Gprop.odds	*Fit Generalized Semiparametric Proportional Odds 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.

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.

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.
`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.
`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.
`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

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

```
library(survival)
data(sTRACE)
# Fits Proportional odds model with stratified baseline
age.c<-scale(sTRACE$age,scale=FALSE);
out<-Gprop.odds(Surv(time,status==9)~diabetes+prop(age.c)+prop(chf)+
prop(sex)+prop(vf),sTRACE,max.time=7,n.sim=500)
summary(out)
par(mfrow=c(2,3))
plot(out,sim.ci=2); plot(out,score=1)

# Fits Proportional odds model with baseline on additive form
# thus giving odds-ratio of dyings for vf and diabetes
out<-Gprop.odds(Surv(time,status==9)~vf+diabetes+prop(age.c)+prop(chf)+
prop(sex),sTRACE,max.time=7,n.sim=500)
summary(out)
par(mfrow=c(2,3))
plot(out,sim.ci=2); plot(out,score=1)
```

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)
```

`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.

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.

Usage

```
aalen(formula,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)
```

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.
`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.
`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.
`residuals`	to returns residuals that can be used for model validation in the function cum.residuals
`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

Details

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

```
library(survival)
data(sTRACE)
# Fits Aalen model
out<-aalen(Surv(time,status==9)~age+sex+diabetes+chf+vf,
sTRACE,max.time=7,n.sim=500)

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=500)

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

`aalen.test` *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.

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.

Usage

```
aalen.test(formula,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,weights=0,offsets=0,fix.gam=0,pseudo.score=0,
approx = "dt", gamma = 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.
`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.
`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.

`residuals` to returns residuals that can be used for model validation in the function cum.residuals

`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.

`offsets` fixed intensity only works for semiparametric, model but fully non-parametric model can be fitted by using fix.gam option, see also example.

`weights` weights for estimation.

`fix.gam` keep gamma parameter fixed.

`pseudo.score` number of simulation for pseudo-score test. If pseudo.score=0 it is not computed. The pseudo-score test for fixed effects appears to be superior to other tests computed.

The semiparametric model is given as

$$\lambda(t) = Y_i(t)(X_i^t \beta(t) + Z_i^T \gamma)$$

then the pseudo-score test is based on considering the test-process

$$\tilde{U}(t) = \int_0^t Z^T H dN - \int_0^t Z^T H Z dt (\int_0^\tau Z^T H Z dt)^{-1} \int_0^t Z^T H dN$$

that can be motivated as a pseudo-score test, with notation as in Martinussen and Scheike (2006), see references.

`approx` describes how the integrals are approximated, "dt" is default and uses a piececonstant approximation for all time points both censoring and event tiems, "death-times" uses only the event times for the approximation.

`gamma` values for potential fixed gamma coefficients.

Details

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` Two components : first component "likelihood" equivalent to partial likelihood for Cox's model, second component sum of least squares of increments.

`intZHZ` computes the integral int ZHZ dt.

`intZHdN` computes the integral int ZH dN.

`obs.pscore` pseudo score test value.

`pscore.iid` iid decomposition of pseudo-score test.

`intZHZt` computes the integral int ZHZ dt over time.

`pstest.pval` pseudo-score p-value.

`sup.pscore` simulated supremum values for pseudo-score test.

Author(s)

Thomas Scheike

References

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

Martinussen and Scheike (200.), Tests for time-varying effects within Aalen's additive hazards model,

Examples

```
library(survival)
data(mela.pop)
# Fits Aalen model  with offsets
dummy<-rnorm(nrow(mela.pop));

out<-aalen.test(Surv(start,stop,status==1)~age+sex+const(dummy),
mela.pop,max.time=7,n.sim=500,offsets=mela.pop$rate,id=mela.pop$id,
fix.gam=1)

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

# Fits semi-parametric additive hazards model  with offsets
out<-aalen.test(Surv(start,stop,status==1)~age+const(sex),
mela.pop,max.time=7,n.sim=500,offsets=mela.pop$rate,id=mela.pop$id)

summary(out)
plot(out)

#################################################################
# Computes pseudo.score tests for fixed effects
data(sTRACE)
out<-aalen.test(Surv(time,status==9)~const(age)+const(sex)+const(diabetes)+
chf+vf,sTRACE,max.time=7,n.sim=500,pseudo.score=500)

summary(out)
```

`lasso.boost` *Fits LASSO model for additive hazards model*

Description

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

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

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

Usage

```
lasso.boost(D,d,lambda,max.it=10000,beta=0,detail=0)
```

Arguments

`D`	defined above
`d`	defined above
`lambda`	l1 regularization
`max.it`	number of steps in l1 boosting algorithm
`detail`	prints details
`beta`	starting value for algorithm.

Details

lasso.boost is the boosting algorithm

lasso.add.hazard computes the exact solution using the quadprog package.

Value

returns a list with the following arguments:

`beta`	regression coefficients
`L`	value of the fitting criterion
`l1`	sum of the absolute value of the coeffcients

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).

Kim and Kim (2004), Gradient LASSO for feature selection, In Proceedings of the 21st International Conference on Machine Learning.

Examples

```
## makes data for pbc complete case
data(pbc)
pbc$time<-pbc$time+runif(418); pbc$time<-pbc$time/365
pbc[pbc==-9]<-NA; rsum<-apply(pbc,1,sum);
pbc<-pbc[!is.na(rsum),]
covs<-as.matrix(pbc[,-c(10,17)])
lcovs<-covs;
lcovs[,c(3,5,6,10,11,13,17,18)]<-
log(lcovs[,c(3,5,6,10,11,13,17,18)])
```

```
## computes the matrices needed for the least squares
## criterion
out2<-aalen.test(Surv(time,status)~const(lcovs),pbc,robust=0,n.sim=0)
S=out2$intZHZ; s=out2$intZHdN;

## lambda=0.39
out.pbc<-lasso.boost(S,s,0.39,max.it=20000)

## exact solution, slow sometimes !!
# library(quadprog)
# out.ex<-lasso.add.hazard(S,s,0.39,0,1,max.it=20000)
# print(round(cbind(out.pbc$beta,out.ex$sol$solution),4))
```

`additive.pls` *Fits PLS for additive hazards model*

Description

Fits the partial least squares estimator for the additive risk model. Time dependent variables and counting process data (multiple events per subject) are possible.

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

Covariates $Z_1, ..., Z_p$, fixed covariates $F(t)$. Algorithm : 1) For pls components $X_1, ..., X_K$, fits

$$\lambda_0(t) + \alpha^T(t)F(t) + \sum_{j=1}^{K} X_j(t)\gamma_j(t) + Z_i(t)\beta_i^{K+1}$$

for $i = 1, ..., p$ 2) compute new pls components $X_{K+1} = \sum \beta_i^{K+1} Z_i(t)$ and iterate.

Usage

```
additive.pls(formula = formula(data), data =
sys.parent(), start.time=0,max.time=NULL,id=NULL, pls.dim=1,
scale=FALSE, weighted.pls=0, constant=1)
```

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.

The const terms are kept as fixed covariates that are not involved in the pls variable reduction. This may covariates that are know to be of clinical importance.

`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.
`pls.dim`	number of pls components
`scale`	to center and scale the covariates
`weighted.pls`	Gartwaith weights if 1.
`constant`	When 0 the coefficient $\gamma_j(t)$ does not depend on time.

Details

const() specifies the $F(t)$ covariates in the above model.

Value

returns an object with the following arguments:

`baseline`	baseline of the semparametric additive risk model
`pls.comp`	the pls components, i.e. the covariates multipled on the beta coefficients.
`beta`	risk regression coefficients related to the pls components.
`beta.pls`	regression coefficients that defines the pls componets.
`tbeta.pls`	the combined regression coefficients from the pls components and the regression coefficients, these leads to risk predictions when applied to new covariates.

Author(s)

Thomas Scheike

References

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

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

Examples

```
library(survival)
data(sTRACE)

### 2 PLS components for the 5 covariates
### based on Gui-Li approach with constant effects
out<-additive.pls(Surv(time,status==9)~age+sex+
vf+diabetes+chf,sTRACE,max.time=7,pls.dim=2,constant=1)

### 2 PLS components for the 5 covariates
### based on Gui-Li approach with timevarying effects
out<-additive.pls(Surv(time,status==9)~age+sex+
```

```
vf+diabetes+chf,sTRACE,max.time=7,pls.dim=2,constant=0)

### 2 PLS components for the 3 covariates
### age and sex are fixed covariates
out<-additive.pls(Surv(time,status==9)~const(age)+const(sex)+
vf+diabetes+chf,sTRACE,max.time=7,pls.dim=2)
```

`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

What source ?

References

Examples

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

`cd4`	*The multicenter AIDS cohort study*

Description

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)
```

Description

Fits a semiparametric model for the cause-specific quantities :

$$P(T \leq t, cause = 1|x) = P_1(t,x) = 1 - \exp(-g(t,x))$$

for the probability of dying from cause 1 in a situation with competing causes of death.

The model is considered in two situations :

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

the additive situation (with additive subdistribution hazard) and the proportional setting that includes the Fine & Gray (FG) model and some extensions

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

The FG model is obtained when $x = 1$. Where p is 1 for the additive model and 0 for the proportional model. In general p may be powers of the same length as z.

Usage

```
comp.risk(formula,data=sys.parent(),cause,times,Nit=50,
gamma=0,n.sim=500,weighted=0,model="additive",
cens.code=0,causeS=1,clusters=NULL,
detail=0,interval=0.01,resample.iid=1,cens.model="KM",time.pow=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. The status indicator is not important here.
`data`	a data.frame with the variables.
`cause`	specifies the causes related to the death times, the value 0 is the censoring value.
`times`	specifies the times at which the estimator is considered. This is typically all cause "1" jump times.
`Nit`	number of iterations for Newton-Raphson algorithm.
`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" or "prop"ortional.

`cens.code` specificies which cause that is censoring

`causeS` specificies which cause we consider.

`clusters` specificies a possible cluster structure.

`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"

`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.

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

Author(s)

Thomas Scheike

References

Scheike, Zhang and Gerds (2007), Predicting cumulative incidence probability by direct binomial regression, Biometrika, to appear.

Scheike and Zhang (2007), Flexible competing risks regression modelling and goodness of fit, work in progress.

Examples

```
data(bmt);
times<-bmt$time[bmt$cause==1];

add<-comp.risk(Surv(time,cause>0)~platelet+age+tcell,bmt,
bmt$cause,times[-1],causeS=1,resample.iid=1)
summary(add)

par(mfrow=c(2,4))
plot(add); plot(add,score=1)

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)

## fits additive model with some constant effects
add.sem<-comp.risk(Surv(time,cause>0)~
const(platelet)+const(age)+const(tcell),bmt,
bmt$cause,times[-1],causeS=1,resample.iid=1)
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(Surv(time,cause>0)~
const(platelet)+const(age)+const(tcell),bmt,
bmt$cause,times[-1],causeS=1,resample.iid=1,model="prop")
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(Surv(time,cause>0)~platelet+age+const(tcell),
bmt,bmt$cause,times[-1],causeS=1,resample.iid=1,model="prop")
```

```
summary(fg.npar);

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

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

`const` *Identifies parametric terms of model*

Description

Specifies which of the regressors that have constant effect.

Author(s)

Thomas Scheike

`cox` *Identifies proportional excess terms of model*

Description

Specifies which of the regressors that lead to proportional excess hazard

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.

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

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

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.

Usage

```
cox.aalen(formula=formula(data),data=sys.parent(),beta=0,Nit=10,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=0,weights=NULL,rate.sim=1,
beta.fixed=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.
`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.
`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.
`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).
`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.
`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.
`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

Details

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.
- `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(survival)
data(sTRACE)
# Fits Cox model
out<-cox.aalen(Surv(time,status==9)~prop(age)+prop(sex)+
prop(vf)+prop(chf)+prop(diabetes),sTRACE,max.time=7,n.sim=500)

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

# Fits Cox-Aalen model
out<-cox.aalen(Surv(time,status==9)~prop(age)+prop(sex)+
vf+chf+prop(diabetes),sTRACE,max.time=7,n.sim=500)

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

`csl` *CSL liver chirrosis data*

Description

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=0,
n.sim=500,weighted.test=1,start.design=1)
```

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.
`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.
`cum.resid`	to compute residuals versus each of the continuous covariates in the model.
`start.design`	if '1' the groupings specified in modelmatrix changes over time, i.e. in the case with time-dependent covariates.

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

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

# constructs and simulates cumulative residuals versus age groups
fit.mg<-cum.residuals(fit,sTRACE,n.sim=200,
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,sTRACE,cum.resid=1,n.sim=200)

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 :

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

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

Usage

```
dynreg(formula,data=sys.parent(),aalenmod,bandwidth=0.5,id=NULL,
bhat=NULL,start.time=0,max.time=NULL,n.sim=500,residuals=0,
meansub=1,weighted.test=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.
`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.
`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.
`residuals`	to returns residuals that can be used for model validation in the function 'cum.residuals'.

`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.

`bhat` initial value for estimates. If NULL local linear estimate is computed.

`meansub` if '1' then the mean of the responses is subtracted before the estimation is carried out.

Details

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.

`residuals` list of 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.

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

```
library(survival)
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=200,bandwidth=0.3,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),csl,
Surv(lt,rt,indi.m)~+1,start.time=0,max.time=3,id=csl$id,
n.sim=200,bandwidth=0.3,meansub=0)
summary(outS)
```

---

pval *For internal use*

Description

for internal use

Author(s)

Thomas Scheike

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

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

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

Usage

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

Arguments

D	defined above
d	defined above
dim	number of pls dimensions

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(pbc)
pbc$time<-pbc$time+runif(418); pbc$time<-pbc$time/365
pbc[pbc==-9]<-NA; rsum<-apply(pbc,1,sum);
pbc<-pbc[!is.na(rsum),]
covs<-as.matrix(pbc[,-c(10,17)])
lcovs<-covs;
lcovs[,c(3,5,6,10,11,13,17,18)]<-
```

```
log(lcovs[,c(3,5,6,10,11,13,17,18)])

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

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

---

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

Description

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 Ø, 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 Ø, 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)
```

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

Description

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

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

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 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)
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=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 "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`	robust standard errors are used to estimate standard error of estimate, otherwise martingale based standard errors 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

```
library(survival)
data(sTRACE)
# Fits Aalen model
out<-aalen(Surv(time,status==9)~age+sex+diabetes+chf+vf,
sTRACE,max.time=7,n.sim=500)

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

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,xlab="Time",
ylab ="Cumulative Residuals",ylim=NULL,score=0,...)
```

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.
`xlab`	label for x-axis. Default is "Time".
`ylab`	label for y-axis. Default is "Cumulative 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.
`...`	unused arguments - for S3 compatibility

Author(s)

Thomas Scheike

References

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

Examples

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

# constructs and simulates cumulative residuals versus age groups
out.mg<-cum.residuals(out,sTRACE,model.matrix(~-1+factor(cut(age,4)),sTRACE))

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

# cumulative residuals against covariates Lin, Wei, Ying style
out.mg<-cum.residuals(out,sTRACE,cum.resid=1)
par(mfrow=c(2,4))
plot(out.mg,score=2)
```

`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

```
library(survival)
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=500,bandwidth=0.3,meansub=0)
```

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

---

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

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 'comprisk':
predict(object,newdata=NULL,X=NULL,Z=NULL,
n.sim=500, uniform=TRUE,se=TRUE,alpha=0.05,...)
```

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.
`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.
`...`	unused arguments - for S3 compatability

Author(s)

Thomas Scheike, Jeremy Silver

References

Scheike, Zhang and Gerds (2007), Predicting cumulative incidence probability by direct binomial regression, Biometrika, to appear.

Scheike and Zhang (2007), Flexible competing risks regression modelling and goodness of fit, work in progress.

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

Examples

```
data(bmt);
times<-bmt$time[bmt$cause==1];

add<-comp.risk(Surv(time,cause>0)~platelet+age+tcell,bmt,
bmt$cause,times[-1],causeS=1,resample.iid=1)
summary(add)

par(mfrow=c(2,4))
plot(add); plot(add,score=1)

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=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.

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

out<-predict(out,X=rbind(c(1,0,0),c(1,1,0)),
Z=rbind(c(55,0,1),c(60,1,1)))
par(mfrow=c(2,2))
plot(out,multiple=1,se=0,uniform=0,col=1:2,lty=1:2)
plot(out,multiple=0,se=1,uniform=1,col=1:2)

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

out<-predict(out,X=rbind(c(1,0,0),c(1,1,0)),
Z=rbind(c(55,0,1),c(60,1,1)))
par(mfrow=c(2,2))
plot(out,multiple=1,se=0,uniform=0,col=1:2,lty=1:2)
plot(out,multiple=0,se=1,uniform=1,col=1:2)
```

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

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

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.

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

Usage

```
prop.excess(formula=formula(data),data=sys.parent(),excess=1
,tol=0.0001,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 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

```
library(survival)
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=2000)
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.

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

Usage

```
prop.odds(formula,data=sys.parent(),beta=0,Nit=10,
detail=0,start.time=0,max.time=NULL,id=NULL,n.sim=500,weighted.test=0,
profile=1,sym=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.
`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.
`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.
`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.

Details

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

```
library(survival)
data(sTRACE)
# Fits Proportional odds model
out<-prop.odds(Surv(time,status==9)~age+diabetes+chf+vf+sex,
sTRACE,max.time=7,n.sim=500)

summary(out)

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

`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.

Returns parameter estimates and their standard errors.

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

Author(s)

Thomas Scheike

References

Martinussen and Scheike,

Examples

```
library(survival)
data(sTRACE)
# Fits Aalen model
out<-aalen(Surv(time,status==9)~age+sex+diabetes+chf+vf,
sTRACE,max.time=7,n.sim=500)

summary(out)
```

`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

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

# constructs and simulates cumulative residuals versus age groups
```

```
# and versus covariates of model
out.mg<-cum.residuals(out,sTRACE,
modelmatrix=model.matrix(~-1+factor(cut(age,4)),sTRACE),cum.resid=1)

summary(out.mg)
```

---

mylars *Fits LASSO model for additive hazards model by Lars algorithm*

Description

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

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

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

This is equivalent to an appropriate normal least squares problem on least squares data.

Usage

```
surv.lars(time,status,z,l1.weights=NULL,...)
```

Arguments

`time`	covariates for artificial least squares data.
`status`	responses for artificial least squares data.
`z`	covariatesspecifies the weights for the L1 penalty.
`l1.weights`	specifies the weights for the L1 penalty.
`...`	unused arguments - for S3 compatibility

Details

Modified version of standard lars program in the LARS package. Essentially all scalings are removed from the lars program.

Value

see lars.

Author(s)

Thomas Scheike

References

Martinussen and Scheike, Model selection for the the additive risks hazards model, submitted.

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

LARS, Efron et al.

Examples

```
## makes data for pbc complete case
data(pbc)
pbc$time<-pbc$time+runif(418); pbc$time<-pbc$time/365
pbc[pbc==-9]<-NA; rsum<-apply(pbc,1,sum);
pbc<-pbc[!is.na(rsum),]
covs<-as.matrix(pbc[,-c(10,17)])

## computes the matrices needed for the least squares
## criterion
#out<-des.aalen(Surv(time,status)~const(covs),pbc)
#x<-as.matrix(out[,1:17]); y<-out$Y

## lasso for survival data
# must load lars library for it to work
#fit<-surv.lars(pbc$time,pbc$status,covs);
#fit<-mylars(x,y)
#cv<-surv.lars.cv(time,status,covs)
#myplot.lars(fit)

#cv.frac<-cv$cv.frac
#beta<-mycoef.lars(fit,s=cv.frac,mode="fraction");

#full<-fit$beta[18,] # least squares solution

## adaptive lasso for survival data
#fit<-surv.lars(time,status,covs,l1.weights=1/abs(full));
#cva<-surv.lars.cv(time,status,covs,l1.weights=1/abs(full))

#cva.frac<-cva$cv.frac
#abeta<-mycoef.lars(afit,s=cv.frac,mode="fraction");
#myplot.lars(afit)
```

`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.

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.

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.

`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.

`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.

`residuals` to returns residuals that can be used for model validation in the function cum.residuals

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

`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 $\lambda_0(t)$.

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

Details

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

```
library(survival)
data(sTRACE)
# Fits time-varying Cox model
out<-timecox(Surv(time/365,status==9)~age+sex+diabetes+chf+vf,
sTRACE,max.time=7,n.sim=500)

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,sTRACE,max.time=7,n.sim=500)

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

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

Description

Fit Clayton-Oakes-Glidden Two-Stage model with Cox-Aalen marginals and Gamma distributed dependence frailties

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

The marginal baselines are on the Cox-Aalen form

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

The model thus contains the Cox's regression model 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.

Usage

```
two.stage(formula=formula(data),data=sys.parent(),beta=0,Nit=10,
detail=0, start.time=0,max.time=NULL,id=NULL,clusters=NULL,
robust=1,rate.sim=1,beta.fixed=0,theta=NULL,inverse=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.
- `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.
- `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.
- `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 keeping beta in the Cox-Aalen model fixed.
- `theta` starting values for the frailty variance (default=0.1).
- `inverse` inverse=0, indicates that the parameter is the variance, in contrast to inverse=1 where the inverse variance is used as the parameter.

Details

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.

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.
`score`	value of score for final estimates.
`theta`	estimate of Gamma variance for frailty.
`var.theta`	estimate of variance of theta.

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(survival)
data(diabetes)
# Marginal Cox model  with treat as covariate

fit<-two.stage(Surv(time,status) ~ prop(treat) + cluster(id),
diabetes,theta=0.95,detail=0,inverse=0)
summary(fit)

fit<-two.stage(Surv(time,status) ~ prop(treat) + cluster(id),
diabetes,theta=0.95,detail=0,inverse=1)
summary(fit)

# to fit model without covariates, fix beta=0, but still need prop term !
fit<-two.stage(Surv(time,status) ~ prop(treat) + cluster(id),
diabetes,theta=0.95,detail=0,beta.fixed=1,inverse=0)
summary(fit)
```

`two.stage.reg` *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.

The key here is that the variance is allowed to depend on covariates fixed within clusters

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

The simple univariate model can also be fitted by the two.stage function.

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.

Usage

```
two.stage.reg(formula=formula(data),data=sys.parent(),beta=0,Nit=10,
detail=0,start.time=0,max.time=NULL,id=NULL,clusters=NULL,
robust=1,rate.sim=1,beta.fixed=0,theta=NULL,theta.des=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.
`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.
`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.
`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 keeping beta in the Cox-Aalen model fixed.
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.

Details

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.

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.
score	value of score for final estimates.
theta	estimate of Gamma variance for frailty.
var.theta	estimate of variance of theta.
S.theta	estimate of derivative of score of theta.

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(survival)
data(diabetes)
# Marginal Cox model  with treat as covariate
fit<-two.stage.reg(Surv(time,status) ~ prop(treat) + cluster(id),
diabetes,Nit=40,theta=1)
summary(fit)

# 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))
fit.s<-two.stage.reg(Surv(time,status) ~
-1+factor(adult)+prop(design.treat)+cluster(id),
diabetes,Nit=40,theta=1,theta.des=theta.des)
summary(fit.s)

# test for common baselines
fit.s1<-cox.aalen(Surv(time,status) ~
factor(adult)+prop(design.treat)+cluster(id),diabetes)
summary(fit.s1)

# with common baselines  and common treatment effect (although test reject this)
fit.s2<-two.stage.reg(Surv(time,status) ~+1+prop(treat) + cluster(id),
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.reg(Surv(time,status) ~+1+prop(treat)+cluster(id),
diabetes,Nit=40,theta=1,theta.des=theta.des)
summary(fit.s3)
```