Package ‘survival’

May 11, 2016

Title Survival Analysis

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

Priority recommended

Version 2.39-4

Depends R (>= 2.13.0)

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

LazyData Yes

LazyLoad Yes

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)

NeedsCompilation yes

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

Repository CRAN

Date/Publication 2016-05-11 11:45:26

R topics documented:

aareg 3
agreg.fit 6
aml 8
anova.coxph 8
attrassign 9
basehaz 11
bladder 12
cch 13
cgd 16

cgd0 . . . 17
clogit . . . 18
cluster . . . 20
colon . . . 21
cox.zph . . . 22
coxph . . . 23
coxph.control . . . 27
coxph.detail . . . 28
coxph.object . . . 29
coxph.wtest . . . 30
dsurvreg . . . 31
finegray . . . 33
flchain . . . 34
frailty . . . 36
genfan . . . 38
heart . . . 39
is.ratetable . . . 40
kidney . . . 41
lines.survfit . . . 42
logan . . . 44
logLik.coxph . . . 44
lung . . . 45
mgus . . . 47
mgus2 . . . 49
model.frame.coxph . . . 50
model.matrix.coxph . . . 51
neardate . . . 52
nwtco . . . 53
ovarian . . . 54
pbc . . . 55
pbcseq . . . 56
plot.aareg . . . 58
plot.cox.zph . . . 59
plot.survfit . . . 60
predict.coxph . . . 62
predict.survreg . . . 64
print.aareg . . . 66
print.summary.coxph . . . 67
print.summary.survexp . . . 67
print.summary.survfit . . . 68
print.survfit . . . 69
pspline . . . 70
pyears . . . 72
quantile.survfit . . . 75
ratetable . . . 76
ratetableDate . . . 77
ratetables . . . 78
rats . . . 79

rats2 . . . 79
residuals.coxph . . . 80
residuals.survreg . . . 82
retinopathy . . . 83
ridge . . . 84
stanford2 . . . 85
strata . . . 86
summary.aareg . . . 87
summary.coxph . . . 88
summary.survexp . . . 89
summary.survfit . . . 91
Surv . . . 92
survConcordance . . . 94
survdiff . . . 96
survexp . . . 98
survexp.fit . . . 101
survexp.object . . . 102
survfit . . . 103
survfit.coxph . . . 104
survfit.formula . . . 107
survfit.matrix . . . 111
survfit.object . . . 113
survfitcoxph.fit . . . 114
survobrien . . . 116
survreg . . . 117
survreg.control . . . 119
survreg.distributions . . . 120
survreg.object . . . 122
survregDtest . . . 123
survSplit . . . 124
tcut . . . 125
tmerge . . . 126
tobin . . . 128
transplant . . . 129
untangle.specials . . . 130
uspop2 . . . 131
veteran . . . 132

Index **133**

`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'),
    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 missi ng, 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 i ncluded, 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 calcualtions 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.
`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 inheritly 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)
```

---

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 advantagous 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)
coxph.fit(x, y, strata, offset, init, control, weights, method, rownames)
```

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 hanling 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.

Details

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

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
```

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.

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 log-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 partial-likelihood values does not make sense, 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` or 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

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

Arguments

`object`	model matrix or linear model object
`tt`	terms object
`...`	ignored

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, centered=TRUE)
```

Arguments

fit	a coxph fit
centered	if TRUE return data from a predicted survival curve at the mean values of the covariates fit$mean, if FALSE return a prediction for all covariates equal to zero.

Details

This function is simply an alias for survfit, which is the actual function that does all the computations. See the manual page for that function for the preferred use. This function survives only for backwards support of prior usage.

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 the survfit result. If there are factor variables in the model, then the default predictions at the "mean" are meaningless since they do not correspond to any possible subject; correct results require use of the newdata argument of survfit. Results for all covariates =0 are normally only of use as a building block for further calcuations.

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 formated 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). Over this extended time these subjects are by definition not at risk for another event in the WLW data set.

Usage

```
bladder1
bladder
bladder2
```

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**.

`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 = sys.parent(), 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 indicatorsfor 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 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

```
cgd
```

Format

id subject identifiction 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 identifiction 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

---

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 logliklihood for a conditional logistic regresson 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 ameleorates this; it was incorporated into version 2.36-11 of the survival package. The compuation remains infeasable 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 'appoximate' option maps to the Breslow approximation for the Cox model, for historical reasons.

Case weights are not allowed when the exact option 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.

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. Using `cluster()` in a formula implies that robust sandwich variance estimators are desired.

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 + cluster(litter), rats,
                        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
```

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).

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.

`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", global=TRUE)
```

Arguments

`fit`	the result of fitting a Cox regression model, using the `coxph` function.
`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.
`global`	should a global chi-square test be done, in addition to the per-variable tests.

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 the correlation coefficient between transformed survival time and the scaled Schoenfeld residuals, a chi-square, and the two-sided p-value. For the global test there is no appropriate correlation, so an NA is entered into the matrix as a placeholder.
`x`	the transformed time axis.
`y`	the matrix of scaled Schoenfeld residuals. There will be one column per variable and one row per event. The row labels contain the original event times (for the identity transform, these will be the same as `x`).

`call` the calling sequence for the routine.

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 is true, `beta(t)` will be a horizontal line. The printout gives a test for `slope=0`.

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=FALSE,
      model=FALSE, x=FALSE, y=TRUE, tt, method, ...)
```

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`, or in the `subset` and the `weights` argument.

weights	vector of case weights. 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. Nearly all Cox regression programs use the Breslow method by default, but not this one. The Efron approximation is used as the default here, it is more accurate when dealing with tied death times, and is as efficient computationally. The "exact partial likelihood" is equivalent to a conditional logistic model, and is appropriate when the times are a small set of discrete values. See further below.
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	this argument has been deprecated, use a cluster term in the model instead. (The two options accomplish the same goal – creation of a robust variance – but the second is more flexible).
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.
...	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. However, arguments to offset are not scaled since there are situations where a large offset value is a purposefully used. Users should not use normally allow large numeric offset values.

Value

an object of class coxph representing the fit. See coxph.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.

Special terms

There are three special terms that may be used in the model equation. A strata term identifies a stratified Cox model; separate baseline hazard functions are fit for each strata. The cluster term is used to compute a robust variance for the model. The term + cluster(id) where each value of id is unique is equivalent to specifying the robust=T argument. If the id variable is not unique, it is assumed that it identifies clusters of correlated observations. 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.

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.

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. (Nearly always the first occurs.) The routine attempts to detect when this has happened, not always successfully. The primary consequence for he 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 logisitic 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 15 of 180 subjects at risk had an event on day 7; 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 compuatation can be insufferably long. With (start, stop) data it is much worse since the recursion needs to start anew for each unique start time.

Penalized regression

`coxph` can now 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

`cluster`, `strata`, `Surv`, `survfit`, `pspline`, `frailty`, `ridge`.

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 controling coxph fits*

Description

This is used to set various numeric parameters controling 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)
```

Arguments

`eps`	Iteration continues until the relative change in the log partial likelihood is less than eps. Must be positive.
`toler.chol`	Tolerance for detection of singularity during a Cholesky decomposion 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.

Value

a list containing the values of each of the above constants

Author(s)

Terry Therneau

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)
```

Arguments

object	a Cox model object, i.e., the result of coxph.
riskmat	include the at-risk indicator matrix in the output?

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$mean 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.
riskmat	a matrix with one row for each time and one column for each observation containing a 0/1 value to indicate whether that observation was (1) or was not (0) at risk at the given time point.

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.

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 column means of the X matrix. Subsequent survival curves are adjusted to this value.
n	the number of observations used in the fit.
nevent	the number of events (usually deaths) used in the fit.
concordance	the concordance, as computed by survConcordance.
first	the first derivative vector at the solution.
weights	the vector of case weights, if one was used.
method	the computation method used.
na.action	the na.action attribute, if any, that was returned by the na.action routine.
	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 choelsky 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

---

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 linear predictors 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); since almost none of these distributions are symmetric it will not actually be a mean, but corresponds instead to the linear predictor of a fitted model. Translation to the usual parameterization found in a textbook is not always obvious. For example, the Weibull distribution is fit using the Extreme value distribution along with a log transformation. Letting $F(t) = 1 - \exp[-(at)^p]$ be the cumulative distribution of the Weibull using a standard parameterization in terms of a and p, the survreg location corresponds to $-\log(a)$ and the scale to $1/p$ (Kalbfleish and Prentice, section 2.2.2).

Value

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

References

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

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)
```

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, subset, na.action= na.pass, etype,
    prefix="fg", count, id)
```

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.
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.

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. In the output data set subjects who did not experience the event of interest become censored subjects whose times are artificially extended over multiple intervals, with a decreasing case weight from interval to interval. The output data set will normally contain many more rows than the input.

Time dependent covariates are allowed, but not (currently) delayed entry. If there are time dependent covariates, 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.

See the competing risks vignette for more details.

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

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)

# 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

```
data(flchain)
```

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 or survreg 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 feasable 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 Akiake'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 critera 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 statment 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 <- survreg(Surv(time, status) ~ rx +
                  frailty.gaussian(litter, df=13, sparse=FALSE), rats,
                  subset= (sex=='f'))
rfit2b <- survreg(Surv(time, status) ~ rx +
                  frailty.gaussian(litter, df=13, sparse=TRUE), rats,
                  subset= (sex=='f'))
```

---

genfan *Generator fans*

Description

The data come from a field engineering study of the time to failure of diesel generator fans. The ultimate goal was to decide whether or not to replace the working fans with a higher quality fan to prevent future failures. Seventy generators were studied. For each one, the number of hours of running time from its first being put into service until fan failure or until the end of the study(whichever came first) was recorded.

Usage

```
data("genfan")
```

Format

A data frame with 70 observations on the following 2 variables.

`hours` hours of service

`status` 1=failure, 0=censored

References

Nelson, Journal of Quality Technology, 1:27-52, 1969

`heart` *Stanford Heart Transplant data*

Description

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

Usage

```
heart
jasa
jasa1
```

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

`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(cancer)      # 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.

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)
```

`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", mark=3, col=1, lty=1,
        lwd=1, cex=1, mark.time=FALSE,
xscale=1,  firstx=0, firsty=1, xmax, fun, conf.int=FALSE,
        conf.times, conf.cap=.005, conf.offset=.012,  ...)
## S3 method for class 'survexp'
lines(x, type="l", ...)
## S3 method for class 'survfit'
points(x, xscale, xmax, fun, ...)
```

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`.
`mark, col, lty, lwd, cex`	vectors giving the mark symbol, color, line type, line width and character size for the added curves.
`...`	other graphical parameters
`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.
`xscale`	this parameter is no longer necessary and is ignored. See the note in `plot.survfit`.
`firstx, firsty`	the starting point for the survival curves. If either of these is set to `NA` or < blank > the plot will start at the first time point of the curve.
`xmax`	the maximum horizontal plot coordinate. This shortens the curve before plotting it, so unlike using the `xlim` graphical parameter, warning messages about out of bounds points are not generated.
`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.

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.

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.

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, xscale=365.24)    #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(~ ratetable(sex=sex,age=age*365.35,year=as.Date('1979/1/1')) +
            sex, data=pbc,  times=(0:24)*182)
temp <- lines(efit, lty=2, xscale=365.24, 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

```
data(logan)
```

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. http://www3.norc.org/GSS+Website.

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
cancer
```

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 (0=good 5=dead)
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

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.

`mgus` *Monoclonal gammapothy data*

Description

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

Usage

```
mgus
mgus1
```

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 protien 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 distribuion will become dominated by a single isotype, the product of the malignant clone, visible as a spike on a serum protein electrophoresis. Monoclonal gammapothy 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, subset=(enum==1))
print(sfit)  # the order of printout is the order in which they plot
```

```
plot(sfit, xscale=365.25, lty=c(2,1,2,1), col=c(1,1,2,2),
     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 gammapothy data*

Description

Natural history of 1341 sequential patients with monoclonal gammapothy of undetermined significance (MGUS).

Usage

```
data("mgus2")
```

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`

`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` occurence of PCM: 0=no, 1=yes

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

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

Details

This is a larger follow-on study of the condition also found in data set `mgus`.

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 formual 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)
```

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)))
```

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

Description

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

Usage

```
nwtco
```

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
```

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 Cirrhosis Data*

Description

D This data is from the Mayo Clinic trial in primary biliary cirrhosis (PBC) of the liver 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
```

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.

`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 paramters. 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 orignal analysis (see Fleming and Harrington, figure 4.6.7).

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 large 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

```
pbc
```

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.

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, ...)
```

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.
`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.
`...`	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=5)      # Look at Karnofsy score, old way of doing plot
plot(temp[5])     # New way with subscripting
abline(0, 0, lty=3)
# Add the linear fit as well
abline(lm(temp$y[,5] ~ temp$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,
 mark=3, col=1, lty=1, lwd=1, cex=1, log=FALSE, xscale=1, yscale=1,
 firstx=0, firsty=1, xmax, ymin=0, fun,
 xlab="", ylab="", xaxs="S",  conf.times, conf.cap=.005,
 conf.offset=.012, ...)
```

Arguments

`x`	an object of class `survfit`, usually returned by the `survfit` function.
`conf.int`	determines whether confidence intervals will be plotted. The default is to do so if there is only 1 curve, i.e., no strata.
`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 which is not also a death time. If `mark.time` is a numeric vector, then curves are marked at the specified time points.
`mark`	vector of mark parameters, which will be used to label the curves. The `lines` help file contains examples of the possible marks. The vector is reused cyclically if it is shorter than the number of curves. If it is present this imples `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.

`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.

`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.

`firstx, firsty` the starting point for the survival curves. If either of these is set to `NA` the plot will start at the first time point of the curve. By default, the plot program obeys tradition by having the plot start at (0,0).

If `start.time` argument is used in `survfit`, `firstx` is set to that value.

`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 that unlike using the `xlim` graphical parameter, warning messages about out of bounds points are not generated.

`ymin` lower boundary for y values. Survival curves are most often drawn in the range of 0-1, even if none of the curves approach zero. The parameter is ignored if the `fun` argument is present, or if it has been set to `NA`.

`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), and `fun=sqrt` would generate a curve on square root scale. 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).

`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`. Survival curves are usually 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.

`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.

`...` for future methods

Details

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

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.

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.

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"),
se.fit=FALSE, na.action=na.pass, terms=names(object$assign), collapse,
reference=c("strata", "sample"),  ...)
```

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"`).
`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. The primary underlying reason is statistical: 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 results. Using the `reference="strata"` option causes this to be true for predictions as well.

When the results of `predict` are used in further calculations it may be desirable to use a fixed reference level. Use of `reference="sample"` will use the overall means, and agrees with the `linear.predictors` component of the coxph object (which uses the overall mean for backwards compatability with older code). Predictions of `type="terms"` are almost invariably passed forward to further calculation, so for these we default to using the sample as the reference.

Predictions of type "expected" incorporate the baseline hazard and are thus absolute instead of relative; the `reference` option has no effect on these.

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.

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.

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)
```

`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"), ...)
```

Arguments

x	the result of a call to summary.coxph
digits	significant digits to print
signif.stars	Show stars to highlight small p-values
...	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. 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 smallest point of intersection, e.g., if the survival curve were exactly equal to 0.5 over an interval.

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`

---

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

```
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 dimmanes 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.

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 object.

Usage

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

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?
`tolerance`	tolerance for checking that the suvival 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 seqment of the survival curve exactly matches one of the requested quantiles the returned value will be the midpoint of the horizontal seqment; 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 *Ratetable reference in formula*

Description

This function matches variable names in data to those in a ratetable for survexp

Usage

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

Arguments

...	tags matching dimensions of the ratetable and variables in the data frame (see example)

Value

A data frame

See Also

survexp,survexp.us,is.ratetable

Examples

```
fit <- survfit(Surv(time, status) ~ sex, pbc,subset=1:312)

# The data set does not have entry date, use the midpoint of the study
efit <- survexp(~ ratetable(sex=sex,age=age*365.35,year=as.Date('1979/1/1')) +
            sex, data=pbc,  times=(0:24)*182)
## Not run:
plot(fit, mark.time=F, xscale=365.25, xlab="Years post diagnosis",
        ylab="Survival")
lines(efit, col=2, xscale=365.25) # Add the expected survival line

## End(Not run)
```

---

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.

Details

us total United States population, by age and sex, 1960 to 1980.

uswhite United States white population, by age and sex, 1950 to 1980. This is no longer included, but can be extracted from `survexp.usr` as shown in the examples.

usr United States population, by age, sex and race, 1960 to 1980. Race is white, nonwhite, or black. For 1960 and 1970 the black population values were not reported separately, so the nonwhite values were used.

mn total Minnesota population, by age and sex, 1970 and 1980.

mnwhite Minnesota white population, by age and sex, 1960 to 1980.

fl total Florida population, by age and sex, 1970 and 1980.

flr Florida population, by age, sex and race, 1970-1980. Race is white, nonwhite, or black. For 1970 the black population values were not reported separately, so the nonwhite values were used.

az total Arizona population, by age and sex, 1970 and 1980.

azr Arizona population, by age, sex and race, 1970-1980. Race is white versus nonwhite. For 1970 the nonwhite population values were not reported separately. In order to make the rate table be a matrix, the 1980 values were repeated. (White and non-white values are quite different).

Each of these tables contains the daily hazard rate for a matched subject from the population, defined as $-\log(1-q)/365.24$ where q is the 1 year probability of death as reported in the original tables. 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 entirely for computational convenience. (The fraction .24 in the denominator is based on 24 leap years per century.)

Each table is stored as an array, with additional attributes, and can be subset and manipulated as standard S arrays. Interpolation between calendar years is done "on the fly" by the `survexp` routine.

Some of the deficiencies, e.g., 1970 Arizona non-white, are a result of local (Mayo Clinic) conditions. The data probably exists, but we don't have a copy it in the library.

The tables have been augmented to contain extrapolated values for 1990 and 2000. The details can be found in Mayo Clinic Biostatistics technical report 63 at `http://www.mayo.edu/hsr/techrpt.html`.

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
```

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
```

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.

---

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=FALSE, ...)
## 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", and "scaledsch". Only enough of the string to determine a unique match is required.
collapse	vector indicating which rows to collapse (sum) over. In time-dependent models more than one row data can pertain to a single individual. If there were 4 individuals represented by 3, 1, 2 and 4 rows of data respectively, then collapse=c(1,1,1, 2, 3,3, 4,4,4,4) could be used to obtain per subject rather than per observation residuals.
weighted	if TRUE and the model was fit with case weights, then the weighted residuals are returned.
...	other unused arguments

Value

For martingale and deviance residuals, the returned object is a vector with one element for each subject (without collapse). For score residuals it is a matrix with one row per subject and one column per variable. The row order will match the input data for the original fit. For Schoenfeld residuals, the returned object is a matrix with one row for each event and one column per variable. The rows are ordered by time within strata, and an attribute strata is attached that contains the number of observations in each strata. The scaled Schoenfeld residuals are used in the cox.zph function.

The score residuals are each individual's contribution to the score vector. 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.

NOTE

For deviance residuals, the status variable may need to be reconstructed. For score and Schoenfeld residuals, the X matrix will need to be reconstructed.

References

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

See Also

coxph

Examples

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

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". See the LaTeX documentation (survival/doc/survival.ps.gz) for more detail.
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 an 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.

References

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

See Also

predict.survreg

Examples

```
fit <- survreg(Surv(time,status) ~x, aml)
rr  <- residuals(fit, type='matrix')
```

---

retinopathy *Diabetic Retinopathy*

Description

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

Usage

```
data("retinopathy")
```

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)
```

`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.

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

`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`

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

The result is identical to the `interaction` function, but for the labeling of the factors (`strata` is 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="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 confidence limits are for the risk change associated with one scale unit.
`...`	for future methods

Value

An object of class `summary.coxph`.

See Also

coxph, print.coxph

Examples

```
fit <- coxph(Surv(time, status) ~ age + sex, lung)
summary(fit)
## Not run:
Call:
coxph(formula = Surv(time, status) ~ age + sex, data = lung)

  n= 228

      coef exp(coef) se(coef)     z      p
age  0.017     1.017  0.00922  1.85 0.0650
sex -0.513     0.599  0.16745 -3.06 0.0022

    exp(coef) exp(-coef) lower .95 upper .95
age     1.017      0.983     0.999     1.036
sex     0.599      1.670     0.431     0.831

Rsquare= 0.06   (max possible= 0.999 )
Likelihood ratio test= 14.1  on 2 df,   p=0.000857
Wald test            = 13.5  on 2 df,   p=0.00119
Score (logrank) test = 13.7  on 2 df,   p=0.00105

## End(Not run)
```

`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 retreive 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'), ...)
```

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 valid if the times argument is present.
rmean	Show restricted mean: see print.survfit for details
...	for future methods

Value

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.

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', 'mstate'),
    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) or 1/2 (2=death). For interval censored data, the status indicator is 0=right censored, 1=event at `time`, 2=left censored, 3=interval censored. 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 an event occurred at the end of the interval.
`type`	character string specifying the type of censoring. Possible values are `"right"`, `"left"`, `"counting"`, `"interval"`, `"interval2"` or `"mstate"`.
`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.
`x`	any R object.

Details

When the `type` argument is missing the code assumes a type based on the following rules:

- If there are two unnamed arguments, they will match `time` and `event` in that order. If there are three unnamed arguments they match `time`, `time2` and `event`.
- If the event variable is a factor then type `mstate` is assumed. Otherwise type `right` if there is no `time2` argumement, and type `counting` if there is.

As a consequence the `type` argument can usually be omitted.

When the survival type is "mstate" then the status variable will be treated as a factor. The first level of the factor is taken to represent censoring and remaining ones a transition to the given state.

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, t) for left censored, (t, 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 a `3` is the code for an event.

For multi-state survival (type= "mstate") the status variable can have multiple levels. The first of these will stand for censoring, and the others for various event types, e.g., causes of death.

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`.

See Also

`coxph`, `survfit`, `survreg`.

Examples

```
with(lung, Surv(time, status))
Surv(heart$start, heart$stop, heart$event)
```

`survConcordance` *Compute a concordance measure.*

Description

This function computes the concordance between a right-censored survival time and a single continuous covariate

Usage

```
survConcordance(formula, data, weights, subset, na.action)
survConcordance.fit(y, x, strata, weight)
```

Arguments

`formula`	a formula with a survival time on the left and a single covariate on the right, along with an optional `strata()` term. The left hand term can also be a numeric vector.
`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

Details

The `survConcordance.fit` function computes the result but does no data checking whatsoever. It is intended as a hook for other packages that wish to compute concordance, and the data has already been assembled and verified.

Concordance is defined as Pr(agreement) for any two randomly chosen observations, where in this case agreement means that the observation with the shorter survival time of the two also has the larger risk score. The predictor (or risk score) will often be the result of a Cox model or other regression.

For continuous covariates concordance is equivalent to Kendall's tau, and for logistic regression is is equivalent to the area under the ROC curve. A value of 1 signifies perfect agreement, .6-.7 is a common result for survival data, .5 is an agreement that is no better than chance, and .3-.4 is the performace of some stock market analysts.

The computation involves all n(n-1)/2 pairs of data points in the sample. For survival data, however, some of the pairs are incomparable. For instance a pair of times (5+, 8), the first being a censored value. We do not know whether the first survival time is greater than or less than the second. Among observations that are comparable, pairs may also be tied on survival time (but only if both are uncensored) or on the predictor. The final concondance is (agree + tied/2)/(agree + disagree + tied).

There is, unfortunately, one aspect of the formula above that is unclear. Should the count of ties include observations that are tied on survival time y, tied on the predictor x, or both? By default the concordance only counts ties in x, treating tied survival times as incomparable; this agrees with the AUC calculation used in logistic regression. The Goodman-Kruskal Gamma statistic is (agree-disagree)/(agree + disagree), ignoring ties. It ranges from -1 to +1 similar to a correlation coefficient. Kendall's tau uses ties of both types. All of the components are returned in the result, however, so people can compute other combinations if interested. (If two observations have the same survival and the same x, they are counted in the tied survival time category).

The algorithm is based on a balanced binary tree, which allows the computation to be done in O(n log n) time.

Value

an object containing the concordance, followed by the number of pairs that agree, disagree, are tied, and are not comparable.

See Also

summary.coxph

Examples

```
survConcordance(Surv(time, status) ~age, data=lung)
```

```
options(na.action=na.exclude)
fit <- coxph(Surv(time, status) ~ ph.ecog + age + sex, lung)
survConcordance(Surv(time, status) ~predict(fit), lung)
## Not run:
  n=227 (1 observations deleted due to missing values)
Concordance= 0.6371102 , Gamma= 0.2759638
concordant discordant  tied risk  tied time
     12544       7117        126         28

## End(Not run)
```

---

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)
```

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.

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.

METHOD

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.

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.

References

Harrington, D. P. and Fleming, T. R. (1982). A class of rank test procedures for censored survival data. *Biometrika* **69**, 553-566.

Examples

```
## Two-sample test
survdiff(Surv(futime, fustat) ~ rx,data=ovarian)

## Stratified 7-sample test

survdiff(Surv(time, status) ~ pat.karno + strata(inst), data=lung)

## Expected survival for heart transplant patients based on
## US mortality tables
expect <- survexp(futime ~ ratetable(age=(accept.dt - birth.dt),
     sex=1,year=accept.dt,race="white"), jasa, cohort=FALSE,
     ratetable=survexp.usr)
## actual survival is much worse (no surprise)
survdiff(Surv(jasa$futime, jasa$fustat) ~ offset(expect))
```

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, 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 particuarl, 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, or a matrix curves. 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 corrspond 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.

Curves can be subscripted using either a single or double subscript. If the set of curves is a matrix, as in the above, and one of the dimensions is 1 then the code allows a single subscript to be used. (That is, it is not quite as general as using a single subscript for a numeric matrix.)

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, 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,
        type,vartype,
        conf.type=c("log","log-log","plain","none"), censor=TRUE, id,
        na.action=na.pass, ...)
```

Arguments

formula	A coxph object.
newdata	a data frame with the same variable names as those that appear in the coxph formula. It is also valid to use a vector, if the data frame would consist of a single row. The curve(s) produced will be representative of a cohort whose covariates correspond to the values in newdata. Default is the mean of the covariates used in the coxph fit.
individual	This argument has been superseded by the id argument and is present only for backwards compatability. A logical value indicating whether each row of newdata represents a distinct individual (FALSE, the default), or if each row of the data frame represents different time epochs for only one individual (TRUE). In the former case the result will have one curve for each row in newdata, in the latter only a single curve will be produced.
conf.int	the level for a two-sided confidence interval on the survival curve(s). Default is 0.95.
se.fit	a logical value indicating whether standard errors should be computed. Default is TRUE.

`type,vartype` a character string specifying the type of survival curve. Possible values are `"aalen"`, `"efron"`, or `"kalbfleisch-prentice"` (only the first two characters are necessary). The default is to match the computation used in the Cox model. The Nelson-Aalen-Breslow estimate for `ties='breslow'`, the Efron estimate for `ties='efron'` and the Kalbfleisch-Prentice estimate for a discrete time model `ties='exact'`. Variance estimates are the Aalen-Link-Tsiatis, Efron, and Greenwood. The default will be the Efron estimate for `ties='efron'` and the Aalen estimate otherwise.

`conf.type` One of `"none"`, `"plain"`, `"log"` (the default), or `"log-log"`. 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 last option bases intervals on the log hazard or log(-log(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, then each group of rows 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 missing, then each individual row of `newdata` is presumed to represent a distinct subject and there will be `nrow(newdata)` times the number of strata curves in the result (one for each strata/subject combination). result.

`na.action` the na.action to be used on the newdata argument

`...` for future methods

Details

Serious thought has been given to removing the default value for `newdata`, which is to use a single "psuedo" subject with covariate values equal to the means of the data set, since the resulting curve(s) almost never make sense. It remains due to an unwarranted attachment to the option shown by some users and by other packages. Two particularly egregious examples are factor variables and interactions. Suppose one were studying interspecies transmission of a virus, and the data set has a factor variable with levels ("pig", "chicken") and about equal numbers of observations for each. The “mean” covariate level will be 1/2 – is this a flying pig? As to interactions assume data with sex coded as 0/1, ages ranging from 50 to 80, and a model with age*sex. The “mean” value for the age:sex interaction term will be about 30, a value that does not occur in the data. Users are strongly advised to use the newdata argument.

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 variables 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.)

When all the coefficients are zero, the Kalbfleisch-Prentice estimator reduces to the Kaplan-Meier, the Aalen estimate to the exponential of Nelson's cumulative hazard estimate, and the Efron estimate to the Fleming-Harrington estimate of survival. The variances of the curves from a Cox model are larger than these non-parametrec estimates, however, due to the variance of the coefficients.

See `survfit` for more details about the counts (number of events, number at risk, etc.)

The censor argument was fixed at FALSE in earlier versions of the code and not made available to the user. The default argument is sensible in most instances — and causes the familiar + sign to appear on plots — it is not sensible for time dependent covariates since it may lead to a large number of spurious marks.

Value

an object of class `"survfit"`. See `survfit.object` for details. Methods defined for survfit objects are `print`, `plot`, `lines`, and `points`.

References

Fleming, T. H. and Harrington, D. P. (1984). Nonparametric estimation of the survival distribution in censored data. *Comm. in Statistics* **13**, 2469-86.

Kablfleisch, 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`.

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)

#
# Time to progression/death for patients with monoclonal gammopathy
#  Competing risk curves (cumulative incidence)
fit1 <- survfit(Surv(stop, event=='progression') ~1, data=mgus1,
                    subset=(start==0))
fit2 <- survfit(Surv(stop, status) ~1, data=mgus1,
                    subset=(start==0), etype=event) #competing risks
# CI curves are always plotted from 0 upwards, rather than 1 down
plot(fit2, fun='event', xscale=365.25, xmax=7300, mark.time=FALSE,
            col=2:3, xlab="Years post diagnosis of MGUS")
lines(fit1, fun='event', xscale=365.25, xmax=7300, mark.time=FALSE,
            conf.int=FALSE)
text(10, .4, "Competing Risk: death", col=3)
text(16, .15,"Competing Risk: progression", col=2)
text(15, .30,"KM:prog")
```

`survfit.formula` *Compute a Survival Curve for Censored Data*

Description

Computes an estimate of a survival curve for censored data using either the Kaplan-Meier or the Fleming-Harrington method. For competing risks data it computes the cumulative incidence curve.

Usage

```
## S3 method for class 'formula'
```

```
survfit(formula, data, weights, subset, na.action,
        etype, id, istate, ...)
```

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`.

`etype` a variable giving the type of event. This has been superseded by multi-state Surv objects; see example below.

`id` identifies individual subjects, when a given person can have multiple lines of data.

`istate` for multi-state models, identifies the initial state of each subject

`...` The following additional arguments are passed to internal functions called by `survfit`.

type a character string specifying the type of survival curve. Possible values are `"kaplan-meier"`, `"fleming-harrington"` or `"fh2"` if a formula is given. This is ignored for competing risks or when the Turnbull estimator is used.

error a character string specifying the error. Possible values are `"greenwood"` for the Greenwood formula or `"tsiatis"` or `"aalen"` for the Tsiatis/Aalen formula, or `"robust"` for a robust variance. The last of these is assumed if non-integer case weights are provided.

conf.type One of `"none"`, `"plain"`, `"log"` (the default), or `"log-log"`. 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 last option bases intervals on the log hazard or log(-log(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"`. T he 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.

se.fit a logical value indicating whether standard errors should be computed. Default is `TRUE`.

Details

The estimates used are the Kalbfleisch-Prentice (Kalbfleisch and Prentice, 1980, p.86) and the Tsiatis/Link/Breslow, which reduce to the Kaplan-Meier and Fleming-Harrington estimates, respectively, when the weights are unity.

The Greenwood formula for the variance is a sum of terms d/(n*(n-m)), where d is the number of deaths at a given time point, n is the sum of weights for all individuals still at risk at that time, and m is the sum of weights for the deaths at that time. The justification is based on a binomial argument when weights are all equal to one; extension to the weighted case is ad hoc. Tsiatis (1981) proposes a sum of terms d/(n*n), based on a counting process argument which includes the weighted case.

The two variants of the F-H estimate have to do with how ties are handled. If there were 3 deaths out of 10 at risk, then the first increments the hazard by 3/10 and the second by 1/10 + 1/9 + 1/8. For the first method S(t) = exp(H), where H is the Nelson-Aalen cumulative hazard estimate, whereas the `fh2` method will give results S(t) results closer to the Kaplan-Meier.

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. When the baseline method is the Kaplan-Meier, this is known to converge to the maximum likelihood estimate.

The cumulative incidence curve is an alternative to the Kaplan-Meier for competing risks data. For instance, in patients with MGUS, conversion to an overt plasma cell malignancy occurs at a nearly constant rate among those still alive. A Kaplan-Meier estimate, treating death due to other causes as censored, gives a 20 year cumulate rate of 33% for the 241 early patients of Kyle. This estimates the incidence of conversion if all other causes of death were removed, which is an unrealistic assumption given the mean starting age of 63 and a median follow up of over 21 years.

The CI estimate, on the other hand, estimates the total number of conversions that will actually occur. Because the population is older, this is much smaller than the KM, 22% at 20 years for Kyle's data. If there were no censoring, then CI(t) could very simply be computed as total number of patients with progression by time t divided by the sample size n.

Value

an object of class `"survfit"`. See `survfit.object` for details. Methods defined for survfit objects are `print`, `plot`, `lines`, and `points`.

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.

Kablfleisch, 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.

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, 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)
```

```
#
# Time to progression/death for patients with monoclonal gammopathy
#  Competing risk curves (cumulative incidence)
fitKM <- survfit(Surv(stop, event=='progression') ~1, data=mgus1,
                    subset=(start==0))

fitCI <- survfit(Surv(stop, status*as.numeric(event), type="mstate") ~1,
                    data=mgus1, subset=(start==0))

# CI curves are always plotted from 0 upwards, rather than 1 down
plot(fitCI, xscale=365.25, xmax=7300, mark.time=FALSE,
            col=2:3, xlab="Years post diagnosis of MGUS")
lines(fitKM, fun='event', xscale=365.25, xmax=7300, mark.time=FALSE,
            conf.int=FALSE)
text(10, .4, "Competing risk: death", col=3)
text(16, .15,"Competing risk: progression", col=2)
text(15, .30,"KM:prog")
```

`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.

Usage

```
## S3 method for class 'matrix'
survfit(formula, p0, method = c("discrete", "matexp"), ...)
```

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.
`method`	use a product of discrete hazards, or a product of matrix exponentials. See details below.
`...`	further arguments for other methods

Details

When using predicted survival curves from a Cox model, it is possible to get predicted hazards that are greater than 1. For this reason it is safer to use exp(-hazard) rather than (1-hazard) as the multiplier for the probability. This is also the statndard approach for ordinarly survival curves from a Cox model. If the `method` argument is not supplied and the input values are derived from a Cox model, this is used as the default.

This approach is easy to use but has some caveats. First, the input curves must be consistent. The routine checks as best it can, but is easy fool. If you 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 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.

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)
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 subjects 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, the number of subjects that enter at time t.
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.
prev, p0	a multi-state survival will have the prev component instead of surv. It will be a matrix containing the estimated probability of each state at each time, one column per state. The p0 matrix contains the initial distribution of states. (On further reflection pstate= "probability in state" would have been a better label than "prevalence", but by that point too many other packages were dependent on the form of the result.)
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 pstate. This difference is a reflection of the fact that each is the natural calculation for that case.
upper	upper confidence limit for the survival curve or probability
lower	lower confidence limit for the survival curve or probability
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.
start.time	the value specified for the start.time argument, if it was used in the call.
n.all	for counting process data, and any time that the start.time argument was used, this contains the total number of observations that were available. Not all may have been used in creating the curve, in which case this value will be larger than n above.

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.

Structure

The following components must be included in a legitimate survfit or survfitms object.

Subscripts

Survfit objects that contain multiple survival curves can be subscripted. This is most often used to plot a subset of the curves. If the surv or pstate object is a matrix then the survfit object will be treated as a matrix, otherwise only a single subscript is used.

See Also

plot.survfit, summary.survfit, print.survfit, survfit.

---

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=Kalbfleish-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`

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, 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' standard errors, based on independence of individuals if there is no cluster() term in the formula, based on independence of clusters if there is.
...	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.

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,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 parameterizaion 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", "Logisitic" 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 mulitple 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, event)
```

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

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.

The routine will normally be 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 `end` and `event` arguments give backwards compatability with an older version which lacked a formula interface.

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` of course. 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)

# a cox.zph plot of the data suggests that the effect of Karnofsky score
#  has faded by 120 days-- plot(cox.zph(fit)[1]) -- and begins to
#  diminish by 60 days.
# Fit a model with separate coefficeints 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)]))
```

`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

```
mdy.date <- function(m,d,y)
   as.Date(paste(ifelse(y<100, y+1900, y), m, d, sep='/'))
temp1 <- mdy.date(6,6,36)
temp2 <- mdy.date(6,6,55)# Now compare the results from person-years
#
temp.age <- tcut(temp2-temp1, floor(c(-1, (18:31 * 365.24))),
labels=c('0-18', paste(18:30, 19:31, sep='-')))
temp.yr  <- tcut(temp2, mdy.date(1,1,1954:1965), labels=1954:1964)
temp.time <- 3700   #total days of fu
py1 <- pyears(temp.time ~ temp.age + temp.yr, scale=1) #output in days
py1
```

---

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 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 and na.rm. 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 optional 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)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 decribes the occurent 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 count is set to the value of `x`. If the option `na.rm=TRUE` missing values of `x` are first removed.

  newname = cumtdc(y,x)Similar to tdc, except that the event count is accumulated over time for each subject.

  newname = event(y,x)Mark an event at time y. In the ususal 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.

Value

a data frame with two extra attributes `tname` and `tcount`. The first contains the names of the key variables; it's persistence from call to call allows the user to avoid constantly reentering the `options` argument. 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". Intervals are assumed to be of the form (a,b], open on the left and closed on the right; a new time exactly coincident with the start of a subject's times will thus be "early".

The most common type will usually be "within", for 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 occurence 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` occurence 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 occurence lands at the join. Events will go to the earlier interval and counts to the latter one.

These extra attributes are ephemeral, and will be discarded if the dataframe is modified in any way. This is intentional.

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
```

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

```
data("transplant")
```

Format

A data frame with 815 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.

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 period
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, temp)
legend(450, .35, levels(transplant$abo), lty=1, col=1:4, lwd=2, bty='n')

# 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)
```

`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

Usage

```
data(uspop2)
```

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 epidemiolgy 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"))
```

`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
```

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, 1=yes

Source

D Kalbfleisch and RL Prentice (1980), *The Statistical Analysis of Failure Time Data*. Wiley, New York.

Index

∗Topic **datasets, survival**
genfan, 38
∗Topic **datasets**
aml, 8
bladder, 12
cgd, 16
cgd0, 17
flchain, 34
heart, 39
logan, 44
lung, 45
mgus, 47
mgus2, 49
nwtco, 53
ovarian, 54
pbc, 55
pbcseq, 56
ratetables, 78
rats, 79
rats2, 79
retinopathy, 83
stanford2, 85
tobin, 128
transplant, 129
uspop2, 131
veteran, 132
∗Topic **distribution**
dsurvreg, 31
∗Topic **hplot**
plot.survfit, 60
∗Topic **manip**
neardate, 52
∗Topic **models**
anova.coxph, 8
attrassign, 9
clogit, 18
∗Topic **print**
print.summary.survfit, 68
∗Topic **regression**
anova.coxph, 8
survreg.object, 122
∗Topic **survival**
aareg, 3
agreg.fit, 6
anova.coxph, 8
basehaz, 11
bladder, 12
cch, 13
cgd, 16
cgd0, 17
clogit, 18
cluster, 20
colon, 21
cox.zph, 22
coxph, 23
coxph.control, 27
coxph.detail, 28
coxph.object, 29
coxph.wtest, 30
finegray, 33
frailty, 36
heart, 39
is.ratetable, 40
kidney, 41
lines.survfit, 42
logLik.coxph, 44
mgus, 47
model.frame.coxph, 50
model.matrix.coxph, 51
ovarian, 54
plot.cox.zph, 59
plot.survfit, 60
predict.coxph, 62
predict.survreg, 64
print.aareg, 66
print.summary.survexp, 67
print.survfit, 69
pspline, 70

pyears, 72
quantile.survfit, 75
ratetable, 76
ratetableDate, 77
ratetables, 78
rats, 79
rats2, 79
residuals.coxph, 80
residuals.survreg, 82
ridge, 84
stanford2, 85
strata, 86
summary.aareg, 87
summary.coxph, 88
summary.survexp, 89
summary.survfit, 91
Surv, 92
survConcordance, 94
survdiff, 96
survexp, 98
survexp.fit, 101
survexp.object, 102
survfit, 103
survfit.coxph, 104
survfit.formula, 107
survfit.matrix, 111
survfit.object, 113
survfitcoxph.fit, 114
survobrien, 116
survreg, 117
survreg.control, 119
survreg.distributions, 120
survreg.object, 122
survregDtest, 123
survSplit, 124
tcut, 125
tmerge, 126
untangle.specials, 130

∗Topic **utilities**
neardate, 52
survSplit, 124

[.Surv *(Surv)*, 92
[.cox.zph *(cox.zph)*, 22
[.coxph.penalty *(coxph)*, 23
[.ratetable *(ratetable)*, 76
[.ratetable2 *(ratetable)*, 76
[.survfit *(survfit.formula)*, 107
[.tcut *(tcut)*, 125

aareg, 3
agreg.fit, 6
aml, 8
anova, *9*
anova.coxph, 8
anova.coxphlist *(anova.coxph)*, 8
anova.survreg *(survreg)*, 117
anova.survreglist *(survreg)*, 117
as.character.Surv *(Surv)*, 92
as.data.frame.Surv *(Surv)*, 92
as.matrix.Surv *(Surv)*, 92
as.POSIXct, *52*
attrassign, 9

basehaz, 11
bladder, 12
bladder1 *(bladder)*, 12
bladder2 *(bladder)*, 12

cancer *(lung)*, 45
cch, 13
cgd, 16, *18*
cgd0, 17
clogit, 18
cluster, 20, *26*
colon, 21
cox.zph, 22, *30*, *59*
coxph, *7*, *9*, *19*, *20*, *23*, 23, *28–30*, *34*, *37*, *59*, *62*, *64*, *71*, *81*, *84*, *86*, *94*, *106*, *110*
coxph.control, *24*, 27
coxph.detail, 28, *30*
coxph.fit *(agreg.fit)*, 6
coxph.object, 29
coxph.wtest, 30
cut, *125*, *126*

dsurvreg, 31

extractAIC.coxph.penal *(coxph.object)*, 29

findInterval, *53*
finegray, 33
flchain, 34
format.Surv *(Surv)*, 92
frailty, *26*, 36, *71*, *84*, *118*

genfan, 38
glm, *19*

heart, 39, *85*

interaction, *86*
is.na.ratetable *(*ratetable*)*, 76
is.na.Surv *(*Surv*)*, 92
is.ratetable, 40, *76*
is.Surv *(*Surv*)*, 92

jasa *(*heart*)*, 39
jasa1 *(*heart*)*, 39

kidney, 41

labels.survreg *(*survreg*)*, 117
leukemia *(*aml*)*, 8
levels.tcut *(*tcut*)*, 125
lines, *43*
lines.survexp *(*lines.survfit*)*, 42
lines.survfit, 42, *62*, *106*, *110*
lm, *122*
logan, 44
logLik, *45*
logLik.coxph, 44
logLik.survreg *(*logLik.coxph*)*, 44
lung, 45

match, *53*
Math.ratetable *(*is.ratetable*)*, 40
Math.Surv *(*Surv*)*, 92
mgus, 47
mgus1 *(*mgus*)*, 47
mgus2, 49
model.frame, *50*
model.frame.coxph, 50
model.frame.survreg *(*survreg*)*, 117
model.matrix, *10*, *51*
model.matrix.coxph, 51

neardate, 52, *128*
Normal, *32*
nwtco, 53

Ops.ratetable *(*is.ratetable*)*, 40
Ops.Surv *(*Surv*)*, 92
options, *68*
ovarian, 54

par, *43*, *62*
pbc, 55
pbcseq, 56

plogis, *121*
plot.aareg, 58
plot.cox.zph, 59
plot.survfit, *42*, *43*, 60, *103*, *104*, *106*, *110*, *114*
pnorm, *121*
points.survfit, *62*
points.survfit *(*lines.survfit*)*, 42
predict, *64*
predict.coxph, 62
predict.survreg, 64, *83*, *85*
print, *68*
print.aareg, 66
print.cox.zph *(*cox.zph*)*, 22
print.coxph *(*coxph.object*)*, 29
print.coxph.null *(*coxph*)*, 23
print.coxph.penal *(*coxph*)*, 23
print.ratetable *(*ratetable*)*, 76
print.summary.coxph, 67
print.summary.survexp, 67
print.summary.survfit, 68, *92*
print.summary.survreg *(*survreg*)*, 117
print.Surv *(*Surv*)*, 92
print.survdiff *(*survdiff*)*, 96
print.survexp *(*survexp*)*, 98
print.survfit, 69, *76*, *91*, *103*, *104*, *106*, *110*, *113*, *114*
print.survreg *(*survreg.object*)*, 122
print.survreg.penal *(*survreg*)*, 117
pspline, *26*, 70, *84*, *118*
psplineinverse *(*pspline*)*, 70
psurvreg *(*dsurvreg*)*, 31
pt, *121*
pweibull, *121*
pyears, *40*, 72, *77*, *100*, *126*

qsurvreg, *76*
qsurvreg *(*dsurvreg*)*, 31
quantile.survfit, *70*, 75, *104*
quantile.survfitms *(*quantile.survfit*)*, 75

ratetable, *74*, 76
ratetableDate, 77
ratetables, 78
rats, 79
rats2, 79
reshape, *125*
residuals, *82*

residuals.coxph, *29*, *30*, 80
residuals.survreg, *65*, 82
retinopathy, 83
ridge, *26*, *71*, 84, *118*
rsurvreg (dsurvreg), 31
rweibull, *121*

stanford2, *40*, 85
strata, *19*, *26*, 86, *106*
summary.aareg, 87
summary.coxph, 88
summary.coxph.penal (coxph), 23
summary.ratetable (ratetable), 76
Summary.Surv (Surv), 92
summary.survexp, 89, *103*
summary.survfit, *68*, *70*, 91, *104*, *114*
summary.survreg (survreg.object), 122
Surv, *13*, *23*, *26*, *74*, 92, *106*, *110*, *125*
survConcordance, 94
survdiff, 96, *117*
survexp, *40*, *43*, *68*, *74*, *76–78*, *90*, 98, *101–103*
survexp.az (ratetables), 78
survexp.azr (ratetables), 78
survexp.fit, *100*, 101
survexp.fl (ratetables), 78
survexp.flr (ratetables), 78
survexp.mn (ratetables), 78
survexp.mnwhite (ratetables), 78
survexp.object, 102
survexp.us, *76*, *100*, *102*
survexp.us (ratetables), 78
survexp.usr (ratetables), 78
survexp.wnc (ratetables), 78
survfit, *26*, *30*, *43*, *62*, *76*, *92*, *94*, *100*, 103, *106*, *112*, *114*
survfit.coxph, *11*, *104*, 104, *110*, *115*
survfit.formula, *104*, 107
survfit.matrix, 111
survfit.object, *104*, *110*, 113
survfitcoxph.fit, 114
survfitms.object (survfit.object), 113
survobrien, 116
survReg (survreg), 117
survreg, *20*, *30*, *32*, *37*, *65*, *71*, *84*, *94*, 117, *119–123*
survreg.control, *118*, 119
survreg.distributions, *118*, 120, *123*
survreg.object, *118*, 122
survregDtest, *121*, 123
survSplit, 124

tcut, 125
termplot, *64*
terms, *10*
tmerge, 126
tobin, 128
transplant, 129

untangle.specials, 130
uspop, *131*
uspop2, 131

vcov.coxph (coxph), 23
vcov.survreg (survreg), 117
veteran, 132