Package ‘timereg’

May 21, 2017

Title Flexible Regression Models for Survival Data
Version 1.9.1
Date 2017-03-25
Author Thomas Scheike with contributions from Torben Martinussen, Jeremy Silver and Klaus Holst
Maintainer Thomas Scheike <ts@biostat.ku.dk>
LazyLoad yes
URL https://github.com/scheike/timereg.git
Depends R (>= 2.15), survival,
Imports lava, numDeriv, stats, graphics, grDevices, utils
License GPL (>= 2)
RoxygenNote 6.0.1
NeedsCompilation yes
Repository CRAN
Date/Publication 2017-05-21 07:51:40 UTC

R topics documented:

aalen .................................................. 2
bmt .................................................. 5
cd4 .................................................. 6
comp.risk .......................................... 7
const ............................................... 13
cox ................................................ 13
cox.aalen ........................................ 14
Fit additive hazards model

Fits both the additive hazards model of Aalen and the semi-parametric additive hazards model of McKeage 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 time-varying 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 time-varying 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


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. Platelet 1: more than $10^9$ 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**

```r
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.
comp.risk

Source


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  Competing 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, Nlt = 50, clusters = NULL, est = NULL, fix.gamma = 0, gamma = 0, n.sim = 0, weighted = 0, model = "lg", detail = 0, interval = 0.01, resample.id = 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.
truncNp truncation weight for delayed entry, \( P(T > \text{entry.time} | Z_i) \), 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
\[
(D_\beta P_1) w(t) (Y(t)/G_c(t) - P_1(t, X)) \text{and}
\]
monotone 1 uses
\[
w(t)(Y(t)/G_c(t) - P_1(t, X)) \text{and}
\]
step step size for Fisher-Scoring algorithm.

Details
We consider the following models: 1) the additive model where 
\[
h(x) = 1 - \exp(-x)
\]
and
\[
g(t, x, z) = x^T A(t) + (\text{diag}(t^p) z)^T \beta
\]
2) the proportional setting that includes the Fine & Gray (FG) "prop" model and some extensions where 
\[
h(x) = 1 - \exp(-\exp(x))
\]
and
\[
g(t, x, z) = (x^T A(t)) \exp((\text{diag}(t^p) z)^T \beta)
\]
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
\[
g(t, x, z) = (x^T A(t)) \exp((\text{diag}(t^p) z)^T \beta)
\]
The FG model is obtained when \( x = 1 \).
3) a "logistic" model where 
\[
h(x) = \exp(x)/(1 + \exp(x))
\]
and
\[
g(t, x, z) = x^T A(t) + (\text{diag}(t^p) z)^T \beta
\]
The "logistic2" is
\[
P_1(t, x, z) = x^T A(t) \exp((\text{diag}(t^p) z)^T \beta)/(1 + x^T A(t) \exp((\text{diag}(t^p) z)^T \beta))
\]
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
\[
g(t, x, z) = x^T A(t) + (\text{diag}(t^p) z)^T \beta
\]
The "rcif2"
\[
P_1(t, x, z) = (x^T A(t)) \exp((\text{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

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

the new code is

```r
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 time-varying 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.


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.id=1,n.sim=100,model="additive")
###

addclust<-comp.risk(Event(time,cause)+1+cluster(clust),data=bmt,cause=1,
resample.id=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.id=1,n.sim=100,model="additive")
summary(add)

par(mfrow=c(2,4))
plot(add);
### 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.id=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.id=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.id=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.id=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.id=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.comprisk(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)
### const

Identifies parametric terms of model

**Description**

Specifies which of the regressors that have constant effect.

**Usage**

```r
cnst(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**

```r
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**

```r
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, covariation = 0, resample.id = 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 return 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 residuals (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. decomposition. 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 derivative 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 oppresses 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 constant 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 appropriately (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 time varying 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 sandwich 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


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)
Description

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

Usage

```r
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 parameters (lava).

Value

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

- `iid`: iid decomposition.
- `coef`: missing data estimtes 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.
cs1

Author(s)

Thomas Scheike

References

Paik et al.

Examples

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

csl

CSL liver cirrhosis data

Description

Survival status for the liver cirrhosis 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


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.
obt.testBeq0 observed absolute value of supremum of cumulative components scaled with the variance.
**cum.residuals**

- **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**


**Examples**

```r
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,
                      model.matrix=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)
```
The Diabetic Retinopathy Data

Description

The data was collected to test a laser treatment for delaying blindness in patients with diabetic retinopathy. The subset of 197 patients 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.
- **trt.eye**: 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


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

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

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.

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 obtained by local linear regression:

\[
\hat{B}(t) = \int_0^t \hat{\beta}(s)ds + 
\int_0^t X^{-1}(Diag(z) - Diag(X^T(s)\hat{\beta}(s)))dp(ds \times dz),
\]

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

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

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).

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

where \( f \) is chosen as the matrix

\[ f(s) = \text{Diag}(X^T(s)\hat{\beta}(s))(I - X_{\alpha}(s)X^-_{\alpha}(s)), \]

where \( X_{\alpha} \) 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.test.beq0 observed absolute value of supremum of cumulative components scaled with the variance.

pval.test.beq0 p-value for covariate effects based on supremum test.

sim.test.beq0 resampled supremum values.

obs.test.beqC 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

Examples

```r
### this runs slowly and is therefore 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

\textit{Event history object}

\textbf{Description}

Constructur for Event History objects

\textbf{Usage}

\begin{verbatim}
Event(time, time2 = TRUE, cause = NULL, cens.code = 0, ...)
\end{verbatim}

\textbf{Arguments}

\begin{itemize}
  \item \texttt{time} \hspace{1cm} Time
  \item \texttt{time2} \hspace{1cm} Time 2
  \item \texttt{cause} \hspace{1cm} Cause
  \item \texttt{cens.code} \hspace{1cm} Censoring code (default 0)
  \item \ldots \hspace{1cm} Additional arguments
\end{itemize}

\textbf{Details}

... content for details

\textbf{Value}

Object of class Event (a matrix)

\textbf{Author(s)}

Klaus K. Holst and Thomas Scheike

\textbf{Examples}

\begin{verbatim}
t1 <- 1:10
t2 <- t1+runif(10)
ca <- rbinom(10,2,0.4)
(x <- Event(t1,t2,ca))
\end{verbatim}
event.split

Description

constructs 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,.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)
```
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)

---

**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.

**Usage**

```r
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 \( \beta \) 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.testBeq\( \emptyset \) observed absolute value of supremum of cumulative components scaled with the variance.
pval.testBeq\( \emptyset \) p-value for covariate effects based on supremum test.
sim.testBeq\( \emptyset \) 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).

Examples

data(sTRACE)

### runs slowly and is therefore donttest
data(sTRACE)
# Fits Proportional odds model with stratified baseline
age.cc<-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)

# 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), data=sTRACE, max.time=7, n.sim=50)
summary(out)
par(mfrow=c(2,3))
plot(out,sim.ci=2); plot(out,score=1)
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 \( D = \int ZH Z dt \) and \( d = \int ZH dN \).

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.

Examples

```r
## 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$intZH; s=out$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**

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.
- 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

Examples
- data(melanoma)
- names(melanoma)

my pbc

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

monotone 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")
**Description**

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

**Usage**

```r
pc.hazard(cumhazard, rr, cum.hazard = TRUE)
```

**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 simulations.
- `cum.hazard`: specifies whether input is cumulative hazard or rates.

**Author(s)**

Thomas Scheike

**Examples**

```r
rates <- c(0, 0.01, 0.05, 0.01, 0.04)
b breaks <- c(0, 10, 20, 30, 40)
haz <- cbind(breaks, rates)
n <- 1000
X <- rbinom(n, 1, 0.5)
eta <- 0.2
rrcox <- exp(X * beta)
cumhaz <- cumsum(c(0, diff(breaks) * rates[-1]))
cumhaz <- cbind(breaks, cumhaz)

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

par(mfrow=c(1, 2))
s <- aalen(Surv(time, status) ~ 1, data = pctime, robust = 0)
plot(ss)
lines(cumhaz, col = 2, lwd = 2)

pctimecox <- pc.hazard(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)
```
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.
pe.sasieni

Nit  number of iterations.
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

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")
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

```r
# 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` specific colors of different components of plot, in order: c(estimate,pointwise.ci,robust.ci,hw.ci,sim.ci)
- `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
plot.cum.residuals

Author(s)

Thomas Scheike

References


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.re...siduals'
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


Examples

# see cum.residuals for examples
**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**

```r
## 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
predict.timereg

Author(s)
Thomas Scheike

References

Examples

```r
### runs slowly and therefore dont test
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)
```

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

```r
## 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 specifies the significance level which 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.
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$so[,1:5]); head(pout$se.so[,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(se)+
const(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),c(1,1,0)),
Z=rbind(c(05,0,1),c(00,1,1)))
head(pout$so[,1:5]); head(pout$se.so[,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(se)+
prop(diabetes)+chf+vf,
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 $1/(H(T_i) \ast G_c(\min(T_i, \tau)))$ 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  species 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


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
############################################################
### (if
### 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)
print.aalen

### Summary

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

### with truncation

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

```r

# Regression

### with truncation correction

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

### with only right-censoring, standard call

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

---

**Description**

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

**Usage**

```r
## 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 parameter 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 score process and 50 resampled score processes (under model). List with 51 elements.
Author(s)
Torben Martinussen

References

Examples

```r
# 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 Odds Model*

Description

Fits a semiparametric proportional odds model:

\[
\text{logit}(1 - S_Z(t)) = \log(G(t)) + \beta^T Z
\]

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

```r
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.
prop.odds

**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**

prop.odds.subdist

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 Odds Model for the competing risks subdistribution

Description

Fits a semiparametric proportional odds model:

\[
\text{logit}(F_1(t; X, Z)) = \log(A(t)) + \beta^T Z
\]

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 time-varying 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 \( \beta \) 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 time-varying 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


Examples

```r
library(thimereg)
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)
```

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

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

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 compatibility.

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 cen-
         soring 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 $A_l(t) = \beta_l t^{\text{time.pow.test}(l)}$. Default is 1 for the additive model and 0 for the proportional model.

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

conv gives convergence criteria in terms of sum of absolute change of parameters of model

estimator specifies what 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 time-varying 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)=1)-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)=1)-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)=1)-factor(tcell)+factor(platelet), data=bmt, cause=1,
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.
restricted.residual.mean

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


Examples

```r
### this example runs slowly and is therefore donttest
data(sTRACE)
sTRACE$cage <- scale(sTRACE$age)
# Fits Cox model and aalen model
out1 <- cox.aalen(Surv(time, status==1)~prop(sex)+prop(diabetes)+prop(chf)+
                  prop(vf), data=sTRACE, max.timepoint.sim=NULL, resample.id=1)
outa1 <- aalen(Surv(time, status==1)~sex+diabetes+chf+vf,
data=sTRACE, resample.id=1)
coxrm <- restricted.residual.mean(out, tau=7,
   x=rbind(c(1,0,0,0,0),c(1,1,0,0,0),c(1,0,1,0,0),c(0,0,1,1,0),c(0,0,0,0,1)), 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+vfa","+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+vfa","+vf")
mm
```
Simulation of output from Cox model.

Description

Simulates data that looks like fit from Cox model. Censor data automatically for highest value of the break points.

Usage

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

Author(s)

Thomas Scheike

Examples

```r
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")])
###library(mets)
###dcor(sim1,-vf+chf+wmi)
###dcor(TRACE,-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)
```
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

```r
## 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

```r
### 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

```r
## 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

```r
# 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

```r
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 time-varying 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 return 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 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]. 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:
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

Examples

```r
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.
two.stage

Source

The TRACE study group.


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 defaults 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 resam-
ple.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

θ_k = Q_k^T ν

. 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

λ_{ki}(t) = Y_{ki}(t)(X_{ki}^T(t)α(t)) exp(Z_{ki}^Tβ)

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 marginals 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.
two.stage

| 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.                                     |
| StthetaInv  | 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.


**Examples**

```r
library(timerreg)
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 without cox uncertainty
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.id=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.id = 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.id = 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.id = 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 coefficients and variances from timereg and coxph objects.

**Usage**

```r
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.
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 specification 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)))
Index

*Topic datasets
  bmt, 5
cd4, 6
csl, 19
diabetes, 22
mela.pop, 32
melanoma, 33
mypbc, 33
trace, 68

*Topic survival
  aalen, 2
  comp.risk, 7
  const, 13
  cox, 13
  cox.aalen, 14
  cox.ipw, 18
  cum.residuals, 20
dynreg, 22
event.split, 27
Gprop.odds, 28
krylow.pls, 31
pava.pred, 34
pc.hazard, 35
pe.sasieni, 36
plot.aalen, 38
plot.cum.residuals, 39
plot.dynreg, 41
predict.timereg, 42
prep.comp.risk, 45
print.aalen, 47
prop, 48
prop.excess, 48
prop.odds, 50
prop.odds.subdist, 53
pval, 56
qcut, 57
res.mean, 57
restricted.residual.mean, 61
sim.cox, 63
summary.aalen, 64
summary.cum.residuals, 65
timecox, 65
two.stage, 69
wald.test, 72
[.Event (Event), 26
aalen, 2
aalen.des (pval), 56
aalen.des2 (pval), 56
aalenBase (pval), 56
aalenBaseC (pval), 56
additive.comps (pval), 56
as.character.Event (Event), 26
as.matrix.Event (Event), 26
bmt, 5
cause.pchazard.sim (pc.hazard), 35
cd4, 6
check.missing (pval), 56
cluster.index.timereg (pval), 56
coef.aalen (pval), 56
coef.comprisk (pval), 56
coef.cox.aalen (pval), 56
coef.cox.ipw (cox.ipw), 18
coef.cox.marg (pval), 56
coef.dynreg (pval), 56
coef.resmean (pval), 56
coef.timecox (pval), 56
coef.two.stage (pval), 56
coefBase (pval), 56
coefcox (pval), 56
comp.risk, 7
comprisk.ipw (pval), 56
const, 13
cox, 13
cox.aalen, 14
cox.aalenBase (pval), 56
cox.ipw, 18
INDEX

cox.marg (pval), 56
Cpred (pval), 56
csl, 19
Csmooth2B (pval), 56
CsmoothB (pval), 56
cum.residuals, 20
des.aalen (pval), 56
diabetes, 22
dynreg, 22
dynregBase (pval), 56
Event, 10, 26, 59
event.split, 27
faster.reshape (pval), 56
format.Event (Event), 26
Gprop.odds, 28
is.diag (pval), 56
kernel (pval), 56
kmplot (pval), 56
krylow.pls, 31
localTimeReg (pval), 56
mela.pop, 32
melanoma, 33
mypbc, 33
nameestimate (pval), 56
namematrix (pval), 56
names2formula (pval), 56
pava (pval), 56
pava.pred, 34
pc.hazard, 35
pchazard.sim (pc.hazard), 35
pe.sasieni, 36
percen (pval), 56
plot.aalen, 38
plot.comprisk (pval), 56
plot.cox.aalen (plot.aalen), 38
plot.cox.aalen2 (pval), 56
plot.cum.residuals, 39
plot.cums (pval), 56
plot.dynreg, 41
plot.predict (pval), 56
plot.prop.excess (plot.aalen), 38
plot.resmean (pval), 56
plot.restricted.residual.mean (pval), 56
plot.timecox (plot.aalen), 38
plot.two.stage (pval), 56
plotScore (pval), 56
pred.cum (pval), 56
pred.des (pval), 56
pred.stratKM (pval), 56
predict.aalen (predict.timereg), 42
predict.comprisk (predict.timereg), 42
predict.cox.aalen (predict.timereg), 42
predict.timereg, 42
predict.two.stage (pval), 56
predictpropodds (pval), 56
prep.comp.risk, 45
prep.glm.comprisk (pval), 56
print.aalen, 47
print.comprisk (print.aalen), 47
print.comprisk.ipw (pval), 56
print.cox.aalen (print.aalen), 47
print.cox.ipw (cox.ipw), 18
print.cox.marg (pval), 56
print.cum.residuals (print.aalen), 47
print.dynreg (print.aalen), 47
print.Event (Event), 26
print.pe.sasieni (pval), 56
print.predict.timereg (pval), 56
print.prop.excess (print.aalen), 47
print.resmean (pval), 56
print.timecox (print.aalen), 47
print.two.stage (pval), 56
prop, 48
prop.excess, 48
prop.excessBase (pval), 56
prop.odds, 50
prop.odds.gam (pval), 56
prop.odds.subdist, 53
prop.odds.subdist.ipw (pval), 56
pval, 56
qcut, 57
rbind.Event (Event), 26
read.design (pval), 56
read.design (pval), 56
read.surv (pval), 56
res.mean, 57
residualsTimereg (pval), 56
restricted.residual.mean, 61
risk.index (pval), 56
rm.missing (pval), 56

semiaalen (pval), 56
semicox (pval), 56
semiregBase (pval), 56
sim.cox, 63
sindex.prodlimit (pval), 56
slaaop (pval), 56
sTRACE (TRACE), 68
summary.aalen, 64
summary.comprisk (pval), 56
summary.cox.aalen (summary.aalen), 64
summary.cox.ipw (cox.ipw), 18
summary.cox.marg (pval), 56
summary.cum.residuals, 65
summary.dynreg (summary.aalen), 64
summary.Event (Event), 26
summary.pe-sasieni (pe.sasieni), 36
summary.pe.sasieni (pval), 56
summary.predict.timereg (pval), 56
summary.prop.excess (summary.aalen), 64
summary.resmean (pval), 56
summary.restricted.residual.mean (pval), 56
summary.timecox (summary.aalen), 64
summary.two.stage (pval), 56
Surv, 10, 59

timecox, 65
timemcBase (pval), 56
timereg.formula (pval), 56
timetest (pval), 56
TRACE, 68
t-tTRACE (TRACE), 68
twin.clustertrunc (pval), 56
two.stage, 69

vcov.aalen (pval), 56
vcov.comp.risk (pval), 56
vcov.cox.aalen (pval), 56
vcov.two.stage (pval), 56

wald.test, 72