essentially truncates the computations near the tail of the data set, when n is small and the calculations can become numerically unstable.
dfbeta	should the array of dfbeta residuals be computed. This implies computation of the sandwich variance estimate. The residuals will always be computed if there is a cluster term in the model formula.
taper	allows for a smoothed variance estimate. Var(x), where x is the set of covariates, is an important component of the calculations for the Aalen regression model. At any given time point t, it is computed over all subjects who are still at risk at time t. The tape argument allows smoothing these estimates, for example taper=(1:4)/4 would cause the variance estimate used at any event time to be a weighted average of the estimated variance matrices at the last 4 death times, with a weight of 1 for the current death time and decreasing to 1/4 for prior event times. The default value gives the standard Aalen model.

test	selects the weighting to be used, for computing an overall "average" coefficient vector over time and the subsequent test for equality to zero.
cluster	the clustering group, optional. The variable will be searched for in the data argument.
model, x, y	should copies of the model frame, the x matrix of predictors, or the response vector y be included in the saved result.

Details

The Aalen model assumes that the cumulative hazard H(t) for a subject can be expressed as a(t) + X B(t), where a(t) is a time-dependent intercept term, X is the vector of covariates for the subject (possibly time-dependent), and B(t) is a time-dependent matrix of coefficients. The estimates are inherently non-parametric; a fit of the model will normally be followed by one or more plots of the estimates.

The estimates may become unstable near the tail of a data set, since the increment to B at time t is based on the subjects still at risk at time t. The tolerance and/or nmin parameters may act to truncate the estimate before the last death. The taper argument can also be used to smooth out the tail of the curve. In practice, the addition of a taper such as 1:10 appears to have little effect on death times when n is still reasonably large, but can considerably dampen wild occilations in the tail of the plot.

Value

an object of class "aareg" representing the fit, with the following components:

n	vector containing the number of observations in the data set, the number of event times, and the number of event times used in the computation
times	vector of sorted event times, which may contain duplicates
nrisk	vector containing the number of subjects at risk, of the same length as times
coefficient	matrix of coefficients, with one row per event and one column per covariate
test.statistic	the value of the test statistic, a vector with one element per covariate
test.var	variance-covariance matrix for the test
test	the type of test; a copy of the test argument above
tweight	matrix of weights used in the computation, one row per event
call	a copy of the call that produced this result

References

Aalen, O.O. (1989). A linear regression model for the analysis of life times. Statistics in Medicine, 8:907-925.

Aalen, O.O (1993). Further results on the non-parametric linear model in survival analysis. Statistics in Medicine. 12:1569-1588.

See Also

print.aareg, summary.aareg, plot.aareg

Examples

```
# Fit a model to the lung cancer data set
lfit <- aareg(Surv(time, status) ~ age + sex + ph.ecog, data=lung,
                    nmin=1)
## Not run:
lfit
Call:
aareg(formula = Surv(time, status) ~ age + sex + ph.ecog, data = lung, nmin = 1
        )

  n=227 (1 observations deleted due to missing values)
    138 out of 138 unique event times used

              slope      coef se(coef)     z        p
Intercept  5.26e-03  5.99e-03 4.74e-03  1.26 0.207000
      age  4.26e-05  7.02e-05 7.23e-05  0.97 0.332000
      sex -3.29e-03 -4.02e-03 1.22e-03 -3.30 0.000976
  ph.ecog  3.14e-03  3.80e-03 1.03e-03  3.70 0.000214

Chisq=26.73 on 3 df, p=6.7e-06; test weights=aalen

plot(lfit[4], ylim=c(-4,4))  # Draw a plot of the function for ph.ecog

## End(Not run)
lfit2 <- aareg(Surv(time, status) ~ age + sex + ph.ecog, data=lung,
                  nmin=1, taper=1:10)
## Not run: lines(lfit2[4], col=2)  # Nearly the same, until the last point

# A fit to the mulitple-infection data set of children with
# Chronic Granuomatous Disease.  See section 8.5 of Therneau and Grambsch.
fita2 <- aareg(Surv(tstart, tstop, status) ~ treat + age + inherit +
                        steroids + cluster(id), data=cgd)
## Not run:
  n= 203
    69 out of 70 unique event times used

                     slope      coef se(coef) robust se     z        p
Intercept         0.004670  0.017800 0.002780  0.003910  4.55 5.30e-06
treatrIFN-g      -0.002520 -0.010100 0.002290  0.003020 -3.36 7.87e-04
age              -0.000101 -0.000317 0.000115  0.000117 -2.70 6.84e-03
inheritautosomal  0.001330  0.003830 0.002800  0.002420  1.58 1.14e-01
steroids          0.004620  0.013200 0.010600  0.009700  1.36 1.73e-01

Chisq=16.74 on 4 df, p=0.0022; test weights=aalen

## End(Not run)
```

---

aeqSurv *Adjudicate near ties in a Surv object*

Description

The check for tied survival times can fail due to floating point imprecision, which can make actual ties appear to be distinct values. Routines that depend on correct identification of ties pairs will then give incorrect results, e.g., a Cox model. This function rectifies these.

Usage

```
aeqSurv(x, tolerance = sqrt(.Machine$double.eps))
```

Arguments

x	a Surv object
tolerance	the tolerance used to detect values that will be considered equal

Details

This routine is called by both `survfit` and `coxph` to deal with the issue of ties that get incorrectly broken due to floating point imprecision. See the short vignette on tied times for a simple example. Use the `timefix` argument of `survfit` or `coxph.control` to control the option if desired.

The rule for 'equality' is identical to that used by the `all.equal` routine. Pairs of values that are within round off error of each other are replaced by the smaller value. An error message is generated if this process causes a 0 length time interval to be created.

Value

a Surv object identical to the original, but with ties restored.

Author(s)

Terry Therneau

See Also

`survfit`, `coxph.control`

---

aggregate.survfit *Average survival curves*

Description

For a survfit object containing multiple curves, create average curves over a grouping.

Usage

```
## S3 method for class 'survfit'
aggregate(x, by = NULL, FUN = mean, ...)
```

Arguments

x	a survfit object which has a data dimension.
by	an optional list or vector of grouping elements, each as long as dim(x)['data'].
FUN	a function to compute the summary statistic of interest.
...	optional further arguments to FUN.

Details

The primary use of this is to take an average over multiple survival curves that were created from a modeling function. That is, a marginal estimate of the survival. It is primarily used to average over multiple predicted curves from a Cox model.

Value

a survfit object of lower dimension.

See Also

survfit

Examples

```
cfit <- coxph(Surv(futime, death) ~ sex + age*hgb, data=mgus2)
# marginal effect of sex, after adjusting for the others
dummy <- rbind(mgus2, mgus2)
dummy$sex <- rep(c("F", "M"), each=nrow(mgus2)) # population data set
dummy <- na.omit(dummy)   # don't count missing hgb in our "population
csurv <- survfit(cfit, newdata=dummy)
dim(csurv)  # 2 * 1384 survival curves
csurv2 <- aggregate(csurv, dummy$sex)
```

---

agreg.fit *Cox model fitting functions*

Description

These are the the functions called by coxph that do the actual computation. In certain situations, e.g. a simulation, it may be advantageous to call these directly rather than the usual coxph call using a model formula.

Usage

```
agreg.fit(x, y, strata, offset, init, control, weights, method,
rownames, resid=TRUE, nocenter=NULL)
coxph.fit(x, y, strata, offset, init, control, weights, method,
rownames, resid=TRUE, nocenter=NULL)
```

Arguments

x	Matix of predictors. This should *not* include an intercept.
y	a Surv object containing either 2 columns (coxph.fit) or 3 columns (agreg.fit).
strata	a vector containing the stratification, or NULL
offset	optional offset vector
init	initial values for the coefficients
control	the result of a call to coxph.control
weights	optional vector of weights
method	method for handling ties, one of "breslow" or "efron"
rownames	this is only needed for a NULL model, in which case it contains the rownames (if any) of the original data.
resid	compute and return residuals.
nocenter	an optional list of values. Any column of the X matrix whose values lie strictly within that set will not be recentered. Note that the coxph function has (-1, 0, 1) as the default.

Details

This routine does no checking that arguments are the proper length or type. Only use it if you know what you are doing!

The resid and concordance arguments will save some compute time for calling routines that only need the likelihood, the generation of a permutation distribution for instance.

Value

a list containing results of the fit

Author(s)

Terry Therneau

See Also

coxph

aml *Acute Myelogenous Leukemia survival data*

Description

Survival in patients with Acute Myelogenous Leukemia. The question at the time was whether the standard course of chemotherapy should be extended ('maintainance') for additional cycles.

Usage

```
aml
leukemia
data(cancer, package="survival")
```

Format

time:	survival or censoring time
status:	censoring status
x:	maintenance chemotherapy given? (factor)

Source

Rupert G. Miller (1997), *Survival Analysis*. John Wiley & Sons. ISBN: 0-471-25218-2.

anova.coxph *Analysis of Deviance for a Cox model.*

Description

Compute an analysis of deviance table for one or more Cox model fits, based on the log partial likelihood.

Usage

```
## S3 method for class 'coxph'
anova(object, ...,  test = 'Chisq')
```

Arguments

object	An object of class coxph
...	Further coxph objects
test	a character string. The appropriate test is a chisquare, all other choices result in no test being done.

Details

Specifying a single object gives a sequential analysis of deviance table for that fit. That is, the reductions in the model Cox log-partial-likelihood as each term of the formula is added in turn are given in as the rows of a table, plus the log-likelihoods themselves. A robust variance estimate is normally used in situations where the model may be mis-specified, e.g., multiple events per subject. In this case a comparison of likelihood values does not make sense (differences no longer have a chi-square distribution), and anova will refuse to print results.

If more than one object is specified, the table has a row for the degrees of freedom and loglikelihood for each model. For all but the first model, the change in degrees of freedom and loglik is also given. (This only make statistical sense if the models are nested.) It is conventional to list the models from smallest to largest, but this is up to the user.

The table will optionally contain test statistics (and P values) comparing the reduction in loglik for each row.

Value

An object of class "anova" inheriting from class "data.frame".

Warning

The comparison between two or more models by anova will only be valid if they are fitted to the same dataset. This may be a problem if there are missing values.

See Also

coxph, anova.

Examples

```
fit <- coxph(Surv(futime, fustat) ~ resid.ds *rx + ecog.ps, data = ovarian)
anova(fit)
fit2 <- coxph(Surv(futime, fustat) ~ resid.ds +rx + ecog.ps, data=ovarian)
anova(fit2,fit)
```

---

attrassign *Create new-style "assign" attribute*

Description

The "assign" attribute on model matrices describes which columns come from which terms in the model formula. It has two versions. R uses the original version, but the alternate version found in S-plus is sometimes useful.

Usage

```
attrassign(object, ...)
## Default S3 method:
attrassign(object, tt,...)
## S3 method for class 'lm'
attrassign(object,...)
```

Arguments

object	model matrix or linear model object
tt	terms object
...	further arguments for other methods

Details

For instance consider the following

```
survreg(Surv(time, status) ~ age + sex + factor(ph.ecog), lung)
```

R gives the compact for for assign, a vector (0, 1, 2, 3, 3, 3); which can be read as "the first column of the X matrix (intercept) goes with none of the terms, the second column of X goes with term 1 of the model equation, the third column of X with term 2, and columns 4-6 with term 3".

The alternate (S-Plus default) form is a list

```
$(Intercept)     1
$age             2
$sex             3
$factor(ph.ecog) 4 5 6
```

Value

A list with names corresponding to the term names and elements that are vectors indicating which columns come from which terms

See Also

terms,model.matrix

Examples

```
formula <- Surv(time,status)~factor(ph.ecog)
tt <- terms(formula)
mf <- model.frame(tt,data=lung)
mm <- model.matrix(tt,mf)
## a few rows of data
mm[1:3,]
## old-style assign attribute
```

```
attr(mm,"assign")
## alternate style assign attribute
attrassign(mm,tt)
```

---

basehaz *Alias for the survfit function*

Description

Compute the predicted survival curve for a Cox model.

Usage

```
basehaz(fit, newdata, centered=TRUE)
```

Arguments

fit	a coxph fit
newdata	a data frame containing one row for each predicted survival curve, said row contains the covariate values for that curve
centered	ignored if the newdata argument is present. Otherwise, if TRUE return data from a predicted survival curve for the covariate values fit$mean, if FALSE return a prediction for all covariates equal to zero.

Details

This function is an alias for survfit.coxph, which does the actual work and has a richer set of options. Look at that help file for more discussion and explanation. This alias exists primarily because some users look for predicted survival estimates under this name.

The function returns a data frame containing the time, cumhaz and optionally the strata (if the fitted Cox model used a strata statement), which are copied from the survfit result.

If H(t; z) is the predicted cumulative hazard for an observation with covariate vector z, then H(t;x) = H(t;z) r(x,z) where r(x,z)= exp(beta[1](x[1]- z[1]) + beta[2](x[2]-z[2]) + ...) = exp(sum(coef(fit) * (x-z))) is the Cox model's hazard ratio for covariate vector x vs covariate vector z. That is, the cumulative hazard H for a single reference value z is sufficient to provide the hazard for any covariate values. The predicted survival curve is S(t; x)= exp(-H(t;x)). There is not a simple transformation for the variance of H, however.

Many textbooks refer to H(t; 0) as "the" baseline hazard for a Cox model; this is returned by the centered= FALSE option. However, due to potential overflow or underflow in the exp() function this can be a very bad idea in practice. The authors do not recommend this option, but for users who insist: caveat emptor. Offset terms can pose a particular challenge for the underlying code and are always recentered; to override this use the newdata argument and include the offset as one of the variables.

Value

a data frame with variable names of `hazard`, `time` and optionally `strata`. The first is actually the cumulative hazard.

See Also

`survfit.coxph`

`bladder` *Bladder Cancer Recurrences*

Description

Data on recurrences of bladder cancer, used by many people to demonstrate methodology for recurrent event modelling.

Bladder1 is the full data set from the study. It contains all three treatment arms and all recurrences for 118 subjects; the maximum observed number of recurrences is 9.

Bladder is the data set that appears most commonly in the literature. It uses only the 85 subjects with nonzero follow-up who were assigned to either thiotepa or placebo, and only the first four recurrences for any patient. The status variable is 1 for recurrence and 0 for everything else (including death for any reason). The data set is laid out in the competing risks format of the paper by Wei, Lin, and Weissfeld.

Bladder2 uses the same subset of subjects as bladder, but formatted in the (start, stop] or Anderson-Gill style. Note that in transforming from the WLW to the AG style data set there is a quite common programming mistake that leads to extra follow-up time for 12 subjects: all those with follow-up beyond their 4th recurrence. This "follow-up" is a side effect of throwing away all events after the fourth while retaining the last follow-up time variable from the original data. The bladder2 data set found here does not make this mistake, but some analyses in the literature have done so; it results in the addition of a small amount of immortal time bias and shrinks the fitted coefficients towards zero.

Usage

```
bladder1
bladder
bladder2
data(cancer, package="survival")
```

Format

bladder1

id:	Patient id
treatment:	Placebo, pyridoxine (vitamin B6), or thiotepa
number:	Initial number of tumours (8=8 or more)
size:	Size (cm) of largest initial tumour

recur:	Number of recurrences
start,stop:	The start and end time of each time interval
status:	End of interval code, 0=censored, 1=recurrence, 2=death from bladder disease, 3=death other/unknown cause
rtumor:	Number of tumors found at the time of a recurrence
rsize:	Size of largest tumor at a recurrence
enum:	Event number (observation number within patient)

bladder

id:	Patient id
rx:	Treatment 1=placebo 2=thiotepa
number:	Initial number of tumours (8=8 or more)
size:	size (cm) of largest initial tumour
stop:	recurrence or censoring time
enum:	which recurrence (up to 4)

bladder2

id:	Patient id
rx:	Treatment 1=placebo 2=thiotepa
number:	Initial number of tumours (8=8 or more)
size:	size (cm) of largest initial tumour
start:	start of interval (0 or previous recurrence time)
stop:	recurrence or censoring time
enum:	which recurrence (up to 4)

Source

Andrews DF, Hertzberg AM (1985), DATA: A Collection of Problems from Many Fields for the Student and Research Worker, New York: Springer-Verlag.

LJ Wei, DY Lin, L Weissfeld (1989), Regression analysis of multivariate incomplete failure time data by modeling marginal distributions. *Journal of the American Statistical Association*, **84**.

---

blogit *Bounded link functions*

Description

Alternate link functions that impose bounds on the input of their link function

Usage

```
blogit(edge = 0.05)
bprobit(edge= 0.05)
bcloglog(edge=.05)
blog(edge=.05)
```

Arguments

edge	input values less than the cutpoint are replaces with the cutpoint. For all be blog input values greater than (1-edge) are replaced with (1-edge)

Details

When using survival psuedovalues for binomial regression, the raw data can be outside the range (0,1), yet we want to restrict the predicted values to lie within that range. A natural way to deal with this is to use glm with family = gaussian(link= "logit"). But this will fail. The reason is that the family object has a component linkfun that does not accept values outside of (0,1).

This function is only used to create initial values for the iteration step, however. Mapping the offending input argument into the range of (egde, 1-edge) before computing the link results in starting estimates that are good enough. The final result of the fit will be no different than if explicit starting estimates were given using the etastart or mustart arguments. These functions create copies of the logit, probit, and complimentary log-log families that differ from the standard ones only in this use of a bounded input argument, and are called a "bounded logit" = blogit, etc.

The same argument hold when using RMST (area under the curve) pseudovalues along with a log link to ensure positive predictions, though in this case only the lower boundary needs to be mapped.

Value

a family object of the same form as make.family.

See Also

stats{make.family}

Examples

```
py <- pseudo(survfit(Surv(time, status) ~1, lung), time=730) #2 year survival
range(py)
pfit <- glm(py ~ ph.ecog, data=lung, family=gaussian(link=blogit()))
# For each +1 change in performance score, the odds of 2 year survival
#  are multiplied by 1/2  = exp of the coefficient.
```

brier *Compute the Brier score for a Cox model*

Description

Compute the Brier score, for a coxph model

Usage

```
brier(fit, times, newdata, ties = TRUE, detail = FALSE, timefix = TRUE,
      efron = FALSE)
```

Arguments

fit	result of a coxph fit
times	time points at which to create the score
newdata	optional, used to validate a prior fit with new data
ties	if TRUE, treate tied event/censoring times properly
detail	if TRUE, the returned object has more detail. This can be useful for debugging or for instruction.
timefix	deal with near ties in the data. See the tied times vignette.
efron	use the same survival estimate for the NULL model as was used in the coxph call

Details

Far more details are found in the vignette. At any time point tau, the scaled Brier score is essentially the R-squared statistic where y = the 0/1 variable "event at or before tau", yhat is the probability of an event by tau, as predicted by the model, and the ybar is the predicted probablity without covariate, normally from a Kaplan-Meier. If $R^2 = 1 - \sum(y - \hat{y})^2 / \sum(y - \mu)^2$, the Brier score is formally only the numerator of the second term. The rescaled value is much more useful, however.

Many, perhaps even most algorithms do not properly deal with a tied censoring time/event time pair. The tied option is present mostly verify that we get the same answer, when we make the same mistake. The numerical size of the inaccuracy is very small; just large enough to generate concern that this function is incorrect.

A sensible argument can be made that the NULL model should be a coxph call with no covariates, rather than the Kaplan-Meier; but it turns out that the effect is very slight. This is allowed by the efron argument.

Value

a list with components

rsquared	the R^2 value, a scaled Brier score. This will be a vector with one entry for each time point.
brier	the brier score, a vector with one entry per time point
times	the time points at which the score was computed

Author(s)

Terry Therneau

See Also

rttright

Examples

```
cfit <- coxph(Surv(rtime, recur) ~ age + meno + size + pmin(nodes,11),
              data= rotterdam)
round(cfit$concordance["concordance"], 3)  # some predictive power
brier(cfit, times=c(4,6)*365.25)   # values at 4 and 6 years
```

---

cch *Fits proportional hazards regression model to case-cohort data*

Description

Returns estimates and standard errors from relative risk regression fit to data from case-cohort studies. A choice is available among the Prentice, Self-Prentice and Lin-Ying methods for unstratified data. For stratified data the choice is between Borgan I, a generalization of the Self-Prentice estimator for unstratified case-cohort data, and Borgan II, a generalization of the Lin-Ying estimator.

Usage

```
cch(formula, data, subcoh, id, stratum=NULL, cohort.size,
    method =c("Prentice","SelfPrentice","LinYing","I.Borgan","II.Borgan"),
    robust=FALSE)
```

Arguments

formula	A formula object that must have a Surv object as the response. The Surv object must be of type "right", or of type "counting".
subcoh	Vector of indicators for subjects sampled as part of the sub-cohort. Code 1 or TRUE for members of the sub-cohort, 0 or FALSE for others. If data is a data frame then subcoh may be a one-sided formula.
id	Vector of unique identifiers, or formula specifying such a vector.
stratum	A vector of stratum indicators or a formula specifying such a vector
cohort.size	Vector with size of each stratum original cohort from which subcohort was sampled
data	An optional data frame in which to interpret the variables occurring in the formula.
method	Three procedures are available. The default method is "Prentice", with options for "SelfPrentice" or "LinYing".
robust	For "LinYing" only, if robust=TRUE, use design-based standard errors even for phase I

Details

Implements methods for case-cohort data analysis described by Therneau and Li (1999). The three methods differ in the choice of "risk sets" used to compare the covariate values of the failure with those of others at risk at the time of failure. "Prentice" uses the sub-cohort members "at risk" plus the failure if that occurs outside the sub-cohort and is score unbiased. "SelfPren" (Self-Prentice) uses just the sub-cohort members "at risk". These two have the same asymptotic variance-covariance matrix. "LinYing" (Lin-Ying) uses the all members of the sub-cohort and all failures outside the sub-cohort who are "at risk". The methods also differ in the weights given to different score contributions.

The `data` argument must not have missing values for any variables in the model. There must not be any censored observations outside the subcohort.

Value

An object of class "cch" incorporating a list of estimated regression coefficients and two estimates of their asymptotic variance-covariance matrix.

`coef`	regression coefficients.
`naive.var`	Self-Prentice model based variance-covariance matrix.
`var`	Lin-Ying empirical variance-covariance matrix.

Author(s)

Norman Breslow, modified by Thomas Lumley

References

Prentice, RL (1986). A case-cohort design for epidemiologic cohort studies and disease prevention trials. Biometrika 73: 1–11.

Self, S and Prentice, RL (1988). Asymptotic distribution theory and efficiency results for case-cohort studies. Annals of Statistics 16: 64–81.

Lin, DY and Ying, Z (1993). Cox regression with incomplete covariate measurements. Journal of the American Statistical Association 88: 1341–1349.

Barlow, WE (1994). Robust variance estimation for the case-cohort design. Biometrics 50: 1064–1072

Therneau, TM and Li, H (1999). Computing the Cox model for case-cohort designs. Lifetime Data Analysis 5: 99–112.

Borgan, O, Langholz, B, Samuelsen, SO, Goldstein, L and Pogoda, J (2000) Exposure stratified case-cohort designs. Lifetime Data Analysis 6, 39-58.

See Also

`twophase` and `svycoxph` in the "survey" package for more general two-phase designs. http://faculty.washington.edu/tlumley/survey/

Examples

```
## The complete Wilms Tumor Data
## (Breslow and Chatterjee, Applied Statistics, 1999)
## subcohort selected by simple random sampling.
##

subcoh <- nwtco$in.subcohort
selccoh <- with(nwtco, rel==1|subcoh==1)
ccoh.data <- nwtco[selccoh,]
ccoh.data$subcohort <- subcoh[selccoh]
## central-lab histology
ccoh.data$histol <- factor(ccoh.data$histol,labels=c("FH","UH"))
## tumour stage
ccoh.data$stage <- factor(ccoh.data$stage,labels=c("I","II","III","IV"))
ccoh.data$age <- ccoh.data$age/12 # Age in years

##
## Standard case-cohort analysis: simple random subcohort
##

fit.ccP <- cch(Surv(edrel, rel) ~ stage + histol + age, data =ccoh.data,
   subcoh = ~subcohort, id=~seqno, cohort.size=4028)


fit.ccP

fit.ccSP <- cch(Surv(edrel, rel) ~ stage + histol + age, data =ccoh.data,
   subcoh = ~subcohort, id=~seqno, cohort.size=4028, method="SelfPren")

summary(fit.ccSP)

##
## (post-)stratified on instit
##
stratsizes<-table(nwtco$instit)
fit.BI<- cch(Surv(edrel, rel) ~ stage + histol + age, data =ccoh.data,
   subcoh = ~subcohort, id=~seqno, stratum=~instit, cohort.size=stratsizes,
   method="I.Borgan")

summary(fit.BI)
```

---

cgd *Chronic Granulotamous Disease data*

Description

Data are from a placebo controlled trial of gamma interferon in chronic granulotomous disease (CGD). Contains the data on time to serious infections observed through end of study for each patient.

Usage

```
cgd
data(cgd)
```

Format

id subject identification number

center enrolling center

random date of randomization

treatment placebo or gamma interferon

sex sex

age age in years, at study entry

height height in cm at study entry

weight weight in kg at study entry

inherit pattern of inheritance

steroids use of steroids at study entry,1=yes

propylac use of prophylactic antibiotics at study entry

hos.cat a categorization of the centers into 4 groups

tstart, tstop start and end of each time interval

status 1=the interval ends with an infection

enum observation number within subject

Details

The `cgd0` data set is in the form found in the references, with one line per patient and no recoding of the variables. The `cgd` data set (this one) has been cast into (start, stop] format with one line per event, and covariates such as center recoded as factors to include meaningful labels.

Source

Fleming and Harrington, Counting Processes and Survival Analysis, appendix D.2.

See Also

`link{cgd0}`

cgd0 *Chronic Granulotomous Disease data*

Description

Data are from a placebo controlled trial of gamma interferon in chronic granulotomous disease (CGD). Contains the data on time to serious infections observed through end of study for each patient.

Usage

```
cgd0
```

Format

id subject identification number

center enrolling center

random date of randomization

treatment placebo or gamma interferon

sex sex

age age in years, at study entry

height height in cm at study entry

weight weight in kg at study entry

inherit pattern of inheritance

steroids use of steroids at study entry,1=yes

propylac use of prophylactic antibiotics at study entry

hos.cat a categorization of the centers into 4 groups

futime days to last follow-up

etime1-etime7 up to 7 infection times for the subject

Details

The `cgdraw` data set (this one) is in the form found in the references, with one line per patient and no recoding of the variables.

The `cgd` data set has been further processed so as to have one line per event, with covariates such as center recoded as factors to include meaningful labels.

Source

Fleming and Harrington, Counting Processes and Survival Analysis, appendix D.2.

See Also

`cgd`

cipoisson *Confidence limits for the Poisson*

Description

Confidence interval calculation for Poisson rates.

Usage

```
cipoisson(k, time = 1, p = 0.95, method = c("exact", "anscombe"))
```

Arguments

k	Number of successes
time	Total time on trial
p	Probability level for the (two-sided) interval
method	The method for computing the interval.

Details

The likelihood method is based on equation 10.10 of Feller, which relates poisson probabilities to tail area of the gamma distribution. The Anscombe approximation is based on the fact that sqrt(k + 3/8) has a nearly constant variance of 1/4, along with a continuity correction.

There are many other proposed intervals: Patil and Kulkarni list and evaluate 19 different suggestions from the literature!. The exact intervals can be overly broad for very small values of k, many of the other approaches try to shrink the lengths, with varying success.

Value

a vector, matrix, or array. If both k and time are single values the result is a vector of length 2 containing the lower an upper limits. If either or both are vectors the result is a matrix with two columns. If k is a matrix or array, the result will be an array with one more dimension; in this case the dimensions and dimnames (if any) of k are preserved.

References

F.J. Anscombe (1949). Transformations of Poisson, binomial and negative-binomial data. Biometrika, 35:246-254.

W.F. Feller (1950). An Introduction to Probability Theory and its Applications, Volume 1, Chapter 6, Wiley.

V. V. Patil and H.F. Kulkarni (2012). Comparison of confidence intervals for the poisson mean: some new aspects. Revstat 10:211-227.

See Also

ppois, qpois

Examples

```
cipoisson(4) # 95\% confidence limit
# lower    upper
# 1.089865 10.24153
ppois(4, 10.24153)     #chance of seeing 4 or fewer events with large rate
# [1] 0.02500096
1-ppois(3, 1.08986)    #chance of seeing 4 or more, with a small rate
# [1] 0.02499961
```

`clogit` *Conditional logistic regression*

Description

Estimates a logistic regression model by maximising the conditional likelihood. Uses a model formula of the form `case.status~exposure+strata(matched.set)`. The default is to use the exact conditional likelihood, a commonly used approximate conditional likelihood is provided for compatibility with older software.

Usage

```
clogit(formula, data, weights, subset, na.action,
 method=c("exact", "approximate", "efron", "breslow"),
 ...)
```

Arguments

`formula`	Model formula
`data`	data frame
`weights`	optional, names the variable containing case weights
`subset`	optional, subset the data
`na.action`	optional na.action argument. By default the global option `na.action` is used.
`method`	use the correct (exact) calculation in the conditional likelihood or one of the approximations
`...`	optional arguments, which will be passed to `coxph.control`

Details

It turns out that the loglikelihood for a conditional logistic regression model = loglik from a Cox model with a particular data structure. Proving this is a nice homework exercise for a PhD statistics class; not too hard, but the fact that it is true is surprising.

When a well tested Cox model routine is available many packages use this 'trick' rather than writing a new software routine from scratch, and this is what the clogit routine does. In detail, a stratified Cox model with each case/control group assigned to its own stratum, time set to a constant, status

of 1=case 0=control, and using the exact partial likelihood has the same likelihood formula as a conditional logistic regression. The clogit routine creates the necessary dummy variable of times (all 1) and the strata, then calls coxph.

The computation of the exact partial likelihood can be very slow, however. If a particular strata had say 10 events out of 20 subjects we have to add up a denominator that involves all possible ways of choosing 10 out of 20, which is 20!/(10! 10!) = 184756 terms. Gail et al describe a fast recursion method which partly ameliorates this; it was incorporated into version 2.36-11 of the survival package. The computation remains infeasible for very large groups of ties, say 100 ties out of 500 subjects, and may even lead to integer overflow for the subscripts – in this latter case the routine will refuse to undertake the task. The Efron approximation is normally a sufficiently accurate substitute.

Most of the time conditional logistic modeling is applied data with 1 case + k controls per set, in which case all of the approximations for ties lead to exactly the same result. The 'approximate' option maps to the Breslow approximation for the Cox model, for historical reasons.

Case weights are not allowed when the exact likelihood (method) is used, as the likelihood is not defined for fractional weights. Even with integer case weights it is not clear how they should be handled. For instance if there are two deaths in a strata, one with weight=1 and one with weight=2, should the likelihood calculation consider all subsets of size 2 or all subsets of size 3? Consequently, case weights are ignored by the routine in this case. For methods other than `exact`, case weights work as they do in an ordinary coxph model.

Value

An object of class `"clogit"`, which is a wrapper for a `"coxph"` object.

References

Michell H Gail, Jay H Lubin and Lawrence V Rubinstein. Likelihood calculations for matched case-control studies and survival studies with tied death times. Biometrika 68:703-707, 1980.

John A. Logan. A multivariate model for mobility tables. Am J Sociology 89:324-349, 1983.

Author(s)

Thomas Lumley

See Also

strata,coxph,glm

Examples

```
## Not run: clogit(case ~ spontaneous + induced + strata(stratum), data=infert)

# A multinomial response recoded to use clogit
#  The revised data set has one copy per possible outcome level, with new
#  variable tocc = target occupation for this copy, and case = whether
#  that is the actual outcome for each subject.
# See the reference below for the data.
resp <- levels(logan$occupation)
```

```
n <- nrow(logan)
indx <- rep(1:n, length(resp))
logan2 <- data.frame(logan[indx,],
                     id = indx,
                     tocc = factor(rep(resp, each=n)))
logan2$case <- (logan2$occupation == logan2$tocc)
clogit(case ~ tocc + tocc:education + strata(id), logan2)
```

`cluster` *Identify clusters.*

Description

This is a special function used in the context of survival models. It identifies correlated groups of observations, and is used on the right hand side of a formula. This style is now discouraged, use the `cluster` option instead.

Usage

```
cluster(x)
```

Arguments

`x`	A character, factor, or numeric variable.

Details

The function's only action is semantic, to mark a variable as the cluster indicator. The resulting variance is what is known as the "working independence" variance in a GEE model. Note that one cannot use both a frailty term and a cluster term in the same model, the first is a mixed-effects approach to correlation and the second a GEE approach, and these don't mix.

Value

`x`

See Also

`coxph`, `survreg`

Examples

```
marginal.model <- coxph(Surv(time, status) ~ rx, data= rats, cluster=litter,
                        subset=(sex=='f'))
frailty.model  <- coxph(Surv(time, status) ~ rx + frailty(litter), rats,
                        subset=(sex=='f'))
```

colon *Chemotherapy for Stage B/C colon cancer*

Description

These are data from one of the first successful trials of adjuvant chemotherapy for colon cancer. Levamisole is a low-toxicity compound previously used to treat worm infestations in animals; 5-FU is a moderately toxic (as these things go) chemotherapy agent. There are two records per person, one for recurrence and one for death

Usage

```
colon
       data(cancer, package="survival")
```

Format

id:	id
study:	1 for all patients
rx:	Treatment - Obs(ervation), Lev(amisole), Lev(amisole)+5-FU
sex:	1=male
age:	in years
obstruct:	obstruction of colon by tumour
perfor:	perforation of colon
adhere:	adherence to nearby organs
nodes:	number of lymph nodes with detectable cancer
time:	days until event or censoring
status:	censoring status
differ:	differentiation of tumour (1=well, 2=moderate, 3=poor)
extent:	Extent of local spread (1=submucosa, 2=muscle, 3=serosa, 4=contiguous structures)
surg:	time from surgery to registration (0=short, 1=long)
node4:	more than 4 positive lymph nodes
etype:	event type: 1=recurrence,2=death

Note

The study is originally described in Laurie (1989). The main report is found in Moertel (1990). This data set is closest to that of the final report in Moertel (1991). A version of the data with less follow-up time was used in the paper by Lin (1994).

Peter Higgins has pointed out a data inconsistency, revealed by `table(colon$nodes, colon$node4)`. We don't know which of the two variables is actually correct so have elected not to 'fix' it. (Real data has warts, why not have some in the example data too?)

References

JA Laurie, CG Moertel, TR Fleming, HS Wieand, JE Leigh, J Rubin, GW McCormack, JB Gerstner, JE Krook and J Malliard. Surgical adjuvant therapy of large-bowel carcinoma: An evaluation of levamisole and the combination of levamisole and fluorouracil: The North Central Cancer Treatment Group and the Mayo Clinic. J Clinical Oncology, 7:1447-1456, 1989.

DY Lin. Cox regression analysis of multivariate failure time data: the marginal approach. Statistics in Medicine, 13:2233-2247, 1994.

CG Moertel, TR Fleming, JS MacDonald, DG Haller, JA Laurie, PJ Goodman, JS Ungerleider, WA Emerson, DC Tormey, JH Glick, MH Veeder and JA Maillard. Levamisole and fluorouracil for adjuvant therapy of resected colon carcinoma. New England J of Medicine, 332:352-358, 1990.

CG Moertel, TR Fleming, JS MacDonald, DG Haller, JA Laurie, CM Tangen, JS Ungerleider, WA Emerson, DC Tormey, JH Glick, MH Veeder and JA Maillard, Fluorouracil plus Levamisole as an effective adjuvant therapy after resection of stage II colon carcinoma: a final report. Annals of Internal Med, 122:321-326, 1991.

`concordance` *Compute the concordance statistic for data or a model*

Description

The concordance statistic computes the agreement between an observed response and a predictor. It is closely related to Kendall's tau-a and tau-b, Goodman's gamma, and Somers' d, all of which can also be calculated from the results of this function.

Usage

```
concordance(object, ...)
## S3 method for class 'formula'
concordance(object, data, weights, subset, na.action,
  cluster, ymin, ymax, timewt= c("n", "S", "S/G", "n/G2", "I"),
  influence=0, ranks = FALSE, reverse=FALSE, timefix=TRUE, keepstrata=10, ...)
## S3 method for class 'lm'
concordance(object, ..., newdata, cluster, ymin, ymax,
  influence=0, ranks=FALSE, timefix=TRUE, keepstrata=10)
## S3 method for class 'coxph'
concordance(object, ..., newdata, cluster, ymin, ymax,
  timewt= c("n", "S", "S/G", "n/G2", "I"), influence=0,
  ranks=FALSE, timefix=TRUE, keepstrata=10)
## S3 method for class 'survreg'
concordance(object, ..., newdata, cluster, ymin, ymax,
  timewt= c("n", "S", "S/G", "n/G2", "I"), influence=0,
  ranks=FALSE, timefix=TRUE, keepstrata=10)
```

Arguments

object	a fitted model or a formula. The formula should be of the form y ~x or y ~ x + strata(z) with a single numeric or survival response and a single predictor. Counts of concordant, discordant and tied pairs are computed separately per stratum, and then added.
data	a data.frame in which to interpret the variables named in the formula, or in the subset and the weights argument. Only applicable if object is a formula.
weights	optional vector of case weights. Only applicable if object is a formula.
subset	expression indicating which subset of the rows of data should be used in the fit. Only applicable if object is a formula.
na.action	a missing-data filter function. This is applied to the model.frame after any subset argument has been used. Default is options()\$na.action. Only applicable if object is a formula.
...	multiple fitted models are allowed. Only applicable if object is a model object.
newdata	optional, a new data frame in which to evaluate (but not refit) the models
cluster	optional grouping vector for calculating the robust variance
ymin, ymax	compute the concordance over the restricted range ymin <= y <= ymax. (For survival data this is a time range.)
timewt	the weighting to be applied. The overall statistic is a weighted mean over event times.
influence	1= return the dfbeta vector, 2= return the full influence matrix, 3 = return both
ranks	if TRUE, return a data frame containing the scaled ranks that make up the overall score.
reverse	by default (FALSE) larger predictions x are associated with larger response values y; this is the expected behavior for a linear model, logistic regression, parametric survival, random forest, ... The exception to this rule is a Cox model, where a larger risk score x corresponds to *shorter* survival time, if x is the prediction from a coxph fit use reverse = TRUE. If concordance is called directly with the result of a coxph fit this is not necessary, as in that case the function knows this is a coxph result and will choose the correct default.
timefix	correct for possible rounding error. See the vignette on tied times for more explanation. Essentially, exact ties are an important part of the concordance computatation, but "exact" can be a subtle issue with floating point numbers.
keepstrata	either TRUE, FALSE, or an integer value. Computations are always done within stratum, then added. If the total number of strata greater than keepstrata, or keepstrata=FALSE, those subtotals are not kept in the output.

Details

The concordance is an estimate of $Pr(x_i < x_j | y_i < y_j)$, for a model fit replace x with $\hat{y}$, the predicted response from the model. For a survival outcome some pairs of values are not comparable, e.g., censored at time 5 and a death at time 6, as we do not know if the first observation will or will not outlive the second. In this case the total number of evaluable pairs is smaller.

Relatations to other statistics: For continuous x and y, 2C- 1 is equal to Somers' d. If the response is binary, C is equal to the area under the receiver operating curve or AUROC. For a survival response and binary predictor C is the numerator of the Gehan-Wilcoxon test.

A naive compuation requires adding up over all n(n-1)/2 comparisons, which can be quite slow for large data sets. This routine uses an O(n log(n)) algorithm. At each uncensored event time y, compute the rank of x for the subject who had the event as compared to the x values for all others with a longer survival, where the rank has value between 0 and 1. The concordance is a weighted mean of these ranks, determined by the `timewt` option. The rank vector can be efficiently updated as subjects are added or removed from the risk set. For further details see the vignette.

The variance is based on an infinetesimal jackknife. One advantage of this approach is that it also gives a valid covariance for the covariance based on multiple different predicted values, even if those predictions come from quite different models. See for instance the example below which has a poisson and two non-nested Cox models. This has been useful to compare a machine learning model to a Cox model fit, say. It is absolutely critical, however, that the predicted values line up exactly, with the same observation in each row; otherwise the result will be nonsense. (Be alert to the impact of missing values.) The IJ variance used here is very close but not precisely identical to the U-statistics approach of DeLong, in our limited experience they have differed by .1 percent or less. Thus a comparison of two binomial models is extremely close to DeLong's test.

The `timewt` option is only applicable to censored data. In this case the default of n(t) corresponds to Harrell's C statistic, which is closely related to the Gehan-Wilcoxon test; `timewt="S"` corrsponds to the Peto-Wilcoxon, `timewt="S/G"` is suggested by Schemper, and `timewt="n/G2"` corresponds to Uno's C. It turns out that the Schemper and Uno weights are computationally identical, we have retained both options as a user convenience. The `timewt= "I"` option is related to the log-rank statistic.

When the number of strata is very large, such as in a conditional logistic regression for instance (`clogit` function), a much faster computation is available when the individual strata results are not retained; use `keepstrata=FALSE` or `keepstrata=0` to do so. In the general case the `keepstrata = 10` default simply keeps the printout managable: it retains and prints per-strata counts if the number of strata is <= 10.

Value

An object of class `concordance` containing the following components:

`concordance`	the estimated concordance value or values
`count`	a vector containing the number of concordant pairs, discordant, tied on x but not y, tied on y but not x, and tied on both x and y
`n`	the number of observations
`var`	a vector containing the estimated variance of the concordance based on the infinitesimal jackknife (IJ) method. If there are multiple models it contains the estimtated variance/covariance matrix.
`cvar`	a vector containing the estimated variance(s) of the concordance values, based on the variance formula for the associated score test from a proportional hazards model. (This was the primary variance used in the `survConcordance` function.)
`dfbeta`	optional, the vector of leverage estimates for the concordance

influence	optional, the matrix of leverage values for each of the counts, one row per observation
ranks	optional, a data frame containing the Somers' d rank at each event time, along with the time weight, and the case weight of the observation. The time weighted sum of the ranks will equal concordant pairs - discordant pairs.

Note

A coxph model that has a numeric failure may have undefined predicted values, in which case the concordance will be NULL.

Computation for an existing coxph model along with newdata has some subtleties with respect to extra arguments in the original call. These include

- tt() terms in the model. This is not supported with newdata.
- subset. Any subset clause in the original call is ignored, i.e., not applied to the new data.
- strata() terms in the model. The new data is expected to have the strata variable(s) found in the original data set, with concordance computed within strata. The levels of the strata variable need not be the same as in the original data.
- id or cluster directives. This has not yet been sorted out.

Author(s)

Terry Therneau

References

E DeLong, D DeLong, and D Clarke-Pearson, Comparing the areas under two or more correlated reciever operating curves: a nonparametric approach, Biometics, 1988.

F Harrell, R Califf, D Pryor, K Lee and R Rosati, Evaluating the yield of medical tests, J Am Medical Assoc, 1982.

R Peto and J Peto, Asymptotically efficient rank invariant test procedures (with discussion), J Royal Stat Soc A, 1972.

M Schemper, Cox analysis of survival data with non-proportional hazard functions, The Statistician, 1992.

H Uno, T Cai, M Pencina, R D'Agnostino and Lj Wei, On the C-statistics for evaluating overall adequacy of risk prediction procedures with censored survival data, Statistics in Medicine, 2011.

See Also

coxph

Examples

```
fit1 <- coxph(Surv(ptime, pstat) ~ age + sex + mspike, mgus2)
concordance(fit1, timewt="n/G2")  # Uno's weighting

# logistic regression
```

```
tdata <- flchain
options(na.action= na.exclude) # fit2a has many more missings, this causes
 # predict to return nrow(flchain) values, but has to be set before the fit
fit2a <- glm(I(sex=='M') ~ age + log(creatinine), binomial, data= flchain)
fit2b <- glm(I(sex=='M') ~ age + kappa + lambda,  binomial, data= flchain)
tdata$eta1 <- predict(fit2a)
tdata$eta2 <- predict(fit2b)
cfit <- concordance(I(sex=="M") ~ eta1 + eta2, tdata)  # equal to the AUC
coef(cfit)  # males and females differ in creatinine, less in kappa/lambda
convec <- c(1,-1) # compute a contrast
c(test= unname(convec%*% coef(cfit)),
  std = unname(sqrt(convec %*% vcov(cfit) %*% convec)))


# compare multiple survival models
options(na.action = na.exclude)   # predict all 1384 obs, including missing
fit3 <- glm(pstat ~ age + sex + mspike + offset(log(ptime)),
            poisson, data= mgus2)
fit4 <- coxph(Surv(ptime, pstat) ~ age + sex + mspike, mgus2)
fit5 <- coxph(Surv(ptime, pstat) ~ age + sex + hgb + creat, mgus2)

tdata <- mgus2; tdata$ptime <- 60   # prediction at 60 months
p3 <- -predict(fit3, newdata=tdata)
p4 <- -predict(fit4) # high risk scores predict shorter survival
p5 <- -predict(fit5)
options(na.action = na.omit)      # return to the R default

cfit <- concordance(Surv(ptime, pstat) ~p3 +  p4 + p5, mgus2)
cfit
round(coef(cfit), 3)
round(cov2cor(vcov(cfit)), 3)  # high correlation

test <- c(1, -1, 0)  # contrast vector for model 1 - model 2
round(c(difference = test %*% coef(cfit),
        sd= sqrt(test %*% vcov(cfit) %*% test)), 3)
```

`concordancefit` *Compute the concordance*

Description

This is the working routine behind the `concordance` function. It is not meant to be called by users, but is available for other packages to use. Input arguments, for instance, are assumed to all be the correct length and type, and missing values are not allowed: the calling routine is responsible for these things.

Usage

```
concordancefit(y, x, strata, weights, ymin = NULL, ymax = NULL,
 timewt = c("n", "S", "S/G", "n/G2", "I"), cluster, influence =0,
```

```
ranks = FALSE, reverse = FALSE, timefix = TRUE, keepstrata=10,
std.err = TRUE)
```

Arguments

y	the response. It can be numeric, factor, or a Surv object
x	the predictor, a numeric vector
strata	optional numeric vector that stratifies the data
weights	options vector of case weights
ymin, ymax	restrict the comparison to response values in this range
timewt	the time weighting to be used
cluster, influence, ranks, reverse, timefix	see the help for the concordance function
keepstrata	either TRUE, FALSE, or an integer value. Computations are always done within stratum, then added. If the total number of strata greater than keepstrata, or keepstrata=FALSE, those subtotals are not kept in the output.
std.err	compute the standard error; not doing so saves some compute time.

Details

This function is provided for those who want a "direct" call to the concordance calculations, without using the formula interface. A primary use has been other packages. The routine does minimal checking of its input arguments, under the assumption that this has already been taken care of by the calling routine.

Value

a list containing the results

Author(s)

Terry Therneau

See Also

concordance

cox.zph *Test the Proportional Hazards Assumption of a Cox Regression*

Description

Test the proportional hazards assumption for a Cox regression model fit (coxph).

Usage

```
cox.zph(fit, transform="km", terms=TRUE, singledf=FALSE, global=TRUE)
```

Arguments

fit	the result of fitting a Cox regression model, using the coxph or coxme functions.
transform	a character string specifying how the survival times should be transformed before the test is performed. Possible values are "km", "rank", "identity" or a function of one argument.
terms	if TRUE, do a test for each term in the model rather than for each separate covariate. For a factor variable with k levels, for instance, this would lead to a k-1 degree of freedom test. The plot for such variables will be a single curve evaluating the linear predictor over time.
singledf	use a single degree of freedom test for terms that have multiple coefficients, i.e., the test that corresponds most closely to the plot. If terms=FALSE this argument has no effect.
global	should a global chi-square test be done, in addition to the per-variable or per-term tests tests.

Details

The computations require the original x matrix of the Cox model fit. Thus it saves time if the x=TRUE option is used in coxph. This function would usually be followed by both a plot and a print of the result. The plot gives an estimate of the time-dependent coefficient $\beta(t)$. If the proportional hazards assumption holds then the true $\beta(t)$ function would be a horizontal line. The table component provides the results of a formal score test for slope=0, a linear fit to the plot would approximate the test.

Random effects terms such a frailty or random effects in a coxme model are not checked for proportional hazards, rather they are treated as a fixed offset in model.

If the model contains strata by covariate interactions, then the y matrix may contain structural zeros, i.e., deaths (rows) that had no role in estimation of a given coefficient (column). These are marked as NA. If an entire row is NA, for instance after subscripting a cox.zph object, that row is removed.

Value

an object of class `"cox.zph"`, with components:

`table`	a matrix with one row for each variable, and optionally a last row for the global test. Columns of the matrix contain a score test of for addition of the time-dependent term, the degrees of freedom, and the two-sided p-value.
`x`	the transformed time axis.
`time`	the untransformed time values; there is one entry for each event time in the data
`strata`	for a stratified `coxph` `model`, the stratum of each of the events
`y`	the matrix of scaled Schoenfeld residuals. There will be one column per term or per variable (depending on the `terms` option above), and one row per event. The row labels are a rounded form of the original times.
`var`	a variance matrix for the covariates, used to create an approximate standard error band for plots
`transform`	the transform of time that was used
`call`	the calling sequence for the routine.

Note

In versions of the package before survival3.0 the function computed a fast approximation to the score test. Later versions compute the actual score test.

References

P. Grambsch and T. Therneau (1994), Proportional hazards tests and diagnostics based on weighted residuals. *Biometrika,* **81**, 515-26.

See Also

`coxph`, `Surv`.

Examples

```
fit <- coxph(Surv(futime, fustat) ~ age + ecog.ps,
             data=ovarian)
temp <- cox.zph(fit)
print(temp)                 # display the results
plot(temp)                  # plot curves
```

coxph *Fit Proportional Hazards Regression Model*

Description

Fits a Cox proportional hazards regression model. Time dependent variables, time dependent strata, multiple events per subject, and other extensions are incorporated using the counting process formulation of Andersen and Gill.

Usage

```
coxph(formula, data=, weights, subset,
      na.action, init, control,
      ties=c("efron","breslow","exact"),
      singular.ok=TRUE, robust,
      model=FALSE, x=FALSE, y=TRUE, tt, method=ties,
      id, cluster, istate, statedata, nocenter=c(-1, 0, 1), ...)
```

Arguments

formula	a formula object, with the response on the left of a ~ operator, and the terms on the right. The response must be a survival object as returned by the Surv function. For a multi-state model the formula may be a list of formulas.
data	a data.frame in which to interpret the variables named in the formula, or in the subset and the weights argument.
weights	vector of case weights, see the note below. For a thorough discussion of these see the book by Therneau and Grambsch.
subset	expression indicating which subset of the rows of data should be used in the fit. All observations are included by default.
na.action	a missing-data filter function. This is applied to the model.frame after any subset argument has been used. Default is options()\$na.action.
init	vector of initial values of the iteration. Default initial value is zero for all variables.
control	Object of class coxph.control specifying iteration limit and other control options. Default is coxph.control(...).
ties	a character string specifying the method for tie handling. If there are no tied death times all the methods are equivalent. The Efron approximation is used as the default, it is more accurate when dealing with tied death times, and is as efficient computationally. (But see below for multi-state models.) The "exact partial likelihood" is equivalent to a conditional logistic model, and is appropriate when the times are a small set of discrete values.
singular.ok	logical value indicating how to handle collinearity in the model matrix. If TRUE, the program will automatically skip over columns of the X matrix that are linear combinations of earlier columns. In this case the coefficients for such columns will be NA, and the variance matrix will contain zeros. For ancillary calculations, such as the linear predictor, the missing coefficients are treated as zeros.

robust	should a robust variance be computed. The default is TRUE if: there is a cluster argument, there are case weights that are not 0 or 1, or there are id values with more than one event.
id	optional variable name that identifies subjects. Only necessary when a subject can have multiple rows in the data, and there is more than one event type. This variable will normally be found in data.
cluster	optional variable which clusters the observations, for the purposes of a robust variance. If present, it implies robust. This variable will normally be found in data.
istate	optional variable giving the current state at the start each interval. This variable will normally be found in data.
statedata	optional data set used to describe multistate models.
model	logical value: if TRUE, the model frame is returned in component model.
x	logical value: if TRUE, the x matrix is returned in component x.
y	logical value: if TRUE, the response vector is returned in component y.
tt	optional list of time-transform functions.
method	alternate name for the ties argument.
nocenter	columns of the X matrix whose values lie strictly within this set are not recentered. Remember that a factor variable becomes a set of 0/1 columns.
...	Other arguments will be passed to coxph.control

Details

The proportional hazards model is usually expressed in terms of a single survival time value for each person, with possible censoring. Andersen and Gill reformulated the same problem as a counting process; as time marches onward we observe the events for a subject, rather like watching a Geiger counter. The data for a subject is presented as multiple rows or "observations", each of which applies to an interval of observation (start, stop].

The routine internally scales and centers data to avoid overflow in the argument to the exponential function. These actions do not change the result, but lead to more numerical stability. Any column of the X matrix whose values lie within nocenter list are not recentered. The practical consequence of the default is to not recenter dummy variables corresponding to factors. However, arguments to offset are not scaled since there are situations where a large offset value is a purposefully used. In general, however, users should not avoid very large numeric values for an offset due to possible loss of precision in the estimates.

All of weights, subset and offset are evaluated in the same way as variables in formula, that is first in data and then in the environment of formula. Note that values calculated inside the formula, such as mean(x) or pspline(x), are evaluated before subsetting – which may lead to unexpected results if used with subset. For more information see the Details section of model.frame.

Value

an object of class coxph representing the fit. See coxph.object and coxphms.object for details.

Side Effects

Depending on the call, the `predict`, `residuals`, and `survfit` routines may need to reconstruct the x matrix created by `coxph`. It is possible for this to fail, as in the example below in which the predict function is unable to find `tform`.

```
tfun <- function(tform) coxph(tform, data=lung)
fit <- tfun(Surv(time, status) ~ age)
predict(fit)
```

In such a case add the `model=TRUE` option to the `coxph` call to obviate the need for reconstruction, at the expense of a larger `fit` object.

Case weights

Case weights are treated as replication weights, i.e., a case weight of 2 is equivalent to having 2 copies of that subject's observation. When computers were much smaller grouping like subjects together was a common trick to used to conserve memory. Setting all weights to 2 for instance will give the same coefficient estimate but halve the variance. When the Efron approximation for ties (default) is employed replication of the data will not give exactly the same coefficients as the weights option, and in this case the weighted fit is arguably the correct one.

When the model includes a `cluster` term or the `robust=TRUE` option the computed variance is based on an infinitesimal jackknife, which treats any weights as sampling weights; setting all weights to 2 will in this case give the same variance as weights of 1. If there are any non-integer weights, the code also defaults `robust=TRUE`.

Special terms

There are a few special terms that may be used in the model equation: `strata`, `tt`, `pspline`, `frailty` and `ridge`. Each look like an ordinary function, e.g. `+ strata(group)` but are specially identifies so that they can be treated in a special way. The term `+ cluster(group)` is also but is depricated, use a a `cluster` arguement outside the formula instead.

A `strata` term identifies a stratified Cox model; separate baseline hazard functions are fit for each strata.

A time-transform term allows variables to vary dynamically in time. In this case the `tt` argument will be a function or a list of functions (if there are more than one tt() term in the model) giving the appropriate transform. See the examples below. If the `id` variable is not unique, it is assumed that it identifies clusters of correlated observations.

A time-transform term allows variables to vary dynamically in time. In this case the `tt` argument will be a function or a list of functions (if there are more than one tt() term in the model) giving the appropriate transform. See the examples below.

One user mistake that has recently arisen is to slavishly follow the advice of some coding guides and prepend `survival::` onto everthing, including the special terms, e.g., `survival::coxph(survival:Surv(time, status) ~ age + survival::strata(inst), data=lung)`

First, for this actually will not fit the model that was intended, as the :: interferes with the recognition of specials by the underlying `model.frame` function; there will a coefficient per institution rather than fitting a stratified model. A similar issue arises from using `stats::offset` as a term in a glm model.

From survival 3.8-1 onward it is also unnessary: the common formula arguments that are part of the survival namespace will be found in that namespace, i.e., Surv, strata, tt, pspline, and cluster. A survival:: prefex found on any of these is actually removed before evaluation of the formula. This only affects the formula itself; whether to use the qualified form `survival::coxph` for the call itself is a different discussion.

Robust variance

The robust estimate arises from many different arguments and thus has had many labels. It is variously known as the Huber sandwich estimator, White's estimate (linear models/econometrics), the Horvitz-Thompson estimate (survey sampling), the working independence variance (generalized estimating equations), the infinitesimal jackknife, and the Wei, Lin, Weissfeld (WLW) estimate.

If there is an `id` or `cluster` argument in the call, or `robust=TRUE`, or there are non-integer case weights, the robust variance is the default. If there are multiple rows per id and no id has more than one event and the underlying proportional hazards model is exactly correct, then the multiple rows for a given id will represent independent increments and the variance (robust or standard) obtained without an `id` statement will be an unbiased estimate. But it will not be equal to the estimate with `id` specified and we do not recommend it.

Convergence

In certain data cases the actual MLE estimate of a coefficient is infinity, e.g., a dichotomous variable where one of the groups has no events. When this happens the associated coefficient grows at a steady pace and a race condition will exist in the fitting routine: either the log likelihood converges, the information matrix becomes effectively singular, an argument to exp becomes too large for the computer hardware, or the maximum number of interactions is exceeded. (Most often number 1 is the first to occur.) The routine attempts to detect when this has happened, not always successfully. The primary consequence for the user is that the Wald statistic = coefficient/se(coefficient) is not valid in this case and should be ignored; the likelihood ratio and score tests remain valid however.

Ties

There are three possible choices for handling tied event times. The Breslow approximation is the easiest to program and hence became the first option coded for almost all computer routines. It then ended up as the default option when other options were added in order to "maintain backwards compatability". The Efron option is more accurate if there are a large number of ties, and it is the default option here. In practice the number of ties is usually small, in which case all the methods are statistically indistinguishable.

Using the "exact partial likelihood" approach the Cox partial likelihood is equivalent to that for matched logistic regression. (The `clogit` function uses the `coxph` code to do the fit.) It is technically appropriate when the time scale is discrete and has only a few unique values, and some packages refer to this as the "discrete" option. There is also an "exact marginal likelihood" due to Prentice which is not implemented here.

The calculation of the exact partial likelihood is numerically intense. Say for instance 180 subjects are at risk on day 7 of which 15 had an event; then the code needs to compute sums over all 180-choose-15 > 10^43 different possible subsets of size 15. There is an efficient recursive algorithm for this task, but even with this the computation can be insufferably long. With (start, stop) data it is much worse since the recursion needs to start anew for each unique start time.

Multi state models are a more difficult case. First of all, a proper extension of the Efron argument is much more difficult to do, and this author is not yet fully convinced that the resulting algorithm is defensible. Secondly, the current code for Efron case does not consistently compute that extended logic (and extension would require major changes in the code). Due to this complexity, the default is `ties='breslow'` for the multistate case. If `ties='efron'` is selected the current code will, in effect, only apply to to tied transitions of the same type.

A separate issue is that of artificial ties due to floating-point imprecision. See the vignette on this topic for a full explanation or the `timefix` option in `coxph.control`. Users may need to add `timefix=FALSE` for simulated data sets.

Penalized regression

`coxph` can maximise a penalised partial likelihood with arbitrary user-defined penalty. Supplied penalty functions include ridge regression (ridge), smoothing splines (pspline), and frailty models (frailty).

References

Andersen, P. and Gill, R. (1982). Cox's regression model for counting processes, a large sample study. *Annals of Statistics* **10**, 1100-1120.

Therneau, T., Grambsch, P., Modeling Survival Data: Extending the Cox Model. Springer-Verlag, 2000.

See Also

`coxph.object`, `coxphms.object`, `coxph.control`, `cluster`, `strata`, `Surv`, `survfit`, `pspline`.

Examples

```
# Create the simplest test data set
test1 <- list(time=c(4,3,1,1,2,2,3),
              status=c(1,1,1,0,1,1,0),
              x=c(0,2,1,1,1,0,0),
              sex=c(0,0,0,0,1,1,1))
# Fit a stratified model
coxph(Surv(time, status) ~ x + strata(sex), test1)
# Create a simple data set for a time-dependent model
test2 <- list(start=c(1,2,5,2,1,7,3,4,8,8),
              stop=c(2,3,6,7,8,9,9,9,14,17),
              event=c(1,1,1,1,1,1,1,0,0,0),
              x=c(1,0,0,1,0,1,1,1,0,0))
summary(coxph(Surv(start, stop, event) ~ x, test2))

#
# Create a simple data set for a time-dependent model
#
test2 <- list(start=c(1, 2, 5, 2, 1, 7, 3, 4, 8, 8),
               stop =c(2, 3, 6, 7, 8, 9, 9, 9,14,17),
               event=c(1, 1, 1, 1, 1, 1, 1, 0, 0, 0),
               x    =c(1, 0, 0, 1, 0, 1, 1, 1, 0, 0) )
```

```
summary( coxph( Surv(start, stop, event) ~ x, test2))

# Fit a stratified model, clustered on patients

bladder1 <- bladder[bladder$enum < 5, ]
coxph(Surv(stop, event) ~ (rx + size + number) * strata(enum),
      cluster = id, bladder1)

# Fit a time transform model using current age
coxph(Surv(time, status) ~ ph.ecog + tt(age), data=lung,
     tt=function(x,t,...) pspline(x + t/365.25))
```

`coxph.control` *Ancillary arguments for controlling coxph fits*

Description

This is used to set various numeric parameters controlling a Cox model fit. Typically it would only be used in a call to `coxph`.

Usage

```
coxph.control(eps = 1e-09, toler.chol = .Machine$double.eps^0.75,
iter.max = 20, toler.inf = sqrt(eps), outer.max = 10, timefix=TRUE)
```

Arguments

`eps`	Iteration continues until the relative change in the log partial likelihood is less than eps, or the absolute change is less than sqrt(eps). Must be positive.
`toler.chol`	Tolerance for detection of singularity during a Cholesky decomposition of the variance matrix, i.e., for detecting a redundant predictor variable.
`iter.max`	Maximum number of iterations to attempt for convergence.
`toler.inf`	Tolerance criteria for the warning message about a possible infinite coefficient value.
`outer.max`	For a penalized coxph model, e.g. with pspline terms, there is an outer loop of iteration to determine the penalty parameters; maximum number of iterations for this outer loop.
`timefix`	Resolve any near ties in the time variables.

Details

The convergence tolerances are a balance. Users think they want THE maximum point of the likelihood surface, and for well behaved data sets where this is quadratic near the max a high accuracy is fairly inexpensive: the number of correct digits approximately doubles with each iteration. Conversely, a drop of .0001 from the maximum in any given direction will be correspond to only about 1/20 of a standard error change in the coefficient. Statistically, more precision than this is straining

at a gnat. Based on this the author originally had set the tolerance to 1e-5, but relented in the face of multiple "why is the answer different than package X" queries.

Asking for results that are too close to machine precision (double.eps) is a fool's errand; a reasonable critera is often the square root of that precision. The Cholesky decompostion needs to be held to a higher standard than the overall convergence criterion, however. The `tolerance.inf` value controls a warning message; if it is too small incorrect warnings can appear, if too large some actual cases of an infinite coefficient will not be detected.

The most difficult cases are data sets where the MLE coefficient is infinite; an example is a data set where at each death time, it was the subject with the largest covariate value who perished. In that situation the coefficient increases at each iteration while the log-likelihood asymptotes to a maximum. As iteration proceeds there is a race condition for three endpoint: exp(coef) overflows, the Hessian matrix become singular, or the change in loglik is small enough to satisfy the convergence criterion. The first two are difficult to anticipate and lead to numeric diffculties, which is another argument for moderation in the choice of `eps`.

See the vignette "Roundoff error and tied times" for a more detailed explanation of the `timefix` option. In short, when time intervals are created via subtraction then two time intervals that are actually identical can appear to be different due to floating point round off error, which in turn can make `coxph` and `survfit` results dependent on things such as the order in which operations were done or the particular computer that they were run on. Such cases are unfortunatedly not rare in practice. The `timefix=TRUE` option adds logic similar to `all.equal` to ensure reliable results. In analysis of simulated data sets, however, where often by defintion there can be no duplicates, the option will often need to be set to `FALSE` to avoid spurious merging of close numeric values.

Value

a list containing the values of each of the above constants

See Also

`coxph`

`coxph.detail` *Details of a Cox Model Fit*

Description

Returns the individual contributions to the first and second derivative matrix, at each unique event time.

Usage

```
coxph.detail(object, riskmat=FALSE, rorder=c("data", "time"))
```

Arguments

object	a Cox model object, i.e., the result of coxph.
riskmat	include the at-risk indicator matrix in the output?
rorder	should the rows of x, y and riskmat be returned in the original data order, or sorted by time within strata.

Details

This function may be useful for those who wish to investigate new methods or extensions to the Cox model. The example below shows one way to calculate the Schoenfeld residuals.

Value

a list with components

time	the vector of unique event times
nevent	the number of events at each of these time points.
means	a matrix with one row for each event time and one column for each variable in the Cox model, containing the weighted mean of the variable at that time, over all subjects still at risk at that time. The weights are the risk weights exp(x %*% fit$coef).
nrisk	number of subjects at risk.
score	the contribution to the score vector (first derivative of the log partial likelihood) at each time point.
imat	the contribution to the information matrix (second derivative of the log partial likelihood) at each time point.
hazard	the hazard increment. Note that the hazard and variance of the hazard are always for some particular future subject. This routine uses object$means as the future subject.
varhaz	the variance of the hazard increment.
x, y	copies of the input data.
strata	only present for a stratified Cox model, this is a table giving the number of time points of component time that were contributed by each of the strata.
wtrisk	the weighted number at risk
riskmat	a matrix with one row for each observation and one colum for each unique event time, containing a 0/1 value to indicate whether that observation was (1) or was not (0) at risk at the given time point. Rows are in the order of the original data (after removal of any missings by coxph), or in time order.

See Also

coxph, residuals.coxph

Examples

```
fit   <- coxph(Surv(futime,fustat) ~ age + rx + ecog.ps, ovarian, x=TRUE)
fitd  <- coxph.detail(fit)
#  There is one Schoenfeld residual for each unique death.  It is a
# vector (covariates for the subject who died) - (weighted mean covariate
# vector at that time).  The weighted mean is defined over the subjects
# still at risk, with exp(X beta) as the weight.

events <- fit$y[,2]==1
etime  <- fit$y[events,1]   #the event times --- may have duplicates
indx   <- match(etime, fitd$time)
schoen <- fit$x[events,] - fitd$means[indx,]
```

---

coxph.object *Proportional Hazards Regression Object*

Description

This class of objects is returned by the coxph class of functions to represent a fitted proportional hazards model. Objects of this class have methods for the functions `print`, `summary`, `residuals`, `predict` and `survfit`.

Arguments

`coefficients`	the vector of coefficients. If the model is over-determined there will be missing values in the vector corresponding to the redundant columns in the model matrix.
`var`	the variance matrix of the coefficients. Rows and columns corresponding to any missing coefficients are set to zero.
`naive.var`	this component will be present only if the `robust` option was true. If so, the `var` component will contain the robust estimate of variance, and this component will contain the ordinary estimate. (A far better name would be `asymp.var` since it contains the model-based asympotitic variance estimate, which is not necessarily "naive"; but that ship has sailed.)
`loglik`	a vector of length 2 containing the log-likelihood with the initial values and with the final values of the coefficients.
`score`	value of the efficient score test, at the initial value of the coefficients.
`rscore`	the robust log-rank statistic, if a robust variance was requested.
`wald.test`	the Wald test of whether the final coefficients differ from the initial values.
`iter`	number of iterations used.
`linear.predictors`	the vector of linear predictors, one per subject. Note that this vector has been centered, see `predict.coxph` for more details.
`residuals`	the martingale residuals.

means	vector of values used as the reference for each covariate. For instance, a later call to predict(fit, type='risk') will give the hazard ratio between an observation and this reference. (For most covariates this will contain the mean.)
n	the number of observations used in the fit.
nevent	the number of events (usually deaths) used in the fit.
n.id	if the call had an id argument, the number of unique id values
concordance	a vector of length 6, containing the number of pairs that are concordant, discordant, tied on x, tied on y, and tied on both, followed by the standard error of the concordance statistic.
first	the first derivative vector at the solution.
weights	the vector of case weights, if one was used.
method	the method used for handling tied survival times.
na.action	the na.action attribute, if any, that was returned by the na.action routine.
timefix	the value of the timefix option used in the fit
...	The object will also contain the following, for documentation see the lm object: terms, assign, formula, call, and, optionally, x, y, and/or frame.

Components

The following components must be included in a legitimate coxph object.

See Also

coxph, coxph.detail, cox.zph, residuals.coxph, survfit, survreg.

---

coxph.wtest *Compute a quadratic form*

Description

This function is used internally by several survival routines. It computes a simple quadratic form, while properly dealing with missings.

Usage

```
coxph.wtest(var, b, toler.chol = 1e-09)
```

Arguments

var	variance matrix
b	vector
toler.chol	tolerance for the internal cholesky decomposition

Details

Compute b' V-inverse b. Equivalent to sum(b * solve(V,b)), except for the case of redundant covariates in the original model, which lead to NA values in V and b.

Value

a real number

Author(s)

Terry Therneau

---

coxphms.object *Multi-state Proportional Hazards Regression Object*

Description

This class of objects is returned by the coxph class of functions to represent a fitted hazards model, when the model has multiple states. The object inherits from the coxph class.

Arguments

states	a character vector listing the states in the model
cmap	the coefficient map. A matrix containing a column for each transition and a row for each coefficient, the value maps that transition/coefficient pair to a position in the coefficient vector. If a particular covariate is not used by a transition the matrix will contain a zero in that position, if two transitions share a coefficient the matrix will contain repeats.
smap	the stratum map. The row labeled '(Baseline)' identifies transitions that do or do not share a baseline hazard. Further rows correspond to strata() terms in the model, each of which may apply to some transitions and not others.
rmap	mapping for the residuals and linear predictors. A two column matrix with one row for each element of the vectors and two columns, the first contains the data row and the second the transition.

Details

In a multi-state model a set of intermediate observations is created during the computation, with a separate set of data rows for each transition. An observation (id and time interval) that is at risk for more than one transition will for instance have a linear predictor and residual for each of the potential transitions. As a result the vector of linear predictors will be longer than the number of observations. The rmap matrix shows the mapping.

Components

The object has all the components of a coxph object, with the following additions and variations.

See Also

coxph, coxph.object

---

coxsurv.fit *A direct interface to the 'computational engine' of survfit.coxph*

Description

This program is mainly supplied to allow other packages to invoke the survfit.coxph function at a 'data' level rather than a 'user' level. It does no checks on the input data that is provided, which can lead to unexpected errors if that data is wrong.

Usage

```
coxsurv.fit(ctype, stype, se.fit, varmat, cluster,
            y, x, wt, risk, position, strata, oldid,
            y2, x2, risk2, strata2, id2, unlist=TRUE)
```

Arguments

stype	survival curve computation: 1=direct, 2=exp(-cumulative hazard)
ctype	cumulative hazard computation: 1=Breslow, 2=Efron
se.fit	if TRUE, compute standard errors
varmat	the variance matrix of the coefficients
cluster	vector to control robust variance
y	the response variable used in the Cox model. (Missing values removed of course.)
x	covariate matrix used in the Cox model
wt	weight vector for the Cox model. If the model was unweighted use a vector of 1s.
risk	the risk score exp(X beta + offset) from the fitted Cox model.
position	optional argument controlling what is counted as 'censored'. Due to time dependent covariates, for instance, a subject might have start, stop times of (1,5)(5,30)(30,100). Times 5 and 30 are not 'real' censorings. Position is 1 for a real start, 2 for an actual end, 3 for both, 0 for neither.
strata	strata variable used in the Cox model. This will be a factor.
oldid	identifier for subjects with multiple rows in the original data.
y2, x2, risk2, strata2	variables for the hypothetical subjects, for which prediction is desired
id2	optional; if present and not NULL this should be a vector of identifiers of length nrow(x2). A non-null value signifies that x2 contains time dependent covariates, in which case this identifies which rows of x2 go with each subject.
unlist	if FALSE the result will be a list with one element for each strata. Otherwise the strata are "unpacked" into the form found in a survfit object.

Value

a list containing nearly all the components of a survfit object. All that is missing is to add the confidence intervals, the type of the original model's response (as in a coxph object), and the class.

Note

The source code for for both this function and survfit.coxph is written using noweb. For complete documentation see the inst/sourcecode.pdf file.

Author(s)

Terry Therneau

See Also

survfit.coxph

---

diabetic *Ddiabetic retinopathy*

Description

Partial results from a trial of laser coagulation for the treatment of diabetic retinopathy.

Usage

```
diabetic
data(diabetic, package="survival")
```

Format

A data frame with 394 observations on the following 8 variables.

id subject id

laser laser type: xenon or argon

age age at diagnosis

eye a factor with levels of left right

trt treatment: 0 = no treatment, 1= laser

risk risk group of 6-12

time time to event or last follow-up

status status of 0= censored or 1 = visual loss

Details

The 197 patients in this dataset were a 50% random sample of the patients with "high-risk" diabetic retinopathy as defined by the Diabetic Retinopathy Study (DRS). Each patient had one eye randomized to laser treatment and the other eye received no treatment. For each eye, the event of interest was the time from initiation of treatment to the time when visual acuity dropped below 5/200 two visits in a row. Thus there is a built-in lag time of approximately 6 months (visits were every 3 months). Survival times in this dataset are therefore the actual time to blindness in months, minus the minimum possible time to event (6.5 months). Censoring was caused by death, dropout, or end of the study.

References

Huster, Brookmeyer and Self, Biometrics, 1989.

American Journal of Ophthalmology, 1976, 81:4, pp 383-396

Examples

```
# juvenile diabetes is defined as and age less than 20
juvenile <- 1*(diabetic$age < 20)
coxph(Surv(time, status) ~ trt + juvenile, cluster= id,
            data= diabetic)
```

---

dsurvreg *Distributions available in survreg.*

Description

Density, cumulative distribution function, quantile function and random generation for the set of distributions supported by the survreg function.

Usage

```
dsurvreg(x, mean, scale=1, distribution='weibull', parms)
psurvreg(q, mean, scale=1, distribution='weibull', parms)
qsurvreg(p, mean, scale=1, distribution='weibull', parms)
rsurvreg(n, mean, scale=1, distribution='weibull', parms)
```

Arguments

x	vector of quantiles. Missing values (NAs) are allowed.
q	vector of quantiles. Missing values (NAs) are allowed.
p	vector of probabilities. Missing values (NAs) are allowed.
n	number of random deviates to produce
mean	vector of location (linear predictor) parameters for the model. This is replicated to be the same length as p, q or n.

scale	vector of (positive) scale factors. This is replicated to be the same length as p, q or n.
distribution	character string giving the name of the distribution. This must be one of the elements of survreg.distributions
parms	optional parameters, if any, of the distribution. For the t-distribution this is the degrees of freedom.

Details

Elements of q or p that are missing will cause the corresponding elements of the result to be missing.

The location and scale values are as they would be for survreg. The label "mean" was an unfortunate choice (made in mimicry of qnorm); a more correct label would be "linear predictor". Since almost none of these distributions are symmetric the location parameter is not actually a mean.

The survreg routines use the parameterization found in chapter 2 of Kalbfleisch and Prentice. Translation to the usual parameterization found in a textbook is not always obvious. For example, the Weibull distribution has cumulative distribution function $F(t) = 1 - e^{-(\lambda t)^p}$. The actual fit uses the fact that $\log(t)$ has an extreme value distribution, with location and scale of α, σ, which are the location and scale parameters reported by the survreg function. The parameters are related by $\sigma = 1/p$ and $\alpha = -\log(\lambda$. The stats::dweibull routine is parameterized in terms of shape and scale parameters which correspond to p and $1/\lambda$ in the K and P notation. Combining these we see that shape = $1/\sigma$ and scale = $\exp alpha$.

Value

density (dsurvreg), probability (psurvreg), quantile (qsurvreg), or for the requested distribution with mean and scale parameters mean and sd.

References

Kalbfleisch, J. D. and Prentice, R. L. (1970). *The Statistical Analysis of Failure Time Data* Wiley, New York.

References

Kalbfleisch, J. D. and Prentice, R. L., The statistical analysis of failure time data, Wiley, 2002.

See Also

survreg, Normal

Examples

```
# List of distributions available
names(survreg.distributions)
## Not run:
 [1] "extreme"     "logistic"    "gaussian"    "weibull"     "exponential"
 [6] "rayleigh"    "loggaussian" "lognormal"   "loglogistic" "t"

## End(Not run)
```

```
# Compare results
all.equal(dsurvreg(1:10, 2, 5, dist='lognormal'), dlnorm(1:10, 2, 5))

# Hazard function for a Weibull distribution
x   <- seq(.1, 3, length=30)
haz <- dsurvreg(x, 2, 3)/ (1-psurvreg(x, 2, 3))
## Not run:
plot(x, haz, log='xy', ylab="Hazard") #line with slope (1/scale -1)

## End(Not run)

# Estimated CDF of a simple Weibull
fit <- survreg(Surv(time, status) ~ 1, data=lung)
pp <- 1:99/100
q1 <- qsurvreg(pp, coef(fit), fit$scale)
q2 <- qweibull(pp, shape= 1/fit$scale, scale= exp(coef(fit)))
all.equal(q1, q2)
## Not run:
plot(q1, pp, type='l', xlab="Months", ylab="CDF")

## End(Not run)
# per the help page for dweibull, the mean is scale * gamma(1 + 1/shape)
c(mean = exp(coef(fit))* gamma(1 + fit$scale))
```

`finegray` *Create data for a Fine-Gray model*

Description

The Fine-Gray model can be fit by first creating a special data set, and then fitting a weighted Cox model to the result. This routine creates the data set.

Usage

```
finegray(formula, data, weights, subset, na.action= na.pass, etype,
    prefix="fg", count, id, timefix=TRUE)
```

Arguments

`formula`	a standard model formula, with survival on the left and covariates on the right.
`data`	an optional data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model.
`weights`	optional vector of observation weights
`subset`	an optional vector specifying a subset of observations to be used in the fitting process.
`na.action`	a function which indicates what should happen when the data contain NAs. The default is set by the na.action setting of options.

etype	the event type for which a data set will be generated. The default is to use whichever is listed first in the multi-state survival object.
prefix	the routine will add 4 variables to the data set: a start and end time for each interval, status, and a weight for the interval. The default names of these are "fgstart", "fgstop", "fgstatus", and "fgwt"; the prefix argument determines the initial portion of the new names.
count	a variable name in the output data set for an optional variable that will contain the the replication count for each row of the input data. If a row is expanded into multiple lines it will contain 1, 2, etc.
id	optional, the variable name in the data set which identifies subjects.
timefix	process times through the aeqSurv function to eliminate potential roundoff issues.

Details

The function expects a multi-state survival expression or variable as the left hand side of the formula, e.g. Surv(atime, astat) where astat is a factor whose first level represents censoring and remaining levels are states. The output data set will contain simple survival data (status = 0 or 1) for a single endpoint of interest. For exposition call this endpoint A and lump all others as endpoint B. In the output data set subjects who experience endpoint B become censored observations whose times are artificially extended to the right, with a decreasing case weight from interval to interval. The output data set will normally contain many more rows than the input.

The algorithm allows for delayed entry, and only a limited form of time-dependent covariates. That is, when subjects with endpoint B are extended, those future covariate values stay constant; so there is an implicit assumption that no more changes would have occurred if the event had not intervened and follow-up had been longer. For predictable time-dependent covariates the final data set could be further processed to fix this, but this is not included in the function. Geskus for example considers an example with different calendar epochs, corresponding to a change in standard medical practice for the disese, as a covariate. dependent covariates. If there are time dependent covariates or delayed entry, e.g.., the input data set had Surv(entry, exit, stat) as the left hand side, then an id statement is required. The program does data checks in this case, and needs to know which rows belong to each subject.

The output data set will often have gaps. Say that there were events at time 50 and 100 (and none between) and censoring at 60, 70, and 80. Formally, a non event subjects at risk from 50 to 100 will have different weights in each of the 3 intervals 50-60, 60-70, and 80-100, but because the middle interval does not span any event times the subsequent Cox model will never use that row. The finegray output omits such rows.

In the simulation results of Geskus, he found that the robust sandwich estimate of the standard error was equal to the Fine and Gray estimate up to 3 digits, we thus recomment including the cluster option in the subsequent coxph call.

Value

a data frame

Author(s)

Terry Therneau

References

Fine JP and Gray RJ (1999) A proportional hazards model for the subdistribution of a competing risk. JASA 94:496-509.

Geskus RB (2011). Cause-Specific Cumulative Incidence Estimation and the Fine and Gray Model Under Both Left Truncation and Right Censoring. Biometrics 67, 39-49.

See Also

coxph, aeqSurv

Examples

```
# Treat time to death and plasma cell malignancy as competing risks
etime <- with(mgus2, ifelse(pstat==0, futime, ptime))
event <- with(mgus2, ifelse(pstat==0, 2*death, 1))
event <- factor(event, 0:2, labels=c("censor", "pcm", "death"))

# FG model for PCM
pdata <- finegray(Surv(etime, event) ~ ., data=mgus2)
fgfit <- coxph(Surv(fgstart, fgstop, fgstatus) ~ age + sex,
                     weight=fgwt, data=pdata, cluster=id)

# Compute the weights separately by sex
adata <- finegray(Surv(etime, event) ~ . + strata(sex),
             data=mgus2, na.action=na.pass)
```

---

flchain *Assay of serum free light chain for 7874 subjects.*

Description

This is a stratified random sample containing 1/2 of the subjects from a study of the relationship between serum free light chain (FLC) and mortality. The original sample contains samples on approximately 2/3 of the residents of Olmsted County aged 50 or greater.

Usage

```
flchain
data(flchain, package="survival")
```

Format

A data frame with 7874 persons containing the following variables.

`age` age in years

`sex` F=female, M=male

`sample.yr` the calendar year in which a blood sample was obtained

`kappa` serum free light chain, kappa portion

`lambda` serum free light chain, lambda portion

`flc.grp` the FLC group for the subject, as used in the original analysis

`creatinine` serum creatinine

`mgus` 1 if the subject had been diagnosed with monoclonal gammapothy (MGUS)

`futime` days from enrollment until death. Note that there are 3 subjects whose sample was obtained on their death date.

`death` 0=alive at last contact date, 1=dead

`chapter` for those who died, a grouping of their primary cause of death by chapter headings of the International Code of Diseases ICD-9

Details

In 1995 Dr. Robert Kyle embarked on a study to determine the prevalence of monoclonal gammopathy of undetermined significance (MGUS) in Olmsted County, Minnesota, a condition which is normally only found by chance from a test (serum electrophoresis) which is ordered for other causes. Later work suggested that one component of immunoglobulin production, the serum free light chain, might be a possible marker for immune disregulation. In 2010 Dr. Angela Dispenzieri and colleagues assayed FLC levels on those samples from the original study for which they had patient permission and from which sufficient material remained for further testing. They found that elevated FLC levels were indeed associated with higher death rates.

Patients were recruited when they came to the clinic for other appointments, with a final random sample of those who had not yet had a visit since the study began. An interesting side question is whether there are differences between early, mid, and late recruits.

This data set contains an age and sex stratified random sample that includes 7874 of the original 15759 subjects. The original subject identifiers and dates have been removed to protect patient identity. Subsampling was done to further protect this information.

Source

The primary investigator (A Dispenzieri) and statistician (T Therneau) for the study.

References

A Dispenzieri, J Katzmann, R Kyle, D Larson, T Therneau, C Colby, R Clark, G Mead, S Kumar, LJ Melton III and SV Rajkumar (2012). Use of monclonal serum immunoglobulin free light chains to predict overall survival in the general population, Mayo Clinic Proceedings 87:512-523.

R Kyle, T Therneau, SV Rajkumar, D Larson, M Plevak, J Offord, A Dispenzieri, J Katzmann, and LJ Melton, III, 2006, Prevalence of monoclonal gammopathy of undetermined significance, New England J Medicine 354:1362-1369.

Examples

```
data(flchain)
age.grp <-  cut(flchain$age, c(49,54, 59,64, 69,74,79, 89, 110),
               labels= paste(c(50,55,60,65,70,75,80,90),
                             c(54,59,64,69,74,79,89,109), sep='-'))
table(flchain$sex, age.grp)
```

frailty *Random effects terms*

Description

The frailty function allows one to add a simple random effects term to a Cox model.

Usage

```
frailty(x, distribution="gamma", ...)
frailty.gamma(x, sparse = (nclass > 5), theta, df, eps = 1e-05,
         method = c("em","aic", "df", "fixed"), ...)
frailty.gaussian(x, sparse = (nclass > 5), theta, df,
         method =c("reml","aic", "df", "fixed"), ...)
frailty.t(x, sparse = (nclass > 5), theta, df, eps = 1e-05, tdf = 5,
         method = c("aic", "df", "fixed"), ...)
```

Arguments

x	the variable to be entered as a random effect. It is always treated as a factor.
distribution	either the gamma, gaussian or t distribution may be specified. The routines frailty.gamma, frailty.gaussian and frailty.t do the actual work.
...	Arguments for specific distribution, including (but not limited to)
sparse	cutoff for using a sparse coding of the data matrix. If the total number of levels of x is larger than this value, then a sparse matrix approximation is used. The correct cutoff is still a matter of exploration: if the number of levels is very large (thousands) then the non-sparse calculation may not be feasible in terms of both memory and compute time. Likewise, the accuracy of the sparse approximation appears to be related to the maximum proportion of subjects in any one class, being best when no one class has a large membership.
theta	if specified, this fixes the variance of the random effect. If not, the variance is a parameter, and a best solution is sought. Specifying this implies method='fixed'.
df	if specified, this fixes the degrees of freedom for the random effect. Specifying this implies method='df'. Only one of theta or df should be specified.
method	the method used to select a solution for theta, the variance of the random effect. The fixed corresponds to a user-specified value, and no iteration is done. The df selects the variance such that the degrees of freedom for the random effect matches a user specified value. The aic method seeks to maximize Akaike's information criteria 2*(partial likelihood - df). The em and reml methods are specific to Cox models with gamma and gaussian random effects, respectively. Please see further discussion below.
tdf	the degrees of freedom for the t-distribution.
eps	convergence criteria for the iteration on theta.

Details

The `frailty` plugs into the general penalized modeling framework provided by the `coxph` and `survreg` routines. This framework deals with likelihood, penalties, and degrees of freedom; these aspects work well with either parent routine.

Therneau, Grambsch, and Pankratz show how maximum likelihood estimation for the Cox model with a gamma frailty can be accomplished using a general penalized routine, and Ripatti and Palmgren work through a similar argument for the Cox model with a gaussian frailty. Both of these are specific to the Cox model. Use of gamma/ml or gaussian/reml with `survreg` does not lead to valid results.

The extensible structure of the penalized methods is such that the penalty function, such as `frailty` or `pspine`, is completely separate from the modeling routine. The strength of this is that a user can plug in any penalization routine they choose. A weakness is that it is very difficult for the modeling routine to know whether a sensible penalty routine has been supplied.

Note that use of a frailty term implies a mixed effects model and use of a cluster term implies a GEE approach; these cannot be mixed.

The `coxme` package has superseded this method. It is faster, more stable, and more flexible.

Value

this function is used in the model statement of either `coxph` or `survreg`. It's results are used internally.

References

S Ripatti and J Palmgren, Estimation of multivariate frailty models using penalized partial likelihood, Biometrics, 56:1016-1022, 2000.

T Therneau, P Grambsch and VS Pankratz, Penalized survival models and frailty, J Computational and Graphical Statistics, 12:156-175, 2003.

See Also

coxph, survreg

Examples

```
# Random institutional effect
coxph(Surv(time, status) ~ age + frailty(inst, df=4), lung)

# Litter effects for the rats data
rfit2a <- coxph(Surv(time, status) ~ rx +
                  frailty.gaussian(litter, df=13, sparse=FALSE), rats,
                  subset= (sex=='f'))
rfit2b <- coxph(Surv(time, status) ~ rx +
                  frailty.gaussian(litter, df=13, sparse=TRUE), rats,
                  subset= (sex=='f'))
```

gbsg *Breast cancer data sets used in Royston and Altman (2013)*

Description

The gbsg data set contains patient records from a 1984-1989 trial conducted by the German Breast Cancer Study Group (GBSG) of 720 patients with node positive breast cancer; it retains the 686 patients with complete data for the prognostic variables.

Usage

```
gbsg
data(cancer, package="survival")
```

Format

A data set with 686 observations and 11 variables.

`pid` patient identifier

`age` age, years

`meno` menopausal status (0= premenopausal, 1= postmenopausal)

`size` tumor size, mm

`grade` tumor grade

`nodes` number of positive lymph nodes

`pgr` progesterone receptors (fmol/l)

`er` estrogen receptors (fmol/l)

`hormon` hormonal therapy, 0= no, 1= yes

`rfstime` recurrence free survival time; days to first of reccurence, death or last follow-up

`status` 0= alive without recurrence, 1= recurrence or death

Details

These data sets are used in the paper by Royston and Altman. The Rotterdam data is used to create a fitted model, and the GBSG data for validation of the model. The paper gives references for the data source.

References

Patrick Royston and Douglas Altman, External validation of a Cox prognostic model: principles and methods. BMC Medical Research Methodology 2013, 13:33

See Also

rotterdam

heart *Stanford Heart Transplant data*

Description

Survival of patients on the waiting list for the Stanford heart transplant program.

Usage

```
heart
data(heart, package="survival")
```

Format

jasa: original data

birth.dt:	birth date
accept.dt:	acceptance into program
tx.date:	transplant date
fu.date:	end of followup
fustat:	dead or alive
surgery:	prior bypass surgery
age:	age (in years)
futime:	followup time
wait.time:	time before transplant
transplant:	transplant indicator
mismatch:	mismatch score
hla.a2:	particular type of mismatch
mscore:	another mismatch score
reject:	rejection occurred

jasa1, heart: processed data

start, stop, event:	Entry and exit time and status for this interval of time
age:	age-48 years
year:	year of acceptance (in years after 1 Nov 1967)
surgery:	prior bypass surgery 1=yes
transplant:	received transplant 1=yes
id:	patient id

Source

J Crowley and M Hu (1977), Covariance analysis of heart transplant survival data. *Journal of the American Statistical Association*, **72**, 27–36.

See Also

stanford2

---

hoel *Mouse cancer data*

Description

Days until occurence of cancer for male mice

Usage

```
data("cancer")
```

Format

A data frame with 181 observations on the following 4 variables.

`trt` treatment assignment: `Control` or `Germ-free`

`days` days until death

`outcome` outcome: `censor`, `thymic lymphoma`, `reticulum cell sarcoma` `other causes`

`id` mouse id

Details

Two groups of male mice were given 300 rads of radiation and followed for cancer incidence. One group was maintained in a germ free environment. The data set is used as an example of competing risks in Kalbfleisch and Prentice. The germ-free environment has little effect on the rate of occurence of thymic lymphoma, but significantly delays the other causes of death.

Note

The Ontology Search website defines reticulm cell sarcoma as "An antiquated term that refers to a non-Hodgkin lymphoma composed of diffuse infiltrates of large, often anaplastic lymphocytes".

Source

The data can be found in appendix I of Kalbfleisch and Prentice.

References

Hoel, D.G. (1972), A representation of mortality data by competing risks. Biometrics 33, 1-30. Kalbfleisch, J.D. and Prentice, R.L. (1980). The statistical analysis of failure time data.

Examples

```
hsurv <- survfit(Surv(days, outcome) ~ trt, data = hoel, id= id)
plot(hsurv, lty=1:2, col=rep(1:3, each=2), lwd=2, xscale=30.5,
      xlab="Months", ylab= "Death")
legend("topleft", c("Lymphoma control", "Lymphoma germ free",
                    "Sarcoma control", "Sarcoma germ free",
                    "Other control", "Other germ free"),
       col=rep(1:3, each=2), lty=1:2, lwd=2, bty='n')
hfit <- coxph(Surv(days, outcome) ~ trt, data= hoel, id = id)
```

---

is.ratetable *Verify that an object is of class ratetable.*

Description

The function verifies not only the `class` attribute, but the structure of the object.

Usage

```
is.ratetable(x, verbose=FALSE)
```

Arguments

x	the object to be verified.
verbose	if TRUE and the object is not a ratetable, then return a character string describing the way(s) in which x fails to be a proper ratetable object.

Details

Rate tables are used by the `pyears` and `survexp` functions, and normally contain death rates for some population, categorized by age, sex, or other variables. They have a fairly rigid structure, and the `verbose` option can help in creating a new rate table.

Value

returns TRUE if x is a ratetable, and FALSE or a description if it is not.

See Also

pyears, survexp.

Examples

```
is.ratetable(survexp.us)  # True
is.ratetable(lung)        # False
```

kidney *Kidney catheter data*

Description

Data on the recurrence times to infection, at the point of insertion of the catheter, for kidney patients using portable dialysis equipment. Catheters may be removed for reasons other than infection, in which case the observation is censored. Each patient has exactly 2 observations.

This data has often been used to illustrate the use of random effects (frailty) in a survival model. However, one of the males (id 21) is a large outlier, with much longer survival than his peers. If this observation is removed no evidence remains for a random subject effect.

Usage

```
kidney
# or
data(cancer, package="survival")
```

Format

patient:	id
time:	time
status:	event status
age:	in years
sex:	1=male, 2=female
disease:	disease type (0=GN, 1=AN, 2=PKD, 3=Other)
frail:	frailty estimate from original paper

Note

The original paper ignored the issue of tied times and so is not exactly reproduced by the survival package.

Source

CA McGilchrist, CW Aisbett (1991), Regression with frailty in survival analysis. *Biometrics* **47**, 461–66.

Examples

```
kfit <- coxph(Surv(time, status)~ age + sex + disease + frailty(id), kidney)
kfit0 <- coxph(Surv(time, status)~ age + sex + disease, kidney)
kfitm1 <- coxph(Surv(time,status) ~ age + sex + disease +
frailty(id, dist='gauss'), kidney)
```

levels.Surv *Return the states of a multi-state Surv object*

Description

For a multi-state Surv object, this will return the names of the states.

Usage

```
## S3 method for class 'Surv'
levels(x)
```

Arguments

x	a Surv object

Value

for a multi-state Surv object, the vector of state names (excluding censoring); or NULL for an ordinary Surv object

Examples

```
y1 <- Surv(c(1,5, 9, 17,21, 30),
           factor(c(0, 1, 2,1,0,2), 0:2, c("censored", "progression", "death")))
levels(y1)

y2 <- Surv(1:6, rep(0:1, 3))
y2
levels(y2)
```

lines.survfit *Add Lines or Points to a Survival Plot*

Description

Often used to add the expected survival curve(s) to a Kaplan-Meier plot generated with plot.survfit.

Usage

```
## S3 method for class 'survfit'
lines(x, type="s", pch=3, col=1, lty=1,
        lwd=1, cex=1, mark.time=FALSE, xmax,
        fun, conf.int=FALSE,
        conf.times, conf.cap=.005, conf.offset=.012,
        conf.type = c("log", "log-log", "plain", "logit", "arcsin"),
```

```
        mark, noplot="(s0)", cumhaz= FALSE,  cumprob= FALSE, ...)
## S3 method for class 'survexp'
lines(x, type="l", ...)
## S3 method for class 'survfit'
points(x, fun, censor=FALSE, col=1, pch,
        noplot="(s0)", cumhaz=FALSE, ...)
```

Arguments

x	a survival object, generated from the survfit or survexp functions.
type	the line type, as described in lines. The default is a step function for survfit objects, and a connected line for survexp objects. All other arguments for lines.survexp are identical to those for lines.survfit.
col, lty, lwd, cex	vectors giving the mark symbol, color, line type, line width and character size for the added curves. Of this set only color is applicable to points.
pch	plotting characters for points, in the style of matplot, i.e., either a single string of characters of which the first will be used for the first curve, etc; or a vector of characters or integers, one element per curve.
mark	a historical alias for pch
censor	should censoring times be displayed for the points function?
mark.time	controls the labeling of the curves. If FALSE, no labeling is done. If TRUE, then curves are marked at each censoring time. If mark.time is a numeric vector, then curves are marked at the specified time points.
xmax	optional cutoff for the right hand of the curves.
fun	an arbitrary function defining a transformation of the survival curve. For example fun=log is an alternative way to draw a log-survival curve (but with the axis labeled with log(S) values). Four often used transformations can be specified with a character argument instead: "log" is the same as using the log=T option, "event" plots cumulative events (f(y) = 1-y), "cumhaz" plots the cumulative hazard function (f(y) = -log(y)) and "cloglog" creates a complimentary log-log survival plot (f(y) = log(-log(y))) along with log scale for the x-axis.
conf.int	if TRUE, confidence bands for the curves are also plotted. If set to "only", then only the CI bands are plotted, and the curve itself is left off. This can be useful for fine control over the colors or line types of a plot.
conf.times	optional vector of times at which to place a confidence bar on the curve(s). If present, these will be used instead of confidence bands.
conf.cap	width of the horizontal cap on top of the confidence bars; only used if conf.times is used. A value of 1 is the width of the plot region.
conf.offset	the offset for confidence bars, when there are multiple curves on the plot. A value of 1 is the width of the plot region. If this is a single number then each curve's bars are offset by this amount from the prior curve's bars, if it is a vector the values are used directly.
conf.type	One of "plain", "log" (the default), "log-log", "logit", or "none". Only enough of the string to uniquely identify it is necessary. The first option causes

	confidence intervals not to be generated. The second causes the standard intervals curve +- k *se(curve), where k is determined from conf.int. The log option calculates intervals based on the cumulative hazard or log(survival). The log-log option bases the intervals on the log hazard or log(-log(survival)), and the logit option on log(survival/(1-survival)).
noplot	for multi-state models, curves with this label will not be plotted. The default corresponds to an unspecified state.
cumhaz	plot the cumulative hazard, rather than the survival or probability in state.
cumprob	for a multi-state curve, plot the probabilities in state 1, (state1 + state2), (state1 + state2 + state3), ... If cumprob is an integer vector the totals will be in the order indicated.
...	other graphical parameters

Details

When the survfit function creates a multi-state survival curve the resulting object has class 'survfitms'. The only difference in the plots is that that it defaults to a curve that goes from lower left to upper right (starting at 0), where survival curves default to starting at 1 and going down. All other options are identical.

If the user set an explicit range in an earlier plot.survfit call, e.g. via xlim or xmax, subsequent calls to this function remember the right hand cutoff. This memory can be erased by options(plot.survfit) <- NULL.

Value

a list with components x and y, containing the coordinates of the last point on each of the curves (but not of the confidence limits). This may be useful for labeling. If cumprob=TRUE then y will be a matrix with one row per curve and x will be all the time points. This may be useful for adding shading.

Side Effects

one or more curves are added to the current plot.

See Also

lines, par, plot.survfit, survfit, survexp.

Examples

```
fit <- survfit(Surv(time, status==2) ~ sex, pbc,subset=1:312)
plot(fit, mark.time=FALSE, xscale=365.25,
        xlab='Years', ylab='Survival')
lines(fit[1], lwd=2)    #darken the first curve and add marks


# Add expected survival curves for the two groups,
#   based on the US census data
# The data set does not have entry date, use the midpoint of the study
```

```
efit <- survexp(~sex, data=pbc, times= (0:24)*182, ratetable=survexp.us,
                rmap=list(sex=sex, age=age*365.35, year=as.Date('1979/01/01')))
temp <- lines(efit, lty=2, lwd=2:1)
text(temp, c("Male", "Female"), adj= -.1) #labels just past the ends
title(main="Primary Biliary Cirrhosis, Observed and Expected")
```

`logan` *Data from the 1972-78 GSS data used by Logan*

Description

Intergenerational occupational mobility data with covariates.

Usage

```
logan
data(logan, package="survival")
```

Format

A data frame with 838 observations on the following 4 variables.

occupation subject's occupation, a factor with levels `farm`, `operatives`, `craftsmen`, `sales`, and `professional`

focc father's occupation

education total years of schooling, 0 to 20

race levels of `non-black` and `black`

Source

General Social Survey data, see the web site for detailed information on the variables. https://gss.norc.org/.

References

Logan, John A. (1983). A Multivariate Model for Mobility Tables. *American Journal of Sociology* 89: 324-349.

`logLik.coxph` *logLik method for a Cox model*

Description

The logLik function for survival models

Usage

```
## S3 method for class 'coxph'
logLik(object, ...)
## S3 method for class 'survreg'
logLik(object, ...)
```

Arguments

`object`	the result of a `coxph` or `survreg` fit
`...`	optional arguments for other instances of the method

Details

The logLik function is used by summary functions in R such as `AIC`. For a Cox model, this method returns the partial likelihood. The number of degrees of freedom (df) used by the fit and the effective number of observations (nobs) are added as attributes. Per Raftery and others, the effective number of observations is the taken to be the number of events in the data set.

For a `survreg` model the proper value for the effective number of observations is still an open question (at least to this author). For right censored data the approach of `logLik.coxph` is the possible the most sensible, but for interval censored observations the result is unclear. The code currently does not add a *nobs* attribute.

Value

an object of class `logLik`

Author(s)

Terry Therneau

References

Robert E. Kass and Adrian E. Raftery (1995). "Bayes Factors". J. American Statistical Assoc. 90 (430): 791.

Raftery A.E. (1995), "Bayesian Model Selection in Social Research", Sociological methodology, 111-196.

See Also

`logLik`

lung *NCCTG Lung Cancer Data*

Description

Survival in patients with advanced lung cancer from the North Central Cancer Treatment Group. Performance scores rate how well the patient can perform usual daily activities.

Usage

```
lung
data(cancer, package="survival")
```

Format

inst:	Institution code
time:	Survival time in days
status:	censoring status 1=censored, 2=dead
age:	Age in years
sex:	Male=1 Female=2
ph.ecog:	ECOG performance score as rated by the physician. 0=asymptomatic, 1= symptomatic but completely ambulator
ph.karno:	Karnofsky performance score (bad=0-good=100) rated by physician
pat.karno:	Karnofsky performance score as rated by patient
meal.cal:	Calories consumed at meals
wt.loss:	Weight loss in last six months (pounds)

Note

The use of 1/2 for alive/dead instead of the usual 0/1 is a historical footnote. For data contained on punch cards, IBM 360 Fortran treated blank as a zero, which led to a policy within the section of Biostatistics to never use "0" as a data value since one could not distinguish it from a missing value. The policy became a habit, as is often the case; and the 1/2 coding rule endured long beyond the demise of punch cards.

Source

Terry Therneau

References

Loprinzi CL. Laurie JA. Wieand HS. Krook JE. Novotny PJ. Kugler JW. Bartel J. Law M. Bateman M. Klatt NE. et al. Prospective evaluation of prognostic variables from patient-completed questionnaires. North Central Cancer Treatment Group. Journal of Clinical Oncology. 12(3):601-7, 1994.

lvcf *Last Value Carried Forward*

Description

Replace missing values in a covariate with the last non-missing value, separately for each id.

Usage

```
lvcf(id, x, time)
```

Arguments

id	subject identifier
x	the covariate which will be modified
time	optional; used to sort observations within subject

Details

If an analysis includes a covariate that was measured repeatedly, cholesterol say, the data may often include visits where this was not measured. It is convenient in that case to use, for each subject, the most recently measured value. When a value is extended beyond the last measurement, rather that just replacement of intermediate values, there is the potential for bias, e.g., if the reason for discontinuation is predictive of the future trajectory of values.

If time is missing, then the order of the values within subject is assumed to be the calendar time order. It is not necessary for the the data to be sorted by id.

Value

an updated copy of x

Note

Other R packages also implement an locf function, e.g., na.locf in the zoo package or LOCF in DescTools. However, they do not include the id argument, so per subjct application requires further processing. The tdc operation in tmerge also performs lvcf substitution.

We have used the phrase "last value" rather than "last observation" since this function only addresses a single variable with each call, not the entire row of a data frame (observation) as some others do.

Author(s)

Terry Therneau

See Also

tmerge

Examples

```
newplat <- with(pbcseq, lvcf(id, platelet))
table(is.na(pbcseq$platelet), is.na(newplat))
```

---

mgus *Monoclonal gammopathy data*

Description

Natural history of 241 subjects with monoclonal gammopathy of undetermined significance (MGUS).

Usage

```
mgus
mgus1
data(cancer, package="survival")
```

Format

mgus: A data frame with 241 observations on the following 12 variables.

id:	subject id
age:	age in years at the detection of MGUS
sex:	`male` or `female`
dxyr:	year of diagnosis
pcdx:	for subjects who progress to a plasma cell malignancy the subtype of malignancy: multiple myeloma (MM) is the most common, followed by amyloidosis (AM), macroglobulinemia (MA), and other lymphprolifative disorders (LP)
pctime:	days from MGUS until diagnosis of a plasma cell malignancy
futime:	days from diagnosis to last follow-up
death:	1= follow-up is until death
alb:	albumin level at MGUS diagnosis
creat:	creatinine at MGUS diagnosis
hgb:	hemoglobin at MGUS diagnosis
mspike:	size of the monoclonal protein spike at diagnosis

mgus1: The same data set in start,stop format. Contains the id, age, sex, and laboratory variable described above along with

start, stop:	sequential intervals of time for each subject
status:	=1 if the interval ends in an event
event:	a factor containing the event type: censor, death, or plasma cell malignancy
enum:	event number for each subject: 1 or 2

Details

Plasma cells are responsible for manufacturing immunoglobulins, an important part of the immune defense. At any given time there are estimated to be about 10^6 different immunoglobulins in the circulation at any one time. When a patient has a plasma cell malignancy the distribution will become dominated by a single isotype, the product of the malignant clone, visible as a spike on a serum protein electrophoresis. Monoclonal gammopathy of undertermined significance (MGUS) is the presence of such a spike, but in a patient with no evidence of overt malignancy. This data set of 241 sequential subjects at Mayo Clinic was the groundbreaking study defining the natural history of such subjects. Due to the diligence of the principle investigator 0 subjects have been lost to follow-up.

Three subjects had MGUS detected on the day of death. In data set `mgus1` these subjects have the time to MGUS coded as .5 day before the death in order to avoid tied times.

These data sets were updated in Jan 2015 to correct some small errors.

Source

Mayo Clinic data courtesy of Dr. Robert Kyle.

References

R Kyle, Benign monoclonal gammopathy – after 20 to 35 years of follow-up, Mayo Clinic Proc 1993; 68:26-36.

Examples

```
# Create the competing risk curves for time to first of death or PCM
sfit <- survfit(Surv(start, stop, event) ~ sex, mgus1, id=id,
                subset=(enum==1))
print(sfit)  # the order of printout is the order in which they plot

plot(sfit, xscale=365.25, lty=c(2,2,1,1), col=c(1,2,1,2),
     xlab="Years after MGUS detection", ylab="Proportion")
legend(0, .8, c("Death/male", "Death/female", "PCM/male", "PCM/female"),
       lty=c(1,1,2,2), col=c(2,1,2,1), bty='n')

title("Curves for the first of plasma cell malignancy or death")
# The plot shows that males have a higher death rate than females (no
# surprise) but their rates of conversion to PCM are essentially the same.
```

---

mgus2 *Monoclonal gammopathy data*

Description

Natural history of 1341 sequential patients with monoclonal gammopathy of undetermined significance (MGUS). This is a superset of the `mgus` data, at a later point in the accrual process

Usage

```
mgus2
data(cancer, package="survival")
```

Format

A data frame with 1384 observations on the following 10 variables.

`id` subject identifier

`age` age at diagnosis, in years

`sex` a factor with levels `F M`

`dxyr` year of diagnosis

`hgb` hemoglobin

`creat` creatinine

`mspike` size of the monoclonal serum splike

`ptime` time until progression to a plasma cell malignancy (PCM) or last contact, in months

`pstat` occurrence of PCM: 0=no, 1=yes

`futime` time until death or last contact, in months

`death` occurrence of death: 0=no, 1=yes

Details

This is an extension of the study found in the `mgus` data set, containing enrollment through 1994 and follow-up through 1999.

Source

Mayo Clinic data courtesy of Dr. Robert Kyle. All patient identifiers have been removed, age rounded to the nearest year, and follow-up times rounded to the nearest month.

References

R. Kyle, T. Therneau, V. Rajkumar, J. Offord, D. Larson, M. Plevak, and L. J. Melton III, A long-terms study of prognosis in monoclonal gammopathy of undertermined significance. New Engl J Med, 346:564-569 (2002).

model.frame.coxph *Model.frame method for coxph objects*

Description

Recreate the model frame of a coxph fit.

Usage

```
## S3 method for class 'coxph'
model.frame(formula, ...)
```

Arguments

formula	the result of a coxph fit
...	other arguments to model.frame

Details

For details, see the manual page for the generic function. This function would rarely be called by a user, it is mostly used inside functions like residual that need to recreate the data set from a model in order to do further calculations.

Value

the model frame used in the original fit, or a parallel one for new data.

Author(s)

Terry Therneau

See Also

model.frame

model.matrix.coxph *Model.matrix method for coxph models*

Description

Reconstruct the model matrix for a cox model.

Usage

```
## S3 method for class 'coxph'
model.matrix(object, data=NULL, contrast.arg =
 object$contrasts, ...)
```

Arguments

object	the result of a coxph model
data	optional, a data frame from which to obtain the data
contrast.arg	optional, a contrasts object describing how factors should be coded
...	other possible argument to model.frame

Details

When there is a data argument this function differs from most of the other model.matrix methods in that the response variable for the original formula is *not* required to be in the data.

If the data frame contains a terms attribute then it is assumed to be the result of a call to model.frame, otherwise a call to model.frame is applied with the data as an argument.

Value

The model matrix for the fit

Author(s)

Terry Therneau

See Also

model.matrix

Examples

```
fit1 <- coxph(Surv(time, status) ~ age + factor(ph.ecog), data=lung)
xfit <- model.matrix(fit1)

fit2 <- coxph(Surv(time, status) ~ age + factor(ph.ecog), data=lung,
                                  x=TRUE)
all.equal(model.matrix(fit1), fit2$x)
```

---

myeloid *Acute myeloid leukemia*

Description

This simulated data set is based on a trial in acute myeloid leukemia.

Usage

```
myeloid
data(cancer, package="survival")
```

Format

A data frame with 646 observations on the following 9 variables.

`id` subject identifier, 1-646

`trt` treatment arm A or B

`sex` f=female, m=male

`flt3` mutations of the FLT3 gene, a factor with levels of A, B, C

`futime` time to death or last follow-up

`death` 1 if `futime` is a death, 0 for censoring

`txtime` time to hematropetic stem cell transplant

`crtime` time to complete response

`rltime` time to relapse of disease

Details

This data set is used to illustrate multi-state survival curves. It is based on the actual study in the reference below. A subset of subjects was de-identifed, reordered, and then all of the time values randomly perturbed.

Mutations in the FLT3 domain occur in about 1/3 of AML patients, the additional agent in treatment arm B was presumed to target this anomaly. All subjects had a FLT mutation, either internal tandem duplications (ITD) (divided into low vs high) +- mutations in the TKD domain, or TKD mutations only. This was a stratification factor for treatment assignment in the study. The levels of A, B, C correspond to increasing severity of the mutation burden.

References

Le-Rademacher JG, Peterson RA, Therneau TM, Sanford BL, Stone RM, Mandrekar SJ. Application of multi-state models in cancer clinical trials. Clin Trials. 2018 Oct; 15 (5):489-498

Examples

```
coxph(Surv(futime, death) ~ trt + flt3, data=myeloid)
# See the mstate vignette for a more complete analysis
```

---

myeloma *Survival times of patients with multiple myeloma*

Description

Survival times of 3882 subjects with multiple myeloma, seen at Mayo Clinic from 1947–1996.

Usage

```
myeloma
data("cancer", package="survival")
```

Format

A data frame with 3882 observations on the following 5 variables.

`id` subject identifier

`year` year of entry into the study

`entry` time from diagnosis of MM until entry (days)

`futime` follow up time (days)

`death` status at last follow-up: 0 = alive, 1 = death

Details

Subjects who were diagnosed at Mayo will have `entry` =0, those who were diagnosed elsewhere and later referred will have positive values.

References

R. Kyle, Long term survival in multiple myeloma. New Eng J Medicine, 1997

Examples

```
# Incorrect survival curve, which ignores left truncation
fit1 <- survfit(Surv(futime, death) ~ 1, myeloma)
# Correct curve
fit2 <- survfit(Surv(entry, futime, death) ~1, myeloma)
```

---

nafld *Non-alcoholic fatty liver disease*

Description

Data sets containing the data from a population study of non-alcoholic fatty liver disease (NAFLD). Subjects with the condition and a set of matched control subjects were followed forward for metabolic conditions, cardiac endpoints, and death.

Usage

```
nafld1
       nafld2
       nafld3
data(nafld, package="survival")
```

Format

nafld1 is a data frame with 17549 observations on the following 10 variables.

id subject identifier

age age at entry to the study

male 0=female, 1=male

weight weight in kg

height height in cm

bmi body mass index

case.id the id of the NAFLD case to whom this subject is matched

futime time to death or last follow-up

status 0= alive at last follow-up, 1=dead

nafld2 is a data frame with 400123 observations and 4 variables containing laboratory data

id subject identifier

days days since index date

test the type of value recorded

value the numeric value

nafld3 is a data frame with 34340 observations and 3 variables containing outcomes

id subject identifier

days days since index date

event the endpoint that occurred

Details

The primary reference for this study is Allen (2018). Nonalcoholic fatty liver disease (NAFLD) was renamed metabolic dysfunction-associated steatotic liver disease (MASLD) in June 2023. The new name is intended to better reflect the disease's underlying causes, identify subgroups of patients, and avoid stigmatizing words.

The incidence of MASLD has been rising rapidly in the last decade and it is now one of the main drivers of hepatology practice *Tapper2018*. It is essentially the presence of excess fat in the liver, and parallels the ongoing obesity epidemic. Approximately 20-25% of MASLD patients will develop the inflammatory state of metabolic dysfunction associated steatohepatitis (MASH), leading to fibrosis and eventual end-stage liver disease. MASLD can be accurately diagnosed by MRI methods, but MASH diagnosis currently requires a biopsy.

The current study constructed a population cohort of all adult MASLD subjects from 1997 to 2014 along with 4 potential controls for each case. To protect patient confidentiality all time intervals are in days since the index date; none of the dates from the original data were retained. Subject age is their integer age at the index date, and the subject identifier is an arbitrary integer. As a final protection, we include only a 90% random sample of the data. As a consequence analyses results will not exactly match the original paper.

There are 3 data sets: nafld1 contains baseline data and has one observation per subject, nafld2 has one observation for each (time dependent) continuous measurement, and nafld3 has one observation for each yes/no outcome that occured.

Source

Data obtained from the author.

References

AM Allen, TM Therneau, JJ Larson, A Coward, VK Somers and PS Kamath, Nonalcoholic Fatty Liver Disease Incidence and Impact on Metabolic Burden and Death: A 20 Year Community Study, Hepatology 67:1726-1736, 2018.

`neardate` *Find the index of the closest value in data set 2, for each entry in data set one.*

Description

A common task in medical work is to find the closest lab value to some index date, for each subject.

Usage

```
neardate(id1, id2, y1, y2, best = c("after", "prior"),
nomatch = NA_integer_)
```

Arguments

`id1`	vector of subject identifiers for the index group
`id2`	vector of identifiers for the reference group
`y1`	normally a vector of dates for the index group, but any orderable data type is allowed
`y2`	reference set of dates
`best`	if `best='prior'` find the index of the first y2 value less than or equal to the target y1 value, for each subject. If `best='after'` find the first y2 value which is greater than or equal to the target y1 value, for each subject.
`nomatch`	the value to return for items without a match

Details

This routine is closely related to `match` and to `findInterval`, the first of which finds exact matches and the second closest matches. This finds the closest matching date within sets of exactly matching identifiers. Closest date matching is often needed in clinical studies. For example data set 1 might contain the subject identifier and the date of some procedure and data set set 2 has the dates and values for laboratory tests, and the query is to find the first test value after the intervention but no closer than 7 days.

The `id1` and `id2` arguments are similar to `match` in that we are searching for instances of `id1` that will be found in `id2`, and the result is the same length as `id1`. However, instead of returning the first match with `id2` this routine returns the one that best matches with respect to `y1`.

The y1 and y2 arguments need not be dates, the function works for any data type such that the expression c(y1, y2) gives a sensible, sortable result. Be careful about matching Date and DateTime values and the impact of time zones, however, see as.POSIXct. If y1 and y2 are not of the same class the user is on their own. Since there exist pairs of unmatched data types where the result could be sensible, the routine will in this case proceed under the assumption that "the user knows what they are doing". Caveat emptor.

Value

the index of the matching observations in the second data set, or the nomatch value for no successful match

Author(s)

Terry Therneau

See Also

match, findInterval

Examples

```
data1 <- data.frame(id = 1:10,
                    entry.dt = as.Date(paste("2011", 1:10, "5", sep='-')))
temp1 <- c(1,4,5,1,3,6,9, 2,7,8,12,4,6,7,10,12,3)
data2 <- data.frame(id = c(1,1,1,2,2,4,4,5,5,5,6,8,8,9,10,10,12),
                    lab.dt = as.Date(paste("2011", temp1, "1", sep='-')),
                    chol = round(runif(17, 130, 280)))

#first cholesterol on or after enrollment
indx1 <- neardate(data1$id, data2$id, data1$entry.dt, data2$lab.dt)
data2[indx1, "chol"]

# Closest one, either before or after.
#
indx2 <- neardate(data1$id, data2$id, data1$entry.dt, data2$lab.dt,
                  best="prior")
ifelse(is.na(indx1), indx2, # none after, take before
       ifelse(is.na(indx2), indx1, #none before
       ifelse(abs(data2$lab.dt[indx2]- data1$entry.dt) <
              abs(data2$lab.dt[indx1]- data1$entry.dt), indx2, indx1)))

# closest date before or after, but no more than 21 days prior to index
indx2 <- ifelse((data1$entry.dt - data2$lab.dt[indx2]) >21, NA, indx2)
ifelse(is.na(indx1), indx2, # none after, take before
       ifelse(is.na(indx2), indx1, #none before
       ifelse(abs(data2$lab.dt[indx2]- data1$entry.dt) <
              abs(data2$lab.dt[indx1]- data1$entry.dt), indx2, indx1)))
```

`nostutter` *Remove repeated events*

Description

For a subject with repeated follow-up visits, do not count the second, third, etc. visit in a row with the same state as a repeated event.

Usage

```
nostutter(id, x, censor = 0, single=FALSE)
```

Arguments

`id`	subject identifier
`x`	covariate of interest
`censor`	censoring code to be used as the "no event at this time" value
`single`	if TRUE, a given level can appear only once

Details

When an covariate is recorded multiple times during follow-up, and that variable is being used as an endpoint in a multi-state hazard model, we will sometimes want to consider each instance as a new event and sometimes not. An example of the first kind is the `cgd` data set; each row in which `status =1` is a new infection. An example of the second is the `nafld` data when we treat the number of co-morbidities as a state; multiple visits with the same total do not count as entry into a new state.

This routine will often be called on a current state variable before using that variable as an event variable in `tmerge`.

Value

an updated vector with repeated values marked as `censor`

See Also

`lvcf`, `tmerge`

nsk *Natural splines with knot heights as the basis.*

Description

Create the design matrix for a natural spline, such that the coefficient of the resulting fit are the values of the function at the knots.

Usage

```
nsk(x, df = NULL, knots = NULL, intercept = FALSE, b = 0.05,
    Boundary.knots = quantile(x, c(b, 1 - b), na.rm = TRUE))
```

Arguments

x	the predictor variable. Missing values are allowed.
df	degrees of freedom. One can supply df rather than knots; ns() then chooses df - 1 - intercept knots at suitably chosen quantiles of x (which will ignore missing values). The default, df = NULL, sets the number of inner knots as length(knots).
knots	breakpoints that define the spline. The default is no knots; together with the natural boundary conditions this results in a basis for linear regression on x. Typical values are the mean or median for one knot, quantiles for more knots. See also Boundary.knots.
intercept	if TRUE, an intercept is included in the basis; default is FALSE
b	default placement of the boundary knots. A value of bs=0 will replicate the default behavior of ns.
Boundary.knots	boundary points at which to impose the natural boundary conditions and anchor the B-spline basis. Beyond these points the function is assumed to be linear. If both knots and Boundary.knots are supplied, the basis parameters do not depend on x. Data can extend beyond Boundary.knots

Details

The nsk function behaves identically to the ns function, with two exceptions. The primary one is that the returned basis is such that coefficients correspond to the value of the fitted function at the knot points. If intercept = FALSE, there will be k-1 coefficients corresponding to the k knots, and they will be the difference in predicted value between knots 2-k and knot 1. The primary advantage to the basis is that the coefficients are directly interpretable. A second is that tests for the linear and non-linear components are simple contrasts.

The second differnce with ns is one of opinion with respect to the default position for the boundary knots. The default here is closer to that found in the rms::rcs function.

This function is a trial if a new idea, it's future inclusion in the package is not yet guarranteed.

Value

A matrix of dimension length(x) * df where either df was supplied or, if knots were supplied, df = length(knots) + 1 + intercept. Attributes are returned that correspond to the arguments to kns, and explicitly give the knots, Boundary.knots etc for use by predict.kns().

Note

A thin flexible metal or wooden strip is called a spline, and is the traditional method for laying out a smooth curve, e.g., for a ship's hull or an airplane wing. Pins are put into a board and the strip is passed through them, each pin is a 'knot'.

A mathematical spline is a piecewise function between each knot. A linear spline will be a set of connected line segments, a quadratic spline is a set of connected local quadratic functions, constrained to have a continuous first derivative, a cubic spline is cubic between each knot, constrained to have continuous first and second derivatives, and etc. Mathematical splines are not an exact representation of natural splines: being a physical object the wood or metal strip will have continuous derivatives of all orders. Cubic splines are commonly used because they are sufficiently smooth to look natural to the human eye.

If the mathematical spline is further constrained to be linear beyond the end knots, this is often called a 'natural spline', due to the fact that a wooden or metal spline will also be linear beyond the last knots. Another name for the same object is a 'restricted cubic spline', since it is achieved in code by adding further constraints. Given a vector of data points and a set of knots, it is possible to create a basis matrix X with one column per knot, such that ordinary regression of X on y will fit the cubic spline function, hence these are also called 'regression splines'. (One of these three labels is no better or worse than another, in our opinion).

Given a basis matrix X with k columns, the matrix Z= XT for any k by k nonsingular matrix T is is also a basis matrix, and will result in identical predicted values, but a new set of coefficients gamma = (T-inverse) beta in place of beta. One can choose the basis functions so that X is easy to construct, to make the regression numerically stable, to make tests easier, or based on other considerations. It seems as though every spline library returns a different basis set, which unfortunately makes fits difficult to compare between packages. This is yet one more basis set, chosen to make the coefficients more interpretable.

See Also

ns

Examples

```
# make some dummy data
tdata <- data.frame(x= lung$age, y = 10*log(lung$age-35) + rnorm(228, 0, 2))
fit1 <- lm(y ~ -1 + nsk(x, df=4, intercept=TRUE) , data=tdata)
fit2 <- lm(y ~ nsk(x, df=3), data=tdata)

# the knots (same for both fits)
knots <- unlist(attributes(fit1$model[[2]])[c('Boundary.knots', 'knots')])
sort(unname(knots))
unname(coef(fit1))  # predictions at the knot points

unname(coef(fit1)[-1] - coef(fit1)[1])  # differences: yhat[2:4] - yhat[1]
```

```
unname(coef(fit2))[-1]                 # ditto

## Not run:
plot(y ~ x, data=tdata)
points(sort(knots), coef(fit1), col=2, pch=19)
coef(fit)[1] + c(0, coef(fit)[-1])

## End(Not run)
```

---

nwtco *Data from the National Wilm's Tumor Study*

Description

Measurement error example. Tumor histology predicts survival, but prediction is stronger with central lab histology than with the local institution determination.

Usage

```
nwtco
data(nwtco, package="survival")
```

Format

A data frame with 4028 observations on the following 9 variables.

`seqno` id number

`instit` Histology from local institution

`histol` Histology from central lab

`stage` Disease stage

`study` study

`rel` indicator for relapse

`edrel` time to relapse

`age` age in months

`in.subcohort` Included in the subcohort for the example in the paper

References

NE Breslow and N Chatterjee (1999), Design and analysis of two-phase studies with binary outcome applied to Wilms tumour prognosis. *Applied Statistics* **48**, 457–68.

Examples

```
with(nwtco, table(instit,histol))
anova(coxph(Surv(edrel,rel)~histol+instit,data=nwtco))
anova(coxph(Surv(edrel,rel)~instit+histol,data=nwtco))
```

ovarian *Ovarian Cancer Survival Data*

Description

Survival in a randomised trial comparing two treatments for ovarian cancer

Usage

```
ovarian
data(cancer, package="survival")
```

Format

futime:	survival or censoring time
fustat:	censoring status
age:	in years
resid.ds:	residual disease present (1=no,2=yes)
rx:	treatment group
ecog.ps:	ECOG performance status (1 is better, see reference)

Source

Terry Therneau

References

Edmunson, J.H., Fleming, T.R., Decker, D.G., Malkasian, G.D., Jefferies, J.A., Webb, M.J., and Kvols, L.K., Different Chemotherapeutic Sensitivities and Host Factors Affecting Prognosis in Advanced Ovarian Carcinoma vs. Minimal Residual Disease. Cancer Treatment Reports, 63:241-47, 1979.

pbc *Mayo Clinic Primary Biliary Cholangitis Data*

Description

Primary biliary cholangitis is an autoimmune disease leading to destruction of the small bile ducts in the liver. Progression is slow but inexhortable, eventually leading to cirrhosis and liver decompensation. The condition has been recognised since at least 1851 and was named "primary biliary cirrhosis" in 1949. Because cirrhosis is a feature only of advanced disease, a change of its name to "primary biliary cholangitis" was proposed by patient advocacy groups in 2014.

This data is from the Mayo Clinic trial in PBC conducted between 1974 and 1984. A total of 424 PBC patients, referred to Mayo Clinic during that ten-year interval, met eligibility criteria for the randomized placebo controlled trial of the drug D-penicillamine. The first 312 cases in the data set participated in the randomized trial and contain largely complete data. The additional 112 cases did not participate in the clinical trial, but consented to have basic measurements recorded and to be followed for survival. Six of those cases were lost to follow-up shortly after diagnosis, so the data here are on an additional 106 cases as well as the 312 randomized participants.

A nearly identical data set found in appendix D of Fleming and Harrington; this version has fewer missing values.

Usage

```
pbc
data(pbc, package="survival")
```

Format

age:	in years
albumin:	serum albumin (g/dl)
alk.phos:	alkaline phosphotase (U/liter)
ascites:	presence of ascites
ast:	aspartate aminotransferase, once called SGOT (U/ml)
bili:	serum bilirunbin (mg/dl)
chol:	serum cholesterol (mg/dl)
copper:	urine copper (ug/day)
edema:	0 no edema, 0.5 untreated or successfully treated 1 edema despite diuretic therapy
hepato:	presence of hepatomegaly or enlarged liver
id:	case number
platelet:	platelet count
protime:	standardised blood clotting time
sex:	m/f
spiders:	blood vessel malformations in the skin
stage:	histologic stage of disease (needs biopsy)
status:	status at endpoint, 0/1/2 for censored, transplant, dead
time:	number of days between registration and the earlier of death, transplantion, or study analysis in July, 1986
trt:	1/2/NA for D-penicillmain, placebo, not randomised
trig:	triglycerides (mg/dl)

Source

T Therneau and P Grambsch (2000), *Modeling Survival Data: Extending the Cox Model*, Springer-Verlag, New York. ISBN: 0-387-98784-3.

See Also

pbcseq

---

pbcseq *Mayo Clinic Primary Biliary Cirrhosis, sequential data*

Description

This data is a continuation of the PBC data set, and contains the follow-up laboratory data for each study patient. An analysis based on the data can be found in Murtagh, et. al.

The primary PBC data set contains only baseline measurements of the laboratory parameters. This data set contains multiple laboratory results, but only on the 312 randomized patients. Some baseline data values in this file differ from the original PBC file, for instance, the data errors in prothrombin time and age which were discovered after the original analysis (see Fleming and Harrington, figure 4.6.7). It also contains further follow-up.

One feature of the data deserves special comment. The last observation before death or liver transplant often has many more missing covariates than other data rows. The original clinical protocol for these patients specified visits at 6 months, 1 year, and annually thereafter. At these protocol visits lab values were obtained for a pre-specified battery of tests. "Extra" visits, often undertaken because of worsening medical condition, did not necessarily have all this lab work. The missing values are thus potentially informative.

Usage

```
pbcseq
data(pbc, package="survival")
```

Format

id:	case number
age:	in years
sex:	m/f
trt:	1/2/NA for D-penicillmain, placebo, not randomised
time:	number of days between registration and the earlier of death, transplantion, or study analysis in July, 1986
status:	status at endpoint, 0/1/2 for censored, transplant, dead
day:	number of days between enrollment and this visit date all measurements below refer to this date
albumin:	serum albumin (mg/dl)
alk.phos:	alkaline phosphotase (U/liter)

ascites:	presence of ascites
ast:	aspartate aminotransferase, once called SGOT (U/ml)
bili:	serum bilirunbin (mg/dl)
chol:	serum cholesterol (mg/dl)
copper:	urine copper (ug/day)
edema:	0 no edema, 0.5 untreated or successfully treated 1 edema despite diuretic therapy
hepato:	presence of hepatomegaly or enlarged liver
platelet:	platelet count
protime:	standardised blood clotting time
spiders:	blood vessel malformations in the skin
stage:	histologic stage of disease (needs biopsy)
trig:	triglycerides (mg/dl)

Source

T Therneau and P Grambsch, "Modeling Survival Data: Extending the Cox Model", Springer-Verlag, New York, 2000. ISBN: 0-387-98784-3.

References

Murtaugh PA. Dickson ER. Van Dam GM. Malinchoc M. Grambsch PM. Langworthy AL. Gips CH. "Primary biliary cirrhosis: prediction of short-term survival based on repeated patient visits." Hepatology. 20(1.1):126-34, 1994.

Fleming T and Harrington D., "Counting Processes and Survival Analysis", Wiley, New York, 1991.

See Also

pbc

Examples

```
# Create the start-stop-event triplet needed for coxph
first <- with(pbcseq, c(TRUE, diff(id) !=0)) #first id for each subject
last  <- c(first[-1], TRUE)  #last id

time1 <- with(pbcseq, ifelse(first, 0, day))
time2 <- with(pbcseq, ifelse(last,  futime, c(day[-1], 0)))
event <- with(pbcseq, ifelse(last,  status, 0))

fit1 <- coxph(Surv(time1, time2, event) ~ age + sex + log(bili), pbcseq)
```

plot.aareg *Plot an aareg object.*

Description

Plot the estimated coefficient function(s) from a fit of Aalen's additive regression model.

Usage

```
## S3 method for class 'aareg'
plot(x, se=TRUE, maxtime, type='s', ...)
```

Arguments

x	the result of a call to the aareg function
se	if TRUE, standard error bands are included on the plot
maxtime	upper limit for the x-axis.
type	graphical parameter for the type of line, default is "steps".
...	other graphical parameters such as line type, color, or axis labels.

Side Effects

A plot is produced on the current graphical device.

References

Aalen, O.O. (1989). A linear regression model for the analysis of life times. Statistics in Medicine, 8:907-925.

See Also

aareg

plot.cox.zph *Graphical Test of Proportional Hazards*

Description

Displays a graph of the scaled Schoenfeld residuals, along with a smooth curve.

Usage

```
## S3 method for class 'cox.zph'
plot(x, resid=TRUE, se=TRUE, df=4, nsmo=40, var,
        xlab="Time", ylab, lty=1:2, col=1, lwd=1, pch=1, cex=1,
    hr=FALSE, plot=TRUE, ...)
```

Arguments

x	result of the cox.zph function.
resid	a logical value, if TRUE the residuals are included on the plot, as well as the smooth fit.
se	a logical value, if TRUE, confidence bands at two standard errors will be added.
df	the degrees of freedom for the fitted natural spline, df=2 leads to a linear fit.
nsmo	number of points to use for the lines
var	the set of variables for which plots are desired. By default, plots are produced in turn for each variable of a model. Selection of a single variable allows other features to be added to the plot, e.g., a horizontal line at zero or a main title. This has been superseded by a subscripting method; see the example below.
hr	if TRUE, label the y-axis using the estimated hazard ratio rather than the estimated coefficient. (The plot does not change, only the axis label.)
xlab	label for the x-axis of the plot
ylab	optional label for the y-axis of the plot. If missing a default label is provided. This can be a vector of labels.
lty, col, lwd	line type, color, and line width for the overlaid curve. Each of these can be vector of length 2, in which case the second element is used for the confidence interval.
plot	if FALSE, return a list containing the x and y values of the curve, instead of drawing a plot
pch	used for points on the plot, see points
cex	used for points on the plot, see points
...	additional graphical arguments passed to the plot function.

Side Effects

a plot is produced on the current graphics device.

See Also

coxph, cox.zph.

Examples

```
vfit <- coxph(Surv(time,status) ~ trt + factor(celltype) +
              karno + age, data=veteran, x=TRUE)
temp <- cox.zph(vfit)
plot(temp, var=3)      # Look at Karnofsy score, old way of doing plot
plot(temp[3])     # New way with subscripting
abline(0, 0, lty=3)
# Add the linear fit as well
abline(lm(temp$y[,3] ~ temp$x)$coefficients, lty=4, col=3)
title(main="VA Lung Study")
```

plot.survfit *Plot method for* survfit *objects*

Description

A plot of survival curves is produced, one curve for each strata. The log=T option does extra work to avoid log(0), and to try to create a pleasing result. If there are zeros, they are plotted by default at 0.8 times the smallest non-zero value on the curve(s).

Curves are plotted in the same order as they are listed by print (which gives a 1 line summary of each). This will be the order in which col, lty, etc are used.

Usage

```
## S3 method for class 'survfit'
plot(x, conf.int=, mark.time=FALSE,
 pch=3, col=1, lty=1, lwd=1, cex=1, log=FALSE, xscale=1, yscale=1,
 xlim, ylim, xmax, fun,
 xlab="", ylab="", xaxs="r",  conf.times, conf.cap=.005,
 conf.offset=.012,
 conf.type = c("log", "log-log", "plain", "logit", "arcsin", "none"),
 mark, mark.col, noplot="(s0)", cumhaz=FALSE,
 firstx, ymin, cumprob=FALSE, ...)
```

Arguments

x	an object of class survfit, usually returned by the survfit function.
conf.int	if TRUE/FALSE, determines whether pointwise confidence intervals will be plotted. The default is to do so if there is only 1 curve, i.e., no strata, using 95% confidence intervals Alternatively, this can be a numeric value giving the desired confidence level.
mark.time	controls the labeling of the curves. If set to FALSE, no labeling is done. If TRUE, then curves are marked at each censoring time. If mark.time is a numeric vector then curves are marked at the specified time points.
pch	vector of characters which will be used to label the curves. The points help file contains examples of the possible marks. A single string such as "abcd" is treated as a vector c("a", "b", "c", "d"). The vector is reused cyclically if it is shorter than the number of curves. If it is present this implies mark.time = TRUE.
col	a vector of integers specifying colors for each curve. The default value is 1.
lty	a vector of integers specifying line types for each curve. The default value is 1.
lwd	a vector of numeric values for line widths. The default value is 1.
cex	a numeric value specifying the size of the marks. This is not treated as a vector; all marks have the same size.

`log` a logical value, if TRUE the y axis wll be on a log scale. Alternately, one of the standard character strings "x", "y", or "xy" can be given to specific logarithmic horizontal and/or vertical axes.

`xscale` a numeric value used like `yscale` for labels on the x axis. A value of 365.25 will give labels in years instead of the original days.

`yscale` a numeric value used to multiply the labels on the y axis. A value of 100, for instance, would be used to give a percent scale. Only the labels are changed, not the actual plot coordinates, so that adding a curve with "`lines(surv.exp(...))`", say, will perform as it did without the `yscale` argument.

`xlim`, `ylim` optional limits for the plotting region.

`xmax` the maximum horizontal plot coordinate. This can be used to shrink the range of a plot. It shortens the curve before plotting it, so unlike using the `xlim` graphical parameter, warning messages about out of bounds points are not generated. It also *looks* nicer when lines don't touch the right margin.

`fun` an arbitrary function defining a transformation of the survival (or probability in state, or cumulative hazard) curves. For example `fun=log` is an alternative way to draw a log-survival curve (but with the axis labeled with log(S) values), and `fun=sqrt` would generate a curve on square root scale. Four often used transformations can be specified with a character argument instead: `"S"` gives the usual survival curve, `"log"` is the same as using the `log=T` option, `"event"` or `"F"` plots the empirical CDF $F(t) = 1 - S(t)$ (f(y) = 1-y), and `"cloglog"` creates a complimentary log-log survival plot (f(y) = log(-log(y)) along with log scale for the x-axis). The terms `"identity"` and `"surv"` are allowed as synonyms for `type="S"`. The argument `"cumhaz"` causes the cumulative hazard function to be plotted.

`xlab` label given to the x-axis.

`ylab` label given to the y-axis.

`xaxs` either `"S"` for a survival curve or a standard x axis style as listed in `par`; "r" (regular) is the R default. Survival curves have historically been displayed with the curve touching the y-axis, but not touching the bounding box of the plot on the other 3 sides, Type `"S"` accomplishes this by manipulating the plot range and then using the `"i"` style internally. The "S" style is becoming increasingly less common, however.

`conf.times` optional vector of times at which to place a confidence bar on the curve(s). If present, these will be used instead of confidence bands.

`conf.cap` width of the horizontal cap on top of the confidence bars; only used if conf.times is used. A value of 1 is the width of the plot region.

`conf.offset` the offset for confidence bars, when there are multiple curves on the plot. A value of 1 is the width of the plot region. If this is a single number then each curve's bars are offset by this amount from the prior curve's bars, if it is a vector the values are used directly.

`conf.type` One of `"plain"`, `"log"` (the default), `"log-log"`, `"logit"`, or `"arcsin"` or `"none"`. Only enough of the string to uniquely identify it is necessary. The last option causes confidence intervals not to be generated. The plain option gives intervals `curve +- k *se(curve)`, where k is determined from `conf.int`. The

	log option calculates intervals based on the cumulative hazard or log(survival). The log-log option bases the intervals on the log hazard or log(-log(survival)), and the logit option on log(survival/(1-survival)) The default is "plain" for a plot of the cumulative hazard.
mark	a historical alias for pch (in S pch could only be character). This option is deprecated, use mark.time and pch
mark.col	color(s) for the marks, if different than the colors the curves
noplot	for multi-state models, curves with this label will not be plotted. (Also see the istate0 argument in survcheck.)
cumhaz	plot the cumulative hazard rather than the probability in state or survival. Optionally, this can be a numeric vector specifying which columns of the cumhaz component to plot.
ymin	this will normally be given as part of the ylim argument
firstx	this will normally be given as part of the xlim argument.
cumprob	for a multi-state curve, plot the probabilities in state 1, (state1 + state2), (state1 + state2 + state3), ... If cumprob is an integer vector the totals will be in the order indicated. This implies conf.type="none"
...	other arguments that will be passed forward to the underlying plot method, such as xlab or ylab.

Details

If the object contains a cumulative hazard curve, then fun='cumhaz' will plot that curve, otherwise it will plot -log(S) as an approximation. Theoretically, S = $\exp(-\Lambda)$ where S is the survival and Λ is the cumulative hazard. The same relationship holds for estimates of S and Λ only in special cases, but the approximation is often close.

When the survfit function creates a multi-state survival curve the resulting object also has class 'survfitms'. Competing risk curves are a common case. In this situation the fun argument is ignored.

When the conf.times argument is used, the confidence bars are offset by conf.offset units to avoid overlap. The bar on each curve are the confidence interval for the time point at which the bar is drawn, i.e., different time points for each curve. If curves are steep at that point, the visual impact can sometimes substantially differ for positive and negative values of conf.offset.

Value

a list with components x and y, containing the coordinates of the last point on each of the curves (but not the confidence limits). This may be useful for labeling. If cumprob=TRUE then y will be a matrix with one row per curve and x will be all the time points. This may be useful for adding shading.

Note

In prior versions the behavior of xscale and yscale differed: the first changed the scale both for the plot and for all subsequent actions such as adding a legend, whereas yscale affected only the axis label. This was normalized in version 2-36.4, and both parameters now only affect the labeling.

In versions prior to approximately 2.36 a survfit object did not contain the cumulative hazard as a separate result, and the use of fun="cumhaz" would plot the approximation -log(surv) to the cumulative hazard. When cumulative hazards were added to the object, the cumhaz=TRUE argument to the plotting function was added. In version 2.3-8 the use of fun="cumhaz" became a synonym for cumhaz=TRUE.

See Also

points.survfit, lines.survfit, par, survfit

Examples

```
leukemia.surv <- survfit(Surv(time, status) ~ x, data = aml)
plot(leukemia.surv, lty = 2:3)
legend(100, .9, c("Maintenance", "No Maintenance"), lty = 2:3)
title("Kaplan-Meier Curves\nfor AML Maintenance Study")
lsurv2 <- survfit(Surv(time, status) ~ x, aml, type='fleming')
plot(lsurv2, lty=2:3, fun="cumhaz",
xlab="Months", ylab="Cumulative Hazard")
```

---

predict.coxph *Predictions for a Cox model*

Description

Compute fitted values and regression terms for a model fitted by coxph

Usage

```
## S3 method for class 'coxph'
predict(object, newdata,
type=c("lp", "risk", "expected", "terms", "survival"),
se.fit=FALSE, na.action=na.pass, terms=names(object$assign), collapse,
reference=c("strata", "sample", "zero"),  ...)
```

Arguments

object	the results of a coxph fit.
newdata	Optional new data at which to do predictions. If absent, predictions are for the data frame used in the original fit. When coxph has been called with a formula argument created in another context, i.e., coxph has been called within another function and the formula was passed as an argument to that function, there can be problems finding the data set. See the note below.
type	the type of predicted value. Choices are the linear predictor ("lp"), the risk score exp(lp) ("risk"), the expected number of events given the covariates and follow-up time ("expected"), and the terms of the linear predictor ("terms"). The survival probability for a subject is equal to exp(-expected).

se.fit	if TRUE, pointwise standard errors are produced for the predictions.
na.action	applies only when the newdata argument is present, and defines the missing value action for the new data. The default is to include all observations. When there is no newdata, then the behavior of missing is dictated by the na.action option of the original fit.
terms	if type="terms", this argument can be used to specify which terms should be included; the default is all.
collapse	optional vector of subject identifiers. If specified, the output will contain one entry per subject rather than one entry per observation.
reference	reference for centering predictions, see details below
...	For future methods

Details

The Cox model is a *relative* risk model; predictions of type "linear predictor", "risk", and "terms" are all relative to the sample from which they came. By default, the reference value for each of these is the mean covariate within strata, with the exception of covariates with values of only -1, 0, or 1 which are not recentered. The underlying reason for subtracting a mean is both statistical and practical. First, a Cox model only predicts relative risks between pairs of subjects within the same strata, and hence the addition of a constant to any covariate, either overall or only within a particular stratum, has no effect on the fitted coefficients, log-likelihood, or standard errors. Second, downstream calculations depend on the risk score exp(linear predictor), which will fall prey to numeric overflow for a linear predictor greater than .Machine$double.max.exp. The coxph routines approximately center the predictors out of self protection. Using the reference="strata" option is the safest centering, since strata occasionally have different means. When the results of predict are used in further calculations it may be desirable to use a single reference level for all observations. Use of reference="sample" will use object$means, which agrees with the linear.predictors component of the coxph object. Predictions of type="terms" are almost invariably passed forward to further calculation, so for these we default to using the sample as the reference. A reference of "zero" causes no centering to be done.

Predictions of type "expected" or "survival" incorporate the baseline hazard and are thus absolute instead of relative; the reference option has no effect on these. These values depend on the follow-up time for the subjects as well as covariates so the newdata argument needs to include both the right and *left* hand side variables from the formula. (The status variable will not be used, but is required since the underlying code needs to reconstruct the entire formula.)

Models that contain a frailty term are a special case: due to the technical difficulty, when there is a newdata argument the predictions will always be for a random effect of zero.

For predictions of type expected a user will normally want to use $\Lambda(t_i)$, i.e., the cumulative hazard at the individual follow-up time $t_i$of each individual subject. This is E in the martingale residual O-E and plays a natural role in assessments of model validation (Crowson 2016). For predictions of type survival, on the other hand, a user will normally want S(tau), where tau is a single pre-specified time point which is the same for all subjects, e.g., predicted 5 year survival. The newdata data set should contain actual survival time(s) for each subject for the first case, as the survival time variable(s), and the target time tau in the second case; (0, tau) for (time1, time2) data.

For counting process data with (time1, time2) form, residuals of type expected estimate the increment in hazard over said interval, and those of type survival the probability of an event at time2

given that the observation is still alive at time1. The estimated hazards over two intervals (t1, t2) and (t2, t3) add to the total hazard over the interval (t1, t3), and the variances also add: the formulas treat these as independent increments, given the covariates. Estimated survivals multiply, but variances do not add.

Value

a vector or matrix of predictions, or a list containing the predictions (element "fit") and their standard errors (element "se.fit") if the se.fit option is TRUE.

Note

Some predictions can be obtained directly from the coxph object, and for others it is necessary for the routine to have the entirety of the original data set, e.g., for type = `terms` or if standard errors are requested. This extra information is saved in the coxph object if `model=TRUE`, if not the original data is reconstructed. If it is known that such residuals will be required overall execution will be slightly faster if the model information is saved.

In some cases the reconstruction can fail. The most common is when coxph has been called inside another function and the formula was passed as one of the arguments to that enclosing function. Another is when the data set has changed between the original call and the time of the prediction call. In each of these the simple solution is to add `model=TRUE` to the original coxph call.

References

C Crowson, E Atkinson and T Therneau, Assessing calibration of prognostic risk scores, Stat Methods Med Res, 2016.

See Also

predict,coxph,termplot

Examples

```
options(na.action=na.exclude) # retain NA in predictions
fit <- coxph(Surv(time, status) ~ age + ph.ecog + strata(inst), lung)
#lung data set has status coded as 1/2
mresid <- (lung$status-1) - predict(fit, type='expected') #Martingale resid
predict(fit,type="lp")
predict(fit,type="expected")
predict(fit,type="risk",se.fit=TRUE)
predict(fit,type="terms",se.fit=TRUE)

# For someone who demands reference='zero'
pzero <- function(fit)
  predict(fit, reference="sample") + sum(coef(fit) * fit$means, na.rm=TRUE)
```

predict.survreg *Predicted Values for a 'survreg' Object*

Description

Predicted values for a survreg object

Usage

```
## S3 method for class 'survreg'
predict(object, newdata,
type=c("response", "link", "lp", "linear", "terms", "quantile",
"uquantile"),
 se.fit=FALSE, terms=NULL, p=c(0.1, 0.9), na.action=na.pass, ...)
```

Arguments

object	result of a model fit using the survreg function.
newdata	data for prediction. If absent predictions are for the subjects used in the original fit.
type	the type of predicted value. This can be on the original scale of the data (response), the linear predictor ("linear", with "lp" as an allowed abbreviation), a predicted quantile on the original scale of the data ("quantile"), a quantile on the linear predictor scale ("uquantile"), or the matrix of terms for the linear predictor ("terms"). At this time "link" and linear predictor ("lp") are identical.
se.fit	if TRUE, include the standard errors of the prediction in the result.
terms	subset of terms. The default for residual type "terms" is a matrix with one column for every term (excluding the intercept) in the model.
p	vector of percentiles. This is used only for quantile predictions.
na.action	applies only when the newdata argument is present, and defines the missing value action for the new data. The default is to include all observations.
...	for future methods

Value

a vector or matrix of predicted values.

References

Escobar and Meeker (1992). Assessing influence in regression analysis with censored data. *Biometrics,* 48, 507-528.

See Also

survreg, residuals.survreg

Examples

```
# Draw figure 1 from Escobar and Meeker, 1992.
fit <- survreg(Surv(time,status) ~ age + I(age^2), data=stanford2,
dist='lognormal')
with(stanford2, plot(age, time, xlab='Age', ylab='Days',
xlim=c(0,65), ylim=c(.1, 10^5), log='y', type='n'))
with(stanford2, points(age, time, pch=c(2,4)[status+1], cex=.7))
pred <- predict(fit, newdata=list(age=1:65), type='quantile',
         p=c(.1, .5, .9))
matlines(1:65, pred, lty=c(2,1,2), col=1)

# Predicted Weibull survival curve for a lung cancer subject with
#  ECOG score of 2
lfit <- survreg(Surv(time, status) ~ ph.ecog, data=lung)
pct <- 1:98/100   # The 100th percentile of predicted survival is at +infinity
ptime <- predict(lfit, newdata=data.frame(ph.ecog=2), type='quantile',
                 p=pct, se=TRUE)
matplot(cbind(ptime$fit, ptime$fit + 2*ptime$se.fit,
                         ptime$fit - 2*ptime$se.fit)/30.5, 1-pct,
        xlab="Months", ylab="Survival", type='l', lty=c(1,2,2), col=1)
```

`print.aareg` *Print an aareg object*

Description

Print out a fit of Aalen's additive regression model

Usage

```
## S3 method for class 'aareg'
print(x, maxtime, test=c("aalen", "nrisk"),scale=1,...)
```

Arguments

`x`	the result of a call to the `aareg` function
`maxtime`	the upper time point to be used in the test for non-zero slope
`test`	the weighting to be used in the test for non-zero slope. The default weights are based on the variance of each coefficient, as a function of time. The alternative weight is proportional to the number of subjects still at risk at each time point.
`scale`	scales the coefficients. For some data sets, the coefficients of the Aalen model will be very small (10-4); this simply multiplies the printed values by a constant, say 1e6, to make the printout easier to read.
`...`	for future methods

Details

The estimated increments in the coefficient estimates can become quite unstable near the end of follow-up, due to the small number of observations still at risk in a data set. Thus, the test for slope will sometimes be more powerful if this last 'tail' is excluded.

Value

the calling argument is returned.

Side Effects

the results of the fit are displayed.

References

Aalen, O.O. (1989). A linear regression model for the analysis of life times. Statistics in Medicine, 8:907-925.

See Also

aareg

`print.summary.coxph` *Print method for summary.coxph objects*

Description

Produces a printed summary of a fitted coxph model

Usage

```
## S3 method for class 'summary.coxph'
print(x, digits=max(getOption("digits") - 3, 3),
  signif.stars = getOption("show.signif.stars"), expand=FALSE, ...)
## S3 method for class 'summary.coxph.penal'
print(x, digits=max(getOption("digits") - 3, 3),
  signif.stars = getOption("show.signif.stars"), maxlabel=25, ...)
```

Arguments

`x`	the result of a call to `summary.coxph`
`digits`	significant digits to print
`signif.stars`	Show stars to highlight small p-values
`expand`	if the summary is for a multi-state coxph fit, print the results in an expanded format
`maxlabel`	penalized terms can sometimes have very long default labels; this truncates them to better fit on the page
`...`	For future methods

print.summary.survexp *Print Survexp Summary*

Description

Prints the results of summary.survexp

Usage

```
## S3 method for class 'summary.survexp'
print(x, digits = max(options()$digits - 4, 3), ...)
```

Arguments

x	an object of class summary.survexp.
digits	the number of digits to use in printing the result.
...	for future methods

Value

x, with the invisible flag set to prevent further printing.

Author(s)

Terry Therneau

See Also

link{summary.survexp}, survexp

print.summary.survfit *Print Survfit Summary*

Description

Prints the result of summary.survfit.

Usage

```
## S3 method for class 'summary.survfit'
print(x, digits = max(options() $digits-4, 3), ...)
```

Arguments

x	an object of class "summary.survfit", which is the result of the summary.survfit function.
digits	the number of digits to use in printing the numbers.
...	for future methods

Value

x, with the invisible flag set to prevent printing.

Side Effects

prints the summary created by summary.survfit.

See Also

options, print, summary.survfit.

---

print.survfit *Print a Short Summary of a Survival Curve*

Description

Print number of observations, number of events, the restricted mean survival and its standard error, and the median survival with confidence limits for the median.

Usage

```
## S3 method for class 'survfit'
print(x, scale=1, digits = max(options()$digits - 4,3),
    print.rmean=getOption("survfit.print.rmean"),
    rmean = getOption('survfit.rmean'),...)
```

Arguments

x	the result of a call to the survfit function.
scale	a numeric value to rescale the survival time, e.g., if the input data to survfit were in days, scale=365 would scale the printout to years.
digits	Number of digits to print
print.rmean, rmean	Options for computation and display of the restricted mean.
...	for future results

Details

The mean and its variance are based on a truncated estimator. That is, if the last observation(s) is not a death, then the survival curve estimate does not go to zero and the mean is undefined. There are four possible approaches to resolve this, which are selected by the `rmean` option. The first is to set the upper limit to a constant, e.g.,`rmean=365`. In this case the reported mean would be the expected number of days, out of the first 365, that would be experienced by each group. This is useful if interest focuses on a fixed period. Other options are `"none"` (no estimate), `"common"` and `"individual"`. The `"common"` option uses the maximum time for all curves in the object as a common upper limit for the auc calculation. For the `"individual"`options the mean is computed as the area under each curve, over the range from 0 to the maximum observed time for that curve. Since the end point is random, values for different curves are not comparable and the printed standard errors are an underestimate as they do not take into account this random variation. This option is provided mainly for backwards compatability, as this estimate was the default (only) one in earlier releases of the code. Note that SAS (as of version 9.3) uses the integral up to the last *event* time of each individual curve; we consider this the worst of the choices and do not provide an option for that calculation.

The median and its confidence interval are defined by drawing a horizontal line at 0.5 on the plot of the survival curve and its confidence bands. If that line does not intersect the curve, then the median is undefined. The intersection of the line with the lower CI band defines the lower limit for the median's interval, and similarly for the upper band. If any of the intersections is not a point then we use the center of the intersection interval, e.g., if the survival curve were exactly equal to 0.5 over an interval. When data is uncensored this agrees with the usual definition of a median.

Value

x, with the invisible flag set to prevent printing. (The default for all print functions in R is to return the object passed to them; print.survfit complies with this pattern. If you want to capture these printed results for further processing, see the `table` component of `summary.survfit`.)

Side Effects

The number of observations, the number of events, the median survival with its confidence interval, and optionally the restricted mean survival (`rmean`) and its standard error, are printed. If there are multiple curves, there is one line of output for each.

References

Miller, Rupert G., Jr. (1981). *Survival Analysis.* New York:Wiley, p 71.

See Also

`summary.survfit`, `quantile.survfit`

pseudo *Pseudo values for survival.*

Description

Produce pseudo values from a survival curve.

Usage

```
pseudo(fit, times, type, collapse= TRUE, data.frame=FALSE, ...)
```

Arguments

fit	a survfit object, or one that inherits that class.
times	a vector of time points, at which to evaluate the pseudo values.
type	the type of value, either the probabilty in state pstate, the cumulative hazard cumhaz or the expected sojourn time in the state sojourn.
collapse	if the original survfit call had an id variable, return one residual per unique id
data.frame	if TRUE, return the data in "long" form as a data.frame with id, state (or transition), curve, time, residual and pseudo as variables.
...	other arguments to the residuals.survfit function, which does the majority of the work, e.g., weighted.

Details

This function computes pseudo values based on a first order Taylor series, also known as the "infinitesimal jackknife" (IJ) or "dfbeta" residuals. To be completely correct the results of this function could perhaps be called 'IJ pseudo values' or even pseudo psuedo-values to distinguish them from Andersen and Pohar-Perme. For moderate to large data, however, the result will be almost identical, numerically, to the ordinary jackknife psuedovalues.

A primary advantage of this approach is computational speed. Two other features, neither good nor bad, are that they will agree with robust standard errors of other survival package estimates, which are based on the IJ, and that the mean of the estimates, over subjects, is exactly the underlying survival estimate.

For the type variable, surv is an acceptable synonym for pstate, chaz for cumhaz, and rmst,rmts and auc are equivalent to sojourn. All of these are case insensitive.

If the orginal survfit call produced multiple curves, the internal computations are done separately for each curve. The result from this routine is the IJ (as computed by resid.survfit) scaled by n and then recentered. If the the survfit call included an id option, n is the number of unique id values, otherwise the number of rows in the data set. If the original survfit call used case weights, those weights are part of the IJ residuals, but are not used to compute the rescaling factor n.

IJ values are well defined for all variants of the Aalen-Johansen estimate; indeed, they are the basis for standard errors of the result. However, understanding properties of the pseudovalues is still evolving. Validity has been verified for the probability in state and sojourn times whenever all

subjects start in the same state; this includes for instance the usual Kaplan-Meier and competing risks cases. On the other hand, regression results based on pseudovalues from left-truncated data will be biased (Parner); and pseudo-values for the cumulative hazard have not been widely explored. When a given subject is spread across multiple (time1, time2) windows, e.g., a data set with a time-dependent covariate, the IJ values from a simple survival (without TD variables) will sum to the overall IJ for that subject; however, whether and how these partial pseudovalues can be directly used in a model is still uncertain. As understanding evolves, treat this routine's results as a reseach tool, not production, for these more complex cases.

Value

A vector, matrix, or array. The first dimension is always the number of observations in the data object, in the same order as the original data set (less any missing values that were removed when creating the survfit object); the second dimension, if applicable, corresponds to `fit$states`, e.g., multi-state survival, and the last dimension to the selected time points. (If there are multiple rows for a given id and `collapse=TRUE`, there is only one row per unique id.)

For the data.frame option, a data frame containing values for id, time, and pseudo. If the original `survfit` call contained an `id` statement, then the values in the `id` column will be taken from that variable. If the `id` statement has a simple form, e.g., `id = patno`, then the name of the id column will be 'patno', otherwise it will be named '(id)'.

Note

The code will be slightly faster if the `model=TRUE` option is used in the `survfit` call. It may be essential if the survfit/pseudo pair is used inside another function.

References

PK Andersen and M Pohar-Perme, Pseudo-observations in surivival analysis, Stat Methods Medical Res, 2010; 19:71-99

ET Parner, PK Andersen and M Overgaard, Regression models for censored time-to-event data using infinitesimal jack-knife pseudo-observations, with applications to left-truncation, Lifetime Data Analysis, 2023, 29:654-671

See Also

`residuals.survfit`

Examples

```
fit1 <- survfit(Surv(time, status) ~ 1, data=lung)
yhat <- pseudo(fit1, times=c(365, 730))
dim(yhat)
lfit <- lm(yhat[,1] ~ ph.ecog + age + sex, data=lung)

# Restricted Mean Time in State (RMST)
rms <- pseudo(fit1, times= 730, type='RMST') # 2 years
rfit <- lm(rms ~ ph.ecog + sex, data=lung)
rhat <- predict(rfit, newdata=expand.grid(ph.ecog=0:3, sex=1:2), se.fit=TRUE)
# print it out nicely
```

```
temp1 <- cbind(matrix(rhat$fit, 4,2))
temp2 <- cbind(matrix(rhat$se.fit, 4, 2))
temp3 <- cbind(temp1[,1], temp2[,1], temp1[,2], temp2[,2])
dimnames(temp3) <- list(paste("ph.ecog", 0:3),
                        c("Male RMST", "(se)", "Female RMST", "(se)"))

round(temp3, 1)
# compare this to the fully non-parametric estimate
fit2 <- survfit(Surv(time, status) ~ ph.ecog, data=lung)
print(fit2, rmean=730)
# the estimate for ph.ecog=3 is very unstable (n=1), pseudovalues smooth it.
#
# In all the above we should be using the robust variance, e.g., svyglm, but
#  a recommended package can't depend on external libraries.
# See the vignette for a more complete exposition.
```

`pspline` *Smoothing splines using a pspline basis*

Description

Specifies a penalised spline basis for the predictor. This is done by fitting a comparatively small set of splines and penalising the integrated second derivative. Traditional smoothing splines use one basis per observation, but several authors have pointed out that the final results of the fit are indistinguishable for any number of basis functions greater than about 2-3 times the degrees of freedom. Eilers and Marx point out that if the basis functions are evenly spaced, this leads to significant computational simplification, they refer to the result as a p-spline.

Usage

```
pspline(x, df=4, theta, nterm=2.5 * df, degree=3, eps=0.1, method,
   Boundary.knots=range(x), intercept=FALSE, penalty=TRUE, combine, ...)

psplineinverse(x)
```

Arguments

`x`	for psline: a covariate vector. The function does not apply to factor variables. For psplineinverse x will be the result of a pspline call.
`df`	the desired degrees of freedom. One of the arguments `df` or `theta`' must be given, but not both. If `df=0`, then the AIC = (loglik -df) is used to choose an "optimal" degrees of freedom. If AIC is chosen, then an optional argument 'caic=T' can be used to specify the corrected AIC of Hurvich et. al.
`theta`	roughness penalty for the fit. It is a monotone function of the degrees of freedom, with theta=1 corresponding to a linear fit and theta=0 to an unconstrained fit of nterm degrees of freedom.
`nterm`	number of splines in the basis

degree	degree of splines
eps	accuracy for df
method	the method for choosing the tuning parameter theta. If theta is given, then 'fixed' is assumed. If the degrees of freedom is given, then 'df' is assumed. If method='aic' then the degrees of freedom is chosen automatically using Akaike's information criterion.
...	optional arguments to the control function
Boundary.knots	the spline is linear beyond the boundary knots. These default to the range of the data.
intercept	if TRUE, the basis functions include the intercept.
penalty	if FALSE a large number of attributes having to do with penalized fits are excluded. This is useful to create a pspline basis matrix for other uses.
combine	an optional vector of increasing integers. If two adjacent values of combine are equal, then the corresponding coefficients of the fit are forced to be equal. This is useful for monotone fits, see the vignette for more details.

Value

Object of class pspline, coxph.penalty containing the spline basis, with the appropriate attributes to be recognized as a penalized term by the coxph or survreg functions.

For psplineinverse the original x vector is reconstructed.

References

Eilers, Paul H. and Marx, Brian D. (1996). Flexible smoothing with B-splines and penalties. Statistical Science, 11, 89-121.

Hurvich, C.M. and Simonoff, J.S. and Tsai, Chih-Ling (1998). Smoothing parameter selection in nonparametric regression using an improved Akaike information criterion, JRSSB, volume 60, 271–293.

See Also

coxph,survreg,ridge, frailty

Examples

```
lfit6 <- survreg(Surv(time, status)~pspline(age, df=2), lung)
plot(lung$age, predict(lfit6), xlab='Age', ylab="Spline prediction")
title("Cancer Data")
fit0 <- coxph(Surv(time, status) ~ ph.ecog + age, lung)
fit1 <- coxph(Surv(time, status) ~ ph.ecog + pspline(age,3), lung)
fit3 <- coxph(Surv(time, status) ~ ph.ecog + pspline(age,8), lung)
fit0
fit1
fit3
```

pyears *Person Years*

Description

This function computes the person-years of follow-up time contributed by a cohort of subjects, stratified into subgroups. It also computes the number of subjects who contribute to each cell of the output table, and optionally the number of events and/or expected number of events in each cell.

Usage

```
pyears(formula, data, weights, subset, na.action,  rmap,
       ratetable, scale=365.25, expect=c('event', 'pyears'),
       model=FALSE, x=FALSE, y=FALSE, data.frame=FALSE)
```

Arguments

formula	a formula object. The response variable will be a vector of follow-up times for each subject, or a Surv object containing the survival time and an event indicator. The predictors consist of optional grouping variables separated by + operators (exactly as in survfit), time-dependent grouping variables such as age (specified with tcut), and optionally a ratetable term. This latter matches each subject to his/her expected cohort.
data	a data frame in which to interpret the variables named in the formula, or in the subset and the weights argument.
weights	case weights.
subset	expression saying that only a subset of the rows of the data should be used in the fit.
na.action	a missing-data filter function, applied to the model.frame, after any subset argument has been used. Default is options()$na.action.
rmap	an optional list that maps data set names to the ratetable names. See the details section below.
ratetable	a table of event rates, such as survexp.uswhite.
scale	a scaling for the results. As most rate tables are in units/day, the default value of 365.25 causes the output to be reported in years.
expect	should the output table include the expected number of events, or the expected number of person-years of observation. This is only valid with a rate table.
data.frame	return a data frame rather than a set of arrays.
model, x, y	If any of these is true, then the model frame, the model matrix, and/or the vector of response times will be returned as components of the final result.

Details

Because pyears may have several time variables, it is necessary that all of them be in the same units. For instance, in the call

```
py <- pyears(futime ~ rx, rmap=list(age=age, sex=sex, year=entry.dt),
                  ratetable=survexp.us)
```

the natural unit of the ratetable is hazard per day, it is important that futime, age and entry.dt all be in days. Given the wide range of possible inputs, it is difficult for the routine to do sanity checks of this aspect.

The ratetable being used may have different variable names than the user's data set, this is dealt with by the rmap argument. The rate table for the above calculation was survexp.us, a call to summary{survexp.us} reveals that it expects to have variables age = age in days, sex, and year = the date of study entry, we create them in the rmap line. The sex variable is not mapped, therefore the code assumes that it exists in mydata in the correct format. (Note: for factors such as sex, the program will match on any unique abbreviation, ignoring case.)

A special function tcut is needed to specify time-dependent cutpoints. For instance, assume that age is in years, and that the desired final arrays have as one of their margins the age groups 0-2, 2-10, 10-25, and 25+. A subject who enters the study at age 4 and remains under observation for 10 years will contribute follow-up time to both the 2-10 and 10-25 subsets. If cut(age, c(0,2,10,25,100)) were used in the formula, the subject would be classified according to his starting age only. The tcut function has the same arguments as cut, but produces a different output object which allows the pyears function to correctly track the subject.

The results of pyears are normally used as input to further calculations. The print routine, therefore, is designed to give only a summary of the table.

Value

a list with components:

pyears	an array containing the person-years of exposure. (Or other units, depending on the rate table and the scale). The dimension and dimnames of the array correspond to the variables on the right hand side of the model equation.
n	an array containing the number of subjects who contribute time to each cell of the pyears array.
event	an array containing the observed number of events. This will be present only if the response variable is a Surv object.
expected	an array containing the expected number of events (or person years if expect ="pyears"). This will be present only if there was a ratetable term.
data	if the data.frame option was set, a data frame containing the variables n, event, pyears and event that supplants the four arrays listed above, along with variables corresponding to each dimension. There will be one row for each cell in the arrays.
offtable	the number of person-years of exposure in the cohort that was not part of any cell in the pyears array. This is often useful as an error check; if there is a mismatch of units between two variables, nearly all the person years may be off table.

tcut	whether the call included any time-dependent cutpoints.
summary	a summary of the rate-table matching. This is also useful as an error check.
call	an image of the call to the function.
observations	the number of observations in the input data set, after any missings were removed.
na.action	the na.action attribute contributed by an na.action routine, if any.

See Also

ratetable, survexp, Surv.

Examples

```
# Look at progression rates jointly by calendar date and age
#
temp.yr  <- tcut(mgus$dxyr, 55:92, labels=as.character(55:91))
temp.age <- tcut(mgus$age, 34:101, labels=as.character(34:100))
ptime <- ifelse(is.na(mgus$pctime), mgus$futime, mgus$pctime)
pstat <- ifelse(is.na(mgus$pctime), 0, 1)
pfit <- pyears(Surv(ptime/365.25, pstat) ~ temp.yr + temp.age + sex,  mgus,
     data.frame=TRUE)
# Turn the factor back into numerics for regression
tdata <- pfit$data
tdata$age <- as.numeric(as.character(tdata$temp.age))
tdata$year<- as.numeric(as.character(tdata$temp.yr))
fit1 <- glm(event ~ year + age+ sex +offset(log(pyears)),
             data=tdata, family=poisson)
## Not run:
# fit a gam model
gfit.m <- gam(y ~ s(age) + s(year) + offset(log(time)),
                       family = poisson, data = tdata)

## End(Not run)

# Example #2  Create the hearta data frame:
hearta <- by(heart, heart$id,
             function(x)x[x$stop == max(x$stop),])
hearta <- do.call("rbind", hearta)
# Produce pyears table of death rates on the surgical arm
#  The first is by age at randomization, the second by current age
fit1 <- pyears(Surv(stop/365.25, event) ~ cut(age + 48, c(0,50,60,70,100)) +
       surgery, data = hearta, scale = 1)
fit2 <- pyears(Surv(stop/365.25, event) ~ tcut(age + 48, c(0,50,60,70,100)) +
       surgery, data = hearta, scale = 1)
fit1$event/fit1$pyears  #death rates on the surgery and non-surg arm

fit2$event/fit2$pyears  #death rates on the surgery and non-surg arm
```

quantile.survfit *Quantiles from a survfit object*

Description

Retrieve quantiles and confidence intervals for them from a survfit or Surv object.

Usage

```
## S3 method for class 'survfit'
quantile(x, probs = c(0.25, 0.5, 0.75), conf.int = TRUE,
  scale, tolerance= sqrt(.Machine$double.eps), ...)
## S3 method for class 'survfitms'
quantile(x, probs = c(0.25, 0.5, 0.75), conf.int = TRUE,
  scale, tolerance= sqrt(.Machine$double.eps), ...)
## S3 method for class 'survfit'
median(x, ...)
```

Arguments

x	a result of the survfit function
probs	numeric vector of probabilities with values in [0,1]
conf.int	should lower and upper confidence limits be returned?
scale	optional scale factor, e.g., scale=365.25 would return results in years if the fit object were in days.
tolerance	tolerance for checking that the survival curve exactly equals one of the quantiles
...	optional arguments for other methods

Details

The kth quantile for a survival curve S(t) is the location at which a horizontal line at height p= 1-k intersects the plot of S(t). Since S(t) is a step function, it is possible for the curve to have a horizontal segment at exactly 1-k, in which case the midpoint of the horizontal segment is returned. This mirrors the standard behavior of the median when data is uncensored. If the survival curve does not fall to 1-k, then that quantile is undefined.

In order to be consistent with other quantile functions, the argument prob of this function applies to the cumulative distribution function F(t) = 1-S(t).

Confidence limits for the values are based on the intersection of the horizontal line at 1-k with the upper and lower limits for the survival curve. Hence confidence limits use the same p-value as was in effect when the curve was created, and will differ depending on the conf.type option of survfit. If the survival curves have no confidence bands, confidence limits for the quantiles are not available.

When a horizontal segment of the survival curve exactly matches one of the requested quantiles the returned value will be the midpoint of the horizontal segment; this agrees with the usual definition of a median for uncensored data. Since the survival curve is computed as a series of products,

however, there may be round off error. Assume for instance a sample of size 20 with no tied times and no censoring. The survival curve after the 10th death is (19/20)(18/19)(17/18) ... (10/11) = 10/20, but the computed result will not be exactly 0.5. Any horizontal segment whose absolute difference with a requested percentile is less than `tolerance` is considered to be an exact match.

Value

The quantiles will be a vector if the `survfit` object contains only a single curve, otherwise it will be a matrix or array. In this case the last dimension will index the quantiles.

If confidence limits are requested, then result will be a list with components `quantile`, `lower`, and `upper`, otherwise it is the vector or matrix of quantiles.

Author(s)

Terry Therneau

See Also

`survfit`, `print.survfit`, `qsurvreg`

Examples

```
fit <- survfit(Surv(time, status) ~ ph.ecog, data=lung)
quantile(fit)

cfit <- coxph(Surv(time, status) ~ age + strata(ph.ecog), data=lung)
csurv<- survfit(cfit, newdata=data.frame(age=c(40, 60, 80)),
                  conf.type ="none")
temp <- quantile(csurv, 1:5/10)
temp[2,3,]  # quantiles for second level of ph.ecog, age=80
quantile(csurv[2,3], 1:5/10)  # quantiles of a single curve, same result
```

---

ratetable *Allow ratetable() terms in a model*

Description

This function supports ratetable() terms in a model statement, within survexp and pyears.

Usage

```
ratetable(...)
```

Arguments

`...`	the named dimensions of a rate table

Details

This way of mapping a rate table's variable names to a user data frame has been superseded, instead use the `rmap` argument of the survexp, pyears, or survdiff routines. The function remains only to allow older code to be run.

Author(s)

Terry Therneau

`ratetableDate` *Convert date objects to ratetable form*

Description

This method converts dates from various forms into the internal form used in `ratetable` objects.

Usage

```
ratetableDate(x)
```

Arguments

`x`	a date. The function currently has methods for Date, date, POSIXt, timeDate, and chron objects.

Details

This function is useful for those who create new ratetables, but is normally invisible to users. It is used internally by the `survexp` and `pyears` functions to map the various date formats; if a new method is added then those routines will automatically be adapted to the new date type.

Value

a numeric vector, the number of days since 1/1/1960.

Author(s)

Terry Therneau

See Also

`pyears`, `survexp`

`ratetables` *Census Data Sets for the Expected Survival and Person Years Functions*

Description

Census data sets for the expected survival and person years functions.

Usage

```
data(survexp, package="survival")
```

Details

survexp.us total United States population, by age and sex, 1940 to 2012.

survexp.usr United States population, by age, sex and race, 1940 to 2014. Race is white or black. For 1960 and 1970 the black population values were not reported separately, so the nonwhite values were used. (Over the years, the reported tables have differed wrt reporting non-white and/or black.)

survexp.mn total Minnesota population, by age and sex, 1970 to 2013.

Each of these tables contains the daily hazard rate for a matched subject from the population, defined as $-\log(1-q)/365.25$ where q is the 1 year probability of death as reported in the original tables from the US Census. For age 25 in 1970, for instance, $p = 1 - q$ is is the probability that a subject who becomes 25 years of age in 1970 will achieve his/her 26th birthday. The tables are recast in terms of hazard per day for computational convenience.

Each table is stored as an array, with additional attributes, and can be subset and manipulated as standard R arrays. See the help page for `ratetable` for details.

All numeric dimensions of a rate table must be in the same units. The `survexp.us` rate table contains daily hazard rates, the age cutpoints are in days, and the calendar year cutpoints are a Date.

See Also

`ratetable`, `survexp`, `pyears`

Examples

```
survexp.uswhite <- survexp.usr[,,"white",]
```

rats *Rat treatment data from Mantel et al*

Description

Rat treatment data from Mantel et al. Three rats were chosen from each of 100 litters, one of which was treated with a drug, and then all followed for tumor incidence.

Usage

```
rats
data(cancer, package="survival")
```

Format

litter:	litter number from 1 to 100
rx:	treatment,(1=drug, 0=control)
time:	time to tumor or last follow-up
status:	event status, 1=tumor and 0=censored
sex:	male or female

Note

Since only 2/150 of the male rats have a tumor, most analyses use only females (odd numbered litters), e.g. Lee et al.

Source

N. Mantel, N. R. Bohidar and J. L. Ciminera. Mantel-Haenszel analyses of litter-matched time to response data, with modifications for recovery of interlitter information. Cancer Research, 37:3863-3868, 1977.

References

E. W. Lee, L. J. Wei, and D. Amato, Cox-type regression analysis for large number of small groups of correlated failure time observations, in "Survival Analysis, State of the Art", Kluwer, 1992.

rats2 *Rat data from Gail et al.*

Description

48 rats were injected with a carcinogen, and then randomized to either drug or placebo. The number of tumors ranges from 0 to 13; all rats were censored at 6 months after randomization.

Usage

```
rats2
data(cancer, package="survival")
```

Format

rat:	id
trt:	treatment,(1=drug, 0=control)
observation:	within rat
start:	entry time
stop:	exit time
status:	event status, 1=tumor, 0=censored

Source

MH Gail, TJ Santner, and CC Brown (1980), An analysis of comparative carcinogenesis experiments based on multiple times to tumor. *Biometrics* **36**, 255–266.

reliability *Reliability data sets*

Description

A set of data for simple reliablility analyses, taken from the book by Meeker and Escobar.

Usage

```
data(reliability, package="survival")
```

Details

- braking: Locomotive age at the time of replacement of braking grids, 1-4 replacements for each locomotive. The grids are part of two manufacturing batches.
- capacitor: Data from a factorial experiment on the life of glass capacitors as a function of voltage and operating temperature. There were 8 capacitors at each combination of temperature and voltage. Testing at each combination was terminated after the fourth failure.
  - temperature: temperature in degrees celcius
  - voltage: applied voltage
  - time: time to failure
  - status: 1=failed, 0=censored
- cracks: Data on the time until the development of cracks in a set of 167 identical turbine parts. The parts were inspected at 8 selected times.
  - day: time of inspection
  - fail: number of fans found to have cracks, at this inspection
- Data set genfan: Time to failure of 70 diesel engine fans.
  - hours: hours of service
  - status: 1=failure, 0=censored

  Data set ifluid: A data frame with two variables describing the time to electrical breakdown of an insulating fluid.
  - time: hours to breakdown
  - voltage: test voltage in kV
- Data set imotor: Breakdown of motor insulation as a function of temperature.
  - temp: temperature of the test
  - time: time to failure or censoring
  - status: 0=censored, 1=failed
- Data set turbine: Each of 432 turbine wheels was inspected once to determine whether a crack had developed in the wheel or not.
  - hours: time of inspection (100s of hours)
  - inspected: number that were inspected
  - failed: number that failed
- Data set valveSeat: Time to replacement of valve seats for 41 diesel engines. More than one seat may be replaced at a particular service, leading to duplicate times in the data set. The final inspection time for each engine will have status=0.
  - id: engine identifier
  - time: time of the inspection, in days
  - status: 1=replacement occured, 0= not

References

Meeker and Escobar, Statistical Methods for Reliability Data, 1998.

Examples

```
survreg(Surv(time, status) ~ temperature + voltage, capacitor)

# Figure 16.7 of Meeker, cumulative replacement of locomotive braking
#  grids
gfit <- survfit(Surv(day1, day2, status) ~ batch, braking, id= locomotive)
plot(gfit, cumhaz=TRUE, col=1:2, xscale=30.5, conf.time= c(6,12,18)*30.5,
     xlab="Locomotive Age in Months",
     ylab="Mean cumulative number of replacements")

# Replacement of valve seats.  In this case the cumulative hazard is the
#  natural target, an estimate of the number of replacements by a given time
#  (known as the cumulative mean function = CMF in relability).
# When two valve seats failed at the same inspection, we need to jitter one
#  of the times, to avoid a (time1, time2) interval of length 0
ties <- which(with(valveSeat, diff(id)==0 & diff(time)==0))  #first of a tie
temp <- valveSeat$time
temp[ties] <- temp[ties] - .1 # jittered time
vdata <- valveSeat
vdata$time1 <- ifelse(!duplicated(vdata$id), 0, c(0, temp[-length(temp)]))
vdata$time2 <- temp
fit2 <- survfit(Surv(time1, time2, status) ~1, vdata, id=id)
## Not run:
plot(fit2, cumhaz= TRUE, xscale= 365.25,
      xlab="Years in service", ylab = "Expected number of repairs")

## End(Not run)
```

`residuals.coxph` *Calculate Residuals for a 'coxph' Fit*

Description

Calculates martingale, deviance, score or Schoenfeld residuals for a Cox proportional hazards model.

Usage

```
## S3 method for class 'coxph'
residuals(object,
       type=c("martingale", "deviance", "score", "schoenfeld",
      "dfbeta", "dfbetas", "scaledsch","partial"),
       collapse=FALSE, weighted= (type %in% c("dfbeta", "dfbetas")),
       na.action, ...)
## S3 method for class 'coxphms'
residuals(object,
       type=c("martingale", "score", "schoenfeld",
      "dfbeta", "dfbetas", "scaledsch"),
       collapse=FALSE,  weighted= (type %in% c("dfbeta", "dfbetas")),
```

```
       na.action, ...)
## S3 method for class 'coxph.null'
residuals(object,
       type=c("martingale", "deviance","score","schoenfeld"),
       collapse=FALSE, weighted= FALSE, ...)
```

Arguments

object	an object inheriting from class coxph, representing a fitted Cox regression model. Typically this is the output from the coxph function.
type	character string indicating the type of residual desired. Possible values are "martingale", "deviance", "score", "schoenfeld", "dfbeta"', "dfbetas", "scaledsch" and "partial". Only enough of the string to determine a unique match is required.
collapse	either a logical (TRUE/FALSE), or a vector indicating which rows to collapse (sum) over. In time-dependent models more than one row data can pertain to a single individual or group.
weighted	if TRUE and the model was fit with case weights, then the weighted residuals are returned.
na.action	either "na.omit" or "na.exclude", the default is the global option in effect when the coxph model was fit. The latter option causes the residuals to be the same size as the original data *before* missings were excluded by coxph, it is useful when plotting residuals versus a variable in the data frame, for instance. Not applicable to Schoenfeld or scaled Schoenfeld residuals, which have a row per event, nor when collapse=TRUE.
...	other unused arguments

Value

a vector or matrix of residuals

Details

If collapse=FALSE then the residuals will be a matrix with one row for each row in the data in the same order as the data, or for simple martingale residuals one element per row of the data. Otherwise, the result will have a row for each unique value of the collapsing vector, as provided by the cluster or id argument in the coxph call, or the collapse argument itself. If the collasping vector is a factor, the result will be in the order of the levels of the factor, otherwise the order in which the unique values are encountered.

The martingale residuals are a vector for each observation, or for a multi-state model they will be a matrix with one row per observation and one column per transition. If a particular observation was not at risk for a given transition, that matrix value will be zero. Deviance residuals are a transform of the martingale residuals, one which has not proven to be useful. They have not been defined for multistate models.

The score residual is each observation's contribution to the score vector (first derivative of the log partial likelihood), and will be a matrix with one row per observation and one column per covariate. Two transformations of this are often more useful: dfbeta is the approximate change

in the coefficient vector if that observation were dropped, and `dfbetas` is the approximate change in the coefficients, scaled by the standard error for the coefficients. For multi-state models the coefficient names are a concatonation of the variable name and the transition, e.g., `male_1:2`; some variables might not be used for all transitions. If a particular observation were not at risk for a 1:2 transition, then the residual value for `male_1:2` would be 0 for that observation.

The Schoenfeld residuals have a row for each event time and a column for each covariate. If the coxph model was stratified, the resulting matrix will contain a `strata` attribute showing which stratum contributed each of the events, and the result will be sorted by time within strata. For multistate models the matrix will have the events for transition 1, then transition 2, etc. and will also have a `transition` attribute. Different transitions will have different event times, so an array form is not practical for Schoenfeld residuals.

References

T. Therneau, P. Grambsch, and T. Fleming. "Martingale based residuals for survival models", *Biometrika*, March 1990.

See Also

coxph, predict.coxph

Examples

```
fit <- coxph(Surv(start, stop, event) ~ (age + surgery)* transplant,
             data=heart)
mresid <- resid(fit, collapse=heart$id)
```

---

residuals.survfit *IJ residuals from a survfit object.*

Description

Return infinitesimal jackknife residuals from a survfit object, for the survival, cumulative hazard, or restricted mean time in state (RMTS).

Usage

```
## S3 method for class 'survfit'
residuals(object, times,
    type="pstate", collapse=FALSE, weighted= collapse, data.frame=FALSE,
    extra = FALSE, ...)
```

Arguments

object	a survfit object
times	a vector of times at which the residuals are desired
type	the type of residual, see below
collapse	add the residuals for all subjects in a cluster
weighted	weight the residuals by each observation's weight
data.frame	if FALSE return a matrix or array
extra	return extra information when data.frame=FALSE. (This is used internally by the psuedo function.)
...	arguments for other methods

Details

This function is designed to efficiently compute the per-observation residuals for a Kaplan-Meier or Aalen-Johansen curve, also known as infinitesimal jackknife (IJ) values, at a small number of time points. Common usages are the creation of psuedo-values (via the pseudo function) and IJ estimates of variance. The residuals matrix has a value for each observation and time point pair. For a multi-state model the state will be a third dimension.

The residuals are the impact of each observation or cluster on the resulting probability in state curves at the given time points, the cumulative hazard curve at those time points, or the expected sojourn time in each state up to the given time points. For a simple Kaplan-Meier the survfit object contains only the probability in the "initial" state, i.e., the survival fraction. In this case the sojourn time, the expected amount of time spent in the initial state, up to the specified endpoint, is commonly known as the restricted mean survival time (RMST). For a multistate model this same quantity is more often referred to as the restricted mean time in state (RMTS). It can be computed as the area under the respective probability in state curve.

The program allows any of pstate, surv, cumhaz, chaz, sojourn, rmst, rmts or auc for the type argument, ignoring upper/lowercase, so users can choose whichever abbreviation they like best.

When collapse=TRUE the result has the cluster identifier (which defaults to the id variable) as the dimname for the first dimension. If the fit object contains more than one curve, and the same identifier is reused in two different curves this approach does not work and the routine will stop with an error. In principle this is not necessary, e.g., the result could contain two rows with the same label, showing the separate effect on each curve, but this was deemed too confusing.

Value

A matrix or array with one row per observation or cluster, and one column for each value in times. For a multi-state model the three dimensions are observation, state, and time. For cumulative hazard, the second dimension is the set of transitions. (A competing risks model for instance has 3 states and 2 transitions.)

Note

The first column of the data frame identifies the origin of the row. If there was an id variable in the survfit call it will contain the values of that variable and be labeled with the variable

name, or "(id)" if there was an expression rather than a name. (For example, `survfit(.... id=abc$def[z])`). If there was no `id` variable the label will be "(row)", and the column will contain the row number of the survfit data. For a matrix result the first component of dimnames has similar structure.

See Also

`survfit`, `survfit.formula`

Examples

```
fit <- survfit(Surv(time, status) ~ x, aml)
resid(fit, times=c(24, 48), type="RMTS")
```

`residuals.survreg` *Compute Residuals for 'survreg' Objects*

Description

This is a method for the function `residuals` for objects inheriting from class `survreg`.

Usage

```
## S3 method for class 'survreg'
residuals(object, type=c("response", "deviance","dfbeta","dfbetas",
"working","ldcase","ldresp","ldshape", "matrix"), rsigma=TRUE,
collapse=FALSE, weighted=FALSE, ...)
```

Arguments

`object`	an object inheriting from class `survreg`.
`type`	type of residuals, with choices of `"response"`, `"deviance"`, `"dfbeta"`, `"dfbetas"`, `"working"`, `"ldcase"`, `"lsresp"`, `"ldshape"`, and `"matrix"`.
`rsigma`	include the scale parameters in the variance matrix, when doing computations. (I can think of no good reason not to).
`collapse`	optional vector of subject groups. If given, this must be of the same length as the residuals, and causes the result to be per group residuals.
`weighted`	give weighted residuals? Normally residuals are unweighted.
`...`	other unused arguments

Value

A vector or matrix of residuals is returned. Response residuals are on the scale of the original data, working residuals are on the scale of the linear predictor, and deviance residuals are on log-likelihood scale. The dfbeta residuals are a matrix, where the ith row gives the approximate change in the coefficients due to the addition of subject i. The dfbetas matrix contains the dfbeta residuals, with each column scaled by the standard deviation of that coefficient.

The matrix type produces a matrix based on derivatives of the log-likelihood function. Let L be the log-likelihood, p be the linear predictor $X\beta$, and s be $\log(\sigma)$. Then the 6 columns of the matrix are L, dL/dp,$\partial^2 L/\partial p^2$, dL/ds, $\partial^2 L/\partial s^2$ and $\partial^2 L/\partial p \partial s$. Diagnostics based on these quantities are discussed in the book and article by Escobar and Meeker. The main ones are the likelihood displacement residuals for perturbation of a case weight (`ldcase`), the response value (`ldresp`), and the `shape`.

For a transformed distribution such as the log-normal or Weibull, matrix residuals are based on the log-likelihood of the transformed data log(y). For a monotone function f the density of f(X) is the density of X divided by the derivative of f (the Jacobian), so subtract log(derivative) from each uncensored observation's loglik value in order to match the `loglik` component of the result. The other colums of the matrix residual are unchanged by the transformation.

References

Escobar, L. A. and Meeker, W. Q. (1992). Assessing influence in regression analysis with censored data. *Biometrics* **48**, 507-528.

Escobar, L. A. and Meeker, W. Q. (1998). Statistical Methods for Reliablilty Data. Wiley.

See Also

`predict.survreg`

Examples

```
fit <- survreg(Surv(futime, death) ~ age + sex, mgus2)
summary(fit)   # age and sex are both important

rr  <- residuals(fit, type='matrix')
sum(rr[,1]) - with(mgus2, sum(log(futime[death==1]))) # loglik

plot(mgus2$age, rr[,2], col= (1+mgus2$death)) # ldresp
```

---

retinopathy *Diabetic Retinopathy*

Description

A trial of laser coagulation as a treatment to delay diabetic retinopathy.

Usage

```
retinopathy
data(retinopathy, package="survival")
```

Format

A data frame with 394 observations on the following 9 variables.

`id` numeric subject id

`laser` type of laser used: `xenon` `argon`

`eye` which eye was treated: `right` `left`

`age` age at diagnosis of diabetes

`type` type of diabetes: `juvenile` `adult`, (diagnosis before age 20)

`trt` 0 = control eye, 1 = treated eye

`futime` time to loss of vision or last follow-up

`status` 0 = censored, 1 = loss of vision in this eye

`risk` a risk score for the eye. This high risk subset is defined as a score of 6 or greater in at least one eye.

Details

The 197 patients in this dataset were a 50% random sample of the patients with "high-risk" diabetic retinopathy as defined by the Diabetic Retinopathy Study (DRS). Each patient had one eye randomized to laser treatment and the other eye received no treatment, and has two observations in the data set. For each eye, the event of interest was the time from initiation of treatment to the time when visual acuity dropped below 5/200 two visits in a row. Thus there is a built-in lag time of approximately 6 months (visits were every 3 months). Survival times in this dataset are the actual time to vision loss in months, minus the minimum possible time to event (6.5 months). Censoring was caused by death, dropout, or end of the study.

References

W. J. Huster, R. Brookmeyer and S. G. Self (1989). Modelling paired survival data with covariates, Biometrics 45:145-156.

A. L. Blair, D. R. Hadden, J. A. Weaver, D. B. Archer, P. B. Johnston and C. J. Maguire (1976). The 5-year prognosis for vision in diabetes, American Journal of Ophthalmology, 81:383-396.

Examples

```
coxph(Surv(futime, status) ~ type + trt, cluster= id, retinopathy)
```

rhDNase *rhDNASE data set*

Description

Results of a randomized trial of rhDNase for the treatment of cystic fibrosis.

Usage

```
rhDNase
data(rhDNase, package="survival")
```

Format

A data frame with 767 observations on the following 8 variables.

`id` subject id

`inst` enrolling institution

`trt` treatment arm: 0=placebo, 1= rhDNase

`entry.dt` date of entry into the study

`end.dt` date of last follow-up

`fev` forced expriatory volume at enrollment, a measure of lung capacity

`ivstart` days from enrollment to the start of IV antibiotics

`ivstop` days from enrollment to the cessation of IV antibiotics

Details

In patients with cystic fibrosis, extracellular DNA is released by leukocytes that accumulate in the airways in response to chronic bacterial infection. This excess DNA thickens the mucus, which then cannot be cleared from the lung by the cilia. The accumulation leads to exacerbations of respiratory symptoms and progressive deterioration of lung function. At the time of this study more than 90% of cystic fibrosis patients eventually died of lung disease.

Deoxyribonuclease I (DNase I) is a human enzyme normally present in the mucus of human lungs that digests extracellular DNA. Genentech, Inc. cloned a highly purified recombinant DNase I (rhDNase or Pulmozyme) which when delivered to the lungs in an aerosolized form cuts extracellular DNA, reducing the viscoelasticity of airway secretions and improving clearance. In 1992 the company conducted a randomized double-blind trial comparing rhDNase to placebo. Patients were then monitored for pulmonary exacerbations, along with measures of lung volume and flow. The primary endpoint was the time until first pulmonary exacerbation; however, data on all exacerbations were collected for 169 days.

The definition of an exacerbation was an infection that required the use of intravenous (IV) antibiotics. Subjects had 0–5 such episodes during the trial, those with more than one have multiple rows in the data set, those with none have NA for the IV start and end times. A few subjects were infected at the time of enrollment, subject 173 for instance has a first infection interval of -21 to 7. We do not count this first infection as an "event", and the subject first enters the risk set at day 7. Subjects

who have an event are not considered to be at risk for another event during the course of antibiotics, nor for an additional 6 days after they end. (If the symptoms reappear immediately after cessation then from a medical standpoint this would not be a new infection.)

This data set reproduces the data in Therneau and Grambsch, it does not exactly reproduce those in Therneau and Hamilton due to data set updates.

References

T. M. Therneau and P. M. Grambsch, Modeling Survival Data: Extending the Cox Model, Springer, 2000.

T. M. Therneau and S.A. Hamilton, rhDNase as an example of recurrent event analysis, Statistics in Medicine, 16:2029-2047, 1997.

Examples

```
# Build the start-stop data set for analysis, and
#  replicate line 2 of table 8.13 in the book
first <- subset(rhDNase, !duplicated(id)) #first row for each subject
dnase <- tmerge(first, first, id=id, tstop=as.numeric(end.dt -entry.dt))

# Subjects whose fu ended during the 6 day window are the reason for
#  this next line
temp.end <- with(rhDNase, pmin(ivstop+6, end.dt-entry.dt))
dnase <- tmerge(dnase, rhDNase, id=id,
                       infect=event(ivstart),
                       end=  event(temp.end))
# toss out the non-at-risk intervals, and extra variables
#  3 subjects had an event on their last day of fu, infect=1 and end=1
dnase <- subset(dnase, (infect==1 | end==0), c(id:trt, fev:infect))
agfit <- coxph(Surv(tstart, tstop, infect) ~ trt + fev, cluster=id,
                data=dnase)
```

`ridge` *Ridge regression*

Description

When used in a coxph or survreg model formula, specifies a ridge regression term. The likelihood is penalised by `theta`/2 time the sum of squared coefficients. If `scale=T` the penalty is calculated for coefficients based on rescaling the predictors to have unit variance. If `df` is specified then `theta` is chosen based on an approximate degrees of freedom.

Usage

```
ridge(..., theta, df=nvar/2, eps=0.1, scale=TRUE)
```

Arguments

...	predictors to be ridged
theta	penalty is theta/2 time sum of squared coefficients
df	Approximate degrees of freedom
eps	Accuracy required for df
scale	Scale variables before applying penalty?

Value

An object of class coxph.penalty containing the data and control functions.

Note

If the expression ridge(x1, x2, x3, ...) is too many characters long then the internal terms() function will add newlines to the variable name and then the coxph routine simply gets lost. (Some labels will have the newline and some won't.) One solution is to bundle all of the variables into a single matrix and use that matrix as the argument to ridge so as to shorten the call, e.g. mdata$many <- as.matrix(mydata[,5:53]).

References

Gray (1992) "Flexible methods of analysing survival data using splines, with applications to breast cancer prognosis" JASA 87:942–951

See Also

coxph,survreg,pspline,frailty

Examples

```
coxph(Surv(futime, fustat) ~ rx + ridge(age, ecog.ps, theta=1),
      ovarian)

lfit0 <- survreg(Surv(time, status) ~1, lung)
lfit1 <- survreg(Surv(time, status) ~ age + ridge(ph.ecog, theta=5), lung)
lfit2 <- survreg(Surv(time, status) ~ sex + ridge(age, ph.ecog, theta=1), lung)
lfit3 <- survreg(Surv(time, status) ~ sex + age + ph.ecog, lung)
```

rotterdam *Breast cancer data set used in Royston and Altman (2013)*

Description

The rotterdam data set includes 2982 primary breast cancers patients whose records were included in the Rotterdam tumor bank.

Usage

```
rotterdam
data(cancer, package="survival")
```

Format

A data frame with 2982 observations on the following 15 variables.

pid patient identifier

year year of surgery

age age at surgery

meno menopausal status (0= premenopausal, 1= postmenopausal)

size tumor size, a factor with levels <=20 20-50 >50

grade differentiation grade

nodes number of positive lymph nodes

pgr progesterone receptors (fmol/l)

er estrogen receptors (fmol/l)

hormon hormonal treatment (0=no, 1=yes)

chemo chemotherapy

rtime days to relapse or last follow-up

recur 0= no relapse, 1= relapse

dtime days to death or last follow-up

death 0= alive, 1= dead

Details

These data sets are used in the paper by Royston and Altman that is referenced below. The Rotterdam data is used to create a fitted model, and the GBSG data for validation of the model. The paper gives references for the data source.

There are 43 subjects who have died without recurrence, but whose death time is greater than the censoring time for recurrence. A common way that this happens is that a death date is updated in the health record sometime after the research study ended, and said value is then picked up when a study data set is created. Vital status information can come from many sources: a patient visit for another condition, correspondence, financial transactions, or social media. But this raises serious

questions about censoring. For instance subject 40 is censored for recurrence at 4.2 years and died at 6.6 years; when creating the endpoint of recurrence free survival (earlier of recurrence or death), treating them as a death at 6.6 years implicitly assumes that they were recurrence free just before death. For this to be true we would have to assume that if they had progressed in the 2.4 year interval before death (while off study), that this information would also have been noted in their general medical record, and would also be captured in the study data set. However, that may be unlikely. Death information is often in a centralized location in electronic health records, easily accessed by a programmer and merged with the study data, while recurrence may require manual review. How best to address this is an open issue.

References

Patrick Royston and Douglas Altman, External validation of a Cox prognostic model: principles and methods. BMC Medical Research Methodology 2013, 13:33

See Also

gbsg

Examples

```
# liberal definition of rfs (count later deaths)
rfs  <- pmax(rotterdam$recur, rotterdam$death)
rfstime <- with(rotterdam, ifelse(recur==1, rtime, dtime))
fit1 <- coxph(Surv(rfstime, rfs) ~ pspline(age) + meno + size +
        pspline(nodes) + er,  data = rotterdam)

# conservative (no deaths after last fu for recurrence)
ignore <- with(rotterdam, recur ==0 & death==1 & rtime < dtime)
table(ignore)
rfs2 <- with(rotterdam, ifelse(recur==1 | ignore, recur, death))
rfstime2 <- with(rotterdam, ifelse(recur==1 | ignore, rtime, dtime))
fit2 <- coxph(Surv(rfstime2, rfs2) ~ pspline(age) + meno + size +
        pspline(nodes) + er,  data = rotterdam)

# Note: Both age and nodes show non-linear effects.
# Royston and Altman used fractional polynomials for the nonlinear terms
```

---

royston *Compute Royston's D for a Cox model*

Description

Compute the D statistic proposed by Royston and Sauerbrei along with several synthetic R square values.

Usage

```
royston(fit, newdata, ties = TRUE, adjust = FALSE)
```

Arguments

fit	a coxph fit
newdata	optional validation data set
ties	make a correction for ties in the risk score
adjust	adjust for possible overfitting

Details

We refer to these estimates of association as *synthetic* values, since they involve only the linear predictor, and not the outcome. They exploit mathematical associations which hold for certain models, e.g. between R-squared and a certain chiquare test of association in the linear model, and assume that the same holds in a Cox model where said test is readily available but not a simple R-square computation.

R.D is the value that corresponsds the Royston and Sauerbrei D statistic. R.KO is the value proposed by Kent and O'Quigley, R.N is the value proposed by Nagelkerke, and C.GH corresponds to Goen and Heller's concordance measure.

An adjustment for D is based on the ratio r= (number of events)/(number of coefficients). For models which have sufficient sample size (r>20) the adjustment will be small.

The Nagelkerke value is the Cox-Snell R-squared divided by a scaling constant. The two separate values are present in the result of summary.coxph as a 2 element vector rsq, and were listed as "Rsquare" and "max possible" in older versions of the print routine. (Since superseded in the default printout by the concordance.) The Nagelkerke estimate is not returned when newdata is present.

Value

a vector containing the value of D, the estimated standard error of D, and three or four synthetic values.

References

M. Goen and G. Heller, Concordance probability and discriminatory power in proportional hazards regression. Biometrika 92:965-970, 2005.

N. Nagelkerke, J. Oosting, J. and A. Hart, A simple test for goodness of fit of Cox's proportional hazards model. Biometrics 40:483-486, 1984.

P. Royston and W. Sauerbrei, A new measure of prognostic separation in survival data. Statistics in Medicine 23:723-748, 2004.

Examples

```
# An example used in Royston and Sauerbrei
pbc2 <- na.omit(pbc)  # no missing values
cfit <- coxph(Surv(time, status==2) ~ age + log(bili) + edema + albumin +
                  stage + copper, data=pbc2, ties="breslow")
royston(cfit)
```

rttright *Compute redistribute-to-the-right weights*

Description

For many survival estimands, one approach is to redistribute each censored observation's weight to those other observations with a longer survival time (think of distributing an estate to the heirs). Then compute on the remaining, uncensored data.

Usage

```
rttright(formula, data, weights, subset, na.action, times, id, timefix = TRUE,
         renorm= TRUE)
```

Arguments

formula	a formula object, which must have a Surv object as the response on the left of the ~ operator and, if desired, terms separated by + operators on the right. Each unique combination of predictors will define a separate strata.
data	a data frame in which to interpret the variables named in the formula, subset and weights arguments.
weights	The weights must be nonnegative and it is strongly recommended that they be strictly positive, since zero weights are ambiguous, compared to use of the subset argument.
subset	expression saying that only a subset of the rows of the data should be used in the fit.
na.action	a missing-data filter function, applied to the model frame, after any subset argument has been used. Default is options()$na.action.
times	a vector of time points, for which to return updated weights. If missing, a time after the largest time in the data is assumed.
id	optional: if the data set has multiple rows per subject, a a variable containing the subect identifier of each row.
timefix	correct for possible round-off error
renorm	the resulting weights sum to 1 within each group

Details

The formula argument is treated exactly the same as in the survfit function.

Redistribution is recursive: redistribute the weight of the first censored observation to all those with longer time, which may include other censored observations. Then redistribute the next smallest and etc. up to the specified time value. After re-distributing the weight for a censored observation to other observations that are not censored, ordinary non-censored methods can often be applied. For example, redistribution of the weights, followed by computation of the weighted cumulative distribution function, reprises the Kaplan-Meier estimator.

A primary use of this routine is illustration of methods or exploration of new methods. Methods that use RTTR directly, such as the Brier score, will often do these compuations internally.

A covariate on the right hand side of the formula causes redistribution to occur within group; a censoring in group 1 redistributes weights to others in group 1, etc. This is appropriate when the censoring pattern depends upon group.

Value

a vector or matrix of weights, with one column for each requested time

See Also

survfit

Examples

```
afit <- survfit(Surv(time, status) ~1, data=aml)
rwt  <- rttright(Surv(time, status) ~1, data=aml)

# Reproduce a Kaplan-Meier
index <- order(aml$time)
cdf <- cumsum(rwt[index])  # weighted CDF
cdf <- cdf[!duplicated(aml$time[index], fromLast=TRUE)]  # remove duplicate times
cbind(time=afit$time, KM= afit$surv, RTTR= 1-cdf)

# Hormonal patients have a diffent censoring pattern
wt2 <- rttright(Surv(dtime, death) ~ hormon, rotterdam, times= 365*c(3, 5))
dim(wt2)
```

---

solder *Data from a soldering experiment*

Description

In 1988 an experiment was designed and implemented at one of AT&T's factories to investigate alternatives in the "wave soldering" procedure for mounting electronic componentes to printed circuit boards. The experiment varied a number of factors relevant to the process. The response, measured by eye, is the number of visible solder skips.

Usage

```
solder
data(solder, package="survival")
```

Format

A data frame with 900 observations on the following 6 variables.

`Opening` the amount of clearance around the mounting pad (3 levels)

`Solder` the amount of solder (Thick or Thin)

`Mask` type and thickness of the material used for the solder mask (A1.5, A3, A6, B3, B6)

`PadType` the geometry and size of the mounting pad (10 levels)

`Panel` each board was divided into 3 panels

`skips` the number of skips

Details

After the first 1/2 of the experiment the A6 mask, which was doing the worst, was abandoned and the freed up space used for further replicates of A3. This leads to an unbalanced experiment with some missing A6 combinations.

This data set is used as a detailed example in chapter 1 of Chambers and Hastie. However, they chose to use only a subset of the data, i.e., observations 1-360 and 541-900 form a balanced design of 3*2*10*3= 180 observations for each of four mask types (A1.5, A3, B3, B6).

References

J Chambers and T Hastie, Statistical models in S. Chapman and Hall, 1993.

Examples

```
fit1 <- glm(skips ~ Opening * Solder, poisson, solder,
            subset= (Mask != "A6"))
anova(fit1)  # The interaction is important
dummy <- expand.grid(Opening= c("S", "M", "L"), Solder=c("Thin", "Thick"))
yhat <- matrix(predict(fit1, newdata=dummy), ncol=2,
            dimnames=list(Opening= c("S", "M", "L"), Solder=c("Thin", "Thick")))
yhat <- cbind(yhat, difference= yhat[,1]- yhat[,2])
round(yhat, 1) # thin and thick have  different patterns

# The balanced subset used by Chambers and Hastie
#   contains the first 180 of each mask and deletes mask A6.
index <- 1 + (1:nrow(solder)) - match(solder$Mask, solder$Mask)
solder.balance <- droplevels(subset(solder, Mask != "A6" & index <= 180))
```

---

stanford2 *More Stanford Heart Transplant data*

Description

This contains the Stanford Heart Transplant data in a different format. The main data set is in `heart`.

Usage

```
stanford2
```

Format

id:	ID number
time:	survival or censoring time
status:	censoring status
age:	in years
t5:	T5 mismatch score

Source

LA Escobar and WQ Meeker Jr (1992), Assessing influence in regression analysis with censored data. *Biometrics* **48**, 507–528. Page 519.

See Also

predict.survreg, heart

---

statefig *Draw a state space figure.*

Description

For multi-state survival models it is useful to have a figure that shows the states and the possible transitions between them. This function creates a simple "box and arrows" figure. It's goal was simplicity.

Usage

```
statefig(layout, connect, margin = 0.03, box = TRUE, cex = 1, col = 1,
  lwd=1, lty=1, bcol=col, acol=col, alwd=lwd, alty=lty, offset=0)
```

Arguments

layout	describes the layout of the boxes on the page. See the detailed description below.
connect	a square matrix with one row for each state. If connect[i,j] !=0 then an arrow is drawn from state i to state j. The row names of the matrix are used as the labels for the states.
margin	the fraction of white space between the label and the surrounding box, and between the box and the arrows, as a function of the plot region size.

box — should boxes be drawn? TRUE or FALSE.

cex, col, lty, lwd — default graphical parameters used for the text and boxes. The last 3 can be a vector of values.

bcol — color for the box, if it differs from that used for the text.

acol, alwd, alty — color, line type and line width for the arrows.

offset — used to slight offset the arrows between two boxes x and y if there is a transition in both directions. The default of 0 leads to a double headed arrow in this case – to arrows are drawn but they coincide. A positive value causes each arrow to shift to the left, from the view of someone standing at the foot of a arrow and looking towards the arrowhead, a negative offset shifts to the right. A value of 1 corresponds to the size of the plotting region.

Details

The arguments for color, line type and line width can all be vectors, in which case they are recycled as needed. Boxes and text are drawn in the order of the rownames of connect, and arrows are drawn in the usual R matrix order.

The layout argument is normally a vector of integers, e.g., the vector (1, 3, 2) describes a layout with 3 columns. The first has a single state, the second column has 3 states and the third has 2. The coordinates of the plotting region are 0 to 1 for both x and y. Within a column the centers of the boxes are evenly spaced, with 1/2 a space between the boxes and the margin, e.g., 4 boxes would be at 1/8, 3/8, 5/8 and 7/8. If layout were a 1 column matrix with values of (1, 3, 2) then the layout will have three rows with 1, 3, and 2 boxes per row, respectively. Alternatively, the user can supply a 2 column matrix that directly gives the centers.

The values of the connect matrix should be 0 for pairs of states that do not have a transition and values between 0 and 2 for those that do. States are connected by an arc that passes through the centers of the two boxes and a third point that is between them. Specifically, consider a line segment joining the two centers and erect a second segment at right angles to the midpoint of length d times the distance from center to midpoint. The arc passes through this point. A value of d=0 gives a straight line, d=1 a right hand half circle centered on the midpoint and d= -1 a left hand half circle. The connect matrix contains values of d+1 with $-1 < d < 1$.

The connecting arrow are drawn from (center of box 1 + offset) to (center of box 2 + offset), where the the amount of offset (white space) is determined by the box and margin parameters. If a pair of states are too close together this can result in an arrow that points the wrong way.

Value

a matrix containing the centers of the boxes, with the invisible attribute set.

Note

The goal of this function is to make "good enough" figures as simply as possible, and thereby to encourage users to draw them. The layout argument was inspired by the diagram package, which can draw more complex and well decorated figures, e.g., many different shapes, shading, multiple types of connecting lines, etc., but at the price of greater complexity.

Because curved lines are drawn as a set of short line segments, line types have almost no effect for that case.

Author(s)

Terry Therneau

Examples

```
# Draw a simple competing risks figure
states <- c("Entry", "Complete response", "Relapse", "Death")
connect <- matrix(0, 4, 4, dimnames=list(states, states))
connect[1, -1] <- c(1.1, 1, 0.9)
statefig(c(1, 3), connect)
```

---

strata *Identify Stratification Variables*

Description

This is a special function used in the context of the Cox survival model. It identifies stratification variables when they appear on the right hand side of a formula.

Usage

```
strata(..., na.group=FALSE, shortlabel, sep=', ')
```

Arguments

...	any number of variables. All must be the same length.
na.group	a logical variable, if TRUE, then missing values are treated as a distinct level of each variable.
shortlabel	if TRUE omit variable names from resulting factor labels. The default action is to omit the names if all of the arguments are factors, and none of them was named.
sep	the character used to separate groups, in the created label

Details

When used outside of a coxph formula the result of the function is essentially identical to the interaction function, though the labels from strata are often more verbose.

Value

a new factor, whose levels are all possible combinations of the factors supplied as arguments.

See Also

coxph, interaction

Examples

```
a <- factor(rep(1:3,4), labels=c("low", "medium", "high"))
b <- factor(rep(1:4,3))
levels(strata(b))
levels(strata(a,b,shortlabel=TRUE))

coxph(Surv(futime, fustat) ~ age + strata(rx), data=ovarian)
```

---

summary.aareg *Summarize an aareg fit*

Description

Creates the overall test statistics for an Aalen additive regression model

Usage

```
## S3 method for class 'aareg'
summary(object, maxtime, test=c("aalen", "nrisk"), scale=1,...)
```

Arguments

object	the result of a call to the aareg function
maxtime	truncate the input to the model at time "maxtime"
test	the relative time weights that will be used to compute the test
scale	scales the coefficients. For some data sets, the coefficients of the Aalen model will be very small (10-4); this simply multiplies the printed values by a constant, say 1e6, to make the printout easier to read.
...	for future methods

Details

It is not uncommon for the very right-hand tail of the plot to have large outlying values, particularly for the standard error. The maxtime parameter can then be used to truncate the range so as to avoid these. This gives an updated value for the test statistics, without refitting the model.

The slope is based on a weighted linear regression to the cumulative coefficient plot, and may be a useful measure of the overall size of the effect. For instance when two models include a common variable, "age" for instance, this may help to assess how much the fit changed due to the other variables, in leiu of overlaying the two plots. (Of course the plots are often highly non-linear, so it is only a rough substitute). The slope is not directly related to the test statistic, as the latter is invariant to any monotone transformation of time.

Value

a list is returned with the following components

table	a matrix with rows for the intercept and each covariate, and columns giving a slope estimate, the test statistic, it's standard error, the z-score and a p-value
test	the time weighting used for computing the test statistics
test.statistic	the vector of test statistics
test.var	the model based variance matrix for the test statistic
test.var2	optionally, a robust variance matrix for the test statistic
chisq	the overall test (ignoring the intercept term) for significance of any variable
n	a vector containing the number of observations, the number of unique death times used in the computation, and the total number of unique death times

See Also

aareg, plot.aareg

Examples

```
afit <- aareg(Surv(time, status) ~ age + sex + ph.ecog, data=lung,
     dfbeta=TRUE)
summary(afit)
## Not run:
              slope   test se(test) robust se     z        p
Intercept  5.05e-03    1.9     1.54      1.55  1.23 0.219000
      age  4.01e-05  108.0   109.00    106.00  1.02 0.307000
      sex -3.16e-03  -19.5     5.90      5.95 -3.28 0.001030
  ph.ecog  3.01e-03   33.2     9.18      9.17  3.62 0.000299

Chisq=22.84 on 3 df, p=4.4e-05; test weights=aalen

## End(Not run)

summary(afit, maxtime=600)
## Not run:
              slope   test se(test) robust se      z        p
Intercept  4.16e-03   2.13     1.48      1.47  1.450 0.146000
      age  2.82e-05  85.80   106.00    100.00  0.857 0.392000
      sex -2.54e-03 -20.60     5.61      5.63 -3.660 0.000256
  ph.ecog  2.47e-03  31.60     8.91      8.67  3.640 0.000271

Chisq=27.08 on 3 df, p=5.7e-06; test weights=aalen

## End(Not run)
```

summary.coxph *Summary method for Cox models*

Description

Produces a summary of a fitted coxph model

Usage

```
## S3 method for class 'coxph'
summary(object, conf.int=0.95, scale=1,...)
```

Arguments

object	the result of a coxph fit
conf.int	level for computation of the confidence intervals. If set to FALSE no confidence intervals are printed
scale	vector of scale factors for the coefficients, defaults to 1. The printed coefficients, se, and confidence intervals will be associated with one scale unit.
...	for future methods

Value

An object of class summary.coxph, with components:

n, nevent	number of observations and number of events, respectively, in the fit
loglik	the log partial likelihood at the initial and final values
coefficients	a matrix with one row for each coefficient, and columns containing the coefficient, the hazard ratio exp(coef), standard error, Wald statistic, and P value.
conf.int	a matrix with one row for each coefficient, containing the confidence limits for exp(coef)
logtest, sctest, waldtest	the overall likelihood ratio, score, and Wald test statistics for the model
concordance	the concordance statistic and its standard error
used.robust	whether an asymptotic or robust variance was used
rsq	an approximate R^2 based on Nagelkerke (Biometrika 1991).
fail	a message, if the underlying coxph call failed
call	a copy of the call
na.action	information on missing values

Note

The pseudo r-squared of Nagelkerke is attractive because it is simple, but further work has shown that it has poor properties and it is now deprecated. The value is no longer printed by default, and will eventually be removed from the object. The royston function now includes it along with several other measures of association.

See Also

coxph, print.coxph

Examples

```
fit <- coxph(Surv(time, status) ~ age + sex, lung)
summary(fit)
```

---

summary.pyears *Summary function for pyears objecs*

Description

Create a printable table of a person-years result.

Usage

```
## S3 method for class 'pyears'
summary(object, header = TRUE, call = header, n = TRUE,
event = TRUE, pyears = TRUE, expected = TRUE, rate = FALSE, rr =expected,
ci.r = FALSE, ci.rr = FALSE, totals=FALSE, legend = TRUE, vline = FALSE,
vertical= TRUE, nastring=".", conf.level = 0.95,
scale = 1, ...)
```

Arguments

object	a pyears object
header	print out a header giving the total number of observations, events, person-years, and total time (if any) omitted from the table
call	print out a copy of the call
n, event, pyears, expected	logical arguments: should these elements be printed in the table?
rate, ci.r	logical arguments: should the incidence rate and/or its confidence interval be given in the table?
rr, ci.rr	logical arguments: should the hazard ratio and/or its confidence interval be given in the table?
totals	should row and column totals be added?
legend	should a legend be included in the printout?
vline	should vertical lines be included in the printed tables?
vertical	when there is only a single predictor, should the table be printed with the predictor on the left (vertical=TRUE) or across the top (vertical=FALSE)?
nastring	what to use for missing values in the table. Some of these are structural, e.g., risk ratios for a cell with no follow-up time.
conf.level	confidence level for any confidence intervals

scale	a scaling factor for printed rates
...	optional arguments which will be passed to the format function; common choices would be digits=2 or nsmall=1.

Details

The pyears function is often used to create initial descriptions of a survival or time-to-event variable; the type of material that is often found in "table 1" of a paper. The summary routine prints this information out using one of pandoc table styles. A primary reason for choosing this style is that Rstudio is then able to automatically render the results in multiple formats: html, rtf, latex, etc.

If the pyears call has only a single covariate then the table will have that covariate as one margin and the statistics of interest as the other. If the pyears call has two predictors then those two predictors are used as margins of the table, while each cell of the table contains the statistics of interest as multiple rows within the cell. If there are more than two predictors then multiple tables are produced, in the same order as the standard R printout for an array.

The "N" entry of a pyears object is the number of observations which contributed to a particular cell. When the original call includes tcut objects then a single observation may contribute to multiple cells.

Value

a copy of the object

Notes

The pandoc system has four table types: with or without vertical bars, and with single or multiple rows of data in each cell. This routine produces all 4 styles depending on options, but currently not all of them are recognized by the Rstudio-pandoc pipeline. (And we don't yet see why.)

Author(s)

Terry Therneau and Elizabeth Atkinson

See Also

cipoisson, pyears, format

---

summary.survexp *Summary function for a survexp object*

Description

Returns a list containing the values of the survival at specified times.

Usage

```
## S3 method for class 'survexp'
summary(object, times, scale = 1, ...)
```

Arguments

object	the result of a call to the survexp function
times	vector of times; the returned matrix will contain 1 row for each time. Missing values are not allowed.
scale	numeric value to rescale the survival time, e.g., if the input data to survfit were in days, scale = 365.25 would scale the output to years.
...	For future methods

Details

A primary use of this function is to retrieve survival at fixed time points, which will be properly interpolated by the function.

Value

a list with the following components:

surv	the estimate of survival at time t.
time	the timepoints on the curve.
n.risk	In expected survival each subject from the data set is matched to a hypothetical person from the parent population, matched on the characteristics of the parent population. The number at risk is the number of those hypothetical subject who are still part of the calculation.

Author(s)

Terry Therneau

See Also

survexp

---

summary.survfit *Summary of a Survival Curve*

Description

Returns a list containing the survival curve, confidence limits for the curve, and other information.

Usage

```
  ## S3 method for class 'survfit'
summary(object, times, censored=FALSE, scale=1,
  extend=FALSE, rmean=getOption('survfit.rmean'), data.frame=FALSE, dosum, ...)
  ## S3 method for class 'survfitms'
summary(object, times, censored=FALSE, scale=1,
  extend=FALSE, rmean=getOption('survfit.rmean'), data.frame=FALSE, dosum, ...)
```

Arguments

object	the result of a call to the survfit function.
times	vector of times; the returned matrix will contain 1 row for each time. The vector will be sorted into increasing order; missing values are not allowed. If censored=T, the default times vector contains all the unique times in fit, otherwise the default times vector uses only the event (death) times.
censored	logical value: should the censoring times be included in the output? This is ignored if the times argument is present.
scale	numeric value to rescale the survival time, e.g., if the input data to survfit were in days, scale = 365.25 would scale the output to years.
extend	logical value: if TRUE, prints information for all specified times, even if there are no subjects left at the end of the specified times. This is only used if the times argument is present.
rmean	Show restricted mean: see print.survfit for details
data.frame	if TRUE, return the results as a data frame, rather than a summary.survfit object
dosum	only applicable if times is present, see details below
...	for future methods

Value

if data.frame = TRUE, a data frame with columns of time, n.risk, n.event, n.censor, surv, strata (if present) and, for survfit.coxph, the relevant row of newdata. Also std.err, upper and lower if the curve had se.fit=TRUE.

if data.frame = FALSE, a list with the following components:

surv	the estimate of survival at time t+0.
time	the timepoints on the curve.
n.risk	the number of subjects at risk at time t-0 (but see the comments on weights in the survfit help file).
n.event	if the times argument is missing, then this column is the number of events that occurred at time t. Otherwise, it is the cumulative number of events that have occurred since the last time listed until time t+0.
n.entered	This is present only for counting process survival data. If the times argument is missing, this column is the number of subjects that entered at time t. Otherwise, it is the cumulative number of subjects that have entered since the last time listed until time t.
n.exit.censored	if the times argument is missing, this column is the number of subjects that left without an event at time t. Otherwise, it is the cumulative number of subjects that have left without an event since the last time listed until time t+0. This is only present for counting process survival data.
std.err	the standard error of the survival value.
conf.int	level of confidence for the confidence intervals of survival.

lower	lower confidence limits for the curve.
upper	upper confidence limits for the curve.
strata	indicates stratification of curve estimation. If strata is not NULL, there are multiple curves in the result and the surv, time, n.risk, etc. vectors will contain multiple curves, pasted end to end. The levels of strata (a factor) are the labels for the curves.
call	the statement used to create the fit object.
na.action	same as for fit, if present.
table	table of information that is returned from print.survfit function.
type	type of data censoring. Passed through from the fit object.

Details

This routine has two uses: printing out a survival curve at specified time points (often yearly), or extracting the values at specified time points for further processing. In the first case we normally want extend=FALSE, i.e., don't print out data past the end of the curve. If the times option only contains values beyond the last point in the curve then there is nothing to print and an error message will result. For the second usage we often want extend=TRUE, so that the results will have a predictable length. If data.frame = TRUE then either might be desired. Be aware, however, that these extended values will often be badly biased; we are essentialy treating the final censored subjects as immortal.

The underlying survival object will have a row for each unique event or censoring time. When the times argument contains values not in the data, the routine can only use a best guess for the number at risk, i.e., the number at risk at the next event/censoring time. When the routine is called with counting process data many users are confused by counts that appear too large. For example, Surv(c(0,0, 1, 5), c(2, 3, 8, 10), c(1, 0, 1, 0)), which prints as (0,2] (0, 3+] (1, 8] (5,10+]. Do survfit followed by summary with a request for the values at time 0. The survfit object has entries only at times 2, 3, 8, and 10; there are 3 subjects at risk at time 2, so that is what will be printed for time 0.

For a printout at fixed times, for example yearly values for a curve, the printed number of events will by default be the total number of events that have occured since the prior line of printout, and likewise for number of censored and number at entry, dosum = TRUE. Alternately, the routine can return the number of events/censors/entry at that time, dosum=FALSE. This feature was added at the request of a user who essentially wanted to use the times argument as a subscript to pick off selected rows of the output, e.g., to select survival values corresponding to the last follow-up times of a new set of observations. The default for dosum is TRUE if the times vector is strictly increasing and FALSE otherwise.

For a survfitms object replace the surv component with pstate. Also, a data frame will not include the cumulative hazard nor its standard error, since it has a different multiplicity: one column per transition rather than one per state.

See Also

survfit, print.summary.survfit

Examples

```
summary( survfit( Surv(futime, fustat)~1, data=ovarian))
summary( survfit( Surv(futime, fustat)~rx, data=ovarian))
```

`Surv` *Create a Survival Object*

Description

Create a survival object, usually used as a response variable in a model formula. Argument matching is special for this function, see Details below.

Usage

```
Surv(time, time2, event,
    type=c('right', 'left', 'interval', 'counting', 'interval2'),
    origin=0)
is.Surv(x)
```

Arguments

`time`	for right censored data, this is the follow up time. For interval data, the first argument is the starting time for the interval.
`event`	The status indicator, normally 0=alive, 1=dead. Other choices are `TRUE`/`FALSE` (`TRUE` = death), 1/2 (2=death), or a factor variable. For interval censored data, the status indicator is 0=right censoed, 1=event at `time`, 2=left censored, 3=interval censored. For multiple endpoint data the event variable will always be a factor, whose first level is treated as censoring, or more formally "no transtition at this time point". There is no constraint on the labels of the factor. Although unusual, the event indicator can be omitted, in which case all subjects are assumed to have an event.
`time2`	ending time of the interval for interval censored or counting process data only. Intervals are assumed to be open on the left and closed on the right, `(start, end]`. For counting process data, `event` indicates whether a transtion to another state occurred at the end of the interval.
`type`	character string specifying the type of censoring. Possible values are `"right"`, `"left"`, `"counting"`, `"interval"`, `"interval2"`. The default is multi-state if `event` is a factor, counting process if `time2` is present, or right censored, in that order.
`origin`	for counting process data, the hazard function origin. This option was intended to be used in conjunction with a model containing time dependent strata in order to align the subjects properly when they cross over from one strata to another, but it has rarely proven useful and is depricated.
`x`	any R object.

Details

Interval censored data can be represented in two ways. For the first use `type = "interval"` and the codes shown above. In that usage the value of the `time2` argument is ignored unless event=3. The second approach is to think of each observation as a time interval with (-infinity, t2) for left censored, (t1, infinity) for right censored, (t,t) for exact and (t1, t2) for an interval. This is the approach used for type = interval2. Infinite values can be represented either by actual infinity (Inf) or NA. The second form has proven to be the more useful one.

Presently, the only methods allowing interval censored data are the parametric models computed by `survreg` and survival curves computed by `survfit`; for both of these, the distinction between open and closed intervals is unimportant. The distinction is important for counting process data and the Cox model.

The function tries to distinguish between the use of 0/1 and 1/2 coding for censored data via the condition `if (max(status)==2)`. If 1/2 coding is used and all the subjects are censored, it will guess wrong. In any questionable case it is safer to use logical coding, e.g., `Surv(time, status==3)` would indicate that '3' is the code for an event. For multi-state survival the status variable will be a factor, whose first level is assumed to correspond to censoring. If there are no uncensored subjects, ensure that the levels of the factor still contain the censored level!

Surv objects can be subscripted either as a vector, e.g. `x[1:3]` using a single subscript, in which case the `drop` argument is ignored and the result will be a survival object; or as a matrix by using two subscripts. If the second subscript is missing and `drop=F` (the default), the result of the subscripting will be a Surv object, e.g., `x[1:3,,drop=F]`, otherwise the result will be a matrix (or vector), in accordance with the default behavior for subscripting matrices.

Value

An object of class `Surv`. There are methods for `print`, `is.na`, and subscripting survival objects. `Surv` objects are implemented as a matrix of 2 or 3 columns that has further attributes. These include the type (left censored, right censored, counting process, etc.) and labels for the states for multi-state objects. Any attributes of the input arguments are also preserved in `inputAttributes`. This may be useful for other packages that have attached further information to data items such as labels; none of the routines in the survival package make use of these values, however.

In the case of `is.Surv`, a logical value `TRUE` if `x` inherits from class `"Surv"`, otherwise an `FALSE`.

Note

The use of 1/2 coding for status is an interesting historical artifact. For data contained on punch cards, IBM 360 Fortran treated blank as a zero, which led to a policy within the Mayo Clinic section of Biostatistics to never use "0" as a data value, since one could not distinguish it from a missing value (blank). Policy became habit, as is often the case, and the use of 1/2 coding for no/yes variables, including death, endured long after the demise of the punch cards that had sired the practice. At the time `Surv` was written many Mayo data sets still used this obsolete convention, e.g., the `lung` data set found in the package.

See Also

`coxph`, `survfit`, `survreg`, `lung`.

Examples

```
with(aml, Surv(time, status))
survfit(Surv(time, status) ~ ph.ecog, data=lung)
Surv(heart$start, heart$stop, heart$event)
```

Surv-methods *Methods for Surv objects*

Description

The list of methods that apply to Surv objects

Usage

```
    ## S3 method for class 'Surv'
anyDuplicated(x, ...)
    ## S3 method for class 'Surv'
as.character(x, ...)
    ## S3 method for class 'Surv'
as.data.frame(x, ...)
    ## S3 method for class 'Surv'
as.matrix(x, ...)
    ## S3 method for class 'Surv'
c(...)
    ## S3 method for class 'Surv'
duplicated(x, ...)
    ## S3 method for class 'Surv'
format(x, ...)
    ## S3 method for class 'Surv'
head(x, ...)
    ## S3 method for class 'Surv'
is.na(x)
    ## S3 method for class 'Surv'
length(x)
    ## S3 method for class 'Surv'
mean(x, ...)
    ## S3 method for class 'Surv'
median(x, na.rm=FALSE, ...)
    ## S3 method for class 'Surv'
names(x)
    ## S3 replacement method for class 'Surv'
names(x) <- value
    ## S3 method for class 'Surv'
quantile(x, probs, na.rm=FALSE, ...)
    ## S3 method for class 'Surv'
plot(x, ...)
    ## S3 method for class 'Surv'
```

```
rep(x, ...)
    ## S3 method for class 'Surv'
rep.int(x, ...)
    ## S3 method for class 'Surv'
rep_len(x, ...)
    ## S3 method for class 'Surv'
rev(x)
    ## S3 method for class 'Surv'
t(x)
    ## S3 method for class 'Surv'
tail(x, ...)
    ## S3 method for class 'Surv'
unique(x, ...)
```

Arguments

x	a Surv object
probs	a vector of probabilities
na.rm	remove missing values from the calculation
value	a character vector of up to the same length as x, or NULL
...	other arguments to the method

Details

These functions extend the standard methods to Surv objects. (There is no central index of R methods, so there may well be useful candidates that the author has missed.) The arguments and results from these are mostly as expected, with the following further details:

- The as.character function uses "5+" for right censored at time 5, "5-" for left censored at time 5, "[2,7]" for an observation that was interval censored between 2 and 7, "(1,6]" for a counting process data denoting an observation which was at risk from time 1 to 6, with an event at time 6, and "(1,6+]" for an observation over the same interval but not ending with and event. For a multi-state survival object the type of event is appended to the event time using ":type".
- The print and format methods make use of as.character.
- The length of a Surv object is the number of survival times it contains, not the number of items required to encode it, e.g., x <- Surv(1:4, 5:8, c(1,0,1,0)); length(x) has a value of 4. Likewise names(x) will be NULL or a vector of length 4. (For technical reasons, any names are actually stored in the rownames attribute of the object.)
- For a multi-state survival object levels returns the names of the endpoints, otherwise it is NULL.
- The median, quantile and plot methods first construct a survival curve using survfit, then apply the appropriate method to that curve.
- The xtfrm method, which underlies sort and order, sorts by time, with censored after uncensored within a tied time. For an interval censored observation the midpoint is used. For (time1, time2) counting process data, sorting is by time2, censoring, and then time1.

- The `unique` method treats censored and uncensored observations at the same time as distinct, it returns a Surv object.
- The concatonation method `c()` is asymmetric, its first argument determines the execution path. For instance `c(Surv(1:4), Surv(5:6))` will return a Surv object of length 6, `c(Surv(1:4), 5:6)` will give an error, and `c(5:6, Surv(1:4))` is equivalent to `c(5:6, as.vector(Surv(1:4)))` which is a numeric of length 10.

See Also

`Surv`

`Surv2` *Create a survival object*

Description

Create a survival object from a timeline style data set. This will almost always be the response variable in a formula.

Usage

```
Surv2(time, event, repeated=FALSE)
```

Arguments

`time`	a timeline variable, such as age, time from enrollment, date, etc.
`event`	the outcome at that time. This can be a 0/1 variable, TRUE/FALSE, or a factor. If the latter, the first level of the factor corresponds to 'no event was observed at this time'.
`repeated`	if the same level of the outcome repeats, without an intervening event of another type, should this be treated as a new event?

Details

This function is still experimental.

When used in a `coxph` or `survfit` model, Surv2 acts as a trigger to internally convert a timeline style data set into counting process style data, which is then acted on by the routine.

The `repeated` argument controls how repeated instances of the same event code are treated. If TRUE, they are treated as new events, an example where this might be desired is repeated infections in a subject. If FALSE, then repeats are not a new event. An example would be a data set where we wanted to use diabetes, say, as an endpoint, but this is repeated at each medical visit.

Value

An object of class `Surv2`. There are methods for `print`, `is.na` and subscripting.

See Also

Surv2data, coxph, survfit

Surv2data *Convert data from timecourse to (time1,time2) style*

Description

The multi-state survival functions coxph and survfit allow for two forms of input data. This routine converts between them. The function is normally called behind the scenes when Surv2 is as the response.

Usage

```
Surv2data(formula, data, subset, id)
```

Arguments

formula	a model formula
data	a data frame
subset	optional, selects rows of the data to be retained
id	a variable that identified multiple rows for the same subject, normally found in the referenced data set

Details

For timeline style data, each row is uniquely identified by an (identifier, time) pair. The time could be a date, time from entry to a study, age, etc, (there may often be more than one time variable). The identifier and time cannot be missing. The remaining covariates represent values that were observed at that time point. Often, a given covariate is observed at only a subset of times and is missing at others. At the time of death, in particular, often only the identifier, time, and status indicator are known.

In the resulting data set missing covariates are replaced by their last known value, and the response y will be a Surv(time1, time2, endpoint) object.

Value

a list with elements

mf	an updated model frame (fewer rows, unchanged columns)
S2.y	the constructed response variable
S2.state	the current state for each of the rows

survcheck *Checks of a survival data set*

Description

Perform a set of consistency checks on survival data

Usage

```
survcheck(formula, data, subset, na.action, id, istate, istate0="(s0)",
timefix=TRUE,...)
```

Arguments

formula	a model formula with a Surv object as the response
data	data frame in which to find the id, istate and formula variables
subset	expression indicating which subset of the rows of data should be used in the fit. All observations are included by default.
na.action	a missing-data filter function. This is applied to the model.frame after any subset argument has been used. Default is options()\$na.action.
id	an identifier that labels unique subjects
istate	an optional vector giving the current state at the start of each interval
istate0	default label for the initial state of each subject (at their first interval) when istate is missing
timefix	process times through the aeqSurv function to eliminate potential roundoff issues.
...	other arguments, which are ignored (but won't give an error if someone added weights for instance)

Details

This routine will examine a multi-state data set for consistency of the data. The basic rules are that if a subject is at risk they have to be somewhere, can not be two places at once, and should make sensible transitions from state to state. It reports the number of instances of the following conditions:

overlap two observations for the same subject that overlap in time, e.g. intervals of (0, 100) and (90, 120). If y is simple (time, status) survival then intervals implicitly start at 0, so in that case any duplicate identifiers will generate an overlap.

gap one or more gaps in a subject's timeline; where they are in the same state at their return as when they left.

jump a hole in a subject's timeline, where they are in one state at the end of the prior interval, but a new state in the at the start subsequent interval.

teleport two adjacent intervals for a subject, with the first interval ending in one state and the subsequent interval starting in another. They have instantaneously changed states in 0 units of time.

duplicate not currently used

The total number of occurences of each is present in the `flags` vector. Optional components give the location and identifiers of the flagged observations. The `Surv` function has already flagged any 0 length intervals as errors.

One important caveat is that survcheck does not deal with reuse of an id value. For instance, a multi-institutional data set where the same subject identifier happens to have been used for two different subjects in two different institutions. The routine is likely generate a "false positive" error in this case, but this is simply unavoidable. Since the routine is used internally by `survfit`, `coxph`, etc. the same errors will appear in other routines in the survival package.

Value

a list with components

`states`	the vector of possible states, a union of what appears in the Surv object and `istate`, with initial states first
`transitions`	a matrix giving the count of transitions from one state to another
`statecount`	table of the number of visits per state, e.g., 18 subjects had 2 visits to the "infection" state
`flags`	a vector giving the counts of each check
`istate`	a constructed istate that best satisfies all the checks
`overlap`	a list with the row number and id of overlaps (not present if there are no overlaps)
`gaps`	a list with the row number and id of gaps (not present if there are no gaps)
`teleport`	a list with the row number and id of inconsistent rows (not present if there are none)
`jumps`	a list with the row number and id of jumps (not present if there are no jumps)

Note

For data sets with time-dependent covariates, a given subject will often have intermediate rows with a status of 'no event at this time', coded as the first level of the factor variable in the Surv() call. For instance a subject who started in state 'a' at time 0, transitioned to state 'b' at time 10, had a covariate `x` change from 135 to 156 at time 20, and a final transition to state 'c' at time 30. The response would be `Surv(c(0, 10, a), c(10, 20, censor), c(20,0,c))` where the state variable is a factor with levels of censor, a, b, c. The state variable records *changes* in state, and there was no change at time 20. The `istate` variable would be (a, b, b); it contains the *current* state, and the value is unchanged when status = censored. (It behaves like a `tdc` variable from `tmerge`).

The intermediate time above is not actually censoring, i.e., a point at which follow-up for the observation ceases. The 'censor' label is traditional, but 'none' may be a more accurate choice.

When there are intermediate observations `istate` is not simply a lagged version of the state, and may be more challenging to create. One approach is to let `survcheck` do the work: call it with an `istate` argument that is correct for the first row of each subject, or no `istate` argument at all, and then insert the returned value into a data frame.

survcondense *Shorten a (time1, time2) survival dataset*

Description

Counting process data sets can sometimes grow to an uweildy size, this can be used to reduce the number of rows.

Usage

```
survcondense(formula, data, subset, weights, na.action= na.pass, id,
             start = "tstart", end = "tstop", event = "event")
```

Arguments

formula	a formula object, with the response on the left of a ~ operator, and the terms on the right. The response must be a survival object as returned by the Surv function.
data	a data.frame in which to interpret the variables named in the formula and the id argument argument.
subset	optional subset expression to apply to the data set
weights	optional variable name for case weights
na.action	optional removal of missing values
id	variable name that identifies subjects
start	optional character string, giving the name of the start time variable in the result
end	optional character string, giving the name of the stop time variable in the result
event	optional character string, giving the name of the event variable in the result

Details

Through the use of the survSplit and tmerge functions, a counting process data set will gain more and more rows of data. Occassionally it is useful to collapse this surplus back down, e.g., when interest is to be focused on only a few covariates, or for debugging. The right hand side of formula will often have only a few variables in this use case.

If a row of data is censored, and represents the same covariates and identifier as the row below it, then the two rows can be merged together using a single (time1, time2) interval. The compression can sometimes be large.

The start, stop and end options are only used when the left hand side of the formula has expressions that are not a simple name, e.g. Surv(time1, time2, death | progression) would be a case where event is used to set the outcome variable's name.

Value

a data frame

Author(s)

Terry Therneau

See Also

survSplit,tmerge

Examples

```
dim(aml)
test1 <- survSplit(Surv(time, status) ~ ., data=aml,
                   cut=c(10, 20, 30), id="newid")
dim(test1)

# remove the added rows
test2 <- survcondense(Surv(tstart, time, status) ~ x, test1, id=newid)
dim(test2)
```

---

survdiff *Test Survival Curve Differences*

Description

Tests if there is a difference between two or more survival curves using the G^ρ family of tests, or for a single curve against a known alternative.

Usage

```
survdiff(formula, data, subset, na.action, rho=0, timefix=TRUE)
```

Arguments

formula	a formula expression as for other survival models, of the form `Surv(time, status) ~ predictors`. For a one-sample test, the predictors must consist of a single `offset(sp)` term, where sp is a vector giving the survival probability of each subject. For a k-sample test, each unique combination of predictors defines a subgroup. A `strata` term may be used to produce a stratified test. To cause missing values in the predictors to be treated as a separate group, rather than being omitted, use the `strata` function with its `na.group=T` argument.
data	an optional data frame in which to interpret the variables occurring in the formula.
subset	expression indicating which subset of the rows of data should be used in the fit. This can be a logical vector (which is replicated to have length equal to the number of observations), a numeric vector indicating which observation numbers are to be included (or excluded if negative), or a character vector of row names to be included. All observations are included by default.

na.action	a missing-data filter function. This is applied to the model.frame after any subset argument has been used. Default is options()$na.action.
rho	a scalar parameter that controls the type of test.
timefix	process times through the aeqSurv function to eliminate potential roundoff issues.

Value

a list with components:

n	the number of subjects in each group.
obs	the weighted observed number of events in each group. If there are strata, this will be a matrix with one column per stratum.
exp	the weighted expected number of events in each group. If there are strata, this will be a matrix with one column per stratum.
chisq	the chisquare statistic for a test of equality.
var	the variance matrix of the test.
strata	optionally, the number of subjects contained in each stratum.
pvalue	the p-value corresponding to the Chisquare statistic

Description

This function implements the G-rho family of Harrington and Fleming (1982), with weights on each death of $S(t)^\rho$, where $S(t)$ is the Kaplan-Meier estimate of survival. With rho = 0 this is the log-rank or Mantel-Haenszel test, and with rho = 1 it is equivalent to the Peto & Peto modification of the Gehan-Wilcoxon test.

Peto and Peto show that the Gehan-Wilcoxon test can be badly biased if the two groups have different censoring patterns, and proposed an alternative. Prentice and Marek later showed an actual example where this issue occurs. For most data sets the Gehan-Wilcoxon and Peto-Peto-Prentice variant will hardly differ, however.

If the right hand side of the formula consists only of an offset term, then a one sample test is done. To cause missing values in the predictors to be treated as a separate group, rather than being omitted, use the factor function with its exclude argument to recode the righ-hand-side covariate.

Note that the ordinary log-rank test is equivalent to the score test from a Cox model, using the Breslow approximation for ties. Use the Cox model form for more complex models, e.g., time-dependent covariates.

References

Harrington, D. P. and Fleming, T. R. (1982). A class of rank test procedures for censored survival data. Biometrika, 553-566.

Peto R. Peto and Peto, J. (1972) Asymptotically efficient rank invariant test procedures (with discussion), JRSSA, 185-206.

Prentice, R. and Marek, P. (1979) A qualitative discrepancy between censored data rank tests, Biometics, 861–867.

Examples

```
## Two-sample test
survdiff(Surv(futime, fustat) ~ rx,data=ovarian)
check <- coxph(Surv(futime, fustat) ~ rx, data=ovarian, ties="breslow")
round(summary(check)$sctest, 3)

## Stratified 8-sample test (7 df)
survdiff(Surv(time, status) ~ pat.karno + strata(inst), data=lung)
check <- coxph(Surv(time, status) ~ factor(pat.karno) + strata(inst), lung)
round(summary(check)$sctest, 3)

## Expected survival for heart transplant patients based on
## US mortality tables
expect <- survexp(futime ~ 1, data=jasa, cohort=FALSE,
                  rmap= list(age=(accept.dt - birth.dt), sex=1, year=accept.dt),
                  ratetable=survexp.us)
## actual survival is much worse (no surprise)
survdiff(Surv(jasa$futime, jasa$fustat) ~ offset(expect))

# The free light chain data set is close to the population.
e2 <- survexp(futime ~ 1, data=flchain, cohort=FALSE,
              rmap= list(age= age*365.25, sex=sex,
                         year=as.Date(paste0(sample.yr, "-07-01"))),
              ratetable= survexp.mn)
survdiff(Surv(futime, death) ~ offset(e2), flchain)
```

`survexp` *Compute Expected Survival*

Description

Returns either the expected survival of a cohort of subjects, or the individual expected survival for each subject.

Usage

```
survexp(formula, data, weights, subset, na.action, rmap, times,
        method=c("ederer", "hakulinen", "conditional", "individual.h",
                 "individual.s"),
        cohort=TRUE, conditional=FALSE,
        ratetable=survival::survexp.us, scale=1,
        se.fit, model=FALSE, x=FALSE, y=FALSE)
```

Arguments

`formula`	formula object. The response variable is a vector of follow-up times and is optional. The predictors consist of optional grouping variables separated by the + operator (as in `survfit`), and is often ~1, i.e., expected survival for the entire group.

data	data frame in which to interpret the variables named in the formula, subset and weights arguments.
weights	case weights. This is most useful when conditional survival for a known population is desired, e.g., the data set would contain all unique age/sex combinations and the weights would be the proportion of each.
subset	expression indicating a subset of the rows of data to be used in the fit.
na.action	function to filter missing data. This is applied to the model frame after subset has been applied. Default is options()$na.action.
rmap	an optional list that maps data set names to the ratetable names. See the details section below.
times	vector of follow-up times at which the resulting survival curve is evaluated. If absent, the result will be reported for each unique value of the vector of times supplied in the response value of the formula.
method	computational method for the creating the survival curves. The individual option does not create a curve, rather it retrieves the predicted survival individual.s or cumulative hazard individual.h for each subject. The default is to use method='ederer' if the formula has no response, and method='hakulinen' otherwise.
cohort	logical value. This argument has been superseded by the method argument. To maintain backwards compatability, if is present and FALSE, it implies method='individual.s'.
conditional	logical value. This argument has been superseded by the method argument. To maintain backwards compatability, if it is present and TRUE it implies method='conditional'.
ratetable	a table of event rates, such as survexp.mn, or a fitted Cox model. Note the survival:: prefix in the default argument is present to avoid the (rare) case of a user who expects the default table but just happens to have an object named "survexp.us" in their own directory.
scale	numeric value to scale the results. If ratetable is in units/day, scale = 365.25 causes the output to be reported in years.
se.fit	compute the standard error of the predicted survival. This argument is currently ignored. Standard errors are not a defined concept for population rate tables (they are treated as coming from a complete census), and for Cox models the calculation is hard. Despite good intentions standard errors for this latter case have not been coded and validated.
model, x, y	flags to control what is returned. If any of these is true, then the model frame, the model matrix, and/or the vector of response times will be returned as components of the final result, with the same names as the flag arguments.

Details

Individual expected survival is usually used in models or testing, to 'correct' for the age and sex composition of a group of subjects. For instance, assume that birth date, entry date into the study, sex and actual survival time are all known for a group of subjects. The survexp.us population tables contain expected death rates based on calendar year, sex and age. Then

```
haz <- survexp(fu.time ~ 1, data=mydata,
                   rmap = list(year=entry.dt, age=(birth.dt-entry.dt)),
                   method='individual.h'))
```

gives for each subject the total hazard experienced up to their observed death time or last follow-up time (variable fu.time) This probability can be used as a rescaled time value in models:

```
glm(status ~ 1 + offset(log(haz)), family=poisson)
glm(status ~ x + offset(log(haz)), family=poisson)
```

In the first model, a test for intercept=0 is the one sample log-rank test of whether the observed group of subjects has equivalent survival to the baseline population. The second model tests for an effect of variable x after adjustment for age and sex.

The ratetable being used may have different variable names than the user's data set, this is dealt with by the rmap argument. The rate table for the above calculation was survexp.us, a call to summary{survexp.us} reveals that it expects to have variables age = age in days, sex, and year = the date of study entry, we create them in the rmap line. The sex variable was not mapped, therefore the function assumes that it exists in mydata in the correct format. (Note: for factors such as sex, the program will match on any unique abbreviation, ignoring case.)

Cohort survival is used to produce an overall survival curve. This is then added to the Kaplan-Meier plot of the study group for visual comparison between these subjects and the population at large. There are three common methods of computing cohort survival. In the "exact method" of Ederer the cohort is not censored, for this case no response variable is required in the formula. Hakulinen recommends censoring the cohort at the anticipated censoring time of each patient, and Verheul recommends censoring the cohort at the actual observation time of each patient. The last of these is the conditional method. These are obtained by using the respective time values as the follow-up time or response in the formula.

Value

if cohort=TRUE an object of class survexp, otherwise a vector of per-subject expected survival values. The former contains the number of subjects at risk and the expected survival for the cohort at each requested time. The cohort survival is the hypothetical survival for a cohort of subjects enrolled from the population at large, but matching the data set on the factors found in the rate table.

References

Berry, G. (1983). The analysis of mortality by the subject-years method. *Biometrics*, 39:173-84.

Ederer, F., Axtell, L. and Cutler, S. (1961). The relative survival rate: a statistical methodology. *Natl Cancer Inst Monogr*, 6:101-21.

Hakulinen, T. (1982). Cancer survival corrected for heterogeneity in patient withdrawal. *Biometrics*, 38:933-942.

Therneau, T. and Grambsch, P. (2000). Modeling survival data: Extending the Cox model. Springer. Chapter 10.

Verheul, H., Dekker, E., Bossuyt, P., Moulijn, A. and Dunning, A. (1993). Background mortality in clinical survival studies. *Lancet*, 341: 872-875.

See Also

survfit, pyears, survexp.us, ratetable, survexp.fit.

Examples

```
#
# Stanford heart transplant data
#  We don't have sex in the data set, but know it to be nearly all males.
# Estimate of conditional survival
fit1 <- survexp(futime ~ 1, rmap=list(sex="male", year=accept.dt,
          age=(accept.dt-birth.dt)), method='conditional', data=jasa)
summary(fit1, times=1:10*182.5, scale=365) #expected survival by 1/2 years

# Estimate of expected  survival stratified by prior surgery
survexp(~ surgery, rmap= list(sex="male", year=accept.dt,
age=(accept.dt-birth.dt)), method='ederer', data=jasa,
        times=1:10 * 182.5)

## Compare the survival curves for the Mayo PBC data to Cox model fit
##
pfit <-coxph(Surv(time,status>0) ~ trt + log(bili) + log(protime) + age +
               platelet, data=pbc)
plot(survfit(Surv(time, status>0) ~ trt, data=pbc), mark.time=FALSE)
lines(survexp( ~ trt, ratetable=pfit, data=pbc), col='purple')
```

`survexp.fit` *Compute Expected Survival*

Description

Compute expected survival times.

Usage

```
survexp.fit(group, x, y, times, death, ratetable)
```

Arguments

`group`	if there are multiple survival curves this identifies the group, otherwise it is a constant. Must be an integer.
`x`	A matrix whose columns match the dimensions of the `ratetable`, in the correct order.
`y`	the follow up time for each subject.
`times`	the vector of times at which a result will be computed.
`death`	a logical value, if `TRUE` the conditional survival is computed, if `FALSE` the cohort survival is computed. See `survexp` for more details.
`ratetable`	a rate table, such as `survexp.uswhite`.

Details

For conditional survival y must be the time of last follow-up or death for each subject. For cohort survival it must be the potential censoring time for each subject, ignoring death.

For an exact estimate times should be a superset of y, so that each subject at risk is at risk for the entire sub-interval of time. For a large data set, however, this can use an inordinate amount of storage and/or compute time. If the times spacing is more coarse than this, an actuarial approximation is used which should, however, be extremely accurate as long as all of the returned values are > .99.

For a subgroup of size 1 and times > y, the conditional method reduces to exp(-h) where h is the expected cumulative hazard for the subject over his/her observation time. This is used to compute individual expected survival.

Value

A list containing the number of subjects and the expected survival(s) at each time point. If there are multiple groups, these will be matrices with one column per group.

Warning

Most users will call the higher level routine survexp. Consequently, this function has very few error checks on its input arguments.

See Also

survexp, survexp.us.

---

survexp.object *Expected Survival Curve Object*

Description

This class of objects is returned by the survexp class of functions to represent a fitted survival curve.

Objects of this class have methods for summary, and inherit the print, plot, points and lines methods from survfit.

Arguments

surv	the estimate of survival at time t+0. This may be a vector or a matrix.
n.risk	the number of subjects who contribute at this time.
time	the time points at which the curve has a step.
std.err	the standard error of the cumulative hazard or -log(survival).
strata	if there are multiple curves, this component gives the number of elements of the time etc. vectors corresponding to the first curve, the second curve, and so on. The names of the elements are labels for the curves.
method	the estimation method used. One of "Ederer", "Hakulinen", or "conditional".

na.action	the returned value from the na.action function, if any. It will be used in the printout of the curve, e.g., the number of observations deleted due to missing values.
call	an image of the call that produced the object.

Structure

The following components must be included in a legitimate survfit object.

Subscripts

Survexp objects that contain multiple survival curves can be subscripted. This is most often used to plot a subset of the curves.

Details

In expected survival each subject from the data set is matched to a hypothetical person from the parent population, matched on the characteristics of the parent population. The number at risk printed here is the number of those hypothetical subject who are still part of the calculation. In particular, for the Ederer method all hypotheticals are retained for all time, so n.risk will be a constant.

See Also

plot.survfit, summary.survexp, print.survfit, survexp.

---

survfit *Create survival curves*

Description

This function creates survival curves from either a formula (e.g. the Kaplan-Meier), a previously fitted Cox model, or a previously fitted accelerated failure time model.

Usage

```
survfit(formula, ...)
```

Arguments

formula	either a formula or a previously fitted model
...	other arguments to the specific method

Details

A survival curve is based on a tabulation of the number at risk and number of events at each unique death time. When time is a floating point number the definition of "unique" is subject to interpretation. The code uses factor() to define the set. For further details see the documentation for the appropriate method, i.e., `?survfit.formula` or `?survfit.coxph`.

A survfit object may contain a single curve, a set of curves (vector), a matrix of curves, or even a 3 way array: `dim(fit)` will reveal the dimensions. Predicted curves from a `coxph` model have one row for each stratum in the Cox model fit and one column for each specified covariate set. Curves from a multi-state model have one row for each stratum and a column for each state, the strata correspond to predictors on the right hand side of the equation. The default printing and plotting order for curves is by column, as with other matrices.

Value

An object of class `survfit` containing one or more survival curves.

Note

Older releases of the code also allowed the specification for a single curve to omit the right hand of the formula, i.e., `survfit(Surv(time, status))`, in which case the formula argument is not actually a formula. Handling this case required some non-standard and fairly fragile manipulations, and this case is no longer supported.

Author(s)

Terry Therneau

See Also

`survfit.formula`, `survfit.coxph`, `survfit.object`, `print.survfit`, `plot.survfit`, `quantile.survfit`, `residuals.survfit`, `summary.survfit`

`survfit.coxph` *Compute a Survival Curve from a Cox model*

Description

Computes the predicted survivor function for a Cox proportional hazards model.

Usage

```
## S3 method for class 'coxph'
survfit(formula, newdata,
        se.fit=TRUE, conf.int=.95, individual=FALSE, stype=2, ctype,
        conf.type=c("log","log-log","plain","none", "logit", "arcsin"),
        censor=TRUE, start.time, id, influence=FALSE,
        na.action=na.pass, type, time0=FALSE, ...)
```

```
## S3 method for class 'coxphms'
survfit(formula, newdata,
        se.fit=FALSE, conf.int=.95, individual=FALSE, stype=2, ctype,
        conf.type=c("log","log-log","plain","none", "logit", "arcsin"),
        censor=TRUE, start.time, id, influence=FALSE,
        na.action=na.pass, type, p0=NULL, time0= FALSE, ...)
```

Arguments

formula	A coxph object.
newdata	a data frame with the same variable names as those that appear in the coxph formula. One curve is produced per row. The curve(s) produced will be representative of a cohort whose covariates correspond to the values in newdata.
se.fit	a logical value indicating whether standard errors should be computed. Default is TRUE for standard models, FALSE for multi-state (code not yet present for that case.)
conf.int	the level for a two-sided confidence interval on the survival curve(s). Default is 0.95.
individual	deprecated argument, replaced by the general id
stype	computation of the survival curve, 1=direct, 2= exponenial of the cumulative hazard.
ctype	whether the cumulative hazard computation should have a correction for ties, 1=no, 2=yes.
conf.type	One of "none", "plain", "log" (the default), "log-log" or "logit". Only enough of the string to uniquely identify it is necessary. The first option causes confidence intervals not to be generated. The second causes the standard intervals curve +- k *se(curve), where k is determined from conf.int. The log option calculates intervals based on the cumulative hazard or log(survival). The log-log option uses the log hazard or log(-log(survival)), and the logit log(survival/(1-survival)).
censor	if FALSE time points at which there are no events (only censoring) are not included in the result.
id	optional variable name of subject identifiers. If this is present, it will be search for in the newdata data frame. Each group of rows in newdata with the same subject id represents the covariate path through time of a single subject, and the result will contain one curve per subject. If the coxph fit had strata then that must also be specified in newdata. If newid is not present, then each individual row of newdata is presumed to represent a distinct subject.
start.time	optional starting time, a single numeric value. If present the returned curve contains survival after start.time conditional on surviving to start.time.
influence	option to return the influence values
na.action	the na.action to be used on the newdata argument
type	older argument that encompassed stype and ctype, now deprecated
p0	optional, a vector of probabilities. The returned curve will be for a cohort with this mixture of starting states. Most often a single state is chosen

time0	include the starting time for the curve in the output
...	for future methods

Details

This routine produces Pr(state) curves based on a coxph model fit. For single state models it produces the single curve for S(t) = Pr(remain in initial state at time t), known as the survival curve; for multi-state models a matrix giving probabilities for all states. The stype argument states the type of estimate, and defaults to the exponential of the cumulative hazard, better known as the Breslow estimate. For a multi-state Cox model this involves the exponential of a matrix. The argument stype=1 uses a non-exponential or 'direct' estimate. For a single endpoint coxph model the code evaluates the Kalbfleich-Prentice estimate, and for a multi-state model it uses an analog of the Aalen-Johansen estimator. The latter approach is the default in the mstate package.

The ctype option affects the estimated cumulative hazard, and if stype=2 the estimated P(state) curves as well. If not present it is chosen so as to be concordant with the ties option in the coxph call. (For multistate coxphms objects, only ctype=1 is currently implemented.) Likewise the choice between a model based and robust variance estimate for the curve will mirror the choice made in the coxph call, any clustering is also inherited from the parent model.

If the newdata argument is missing, then a curve is produced for a single "pseudo" subject with covariate values equal to the means component of the fit. The resulting curve(s) rarely make scientific sense, but the default remains due to an unwarranted belief by many that it represents an "average" curve, and it's use as a default in other packages. For coxph, the means component will contain the value 0 for any 0/1 or TRUE/FALSE variables, and the mean value in the data for others. Its primary reason for this default is to increase numerical accuracy in internal computations of the routine via recentering the X matrix; there is no reason to assume this represents an 'interesting' hypothetical subject for prediction of their survival curve. Users are strongly advised to use the newdata argument; predictions from a multistate coxph model require the newdata argument.

If the coxph model contained an offset term, then the data set in the newdata argument should also contain that variable.

When the original model contains time-dependent covariates, then the path of that covariate through time needs to be specified in order to obtain a predicted curve. This requires newdata to contain multiple lines for each hypothetical subject which gives the covariate values, time interval, and strata for each line (a subject can change strata), along with an id variable which demarks which rows belong to each subject. The time interval must have the same (start, stop, status) variables as the original model: although the status variable is not used and thus can be set to a dummy value of 0 or 1, it is necessary for the response to be recognized as a Surv object. Last, although predictions with a time-dependent covariate path can be useful, it is very easy to create a prediction that is senseless. Users are encouraged to seek out a text that discusses the issue in detail.

When a model contains strata but no time-dependent covariates the user of this routine has a choice. If newdata argument does not contain strata variables then the returned object will be a matrix of survival curves with one row for each strata in the model and one column for each row in newdata. (This is the historical behavior of the routine.) If newdata does contain strata variables, then the result will contain one curve per row of newdata, based on the indicated stratum of the original model. In the rare case of a model with strata by covariate interactions the strata variable must be included in newdata, the routine does not allow it to be omitted (predictions become too confusing). (Note that the model Surv(time, status) ~ age*strata(sex) expands internally to strata(sex) + age:sex; the sex variable is needed for the second term of the model.)

See survfit for more details about the counts (number of events, number at risk, etc.)

Value

an object of class "survfit". See survfit.object for details. Methods defined for survfit objects are print, plot, lines, and points.

Notes

If the following pair of lines is used inside of another function then the model=TRUE argument must be added to the coxph call: fit <- coxph(...); survfit(fit). This is a consequence of the non-standard evaluation process used by the model.frame function when a formula is involved.

Let $\log[S(t;z)]$ be the log of the survival curve for a fixed covariate vector z, then $\log[S(t;x)] = e^{(x-z)\beta}\log[S(t;z)]$ is the log of the curve for any new covariate vector x. There is an unfortunate tendency to refer to the reference curve with $z = 0$ as 'THE' baseline hazard. However, any z can be used as the reference point, and more importantly, if $x - z$ is large the compuation can suffer severe roundoff error. It is always safest to provide the desired x values directly via newdata.

References

Fleming, T. H. and Harrington, D. P. (1984). Nonparametric estimation of the survival distribution in censored data. *Comm. in Statistics* **13**, 2469-86.

Kalbfleisch, J. D. and Prentice, R. L. (1980). *The Statistical Analysis of Failure Time Data.* New York:Wiley.

Link, C. L. (1984). Confidence intervals for the survival function using Cox's proportional hazards model with covariates. *Biometrics* **40**, 601-610.

Therneau T and Grambsch P (2000), Modeling Survival Data: Extending the Cox Model, Springer-Verlag.

Tsiatis, A. (1981). A large sample study of the estimate for the integrated hazard function in Cox's regression model for survival data. *Annals of Statistics* **9**, 93-108.

See Also

print.survfit, plot.survfit, lines.survfit, coxph, Surv, strata.

---

survfit.formula *Compute a Survival Curve for Censored Data*

Description

Computes an estimate of a survival curve for censored data using the Aalen-Johansen estimator. For ordinary (single event) survival this reduces to the Kaplan-Meier estimate.

Usage

```
## S3 method for class 'formula'
survfit(formula, data, weights, subset, na.action,
        stype=1, ctype=1, id, cluster, robust, istate, timefix=TRUE,
        etype, model=FALSE, error, entry=FALSE, time0=FALSE, ...)
```

Arguments

formula	a formula object, which must have a Surv object as the response on the left of the ~ operator and, if desired, terms separated by + operators on the right. One of the terms may be a strata object. For a single survival curve the right hand side should be ~ 1.
data	a data frame in which to interpret the variables named in the formula, subset and weights arguments.
weights	The weights must be nonnegative and it is strongly recommended that they be strictly positive, since zero weights are ambiguous, compared to use of the subset argument.
subset	expression saying that only a subset of the rows of the data should be used in the fit.
na.action	a missing-data filter function, applied to the model frame, after any subset argument has been used. Default is options()$na.action.
stype	the method to be used estimation of the survival curve: 1 = direct, 2 = exp(cumulative hazard).
ctype	the method to be used for estimation of the cumulative hazard: 1 = Nelson-Aalen formula, 2 = Fleming-Harrington correction for tied events.
id	identifies individual subjects, when a given person can have multiple lines of data.
cluster	used to group observations for the infinitesimal jackknife variance estimate, defaults to the value of id.
robust	logical, should the function compute a robust variance. For multi-state survival curves or interval censored data this is true by default. For single state data see details, below.
istate	for multi-state models, identifies the initial state of each subject or observation. This also forces time0 =TRUE.
timefix	process times through the aeqSurv function to eliminate potential roundoff issues.
etype	a variable giving the type of event. This has been superseded by multi-state Surv objects and is deprecated; see example below.
model	include a copy of the model frame in the output
error	this argument is no longer used
entry	if TRUE, the output will contain n.enter which is the number of observations entering the risk set at any time; extra rows of output are created, if needed, for each unique entry time. Only applicable if there is an id statement.

time0 if TRUE, the output will include estimates at the starting point of the curve or 'time 0'. See discussion below.

... The following additional arguments are passed to internal functions called by survfit.

se.fit a logical value indicating whether standard errors should be computed. Default is TRUE. For a multistate model, where the infinitesimal jackknife (robust) standar error is used, the compute time for the standard error is O(ndp) where n = number of observations, d = number of events and p = number of states, while that for all other portions of the output (pstate, cumhaz and counts) is O((n+d)p). For a moderate to large data set the compute time difference between nd and n+d can be huge; using se.fit = FALSE may be a wise choice.

conf.type One of "none", "plain", "log" (the default), "log-log", "logit" or "arcsin". Only enough of the string to uniquely identify it is necessary. The first option causes confidence intervals not to be generated. The second causes the standard intervals curve +- k *se(curve), where k is determined from conf.int. The log option calculates intervals based on the cumulative hazard or log(survival). The log-log option bases the intervals on the log hazard or log(-log(survival)), the logit option on log(survival/(1-survival)) and arcsin on arcsin(sqrt(survival)).

conf.lower a character string to specify modified lower limits to the curve, the upper limit remains unchanged. Possible values are "usual" (unmodified), "peto", and "modified". The modified lower limit is based on an "effective n" argument. The confidence bands will agree with the usual calculation at each death time, but unlike the usual bands the confidence interval becomes wider at each censored observation. The extra width is obtained by multiplying the usual variance by a factor m/n, where n is the number currently at risk and m is the number at risk at the last death time. (The bands thus agree with the un-modified bands at each death time.) This is especially useful for survival curves with a long flat tail.

The Peto lower limit is based on the same "effective n" argument as the modified limit, but also replaces the usual Greenwood variance term with a simple approximation. It is known to be conservative.

start.time numeric value specifying a time to start calculating survival information. The resulting curve is the survival conditional on surviving to start.time.

conf.int the level for a two-sided confidence interval on the survival curve(s). Default is 0.95.

influence a logical value indicating whether to return the infinitesimal jackknife (influence) values for each subject. See details below.

p0 this applies only to multi-state curves. An optional vector giving the initial probability across the states. If this is missing, then p0 is estimated using the frequency of the starting states of all observations at risk at start.time, or if that is not specified, at the time of the first event.

entry by default, the survfit routines only return information at the event/censoring times. If entry=TRUE then also return a n.enter component containing the number who joined the risk set at each time; if necessary add extra rows to

the output for each unique entry time. This is only applicable for (time1, time2) survival data, and if there is an `id` statement. If a single subject had times of (0,10), (10, 20), (25,30) with an event at 30, then time 10 is not an entry or censoring time, but 20 counts as censored and 25 as an entry.

type an older argument that combined `stype` and `ctype`, now deprecated. Legal values were "kaplan-meier" which is equivalent to `stype=1, ctype=1`, "fleming-harrington" which is equivalent to `stype=2, ctype=1`, and "fh2" which is equivalent to `stype=2, ctype=2`.

Details

If there is a `data` argument, then variables in the `formula`, `weights`, `subset`, `id`, `cluster` and `istate` arguments will be searched for in that data set.

The routine returns both an estimated probability in state and an estimated cumulative hazard estimate. For simple survival the probability in state = probability alive, i.e, the estimated survival. For multi-state it will be a matrix with one row per time and a column per state, rows sum to 1. The cumulative hazard estimate is the Nelson-Aalen (NA) estimate or the Fleming-Harrington (FH) estimate, the latter includes a correction for tied event times. The estimated probability in state can estimated either using the exponential of the cumulative hazard, or as a direct estimate using the Aalen-Johansen approach. For single state data the AJ estimate reduces to the Kaplan-Meier and the probability in state to the survival curve; for competing risks data the AJ reduces to the cumulative incidence (CI) estimator. For backward compatability the `type` argument can be used instead.

When the data set includes left censored or interval censored data (or both), then the EM approach of Turnbull is used to compute the overall curve. Currently this algorithm is very slow, only applies to simple survival (not multi-state), and defaults to a robust variance. Other R packages are available which implement the iterative convex minorant (ICM) algorithm for interval censored data, which is much faster than Turnbull's method. Based on Sun (2001) the robust variance may be preferred, as the naive estimate ignores the estimation of the weights. The standard estimate can be obtained with `robust= FALSE`.

Without interval or left censored data (the usual case) the underlying algorithm for the routine is the Aalen-Johansen estimate, of which the Kaplan-Meier (for single outcome data) and the cumulative incidence (CI) estimate (for competing risks) are each a special case. For multi-state, the estimate can be written as $p(t_0)H(t_1)H(t_2)\dots$ where $p(t_0)$ is the prevalance vector across the states at starting point t_0, $t_1, t_2, \dots$ are the times at which events (transitions between states) occur, and H are square transtion matrices with a row for each state.

Starting point: When diffent subjects (`id`) start at different time points, data using age as the time scale for instance, deciding the default "time 0" can be complex. This value is the starting point for the restricted mean estimate (area under the curve), the initial prevalence p0, and the first row of output if `time0 = TRUE`. The order of the decision is

1. For a 2 column response (simple survival or competing risks) use the minimum of 0 and the smallest time value (times can be negative).
2. If all subjects start in the same state, start at the same time, or if `p0` is specified, use the minimum observed starting time. If there is no `istate` argument all observations are assumed to start in a state "(s0)".
3. Use the minimum observed event time, if the number at risk at that time is >0 for every curve that will be created.

4. Use the minimum event time for each curve, separately.

The last two above are a failsafe to prevent the routine from basing the initial prevalence of the states on none or only a handful of observations. That does not mean such curves will be scientfically sensible: when using age scale the user may wish to specify an explicit starting time. If `time0 = TRUE` the first row of output for each curve will be at the starting time, otherwise the first event time (for each curve separately).

Robust variance: If a `robust` is TRUE, or for multi-state curves, then the standard errors of the results will be based on an infinitesimal jackknife (IJ) estimate, otherwise the standard model based estimate will be used. For single state curves, the default for `robust` will be TRUE if one of: there is a `cluster` argument, there are non-integer weights, or there is a `id` statement and at least one of the id values has multiple events, and FALSE otherwise. The default represents our best guess about when one would most often desire a robust variance. When there are non-integer case weights and (time1, time2) survival data the routine is at an impasse: a robust variance likely is called for, but requires either `id` or `cluster` information to be done correctly; it will default to robust=FALSE if they are not present.

With the IJ estimate, the leverage values themselves can be returned as an array using the `influence` argument. Be forwarned that this array can be huge. Post fit influence using the `resid` method is more flexible and would normally be preferred, in particular to get influence at only a select set of time points. The `influence` option is currently used mostly in the package's validity checks.

Let $U(t)$ be the matrix of IJ values at time t, which has one row per observation, one column per state. The robust variance compuation uses the collapsed weighted matrix `rowsum(wU, cluster)`, where w is the vector of weights and cluster is the grouping (most often the id). The result for each curve is an array with dimensions (number of clusters, number of states, number of times), or a matrix for single state data. When there are multiple curves, the influence is a list with one element per curve.

All of weights, subset and offset are evaluated in the same way as variables in formula, that is first in data and then in the environment of formula. Note that values calculated inside the formula, such as mean(x), are evaluated before subsetting - which may lead to unexpected results if used with subset. For more information see the Details section of `model.frame`.

Value

an object of class `"survfit"`. See `survfit.object` for details. Some of the methods defined for survfit objects are `print`, `plot`, `lines`, `points` and `residual`.

References

Dorey, F. J. and Korn, E. L. (1987). Effective sample sizes for confidence intervals for survival probabilities. *Statistics in Medicine* **6**, 679-87.

Fleming, T. H. and Harrington, D. P. (1984). Nonparametric estimation of the survival distribution in censored data. *Comm. in Statistics* **13**, 2469-86.

Kalbfleisch, J. D. and Prentice, R. L. (1980). *The Statistical Analysis of Failure Time Data.* New York:Wiley.

Kyle, R. A. (1997). Moncolonal gammopathy of undetermined significance and solitary plasmacytoma. Implications for progression to overt multiple myeloma}, *Hematology/Oncology Clinics N. Amer.* **11**, 71-87.

Link, C. L. (1984). Confidence intervals for the survival function using Cox's proportional hazards model with covariates. *Biometrics* **40**, 601-610.

Sun, J. (2001). Variance estimation of a survival function for interval-censored data. *Stat Med* **20**, 1949-1957.

Turnbull, B. W. (1974). Nonparametric estimation of a survivorship function with doubly censored data. *J Am Stat Assoc*, **69**, 169-173.

See Also

survfit.coxph for survival curves from Cox models, survfit.object for a description of the components of a survfit object, print.survfit, plot.survfit, lines.survfit, residuals.survfit, coxph, Surv.

Examples

```
#fit a Kaplan-Meier and plot it
fit <- survfit(Surv(time, status) ~ x, data = aml)
plot(fit, lty = 2:3)
legend(100, .8, c("Maintained", "Nonmaintained"), lty = 2:3)

#fit a Cox proportional hazards model and plot the
#predicted survival for a 60 year old
fit <- coxph(Surv(futime, fustat) ~ age, data = ovarian)
plot(survfit(fit, newdata=data.frame(age=60)),
     xscale=365.25, xlab = "Years", ylab="Survival")

# Here is the data set from Turnbull
#  There are no interval censored subjects, only left-censored (status=3),
#  right-censored (status 0) and observed events (status 1)
#
#                             Time
#                         1    2   3   4
# Type of observation
#           death        12    6   2   3
#          losses         3    2   0   3
#      late entry         2    4   2   5
#
tdata <- data.frame(time  =c(1,1,1,2,2,2,3,3,3,4,4,4),
                    status=rep(c(1,0,2),4),
                    n     =c(12,3,2,6,2,4,2,0,2,3,3,5))
fit  <- survfit(Surv(time, time, status, type='interval') ~1,
              data=tdata, weight=n)

#
# Three curves for patients with monoclonal gammopathy.
#  1. KM of time to PCM, ignoring death (statistically incorrect)
#  2. Competing risk curves (also known as "cumulative incidence")
#  3. Multi-state, showing Pr(in each state, at time t)
#
fitKM <- survfit(Surv(stop, event=='pcm') ~1, data=mgus1,
                    subset=(start==0))
```

```
fitCR <- survfit(Surv(stop, event) ~1,
                 data=mgus1, subset=(start==0))
fitMS <- survfit(Surv(start, stop, event) ~ 1, id=id, data=mgus1)
## Not run:
# CR curves show the competing risks
plot(fitCR, xscale=365.25, xmax=7300, mark.time=FALSE,
            col=2:3, xlab="Years post diagnosis of MGUS",
            ylab="P(state)")
lines(fitKM, fun='event', xmax=7300, mark.time=FALSE,
            conf.int=FALSE)
text(3652, .4, "Competing risk: death", col=3)
text(5840, .15,"Competing risk: progression", col=2)
text(5480, .30,"KM:prog")

## End(Not run)
```

`survfit.matrix` *Create Aalen-Johansen estimates of multi-state survival from a matrix of hazards.*

Description

This allows one to create the Aalen-Johansen estimate of P, a matrix with one column per state and one row per time, starting with the individual hazard estimates. Each row of P will sum to 1. Note that this routine has been superseded by the use of multi-state Cox models, and will eventually be removed.

Usage

```
## S3 method for class 'matrix'
survfit(formula, p0, method = c("discrete", "matexp"),
        start.time, ...)
```

Arguments

`formula`	a matrix of lists, each element of which is either NULL or a survival curve object.
`p0`	the initial state vector. The names of this vector are used as the names of the states in the output object. If there are multiple curves then `p0` can be a matrix with one row per curve.
`method`	use a product of discrete hazards, or a product of matrix exponentials. See details below.
`start.time`	optional; start the calculations at a given starting point
`...`	further arguments used by other survfit methods

Details

On input the matrix should contain a set of predicted curves for each possible transition, and NULL in other positions. Each of the predictions will have been obtained from the relevant Cox model. This approach for multistate curves is easy to use but has some caveats. First, the input curves must be consistent. The routine checks as best it can, but can easy be fooled. For instance, if one were to fit two Cox models, obtain predictions for males and females from one, and for treatment A and B from the other, this routine will create two curves but they are not meaningful. A second issue is that standard errors are not produced.

The names of the resulting states are taken from the names of the vector of initial state probabilities. If they are missing, then the dimnames of the input matrix are used, and lacking that the labels '1', '2', etc. are used.

For the usual Aalen-Johansen estimator the multiplier at each event time is the matrix of hazards H (also written as I + dA). When using predicted survival curves from a Cox model, however, it is possible to get predicted hazards that are greater than 1, which leads to probabilities less than 0. If the `method` argument is not supplied and the input curves are derived from a Cox model this routine instead uses the approximation expm(H-I) as the multiplier, which always gives valid probabilities. (This is also the standard approach for ordinary survival curves from a Cox model.)

Value

a survfitms object

Note

The R syntax for creating a matrix of lists is very fussy.

Author(s)

Terry Therneau

See Also

survfit

Examples

```
etime <- with(mgus2, ifelse(pstat==0, futime, ptime))
event <- with(mgus2, ifelse(pstat==0, 2*death, 1))
event <- factor(event, 0:2, labels=c("censor", "pcm", "death"))

cfit1 <- coxph(Surv(etime, event=="pcm") ~ age + sex, mgus2)
cfit2 <- coxph(Surv(etime, event=="death") ~ age + sex, mgus2)

# predicted competing risk curves for a 72 year old with mspike of 1.2
# (median values), male and female.
# The survfit call is a bit faster without standard errors.
newdata <- expand.grid(sex=c("F", "M"), age=72, mspike=1.2)

AJmat <- matrix(list(), 3,3)
AJmat[1,2] <- list(survfit(cfit1, newdata, std.err=FALSE))
```

```
AJmat[1,3] <- list(survfit(cfit2, newdata, std.err=FALSE))
csurv  <- survfit(AJmat, p0 =c(entry=1, PCM=0, death=0))
```

`survfit.object` *Survival Curve Object*

Description

This class of objects is returned by the `survfit` class of functions to represent a fitted survival curve. For a multi-state model the object has class `c('survfitms', 'survfit')`.

Objects of this class have methods for the functions `print`, `summary`, `plot`, `points` and `lines`. The `print.survfit` method does more computation than is typical for a print method and is documented on a separate page.

Arguments

`n`	total number of observations in each curve.
`time`	the time points at which the curve has a step.
`n.risk`	the number of subjects at risk at t.
`n.event`	the number of events that occur at time t.
`n.enter`	for counting process data only, and only if there was an `id` argument, the number of subjects that enter the risk set during the current interval. If there are event/censoring times at 1, 3, 5 for instance, someone who enters at time 1 is counted in the (1, 3] interval, i.e., appears in the row for time 3.
`n.censor`	for counting process data only, the number of subjects who exit the risk set, without an event, at time t. (For right censored data, this number can be computed from the successive values of the number at risk).
`surv`	the estimate of survival at time t+0. This may be a vector or a matrix. The latter occurs when a set of survival curves is created from a single Cox model, in which case there is one column for each covariate set.
`pstate`	a multi-state survival will have the `pstate` component instead of `surv`. It will be a matrix containing the estimated probability of each state at each time, one column per state.
`std.err`	for a survival curve this contains standard error of the cumulative hazard or -log(survival), for a multi-state curve it contains the standard error of prev. This difference is a reflection of the fact that each is the natural calculation for that case.
`cumhaz`	optional. Contains the cumulative hazard for each possible transition.
`counts`	optional. If weights were used, the `n.risk` etc elements contain weighted sums; the `counts` matrix will contain unweighted values. Weighted values are normally more useful for further computation, unweighted may be preferred for labeling or printout.

`strata` if there are multiple curves, this component gives the number of elements of the `time` vector corresponding to the first curve, the second curve, and so on. The names of the elements are labels for the curves.

`upper` optional upper (2-sided) confidence limit for the survival curve or probability in state

`lower` optional lower (2-sidedconfidence limit for the survival curve or probability in state

`t0` optional, the starting time for the curve

`p0`, `sp0` for a multistate object, the distribution of starting states. If the curve has a strata dimension, this will be a matrix one row per stratum. The `sp0` element has the standard error of p0, if p0 was estimated.

`newdata` for survival curves from a fitted model, this contains the covariate values for the curves

`n.id` the total number of unique id values that contributed to the curve. This is only available if the original call used the id option.

`conf.type` the approximation used to compute the confidence limits.

`conf.int` the level of the confidence limits, e.g. 90 or 95%.

`transitions` for multi-state data, the total number of transitions of each type.

`na.action` the returned value from the na.action function, if any. It will be used in the printout of the curve, e.g., the number of observations deleted due to missing values.

`call` an image of the call that produced the object.

`type` type of survival censoring.

`influence.p`, `influence.c` optional influence matrices for the `pstate` (or `surv`) and for the `cumhaz` estimates. A list with one element per stratum, each element of the list is an array indexed by subject, time, state.

`version` the version of the object. Will be missing, 2, or 3

Structure

The following components must be included in a legitimate `survfit` or `survfitms` object.

Subscripts

Survfit objects can be subscripted. This is often used to plot a subset of the curves, for instance. From the user's point of view the `survfit` object appears to be a vector, matrix, or array of curves. The first dimension is always the underlying number of curves or "strata"; for multi-state models the state is always the last dimension. Predicted curves from a Cox model can have a second dimension which is the number of different covariate prediction vectors.

Details

The survfit object has evolved over time: when first created there was no thought of multi-state models for instance. This evolution has almost entirely been accomplished by the addition of new elements.

For both plots of the curves and computation of the restricted mean time in state (RMTS) we need the concept of a starting point t0 and starting prevalence of the states p0 for each curve. (Sojourn time, area under the curve and restricted mean survival time are other labels for the RMTS). Time 0 is not, by default, included as part of the standard tableau of results, i.e., time, number at risk, number of events, etc. For simple survival with a 0/1 status variable, the starting state p0 is the obvious default of "everyone alive", and t0 is formally not discernable from the data and so was left out. (A design decision made in 1986, and now far too late to change.) However, for plots t0 is assumed to be the minimum of 0 and all observed times. Negative survival times are unusual but not invalid. Multi-state survival curves include t0 and p0 as a part of the returned object. The first is a single value for all curves, the second is per curve.

The survfit0 routine can be used to add these values to the main curve data, this is done by the default print, plot, and summary methods for survfit objects. The methods vignette has discussion of the rationale of how t0 and p0 are chosen in the multi-state case. Notice that if there is an event at time t0, e.g., a death on day 0 for competing risks, then p0 will contain the prevalence just before that event occured.

See Also

plot.survfit, summary.survfit, print.survfit, survfit, survfit0

---

survfit0 *Convert the format of a survfit object.*

Description

Add the point for a starting time ("time 0") to a survfit object's elements. This is useful for plotting.

Usage

```
survfit0(x, ...)
```

Arguments

x	a survfit object
...	any other arguments are ignored

Details

Survival curves are traditionally plotted forward from time 0, but since the true starting time is not known as a part of the data, the survfit routine does not include a time 0 value in the resulting object. Someone might look at cumulative mortgage defaults versus calendar year, for instance, with the 'time' value a Date object. The plotted curve probably should not start at 0 = 1970-01-01.

Due to this uncertainty, it was decided not to include a "time 0" as part of a survfit object. Whether that (1989) decision was wise or foolish, it is now far too late to change it. (We tried it once as a trial, resulting in over 20 errors in the survival test suite. We extrapolated that it might break 1/3 of the other CRAN packages that depend on survival, if made a default.) Many curves do include a value `t0` for "time 0", which is where the survfit routine has surmised that the curve would start.

One problem with this choice is that some functions must choose a starting point, plots and computation of the restricted mean survival time are two primary examples. This utility function is used by `plot.survfit` and `summary.survfit` to fill in that gap.

The value used for this first time point is the first one below

1. a `t0` value found in the in the object.
2. for single state survival
   - min(0, time) for Surv(time, status) data
   - min(time1) for Surv(time1, time2, status) data
3. for multi state survival
   - min(0, time) for Surv(time, event) data, e.g., competing risks
   - min(time1) for Surv(time1, time2, event) data, if everyone starts in the same state

(Remember that negative times are allowed in Surv objects.)

This function will add a new time point at the front of each curve, but only if said time point is less than existing points in the curve. If there were a death on day 0, for instance, it will not add a (time=0, survival=1) point. (The question of whether the plotted curve in this case should or should not start with a vertical segment can be debated ad nauseum. It has no effect on the area under the curve (RMST), and the summary for time 0 should report the smaller value.)

The resulting object is *not* currently guarranteed to work with functions that further manipulate a `survfit` object such as subscripting, aggregation, pseudovalues, etc. (remember the 20 errors). Rather it is intended as a penultimate step, most often when creating a plot or summary of the curve(s).

Value

a reformulated version of the object with an initial data point added. The `time`, `surv`, `pstate`, `cumhaz`, `std.err`, `std.cumhaz` and other components will all be aligned, so as to make plots and summaries easier to produce.

`survfitcoxph.fit` *A direct interface to the 'computational engine' of survfit.coxph*

Description

This program is mainly supplied to allow other packages to invoke the survfit.coxph function at a 'data' level rather than a 'user' level. It does no checks on the input data that is provided, which can lead to unexpected errors if that data is wrong.

Usage

```
survfitcoxph.fit(y, x, wt, x2, risk, newrisk, strata, se.fit, survtype,
vartype, varmat, id, y2, strata2, unlist=TRUE)
```

Arguments

y	the response variable used in the Cox model. (Missing values removed of course.)
x	covariate matrix used in the Cox model
wt	weight vector for the Cox model. If the model was unweighted use a vector of 1s.
x2	matrix describing the hypothetical subjects for which a curve is desired. Must have the same number of columns as x.
risk	the risk score exp(X beta) from the fitted Cox model. If the model had an offset, include it in the argument to exp.
newrisk	risk scores for the hypothetical subjects
strata	strata variable used in the Cox model. This will be a factor.
se.fit	if TRUE the standard errors of the curve(s) are returned
survtype	1=Kalbfleisch-Prentice, 2=Nelson-Aalen, 3=Efron. It is usual to match this to the approximation for ties used in the coxph model: KP for 'exact', N-A for 'breslow' and Efron for 'efron'.
vartype	1=Greenwood, 2=Aalen, 3=Efron
varmat	the variance matrix of the coefficients
id	optional; if present and not NULL this should be a vector of identifiers of length nrow(x2). A mon-null value signifies that x2 contains time dependent covariates, in which case this identifies which rows of x2 go with each subject.
y2	survival times, for time dependent prediction. It gives the time range (time1,time2] for each row of x2. Note: this must be a Surv object and thus contains a status indicator, which is never used in the routine, however.
strata2	vector of strata indicators for x2. This must be a factor.
unlist	if FALSE the result will be a list with one element for each strata. Otherwise the strata are "unpacked" into the form found in a survfit object.

Value

a list containing nearly all the components of a survfit object. All that is missing is to add the confidence intervals, the type of the original model's response (as in a coxph object), and the class.

Note

The source code for for both this function and survfit.coxph is written using noweb. For complete documentation see the inst/sourcecode.pdf file.

Author(s)

Terry Therneau

See Also

survfit.coxph

`survfit_confint` *Confidence intervals for survival curves*

Description

Compute the confidence intervals for a survfit object, using one of several approximations.

Usage

```
survfit_confint(p, se, logse = TRUE, conf.type, conf.int = 0.95, selow, ulimit = TRUE)
```

Arguments

p	the estimate, i.e., the surv, pstate or cumhaz component of a survfit object
se	vector or matrix of standard errors
logse	if TRUE, the provided se component is the std of log(p)
conf.type	one of "plain", "log", "log-log", "logit", or "arcsin"
conf.int	the confidence level, a value strictly between 0 and 1
selow	alternate standard error for the lower limit
ulimit	if TRUE, constrain the result to be between 0 and 1

Details

This routine is used internally to create confidence intervals and would rarely be called by a user (but perhaps by another package). As such it does very little checking of the input arguments.

"Plain" intervals of p +- 1.96*se have been shown by many authors to perform poorly. Which of the other choices is "best" depends on the details of the simulation study used to evaluate the method, all do quite well overall.

For log intervals, the CI at p=0 is exp(log(0) + se) which becomes NA in R, and likewise for p= 0 or 1 when using log-log or logit. If the se is 0, however, the routine returns p as the upper and lower limit. (One side effect of the NA is that the plotted se bands for a Kaplan-Meier whose final time point is a death will stop at the death time and not plunge to 0; i.e., not have the final stair step, a graphical aspect we prefer.)

Value

a list with components lower and upper.

See Also

survfit.object

---

survival-deprecated *Deprecated functions in package* **survival**

Description

These functions are temporarily retained for compatability with older programs, and may transition to defunct status.

Usage

```
survConcordance(formula, data, weights, subset, na.action) # use concordance
survConcordance.fit(y, x, strata, weight)    # use concordancefit
```

Arguments

formula	a formula object, with the response on the left of a ~ operator, and the terms on the right. The response must be a survival object as returned by the Surv function.
data	a data frame
weights, subset, na.action	as for coxph
x, y, strata, weight	predictor, response, strata, and weight vectors for the direct call

See Also

Deprecated

---

survobrien *O'Brien's Test for Association of a Single Variable with Survival*

Description

Peter O'Brien's test for association of a single variable with survival This test is proposed in Biometrics, June 1978.

Usage

```
survobrien(formula, data, subset, na.action, transform)
```

Arguments

formula	a valid formula for a cox model.
data	a data.frame in which to interpret the variables named in the formula, or in the subset and the weights argument.
subset	expression indicating which subset of the rows of data should be used in the fit. All observations are included by default.
na.action	a missing-data filter function. This is applied to the model.frame after any subset argument has been used. Default is options()\$na.action.
transform	the transformation function to be applied at each time point. The default is O'Brien's suggestion logit(tr) where tr = (rank(x)- 1/2)/ length(x) is the rank shifted to the range 0-1 and logit(x) = log(x/(1-x)) is the logit transform.

Value

a new data frame. The response variables will be column names returned by the Surv function, i.e., "time" and "status" for simple survival data, or "start", "stop", "status" for counting process data. Each individual event time is identified by the value of the variable .strata.. Other variables retain their original names. If a predictor variable is a factor or is protected with I(), it is retained as is. Other predictor variables have been replaced with time-dependent logit scores.

The new data frame will have many more rows that the original data, approximately the original number of rows * number of deaths/2.

Method

A time-dependent cox model can now be fit to the new data. The univariate statistic, as originally proposed, is equivalent to single variable score tests from the time-dependent model. This equivalence is the rationale for using the time dependent model as a multivariate extension of the original paper.

In O'Brien's method, the x variables are re-ranked at each death time. A simpler method, proposed by Prentice, ranks the data only once at the start. The results are usually similar.

Note

A prior version of the routine returned new time variables rather than a strata. Unfortunately, that strategy does not work if the original formula has a strata statement. This new data set will be the same size, but the coxph routine will process it slightly faster.

References

O'Brien, Peter, "A Nonparametric Test for Association with Censored Data", *Biometrics* 34: 243-250, 1978.

See Also

survdiff

Examples

```
xx <- survobrien(Surv(futime, fustat) ~ age + factor(rx) + I(ecog.ps),
       data=ovarian)
coxph(Surv(time, status) ~ age + strata(.strata.), data=xx)
```

---

survreg *Regression for a Parametric Survival Model*

Description

Fit a parametric survival regression model. These are location-scale models for an arbitrary transform of the time variable; the most common cases use a log transformation, leading to accelerated failure time models.

Usage

```
survreg(formula, data, weights, subset,
        na.action, dist="weibull", init=NULL, scale=0,
        control,parms=NULL,model=FALSE, x=FALSE,
        y=TRUE, robust=FALSE, cluster, score=FALSE, ...)
```

Arguments

formula	a formula expression as for other regression models. The response is usually a survival object as returned by the Surv function. See the documentation for Surv, lm and formula for details.
data	a data frame in which to interpret the variables named in the formula, weights or the subset arguments.
weights	optional vector of case weights
subset	subset of the observations to be used in the fit
na.action	a missing-data filter function, applied to the model.frame, after any subset argument has been used. Default is options()\$na.action.
dist	assumed distribution for y variable. If the argument is a character string, then it is assumed to name an element from survreg.distributions. These include "weibull", "exponential", "gaussian", "logistic","lognormal" and "loglogistic". Otherwise, it is assumed to be a user defined list conforming to the format described in survreg.distributions.
parms	a list of fixed parameters. For the t-distribution for instance this is the degrees of freedom; most of the distributions have no parameters.
init	optional vector of initial values for the parameters.
scale	optional fixed value for the scale. If set to <=0 then the scale is estimated.
control	a list of control values, in the format produced by survreg.control. The default value is survreg.control()

model, x, y	flags to control what is returned. If any of these is true, then the model frame, the model matrix, and/or the vector of response times will be returned as components of the final result, with the same names as the flag arguments.
score	return the score vector. (This is expected to be zero upon successful convergence.)
robust	Use robust sandwich error instead of the asymptotic formula. Defaults to TRUE if there is a cluster argument.
cluster	Optional variable that identifies groups of subjects, used in computing the robust variance. Like model variables, this is searched for in the dataset pointed to by the data argument.
...	other arguments which will be passed to survreg.control.

Details

All the distributions are cast into a location-scale framework, based on chapter 2.2 of Kalbfleisch and Prentice. The resulting parameterization of the distributions is sometimes (e.g. gaussian) identical to the usual form found in statistics textbooks, but other times (e.g. Weibull) it is not. See the book for detailed formulas.

When using weights be aware of the difference between replication weights and sampling weights. In the former, a weight of '2' means that there are two identical observations, which have been combined into a single row of data. With sampling weights there is a single observed value, with a weight used to achieve balance with respect to some population. To get proper variance with replication weights use the default variance, for sampling weights use the robust variance. Replication weights were once common (when computer memory was much smaller) but are now rare.

All of weights, subset and offset are evaluated in the same way as variables in formula, that is first in data and then in the environment of formula. Note that values calculated inside the formula, such as mean(x), are evaluated before subsetting - which may lead to unexpected results if used with subset. For more information see the Details section of model.frame.

Value

an object of class survreg is returned.

References

Kalbfleisch, J. D. and Prentice, R. L., The statistical analysis of failure time data, Wiley, 2002.

See Also

survreg.object, survreg.distributions, pspline, frailty, ridge

Examples

```
# Fit an exponential model: the two fits are the same
survreg(Surv(futime, fustat) ~ ecog.ps + rx, ovarian, dist='weibull',
                                    scale=1)
survreg(Surv(futime, fustat) ~ ecog.ps + rx, ovarian,
        dist="exponential")
```

```
#
# A model with different baseline survival shapes for two groups, i.e.,
#   two different scale parameters
survreg(Surv(time, status) ~ ph.ecog + age + strata(sex), lung)

# There are multiple ways to parameterize a Weibull distribution. The survreg
# function embeds it in a general location-scale family, which is a
# different parameterization than the rweibull function, and often leads
# to confusion.
#   survreg's scale  =    1/(rweibull shape)
#   survreg's intercept = log(rweibull scale)
#   For the log-likelihood all parameterizations lead to the same value.
y <- rweibull(1000, shape=2, scale=5)
survreg(Surv(y)~1, dist="weibull")

# Economists fit a model called `tobit regression', which is a standard
# linear regression with Gaussian errors, and left censored data.
tobinfit <- survreg(Surv(durable, durable>0, type='left') ~ age + quant,
            data=tobin, dist='gaussian')
```

`survreg.control` *Package options for survreg and coxph*

Description

This functions checks and packages the fitting options for `survreg`

Usage

```
survreg.control(maxiter=30, rel.tolerance=1e-09,
toler.chol=1e-10, iter.max, debug=0, outer.max=10)
```

Arguments

`maxiter`	maximum number of iterations
`rel.tolerance`	relative tolerance to declare convergence
`toler.chol`	Tolerance to declare Cholesky decomposition singular
`iter.max`	same as `maxiter`
`debug`	print debugging information
`outer.max`	maximum number of outer iterations for choosing penalty parameters

Value

A list with the same elements as the input

See Also

survreg

`survreg.distributions` *Parametric Survival Distributions*

Description

List of distributions for accelerated failure models. These are location-scale families for some transformation of time. The entry describes the cdf F and density f of a canonical member of the family.

Usage

```
survreg.distributions
```

Format

There are two basic formats, the first defines a distribution de novo, the second defines a new distribution in terms of an old one.

name:	name of distribution
variance:	function(parms) returning the variance (currently unused)
init(x,weights,...):	Function returning an initial estimate of the mean and variance (used for initial values in the iteration)
density(x,parms):	Function returning a matrix with columns F, $1-F$, f, f'/f, and f''/f
quantile(p,parms):	Quantile function
scale:	Optional fixed value for the scale parameter
parms:	Vector of default values and names for any additional parameters
deviance(y,scale,parms):	Function returning the deviance for a saturated model; used only for deviance residuals.

and to define one distribution in terms of another

name:	name of distribution
dist:	name of parent distribution
trans:	transformation (eg log)
dtrans:	derivative of transformation
itrans:	inverse of transformation
scale:	Optional fixed value for scale parameter

Details

There are four basic distributions:`extreme`, `gaussian`, `logistic` and `t`. The last three are parametrised in the same way as the distributions already present in R. The extreme value cdf is

$$F = 1 - e^{-e^t}.$$

When the logarithm of survival time has one of the first three distributions we obtain respectively `weibull`, `lognormal`, and `loglogistic`. The location-scale parameterization of a Weibull distribution found in `survreg` is not the same as the parameterization of `rweibull`.

The other predefined distributions are defined in terms of these. The `exponential` and `rayleigh` distributions are Weibull distributions with fixed `scale` of 1 and 0.5 respectively, and `loggaussian` is a synonym for `lognormal`.

For speed parts of the three most commonly used distributions are hardcoded in C; for this reason the elements of `survreg.distributions` with names of "Extreme value", "Logistic" and "Gaussian" should not be modified. (The order of these in the list is not important, recognition is by name.) As an alternative to modifying `survreg.distributions` a new distribution can be specified as a separate list. This is the preferred method of addition and is illustrated below.

See Also

`survreg`, `pweibull`, `pnorm`,`plogis`, `pt`, `survregDtest`

Examples

```
# time transformation
survreg(Surv(time, status) ~ ph.ecog + sex, dist='weibull', data=lung)
# change the transformation to work in years
# intercept changes by log(365), everything else stays the same
my.weibull <- survreg.distributions$weibull
my.weibull$trans <- function(y) log(y/365)
my.weibull$itrans <- function(y) 365*exp(y)
survreg(Surv(time, status) ~ ph.ecog + sex, lung, dist=my.weibull)

# Weibull parametrisation
y<-rweibull(1000, shape=2, scale=5)
survreg(Surv(y)~1, dist="weibull")
# survreg scale parameter maps to 1/shape, linear predictor to log(scale)

# Cauchy fit
mycauchy <- list(name='Cauchy',
                 init= function(x, weights, ...)
                      c(median(x), mad(x)),
                 density= function(x, parms) {
                      temp <- 1/(1 + x^2)
                      cbind(.5 + atan(x)/pi, .5+ atan(-x)/pi,
                            temp/pi, -2 *x*temp, 2*temp*(4*x^2*temp -1))
                      },
                 quantile= function(p, parms) tan((p-.5)*pi),
                 deviance= function(...) stop('deviance residuals not defined')
                 )
survreg(Surv(log(time), status) ~ ph.ecog + sex, lung, dist=mycauchy)
```

`survreg.object` *Parametric Survival Model Object*

Description

This class of objects is returned by the `survreg` function to represent a fitted parametric survival model. Objects of this class have methods for the functions `print`, `summary`, `predict`, and `residuals`.

COMPONENTS

The following components must be included in a legitimate `survreg` object.

coefficients the coefficients of the `linear.predictors`, which multiply the columns of the model matrix. It does not include the estimate of error (sigma). The names of the coefficients are the names of the single-degree-of-freedom effects (the columns of the model matrix). If the model is over-determined there will be missing values in the coefficients corresponding to non-estimable coefficients.

icoef coefficients of the baseline model, which will contain the intercept and log(scale), or multiple scale factors for a stratified model.

var the variance-covariance matrix for the parameters, including the log(scale) parameter(s).

loglik a vector of length 2, containing the log-likelihood for the baseline and full models.

iter the number of iterations required

linear.predictors the linear predictor for each subject.

df the degrees of freedom for the final model. For a penalized model this will be a vector with one element per term.

scale the scale factor(s), with length equal to the number of strata.

idf degrees of freedom for the initial model.

means a vector of the column means of the coefficient matrix.

dist the distribution used in the fit.

weights included for a weighted fit.

The object will also have the following components found in other model results (some are optional): `linear predictors`, `weights`, `x`, `y`, `model`, `call`, `terms` and `formula`. See `lm`.

See Also

`survreg`, `lm`

survregDtest *Verify a survreg distribution*

Description

This routine is called by survreg to verify that a distribution object is valid.

Usage

```
survregDtest(dlist, verbose = F)
```

Arguments

dlist	the list describing a survival distribution
verbose	return a simple TRUE/FALSE from the test for validity (the default), or a verbose description of any flaws.

Details

If the survreg function rejects your user-supplied distribution as invalid, this routine will tell you why it did so.

Value

TRUE if the distribution object passes the tests, and either FALSE or a vector of character strings if not.

Author(s)

Terry Therneau

See Also

survreg.distributions, survreg

Examples

```
# An invalid distribution (it should have "init =" on line 2)
#  surveg would give an error message
mycauchy <- list(name='Cauchy',
                 init<- function(x, weights, ...)
                      c(median(x), mad(x)),
                 density= function(x, parms) {
                      temp <- 1/(1 + x^2)
                      cbind(.5 + atan(temp)/pi, .5+ atan(-temp)/pi,
                            temp/pi, -2 *x*temp, 2*temp^2*(4*x^2*temp -1))
                      },
                 quantile= function(p, parms) tan((p-.5)*pi),
                 deviance= function(...) stop('deviance residuals not defined')
```

```
                )

survregDtest(mycauchy, TRUE)
```

`survSplit` *Split a survival data set at specified times*

Description

Given a survival data set and a set of specified cut times, split each record into multiple subrecords at each cut time. The new data set will be in 'counting process' format, with a start time, stop time, and event status for each record.

Usage

```
survSplit(formula, data, subset, na.action=na.pass,
            cut, start="tstart", id, zero=0, episode,
                              end="tstop", event="event", added)
```

Arguments

`formula`	a model formula
`data`	a data frame
`subset`, `na.action`	rows of the data to be retained
`cut`	the vector of timepoints to cut at
`start`	character string with the name of a start time variable (will be created if needed)
`id`	character string with the name of new id variable to create (optional). This can be useful if the data set does not already contain an identifier.
`zero`	If `start` doesn't already exist, this is the time that the original records start.
`episode`	character string with the name of new episode variable (optional)
`end`	character string with the name of event time variable
`event`	character string with the name of censoring indicator
`added`	character string with the name of the new "this observation was added" variable (optional)

Details

Each interval in the original data is cut at the given points; if an original row were (15, 60] with a cut vector of (10,30, 40) the resulting data set would have intervals of (15,30], (30,40] and (40, 60].

Each row in the final data set will lie completely within one of the cut intervals. Which interval for each row of the output is shown by the `episode` variable, where 1= less than the first cutpoint, 2= between the first and the second, etc. For the example above the values would be 2, 3, and 4. Whether a given row was added is shown by the `added` variable.

The routine is called with a formula as the first argument. The right hand side of the formula can be used to delimit variables that should be retained; normally one will use ~ . as a shorthand to retain them all. The routine will try to retain variable names, e.g. Surv(adam, joe, fred)~. will result in a data set with those same variable names for tstart, end, and event options rather than the defaults. Any user specified values for these options will be used if they are present, of course. However, the routine is not sophisticated; it only does this substitution for simple names. A call of Surv(time, stat==2) for instance will not retain "stat" as the name of the event variable.

Rows of data with a missing time or status are copied across unchanged, unless the na.action argument is changed from its default value of na.pass. But in the latter case any row that is missing for any variable will be removed, which is rarely what is desired.

Value

New, longer, data frame.

See Also

Surv, cut, reshape

Examples

```
fit1 <- coxph(Surv(time, status) ~ karno + age + trt, veteran)
plot(cox.zph(fit1)[1])
# a cox.zph plot of the data suggests that the effect of Karnofsky score
#  begins to diminish by 60 days and has faded away by 120 days.
# Fit a model with separate coefficients for the three intervals.
#
vet2 <- survSplit(Surv(time, status) ~., veteran,
                  cut=c(60, 120), episode ="timegroup")
fit2 <- coxph(Surv(tstart, time, status) ~ karno* strata(timegroup) +
                age + trt, data= vet2)
c(overall= coef(fit1)[1],
  t0_60  = coef(fit2)[1],
  t60_120= sum(coef(fit2)[c(1,4)]),
  t120   = sum(coef(fit2)[c(1,5)]))

# Sometimes we want to split on one scale and analyse on another
#  Add a "current age" variable to the mgus2 data set.
temp1 <- mgus2
temp1$endage <- mgus2$age + mgus2$futime/12    # futime is in months
temp1$startage <- temp1$age
temp2 <- survSplit(Surv(age, endage, death) ~ ., temp1, cut=25:100,
                   start= "age1", end= "age2")

# restore the time since enrollment scale
temp2$time1 <- (temp2$age1 - temp2$startage)*12
temp2$time2 <- (temp2$age2 - temp2$startage)*12

# In this data set, initial age and current age have similar utility
mfit1 <- coxph(Surv(futime, death) ~ age + sex, data=mgus2)
mfit2 <- coxph(Surv(time1, time2, death) ~ age1 + sex, data=temp2)
```

tcut *Factors for person-year calculations*

Description

Attaches categories for person-year calculations to a variable without losing the underlying continuous representation

Usage

```
tcut(x, breaks, labels, scale=1)
## S3 method for class 'tcut'
levels(x)
```

Arguments

x	numeric/date variable
breaks	breaks between categories, which are right-continuous
labels	labels for categories
scale	Multiply x and breaks by this.

Value

An object of class tcut

See Also

cut, pyears

Examples

```
# For pyears, all time variable need to be on the same scale; but
# futime is in months and age is in years
test <- mgus2
test$years <- test$futime/30.5   # follow-up in years

# first grouping based on years from starting age (= current age)
# second based on years since enrollment (all start at 0)
test$agegrp <- tcut(test$age, c(0,60, 70, 80, 100),
                    c("<=60", "60-70", "70-80", ">80"))
test$fgrp  <- tcut(rep(0, nrow(test)), c(0, 1, 5, 10, 100),
                   c("0-1yr", "1-5yr", "5-10yr", ">10yr"))

# death rates per 1000, by age group
pfit1 <- pyears(Surv(years, death) ~ agegrp, scale =1000, data=test)
round(pfit1$event/ pfit1$pyears)

#death rates per 100, by follow-up year and age
```

```
# there are excess deaths in the first year, within each age stratum
pfit2 <- pyears(Surv(years, death) ~ fgrp + agegrp, scale =1000, data=test)
round(pfit2$event/ pfit2$pyears)
```

`timeline` *Convert to/from a timeline data set format*

Description

Convert from a 'timeline' data set format for survival data to the counting process form, and vice versa.

Usage

```
totimeline(formula, data, id, istate)
fromtimeline(formula, data, id, istate="istate")
```

Arguments

`formula`	a model formula with a `Surv` object on the left
`data`	data set in which to evaluate the formula
`id`	the name of the identifier variable, which will be searched first in the `data`. Multiple rows for the same subject will share the same `id` value.
`istate`	for `totimeline` the name of the variable in the counting process data set that contains the initial state. For `fromtimeline` the variable name to use for the initial state.

Details

Counting process style data sets are heavily used in the `survival` package for both time-dependent covariates and multistate data. Each row of the data will contain a time interval (t1, t2), status or state at the end of the interval, covariate values that apply over the interval, and an id variable. A timeline data set will have a single time covariate, an `id` variable, along with other covariate and outcome values that were observed at that time point. If some covariates are observed at a particular time point but others were not, these other values would be missing for that row. (The exception are covariates that are constant, like birthdate or a genetic marker, which will normally appear across all rows).

A disadvantage of the counting process form is that it requires special tools for manipulation, e.g., `tmerge`; timeline data sets are much simpler in structure and thus can benefit from a much wider variety of tools in their creation. They are also more direct wrt ensuring validity: each row should encode what was *actually observed* at that time point. Another potential advantage is for variables such as diabetes, which might be used as an outcome in one model and a predictor in another. This requires two separate variables in a counting process data set, since covariates change at the beginning of a time interval and outcomes happen at the end of it.

The conversion from timeline to counting process form uses the same rules with respect to missing values as `tmerge`, it is in fact what is used behind the scenes to do the conversion.

Value

a data set of the proper form

Note

This is at present an experimental feature.

See Also

tmerge, survSplit

`tmerge` *Time based merge for survival data*

Description

A common task in survival analysis is the creation of start,stop data sets which have multiple intervals for each subject, along with the covariate values that apply over that interval. This function aids in the creation of such data sets.

Usage

```
tmerge(data1, data2,  id,..., tstart, tstop, options)
```

Arguments

`data1`	the primary data set, to which new variables and/or observation will be added
`data2`	second data set in which all the other arguments will be found
`id`	subject identifier
`...`	operations that add new variables or intervals, see below
`tstart`	optional variable to define the valid time range for each subject, only used on an initial call
`tstop`	optional variable to define the valid time range for each subject, only used on an initial call
`options`	a list of options. Valid ones are idname, tstartname, tstopname, delay, na.rm, and tdcstart. See the explanation below.

Details

The program is often run in multiple passes, the first of which defines the basic structure, and subsequent ones that add new variables to that structure. For a more complete explanation of how this routine works refer to the vignette on time-dependent variables.

There are 4 types of operational arguments: a time dependent covariate (tdc), cumulative count (cumtdc), event (event) or cumulative event (cumevent). Time dependent covariates change their values before an event, events are outcomes.

- newname = tdc(y, x, init): A new time dependent covariate variable will created. The argument y is assumed to be on the scale of the start and end time, and each instance describes the occurrence of a "condition" at that time. The second argument x is optional. In the case where x is missing the count variable starts at 0 for each subject and becomes 1 at the time of the event. If x is present the value of the time dependent covariate is initialized to value of init, if present, or the tdcstart option otherwise, and is updated to the value of x at each observation. If the option na.rm=TRUE missing values of x are first removed, i.e., the update will not create missing values.
- newname = cumtdc(y,x, init): Similar to tdc, except that the event count is accumulated over time for each subject. The variable x must be numeric.
- newname = event(y,x): Mark an event at time y. In the usual case that x is missing the new 0/1 variable will be similar to the 0/1 status variable of a survival time.
- newname = cumevent(y,x): Cumulative events.

The function adds three new variables to the output data set: tstart, tstop, and id. The options argument can be used to change these names. If, in the first call, the id argument is a simple name, that variable name will be used as the default for the idname option. If data1 contains the tstart variable then that is used as the starting point for the created time intervals, otherwise the initial interval for each id will begin at 0 by default. This will lead to an invalid interval and subsequent error if say a death time were <= 0.

The na.rm option affects creation of time-dependent covariates. Should a data row in data2 that has a missing value for the variable be ignored or should it generate an observation with a value of NA? The default of TRUE causes the last non-missing value to be carried forward. The delay option causes a time-dependent covariate's new value to be delayed, see the vignette for an example.

Value

a data frame with two extra attributes tm.retain and tcount. The first contains the names of the key variables, and which names correspond to tdc or event variables. The tcount variable contains counts of the match types. New time values that occur before the first interval for a subject are "early", those after the last interval for a subject are "late", and those that fall into a gap are of type "gap". All these are are considered to be outside the specified time frame for the given subject. An event of this type will be discarded. An observation in data2 whose identifier matches no rows in data1 is of type "missid" and is also discarded. A time-dependent covariate value will be applied to later intervals but will not generate a new time point in the output.

The most common type will usually be "within", corresponding to those new times that fall inside an existing interval and cause it to be split into two. Observations that fall exactly on the edge of an interval but within the (min, max] time for a subject are counted as being on a "leading" edge, "trailing" edge or "boundary". The first corresponds for instance to an occurrence at 17 for someone with an intervals of (0,15] and (17, 35]. A tdc at time 17 will affect this interval but an event at 17 would be ignored. An event occurrence at 15 would count in the (0,15] interval. The last case is where the main data set has touching intervals for a subject, e.g. (17, 28] and (28,35] and a new occurrence lands at the join. Events will go to the earlier interval and counts to the latter one. A last column shows the number of additions where the id and time point were identical. When this occurs, the tdc and event operators will use the final value in the data (last edit wins), but ignoring missing, while cumtdc and cumevent operators add up the values.

These extra attributes are ephemeral and will be discarded if the dataframe is modified. This is intentional, since they will become invalid if for instance a subset were selected.

Author(s)

Terry Therneau

See Also

neardate

Examples

```
# The pbc data set contains baseline data and follow-up status
# for a set of subjects with primary biliary cirrhosis, while the
# pbcseq data set contains repeated laboratory values for those
# subjects.
# The first data set contains data on 312 subjects in a clinical trial plus
# 106 that agreed to be followed off protocol, the second data set has data
# only on the trial subjects.
temp <- subset(pbc, id <= 312, select=c(id:sex, stage)) # baseline data
pbc2 <- tmerge(temp, temp, id=id, endpt = event(time, status))
pbc2 <- tmerge(pbc2, pbcseq, id=id, ascites = tdc(day, ascites),
               bili = tdc(day, bili), albumin = tdc(day, albumin),
               protime = tdc(day, protime), alk.phos = tdc(day, alk.phos))

fit <- coxph(Surv(tstart, tstop, endpt==2) ~ protime + log(bili), data=pbc2)
```

---

tobin *Tobin's Tobit data*

Description

Economists fit a parametric censored data model called the 'tobit'. These data are from Tobin's original paper.

Usage

```
tobin
data(tobin, package="survival")
```

Format

A data frame with 20 observations on the following 3 variables.

durable Durable goods purchase

age Age in years

quant Liquidity ratio (x 1000)

Source

J Tobin (1958), Estimation of relationships for limited dependent variables. *Econometrica* **26**, 24–36.

Examples

```
tfit <- survreg(Surv(durable, durable>0, type='left') ~age + quant,
                data=tobin, dist='gaussian')

predict(tfit,type="response")
```

`transplant` *Liver transplant waiting list*

Description

Subjects on a liver transplant waiting list from 1990-1999, and their disposition: received a transplant, died while waiting, withdrew from the list, or censored.

Usage

```
transplant
data(transplant, package="survival")
```

Format

A data frame with 815 (transplant) observations on the following 6 variables.

`age` age at addition to the waiting list

`sex` `m` or `f`

`abo` blood type: `A`, `B`, `AB` or `O`

`year` year in which they entered the waiting list

`futime` time from entry to final disposition

`event` final disposition: `censored`, `death`, `ltx` or `withdraw`

Details

This represents the transplant experience in a particular region, over a time period in which liver transplant became much more widely recognized as a viable treatment modality. The number of liver transplants rises over the period, but the number of subjects added to the liver transplant waiting list grew much faster. Important questions addressed by the data are the change in waiting time, who waits, and whether there was an consequent increase in deaths while on the list.

Blood type is an important consideration. Donor livers from subjects with blood type O can be used by patients with A, B, AB or 0 blood types, whereas an AB liver can only be used by an AB recipient. Thus type O subjects on the waiting list are at a disadvantage, since the pool of competitors is larger for type O donor livers.

This data is of historical interest and provides a useful example of competing risks, but it has little relevance to current practice. Liver allocation policies have evolved and now depend directly on each individual patient's risk and need, assessments of which are regularly updated while a patient is on the waiting list. The overall organ shortage remains acute, however.

The `transplant` data set was a version used early in the analysis, `transplant2` has several additions and corrections, and was the final data set and matches the paper.

References

Kim WR, Therneau TM, Benson JT, Kremers WK, Rosen CB, Gores GJ, Dickson ER. Deaths on the liver transplant waiting list: An analysis of competing risks. Hepatology 2006 Feb; 43(2):345-51.

Examples

```
#since event is a factor, survfit creates competing risk curves
pfit <- survfit(Surv(futime, event) ~ abo, transplant)
pfit[,2]  #time to liver transplant, by blood type
plot(pfit[,2], mark.time=FALSE, col=1:4, lwd=2, xmax=735,
       xscale=30.5, xlab="Months", ylab="Fraction transplanted",
       xaxt = 'n')
temp <- c(0, 6, 12, 18, 24)
axis(1, temp*30.5, temp)
legend(450, .35, levels(transplant$abo), lty=1, col=1:4, lwd=2)

# competing risks for type O
plot(pfit[4,], xscale=30.5, xmax=735, col=1:3, lwd=2)
legend(450, .4, c("Death", "Transpant", "Withdrawal"), col=1:3, lwd=2)
```

---

udca *Data from a trial of usrodeoxycholic acid*

Description

Data from a trial of ursodeoxycholic acid (UDCA) in patients with primary biliary cirrohosis (PBC).

Usage

```
udca
udca2
data(udca, package="survival")
```

Format

A data frame with 170 observations on the following 15 variables.

`id` subject identifier

`trt` treatment of 0=placebo, 1=UDCA

`entry.dt` date of entry into the study

`last.dt` date of last on-study visit

`stage` stage of disease

`bili` bilirubin value at entry

`riskscore` the Mayo PBC risk score at entry

`death.dt` date of death

`tx.dt` date of liver transplant

`hprogress.dt` date of histologic progression

`varices.dt` appearance of esphogeal varices

`ascites.dt` appearance of ascites

`enceph.dt` appearance of encephalopathy

`double.dt` doubling of initial bilirubin

`worsen.dt` worsening of symptoms by two stages

Details

This data set is used in the Therneau and Grambsh. The `udca1` data set contains the baseline variables along with the time until the first endpoint (any of death, transplant, ..., worsening). The `udca2` data set treats all of the endpoints as parallel events and has a stratum for each.

References

T. M. Therneau and P. M. Grambsch, Modeling survival data: extending the Cox model. Springer, 2000.

K. D. Lindor, E. R. Dickson, W. P Baldus, R.A. Jorgensen, J. Ludwig, P. A. Murtaugh, J. M. Harrison, R. H. Weisner, M. L. Anderson, S. M. Lange, G. LeSage, S. S. Rossi and A. F. Hofman. Ursodeoxycholic acid in the treatment of primary biliary cirrhosis. Gastroenterology, 106:1284-1290, 1994.

Examples

```
# values found in table 8.3 of the book
fit1 <- coxph(Surv(futime, status) ~ trt + log(bili) + stage,
          cluster =id , data=udca1)
fit2 <- coxph(Surv(futime, status) ~ trt + log(bili) + stage +
          strata(endpoint), cluster=id,  data=udca2)
```

`untangle.specials` *Help Process the 'specials' Argument of the 'terms' Function.*

Description

Given a `terms` structure and a desired special name, this returns an index appropriate for subscripting the `terms` structure and another appropriate for the data frame.

Usage

```
untangle.specials(tt, special, order=1)
```

Arguments

tt	a terms object.
special	the name of a special function, presumably used in the terms object.
order	the order of the desired terms. If set to 2, interactions with the special function will be included.

Value

a list with two components:

vars	a vector of variable names, as would be found in the data frame, of the specials.
terms	a numeric vector, suitable for subscripting the terms structure, that indexes the terms in the expanded model formula which involve the special.

Examples

```
formula <- Surv(tt,ss) ~ x + z*strata(id)
tms <- terms(formula, specials="strata")
## the specials attribute
attr(tms, "specials")
## main effects
untangle.specials(tms, "strata")
## and interactions
untangle.specials(tms, "strata", order=1:2)
```

---

uspop2 *Projected US Population*

Description

US population by age and sex, for 2000 through 2020

Format

The data is a matrix with dimensions age, sex, and calendar year. Age goes from 0 through 100, where the value for age 100 is the total for all ages of 100 or greater.

Details

This data is often used as a "standardized" population for epidemiology studies.

Source

NP2008_D1: Projected Population by Single Year of Age, Sex, Race, and Hispanic Origin for the United States: July 1, 2000 to July 1, 2050, www.census.gov/population/projections.

See Also

uspop

Examples

```
us50 <- uspop2[51:101,, "2000"]  #US 2000 population, 50 and over
age <- as.integer(dimnames(us50)[[1]])
smat <- model.matrix( ~ factor(floor(age/5)) -1)
ustot <- t(smat) %*% us50  #totals by 5 year age groups
temp <- c(50,55, 60, 65, 70, 75, 80, 85, 90, 95)
dimnames(ustot) <- list(c(paste(temp, temp+4, sep="-"), "100+"),
                        c("male", "female"))
```

---

vcov.coxph *Variance-covariance matrix*

Description

Extract and return the variance-covariance matrix.

Usage

```
## S3 method for class 'coxph'
vcov(object, complete=TRUE, ...)
## S3 method for class 'coxphms'
vcov(object, complete=TRUE, matrix=FALSE, ...)
## S3 method for class 'survreg'
vcov(object, complete=TRUE, ...)
```

Arguments

object	a fitted model object
complete	logical indicating if the full variance-covariance matrix should be returned. This has an effect only for an over-determined fit where some of the coefficients are undefined, and coef(object) contains corresponding NA values. If complete=TRUE the returned matrix will have row/column for each coefficient, if FALSE it will contain rows/columns corresponding to the non-missing coefficients. The coef() function has a similar complete argument.
matrix	if TRUE the result will be an array with one covariance matrix per transition in the model; vcov(object, matrix=TRUE)[,,i] will be the variance matrix corresponding to coef(object, matrix=TRUE)[,i]
...	additional arguments for method functions

Details

For the coxph and survreg functions the returned matrix is a particular generalized inverse: the row and column corresponding to any NA coefficients will be zero. This is a side effect of the generalized cholesky decomposion used in the unerlying compuatation.

Value

a matrix or array

`veteran` *Veterans' Administration Lung Cancer study*

Description

Randomised trial of two treatment regimens for lung cancer. This is a standard survival analysis data set.

Usage

```
veteran
data(cancer, package="survival")
```

Format

trt:	1=standard 2=test
celltype:	1=squamous, 2=smallcell, 3=adeno, 4=large
time:	survival time
status:	censoring status
karno:	Karnofsky performance score (100=good)
diagtime:	months from diagnosis to randomisation
age:	in years
prior:	prior therapy 0=no, 10=yes

Source

D Kalbfleisch and RL Prentice (1980), *The Statistical Analysis of Failure Time Data*. Wiley, New York.

`xtfrm.Surv` *Sorting order for Surv objects*

Description

Sort survival objects into a partial order, which is the same one used internally for many of the calculations.

Usage

```
## S3 method for class 'Surv'
xtfrm(x)
```

Arguments

x	a Surv object

Details

This creates a partial ordering of survival objects. The result is sorted in time order, for tied pairs of times right censored events come after observed events (censor after death), and left censored events are sorted before observed events. For counting process data (`tstart, tstop, status`) the ordering is by stop time, status, and start time, again with censoring last. Interval censored data is sorted using the midpoint of each interval.

The `xtfrm` routine is used internally by `order` and `sort`, so these results carry over to those routines.

Value

a vector of integers which will have the same sort order as `x`.

Author(s)

Terry Therneau

See Also

`sort`, `order`

Examples

```
test <- c(Surv(c(10, 9,9, 8,8,8,7,5,5,4), rep(1:0, 5)), Surv(6.2, NA))
test
sort(test)
```

yates *Population prediction*

Description

Compute population marginal means (PMM) from a model fit, for a chosen population and statistic.

Usage

```
yates(fit, term, population = c("data", "factorial", "sas"),
levels, test = c("global", "trend", "pairwise"), predict = "linear",
options, nsim = 200, method = c("direct", "sgtt"))
```

Arguments

fit	a model fit. Examples using lm, glm, and coxph objects are given in the vignette.
term	the term from the model whic is to be evaluated. This can be written as a character string or as a formula.
population	the population to be used for the adjusting variables. User can supply their own data frame or select one of the built in choices. The argument also allows "empirical" and "yates" as aliases for data and factorial, respectively, and ignores case.
levels	optional, what values for term should be used.
test	the test for comparing the population predictions.
predict	what to predict. For a glm model this might be the 'link' or 'response'. For a coxph model it can be linear, risk, or survival. User written functions are allowed.
options	optional arguments for the prediction method.
nsim	number of simulations used to compute a variance for the predictions. This is not needed for the linear predictor.
method	the computational approach for testing equality of the population predictions. Either the direct approach or the algorithm used by the SAS glim procedure for "type 3" tests.

Details

The many options and details of this function are best described in a vignette on population prediction.

Value

an object of class yates with components of

estimate	a data frame with one row for each level of the term, and columns containing the level, the mean population predicted value (mppv) and its standard deviation.

tests	a matrix giving the test statistics
mvar	the full variance-covariance matrix of the mppv values
summary	optional: any further summary if the values provided by the prediction method.

Author(s)

Terry Therneau

Examples

```
fit1 <- lm(skips ~ Solder*Opening + Mask, data = solder)
yates(fit1, ~Opening, population = "factorial")

fit2 <- coxph(Surv(time, status) ~ factor(ph.ecog)*sex + age, lung)
yates(fit2, ~ ph.ecog, predict="risk")  # hazard ratio
```

---

yates_setup *Method for adding new models to the* yates *function.*

Description

This is a method which is called by the yates function, in order to setup the code to handle a particular model type. Methods for glm, coxph, and default are part of the survival package.

Usage

```
yates_setup(fit, ...)
```

Arguments

fit	a fitted model object
...	optional arguments for some methods

Details

If the predicted value should be the linear predictor, the function should return NULL. The yates routine has particularly efficient code for this case. Otherwise it should return a prediction function or a list of two elements containing the prediction function and a summary function. The prediction function will be passed the linear predictor as a single argument and should return a vector of predicted values.

Note

See the vignette on population prediction for more details.

Author(s)

Terry Therneau

See Also

yates

Index