Package 'timereg'

July 30, 2019

Title Flexible Regression Models for Survival Data

Version 1.9.4

Date 2019-07-29

Author Thomas Scheike with contributions from Torben Martinussen, Jeremy Silver and Klaus Holst

Maintainer Thomas Scheike <ts@biostat.ku.dk>

Description Programs for Martinussen and Scheike (2006), `Dynamic Regression Models for Survival Data', Springer Verlag. Plus more recent developments. Additive survival model, semiparametric proportional odds model, fast cumulative residuals, excess risk models and more. Flexible competing risks regression including GOF-tests. Two-stage frailty modelling. PLS for the additive risk model. Lasso in the 'ahaz' package.

LazyLoad yes

URL https://github.com/scheike/timereg.git

Depends R (>= 2.15), survival

Imports lava, numDeriv, stats, graphics, grDevices, utils, methods

Suggests mets,

License GPL (>= 2)

RoxygenNote 6.1.1

NeedsCompilation yes

Repository CRAN

Date/Publication 2019-07-30 10:50:02 UTC

R topics documented:

`aalen` *Fit additive hazards model*

Description

Fits both the additive hazards model of Aalen and the semi-parametric additive hazards model of McKeague and Sasieni. Estimates are un-weighted. Time dependent variables and counting process data (multiple events per subject) are possible.

Usage

```
aalen(formula = formula(data), data = sys.parent(), start.time = 0,
  max.time = NULL, robust = 1, id = NULL, clusters = NULL,
  residuals = 0, n.sim = 1000, weighted.test = 0, covariance = 0,
  resample.iid = 0, deltaweight = 1, silent = 1, weights = NULL,
  max.clust = 1000, gamma = NULL, offsets = 0, caseweight = NULL)
```

Arguments

`formula`	a formula object with the response on the left of a '~' operator, and the independent terms on the right as regressors.The response must be a survival object as returned by the 'Surv' function. Time- invariant regressors are specified by the wrapper const(), and cluster variables (for computing robust variances) by the wrapper cluster().
`data`	a data.frame with the variables.
`start.time`	start of observation period where estimates are computed.
`max.time`	end of observation period where estimates are computed. Estimates thus computed from [start.time, max.time]. Default is max of data.
`robust`	to compute robust variances and construct processes for resampling. May be set to 0 to save memory.
`id`	For timevarying covariates the variable must associate each record with the id of a subject.
`clusters`	cluster variable for computation of robust variances.
`residuals`	to returns residuals that can be used for model validation in the function cum.residuals
`n.sim`	number of simulations in resampling.
`weighted.test`	to compute a variance weighted version of the test-processes used for testing time-varying effects.
`covariance`	to compute covariance estimates for nonparametric terms rather than just the variances.
`resample.iid`	to return i.i.d. representation for nonparametric and parametric terms.
`deltaweight`	uses weights to estimate semiparametric model, under construction, default=1 is standard least squares estimates

silent	set to 0 to print warnings for non-inverible design-matrices for different time-points, default is 1.
weights	weights for estimating equations.
max.clust	sets the total number of i.i.d. terms in i.i.d. decompostition. This can limit the amount of memory used by coarsening the clusters. When NULL then all clusters are used. Default is 1000 to save memory and time.
gamma	fixes gamme at this value for estimation.
offsets	offsets for the additive model, to make excess risk modelling.
caseweight	caseweight: mutiplied onto dN for score equations.

Details

Resampling is used for computing p-values for tests of time-varying effects.

The modelling formula uses the standard survival modelling given in the **survival** package.

The data for a subject is presented as multiple rows or 'observations', each of which applies to an interval of observation (start, stop]. For counting process data with the)start,stop] notation is used, the 'id' variable is needed to identify the records for each subject. The program assumes that there are no ties, and if such are present random noise is added to break the ties.

Value

returns an object of type "aalen". With the following arguments:

cum	cumulative timevarying regression coefficient estimates are computed within the estimation interval.
var.cum	the martingale based pointwise variance estimates for cumulatives.
robvar.cum	robust pointwise variances estimates for cumulatives.
gamma	estimate of parametric components of model.
var.gamma	variance for gamma.
robvar.gamma	robust variance for gamma.
residuals	list with residuals. Estimated martingale increments (dM) and corresponding time vector (time).
obs.testBeq0	observed absolute value of supremum of cumulative components scaled with the variance.
pval.testBeq0	p-value for covariate effects based on supremum test.
sim.testBeq0	resampled supremum values.
obs.testBeqC	observed absolute value of supremum of difference between observed cumulative process and estimate under null of constant effect.
pval.testBeqC	p-value based on resampling.
sim.testBeqC	resampled supremum values.

`obs.testBeqC.is`
: observed integrated squared differences between observed cumulative and estimate under null of constant effect.

`pval.testBeqC.is`
: p-value based on resampling.

`sim.testBeqC.is`
: resampled supremum values.

`conf.band`
: resampling based constant to construct robust 95% uniform confidence bands.

`test.procBeqC`
: observed test-process of difference between observed cumulative process and estimate under null of constant effect over time.

`sim.test.procBeqC`
: list of 50 random realizations of test-processes under null based on resampling.

`covariance`
: covariances for nonparametric terms of model.

`B.iid`
: Resample processes for nonparametric terms of model.

`gamma.iid`
: Resample processes for parametric terms of model.

`deviance`
: Least squares of increments.

Author(s)

Thomas Scheike

References

Martinussen and Scheike, Dynamic Regression Models for Survival Data, Springer (2006).

Examples

```
data(sTRACE)
# Fits Aalen model
out<-aalen(Surv(time,status==9)~age+sex+diabetes+chf+vf,
sTRACE,max.time=7,n.sim=100)

summary(out)
par(mfrow=c(2,3))
plot(out)

# Fits semi-parametric additive hazards model
out<-aalen(Surv(time,status==9)~const(age)+const(sex)+const(diabetes)+chf+vf,
sTRACE,max.time=7,n.sim=100)

summary(out)
par(mfrow=c(2,3))
plot(out)

## Excess risk additive modelling
data(mela.pop)
dummy<-rnorm(nrow(mela.pop));
```

```
# Fits Aalen model  with offsets
out<-aalen(Surv(start,stop,status==1)~age+sex+const(dummy),
mela.pop,max.time=7,n.sim=100,offsets=mela.pop$rate,id=mela.pop$id,
gamma=0)
summary(out)
par(mfrow=c(2,3))
plot(out,main="Additive excess riks model")

# Fits semi-parametric additive hazards model  with offsets
out<-aalen(Surv(start,stop,status==1)~age+const(sex),
mela.pop,max.time=7,n.sim=100,offsets=mela.pop$rate,id=mela.pop$id)
summary(out)
plot(out,main="Additive excess riks model")
```

---

bmt *The Bone Marrow Transplant Data*

Description

Bone marrow transplant data with 408 rows and 5 columns.

Format

The data has 408 rows and 5 columns.

cause a numeric vector code. Survival status. 1: dead from treatment related causes, 2: relapse , 0: censored.

time a numeric vector. Survival time.

platelet a numeric vector code. Plalelet 1: more than 100 x 10^9 per L, 0: less.

tcell a numeric vector. T-cell depleted BMT 1:yes, 0:no.

age a numeric vector code. Age of patient, scaled and centered ((age-35)/15).

Source

Simulated data

References

NN

Examples

```
data(bmt)
names(bmt)
```

`cd4` *The multicenter AIDS cohort study*

Description

CD4 counts collected over time.

Format

This data frame contains the following columns:

obs a numeric vector. Number of observations.

id a numeric vector. Id of subject.

visit a numeric vector. Timings of the visits in years.

smoke a numeric vector code. 0: non-smoker, 1: smoker.

age a numeric vector. Age of the patient at the start of the trial.

cd4 a numeric vector. CD4 percentage at the current visit.

cd4.prev a numeric vector. CD4 level at the preceding visit.

precd4 a numeric vector. Post-infection CD4 percentage.

lt a numeric vector. Gives the starting time for the time-intervals.

rt a numeric vector. Gives the stopping time for the time-interval.

Source

MACS Public Use Data Set Release PO4 (1984-1991). See reference.

References

Kaslow et al. (1987), The multicenter AIDS cohort study: rational, organisation and selected characteristics of the participants. Am. J. Epidemiology 126, 310–318.

Examples

```
data(cd4)
names(cd4)
```

comp.risk *Competings Risks Regression*

Description

Fits a semiparametric model for the cause-specific quantities :

$$P(T < t, cause = 1|x, z) = P_1(t, x, z) = h(g(t, x, z))$$

for a known link-function $h()$ and known prediction-function $g(t, x, z)$ for the probability of dying from cause 1 in a situation with competing causes of death.

Usage

```
comp.risk(formula, data = sys.parent(), cause, times = NULL,
  Nit = 50, clusters = NULL, est = NULL, fix.gamma = 0,
  gamma = 0, n.sim = 0, weighted = 0, model = "fg", detail = 0,
  interval = 0.01, resample.iid = 1, cens.model = "KM",
  cens.formula = NULL, time.pow = NULL, time.pow.test = NULL,
  silent = 1, conv = 1e-06, weights = NULL, max.clust = 1000,
  n.times = 50, first.time.p = 0.05, estimator = 1, trunc.p = NULL,
  cens.weights = NULL, admin.cens = NULL, conservative = 1,
  monotone = 0, step = NULL)
```

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 'Event' function. The status indicator is not important here. Time-invariant regressors are specified by the wrapper const(), and cluster variables (for computing robust variances) by the wrapper cluster().
data	a data.frame with the variables.
cause	For competing risk models specificies which cause we consider.
times	specifies the times at which the estimator is considered. Defaults to all the times where an event of interest occurs, with the first 10 percent or max 20 jump points removed for numerical stability in simulations.
Nit	number of iterations for Newton-Raphson algorithm.
clusters	specifies cluster structure, for backwards compability.
est	possible starting value for nonparametric component of model.
fix.gamma	to keep gamma fixed, possibly at 0.
gamma	starting value for constant effects.
n.sim	number of simulations in resampling.
weighted	Not implemented. To compute a variance weighted version of the test-processes used for testing time-varying effects.

`model` "additive", "prop"ortional, "rcif", or "logistic".

`detail` if 0 no details are printed during iterations, if 1 details are given.

`interval` specifies that we only consider timepoints where the Kaplan-Meier of the censoring distribution is larger than this value.

`resample.iid` to return the iid decomposition, that can be used to construct confidence bands for predictions

`cens.model` specified which model to use for the ICPW, KM is Kaplan-Meier alternatively it may be "cox"

`cens.formula` specifies the regression terms used for the regression model for chosen regression model. When cens.model is specified, the default is to use the same design as specified for the competing risks model.

`time.pow` specifies that the power at which the time-arguments is transformed, for each of the arguments of the const() terms, default is 1 for the additive model and 0 for the proportional model.

`time.pow.test`
specifies that the power the time-arguments is transformed for each of the arguments of the non-const() terms. This is relevant for testing if a coefficient function is consistent with the specified form A_l(t)=beta_l t^time.pow.test(l). Default is 1 for the additive model and 0 for the proportional model.

`silent` if 0 information on convergence problems due to non-invertible derviates of scores are printed.

`conv` gives convergence criterie in terms of sum of absolute change of parameters of model

`weights` weights for estimating equations.

`max.clust` sets the total number of i.i.d. terms in i.i.d. decompostition. This can limit the amount of memory used by coarsening the clusters. When NULL then all clusters are used. Default is 1000 to save memory and time.

`n.times` only uses 50 points for estimation, if NULL then uses all points, subject to p.start condition.

`first.time.p` first point for estimation is pth percentile of cause jump times.

`estimator` default estimator is 1.

`trunc.p` truncation weight for delayed entry, P(T > entry.time | Z_i), typically Cox model.

`cens.weights` censoring weights can be given here rather than calculated using the KM, cox or aalen models.

`admin.cens` censoring times for the administrative censoring

`conservative` set to 0 to compute correct variances based on censoring weights, default is conservative estimates that are much quicker.

`monotone` monotone=0, uses estimating equations

$$(D_\beta P_1) w(t)(Y(t)/G_c(t) - P_1(t,X)) and$$

montone 1 uses

$$w(t)(Y(t)/G_c(t) - P_1(t,X)) and$$

`step` step size for Fisher-Scoring algorithm.

Details

We consider the following models : 1) the additive model where $h(x) = 1 - \exp(-x)$ and

$$g(t, x, z) = x^T A(t) + (diag(t^p)z)^T \beta$$

2) the proportional setting that includes the Fine & Gray (FG) "prop" model and some extensions where $h(x) = 1 - \exp(-\exp(x))$ and

$$g(t, x, z) = (x^T A(t) + (diag(t^p)z)^T \beta)$$

The FG model is obtained when $x = 1$, but the baseline is parametrized as $\exp(A(t))$.

The "fg" model is a different parametrization that contains the FG model, where $h(x) = 1 - \exp(-x)$ and

$$g(t, x, z) = (x^T A(t)) \exp((diag(t^p)z)^T \beta)$$

The FG model is obtained when $x = 1$.

3) a "logistic" model where $h(x) = \exp(x)/(1 + \exp(x))$ and

$$g(t, x, z) = x^T A(t) + (diag(t^p)z)^T \beta$$

The "logistic2" is

$$P_1(t, x, z) = x^T A(t) exp((diag(t^p)z)^T \beta)/(1 + x^T A(t) exp((diag(t^p)z)^T \beta))$$

The simple logistic model with just a baseline can also be fitted by an alternative procedure that has better small sample properties see prop.odds.subist().

4) the relative cumulative incidence function "rcif" model where $h(x) = \exp(x)$ and

$$g(t, x, z) = x^T A(t) + (diag(t^p)z)^T \beta$$

The "rcif2"

$$P_1(t, x, z) = (x^T A(t)) \exp((diag(t^p)z)^T \beta)$$

Where p by default is 1 for the additive model and 0 for the other models. In general p may be powers of the same length as z.

Since timereg version 1.8.4. the response must be specified with the `Event` function instead of the `Surv` function and the arguments. For example, if the old code was

comp.risk(Surv(time,cause>0)~x1+x2,data=mydata,cause=mydata$cause,causeS=1)

the new code is

comp.risk(Event(time,cause)~x1+x2,data=mydata,cause=1)

Also the argument cens.code is now obsolete since cens.code is an argument of `Event`.

Value

returns an object of type 'comprisk'. With the following arguments:

`cum`	cumulative timevarying regression coefficient estimates are computed within the estimation interval.
`var.cum`	pointwise variances estimates.
`gamma`	estimate of proportional odds parameters of model.
`var.gamma`	variance for gamma.
`score`	sum of absolute value of scores.
`gamma2`	estimate of constant effects based on the non-parametric estimate. Used for testing of constant effects.
`obs.testBeq0`	observed absolute value of supremum of cumulative components scaled with the variance.
`pval.testBeq0`	p-value for covariate effects based on supremum test.
`obs.testBeqC`	observed absolute value of supremum of difference between observed cumulative process and estimate under null of constant effect.
`pval.testBeqC`	p-value based on resampling.
`obs.testBeqC.is`	observed integrated squared differences between observed cumulative and estimate under null of constant effect.
`pval.testBeqC.is`	p-value based on resampling.
`conf.band`	resampling based constant to construct 95% uniform confidence bands.
`B.iid`	list of iid decomposition of non-parametric effects.
`gamma.iid`	matrix of iid decomposition of parametric effects.
`test.procBeqC`	observed test process for testing of time-varying effects
`sim.test.procBeqC`	50 resample processes for for testing of time-varying effects
`conv`	information on convergence for time points used for estimation.

Author(s)

Thomas Scheike

References

Scheike, Zhang and Gerds (2008), Predicting cumulative incidence probability by direct binomial regression,Biometrika, 95, 205-220.

Scheike and Zhang (2007), Flexible competing risks regression modelling and goodness of fit, LIDA, 14, 464-483.

Martinussen and Scheike (2006), Dynamic regression models for survival data, Springer.

Examples

```
data(bmt);

clust <- rep(1:204,each=2)
addclust<-comp.risk(Event(time,cause)~platelet+age+tcell+cluster(clust),data=bmt,
cause=1,resample.iid=1,n.sim=100,model="additive")
###

addclust<-comp.risk(Event(time,cause)~+1+cluster(clust),data=bmt,cause=1,
    resample.iid=1,n.sim=100,model="additive")
pad <- predict(addclust,X=1)
plot(pad)

add<-comp.risk(Event(time,cause)~platelet+age+tcell,data=bmt,
cause=1,resample.iid=1,n.sim=100,model="additive")
summary(add)

par(mfrow=c(2,4))
plot(add);
### plot(add,score=1) ### to plot score functions for test

ndata<-data.frame(platelet=c(1,0,0),age=c(0,1,0),tcell=c(0,0,1))
par(mfrow=c(2,3))
out<-predict(add,ndata,uniform=1,n.sim=100)
par(mfrow=c(2,2))
plot(out,multiple=0,uniform=1,col=1:3,lty=1,se=1)

add<-comp.risk(Event(time,cause)~platelet+age+tcell,data=bmt,
       cause=1,resample.iid=0,n.sim=0,cens.model="cox",
       cens.formula=~factor(platelet),model="additive")

out<-predict(add,ndata,se=0,uniform=0)
par(mfrow=c(2,2))
plot(out,multiple=0,se=0,uniform=0,col=1:3,lty=1)

## fits additive model with some constant effects
add.sem<-comp.risk(Event(time,cause)~
const(platelet)+const(age)+const(tcell),data=bmt,
cause=1,resample.iid=1,n.sim=100,model="additive")
summary(add.sem)

out<-predict(add.sem,ndata,uniform=1,n.sim=100)
par(mfrow=c(2,2))
plot(out,multiple=0,uniform=1,col=1:3,lty=1,se=0)

## Fine & Gray model
fg<-comp.risk(Event(time,cause)~
const(platelet)+const(age)+const(tcell),data=bmt,
cause=1,resample.iid=1,model="fg",n.sim=100)
summary(fg)
```

```
out<-predict(fg,ndata,uniform=1,n.sim=100)

par(mfrow=c(2,2))
plot(out,multiple=1,uniform=0,col=1:3,lty=1,se=0)

## extended model with time-varying effects
fg.npar<-comp.risk(Event(time,cause)~platelet+age+const(tcell),
data=bmt,cause=1,resample.iid=1,model="prop",n.sim=100)
summary(fg.npar);

out<-predict(fg.npar,ndata,uniform=1,n.sim=100)
head(out$P1[,1:5]); head(out$se.P1[,1:5])

par(mfrow=c(2,2))
plot(out,multiple=1,uniform=0,col=1:3,lty=1,se=0)

## Fine & Gray model with alternative parametrization for baseline
fg2<-comp.risk(Event(time,cause)~const(platelet)+const(age)+const(tcell),data=bmt,
cause=1,resample.iid=1,model="prop",n.sim=100)
summary(fg2)

######################################################################
## Delayed entry models,
######################################################################
nn <- nrow(bmt)
entrytime <- rbinom(nn,1,0.5)*(bmt$time*runif(nn))
bmt$entrytime <- entrytime
times <- seq(5,70,by=1)

bmtw <- prep.comp.risk(bmt,times=times,time="time",entrytime="entrytime",cause="cause")

## non-parametric model
outnp <- comp.risk(Event(time,cause)~tcell+platelet+const(age),
   data=bmtw,cause=1,fix.gamma=1,gamma=0,
 cens.weights=bmtw$cw,weights=bmtw$weights,times=times,n.sim=0)
par(mfrow=c(2,2))
plot(outnp)

outnp <- comp.risk(Event(time,cause)~tcell+platelet,
   data=bmtw,cause=1,
 cens.weights=bmtw$cw,weights=bmtw$weights,times=times,n.sim=0)
par(mfrow=c(2,2))
plot(outnp)


## semiparametric model
out <- comp.risk(Event(time,cause)~const(tcell)+const(platelet),data=bmtw,cause=1,
 cens.weights=bmtw$cw,weights=bmtw$weights,times=times,n.sim=0)
summary(out)
```

const *Identifies parametric terms of model*

Description

Specifies which of the regressors that have constant effect.

Usage

```
const(x)
```

Arguments

x	variable

Author(s)

Thomas Scheike

cox *Identifies proportional excess terms of model*

Description

Specifies which of the regressors that lead to proportional excess hazard

Usage

```
cox(x)
```

Arguments

x	variable

Author(s)

Thomas Scheike

`cox.aalen` *Fit Cox-Aalen survival model*

Description

Fits an Cox-Aalen survival model. Time dependent variables and counting process data (multiple events per subject) are possible.

Usage

```
cox.aalen(formula = formula(data), data = sys.parent(), beta = NULL,
  Nit = 20, detail = 0, start.time = 0, max.time = NULL,
  id = NULL, clusters = NULL, n.sim = 500, residuals = 0,
  robust = 1, weighted.test = 0, covariance = 0, resample.iid = 1,
  weights = NULL, rate.sim = 1, beta.fixed = 0, max.clust = 1000,
  exact.deriv = 1, silent = 1, max.timepoint.sim = 100,
  basesim = 0, offsets = NULL, strata = NULL, propodds = 0,
  caseweight = NULL)
```

Arguments

`formula`	a formula object with the response on the left of a '~' operator, and the independent terms on the right as regressors. The response must be a survival object as returned by the 'Surv' function. Terms with a proportional effect are specified by the wrapper prop(), and cluster variables (for computing robust variances) by the wrapper cluster().
`data`	a data.frame with the variables.
`beta`	starting value for relative risk estimates.
`Nit`	number of iterations for Newton-Raphson algorithm.
`detail`	if 0 no details is printed during iterations, if 1 details are given.
`start.time`	start of observation period where estimates are computed.
`max.time`	end of observation period where estimates are computed. Estimates thus computed from [start.time, max.time]. Default is max of data.
`id`	For timevarying covariates the variable must associate each record with the id of a subject.
`clusters`	cluster variable for computation of robust variances.
`n.sim`	number of simulations in resampling.
`residuals`	to returns residuals that can be used for model validation in the function cum.residuals. Estimated martingale increments (dM) and corresponding time vector (time). When rate.sim=1 returns estimated martingales, dM_i(t) and if rate.sim=0, returns a matrix of dN_i(t).
`robust`	to compute robust variances and construct processes for resampling. May be set to 0 to save memory and time, in particular for rate.sim=1.

`weighted.test`
: to compute a variance weighted version of the test-processes used for testing time-varying effects.

`covariance`
: to compute covariance estimates for nonparametric terms rather than just the variances.

`resample.iid`
: to return i.i.d. representation for nonparametric and parametric terms. based on counting process or martingale resduals (rate.sim).

`weights`
: weights for weighted analysis.

`rate.sim`
: rate.sim=1 such that resampling of residuals is based on estimated martingales and thus valid in rate case, rate.sim=0 means that resampling is based on counting processes and thus only valid in intensity case.

`beta.fixed`
: option for computing score process for fixed relative risk parameter

`max.clust`
: sets the total number of i.i.d. terms in i.i.d. decompostition. This can limit the amount of memory used by coarsening the clusters. When NULL then all clusters are used. Default is 1000 to save memory and time.

`exact.deriv`
: if 1 then uses exact derivative in last iteration, if 2 then uses exact derivate for all iterations, and if 0 then uses approximation for all computations and there may be a small bias in the variance estimates. For Cox model always exact and all options give same results.

`silent`
: if 1 then opppresses some output.

`max.timepoint.sim`
: considers only this resolution on the time scale for simulations, see time.sim.resolution argument

`basesim`
: 1 to get simulations for cumulative baseline, including tests for contant effects.

`offsets`
: offsets for analysis on log-scale. RR=exp(offsets+ x beta).

`strata`
: future option for making strata in a different day than through X design in cox-aalen model (~-1+factor(strata)).

`propodds`
: if 1 will fit the proportional odds model. Slightly less efficient than prop.odds() function but much quicker, for large data this also works.

`caseweight`
: these weights have length equal to number of jump times, and are multiplied all jump times dN. Useful for getting the program to fit for example the proportional odds model or frailty models.

Details

$$\lambda_i(t) = Y_i(t)(X_i^T(t)\alpha(t)) \exp(Z_i^T \beta)$$

The model thus contains the Cox's regression model as special case.

To fit a stratified Cox model it is important to parametrize the baseline apppropriately (see example below).

Resampling is used for computing p-values for tests of time-varying effects. Test for proportionality is considered by considering the score processes for the proportional effects of model.

The modelling formula uses the standard survival modelling given in the **survival** package.

The data for a subject is presented as multiple rows or 'observations', each of which applies to an interval of observation (start, stop]. For counting process data with the)start,stop] notation is used, the 'id' variable is needed to identify the records for each subject. The program assumes that there are no ties, and if such are present random noise is added to break the ties.

Value

returns an object of type "cox.aalen". With the following arguments:

`cum`	cumulative timevarying regression coefficient estimates are computed within the estimation interval.
`var.cum`	the martingale based pointwise variance estimates.
`robvar.cum`	robust pointwise variances estimates.
`gamma`	estimate of parametric components of model.
`var.gamma`	variance for gamma sandwhich estimator based on optional variation estimator of score and 2nd derivative.
`robvar.gamma`	robust variance for gamma.
`residuals`	list with residuals.
`obs.testBeq0`	observed absolute value of supremum of cumulative components scaled with the variance.
`pval.testBeq0`	p-value for covariate effects based on supremum test.
`sim.testBeq0`	resampled supremum values.
`obs.testBeqC`	observed absolute value of supremum of difference between observed cumulative process and estimate under null of constant effect.
`pval.testBeqC`	p-value based on resampling.
`sim.testBeqC`	resampled supremum values.
`obs.testBeqC.is`	observed integrated squared differences between observed cumulative and estimate under null of constant effect.
`pval.testBeqC.is`	p-value based on resampling.
`sim.testBeqC.is`	resampled supremum values.
`conf.band`	resampling based constant to construct robust 95% uniform confidence bands.
`test.procBeqC`	observed test-process of difference between observed cumulative process and estimate under null of constant effect over time.
`sim.test.procBeqC`	list of 50 random realizations of test-processes under null based on resampling.
`covariance`	covariances for nonparametric terms of model.
`B.iid`	Resample processes for nonparametric terms of model.
`gamma.iid`	Resample processes for parametric terms of model.

`loglike` approximate log-likelihood for model, similar to Cox's partial likelihood. Only computed when robust=1.

`D2linv` inverse of the derivative of the score function.

`score` value of score for final estimates.

`test.procProp` observed score process for proportional part of model.

`var.score` variance of score process (optional variation estimator for beta.fixed=1 and robust estimator otherwise).

`pval.Prop` p-value based on resampling.

`sim.supProp` re-sampled absolute supremum values.

`sim.test.procProp` list of 50 random realizations of test-processes for proportionality under the model based on resampling.

Author(s)

Thomas Scheike

References

Martinussen and Scheike, Dynamic Regression Models for Survival Data, Springer (2006).

Examples

```
library(timereg)
data(sTRACE)
# Fits Cox model
out<-cox.aalen(Surv(time,status==9)~prop(age)+prop(sex)+
prop(vf)+prop(chf)+prop(diabetes),data=sTRACE)

# makes Lin, Wei, Ying test for proportionality
summary(out)
par(mfrow=c(2,3))
plot(out,score=1)

# Fits stratified Cox model
out<-cox.aalen(Surv(time,status==9)~-1+factor(vf)+ prop(age)+prop(sex)+
       prop(chf)+prop(diabetes),data=sTRACE,max.time=7,n.sim=100)
summary(out)
par(mfrow=c(1,2)); plot(out);
# Same model, but needs to invert the entire marix for the aalen part: X(t)
out<-cox.aalen(Surv(time,status==9)~factor(vf)+ prop(age)+prop(sex)+
       prop(chf)+prop(diabetes),data=sTRACE,max.time=7,n.sim=100)
summary(out)
par(mfrow=c(1,2)); plot(out);


# Fits Cox-Aalen model
out<-cox.aalen(Surv(time,status==9)~prop(age)+prop(sex)+
```

```
                vf+chf+prop(diabetes),data=sTRACE,max.time=7,n.sim=100)
summary(out)
par(mfrow=c(2,3))
plot(out)
```

`cox.ipw` *Missing data IPW Cox*

Description

Fits an Cox-Aalen survival model with missing data, with glm specification of probability of missingness.

Usage

```
cox.ipw(survformula, glmformula, d = sys.parent(), max.clust = NULL,
  ipw.se = FALSE, tie.seed = 100)
```

Arguments

`survformula`	a formula object with the response on the left of a '~' operator, and the independent terms on the right as regressors. The response must be a survival object as returned by the 'Surv' function. Adds the prop() wrapper internally for using cox.aalen function for fitting Cox model.
`glmformula`	formula for "being" observed, that is not missing.
`d`	data frame.
`max.clust`	number of clusters in iid approximation. Default is all.
`ipw.se`	if TRUE computes standard errors based on iid decompositon of cox and glm model, thus should be asymptotically correct.
`tie.seed`	if there are ties these are broken, and to get same break the seed must be the same. Recommend to break them prior to entering the program.

Details

Taylor expansion of Cox's partial likelihood in direction of glm parameters using num-deriv and iid expansion of Cox and glm paramters (lava).

Value

returns an object of type "cox.aalen". With the following arguments:

`iid`	iid decomposition.
`coef`	missing data estiamtes for weighted cox.
`var`	robust pointwise variances estimates.

se	robust pointwise variances estimates.
se.naive	estimate of parametric components of model.
ties	list of ties and times with random noise to break ties.
cox	output from weighted cox model.

Author(s)

Thomas Scheike

References

Paik et al.

Examples

```
### fit <- cox.ipw(Surv(time,status)~X+Z,obs~Z+X+time+status,data=d,ipw.se=TRUE)
### summary(fit)
```

---

csl *CSL liver chirrosis data*

Description

Survival status for the liver chirrosis patients of Schlichting et al.

Format

This data frame contains the following columns:

id a numeric vector. Id of subject.

time a numeric vector. Time of measurement.

prot a numeric vector. Prothrombin level at measurement time.

dc a numeric vector code. 0: censored observation, 1: died at eventT.

eventT a numeric vector. Time of event (death).

treat a numeric vector code. 0: active treatment of prednisone, 1: placebo treatment.

sex a numeric vector code. 0: female, 1: male.

age a numeric vector. Age of subject at inclusion time subtracted 60.

prot.base a numeric vector. Prothrombin base level before entering the study.

prot.prev a numeric vector. Level of prothrombin at previous measurement time.

lt a numeric vector. Gives the starting time for the time-intervals.

rt a numeric vector. Gives the stopping time for the time-intervals.

Source

P.K. Andersen

References

Schlichting, P., Christensen, E., Andersen, P., Fauerholds, L., Juhl, E., Poulsen, H. and Tygstrup, N. (1983), The Copenhagen Study Group for Liver Diseases, Hepatology 3, 889–895

Examples

```
data(csl)
names(csl)
```

---

cum.residuals *Model validation based on cumulative residuals*

Description

Computes cumulative residuals and approximative p-values based on resampling techniques.

Usage

```
cum.residuals(object, data = sys.parent(), modelmatrix = 0,
  cum.resid = 1, n.sim = 500, weighted.test = 0,
  max.point.func = 50, weights = NULL)
```

Arguments

object	an object of class 'aalen', 'timecox', 'cox.aalen' where the residuals are returned ('residuals=1')
data	data frame based on which residuals are computed.
modelmatrix	specifies a grouping of the data that is used for cumulating residuals. Must have same size as data and be ordered in the same way.
cum.resid	to compute residuals versus each of the continuous covariates in the model.
n.sim	number of simulations in resampling.
weighted.test	to compute a variance weighted version of the test-processes used for testing constant effects of covariates.
max.point.func	limits the amount of computations, only considers a max of 50 points on the covariate scales.
weights	weights for sum of martingale residuals, now for cum.resid=1.

Value

returns an object of type "cum.residuals" with the following arguments:

`cum`	cumulative residuals versus time for the groups specified by modelmatrix.
`var.cum`	the martingale based pointwise variance estimates.
`robvar.cum`	robust pointwise variances estimates of cumulatives.
`obs.testBeq0`	observed absolute value of supremum of cumulative components scaled with the variance.
`pval.testBeq0`	p-value covariate effects based on supremum test.
`sim.testBeq0`	resampled supremum value.
`conf.band`	resampling based constant to construct robust 95% uniform confidence bands for cumulative residuals.
`obs.test`	absolute value of supremum of observed test-process.
`pval.test`	p-value for supremum test statistic.
`sim.test`	resampled absolute value of supremum cumulative residuals.
`proc.cumz`	observed cumulative residuals versus all continuous covariates of model.
`sim.test.proccumz`	list of 50 random realizations of test-processes under model for all continuous covariates.

Author(s)

Thomas Scheike

References

Martinussen and Scheike, Dynamic Regression Models for Survival Data, Springer (2006).

Examples

```
data(sTRACE)
# Fits Aalen model and returns residuals
fit<-aalen(Surv(time,status==9)~age+sex+diabetes+chf+vf,
           data=sTRACE,max.time=7,n.sim=0,residuals=1)

# constructs and simulates cumulative residuals versus age groups
fit.mg<-cum.residuals(fit,data=sTRACE,n.sim=100,
modelmatrix=model.matrix(~-1+factor(cut(age,4)),sTRACE))

par(mfrow=c(1,4))
# cumulative residuals with confidence intervals
plot(fit.mg);
# cumulative residuals versus processes under model
plot(fit.mg,score=1);
summary(fit.mg)
```

```
# cumulative residuals vs. covariates Lin, Wei, Ying style
fit.mg<-cum.residuals(fit,data=sTRACE,cum.resid=1,n.sim=100)

par(mfrow=c(2,4))
plot(fit.mg,score=2)
summary(fit.mg)
```

`diabetes` *The Diabetic Retinopathy Data*

Description

The data was colleceted to test a laser treatment for delaying blindness in patients with dibetic retinopathy. The subset of 197 patiens given in Huster et al. (1989) is used.

Format

This data frame contains the following columns:

id a numeric vector. Patient code.

agedx a numeric vector. Age of patient at diagnosis.

time a numeric vector. Survival time: time to blindness or censoring.

status a numeric vector code. Survival status. 1: blindness, 0: censored.

trteye a numeric vector code. Random eye selected for treatment. 1: left eye 2: right eye.

treat a numeric vector. 1: treatment 0: untreated.

adult a numeric vector code. 1: younger than 20, 2: older than 20.

Source

Huster W.J. and Brookmeyer, R. and Self. S. (1989) MOdelling paired survival data with covariates, Biometrics 45, 145-56.

Examples

```
data(diabetes)
names(diabetes)
```

`dynreg` *Fit time-varying regression model*

Description

Fits time-varying regression model with partly parametric components. Time-dependent variables for longitudinal data. The model assumes that the mean of the observed responses given covariates is a linear time-varying regression model :

Usage

```
dynreg(formula, data = sys.parent(), aalenmod, bandwidth = 0.5,
  id = NULL, bhat = NULL, start.time = 0, max.time = NULL,
  n.sim = 500, meansub = 1, weighted.test = 0, resample = 0)
```

Arguments

`formula`	a formula object with the response on the left of a '~' operator, and the independent terms on the right as regressors.
`data`	a data.frame with the variables.
`aalenmod`	Aalen model for measurement times. Specified as a survival model (see aalen function).
`bandwidth`	bandwidth for local iterations. Default is 50% of the range of the considered observation period.
`id`	For timevarying covariates the variable must associate each record with the id of a subject.
`bhat`	initial value for estimates. If NULL local linear estimate is computed.
`start.time`	start of observation period where estimates are computed.
`max.time`	end of observation period where estimates are computed. Estimates thus computed from [start.time, max.time]. Default is max of data.
`n.sim`	number of simulations in resampling.
`meansub`	if '1' then the mean of the responses is subtracted before the estimation is carried out.
`weighted.test`	to compute a variance weighted version of the test-processes used for testing time-varying effects.
`resample`	returns resample processes.

Details

$$E(Z_{ij}|X_{ij}(t)) = \beta^T(t)X^1_{ij}(t) + \gamma^T X^2_{ij}(t)$$

where Z_{ij} is the j'th measurement at time t for the i'th subject with covariates X^1_{ij} and X^2_{ij}. Resampling is used for computing p-values for tests of timevarying effects.

The data for a subject is presented as multiple rows or 'observations', each of which applies to an interval of observation (start, stop]. For counting process data with the)start,stop] notation is used the 'id' variable is needed to identify the records for each subject. The program assumes that there are no ties, and if such are present random noise is added to break the ties.

Value

returns an object of type "dynreg". With the following arguments:

`cum` the cumulative regression coefficients. This is the efficient estimator based on an initial smoother obtained by local linear regression :

$$\hat{B}(t) = \int_0^t \tilde{\beta}(s)ds +$$

$$\int_0^t X^-(Diag(z) - Diag(X^T(s)\tilde{\beta}(s)))dp(ds \times dz),$$

where $\tilde{\beta}(t)$ is an initial estimate either provided or computed by local linear regression. To plot this estimate use type="eff.smooth" in the plot() command.

`var.cum` the martingale based pointwise variance estimates.

`robvar.cum` robust pointwise variances estimates.

`gamma` estimate of semi-parametric components of model.

`var.gamma` variance for gamma.

`robvar.gamma` robust variance for gamma.

`cum0` simple estimate of cumulative regression coefficients that does not use use an initial smoothing based estimate

$$\hat{B}_0(t) = \int_0^t X^- Diag(z)dp(ds \times dz).$$

To plot this estimate use type="0.mpp" in the plot() command.

`var.cum0` the martingale based pointwise variance estimates of cum0.

`cum.ms` estimate of cumulative regression coefficients based on initial smoother (but robust to this estimator).

$$\hat{B}_{ms}(t) = \int_0^t X^-(Diag(z) - f(s))dp(ds \times dz),$$

where f is chosen as the matrix

$$f(s) = Diag(X^T(s)\tilde{\beta}(s))(I - X_\alpha(s)X_\alpha^-(s)),$$

where X_α is the design for the sampling intensities.
This is also an efficient estimator when the initial estimator is consistent for $\beta(t)$ and then asymptotically equivalent to cum, but small sample properties appear inferior. Its variance is estimated by var.cum.
To plot this estimate use type="ms.mpp" in the plot() command.

`cum.ly` estimator where local averages are subtracted. Special case of cum.ms. To plot this estimate use type="ly.mpp" in plot.

`var.cum.ly` the martingale based pointwise variance estimates.

`gamma0` estimate of parametric component of model.

`var.gamma0` estimate of variance of parametric component of model.

`gamma.ly` estimate of parametric components of model.

`var.gamma.ly` estimate of variance of parametric component of model.

`gamma.ms` estimate of variance of parametric component of model.

`var.gamma.ms` estimate of variance of parametric component of model.

`obs.testBeq0` observed absolute value of supremum of cumulative components scaled with the variance.

`pval.testBeq0` p-value for covariate effects based on supremum test.

`sim.testBeq0` resampled supremum values.

`obs.testBeqC` observed absolute value of supremum of difference between observed cumulative process and estimate under null of constant effect.

`pval.testBeqC` p-value based on resampling.

`sim.testBeqC` resampled supremum values.

`obs.testBeqC.is` observed integrated squared differences between observed cumulative and estimate under null of constant effect.

`pval.testBeqC.is` p-value based on resampling.

`sim.testBeqC.is` resampled supremum values.

`conf.band` resampling based constant to construct robust 95% uniform confidence bands.

`test.procBeqC` observed test-process of difference between observed cumulative process and estimate under null of constant effect.

`sim.test.procBeqC` list of 50 random realizations of test-processes under null based on resampling.

`covariance` covariances for nonparametric terms of model.

Author(s)

Thomas Scheike

References

Martinussen and Scheike, Dynamic Regression Models for Survival Data, Springer (2006).

Examples

```
## this runs slowly and is therfore donttest
data(csl)
indi.m<-rep(1,length(csl$lt))

# Fits time-varying regression model
out<-dynreg(prot~treat+prot.prev+sex+age,data=csl,
Surv(lt,rt,indi.m)~+1,start.time=0,max.time=2,id=csl$id,
n.sim=100,bandwidth=0.7,meansub=0)
summary(out)
par(mfrow=c(2,3))
plot(out)

# Fits time-varying semi-parametric regression model.
outS<-dynreg(prot~treat+const(prot.prev)+const(sex)+const(age),data=csl,
Surv(lt,rt,indi.m)~+1,start.time=0,max.time=2,id=csl$id,
n.sim=100,bandwidth=0.7,meansub=0)
summary(outS)
```

`Event` *Event history object*

Description

Constructur for Event History objects

Usage

```
Event(time, time2 = TRUE, cause = NULL, cens.code = 0, ...)
```

Arguments

`time`	Time
`time2`	Time 2
`cause`	Cause
`cens.code`	Censoring code (default 0)
`...`	Additional arguments

Details

... content for details

Value

Object of class Event (a matrix)

Author(s)

Klaus K. Holst and Thomas Scheike

Examples

```
t1 <- 1:10
t2 <- t1+runif(10)
ca <- rbinom(10,2,0.4)
(x <- Event(t1,t2,ca))
```

---

event.split *EventSplit (SurvSplit).*

Description

contstructs start stop formulation of event time data after a variable in the data.set. Similar to SurvSplit of the survival package but can also split after random time given in data frame.

Usage

```
event.split(data, time = "time", status = "status", cuts = "cuts",
  name.id = "id", name.start = "start", cens.code = 0,
  order.id = TRUE, time.group = TRUE)
```

Arguments

data	data to be split
time	time variable.
status	status variable.
cuts	cuts variable or numeric cut (only one value)
name.id	name of id variable.
name.start	name of start variable in data, start can also be numeric "0"
cens.code	code for the censoring.
order.id	order data after id and start.
time.group	make variable "before"."cut" that keeps track of wether start,stop is before (1) or after cut (0).

Author(s)

Thomas Scheike

Examples

```
set.seed(1)
d <- data.frame(event=round(5*runif(5),2),start=1:5,time=2*1:5,
status=rbinom(5,1,0.5),x=1:5)
d

d0 <- event.split(d,cuts="event",name.start=0)
d0

dd <- event.split(d,cuts="event")
dd
ddd <- event.split(dd,cuts=3.5)
ddd
event.split(ddd,cuts=5.5)

### successive cutting for many values
dd <- d
for  (cuts in seq(2,3,by=0.3)) dd <- event.split(dd,cuts=cuts)
dd
```

`Gprop.odds` *Fit Generalized Semiparametric Proportional 0dds Model*

Description

Fits a semiparametric proportional odds model:

$$logit(1 - S_{X,Z}(t)) = log(X^T A(t)) + \beta^T Z$$

where A(t) is increasing but otherwise unspecified. Model is fitted by maximising the modified partial likelihood. A goodness-of-fit test by considering the score functions is also computed by resampling methods.

Usage

```
Gprop.odds(formula = formula(data), data = sys.parent(), beta = 0,
  Nit = 50, detail = 0, start.time = 0, max.time = NULL,
  id = NULL, n.sim = 500, weighted.test = 0, sym = 0,
  mle.start = 0)
```

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 with the variables.

beta	starting value for relative risk estimates
Nit	number of iterations for Newton-Raphson algorithm.
detail	if 0 no details is printed during iterations, if 1 details are given.
start.time	start of observation period where estimates are computed.
max.time	end of observation period where estimates are computed. Estimates thus computed from [start.time, max.time]. This is very useful to obtain stable estimates, especially for the baseline. Default is max of data.
id	For timevarying covariates the variable must associate each record with the id of a subject.
n.sim	number of simulations in resampling.
weighted.test	to compute a variance weighted version of the test-processes used for testing time-varying effects.
sym	to use symmetrized second derivative in the case of the estimating equation approach (profile=0). This may improve the numerical performance.
mle.start	starting values for relative risk parameters.

Details

An alternative way of writing the model :

$$S_{X,Z}(t)) = \frac{\exp(-\beta^T Z)}{(X^T A(t)) + \exp(-\beta^T Z)}$$

such that β is the log-odds-ratio of dying before time t, and $A(t)$ is the odds-ratio.

The modelling formula uses the standard survival modelling given in the **survival** package.

The data for a subject is presented as multiple rows or "observations", each of which applies to an interval of observation (start, stop]. The program essentially assumes no ties, and if such are present a little random noise is added to break the ties.

Value

returns an object of type 'cox.aalen'. With the following arguments:

cum	cumulative timevarying regression coefficient estimates are computed within the estimation interval.
var.cum	the martingale based pointwise variance estimates.
robvar.cum	robust pointwise variances estimates.
gamma	estimate of proportional odds parameters of model.
var.gamma	variance for gamma.
robvar.gamma	robust variance for gamma.
residuals	list with residuals. Estimated martingale increments (dM) and corresponding time vector (time).
obs.testBeq0	observed absolute value of supremum of cumulative components scaled with the variance.

`pval.testBeq0`
: p-value for covariate effects based on supremum test.

`sim.testBeq0`
: resampled supremum values.

`obs.testBeqC`
: observed absolute value of supremum of difference between observed cumulative process and estimate under null of constant effect.

`pval.testBeqC`
: p-value based on resampling.

`sim.testBeqC`
: resampled supremum values.

`obs.testBeqC.is`
: observed integrated squared differences between observed cumulative and estimate under null of constant effect.

`pval.testBeqC.is`
: p-value based on resampling.

`sim.testBeqC.is`
: resampled supremum values.

`conf.band`
: resampling based constant to construct robust 95% uniform confidence bands.

`test.procBeqC`
: observed test-process of difference between observed cumulative process and estimate under null of constant effect over time.

`loglike`
: modified partial likelihood, pseudo profile likelihood for regression parameters.

`D2linv`
: inverse of the derivative of the score function.

`score`
: value of score for final estimates.

`test.procProp`
: observed score process for proportional odds regression effects.

`pval.Prop`
: p-value based on resampling.

`sim.supProp`
: re-sampled supremum values.

`sim.test.procProp`
: list of 50 random realizations of test-processes for constant proportional odds under the model based on resampling.

Author(s)

Thomas Scheike

References

Scheike, A flexible semiparametric transformation model for survival data, Lifetime Data Anal. (to appear).

Martinussen and Scheike, Dynamic Regression Models for Survival Data, Springer (2006).

Examples

```
data(sTRACE)

### runs slowly and is therefore donttest
```

```
data(sTRACE)
# Fits Proportional odds model with stratified baseline
age.c<-scale(sTRACE$age,scale=FALSE);
out<-Gprop.odds(Surv(time,status==9)~-1+factor(diabetes)+prop(age.c)+prop(chf)+
                prop(sex)+prop(vf),data=sTRACE,max.time=7,n.sim=50)
summary(out)
par(mfrow=c(2,3))
plot(out,sim.ci=2); plot(out,score=1)
```

`invsubdist` *Finds inverse of piecwise linear sub-distribution*

Description

Finds inverse of piecwise linear sub-distribution to be used for simulation of subdistributions

Usage

```
invsubdist(F1, u, entry = NULL, cond = 1, ptrunc = NULL)
```

Arguments

`F1`	matrix with x, and F1(x)
`u`	points for which to compute inverse
`entry`	possible delayed entry points
`cond`	1 indcates that we draw given that this subdistribution is used, so scales mass to 1 to get conditional distribution function
`ptrunc`	possible trunction weigth for delayed entry, if NULL then uses ptrunc=1-F1(entry)

Author(s)

Thomas Scheike

Examples

```
F1 <- cbind(c(0,5,8,10),c(0,0.1,0.3,0.9))
plot(F1,type="l")
u <- runif(100)
Fiu <- invsubdist(F1,u,cond=0)
points(Fiu$time,u,pch="x")

F1cond <- F1
F1cond[,2] <- F1cond[,2]/0.9
plot(F1cond,type="l")
u <- runif(100)
```

```
Ficond <- invsubdist(F1cond,u,cond=0)
points(Ficond$time,u,pch="-")
Fiu <- invsubdist(F1,u,cond=1)
points(Fiu$time,u,pch="x")

entry <- 4
###
F1entry <- subdist(F1,entry)[,2]
ptrunc <- 1-F1entry
###
F1entry5 <- F1
F1entry5[,1] <- F1entry5[,1]-entry
F1entry5[,2] <- (F1entry5[,2]-F1entry)/ptrunc
pos <- F1entry5[,1]>=0
F1entry5 <- rbind(c(0,0),F1entry5[pos,])
###
plot(F1entry5,ylim=c(0,1),type="l")
u <- runif(100)
Fiu <- invsubdist(F1entry5,u,cond=0)
points(Fiu$time,u,pch="-")
###
Fiu2 <- invsubdist(F1,u,cond=0,entry=entry)
points(Fiu2$time-entry,u,pch="x")
sum(Fiu2$time-entry-Fiu$time)

F1ce <- F1entry5
F1ce[,2] <- F1ce[,2]/tail(F1entry5[,2],1)
plot(F1ce,type="l")
u <- runif(100)
Fi1ce <- invsubdist(F1ce,u,cond=0)
points(Fi1ce$time,u,pch="-")
Fice <- invsubdist(F1,u,cond=1,entry=entry)
points(Fice$time-entry,u,pch="x")
sum(Fice$time-entry-Fi1ce$time)


## simulation of distribution with delayed entry starting at 3
par(mfrow=c(1,1))
F1 <- cbind(c(0,5,8,10),c(0,0.5,0.6,0.9))
F1
plot(F1,ylim=c(0,1),type="l")

n <- 100000
entry <- c(rep(3,10000),runif(n)*7+3)
###entry <- rep(3,n)
u <- runif(n+10000)
###
Fiu <- invsubdist(F1,u,cond=0,entry=entry)
###
# library(prodlim)
# pp <- prodlim(Hist(time,status,entry=entry)~+1,data=Fiu)
# plot(pp,xlim=c(3,10))
###
```

```
entry <- 3
###
F1entry <- subdist(F1,entry)[,2]
ptrunc <- 1-F1entry
###
F1entry5 <- F1
F1entry5[,1] <- F1entry5[,1]-entry
F1entry5[,2] <- (F1entry5[,2]-F1entry)/ptrunc
pos <- F1entry5[,1]>=0
F1entry5 <- rbind(c(0,0),F1entry5[pos,])
#
# lines(entry+F1entry5[,1],1-F1entry5[,2],col=2)

#################################################################
## Simulations of two cumulative incidence functions with truncation
#################################################################
par(mfrow=c(1,1))
F1 <- cbind(c(0,5,8,10),c(0,0.5,0.6,0.9)*0.3)
F2 <- cbind(c(0,5,8,10),c(0,0.5,0.6,0.9)*0.5)
plot(F1,ylim=c(0,1),type="l")
lines(F2,col=2)
entry1 <- 3
###
F1entry <- subdist(F1,entry1)[,2]
F2entry <- subdist(F2,entry1)[,2]
ptrunc <- 1-F1entry-F2entry
###
F1e <- F1
F1e[,1] <- F1e[,1]-entry1
F1e[,2] <- (F1e[,2]-F1entry)/ptrunc
pos <- F1e[,1]>=0
F1e <- rbind(c(0,0),F1e[pos,])
F2e <- F2
F2e[,1] <- F2e[,1]-entry1
F2e[,2] <- (F2e[,2]-F2entry)/ptrunc
pos <- F2e[,1]>=0
F2e <- rbind(c(0,0),F2e[pos,])
#
# truncated identifiable version
lines(entry1+F1e[,1],F1e[,2],col=1)
lines(entry1+F2e[,1],F2e[,2],col=2)

n <- 10000
entry <- c(rep(entry1,10000),runif(n)*(10-entry1)+entry1)
u <- runif(n+10000)
###
F1entry <- subdist(F1,entry)[,2]
F2entry <- subdist(F2,entry)[,2]
ptrunc <- 1-( F1entry+F2entry)
Fiu1 <- invsubdist(F1,u,cond=1,entry=entry,ptrunc=ptrunc)
Fiu2 <- invsubdist(F1,u,cond=1,entry=entry,ptrunc=ptrunc)
###
ptot <- (tail(F1[,2],1)+tail(F2[,2],1)-F1entry-F2entry)/(ptrunc)
```

```
rt <- rbinom(n+10000,1,ptot)
p1 <- ((tail(F1[,2],1)-F1entry)/ptrunc)
p2 <- ((tail(F2[,2],1)-F2entry)/ptrunc)
rb <- rbinom(n+10000,1,p1/ptot)
cause=ifelse(rb==1,1,2)
time=ifelse(cause==1,Fiu1$time,Fiu2$time)
cause <- rt*cause
time[cause==0] <- 10
### simulated data, now checking that things are working
# pp <- prodlim(Hist(time,cause,entry=entry)~+1)
# plot(pp,xlim=c(entry1,10),cause=1)
# plot(pp,xlim=c(entry1,10),cause=2,add=TRUE)
###
# lines(entry1+F1e[,1],F1e[,2],col=2)
# lines(entry1+F2e[,2],F2e[,2],col=2)
```

`krylow.pls` *Fits Krylow based PLS for additive hazards model*

Description

Fits the PLS estimator for the additive risk model based on the least squares fitting criterion

Usage

```
krylow.pls(D, d, dim = 1)
```

Arguments

`D`	defined above
`d`	defined above
`dim`	number of pls dimensions

Details

$$L(\beta, D, d) = \beta^T D\beta - 2\beta^T d$$

where $D = \int ZHZdt$ and $d = \int ZHdN$.

Value

returns a list with the following arguments:

`beta`	PLS regression coefficients

Author(s)

Thomas Scheike

References

Martinussen and Scheike, The Aalen additive hazards model with high-dimensional regressors, submitted.

Martinussen and Scheike, Dynamic Regression Models for Survival Data, Springer (2006).

Examples

```
## makes data for pbc complete case
data(mypbc)
pbc<-mypbc
pbc$time<-pbc$time+runif(418)*0.1; pbc$time<-pbc$time/365
pbc<-subset(pbc,complete.cases(pbc));
covs<-as.matrix(pbc[,-c(1:3,6)])
covs<-cbind(covs[,c(1:6,16)],log(covs[,7:15]))

## computes the matrices needed for the least squares
## criterion
out<-aalen(Surv(time,status>=1)~const(covs),pbc,robust=0,n.sim=0)
S=out$intZHZ; s=out$intZHdN;

out<-krylow.pls(S,s,dim=2)
```

---

mela.pop *Melanoma data and Danish population mortality by age and sex*

Description

Melanoma data with background mortality of Danish population.

Format

This data frame contains the following columns:

id a numeric vector. Gives patient id.

sex a numeric vector. Gives sex of patient.

start a numeric vector. Gives the starting time for the time-interval for which the covariate rate is representative.

stop a numeric vector. Gives the stopping time for the time-interval for which the covariate rate is representative.

status a numeric vector code. Survival status. 1: dead from melanoma, 0: alive or dead from other cause.

age a numeric vector. Gives the age of the patient at removal of tumor.

rate a numeric vector. Gives the population mortality for the given sex and age. Based on Table A.2 in Andersen et al. (1993).

Source

Andersen, P.K., Borgan O, Gill R.D., Keiding N. (1993), *Statistical Models Based on Counting Processes*, Springer-Verlag.

Examples

```
data(mela.pop)
names(mela.pop)
```

`melanoma` *The Melanoma Survival Data*

Description

The melanoma data frame has 205 rows and 7 columns. It contains data relating to survival of patients after operation for malignant melanoma collected at Odense University Hospital by K.T. Drzewiecki.

Format

This data frame contains the following columns:

no a numeric vector. Patient code.

status a numeric vector code. Survival status. 1: dead from melanoma, 2: alive, 3: dead from other cause.

days a numeric vector. Survival time.

ulc a numeric vector code. Ulceration, 1: present, 0: absent.

thick a numeric vector. Tumour thickness (1/100 mm).

sex a numeric vector code. 0: female, 1: male.

Source

Andersen, P.K., Borgan O, Gill R.D., Keiding N. (1993), *Statistical Models Based on Counting Processes*, Springer-Verlag.

Drzewiecki, K.T., Ladefoged, C., and Christensen, H.E. (1980), Biopsy and prognosis for cutaneous malignant melanoma in clinical stage I. Scand. J. Plast. Reconstru. Surg. 14, 141-144.

Examples

```
data(melanoma)
names(melanoma)
```

mypbc *my version of the PBC data of the survival package*

Description

my version of the PBC data of the survival package

Source

survival package

pava.pred *Make predictions of predict functions in rows mononotone*

Description

Make predictions of predict functions in rows mononotone using the pool-adjacent-violators-algorithm

Usage

```
pava.pred(pred, increasing = TRUE)
```

Arguments

pred	predictions, either vector or rows of predictions.
increasing	increasing or decreasing.

Value

mononotone predictions.

Author(s)

Thomas Scheike

Examples

```
data(bmt);

## competing risks
add<-comp.risk(Event(time,cause)~platelet+age+tcell,data=bmt,cause=1)
ndata<-data.frame(platelet=c(1,0,0),age=c(0,1,0),tcell=c(0,0,1))
out<-predict(add,newdata=ndata,uniform=0)

par(mfrow=c(1,1))
head(out$P1)
```

```
matplot(out$time,t(out$P1),type="s")

###P1m <- t(apply(out$P1,1,pava))
P1monotone <- pava.pred(out$P1)
head(P1monotone)
matlines(out$time,t(P1monotone),type="s")
```

`pe.sasieni` *Fits Proportional excess hazards model with fixed offsets*

Description

Fits proportional excess hazards model. The Sasieni proportional excess risk model.

Usage

```
pe.sasieni(formula = formula(data), data = sys.parent(), id = NULL,
  start.time = 0, max.time = NULL, offsets = 0, Nit = 50,
  detail = 0, n.sim = 500)
```

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 with the variables.
`id`	gives the number of individuals.
`start.time`	starting time for considered time-period.
`max.time`	stopping considered time-period if different from 0. Estimates thus computed from [0,max.time] if max.time>0. Default is max of data.
`offsets`	fixed offsets giving the mortality.
`Nit`	number of itterations.
`detail`	if detail is one, prints iteration details.
`n.sim`	number of simulations, 0 for no simulations.

Details

The models are written using the survival modelling given in the survival package.

The program assumes that there are no ties, and if such are present random noise is added to break the ties.

Value

Returns an object of type "pe.sasieni". With the following arguments:

`cum`	baseline of Cox model excess risk.
`var.cum`	pointwise variance estimates for estimated cumulatives.
`gamma`	estimate of relative risk terms of model.
`var.gamma`	variance estimates for gamma.
`Ut`	score process for Cox part of model.
`D2linv`	The inverse of the second derivative.
`score`	final score
`test.Prop`	re-sampled absolute supremum values.
`pval.Prop`	p-value based on resampling.

Author(s)

Thomas Scheike

References

Martinussen and Scheike, Dynamic Regression Models for Survival Data, Springer Verlag (2006).

Sasieni, P.D., Proportional excess hazards, Biometrika (1996), 127–41.

Cortese, G. and Scheike, T.H., Dynamic regression hazards models for relative survival (2007), submitted.

Examples

```
data(mela.pop)
out<-pe.sasieni(Surv(start,stop,status==1)~age+sex,mela.pop,
id=1:205,Nit=10,max.time=7,offsets=mela.pop$rate,detail=0,n.sim=100)
summary(out)

ul<-out$cum[,2]+1.96*out$var.cum[,2]^.5
ll<-out$cum[,2]-1.96*out$var.cum[,2]^.5
plot(out$cum,type="s",ylim=range(ul,ll))
lines(out$cum[,1],ul,type="s"); lines(out$cum[,1],ll,type="s")
# see also prop.excess function
```

`plot.aalen` *Plots estimates and test-processes*

Description

This function plots the non-parametric cumulative estimates for the additive risk model or the test-processes for the hypothesis of time-varying effects with re-sampled processes under the null.

Usage

```
## S3 method for class 'aalen'
plot(x, pointwise.ci = 1, hw.ci = 0, sim.ci = 0,
  robust.ci = 0, col = NULL, specific.comps = FALSE, level = 0.05,
  start.time = 0, stop.time = 0, add.to.plot = FALSE, mains = TRUE,
  xlab = "Time", ylab = "Cumulative coefficients", score = FALSE,
  ...)
```

Arguments

`x`	the output from the "aalen" function.
`pointwise.ci`	if >1 pointwise confidence intervals are plotted with lty=pointwise.ci
`hw.ci`	if >1 Hall-Wellner confidence bands are plotted with lty=hw.ci. Only 0.95 % bands can be constructed.
`sim.ci`	if >1 simulation based confidence bands are plotted with lty=sim.ci. These confidence bands are robust to non-martingale behaviour.
`robust.ci`	robust standard errors are used to estimate standard error of estimate, otherwise martingale based standard errors are used.
`col`	specifice colors of different components of plot, in order: c(estimate,pointwise.ci,robust.ci,hw.ci,sim.ci) so for example, when we ask to get pointwise.ci, hw.ci and sim.ci we would say c(1,2,3,4) to use colors as specified.
`specific.comps`	all components of the model is plotted by default, but a list of components may be specified, for example first and third "c(1,3)".
`level`	gives the significance level.
`start.time`	start of observation period where estimates are plotted.
`stop.time`	end of period where estimates are plotted. Estimates thus plotted from [start.time, max.time].
`add.to.plot`	to add to an already existing plot.
`mains`	add names of covariates as titles to plots.
`xlab`	label for x-axis.
`ylab`	label for y-axis.
`score`	to plot test processes for test of time-varying effects along with 50 random realization under the null-hypothesis.
`...`	unused arguments - for S3 compatibility

Author(s)

Thomas Scheike

References

Martinussen and Scheike, Dynamic Regression models for Survival Data, Springer (2006).

Examples

```
# see help(aalen)
data(sTRACE)
out<-aalen(Surv(time,status==9)~chf+vf,sTRACE,max.time=7,n.sim=100)
par(mfrow=c(2,2))
plot(out,pointwise.ci=1,hw.ci=1,sim.ci=1,col=c(1,2,3,4))
par(mfrow=c(2,2))
plot(out,pointwise.ci=0,robust.ci=1,hw.ci=1,sim.ci=1,col=c(1,2,3,4))
```

---

plot.cum.residuals *Plots cumulative residuals*

Description

This function plots the output from the cumulative residuals function "cum.residuals". The cumulative residuals are compared with the performance of similar processes under the model.

Usage

```
## S3 method for class 'cum.residuals'
plot(x, pointwise.ci = 1, hw.ci = 0,
  sim.ci = 0, robust = 1, specific.comps = FALSE, level = 0.05,
  start.time = 0, stop.time = 0, add.to.plot = FALSE, mains = TRUE,
  main = NULL, xlab = NULL, ylab = "Cumulative MG-residuals",
  ylim = NULL, score = 0, conf.band = FALSE, ...)
```

Arguments

x	the output from the "cum.residuals" function.
pointwise.ci	if >1 pointwise confidence intervals are plotted with lty=pointwise.ci
hw.ci	if >1 Hall-Wellner confidence bands are plotted with lty=hw.ci. Only 95% bands can be constructed.
sim.ci	if >1 simulation based confidence bands are plotted with lty=sim.ci. These confidence bands are robust to non-martingale behaviour.
robust	if "1" robust standard errors are used to estimate standard error of estimate, otherwise martingale based estimate are used.

`specific.comps`
: all components of the model is plotted by default, but a list of components may be specified, for example first and third "c(1,3)".

`level`
: gives the significance level. Default is 0.05.

`start.time`
: start of observation period where estimates are plotted. Default is 0.

`stop.time`
: end of period where estimates are plotted. Estimates thus plotted from [start.time, max.time].

`add.to.plot`
: to add to an already existing plot. Default is "FALSE".

`mains`
: add names of covariates as titles to plots.

`main`
: vector of names for titles in plots.

`xlab`
: label for x-axis. NULL is default which leads to "Time" or "". Can also give a character vector.

`ylab`
: label for y-axis. Default is "Cumulative MG-residuals".

`ylim`
: limits for y-axis.

`score`
: if '0' plots related to modelmatrix are specified, thus resulting in grouped residuals, if '1' plots for modelmatrix but with random realizations under model, if '2' plots residuals versus continuous covariates of model with random realizations under the model.

`conf.band`
: makes simulation based confidence bands for the test processes under the 0 based on variance of these processes limits for y-axis. These will give additional information of whether the observed cumulative residuals are extreme or not when based on a variance weighted test.

`...`
: unused arguments - for S3 compatibility

Author(s)

Thomas Scheike

References

Martinussen and Scheike, Dynamic Regression Models for Survival Data, Springer (2006).

Examples

```
# see cum.residuals for examples
```

`plot.dynreg` *Plots estimates and test-processes*

Description

This function plots the non-parametric cumulative estimates for the additive risk model or the test-processes for the hypothesis of constant effects with re-sampled processes under the null.

Usage

```
## S3 method for class 'dynreg'
plot(x, type = "eff.smooth", pointwise.ci = 1,
  hw.ci = 0, sim.ci = 0, robust = 0, specific.comps = FALSE,
  level = 0.05, start.time = 0, stop.time = 0, add.to.plot = FALSE,
  mains = TRUE, xlab = "Time", ylab = "Cumulative coefficients",
  score = FALSE, ...)
```

Arguments

`x`	the output from the "dynreg" function.
`type`	the estimator plotted. Choices "eff.smooth", "ms.mpp", "0.mpp" and "ly.mpp". See the dynreg function for more on this.
`pointwise.ci`	if >1 pointwise confidence intervals are plotted with lty=pointwise.ci
`hw.ci`	if >1 Hall-Wellner confidence bands are plotted with lty=hw.ci. Only 0.95 % bands can be constructed.
`sim.ci`	if >1 simulation based confidence bands are plotted with lty=sim.ci. These confidence bands are robust to non-martingale behaviour.
`robust`	robust standard errors are used to estimate standard error of estimate, otherwise martingale based estimate are used.
`specific.comps`	all components of the model is plotted by default, but a list of components may be specified, for example first and third "c(1,3)".
`level`	gives the significance level.
`start.time`	start of observation period where estimates are plotted.
`stop.time`	end of period where estimates are plotted. Estimates thus plotted from [start.time, max.time].
`add.to.plot`	to add to an already existing plot.
`mains`	add names of covariates as titles to plots.
`xlab`	label for x-axis.
`ylab`	label for y-axis.
`score`	to plot test processes for test of time-varying effects along with 50 random realization under the null-hypothesis.
`...`	unused arguments - for S3 compatibility

Author(s)

Thomas Scheike

References

Martinussen and Scheike, Dynamic Regression Models for Survival Data, Springer (2006).

Examples

```
### runs slowly and therefore donttest
data(csl)
indi.m<-rep(1,length(csl$lt))

# Fits time-varying regression model
out<-dynreg(prot~treat+prot.prev+sex+age,csl,
Surv(lt,rt,indi.m)~+1,start.time=0,max.time=3,id=csl$id,
n.sim=100,bandwidth=0.7,meansub=0)

par(mfrow=c(2,3))
# plots estimates
plot(out)
# plots tests-processes for time-varying effects
plot(out,score=TRUE)
```

---

predict.timereg *Predictions for Survival and Competings Risks Regression for timereg*

Description

Make predictions based on the survival models (Aalen and Cox-Aalen) and the competing risks models for the cumulative incidence function (comp.risk). Computes confidence intervals and confidence bands based on resampling.

Usage

```
## S3 method for class 'timereg'
predict(object, newdata = NULL, X = NULL,
  times = NULL, Z = NULL, n.sim = 500, uniform = TRUE, se = TRUE,
  alpha = 0.05, resample.iid = 0, ...)
```

Arguments

object	an object belonging to one of the following classes: comprisk, aalen or cox.aalen
newdata	specifies the data at which the predictions are wanted.
X	alternative to newdata, specifies the nonparametric components for predictions.
times	times in which predictions are computed, default is all time-points for baseline
Z	alternative to newdata, specifies the parametric components of the model for predictions.
n.sim	number of simulations in resampling.
uniform	computes resampling based uniform confidence bands.
se	computes pointwise standard errors
alpha	specificies the significance levelwhich cause we consider.
resample.iid	set to 1 to return iid decomposition of estimates, 3-dim matrix (predictions x times x subjects)
...	unused arguments - for S3 compatability

Value

time	vector of time points where the predictions are computed.
unif.band	resampling based constant to construct 95% uniform confidence bands.
model	specifies what model that was fitted.
alpha	specifies the significance level for the confidence intervals. This relates directly to the constant given in unif.band.
newdata	specifies the newdata given in the call.
RR	gives relative risk terms for Cox-type models.
call	gives call for predict funtion.
initial.call	gives call for underlying object used for predictions.
P1	gives cumulative inicidence predictions for competing risks models. Predictions given in matrix form with different subjects in different rows.
S0	gives survival predictions for survival models. Predictions given in matrix form with different subjects in different rows.
se.P1	pointwise standard errors for predictions of P1.
se.S0	pointwise standard errors for predictions of S0.

Author(s)

Thomas Scheike, Jeremy Silver

References

Scheike, Zhang and Gerds (2008), Predicting cumulative incidence probability by direct binomial regression, Biometrika, 95, 205-220.

Scheike and Zhang (2007), Flexible competing risks regression modelling and goodness of fit, LIDA, 14, 464-483 .

Martinussen and Scheike (2006), Dynamic regression models for survival data, Springer.

Examples

```
data(bmt);

## competing risks
add<-comp.risk(Event(time,cause)~platelet+age+tcell,data=bmt,cause=1)

ndata<-data.frame(platelet=c(1,0,0),age=c(0,1,0),tcell=c(0,0,1))
out<-predict(add,newdata=ndata,uniform=1,n.sim=1000)
par(mfrow=c(2,2))
plot(out,multiple=0,uniform=1,col=1:3,lty=1,se=1)
# see comp.risk for further examples.

add<-comp.risk(Event(time,cause)~factor(tcell),data=bmt,cause=1)
summary(add)
out<-predict(add,newdata=ndata,uniform=1,n.sim=1000)
plot(out,multiple=1,uniform=1,col=1:3,lty=1,se=1)

add<-prop.odds.subdist(Event(time,cause)~factor(tcell),
       data=bmt,cause=1)
out <- predict(add,X=1,Z=1)
plot(out,multiple=1,uniform=1,col=1:3,lty=1,se=1)


## SURVIVAL predictions aalen function
data(sTRACE)
out<-aalen(Surv(time,status==9)~sex+ diabetes+chf+vf,
data=sTRACE,max.time=7,n.sim=0,resample.iid=1)

pout<-predict(out,X=rbind(c(1,0,0,0,0),rep(1,5)))
head(pout$S0[,1:5]); head(pout$se.S0[,1:5])
par(mfrow=c(2,2))
plot(pout,multiple=1,se=0,uniform=0,col=1:2,lty=1:2)
plot(pout,multiple=0,se=1,uniform=1,col=1:2)

out<-aalen(Surv(time,status==9)~const(age)+const(sex)+
const(diabetes)+chf+vf,
data=sTRACE,max.time=7,n.sim=0,resample.iid=1)

pout<-predict(out,X=rbind(c(1,0,0),c(1,1,0)),
Z=rbind(c(55,0,1),c(60,1,1)))
head(pout$S0[,1:5]); head(pout$se.S0[,1:5])
par(mfrow=c(2,2))
plot(pout,multiple=1,se=0,uniform=0,col=1:2,lty=1:2)
plot(pout,multiple=0,se=1,uniform=1,col=1:2)

pout<-predict(out,uniform=0,se=0,newdata=sTRACE[1:10,])
plot(pout,multiple=1,se=0,uniform=0)

#### cox.aalen
out<-cox.aalen(Surv(time,status==9)~prop(age)+prop(sex)+
prop(diabetes)+chf+vf,
```

```
data=sTRACE,max.time=7,n.sim=0,resample.iid=1)

pout<-predict(out,X=rbind(c(1,0,0),c(1,1,0)),Z=rbind(c(55,0,1),c(60,1,1)))
head(pout$S0[,1:5]); head(pout$se.S0[,1:5])
par(mfrow=c(2,2))
plot(pout,multiple=1,se=0,uniform=0,col=1:2,lty=1:2)
plot(pout,multiple=0,se=1,uniform=1,col=1:2)

pout<-predict(out,uniform=0,se=0,newdata=sTRACE[1:10,])
plot(pout,multiple=1,se=0,uniform=0)

#### prop.odds model
add<-prop.odds(Event(time,cause!=0)~factor(tcell),data=bmt)
out <- predict(add,X=1,Z=0)
plot(out,multiple=1,uniform=1,col=1:3,lty=1,se=1)
```

`prep.comp.risk` *Set up weights for delayed-entry competing risks data for comp.risk function*

Description

Computes the weights of Geskus (2011) modified to the setting of the comp.risk function. The returned weights are $1/(H(T_i) * G_c(min(T_i, tau)))$ and tau is the max of the times argument, here H is the estimator of the truncation distribution and G_c is the right censoring distribution.

Usage

```
prep.comp.risk(data, times = NULL, entrytime = NULL, time = "time",
  cause = "cause", cname = "cweight", tname = "tweight",
  strata = NULL, nocens.out = TRUE, cens.formula = NULL,
  cens.code = 0, prec.factor = 100, trunc.mintau = FALSE)
```

Arguments

`data`	data frame for comp.risk.
`times`	times for estimating equations.
`entrytime`	name of delayed entry variable, if not given computes right-censoring case.
`time`	name of survival time variable.
`cause`	name of cause indicator
`cname`	name of censoring weight.
`tname`	name of truncation weight.
`strata`	strata variable to obtain stratified weights.
`nocens.out`	returns only uncensored part of data-frame
`cens.formula`	censoring model formula for Cox models for the truncation and censoring model.

`cens.code` code for censoring among causes.

`prec.factor` precision factor, for ties between censoring/even times, truncation times/event times

`trunc.mintau` specicies wether the truncation distribution is evaluated in death times or death times minimum max(times), FALSE makes the estimator equivalent to Kaplan-Meier (in the no covariate case).

Value

Returns an object. With the following arguments:

`dataw` a data.frame with weights.

The function wants to make two new variables "weights" and "cw" so if these already are in the data frame it tries to add an "_" in the names.

Author(s)

Thomas Scheike

References

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

Shen (2011), Proportional subdistribution hazards regression for left-truncated competing risks data, Journal of Nonparametric Statistics (2011), 23, 885-895

Examples

```
data(bmt)
nn <- nrow(bmt)
entrytime <- rbinom(nn,1,0.5)*(bmt$time*runif(nn))
bmt$entrytime <- entrytime
times <- seq(5,70,by=1)

### adds weights to uncensored observations
bmtw <- prep.comp.risk(bmt,times=times,time="time",
       entrytime="entrytime",cause="cause")

########################################
### nonparametric estimates
########################################
## {{{
### nonparametric estimates, right-censoring only
out <- comp.risk(Event(time,cause)~+1,data=bmt,
 cause=1,model="rcif2",
 times=c(5,30,70),n.sim=0)
out$cum
### same as
###out <- prodlim(Hist(time,cause)~+1,data=bmt)
```

```
###summary(out,cause="1",times=c(5,30,70))

### with truncation
out <- comp.risk(Event(time,cause)~+1,data=bmtw,cause=1,
  model="rcif2",
  cens.weight=bmtw$cw,weights=bmtw$weights,times=c(5,30,70),
  n.sim=0)
out$cum
### same as
###out <- prodlim(Hist(entry=entrytime,time,cause)~+1,data=bmt)
###summary(out,cause="1",times=c(5,30,70))
## }}}

########################################
### Regression
########################################
## {{{
### with truncation correction
out <- comp.risk(Event(time,cause)~const(tcell)+const(platelet),data=bmtw,
 cause=1,cens.weight=bmtw$cw,
 weights=bmtw$weights,times=times,n.sim=0)
summary(out)

### with only righ-censoring, standard call
outn <- comp.risk(Event(time,cause)~const(tcell)+const(platelet),data=bmt,
  cause=1,times=times,n.sim=0)
summary(outn)
## }}}
```

`print.aalen` *Prints call*

Description

Prints call for object. Lists nonparametric and parametric terms of model

Usage

```
## S3 method for class 'aalen'
print(x, ...)
```

Arguments

`x`	an aalen object
`...`	unused arguments - for S3 compatibility

Author(s)

Thomas Scheike

prop — *Identifies the multiplicative terms in Cox-Aalen model and proportional excess risk model*

Description

Specifies which of the regressors that belong to the multiplicative part of the Cox-Aalen model

Usage

```
prop(x)
```

Arguments

x	variable

Details

$$\lambda_i(t) = Y_i(t)(X_i^T(t)\alpha(t)) \exp(Z_i^T(t)\beta)$$

for this model prop specified the covariates to be included in $Z_i(t)$

Author(s)

Thomas Scheike

prop.excess — *Fits Proportional excess hazards model*

Description

Fits proportional excess hazards model.

Usage

```
prop.excess(formula = formula(data), data = sys.parent(), excess = 1,
  tol = 1e-04, max.time = NULL, n.sim = 1000, alpha = 1,
  frac = 1)
```

Arguments

formula	a formula object, with the response on the left of a '~' operator, and the terms on the right. The response must be a survival object as returned by the 'Surv' function.
data	a data.frame with the variables.
excess	specifies for which of the subjects the excess term is present. Default is that the term is present for all subjects.
tol	tolerance for numerical procedure.
max.time	stopping considered time-period if different from 0. Estimates thus computed from [0,max.time] if max.time>0. Default is max of data.
n.sim	number of simulations in re-sampling.
alpha	tuning paramter in Newton-Raphson procedure. Value smaller than one may give more stable convergence.
frac	number between 0 and 1. Is used in supremum test where observed jump times t1, ..., tk is replaced by t1, ..., tl with l=round(frac*k).

Details

The models are written using the survival modelling given in the survival package.

The program assumes that there are no ties, and if such are present random noise is added to break the ties.

Value

Returns an object of type "prop.excess". With the following arguments:

cum	estimated cumulative regression functions. First column contains the jump times, then follows the estimated components of additive part of model and finally the excess cumulative baseline.
var.cum	robust pointwise variance estimates for estimated cumulatives.
gamma	estimate of parametric components of model.
var.gamma	robust variance estimate for gamma.
pval	p-value of Kolmogorov-Smirnov test (variance weighted) for excess baseline and Aalen terms, H: B(t)=0.
pval.HW	p-value of supremum test (corresponding to Hall-Wellner band) for excess baseline and Aalen terms, H: B(t)=0. Reported in summary.
pval.CM	p-value of Cramer von Mises test for excess baseline and Aalen terms, H: B(t)=0.
quant	95 percent quantile in distribution of resampled Kolmogorov-Smirnov test statistics for excess baseline and Aalen terms. Used to construct 95 percent simulation band.
quant95HW	95 percent quantile in distribution of resampled supremum test statistics corresponding to Hall-Wellner band for excess baseline and Aalen terms. Used to construct 95 percent Hall-Wellner band.
simScoreProp	observed scoreprocess and 50 resampled scoreprocesses (under model). List with 51 elements.

Author(s)

Torben Martinussen

References

Martinussen and Scheike, Dynamic Regression Models for Survival Data, Springer Verlag (2006).

Examples

```
###working on memory leak issue, 3/3-2015
###data(melanoma)
###lt<-log(melanoma$thick)          # log-thickness
###excess<-(melanoma$thick>=210)    # excess risk for thick tumors
###
#### Fits Proportional Excess hazards model
###fit<-prop.excess(Surv(days/365,status==1)~sex+ulc+cox(sex)+
###                 cox(ulc)+cox(lt),melanoma,excess=excess,n.sim=100)
###summary(fit)
###par(mfrow=c(2,3))
###plot(fit)
```

---

prop.odds *Fit Semiparametric Proportional 0dds Model*

Description

Fits a semiparametric proportional odds model:

$$logit(1 - S_Z(t)) = log(G(t)) + \beta^T Z$$

where G(t) is increasing but otherwise unspecified. Model is fitted by maximising the modified partial likelihood. A goodness-of-fit test by considering the score functions is also computed by resampling methods.

Usage

```
prop.odds(formula, data = sys.parent(), beta = NULL, Nit = 20,
  detail = 0, start.time = 0, max.time = NULL, id = NULL,
  n.sim = 500, weighted.test = 0, profile = 1, sym = 0,
  baselinevar = 1, clusters = NULL, max.clust = 1000,
  weights = NULL)
```

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 Event object as returned by the 'Event' function.
data	a data.frame with the variables.
beta	starting value for relative risk estimates
Nit	number of iterations for Newton-Raphson algorithm.
detail	if 0 no details is printed during iterations, if 1 details are given.
start.time	start of observation period where estimates are computed.
max.time	end of observation period where estimates are computed. Estimates thus computed from [start.time, max.time]. This is very useful to obtain stable estimates, especially for the baseline. Default is max of data.
id	For timevarying covariates the variable must associate each record with the id of a subject.
n.sim	number of simulations in resampling.
weighted.test	to compute a variance weighted version of the test-processes used for testing time-varying effects.
profile	if profile is 1 then modified partial likelihood is used, profile=0 fits by simple estimating equation. The modified partial likelihood is recommended.
sym	to use symmetrized second derivative in the case of the estimating equation approach (profile=0). This may improve the numerical performance.
baselinevar	set to 0 to omit calculations of baseline variance.
clusters	to compute cluster based standard errors.
max.clust	number of maximum clusters to be used, to save time in iid decomposition.
weights	weights for score equations.

Details

The modelling formula uses the standard survival modelling given in the **survival** package.

For large data sets use the divide.conquer.timereg of the mets package to run the model on splits of the data, or the alternative estimator by the cox.aalen function.

The data for a subject is presented as multiple rows or "observations", each of which applies to an interval of observation (start, stop]. The program essentially assumes no ties, and if such are present a little random noise is added to break the ties.

Value

returns an object of type 'cox.aalen'. With the following arguments:

cum	cumulative timevarying regression coefficient estimates are computed within the estimation interval.
var.cum	the martingale based pointwise variance estimates.

`robvar.cum` robust pointwise variances estimates.

`gamma` estimate of proportional odds parameters of model.

`var.gamma` variance for gamma.

`robvar.gamma` robust variance for gamma.

`residuals` list with residuals. Estimated martingale increments (dM) and corresponding time vector (time).

`obs.testBeq0` observed absolute value of supremum of cumulative components scaled with the variance.

`pval.testBeq0` p-value for covariate effects based on supremum test.

`sim.testBeq0` resampled supremum values.

`obs.testBeqC` observed absolute value of supremum of difference between observed cumulative process and estimate under null of constant effect.

`pval.testBeqC` p-value based on resampling.

`sim.testBeqC` resampled supremum values.

`obs.testBeqC.is` observed integrated squared differences between observed cumulative and estimate under null of constant effect.

`pval.testBeqC.is` p-value based on resampling.

`sim.testBeqC.is` resampled supremum values.

`conf.band` resampling based constant to construct robust 95% uniform confidence bands.

`test.procBeqC` observed test-process of difference between observed cumulative process and estimate under null of constant effect over time.

`loglike` modified partial likelihood, pseudo profile likelihood for regression parameters.

`D2linv` inverse of the derivative of the score function.

`score` value of score for final estimates.

`test.procProp` observed score process for proportional odds regression effects.

`pval.Prop` p-value based on resampling.

`sim.supProp` re-sampled supremum values.

`sim.test.procProp` list of 50 random realizations of test-processes for constant proportional odds under the model based on resampling.

Author(s)

Thomas Scheike

References

Martinussen and Scheike, Dynamic Regression Models for Survival Data, Springer (2006).

Examples

```
data(sTRACE)
# Fits Proportional odds model
out<-prop.odds(Event(time,status==9)~age+diabetes+chf+vf+sex,
sTRACE,max.time=7,n.sim=100)
summary(out)

par(mfrow=c(2,3))
plot(out,sim.ci=2)
plot(out,score=1)

pout <- predict(out,Z=c(70,0,0,0,0))
plot(pout)

### alternative estimator for large data sets
form <- Surv(time,status==9)~age+diabetes+chf+vf+sex
pform <- timereg.formula(form)
out2<-cox.aalen(pform,data=sTRACE,max.time=7,
propodds=1,n.sim=0,robust=0,detail=0,Nit=40)
summary(out2)
```

`prop.odds.subdist` *Fit Semiparametric Proportional 0dds Model for the competing risks subdistribution*

Description

Fits a semiparametric proportional odds model:

$$logit(F_1(t;X,Z)) = log(A(t)) + \beta^T Z$$

where A(t) is increasing but otherwise unspecified. Model is fitted by maximising the modified partial likelihood. A goodness-of-fit test by considering the score functions is also computed by resampling methods.

Usage

```
prop.odds.subdist(formula, data = sys.parent(), cause = 1,
  beta = NULL, Nit = 10, detail = 0, start.time = 0,
  max.time = NULL, id = NULL, n.sim = 500, weighted.test = 0,
  profile = 1, sym = 0, cens.model = "KM", cens.formula = NULL,
  clusters = NULL, max.clust = 1000, baselinevar = 1,
  weights = NULL, cens.weights = NULL)
```

Arguments

formula	a formula object, with the response on the left of a '~' operator, and the terms on the right. The response must be an object as returned by the 'Event' function.
data	a data.frame with the variables.
cause	cause indicator for competing risks.
beta	starting value for relative risk estimates
Nit	number of iterations for Newton-Raphson algorithm.
detail	if 0 no details is printed during iterations, if 1 details are given.
start.time	start of observation period where estimates are computed.
max.time	end of observation period where estimates are computed. Estimates thus computed from [start.time, max.time]. This is very useful to obtain stable estimates, especially for the baseline. Default is max of data.
id	For timevarying covariates the variable must associate each record with the id of a subject.
n.sim	number of simulations in resampling.
weighted.test	to compute a variance weighted version of the test-processes used for testing time-varying effects.
profile	use profile version of score equations.
sym	to use symmetrized second derivative in the case of the estimating equation approach (profile=0). This may improve the numerical performance.
cens.model	specifies censoring model. So far only Kaplan-Meier "KM".
cens.formula	possible formula for censoring distribution covariates. Default all !
clusters	to compute cluster based standard errors.
max.clust	number of maximum clusters to be used, to save time in iid decomposition.
baselinevar	set to 0 to save time on computations.
weights	additional weights.
cens.weights	specify censoring weights related to the observations.

Details

An alternative way of writing the model :

$$F_1(t; X, Z) = \frac{\exp(\beta^T Z)}{(A(t)) + \exp(\beta^T Z)}$$

such that β is the log-odds-ratio of cause 1 before time t, and $A(t)$ is the odds-ratio.

The modelling formula uses the standard survival modelling given in the **survival** package.

The data for a subject is presented as multiple rows or "observations", each of which applies to an interval of observation (start, stop]. The program essentially assumes no ties, and if such are present a little random noise is added to break the ties.

Value

returns an object of type 'cox.aalen'. With the following arguments:

`cum`	cumulative timevarying regression coefficient estimates are computed within the estimation interval.
`var.cum`	the martingale based pointwise variance estimates.
`robvar.cum`	robust pointwise variances estimates.
`gamma`	estimate of proportional odds parameters of model.
`var.gamma`	variance for gamma.
`robvar.gamma`	robust variance for gamma.
`residuals`	list with residuals. Estimated martingale increments (dM) and corresponding time vector (time).
`obs.testBeq0`	observed absolute value of supremum of cumulative components scaled with the variance.
`pval.testBeq0`	p-value for covariate effects based on supremum test.
`sim.testBeq0`	resampled supremum values.
`obs.testBeqC`	observed absolute value of supremum of difference between observed cumulative process and estimate under null of constant effect.
`pval.testBeqC`	p-value based on resampling.
`sim.testBeqC`	resampled supremum values.
`obs.testBeqC.is`	observed integrated squared differences between observed cumulative and estimate under null of constant effect.
`pval.testBeqC.is`	p-value based on resampling.
`sim.testBeqC.is`	resampled supremum values.
`conf.band`	resampling based constant to construct robust 95% uniform confidence bands.
`test.procBeqC`	observed test-process of difference between observed cumulative process and estimate under null of constant effect over time.
`loglike`	modified partial likelihood, pseudo profile likelihood for regression parameters.
`D2linv`	inverse of the derivative of the score function.
`score`	value of score for final estimates.
`test.procProp`	observed score process for proportional odds regression effects.
`pval.Prop`	p-value based on resampling.
`sim.supProp`	re-sampled supremum values.
`sim.test.procProp`	list of 50 random realizations of test-processes for constant proportional odds under the model based on resampling.

Author(s)

Thomas Scheike

References

Eriksson, Li, Zhang and Scheike (2014), The proportional odds cumulative incidence model for competing risks, Biometrics, to appear.

Scheike, A flexible semiparametric transformation model for survival data, Lifetime Data Anal. (2007).

Martinussen and Scheike, Dynamic Regression Models for Survival Data, Springer (2006).

Examples

```
library(timereg)
data(bmt)
# Fits Proportional odds model
out <- prop.odds.subdist(Event(time,cause)~platelet+age+tcell,data=bmt,
 cause=1,cens.model="KM",detail=0,n.sim=1000)
summary(out)
par(mfrow=c(2,3))
plot(out,sim.ci=2);
plot(out,score=1)

# simple predict function without confidence calculations
pout <- predictpropodds(out,X=model.matrix(~platelet+age+tcell,data=bmt)[,-1])
matplot(pout$time,pout$pred,type="l")

# predict function with confidence intervals
pout2 <- predict(out,Z=c(1,0,1))
plot(pout2,col=2)
pout1 <- predictpropodds(out,X=c(1,0,1))
lines(pout1$time,pout1$pred,type="l")

# Fits Proportional odds model with stratified baseline, does not work yet!
###out <- Gprop.odds.subdist(Surv(time,cause==1)~-1+factor(platelet)+
###prop(age)+prop(tcell),data=bmt,cause=bmt$cause,
###cens.code=0,cens.model="KM",causeS=1,detail=0,n.sim=1000)
###summary(out)
###par(mfrow=c(2,3))
###plot(out,sim.ci=2);
###plot(out,score=1)
```

---

pval *For internal use*

Description

for internal use

Author(s)

Thomas Scheike

`qcut` *Cut a variable*

Description

Calls the cut function to cut variables on data frame.

Usage

```
qcut(x, cuts = 4, breaks = NULL, ...)
```

Arguments

`x`	variable to cut
`cuts`	number of groups, 4 gives quartiles
`breaks`	can also give breaks
`...`	other argument for cut function of R

Author(s)

Thomas Scheike

Examples

```
data(sTRACE)
gx <- qcut(sTRACE$age)
table(gx)
```

rchaz *Simulation of Piecewise constant hazard model (Cox).*

Description

Simulates data from piecwise constant baseline hazard that can also be of Cox type. Censor data at highest value of the break points.

Usage

```
rchaz(cumhazard, rr, n = NULL, entry = NULL, cum.hazard = TRUE,
  cause = 1, extend = FALSE)
```

Arguments

cumhazard	cumulative hazard, or piece-constant rates for periods defined by first column of input.
rr	number of simulations or vector of relative risk for simuations.
n	number of simulations given as "n"
entry	delayed entry time for simuations.
cum.hazard	specifies wheter input is cumulative hazard or rates.
cause	name of cause
extend	to extend piecewise constant with constant rate. Default is average rate over time from cumulative (when TRUE), if numeric then uses given rate.

Author(s)

Thomas Scheike

Examples

```
rates <-  c(0,0.01,0.052,0.01,0.04)
breaks <- c(0,10,   20,  30,   40)
haz <- cbind(breaks,rates)
n <- 1000
X <- rbinom(n,1,0.5)
beta <- 0.2
rrcox <- exp(X * beta)
cumhaz <- cumsum(c(0,diff(breaks)*rates[-1]))
cumhaz <- cbind(breaks,cumhaz)

pctime <- rchaz(haz,n=1000,cum.hazard=FALSE)

par(mfrow=c(1,2))
ss <- aalen(Surv(time,status)~+1,data=pctime,robust=0)
plot(ss)
```

```
lines(cumhaz,col=2,lwd=2)

pctimecox <- rchaz(cumhaz,rrcox)
pctime <- cbind(pctime,X)
ssx <- cox.aalen(Surv(time,status)~+prop(X),data=pctimecox,robust=0)
plot(ssx)
lines(cumhaz,col=2,lwd=2)

### simulating data with hazard as real data
data(TRACE)

par(mfrow=c(1,2))
ss <- cox.aalen(Surv(time,status==9)~+prop(vf),data=TRACE,robust=0)
par(mfrow=c(1,2))
plot(ss)
###
pctime <- rchaz(ss$cum,n=1000)
###
sss <- aalen(Surv(time,status)~+1,data=pctime,robust=0)
lines(sss$cum,col=2,lwd=2)

pctime <- rchaz(ss$cum,rrcox)
pctime <- cbind(pctime,X)
###
sss <- cox.aalen(Surv(time,status)~+prop(X),data=pctime,robust=0)
summary(sss)
plot(ss)
lines(sss$cum,col=3,lwd=3)
```

`rcrisk` *Simulation of Piecewise constant hazard models with two causes (Cox).*

Description

Simulates data from piecwise constant baseline hazard that can also be of Cox type. Censor data at highest value of the break points.

Usage

```
rcrisk(cumhaz1, cumhaz2, rr1, rr2, cens = NULL, rrc = NULL, ...)
```

Arguments

`cumhaz1`	cumulative hazard of cause 1
`cumhaz2`	cumulative hazard of cause 1
`rr1`	number of simulations or vector of relative risk for simuations.
`rr2`	number of simulations or vector of relative risk for simuations.

cens	to censor further , rate or cumumlative hazard
rrc	retlativ risk for censoring.
...	arguments for rchaz

Author(s)

Thomas Scheike

Examples

```
data(TRACE)

cox1 <- cox.aalen(Surv(time,status==9)~prop(vf)+prop(chf)+prop(wmi),
            data=TRACE,robust=0)
cox2 <-  cox.aalen(Surv(time,status==0)~prop(vf)+prop(chf)+prop(wmi),
            data=TRACE,robust=0)

X1 <- TRACE[,c("vf","chf","wmi")]
n <- 1000
xid <- sample(1:nrow(X1),n,replace=TRUE)
Z1 <- X1[xid,]
Z2 <- X1[xid,]
rr1 <- exp(as.matrix(Z1) %*% cox1$gamma)
rr2 <- exp(as.matrix(Z2) %*% cox2$gamma)

cumhaz1 <- cox1$cum
cumhaz2 <- cox2$cum
d <-  rcrisk(cox1$cum,cox2$cum,rr1,rr2)
dd <- cbind(d,Z1)
sc1 <-   cox.aalen(Surv(time,status==1)~prop(vf)+prop(chf)+prop(wmi),
                 data=dd,robust=0)
cbind(sc1$gamma, cox1$gamma)
sc2 <-  cox.aalen(Surv(time,status==2)~prop(vf)+prop(chf)+prop(wmi),
                 data=dd,robust=0)
cbind(sc2$gamma, cox2$gamma)
par(mfrow=c(1,2))
plot(cox1); lines(sc1$cum,col=2)
plot(cox2$cum,type="l");
lines(sc2$cum,col=2)
```

---

recurrent.marginal.coxmean

Estimates marginal mean of recurrent events based on two cox models

---

Description

Fitting two Cox models for death and recurent events these are combined to prducte the estimator

$$\int_0^t S(u|x=0)dR(u|x=0)$$

the mean number of recurrent events, here

$$S(u|x=0)$$

is the probability of survival, and

$$dR(u|x=0)$$

is the probability of an event among survivors. For now the estimator is based on the two-baselines so

$$x=0$$

, but covariates can be rescaled to look at different x's and extensions possible.

Usage

```
recurrent.marginal.coxmean(recurrent, death)
```

Arguments

recurrent	aalen model for recurrent events
death	cox.aalen (cox) model for death events

Details

IID versions along the lines of Ghosh & Lin (2000) variance. See also mets package for quick version of this for large data. IID versions used for Ghosh & Lin (2000) variance. See also mets package for quick version of this for large data mets:::recurrent.marginal, these two version should give the same when there are now ties.

Author(s)

Thomas Scheike

References

Ghosh and Lin (2002) Nonparametric Analysis of Recurrent events and death, Biometrics, 554–562.

Examples

```
### do not test because iid slow  and uses data from mets
library(mets)
data(base1cumhaz)
data(base4cumhaz)
data(drcumhaz)
dr <- drcumhaz
```

```
base1 <- base1cumhaz
base4 <- base4cumhaz
rr <- simRecurrent(100,base1,death.cumhaz=dr)
rr$x <- rnorm(nrow(rr))
rr$strata <- floor((rr$id-0.01)/50)
drename(rr) <- start+stop~entry+time

ar <- cox.aalen(Surv(start,stop,status)~+1+prop(x)+cluster(id),data=rr,
                  resample.iid=1,,max.clust=NULL,max.timepoint.sim=NULL)
ad <- cox.aalen(Surv(start,stop,death)~+1+prop(x)+cluster(id),data=rr,
                  resample.iid=1,,max.clust=NULL,max.timepoint.sim=NULL)
mm <- recurrent.marginal.coxmean(ar,ad)
with(mm,plot(times,mu,type="s"))
with(mm,lines(times,mu+1.96*se.mu,type="s",lty=2))
with(mm,lines(times,mu-1.96*se.mu,type="s",lty=2))
```

`recurrent.marginal.mean`

Estimates marginal mean of recurrent events

Description

Fitting two aalen models for death and recurent events these are combined to prducte the estimator

$$\int_0^t S(u) dR(u)$$

the mean number of recurrent events, here

$$S(u)$$

is the probability of survival, and

$$dR(u)$$

is the probability of an event among survivors.

Usage

```
recurrent.marginal.mean(recurrent, death)
```

Arguments

`recurrent`	aalen model for recurrent events
`death`	aalen model for recurrent events

Details

IID versions used for Ghosh & Lin (2000) variance. See also mets package for quick version of this for large data mets:::recurrent.marginal, these two version should give the same when there are no ties.

Author(s)

Thomas Scheike

References

Ghosh and Lin (2002) Nonparametric Analysis of Recurrent events and death, Biometrics, 554–562.

Examples

```
### get some data using mets simulaitons
library(mets)
data(base1cumhaz)
data(base4cumhaz)
data(drcumhaz)
dr <- drcumhaz
base1 <- base1cumhaz
base4 <- base4cumhaz
rr <- simRecurrent(100,base1,death.cumhaz=dr)
rr$x <- rnorm(nrow(rr))
rr$strata <- floor((rr$id-0.01)/50)
drename(rr) <- start+stop~entry+time

ar <- aalen(Surv(start,stop,status)~+1+cluster(id),data=rr,resample.iid=1
                                                        ,max.clust=NULL)
ad <- aalen(Surv(start,stop,death)~+1+cluster(id),data=rr,resample.iid=1,
                                                        ,max.clust=NULL)
mm <- recurrent.marginal.mean(ar,ad)
with(mm,plot(times,mu,type="s"))
with(mm,lines(times,mu+1.96*se.mu,type="s",lty=2))
with(mm,lines(times,mu-1.96*se.mu,type="s",lty=2))
```

`res.mean` *Residual mean life (restricted)*

Description

Fits a semiparametric model for the residual life (estimator=1):

$$E(\min(Y,\tau) - t | Y >= t) = h_1(g(t,x,z))$$

or cause specific years lost of Andersen (2012) (estimator=3)

$$E(\tau - \min(Y_j,\tau) | Y >= 0) = \int_0^t (1 - F_j(s))ds = h_2(g(t,x,z))$$

where $Y_j = \sum_j Y I(\epsilon = j) + \infty * I(\epsilon = 0)$ or (estimator=2)

$$E(\tau - \min(Y_j,\tau) | Y < \tau, \epsilon = j) = h_3(g(t,x,z)) = h_2(g(t,x,z))F_j(\tau,x,z)$$

where $F_j(s,x,z) = P(Y < \tau, \epsilon = j | x,z)$ for a known link-function $h()$ and known prediction-function $g(t,x,z)$

Usage

```
res.mean(formula, data = sys.parent(), cause = 1, restricted = NULL,
  times = NULL, Nit = 50, clusters = NULL, gamma = 0, n.sim = 0,
  weighted = 0, model = "additive", detail = 0, interval = 0.01,
  resample.iid = 1, cens.model = "KM", cens.formula = NULL,
  time.pow = NULL, time.pow.test = NULL, silent = 1, conv = 1e-06,
  estimator = 1, cens.weights = NULL, conservative = 1,
  weights = NULL)
```

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 'Event' function. The status indicator is not important here. Time-invariant regressors are specified by the wrapper const(), and cluster variables (for computing robust variances) by the wrapper cluster().
data	a data.frame with the variables.
cause	For competing risk models specificies which cause we consider.
restricted	gives a possible restriction times for means.
times	specifies the times at which the estimator is considered. Defaults to all the times where an event of interest occurs, with the first 10 percent or max 20 jump points removed for numerical stability in simulations.
Nit	number of iterations for Newton-Raphson algorithm.
clusters	specifies cluster structure, for backwards compability.
gamma	starting value for constant effects.
n.sim	number of simulations in resampling.
weighted	Not implemented. To compute a variance weighted version of the test-processes used for testing time-varying effects.
model	"additive", "prop"ortional.
detail	if 0 no details are printed during iterations, if 1 details are given.
interval	specifies that we only consider timepoints where the Kaplan-Meier of the censoring distribution is larger than this value.
resample.iid	to return the iid decomposition, that can be used to construct confidence bands for predictions
cens.model	specified which model to use for the ICPW, KM is Kaplan-Meier alternatively it may be "cox" or "aalen" model for further flexibility.
cens.formula	specifies the regression terms used for the regression model for chosen regression model. When cens.model is specified, the default is to use the same design as specified for the competing risks model. "KM","cox","aalen","weights". "weights" are user specified weights given is cens.weight argument.
time.pow	specifies that the power at which the time-arguments is transformed, for each of the arguments of the const() terms, default is 1 for the additive model and 0 for the proportional model.

`time.pow.test`
: specifies that the power the time-arguments is transformed for each of the arguments of the non-const() terms. This is relevant for testing if a coefficient function is consistent with the specified form A_l(t)=beta_l t^time.pow.test(l). Default is 1 for the additive model and 0 for the proportional model.

`silent`
: if 0 information on convergence problems due to non-invertible derviates of scores are printed.

`conv`
: gives convergence criterie in terms of sum of absolute change of parameters of model

`estimator`
: specifies what that is estimated.

`cens.weights`
: censoring weights for estimating equations.

`conservative`
: for slightly conservative standard errors.

`weights`
: weights for estimating equations.

Details

Uses the IPCW for the score equations based on

$$w(t)\Delta(\tau)/P(\Delta(\tau)=1|T,\epsilon,X,Z)(Y(t)-h_1(t,X,Z))$$

and where $\Delta(\tau)$ is the at-risk indicator given data and requires a IPCW model.

Since timereg version 1.8.4. the response must be specified with the `Event` function instead of the `Surv` function and the arguments.

Value

returns an object of type 'comprisk'. With the following arguments:

`cum`
: cumulative timevarying regression coefficient estimates are computed within the estimation interval.

`var.cum`
: pointwise variances estimates.

`gamma`
: estimate of proportional odds parameters of model.

`var.gamma`
: variance for gamma.

`score`
: sum of absolute value of scores.

`gamma2`
: estimate of constant effects based on the non-parametric estimate. Used for testing of constant effects.

`obs.testBeq0`
: observed absolute value of supremum of cumulative components scaled with the variance.

`pval.testBeq0`
: p-value for covariate effects based on supremum test.

`obs.testBeqC`
: observed absolute value of supremum of difference between observed cumulative process and estimate under null of constant effect.

`pval.testBeqC`
: p-value based on resampling.

`obs.testBeqC.is`
: observed integrated squared differences between observed cumulative and estimate under null of constant effect.

`pval.testBeqC.is`
: p-value based on resampling.

`conf.band`
: resampling based constant to construct 95% uniform confidence bands.

`B.iid`
: list of iid decomposition of non-parametric effects.

`gamma.iid`
: matrix of iid decomposition of parametric effects.

`test.procBeqC`
: observed test process for testing of time-varying effects

`sim.test.procBeqC`
: 50 resample processes for for testing of time-varying effects

`conv`
: information on convergence for time points used for estimation.

Author(s)

Thomas Scheike

References

Andersen (2013), Decomposition of number of years lost according to causes of death, Statistics in Medicine, 5278-5285.

Scheike, and Cortese (2015), Regression Modelling of Cause Specific Years Lost,

Scheike, Cortese and Holmboe (2015), Regression Modelling of Restricted Residual Mean with Delayed Entry,

Examples

```
data(bmt);
tau <- 100

### residual restricted mean life
out<-res.mean(Event(time,cause>=1)~factor(tcell)+factor(platelet),data=bmt,cause=1,
      times=0,restricted=tau,n.sim=0,model="additive",estimator=1);
summary(out)

out<-res.mean(Event(time,cause>=1)~factor(tcell)+factor(platelet),data=bmt,cause=1,
      times=seq(0,90,5),restricted=tau,n.sim=0,model="additive",estimator=1);
par(mfrow=c(1,3))
plot(out)

### restricted years lost given death
out21<-res.mean(Event(time,cause)~factor(tcell)+factor(platelet),data=bmt,cause=1,
      times=0,restricted=tau,n.sim=0,model="additive",estimator=2);
summary(out21)
out22<-res.mean(Event(time,cause)~factor(tcell)+factor(platelet),data=bmt,cause=2,
      times=0,restricted=tau,n.sim=0,model="additive",estimator=2);
summary(out22)
```

```
### total restricted years lost
out31<-res.mean(Event(time,cause)~factor(tcell)+factor(platelet),data=bmt,cause=1,
      times=0,restricted=tau,n.sim=0,model="additive",estimator=3);
summary(out31)
out32<-res.mean(Event(time,cause)~factor(tcell)+factor(platelet),data=bmt,cause=2,
      times=0,restricted=tau,n.sim=0,model="additive",estimator=3);
summary(out32)


### delayed entry
nn <- nrow(bmt)
entrytime <- rbinom(nn,1,0.5)*(bmt$time*runif(nn))
bmt$entrytime <- entrytime

bmtw <- prep.comp.risk(bmt,times=tau,time="time",entrytime="entrytime",cause="cause")

out<-res.mean(Event(time,cause>=1)~factor(tcell)+factor(platelet),data=bmtw,cause=1,
      times=0,restricted=tau,n.sim=0,model="additive",estimator=1,
               cens.model="weights",weights=bmtw$cw,cens.weights=1/bmtw$weights);
summary(out)
```

`restricted.residual.mean` *Estimates restricted residual mean for Cox or Aalen model*

Description

The restricted means are the

$$\int_0^\tau S(t)dt$$

the standard errors are computed using the i.i.d. decompositions from the cox.aalen (that must be called with the argument "max.timpoint.sim=NULL") or aalen function.

Usage

```
restricted.residual.mean(out, x = 0, tau = 10, iid = 0)
```

Arguments

`out`	an "cox.aalen" with a Cox model or an "aalen" model.
`x`	matrix with covariates for Cox model or additive hazards model (aalen).
`tau`	restricted residual mean.
`iid`	if iid=1 then uses iid decomposition for estimation of standard errors.

Details

must have computed iid decomposition of survival models for standard errors to be computed. Note that competing risks models can be fitted but then the interpretation is not clear.

Value

Returns an object. With the following arguments:

`mean`	restricted mean for different covariates.
`var.mean`	variance matrix.
`se`	standard errors.
`S0tau`	estimated survival functions on time-range [0,tau].
`timetau`	vector of time arguments for S0tau.

Author(s)

Thomas Scheike

References

D. M. Zucker, Restricted mean life with covariates: Modification and extension of a useful survival analysis method, J. Amer. Statist. Assoc. vol. 93 pp. 702-709, 1998.

Martinussen and Scheike, Dynamic Regression Models for Survival Data, Springer (2006).

Examples

```
### this example runs slowly and is therefore donttest
data(sTRACE)
sTRACE$cage <- scale(sTRACE$age)
# Fits Cox model  and aalen model
out<-cox.aalen(Surv(time,status>=1)~prop(sex)+prop(diabetes)+prop(chf)+
       prop(vf),data=sTRACE,max.timepoint.sim=NULL,resample.iid=1)
outa<-aalen(Surv(time,status>=1)~sex+diabetes+chf+vf,
data=sTRACE,resample.iid=1)

coxrm <- restricted.residual.mean(out,tau=7,
   x=rbind(c(0,0,0,0),c(0,0,1,0),c(0,0,1,1),c(0,0,0,1)),iid=1)
plot(coxrm)
summary(coxrm)

### aalen model not optimal here
aalenrm <- restricted.residual.mean(outa,tau=7,
   x=rbind(c(1,0,0,0,0),c(1,0,0,1,0),c(1,0,0,1,1),c(1,0,0,0,1)),iid=1)
with(aalenrm,matlines(timetau,S0tau,type="s",ylim=c(0,1)))
legend("bottomleft",c("baseline","+chf","+chf+vf","+vf"),col=1:4,lty=1)
summary(aalenrm)

mm <-cbind(coxrm$mean,coxrm$se,aalenrm$mean,aalenrm$se)
```

```
colnames(mm)<-c("cox-res-mean","se","aalen-res-mean","se")
rownames(mm)<-c("baseline","+chf","+chf+vf","+vf")
mm
```

`sim.cause.cox` *Simulation of cause specific from Cox models.*

Description

Simulates data that looks like fit from cause specific Cox models. Censor data automatically. When censoring is given in the list of causes this will give censoring that looks like the data. Covariates are drawn from data-set with replacement. This gives covariates like the data.

Usage

```
## S3 method for class 'cause.cox'
sim(coxs, n, data = NULL, cens = NULL,
  rrc = NULL, ...)
```

Arguments

`coxs`	list of cox models.
`n`	number of simulations.
`data`	to extract covariates for simulations (draws from observed covariates).
`cens`	specifies censoring model, if NULL then only censoring for each cause at end of last event of this type. if "is.matrix" then uses cumulative. hazard given, if "is.scalar" then uses rate for exponential, and if not given then takes average rate of in simulated data from cox model. But censoring can also be given as a cause.
`rrc`	possible vector of relative risk for cox-type censoring.
`...`	arguments for rchaz, for example entry-time

Author(s)

Thomas Scheike

Examples

```
nsim <- 1000
data(bmt)
cox1 <- cox.aalen(Surv(time,cause==1)~prop(tcell)+prop(platelet),data=bmt,robust=0)
cox2 <- cox.aalen(Surv(time,cause==2)~prop(tcell)+prop(platelet),data=bmt,robust=0)
coxs <- list(cox1,cox2)
dd <- sim.cause.cox(coxs,nsim,data=bmt)
scox1 <- cox.aalen(Surv(time,status==1)~prop(tcell)+prop(platelet),data=dd,robust=0)
```

```
scox2 <- cox.aalen(Surv(time,status==2)~prop(tcell)+prop(platelet),data=dd,robust=0)
###
cbind(cox1$gamma,scox1$gamma)
cbind(cox2$gamma,scox2$gamma)
par(mfrow=c(1,2))
plot(cox1); lines(scox1$cum,col=2)
plot(cox2$cum,type="l");
lines(scox2$cum,col=2)



### do not test to avoid dependence on mets
library(mets)
data(bmt)
cox1 <- phreg(Surv(time,cause==1)~tcell+platelet,data=bmt)
cox2 <- phreg(Surv(time,cause==2)~tcell+platelet,data=bmt)
coxs <- list(cox1,cox2)
dd <- sim.cause.cox(coxs,nsim,data=bmt)
scox1 <- phreg(Surv(time,status==1)~tcell+platelet,data=dd)
scox2 <- phreg(Surv(time,status==2)~tcell+platelet,data=dd)
cbind(cox1$coef,scox1$coef)
cbind(cox2$coef,scox2$coef)
par(mfrow=c(1,2))
basehazplot.phreg(cox1); basehazplot.phreg(scox1,add=TRUE);
basehazplot.phreg(cox2); basehazplot.phreg(scox2,add=TRUE);

cox1 <- phreg(Surv(time,cause==1)~strata(tcell)+platelet,data=bmt)
cox2 <- phreg(Surv(time,cause==2)~strata(tcell)+platelet,data=bmt)
coxs <- list(cox1,cox2)
dd <- sim.cause.cox(coxs,nsim,data=bmt)
scox1 <- phreg(Surv(time,status==1)~strata(tcell)+platelet,data=dd)
scox2 <- phreg(Surv(time,status==2)~strata(tcell)+platelet,data=dd)
cbind(cox1$coef,scox1$coef)
cbind(cox2$coef,scox2$coef)
par(mfrow=c(1,2))
basehazplot.phreg(cox1); basehazplot.phreg(scox1,add=TRUE);
basehazplot.phreg(cox2); basehazplot.phreg(scox2,add=TRUE);

# coxph
cox1 <- coxph(Surv(time,cause==1)~tcell+platelet,data=bmt)
cox2 <- coxph(Surv(time,cause==2)~tcell+platelet,data=bmt)
coxs <- list(cox1,cox2)
dd <- sim.cause.cox(coxs,nsim,data=bmt)
scox1 <- coxph(Surv(time,status==1)~tcell+platelet,data=dd)
scox2 <- coxph(Surv(time,status==2)~tcell+platelet,data=dd)
cbind(cox1$coef,scox1$coef)
cbind(cox2$coef,scox2$coef)
```

`sim.cif` *Simulation of output from Cumulative incidence regression model*

Description

Simulates data that looks like fit from fitted cumulative incidence model

Usage

```
## S3 method for class 'cif'
sim(cif, n, data = NULL, Z = NULL, drawZ = TRUE,
  cens = NULL, rrc = NULL, cumstart = c(0, 0), ...)
```

Arguments

cif	output form prop.odds.subdist or ccr (cmprsk), can also call invsubdist with with cumulative and linear predictor
n	number of simulations.
data	to extract covariates for simulations (draws from observed covariates).
Z	to use these covariates for simulation rather than drawing new ones.
drawZ	to random sample from Z or not
cens	specifies censoring model, if "is.matrix" then uses cumulative hazard given, if "is.scalar" then uses rate for exponential, and if not given then takes average rate of in simulated data from cox model.
rrc	possible vector of relative risk for cox-type censoring.
cumstart	to start cumulatives at time 0 in 0.
...	arguments for invsubdist

Author(s)

Thomas Scheike

Examples

```
data(TRACE)

## Logit link for proportional odds model, using comp.risk to save time
#' cif <-  prop.odds.subdist(Event(time,status)~vf+chf+wmi,data=TRACE,cause=9)
cif <-  comp.risk(Event(time,status)~const(vf)+const(chf)+const(wmi),
                  data=TRACE,cause=9,model="logistic2")
sim1 <- sim.cif(cif,500,data=TRACE)
#' cc <-  prop.odds.subdist(Event(time,status)~vf+chf+wmi,data=sim1,cause=1)
cc <-  comp.risk(Event(time,status)~const(vf)+const(chf)+const(wmi),
                  data=sim1,cause=1,model="logistic2")
cbind(cif$gamma,cc$gamma)

plot(cif)
lines(cc$cum)

###################################################################
## Fine-Gray model model, using comp.risk to avoid dependcies
###################################################################
```

```
cif <-  comp.risk(Event(time,status)~const(vf)+const(chf)+const(wmi),
                  data=TRACE,cause=9)
sim1 <- sim.cif(cif,500,data=TRACE)
#' cc <-  crr
cc <-  comp.risk(Event(time,status)~const(vf)+const(chf)+const(wmi),
                  data=sim1,cause=1)
cbind(cif$gamma,cc$gamma)
plot(cif)
lines(cc$cum)

# data(TRACE)
#  mm <- model.matrix(~vf+chf+wmi,data=TRACE)[,-1]
#  library(cmprsk)
#  cif <-  crr(TRACE$time,TRACE$status,mm,failcode=9)
#  sim1 <- sim.cif(cif,10000,data=TRACE,Z=mm)
#  mms <- model.matrix(~vf+chf+wmi,data=sim1)[,-1]
#  #' cc <-  prop.odds.subdist(Event(time,status)~vf+chf+wmi,data=sim1,cause=1)
#  cif1 <-  crr(sim1$time,sim1$status,mms,failcode=1)
#  cbind(cif$coef,cif1$coef)
#
################################################################
#  simulating several causes with specific cumulatives
################################################################
data(bmt)
cif1 <-  comp.risk(Event(time,cause)~const(tcell)+const(age),
                  data=bmt,cause=1,model="logistic2")
cif2 <-  comp.risk(Event(time,cause)~const(tcell)+const(age),
                  data=bmt,cause=2,model="logistic2")

## must look at same time-scale
cifs <- pre.cifs(list(cif1,cif2))
plot(cifs[[1]]$cum,type="l")
lines(cifs[[2]]$cum,col=2)
legend("topleft",c("cause1","cause2"),lty=1,col=1:2)

 n <- 500
 sim1 <- sim.cif(cifs[[1]],n,data=bmt)
 Z <- sim1[,c("tcell","age")]
 sim2 <- sim.cif(cifs[[2]],n,data=bmt,Z=Z,drawZ=FALSE)
 ###
 rt <- rbinom(n,1,(sim1$F1tau+sim2$F1tau))
 rb <- rbinom(n,1,sim1$F1tau/(sim1$F1tau+sim2$F1tau))
 cause=ifelse(rb==1,1,2)
 time=ifelse(cause==1,sim1$timecause,sim2$timecause)
 cause <- rt*cause
 time[cause==0] <- tail(cifs[[1]]$cum[,1],1)

 bt <- data.frame(time=time,cause=cause,tcell=sim1$tcell,age=sim1$age)
 scif1 <-  comp.risk(Event(time,cause)~const(tcell)+const(age),
                   data=bt,cause=1,model="logistic2")
 scif2 <-  comp.risk(Event(time,cause)~const(tcell)+const(age),
                   data=bt,cause=2,model="logistic2")
```

```
 plot(scif1$cum,type="l")
 lines(scif2$cum,col=1,lty=2)
 legend("topleft",c("cause1","cause2"),lty=1:2,col=1:1)
 lines(cifs[[1]]$cum,col=2)
 lines(cifs[[2]]$cum,col=2,lty=2)

#  Everyhing wraped in a call assuming covariates work in the same way for two models
dd <- sim.cifs(list(cif1,cif2),2000,data=bmt)
scif1 <-  comp.risk(Event(time,cause)~const(tcell)+const(age),
                  data=dd,cause=1,model="logistic2")
scif2 <-  comp.risk(Event(time,cause)~const(tcell)+const(age),
                  data=dd,cause=2,model="logistic2")

plot(scif1$cum,type="l")
lines(scif2$cum,col=1,lty=2)
legend("topleft",c("cause1","cause2"),lty=1:2,col=1:1)
lines(cifs[[1]]$cum,col=2)
lines(cifs[[2]]$cum,col=2,lty=2)
```

`sim.cox` *Simulation of output from Cox model.*

Description

Simulates data that looks like fit from Cox model. Censor data automatically for highest value of the event times by using cumulative hazard.

Usage

```
## S3 method for class 'cox'
sim(cox, n, data = NULL, cens = NULL, rrc = NULL,
  entry = NULL, ...)
```

Arguments

`cox`	output form coxph or cox.aalen model fitting cox model.
`n`	number of simulations.
`data`	to extract covariates for simulations (draws from observed covariates).
`cens`	specifies censoring model, if "is.matrix" then uses cumulative hazard given, if "is.scalar" then uses rate for exponential, and if not given then takes average rate of in simulated data from cox model.
`rrc`	possible vector of relative risk for cox-type censoring.
`entry`	delayed entry variable for simulation.
`...`	arguments for rchaz, for example entry-time

Author(s)

Thomas Scheike

Examples

```
data(TRACE)

cox <-  coxph(Surv(time,status==9)~vf+chf+wmi,data=TRACE)
sim1 <- sim.cox(cox,1000,data=TRACE)
cc <- coxph(Surv(time,status)~vf+chf+wmi,data=sim1)
cbind(cox$coef,cc$coef)

cor(sim1[,c("vf","chf","wmi")])
cor(TRACE[,c("vf","chf","wmi")])

cox <-  cox.aalen(Surv(time, status==9) ~ prop(vf)+prop(chf)+prop(wmi),TRACE,robust=0)
sim2 <- sim.cox(cox,1000,data=TRACE)
cc <-  cox.aalen(Surv(time, status)~prop(vf)+prop(chf)+prop(wmi),data=sim2,robust=0)
###
plot(cox)
lines(cc$cum,type="s",col=2)
cbind(cox$gamma,cc$gamma)


### do not test to avoid dependence on mets
library(mets)
cox <-  phreg(Surv(time, status==9)~vf+chf+wmi,data=TRACE)
sim3 <- sim.cox(cox,1000,data=TRACE)
cc <-  phreg(Surv(time, status)~vf+chf+wmi,data=sim3)
cbind(cox$coef,cc$coef)
basehazplot.phreg(cox,se=TRUE)
lines(cc$cumhaz,col=2)

cox <-  phreg(Surv(time,status==9)~strata(chf)+vf+wmi,data=TRACE)
sim3 <- sim.cox(cox,1000,data=TRACE)
cc <-   phreg(Surv(time, status)~strata(chf)+vf+wmi,data=sim3)
cbind(cox$coef,cc$coef)
basehazplot.phreg(cox)
basehazplot.phreg(cc,add=TRUE)
```

---

simsubdist *Simulation from subdistribution function assuming piecwise linearity*

Description

Simulation from subdistribution function assuming piecwise linearity for Fine-Gray or logistic link.

Usage

```
simsubdist(cumhazard, rr, entry = NULL, type = "cloglog",
  startcum = c(0, 0), ...)
```

Arguments

cumhazard	matrix that specified times and values of some cumulative hazard.
rr	"relative risk" terms
entry	not implemented yet
type	either cloglog or logistic
startcum	c(0,0) to make cumulativ start in 0 with 0 cumhazard.
...	further arguments

Author(s)

Thomas Scheike

Examples

```
data(sTRACE)

cif <- comp.risk(Event(time,status)~const(vf),data=sTRACE,cause=9,model="logistic2")
cumhaz <- cif$cum

## 1000 logistic without covariates with baseline from model fit
sim1 <- simsubdist(cumhaz,1000,type="logistic")
###
cifs <- comp.risk(Event(time,status)~+1,data=sim1,cause=1,model="logistic2")
###
plot(cifs)
lines(cifs$cum,col=2)

## 1000 logistic with covariates with baseline from model fit
x <- rbinom(1000,1,0.5)
rr <- exp(x*0.3)
sim1 <- simsubdist(cumhaz,rr,type="logistic")
sim1$x <- x

cifs <- comp.risk(Event(time,status)~+const(x),data=sim1,cause=1,model="logistic2")
###
cifs$gamma
plot(cifs)
lines(cumhaz,col=2)

#################################################################
### simulation of cumulative incidence with specified functions
#################################################################
F1logit<-function(t,lam0=0.2,beta=0.3,x=0)
{ pt <- t*lam0; rr <- exp(x*beta); return(pt*rr/(1+pt*rr)); }
```

```
F1p<-function(t,lam0=0.4,beta=0.3,x=0) # proportional version
{ return( 1 - exp(-(t*lam0)*exp(x*beta))) }

n=10000
tt=seq(0,3,by=.01)
tt=seq(0,3,by=.01)
t1 <- invsubdist(cbind(tt,F1p(tt)),runif(n))
t2 <- invsubdist(cbind(tt,F1p(tt,lam0=0.1)),runif(n))
rt <- rbinom(n,1,(F1p(3)+F1p(3,lam0=0.1)))
rb <- rbinom(n,1,F1p(3)/(F1p(3)+F1p(3,lam0=0.1)))
cause=ifelse(rb==1,1,2)
time=ifelse(cause==1,t1$time,t2$time)
cause <- rt*cause
time[cause==0] <- 3

datC=data.frame(time=time,cause=cause)
p1=comp.risk(Event(time,cause)~+1,data=datC,cause=1)
p2=comp.risk(Event(time,cause)~+1,data=datC,cause=2)
pp1=predict(p1,X=1,se=0)
pp2=predict(p2,X=1,se=0)
par(mfrow=c(1,2))
plot(pp1)
lines(tt,F1p(tt),col=2)
plot(pp2)
lines(tt,F1p(tt,lam0=0.1),col=2)

#to avoid dependencies when checking
#library(prodlim)
#pp=prodlim(Hist(time,cause)~+1)
#par(mfrow=c(1,2))
#plot(pp,cause="1")
#lines(tt,F1p(tt),col=2)
#plot(pp,cause="2")
#lines(tt,F1p(tt,lam0=0.1),col=2)
```

`summary.aalen` *Prints summary statistics*

Description

Computes p-values for test of significance for nonparametric terms of model, p-values for test of constant effects based on both supremum and integrated squared difference.

Usage

```
## S3 method for class 'aalen'
summary(object, digits = 3, ...)
```

Arguments

object	an aalen object.
digits	number of digits in printouts.
...	unused arguments - for S3 compatibility

Details

Returns parameter estimates and their standard errors.

Author(s)

Thomas Scheike

References

Martinussen and Scheike,

Examples

```
### see help(aalen)
```

---

summary.cum.residuals

Prints summary statistics for goodness-of-fit tests based on cumulative residuals

---

Description

Computes p-values for extreme behaviour relative to the model of various cumulative residual processes.

Usage

```
## S3 method for class 'cum.residuals'
summary(object, digits = 3, ...)
```

Arguments

object	output from the cum.residuals() function.
digits	number of digits in printouts.
...	unused arguments - for S3 compatibility

Author(s)

Thomas Scheike

Examples

```
# see cum.residuals for examples
```

`timecox` *Fit Cox model with partly timevarying effects.*

Description

Fits proportional hazards model with some effects time-varying and some effects constant. Time dependent variables and counting process data (multiple events per subject) are possible.

Usage

```
timecox(formula = formula(data), data = sys.parent(), start.time = 0,
  max.time = NULL, id = NULL, clusters = NULL, n.sim = 1000,
  residuals = 0, robust = 1, Nit = 20, bandwidth = 0.5,
  method = "basic", weighted.test = 0, degree = 1, covariance = 0)
```

Arguments

`formula`	a formula object with the response on the left of a '~' operator, and the independent terms on the right as regressors. The response must be a survival object as returned by the 'Surv' function. Time-invariant regressors are specified by the wrapper const(), and cluster variables (for computing robust variances) by the wrapper cluster().
`data`	a data.frame with the variables.
`start.time`	start of observation period where estimates are computed.
`max.time`	end of observation period where estimates are computed. Estimates thus computed from [start.time, max.time]. Default is max of data.
`id`	For timevarying covariates the variable must associate each record with the id of a subject.
`clusters`	cluster variable for computation of robust variances.
`n.sim`	number of simulations in resampling.
`residuals`	to returns residuals that can be used for model validation in the function cum.residuals
`robust`	to compute robust variances and construct processes for resampling. May be set to 0 to save memory.
`Nit`	number of iterations for score equations.
`bandwidth`	bandwidth for local iterations. Default is 50 % of the range of the considered observation period.
`method`	Method for estimation. This refers to different parametrisations of the baseline of the model. Options are "basic" where the baseline is written as $\lambda_0(t) = \exp(\alpha_0(t))$ or the "breslow" version where the baseline is parametrised as $\lambda_0(t)$.

`weighted.test`
: to compute a variance weighted version of the test-processes used for testing time-varying effects.

`degree`
: gives the degree of the local linear smoothing, that is local smoothing. Possible values are 1 or 2.

`covariance`
: to compute covariance estimates for nonparametric terms rather than just the variances.

Details

Resampling is used for computing p-values for tests of timevarying effects.

The modelling formula uses the standard survival modelling given in the **survival** package.

The data for a subject is presented as multiple rows or 'observations', each of which applies to an interval of observation (start, stop]. When counting process data with the)start,stop] notation is used, the 'id' variable is needed to identify the records for each subject. The program assumes that there are no ties, and if such are present random noise is added to break the ties.

Value

Returns an object of type "timecox". With the following arguments:

`cum`
: cumulative timevarying regression coefficient estimates are computed within the estimation interval.

`var.cum`
: the martingale based pointwise variance estimates.

`robvar.cum`
: robust pointwise variances estimates.

`gamma`
: estimate of parametric components of model.

`var.gamma`
: variance for gamma.

`robvar.gamma`
: robust variance for gamma.

`residuals`
: list with residuals. Estimated martingale increments (dM) and corresponding time vector (time).

`obs.testBeq0`
: observed absolute value of supremum of cumulative components scaled with the variance.

`pval.testBeq0`
: p-value for covariate effects based on supremum test.

`sim.testBeq0`
: resampled supremum values.

`obs.testBeqC`
: observed absolute value of supremum of difference between observed cumulative process and estimate under null of constant effect.

`pval.testBeqC`
: p-value based on resampling.

`sim.testBeqC`
: resampled supremum values.

`obs.testBeqC.is`
: observed integrated squared differences between observed cumulative and estimate under null of constant effect.

`pval.testBeqC.is`
: p-value based on resampling.

`sim.testBeqC.is`
: resampled supremum values.

`conf.band`
: resampling based constant to construct robust 95% uniform confidence bands.

`test.procBeqC`
: observed test-process of difference between observed cumulative process and estimate under null of constant effect over time.

`sim.test.procBeqC`
: list of 50 random realizations of test-processes under null based on resampling.

`schoenfeld.residuals`
: Schoenfeld residuals are returned for "breslow" parametrisation.

Author(s)

Thomas Scheike

References

Martinussen and Scheike, Dynamic Regression Models for Survival Data, Springer (2006).

Examples

```
data(sTRACE)
# Fits time-varying Cox model
out<-timecox(Surv(time/365,status==9)~age+sex+diabetes+chf+vf,
data=sTRACE,max.time=7,n.sim=100)

summary(out)
par(mfrow=c(2,3))
plot(out)
par(mfrow=c(2,3))
plot(out,score=TRUE)

# Fits semi-parametric time-varying Cox model
out<-timecox(Surv(time/365,status==9)~const(age)+const(sex)+
const(diabetes)+chf+vf,data=sTRACE,max.time=7,n.sim=100)

summary(out)
par(mfrow=c(2,3))
plot(out)
```

TRACE *The TRACE study group of myocardial infarction*

Description

The TRACE data frame contains 1877 patients and is a subset of a data set consisting of approximately 6000 patients. It contains data relating survival of patients after myocardial infarction to various risk factors.

sTRACE is a subsample consisting of 300 patients.

tTRACE is a subsample consisting of 1000 patients.

Format

This data frame contains the following columns:

id a numeric vector. Patient code.

status a numeric vector code. Survival status. 9: dead from myocardial infarction, 0: alive, 7: dead from other causes.

time a numeric vector. Survival time in years.

chf a numeric vector code. Clinical heart pump failure, 1: present, 0: absent.

diabetes a numeric vector code. Diabetes, 1: present, 0: absent.

vf a numeric vector code. Ventricular fibrillation, 1: present, 0: absent.

wmi a numeric vector. Measure of heart pumping effect based on ultrasound measurements where 2 is normal and 0 is worst.

sex a numeric vector code. 1: female, 0: male.

age a numeric vector code. Age of patient.

Source

The TRACE study group.

Jensen, G.V., Torp-Pedersen, C., Hildebrandt, P., Kober, L., F. E. Nielsen, Melchior, T., Joen, T. and P. K. Andersen (1997), Does in-hospital ventricular fibrillation affect prognosis after myocardial infarction?, European Heart Journal 18, 919–924.

Examples

```
data(TRACE)
names(TRACE)
```

---

two.stage *Fit Clayton-Oakes-Glidden Two-Stage model*

Description

Fit Clayton-Oakes-Glidden Two-Stage model with Cox-Aalen marginals and regression on the variance parameters.

Usage

```
two.stage(margsurv, data = sys.parent(), Nit = 60, detail = 0,
  start.time = 0, max.time = NULL, id = NULL, clusters = NULL,
  robust = 1, theta = NULL, theta.des = NULL, var.link = 0,
  step = 0.5, notaylor = 0, se.clusters = NULL)
```

Arguments

margsurv	fit of marginal survival cox.aalen model with residuals=2, and resample.iid=1 to get fully correct standard errors. See notaylor below.
data	a data.frame with the variables.
Nit	number of iterations for Newton-Raphson algorithm.
detail	if 0 no details is printed during iterations, if 1 details are given.
start.time	start of observation period where estimates are computed.
max.time	end of observation period where estimates are computed. Estimates thus computed from [start.time, max.time]. Default is max of data.
id	For timevarying covariates the variable must associate each record with the id of a subject.
clusters	cluster variable for computation of robust variances.
robust	if 0 then totally omits computation of standard errors.
theta	starting values for the frailty variance (default=0.1).
theta.des	design for regression for variances. The defauls is NULL that is equivalent to just one theta and the design with only a baseline.
var.link	default "0" is that the regression design on the variances is without a link, and "1" uses the link function exp.
step	step size for Newton-Raphson.
notaylor	if 1 then ignores variation due to survival model, this is quicker and then resample.iid=0 and residuals=0 is ok for marginal survival model that then is much quicker.
se.clusters	cluster variable for sandwich estimator of variance.

Details

The model specifikatin allows a regression structure on the variance of the random effects, such it is allowed to depend on covariates fixed within clusters

$$\theta_k = Q_k^T \nu$$

. This is particularly useful to model jointly different groups and to compare their variances.

Fits an Cox-Aalen survival model. Time dependent variables and counting process data (multiple events per subject) are not possible !

The marginal baselines are on the Cox-Aalen form

$$\lambda_{ki}(t) = Y_{ki}(t)(X_{ki}^T(t)\alpha(t)) \exp(Z_{ki}^T \beta)$$

The model thus contains the Cox's regression model and the additive hazards model as special cases. (see cox.aalen function for more on this).

The modelling formula uses the standard survival modelling given in the **survival** package. Only for right censored survival data.

The data for a subject is presented as multiple rows or 'observations', each of which applies to an interval of observation (start, stop]. For counting process data with the)start,stop] notation is used the 'id' variable is needed to identify the records for each subject. Only one record per subject is allowed in the current implementation for the estimation of theta. The program assumes that there are no ties, and if such are present random noise is added to break the ties.

Left truncation is dealt with. Here the key assumption is that the maginals are correctly estimated and that we have a common truncation time within each cluster.

Value

returns an object of type "two.stage". With the following arguments:

`cum`	cumulative timevarying regression coefficient estimates are computed within the estimation interval.
`var.cum`	the martingale based pointwise variance estimates.
`robvar.cum`	robust pointwise variances estimates.
`gamma`	estimate of parametric components of model.
`var.gamma`	variance for gamma.
`robvar.gamma`	robust variance for gamma.
`D2linv`	inverse of the derivative of the score function from marginal model.
`score`	value of score for final estimates.
`theta`	estimate of Gamma variance for frailty.
`var.theta`	estimate of variance of theta.
`SthetaInv`	inverse of derivative of score of theta.
`theta.score`	score for theta parameters.

Author(s)

Thomas Scheike

References

Glidden (2000), A Two-Stage estimator of the dependence parameter for the Clayton Oakes model.

Martinussen and Scheike, Dynamic Regression Models for Survival Data, Springer (2006).

Examples

```
library(timereg)
data(diabetes)
# Marginal Cox model  with treat as covariate
```

```
marg <- cox.aalen(Surv(time,status)~prop(treat)+prop(adult)+
  cluster(id),data=diabetes,resample.iid=1)
fit<-two.stage(marg,data=diabetes,theta=1.0,Nit=40)
summary(fit)

# using coxph and giving clusters, but SE wittout cox uncetainty
margph <- coxph(Surv(time,status)~treat,data=diabetes)
fit<-two.stage(margph,data=diabetes,theta=1.0,Nit=40,clusters=diabetes$id)


# Stratification after adult
theta.des<-model.matrix(~-1+factor(adult),diabetes);
des.t<-model.matrix(~-1+factor(treat),diabetes);
design.treat<-cbind(des.t[,-1]*(diabetes$adult==1),
                    des.t[,-1]*(diabetes$adult==2))

# test for common baselines included here
marg1<-cox.aalen(Surv(time,status)~-1+factor(adult)+prop(design.treat)+cluster(id),
 data=diabetes,resample.iid=1,Nit=50)

fit.s<-two.stage(marg1,data=diabetes,Nit=40,theta=1,theta.des=theta.des)
summary(fit.s)

# with common baselines  and common treatment effect (although test reject this)
fit.s2<-two.stage(marg,data=diabetes,Nit=40,theta=1,theta.des=theta.des)
summary(fit.s2)

# test for same variance among the two strata
theta.des<-model.matrix(~factor(adult),diabetes);
fit.s3<-two.stage(marg,data=diabetes,Nit=40,theta=1,theta.des=theta.des)
summary(fit.s3)

# to fit model without covariates, use beta.fixed=1 and prop or aalen function
marg <- aalen(Surv(time,status)~+1+cluster(id),
 data=diabetes,resample.iid=1,n.sim=0)
fita<-two.stage(marg,data=diabetes,theta=0.95,detail=0)
summary(fita)

# same model but se's without variation from marginal model to speed up computations
marg <- aalen(Surv(time,status) ~+1+cluster(id),data=diabetes,
      resample.iid=0,n.sim=0)
fit<-two.stage(marg,data=diabetes,theta=0.95,detail=0)
summary(fit)

# same model but se's now with fewer time-points for approx of iid decomp of marginal
# model to speed up computations
marg <- cox.aalen(Surv(time,status) ~+prop(treat)+cluster(id),data=diabetes,
      resample.iid=1,n.sim=0,max.timepoint.sim=5,beta.fixed=1,beta=0)
fit<-two.stage(marg,data=diabetes,theta=0.95,detail=0)
summary(fit)
```

`wald.test` *Makes wald test*

Description

Makes wald test, either by contrast matrix or testing components to 0. Can also specify the regression coefficients and the variance matrix. Also makes confidence intervals of the defined contrasts. Reads coefficientes and variances from timereg and coxph objects.

Usage

```
wald.test(object = NULL, coef = NULL, Sigma = NULL, contrast,
  coef.null = NULL, null = NULL, print.coef = TRUE, alpha = 0.05)
```

Arguments

`object`	timereg object
`coef`	estimates from some model
`Sigma`	variance of estimates
`contrast`	contrast matrix for testing
`coef.null`	which indeces to test to 0
`null`	mean of null, 0 by default
`print.coef`	print the coefficients of the linear combinations.
`alpha`	significance level for CI for linear combinations of coefficients.

Author(s)

Thomas Scheike

Examples

```
data(sTRACE)
# Fits Cox model
out<-cox.aalen(Surv(time,status==9)~prop(age)+prop(sex)+
prop(vf)+prop(chf)+prop(diabetes),data=sTRACE,n.sim=0)

wald.test(out,coef.null=c(1,2,3))
### test age=sex   vf=chf
wald.test(out,contrast=rbind(c(1,-1,0,0,0),c(0,0,1,-1,0)))

### now same with direct specifation of estimates and variance
wald.test(coef=out$gamma,Sigma=out$var.gamma,coef.null=c(1,2,3))
wald.test(coef=out$gamma,Sigma=out$robvar.gamma,coef.null=c(1,2,3))
### test age=sex   vf=chf
wald.test(coef=out$gamma,Sigma=out$var.gamma,
  contrast=rbind(c(1,-1,0,0,0),c(0,0,1,-1,0)))
```