Package ‘mets’

August 13, 2018

Type Package

Title Analysis of Multivariate Event Times

Version 1.2.4

Date 2018-08-12

Author Klaus K. Holst and Thomas Scheike

Maintainer Klaus K. Holst <klaus@holst.it>

Description Implementation of various statistical models for multivariate event history data <doi:10.1007/s10985-013-9244-x>. Including multivariate cumulative incidence models <doi:10.1002/sim.6016>, and bivariate random effects probit models (Liability models) <doi:10.1016/j.csda.2015.01.014>. Also contains two-stage binomial modelling that can do pairwise odds-ratio dependence modelling based marginal logistic regression models. This is an alternative to the alternating logistic regression approach (ALR).

License GPL (>= 2)

LazyLoad yes

URL https://github.com/kkholst/mets

BugReports https://github.com/kkholst/mets/issues

Depends R (>= 3.3), timereg (>= 1.9.2), lava (>= 1.6.3)

Imports numDeriv, compiler, Rcpp, splines, survival

Suggests lava.tobit (>= 0.4-7), prodlim, testthat (>= 0.11), ucminf, R.rsp (>= 0.40)

VignetteBuilder R.rsp

ByteCompile yes

LinkingTo Rcpp, RcppArmadillo

RoxygenNote 6.0.1

NeedsCompilation yes

Repository CRAN

Date/Publication 2018-08-13 05:30:12 UTC
### R topics documented:

- mets-package ................................................. 4
- aalenfrailty ............................................ 4
- back2timereg ............................................ 5
- base1cumhaz ............................................. 6
- base44cumhaz ............................................ 6
- base4cumhaz ............................................. 6
- basehazplot.phreg ....................................... 7
- bicomprisk ............................................... 8
- binomial.twostage ....................................... 10
- biprobit .................................................. 14
- blocksample ............................................. 16
- bptwin ..................................................... 18
- casewise .................................................. 19
- casewise.test ............................................ 20
- cif ......................................................... 22
- ClaytonOakes ............................................ 23
- cluster.index ............................................ 24
- concordance ............................................. 25
- cor.cif .................................................... 26
- count.history ........................................... 30
- covarianceRecurrent .................................... 31
- dagggregate ............................................. 32
- Dbvn ....................................................... 34
- dby ......................................................... 34
- dcov ....................................................... 36
- dcut ....................................................... 37
- dermalridges ............................................. 39
- dermalridgesMZ ......................................... 40
- divide.conquer .......................................... 40
- divide.conquer.timereg ................................ 41
- dlag ....................................................... 42
- dprint ..................................................... 43
- drcumhaz ................................................ 44
- drelevel ................................................... 44
- dsort ....................................................... 46
- dspline .................................................... 48
- dtable ...................................................... 49
- dtable ...................................................... 50
- dtransform ............................................... 51
- easy.binomial.twostage ................................. 52
- easy.survival.twostage .................................. 55
- EVaddGam ............................................... 57
- eventpois ............................................... 59
- familycluster.index ..................................... 60
- familyclusterWithProbands.index ..................... 61
- fast.approx ............................................... 62
R topics documented:

- fast.pattern .......................................................... 62
- fast.reshape .......................................................... 63
- gof.phreg .............................................................. 65
- gofG.phreg ............................................................. 66
- gofM.phreg ............................................................. 67
- Grandom.cif ........................................................... 68
- ipw ................................................................. 71
- ipw2 ................................................................. 73
- km ................................................................. 74
- lifecourse ............................................................. 75
- lifetable.matrix ....................................................... 76
- LinSpline ............................................................. 77
- mena ................................................................. 78
- mets.options .......................................................... 78
- migr ................................................................. 79
- multcif ............................................................... 79
- np ................................................................. 79
- npc ................................................................. 79
- phreg ............................................................... 80
- plack.cif ............................................................. 81
- pmvn ................................................................. 81
- predict.phreg ......................................................... 82
- print.casewise ........................................................ 82
- prob.exceed.recurrent ............................................... 83
- prt ................................................................. 85
- random.cif ........................................................... 85
- recurrentMarginal .................................................... 87
- simAalenFrailty ......................................................... 90
- simClaytonOakes ....................................................... 91
- simClaytonOakesWei ................................................... 91
- simRecurrent .......................................................... 92
- simRecurrentII ......................................................... 94
- summary.cor .......................................................... 95
- survival.iterative ..................................................... 97
- survival.twostage .................................................... 100
- test.conc ............................................................ 105
- tetrachoric ............................................................ 106
- twin.clustertrunc ..................................................... 107
- twinbmi .............................................................. 108
- twinlm .............................................................. 108
- twinsim .............................................................. 110
- twinstut .............................................................. 111

Index .............................. 112
Description
Implementation of various statistical models for multivariate event history data. Including multivariate cumulative incidence models, and bivariate random effects probit models (liability models).

Author(s)
Klaus K. Holst and Thomas Scheike

Examples

```r
## To appear
```

aalenfrailty

Description
Additive hazards model with (gamma) frailty

Usage

```r
aalenfrailty(time, status, X, id, theta, B = NULL, ...)
```

Arguments

- `time`: Time variable
- `status`: Status variable (0,1)
- `X`: Covariate design matrix
- `id`: Cluster variable
- `theta`: List of thetas (returns score evaluated here), or starting point for optimization (defaults to magic number 0.1)
- `B`: (optional) Cumulative coefficients (update theta by fixing B)
- `...`: Additional arguments to lower level functions

Details
Aalen frailty model
Value
Parameter estimates

Author(s)
Klaus K. Holst

Examples

```r
class="timereg"
library("timereg")
dd <- simAalenFrailty(5000)
f <- ~1+x
X <- model.matrix(f, dd) # design matrix for non-parametric terms
dd$t <- system.time(update(f, Surv(time, status)~-1, dd, n.sim=0, robust=0))
dix <- which(dd$status==1)
t1 <- system.time(bb <- .Call("Bhat", as.integer(dd$status),
    , x, 0.2, as.integer(dd$id), NULL, NULL,
    PACKAGE="mets"))
spec <- 1
spec <- 1
# plot(out, spec=spec)
# plot(dd$time[dix], bb$Bhat[, spec], col="red", type="s",
#  ylim=c(0, max(dd$time)*c(beta0, beta)[spec]))
# abline(a=0, b=c(beta0, beta)[spec])

# Not run:
thetas <- seq(0.1, 2, length.out=10)
Us <- unlist(aalenfrailty(dd$time, dd$status, X, dd$id, as.list(thetas)))
# plot(thetas, Us, type="l", ylim=c(-.5, 1)); abline(h=0, lty=2); abline(v=theta, lty=2)

op <- aalenfrailty(dd$time, dd$status, X, dd$id)

op

```

Description
convert to timereg object

Usage
back2timereg(obj)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>obj</td>
<td>no use</td>
</tr>
<tr>
<td>base1cumhaz</td>
<td>rate of CRBSI for HPN patients of Copenhagen</td>
</tr>
<tr>
<td>-------------</td>
<td>--------------------------------------------</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>rate of CRBSI for HPN patients of Copenhagen</td>
</tr>
<tr>
<td><strong>Source</strong></td>
<td>Estimated data</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>base44cumhaz</th>
<th>rate of Occlusion/Thrombosis complication for catheter of HPN patients of Copenhagen</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>rate of Occlusion/Thrombosis complication for catheter of HPN patients of Copenhagen</td>
</tr>
<tr>
<td><strong>Source</strong></td>
<td>Estimated data</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>base4cumhaz</th>
<th>rate of Mechanical (hole/defect) complication for catheter of HPN patients of Copenhagen</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>rate of Mechanical (hole/defect) complication for catheter of HPN patients of Copenhagen</td>
</tr>
<tr>
<td><strong>Source</strong></td>
<td>Estimated data</td>
</tr>
</tbody>
</table>
basehazplot.phreg

Plotting the baslines of stratified Cox

Description

Plotting the baslines of stratified Cox

Usage

basehazplot.phreg(x, se = FALSE, time = NULL, add = FALSE, ylim = NULL,
   _xlim = NULL, lty = NULL, col = NULL, legend = TRUE, ylab = NULL,
    polygon = TRUE, level = 0.95, stratas = NULL, robust = FALSE, ...)

Arguments

  x            phreg object
  se           to include standard errors
  time         to plot for specific time variables
  add          to add to previous plot
  ylim         to give ylim
  xlim         to give xlim
  lty          to specify lty of components
  col          to specify col of components
  legend       to specify col of components
  ylab         to specify ylab
  polygon      to get standard error in shaded form
  level        of standard errors
  stratas      wich strata to plot
  robust       to use robust standard errors if possible
  ...          Additional arguments to lower level funtions

Author(s)

Klaus K. Holst, Thomas Scheike

Examples

data(TRACE)
dcut(TRACE) <- ~.
out1 <- phreg(Surv(time,status==9)-vf+chf+strata(wmicat.4),data=TRACE)  
par(mfrow=c(2,2))
bplot(out1)
bplot(out1,stratas=c(0,3))
Estimation of concordance in bivariate competing risks data

**Usage**

```r
bicomprisk(formula, data, cause = c(1, 1), cens = 0, causes, indiv,
    strata = NULL, id, num, max.clust = 1000, marg = NULL,
    se.clusters = NULL, wname = NULL, prodlim = FALSE, messages = TRUE,
    model, return.data = 0, uniform = 0, conservative = 1,
    resample.iid = 1, ...)
```

**Arguments**

- **formula**: Formula with left-hand-side being a `Event` object (see example below) and the left-hand-side specifying the covariate structure.
- **data**: Data frame.
- **cause**: Causes (default (1,1)) for which to estimate the bivariate cumulative incidence.
- **cens**: The censoring code.
- **causes**: Causes.
- **indiv**: Indiv.
- **strata**: Strata.
- **id**: Clustering variable.
- **num**: Num.
- **max.clust**: Max number of clusters in `comp.risk` call for iid decomposition, `max.clust=NULL` uses all clusters otherwise rougher grouping.
- **marg**: Marginal cumulative incidence to make standard errors for same clusters for subsequent use in `casewise.test()`.
- **se.clusters**: To specify clusters for standard errors. Either a vector of cluster indices or a column name in `data`. Defaults to the `id` variable.
- **wname**: Name of additional weight used for paired competing risks data.
- **prodlim**: Prodlim to use prodlim estimator (Aalen-Johansen) rather than IPCW weighted estimator based on `comp.risk` function. These are equivalent in the case of no covariates. These estimators are the same in the case of stratified fitting.
- **messages**: Control amount of output.
**bicomprisk**

- **model** Type of competing risk model (default is Fine-Gray model "fg", see comp.risk).
- **return.data** Should data be returned (skipping modeling).
- **uniform** to compute uniform standard errors for concordance estimates based on resampling.
- **conservative** for conservative standard errors, recommended for larger data-sets.
- **resample.iid** to return iid residual processes for further computations such as tests.
- **...** Additional arguments to comp.risk function

**Author(s)**

Thomas Scheike, Klaus K. Holst

**Examples**

```r
library("timereg")

## Simulated data example
prt <- simnordic.random(2000, delayed=TRUE, ptrunc=0.7,
cordz=0.5, cormz=2, lamz=0.3)
## Bivariate competing risk, concordance estimates
p11 <- bicomprisk(Event(time, cause)=strata(zyg)+id(id), data=prt, cause=c(1,1))

p11$mz <- p11$model$"MZ"
p11$dz <- p11$model$"DZ"
par(mfrow=c(1,2))
## Concordance
plot(p11$mz ylim=c(0,0.1));
plot(p11$dz ylim=c(0,0.1));

## entry time, truncation weighting
## other weighting procedure
prt1 <- prt[!prt$truncated,]
prt2 <- ipw2(prt1, cluster="id", same.cens=TRUE,
time="time", cause="cause", entrytime="entry",
pairs=TRUE, strata="zyg", obs.only=TRUE)

prt2$event <- (prt2$cause1==1)*(prt2$cause2==1)*1
prt2$time1 <- pmax(prt2$time1, prt2$time2)
ipwc <- comp.risk(Event(time1, event)~1+factor(zyg),
data=prt2, cause=1, n.sim=0, model="rcif2", times=50:90,
weights=prt2$weights1, cens.weights=rep(1, nrow(prt2)))

p11$mz <- ipwc$cum[,2]
p11$dz <- ipwc$cum[,3]
lines(ipwc$cum[,1], p11$mz, col=3)
lines(ipwc$cum[,1], p11$dz, col=3)
```
Fits Clayton-Oakes or bivariate Plackett (OR) models for binary data using marginals that are on logistic form. If clusters contain more than two times, the algorithm uses a composite likelihood based on all pairwise bivariate models.

Description

The pairwise pairwise odds ratio model provides an alternative to the alternating logistic regression (ALR).

Usage

```r
binomial.twostage(marginbin, data = sys.parent(),
    score.method = "fisher.scoring", Nit = 60, detail = 0,
    clusters = NULL, silent = 1, weights = NULL, control = list(),
    theta = NULL, theta.des = NULL, var.link = 0, var.par = 1,
    var.func = NULL, iid = 1, step = 1, notaylor = 1,
    model = "plackett", marginal.p = NULL, beta.iid = NULL,
    Dbeta.iid = NULL, strata = NULL, max.clust = NULL, se.clusters = NULL,
    numDeriv = 0, random.design = NULL, pairs = NULL, pairs.rvs = NULL,
    additive.gamma.sum = NULL, pair.ascertained = 0, case.control = 0,
    twostage = 1, beta = NULL)
```

Arguments

- `marginbin` Marginal binomial model
- `data` data frame
- `score.method` Scoring method default is "fisher.scoring" among "fisher.scoring","nlminb","optimize","nlm"
- `Nit` Number of iterations
- `detail` Detail
- `clusters` Cluster variable
- `silent` Debug information
- `weights` Weights for log-likelihood, can be used for each type of outcome in 2x2 tables.
- `control` Optimization arguments
- `theta` Starting values for variance components
- `theta.des` design for dependence parameters, when pairs are given this is could be a (pairs) x (numer of parameters) x (max number random effects) matrix
- `var.link` Link function for variance
- `var.par` parametrization
- `var.func` when alternative parametrizations are used this function can specify how the parameters are related to the $\lambda_j$'s.
iid  Calculate i.i.d. decomposition when iid>=1, when iid=2 then avoids adding the uncertainty for marginal parameters for additive gamma model (default).

step  Step size

notaylor  Taylor expansion

model  model

marginal.p  vector of marginal probabilities

beta.iid  iid decomposition of marginal probability estimates for each subject, if based on GLM model this is computed.

Dbeta.iid  derivatives of marginal model wrt marginal parameters, if based on GLM model this is computed.

strata  strata for fitting: considers only pairs where both are from same strata

max.clust  max clusters

se.clusters  clusters for iid decomposition for robust standard errors

numDeriv  uses Fisher scoring approx of second derivative if 0, otherwise numerical derivatives

random.design  random effect design for additive gamma model, when pairs are given this is a (pairs) x (2) x (max number random effects) matrix, see pairs.rvs below

pairs  matrix with rows of indices (two-columns) for the pairs considered in the pairwise composite score, useful for case-control sampling when marginal is known.

pairs.rvs  for additive gamma model and random.design and theta.des are given as arrays, this specifies number of random effects for each pair.

additive.gamma.sum  this is specification of the lamtot in the models via a matrix that is multiplied onto the parameters theta (dimensions=(number random effects x number of theta parameters), when null then sums all parameters. Default is a matrix of 1’s

pair.ascertained  if pairs are sampled only when there are events in the pair i.e. Y1+Y2>=1.

case.control  if data is case control data for pair call, and here 2nd column of pairs are probands (cases or controls)

twostage  default twostage=1, to fit MLE use twostage=0

beta  is starting value for beta for MLE version

Details

The reported standard errors are based on a cluster corrected score equations from the pairwise likelihoods assuming that the marginals are known. This gives correct standard errors in the case of the Odds-Ratio model (Plackett distribution) for dependence, but incorrect standard errors for the Clayton-Oakes types model (that is also called "gamma"-frailty). For the additive gamma version of the standard errors are adjusted for the uncertainty in the marginal models via an iid decomposition using the iid() function of lava. For the clayton oakes model that is not specified via the random effects these can be fixed subsequently using the iid influence functions for the marginal model, but typically this does not change much.
For the Clayton-Oakes version of the model, given the gamma distributed random effects it is assumed that the probabilities are independent, and that the marginal survival functions are on logistic form

$$\text{logit}(P(Y = 1|X)) = \alpha + x^T \beta$$

therefore conditional on the random effect the probability of the event is

$$\text{logit}(P(Y = 1|X, Z)) = \exp(-Z \cdot \text{Laplace}^{-1}(\text{lamtot}, \text{lamtot}, P(Y = 1|x)))$$

Can also fit a structured additive gamma random effects model, such the ACE, ADE model for survival data:

Now random.design specifies the random effects for each subject within a cluster. This is a matrix of 1’s and 0’s with dimension n x d. With d random effects. For a cluster with two subjects, we let the random.design rows be v1 and v2. Such that the random effects for subject 1 is

$$v_1^T(Z_1, ..., Z_d)$$

, for d random effects. Each random effect has an associated parameter ($\lambda_1, ..., \lambda_d$). By construction subjects 1’s random effect are Gamma distributed with mean $\lambda_j/v_1^T \lambda$ and variance $\lambda_j/(v_1^T \lambda)^2$. Note that the random effect $v_1^T(Z_1, ..., Z_d)$ has mean 1 and variance $1/(v_1^T \lambda)$. It is here assumed that $\text{lamtot} = v_1^T \lambda$ is fixed over all clusters as it would be for the ACE model below.

The DEFAULT parametrization uses the variances of the random effects (var.par=1)

$$\theta_j = \lambda_j/(v_1^T \lambda)^2$$

For alternative parametrizations (var.par=0) one can specify how the parameters relate to $\lambda_j$ with the function

Based on these parameters the relative contribution (the heritability, h) is equivalent to the expected values of the random effects $\lambda_j/v_1^T \lambda$

Given the random effects the probabilities are independent and on the form

$$\text{logit}(P(Y = 1|X)) = \exp(-\text{Laplace}^{-1}(\text{lamtot}, \text{lamtot}, P(Y = 1|x)))$$

with the inverse laplace of the gamma distribution with mean 1 and variance lamtot.

The parameters ($\lambda_1, ..., \lambda_d$) are related to the parameters of the model by a regression construction $\text{pard}$ (d x k), that links the d $\lambda$ parameters with the (k) underlying $\theta$ parameters

$$\lambda = \theta_{\text{des}} \theta$$

here using theta.des to specify these low-dimension association. Default is a diagonal matrix.

Author(s)

Thomas Scheike

References

Two-stage binomial modelling
Examples

library("timereg")
data("twinstut", package = "mets")
twinstut0 <- subset(twinstut, tvparnr < 2300000)
twinstut <- twinstut0

twinstut$binstut <- (twinstut$stutter == "yes") * 1
theta.des <- model.matrix(~ -1 + factor(zyg), data = twinstut)
margbin <- glm(binstut ~ factor(sex) + age, data = twinstut, family = binomial())
bin <- binomial.twostage(margbin, data = twinstut, var.link = 1,
                        clusters = twinstut$tvparnr, theta.des = theta.des, detail = 0,
                        score.method = "fisher.scoring")
summary(bin)

twinstut$cage <- scale(twinstut$age)
theta.des <- model.matrix(~ -1 + factor(zyg) + cage, data = twinstut)
bina <- binomial.twostage(margbin, data = twinstut, var.link = 1,
                          clusters = twinstut$tvparnr, theta.des = theta.des)
summary(bina)

theta.des <- model.matrix(~ -1 + factor(zyg) + factor(zyg) * cage, data = twinstut)
bina <- binomial.twostage(margbin, data = twinstut, var.link = 1,
                          clusters = twinstut$tvparnr, theta.des = theta.des)
summary(bina)

## refers to zygosity of first subject in each pair : zyg1
## could also use zyg2 (since zyg2 = zyg1 within twinpair's)
out <- easy.binomial.twostage(stutter ~ factor(sex) + age, data = twinstut,
                           response = "binstut", id = "tvparnr", var.link = 1,
                           theta.formula = ~ -1 + factor(zyg1))
summary(out)

## refers to zygosity of first subject in each pair : zyg1
## could also use zyg2 (since zyg2 = zyg1 within twinpair's)
desfs <- function(x, num1 = "zyg1", num2 = "zyg2")
  c(x[num1] == "dx", x[num1] == "mz", x[num1] == "os") * 1
out3 <- easy.binomial.twostage(binstut ~ factor(sex) + age,
                              data = twinstut, response = "binstut", id = "tvparnr", var.link = 1,
                              theta.formula = desfs, desnames = c("mz", "dz", "os"))
summary(out3)

#### use of clayton oakes binomial additive gamma model

## Reduce Ex.Timings
data <- simbinClaytonOakes.family.ace(10000, 2, 1, beta = NULL, alpha = NULL)
margbin <- glm(ybin ~ x, data = data, family = binomial())
margbin

head(data)
data$number <- c(1, 2, 3, 4)
data$child <- 1 * (data$number == 3)
### Description

**Bivariate Probit model**

**Usage**

```r
biprobit(x, data, id, rho = ~1, num = NULL, strata = NULL, 
  eqmarg = TRUE, indep = FALSE, weights = NULL, biweight, 
  samecens = TRUE, randomeffect = FALSE, vcov = "robust", 
  pairs.only = FALSE, allmarg = samecens & !is.null(weights), 
  control = list(trace = 0), messages = 1, constrain = NULL, 
  table = pairs.only, p = NULL, ...) 
```

**Arguments**

- `x` formula (or vector)
- `data` data.frame
- `id` The name of the column in the dataset containing the cluster id-variable.
- `rho` Formula specifying the regression model for the dependence parameter.
**biprobit**

- **num**: Optional name of order variable
- **strata**: Strata
- **eqmarg**: If TRUE same marginals are assumed (exchangeable)
- **indep**: Independence
- **weights**: Weights
- **biweight**: Function defining the bivariate weight in each cluster
- **samecens**: Same censoring
- **randomeffect**: If TRUE a random effect model is used (otherwise correlation parameter is estimated allowing for both negative and positive dependence)
- **vcov**: Type of standard errors to be calculated
- **pairs.only**: Include complete pairs only?
- **allmarg**: Should all marginal terms be included
- **control**: Control argument parsed on to the optimization routine. Starting values may be parsed as 'start'.
- **messages**: Control amount of messages shown
- **constrain**: Vector of parameter constraints (NA where free). Use this to set an offset.
- **table**: Type of estimation procedure
- **p**: Parameter vector \( p \) in which to evaluate log-Likelihood and score function
- ... Optional arguments

### Examples

```r
data(prt)
prt0 <- subset(prt,country=="Denmark")
a <- biprobit(cancer~1+zyg, ~1+zyg, data=prt0, id="id")
b <- biprobit(cancer~1+zyg, ~1+zyg, data=prt0, id="id",pairs.only=TRUE)
predict(b,newdata=lava::Expand(prt,zyg=c("MZ")))
predict(b,newdata=lava::Expand(prt,zyg=c("MZ","DZ")))

## Reduce Ex.Timings
library(lava)
m <- lvm(c(y1,y2)-x)
covariance(m,y1,y2) <- "r"
constrain(m,r-x+a+b) <- function(x) tanh(x[2]*x[3])*x[1])
distribution(m,-x) <- uniform.lvm(a=-1,b=1)
ordinal(m) <- ~y1+y2
d <- sim(m,1000,p=c(a=0,b=-1)); d <- d[order(d$x),]
dd <- fast.reshape(d)
a <- biprobit(y~l+x,rho=1+x,data=dd,id="id")
summary(a, mean.contrast=c(1,.5), cor.contrast=c(1,.5))
with(predict(a,data.frame(x=seq(-1,1,by=.1))), plot(p00~x,type="l",title="Concordance", lwd=2, xlab="x", ylab="Concordance", lty=0, col=Col(1)))

pp <- predict(a,data.frame(x=seq(-1.1,1,by=.1)),which=c(1))
plot(pp[,1]-pp$x, type="l", xlab="x", ylab="Concordance", lwd=2, xaxs="i")
confband(pp$x,pp[,2],pp[,3],polygon=TRUE,lty=0,col=Col(1))
```
blocksample

Description
Sample blockwise from clustered data

Usage
blocksample(data, size, idvar = NULL, replace = TRUE, ...)

```r
pp <- predict(a, data.frame(x = seq(-1, 1, by = .1)), which = c(9)) # rhoo
plot(pp[, 1] - pp$x, type = "l", xlab = "x", ylab = "Correlation", lwd = 2, xaxs = "i")
confband(pp$x, pp[, 2], pp[, 3], polygon = TRUE, lty = 0, col = Col(1))
with(pp, lines(x, tanh(-x), lwd = 2, lty = 2))

xp <- seq(-1, 1, length.out = 6); delta <- mean(diff(xp))
a2 <- biprobit(y = 1 + x, rho = -1 + 1 * cut(x, breaks = xp), data = dd, id = "id")
pp2 <- predict(a2, data.frame(x = xp[-1] - delta/2), which = c(9)) # rhoo
confband(pp2$x, pp2[, 2], pp2[, 3], center = pp2[, 1])

## Time
## Not run:
a <- biprobit.time(cancer = 1, rho = -1 + zyg, id = "id", data = prt, eqmarg = TRUE,
                    cens.formula = Surv(time, status == 0) ~ 1,
                    breaks = seq(75, 100, by = 3), fix.censweights = TRUE)
a <- biprobit.time2(cancer = 1 + zyg, rho = -1 + zyg, id = "id", data = prt0, eqmarg = TRUE,
                    cens.formula = Surv(time, status == 0) ~ zyg, breaks = 100)
a1 <- biprobit.time2(cancer = 1, rho = -1, id = "id", data = subset(prt0, zyg == "MZ"), eqmarg = TRUE,
                    cens.formula = Surv(time, status == 0) ~ 1,
                    breaks = 100, pairs-only = TRUE)
a2 <- biprobit.time2(cancer = 1, rho = -1, id = "id", data = subset(prt0, zyg == "DZ"), eqmarg = TRUE,
                    cens.formula = Surv(time, status == 0) ~ 1,
                    breaks = 100, pairs-only = TRUE)
prt0$trunc <- (prt0$time * runif(nrow(prt0)) * rbinom(nrow(prt0), 1, 0.5))
a3 <- biprobit.time(cancer = 1, rho = -1, id = "id", data = subset(prt0, zyg == "DZ"), eqmarg = TRUE,
                    cens.formula = Surv(trunc, time, status == 0) ~ 1,
                    breaks = 100, pairs-only = TRUE)

plot(a, which = 3, ylim = c(0, 0.1))
## End(Not run)
```
**blocksample**

**Arguments**

- **data**  
  Data frame
- **size**  
  Size of samples
- **idvar**  
  Column defining the clusters
- **replace**  
  Logical indicating whether to sample with replacement
- **...**  
  Additional arguments to lower level functions

**Details**

Original id is stored in the attribute 'id'

**Value**

- **data.frame**

**Author(s)**

- Klaus K. Holst

**Examples**

```r

d <- data.frame(x=rnorm(5), z=rnorm(5), id=c(4,10,10,5,5), v=rnorm(5))
(dd <- blocksample(d,size=20,-id))
attributes(dd)$id

## Not run:
blocksampling(data.table::data.table(d),1e6,-id)

## End(Not run)

d <- data.frame(x=c(1,rnorm(9)),
  z=rnorm(10),
  id=c(4,10,10,5,5,4,4,5,10,5),
  id2=c(1,1,2,1,2,1,1,1,1,2),
  v=rnorm(10))
dsample(d,-id, size=2)
dsample(d,-id+id2)
dsample(d,x+2*~id|x>0, size=5)
```
Liability model for twin data

**Description**

Liability-threshold model for twin data

**Usage**

```r
bptwin(x, data, id, zyg, DZ, group = NULL, num = NULL, weights = NULL, 
biweight = function(x) 1/min(x), strata = NULL, messages = 1, 
control = list(trace = 0), type = "ace", eqmean = TRUE, 
pairs.only = FALSE, samecens = TRUE, allmarg = samecens & 
!is.null(weights), stderr = TRUE, robustvar = TRUE, p, indiv = FALSE, 
constrain, bound = FALSE, varlink, ...)
```

**Arguments**

- **x**: Formula specifying effects of covariates on the response.
- **data**: data.frame with one observation per row. In addition a column with the zygosity (DZ or MZ given as a factor) of each individual must be specified as well as a twin id variable giving a unique pair of numbers/factors to each twin pair.
- **id**: The name of the column in the dataset containing the twin-id variable.
- **zyg**: The name of the column in the dataset containing the zygosity variable.
- **DZ**: Character defining the level in the zyg variable corresponding to the dizygotic twins.
- **group**: Optional. Variable name defining group for interaction analysis (e.g., gender)
- **num**: Optional twin number variable
- **weights**: Weight matrix if needed by the chosen estimator (IPCW)
- **biweight**: Function defining the bivariate weight in each cluster
- **strata**: Strata
- **messages**: Control amount of messages shown
- **control**: Control argument parsed on to the optimization routine. Starting values may be parsed as 'start'.
- **type**: Character defining the type of analysis to be performed. Should be a subset of "ace" (additive genetic factors, common environmental factors, dominant genetic factors, unique environmental factors).
- **eqmean**: Equal means (with type="cor")?
- **pairs.only**: Include complete pairs only?
- **samecens**: Same censoring
- **allmarg**: Should all marginal terms be included
- **stderr**: Should standard errors be calculated?
robustvar  If TRUE robust (sandwich) variance estimates of the variance are used
p  Parameter vector p in which to evaluate log-Likelihood and score function
indiv  If TRUE the score and log-Likelihood contribution of each twin-pair
constrain  Development argument
bound  Development argument
varlink  Link function for variance parameters
...  Additional arguments to lower level functions

Author(s)
Klaus K. Holst

See Also
twinlm, twinlm.time, twinlm.strata, twinsim

Examples
data(twinstut)
b0 <- bptwin(stutter~sex,
    data=droplevels(subset(twinstut, zyg%in%c("mz","dz"))),
    id=tvparnr, zyg="zyg", DZ="dz", type="ae")
summary(b0)

casewise  Estimates the casewise concordance based on Concordance and marginal estimate using prodlim but no testing

Description

Usage
casewise(conc, marg, cause.marg)

Arguments
conc  Concordance
marg  Marginal estimate
cause.marg  specifies which cause that should be used for marginal cif based on prodlim

Author(s)
Thomas Scheike
Examples

```r
## Reduce Ex.Timings
library(prodlim)
data(prt);

### marginal cumulative incidence of prostate cancer###
outm <- prodlim(Hist(time,status)+1,data=prt)
times <- 0:100
cifmz <- predict(outm,cause=2,time=times,newdata=data.frame(zyg="MZ")) ## cause is 2 (second cause)
cifdz <- predict(outm,cause=2,time=times,newdata=data.frame(zyg="DZ"))

### concordance for MZ and DZ twins
cc <- bicomprisk(Event(time,status)-strata(zyg)+id(id),data=prt,cause=c(2,2),prodlim=TRUE)
cdz <- cc$smodel$DZ
cmz <- cc$smodel$MZ

cdz <- casewise(cdz,outm,cause.marg=2)
cmz <- casewise(cmz,outm,cause.marg=2)
plot(cmz,ci=NULL,ylim=c(0,0.5),xlim=c(60,100),legend=TRUE,col=c(3,2,1))
par(new=TRUE)
plot(cdz,ci=NULL,ylim=c(0,0.5),xlim=c(60,100),legend=TRUE)
summary(cdz)
summary(cmz)
```

casewise.test

Estimates the casewise concordance based on Concordance and marginal estimate using timereg and performs test for independence

Description

Estimates the casewise concordance based on Concordance and marginal estimate using timereg and performs test for independence

Usage

casewise.test(conc, marg, test = "no-test", p = 0.01)

Arguments

- `conc`: Concordance
- `marg`: Marginal estimate
- `test`: Type of test for independence assumption. "conc" makes test on concordance scale and "case" means a test on the casewise concordance
- `p`: check that marginal probability is greater at some point than p
Details

Uses cluster based conservative standard errors for marginal

Author(s)

Thomas Scheike

Examples

```r
## Reduce Ex.Timings
library("timerreg")
data("prt",package="mets");

prt <- prt[which(prt$id %in% sample(unique(prt$id),7500)),]
### marginal cumulative incidence of prostate cancer
times <- seq(60,100,by=2)
outm <- comp.risk(Event(time,status)+1,data=prt,cause=2,times=times)
cifmz <- predict(outm,X=1,uniform=0,resample.iid=1)
cifdz <- predict(outm,X=1,uniform=0,resample.iid=1)
### concordance for MZ and DZ twins
c <- bicomprisk(Event(time,status)=strata(zyg)+id(id),
  data=prt,cause=c(2,2))
cdz <- cc$model$"DZ"
cmz <- cc$model$"MZ"

### To compute casewise cluster argument must be passed on,
### here with a max of 100 to limit comp-time
outm <- comp.risk(Event(time,status)+1,data=prt,cause=2,times=times,max.clust=100)
cifmz <- predict(outm,X=1,uniform=0,resample.iid=1)
c <- bicomprisk(Event(time,status)=strata(zyg)+id(id),data=prt,
  cause=c(2,2),se.clusters=outm$clusters)
cdz <- cc$model$"DZ"
cmz <- cc$model$"MZ"

cdz <- casewise.test(cdz,cifmz,test="case")  ## test based on casewise
cmz <- casewise.test(cmz,cifmz,test="conc")  ## based on concordance

plot(cmz,ylim=c(0,0.7),xlim=c(60,100))
par(new=TRUE)
plot(cdz,ylim=c(0,0.7),xlim=c(60,100))
slope.process(cdz$casewise[,1],cdz$casewise[,2],iid=cdz$casewise.iid)
slope.process(cmz$casewise[,1],cmz$casewise[,2],iid=cmz$casewise.iid)
```
cif

Cumulative incidence with robust standard errors

Description

Cumulative incidence with robust standard errors Robust variance is default variance with the summary.

Usage

cif(formula, data = data, cause = 1, cens.code = 0, ...)

Arguments

formula       formula with 'Surv' outcome (see coxph)
data          data frame
cause         NULL looks at all, otherwise specify which cause to consider
cens.code     censoring code "0" is default
...           Additional arguments to lower level funtions

Author(s)

Thomas Scheike

Examples

data(TRACE)
TRACE$cluster <- sample(1:100,1878,replace=TRUE)
out1 <- cif(Event(time,status)-+1,data=TRACE,cause=9)
out2 <- cif(Event(time,status)-+1+cluster(cluster),data=TRACE,cause=9)

out1 <- cif(Event(time,status)-strata(vf,chf),data=TRACE,cause=9)
out2 <- cif(Event(time,status)-strata(vf,chf)+cluster(cluster),data=TRACE,cause=9)

par(mfrow=c(1,2))
bplot(out1,se=TRUE)
bplot(out2,se=TRUE)
Clayton-Oakes model with piece-wise constant hazards

Description

Clayton-Oakes frailty model

Usage

ClaytonOakes(formula, data = parent.frame(), cluster, var.formula = ~1,
cuts = NULL, type = "piecewise", start, control = list(),
var.invlink = exp, ...)

Arguments

formula formula specifying the marginal proportional (piecewise constant) hazard structure with the right-hand-side being a survival object (Surv) specifying the entry time (optional), the follow-up time, and event/censoring status at follow-up. The clustering can be specified using the special function cluster (see example below).
data Data frame
cluster Variable defining the clustering (if not given in the formula)
var.formula Formula specifying the variance component structure (if not given via the cluster special function in the formula) using a linear model with log-link.
cuts Cut points defining the piecewise constant hazard
type when equal to two.stage, the Clayton-Oakes-Glidden estimator will be calculated via the timereg package
start Optional starting values
control Control parameters to the optimization routine
var.invlink Inverse link function for variance structure model
... Additional arguments

Author(s)

Klaus K. Holst

Examples

set.seed(1)
d <- subset(simClaytonOakes(500,4,2,1,stop.time=2,left=2),truncated)
e <- ClaytonOakes(survival::Surv(lefttime,time,status)=x+cluster(~1,cluster),
cuts=c(0,0,5,1,2),data=d)
e
cluster.index

Finds subjects related to same cluster

d2 <- simClaytonOakes(500,4,2,1,stop.time=2,left=0)
d2$z <- rep(1,nrow(d2));d2$z[d2$cluster%in%sample(d2$cluster,100)] <- 0

## Marginal=Cox Proportional Hazards model:
ts <- ClaytonOakes(survival::Surv(time,status)-timereg::prop(x)+cluster(~1,cluster),
  data=d2,type="two.stage")

## Marginal=Aalen's additive model:
ts2 <- ClaytonOakes(survival::Surv(time,status)-x+cluster(~1,cluster),
  data=d2,type="two.stage")

## Marginal=Piecewise constant:
e2 <- ClaytonOakes(survival::Surv(time,status)-x+cluster(~1+factor(z),cluster),
  cuts=c(0,0.5,1,2),data=d2)
e2

plot(ts)
plot(e2,add=TRUE)

e3 <- ClaytonOakes(survival::Surv(time,status)-x+cluster(~1,cluster),cuts=c(0,0.5,1,2),
  data=d, var.invlink=identity)
e3

cluster.index

Description
Finds subjects related to same cluster

Usage
cluster.index(clusters, index.type = FALSE, num = NULL, Rindex = 0,
  mat = NULL, return.all = FALSE, code.na = NA)

Arguments

clusters list of indeces
index.type if TRUE then already list of integers of index.type
num to get numbering according to num-type in separate columns
Rindex index starts with 1, in C is it is 0
mat to return matrix of indeces
return.all return all arguments
code.na how to code missing values

Author(s)
Klaus Holst, Thomas Scheike

References
Cluster indeces
concordance

See Also

familycluster.index familyclusterWithProbands.index

Examples

```r
i <- c(1,1,2,2,1,3)
d <- cluster.index(i)
print(d)

type <- c("m","f","m","c","c","c")
d <- cluster.index(i, num=type, Rindex=1)
print(d)
```

Description

Concordance

Usage

```r
concordance(object, cif1, cif2 = NULL, messages = TRUE, model = NULL, coefs = NULL, ...)
```

Arguments

- `object`: Output from the cor.cif, rr.cif or or.cif function
- `cif1`: Marginal cumulative incidence
- `cif2`: Marginal cumulative incidence of other cause (cause2) if it is different from cause1
- `messages`: To print messages
- `model`: Specifies which model that is considered if object not given.
- `coefs`: Specifies dependence parameters if object is not given.
- `...`: Extra arguments, not used.

Author(s)

Thomas Scheike
Cross-odds-ratio, OR or RR risk regression for competing risks

**Description**

Fits a parametric model for the log-cross-odds-ratio for the predictive effect of for the cumulative incidence curves for $T_1$ experiencing cause $i$ given that $T_2$ has experienced a cause $k$:

$$\log(COR(i|k)) = h(\theta, z_1, i, z_2, k, t) = \text{default} \theta^T z =$$

with the log cross odds ratio being

$$COR(i|k) = \frac{O(T_1 \leq t, cause_1 = i | T_2 \leq t, cause_2 = k)}{O(T_1 \leq t, cause_1 = i)}$$

the conditional odds divided by the unconditional odds, with the odds being, respectively

$$O(T_1 \leq t, cause_1 = i | T_2 \leq t, cause_2 = k) = \frac{P_x(T_1 \leq t, cause_1 = i | T_2 \leq t, cause_2 = k)}{P_x((T_1 \leq t, cause_1 = i)^c | T_2 \leq t, cause_2 = k)}$$

and

$$O(T_1 \leq t, cause_1 = i) = \frac{P_x(T_1 \leq t, cause_1 = i)}{P_x((T_1 \leq t, cause_1 = i)^c)}.$$

Here $B^c$ is the complement event of $B$, $P_x$ is the distribution given covariates ($x$ are subject specific and $z$ are cluster specific covariates), and $h()$ is a function that is the simple identity $\theta^T z$ by default.

**Usage**

```r
cor.cif(cif, data, cause = NULL, times = NULL, cause1 = 1, cause2 = 1, cens.code = NULL, cens.model = "KM", Nit = 40, detail = 0, clusters = NULL, theta = NULL, theta.des = NULL, step = 1, sym = 0, weights = NULL, par.func = NULL, dpar.func = NULL, dimpar = NULL, score.method = "nlminb", same.cens = FALSE, censoring.weights = NULL, silent = 1, ...)```

**Arguments**

- `cif` a model object from the comp.risk function with the marginal cumulative incidence of cause1, i.e., the event of interest, and whose odds the comparison is compared to the conditional odds given cause2.
- `data` a data.frame with the variables.
- `cause` specifies the causes related to the death times, the value cens.code is the censoring value. When missing it comes from marginal cif.
- `times` time-vector that specifies the times used for the estimating euqations for the cross-odds-ratio estimation.
- `cause1` specifies the cause considered.
- `cause2` specifies the cause that is conditioned on.
cens.code specifies the code for the censoring if NULL then uses the one from the marginal cif model.
cens.model specified which model to use for the ICPW, KM is Kaplan-Meier alternatively it may be "cox"
Nit number of iterations for Newton-Raphson algorithm.
detail if 0 no details are printed during iterations, if 1 details are given.
clusters specifies the cluster structure.
theta specifies starting values for the cross-odds-ratio parameters of the model.
theta.des specifies a regression design for the cross-odds-ratio parameters.
step specifies the step size for the Newton-Raphson algorithm.
sym specifies if symmetry is used in the model.
weights weights for estimating equations.
par.func parfunc
dpar.func  dparfunc
dimpar dimpar
score.method "nlminb", can also use "fisher-scoring".
same.cens if true then censoring within clusters are assumed to be the same variable, default is independent censoring.
censoring.weights these probabilities are used for the bivariate censoring dist.
silent 1 to suppress output about convergence related issues.
...
Not used.

Details

The OR dependence measure is given by

\[
OR(i, k) = \log \left( \frac{O(T_1 \leq t, cause_1 = i | T_2 \leq t, cause_2 = k)}{O(T_1 \leq t, cause_1 = i | T_2 \leq t, cause_2 = k)} \right)
\]

This measure is numerically more stable than the COR measure, and is symmetric in i,k.

The RR dependence measure is given by

\[
RR(i, k) = \log \left( \frac{P(T_1 \leq t, cause_1 = i, T_2 \leq t, cause_2 = k)}{P(T_1 \leq t, cause_1 = i)P(T_2 \leq t, cause_2 = k)} \right)
\]

This measure is numerically more stable than the COR measure, and is symmetric in i,k.

The model is fitted under symmetry (sym=1), i.e., such that it is assumed that \(T_1\) and \(T_2\) can be interchanged and leads to the same cross-odd-ratio (i.e. \(COR(i|k) = COR(k|i)\)), as would be expected for twins or without symmetry as might be the case with mothers and daughters (sym=0).

\(h()\) may be specified as an R-function of the parameters, see example below, but the default is that it is simply \(\theta^T z\).
Value

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

theta estimate of proportional odds parameters of model.
var.theta variance for gamma.
hess the derivative of the used score.
score scores at final stage.
theta.iid matrix of iid decomposition of parametric effects.

Author(s)

Thomas Scheike

References

Cross odds ratio Modelling of dependence for Multivariate Competing Risks Data, Scheike and Sun (2010), work in progress.
A Semiparametric Random Effects Model for Multivariate Competing Risks Data, Scheike, Zhang, Sun, Jensen (2010), Biometrika.

Examples

library("timereg")
data(multcif);
multcif$cause[multcif$cause==0] <- 2
zyg <- rep(rbinom(200,1,0.5),each=2)
theta.des <- model.matrix(~-1+factor(zyg))
times=seq(0.05,1,by=0.05) # to speed up computations use only these time-points
add<-comp.risk(Event(time,cause)-1+cluster(id),data=multcif,cause=1,
n.sim=0,times=times,model="fg",max.clust=NULL)
add2<-comp.risk(Event(time,cause)-1+cluster(id),data=multcif,cause=2,
n.sim=0,times=times,model="fg",max.clust=NULL)
out1<cor.cif(add,data=multcif,cause1=1,cause2=1)
summary(out1)

out2<cor.cif(add,data=multcif,cause1=1,cause2=1,theta.des=theta.des)
summary(out2)

##out3<cor.cif(add,data=multcif,cause1=1,cause2=2,cif2=add2)
##summary(out3)

# investigating further models using parfunc and dparfunc
# Reduce Ex.Timings
set.seed(100)
prt<-simnordic.random(2000,cordz=2,coroz=5)
```r
prt$status <- prt$cause
table(prt$status)

times <- seq(40, 100, by=10)
cifmod <- comp.risk(Event(time, cause)~1+cluster(id), data=prt, 
  cause=1, n.sim=0, 
  times=times, conservative=1, max.clust=NULL, model="fg")
theta.des <- model.matrix(~1+factor(zyg), data=prt)

parfunc <- function(par, t, pardes)
{
  par <- pardes %*% c(par[1], par[2]) + 
  par
}
head(parfunc(c(0.1, 1, 0.1, 1), 50, theta.des))

dparfunc <- function(par, t, pardes)
{
  dpar <- cbind(pardes, t(t(pardes) * c( (t-60)/12, (t-60)/12) )
    dpar
}
head(dparfunc(c(0.1, 1, 0.1, 1), 50, theta.des))

names(prt)
or1 <- or.cif(cifmod, data=prt, cause1=1, cause2=1, theta.des=theta.des, 
  same.cens=TRUE, theta=c(0.6, 1.1, 0.1, 0.1), 
  par.func=parfunc, dpar.func=dparfunc, dimpar=4, 
  score.method="fisher.scoring", detail=1)
summary(or1)

cor1 <- cor.cif(cifmod, data=prt, cause1=1, cause2=1, theta.des=theta.des, 
  same.cens=TRUE, theta=c(0.5, 1.0, 0.1, 0.1), 
  par.func=parfunc, dpar.func=dparfunc, dimpar=4, 
  control=list(trace=TRUE), detail=1)
summary(cor1)

### piecewise constant OR model
gparfunc <- function(par, t, pardes)
{
  cuts <- c(0, 80, 90, 120)
  grop <- diff(t<cuts)
  paru <- (pardes[,1]==1) * sum(grop*par[1:3]) + 
    (pardes[,2]==1) * sum(grop*par[4:6])
  paru
}

dgparfunc <- function(par, t, pardes)
{
  cuts <- c(0, 80, 90, 120)
  grop <- diff(t<cuts)
  par1 <- matrix(c(grop), nrow(pardes), length(grop), byrow=TRUE)
  parmz <- par1* (pardes[,1]==1)
```
count.history

Counts the number of previous events of two types for recurrent events processes

Description

Counts the number of previous events of two types for recurrent events processes

Usage

count.history(data, status = "status", id = "id", types = 1:2, 
names.count = "Count", lag = TRUE)

Arguments

data data-frame
status name of status
id id
types types of the events (code) related to status
names.count name of Counts, for example Count1 Count2 when types=c(1,2)
lag if true counts previously observed, and if lag=FALSE counts up to know

Author(s)

Thomas Scheike
**Examples**

```r
### getting some rates to mimic

data(base1cumhaz)
data(base4cumhaz)
data(drcumhaz)
dr <- drcumhaz
base1 <- base1cumhaz
base4 <- base4cumhaz

### simulating simple model that mimicks data
### now with two event types and second type has same rate as death rate

rr <- simRecurrenceII(1000, base1, dr, death.cumhaz = base4)
rr <- count.history(rr)
dtable(rr, ~ Count + status, level = 1)
```

---

**covarianceRecurrent**

**Estimation of covariance for bivariate recurrent events with terminal event**

**Description**

Estimation of probability of more than \( k \) events for recurrent events process where there is terminal event.

**Usage**

```r
covarianceRecurrent(data, type1, type2, status = "status", death = "death", start = "start", stop = "stop", id = "id", names.count = "Count")
```

**Arguments**

- **data**: data-frame
- **type1**: type of first event (code) related to status
- **type2**: type of second event (code) related to status
- **status**: name of status
- **death**: name of death indicator
- **start**: start stop call of `Hist()` of `prodlim`
- **stop**: start stop call of `Hist()` of `prodlim`
- **id**: id
- **names.count**: name of count for number of previous event of different types, here generated by `count.history()`
Author(s)

Thomas Scheike

References

Scheike, Eriksson, Tribler (2018) The mean, variance and correlation for bivariate recurrent events with a terminal event, work in progress

Examples

```r
# getting some data to work on
data(base1cumhaz)
data(base4cumhaz)
data(drcumhaz)
dr <- drcumhaz
base1 <- base1cumhaz
base4 <- base4cumhaz
rr <- simRecurrent(1000,base1,cumhaz2=base4,death.cumhaz=dr)
rr <- count.history(rr)
rr$strata <- 1
dtable(rr,~death+status)

covrp <- covarianceRecurrent(rr,1,2,status="status",death="death",
                              start="entry",stop="time",id="id",names.count="Count")
par(mfrow=c(1,3))
plot(covrp)

### with strata, each strata in matrix column, provides basis for fast Bootstrap

covrps <- covarianceRecurrent$rr,1,2,status="status",death="death",
          start="entry",stop="time",strata="strata",id="id",names.count="Count")
```

daggregate

aggregating for for data frames

Description

aggregating for for data frames

Usage

daggregate(data, y = NULL, x = NULL, subset, ..., fun = "summary",
          regex = mets.options$regex, missing = FALSE, remove.empty = FALSE,
          matrix = FALSE, silent = FALSE, na.action = na.pass, convert = NULL)
Arguments

data    data.frame
y        name of variable, or formula, or names of variables on data frame.
x        name of variable, or formula, or names of variables on data frame.
subset  subset expression
...     additional arguments to lower level functions
fun      function defining aggregation
regex    interpret x,y as regular expressions
missing  Missing used in groups (x)
remove.empty  remove empty groups from output
matrix   if TRUE a matrix is returned instead of an array
silent   suppress messages
na.action How model.frame deals with 'NA's
convert  if TRUE try to coerce result into matrix. Can also be a user-defined function

Examples

data("sTRACE", package="timerreg")
daggregate(iris, ".'.e.al", x="Species", fun=cor, regex=TRUE)
daggregate(iris, Sepal.Length+Petal.Length ~Species, fun=summary)
daggregate(iris, log(Sepal.Length)+I(Petal.Length>1.5) ~ Species, fun=summary)
daggregate(iris, "'xLength'*", x="Species", fun=head)
daggregate(iris, "'.e.al", x="Species", fun=tail, regex=TRUE)
ddaggerate(sTRACE, status ~ diabetes, fun=table)
ddaggerate(sTRACE, status ~ diabetes+sex, fun=table)
ddaggerate(sTRACE, status + diabetes+sex ~ vf+I(wmi>1.4), fun=table)
ddaggerate(iris, "'.'e.al", x="Species",regex=TRUE)
dlist(iris,Petal.Length+Sepal.Length ~ Species | Petal.Length>1.3 & Sepal.Length>5,
n=list(1:3,3:1)))
daggregate(iris, I(Sepal.Length>7)~Species | I(Petal.Length>1.5))
daggregate(iris, I(Sepal.Length>7)~Species | I(Petal.Length>1.5), fun=table)

dsum(iris, .~Species, matrix=TRUE, missing=TRUE)

par(mfrow=c(1,2))
data(iris)
rename(iris) <- ~.
daggregate(iris,'sepal*~species|species!="virginica",fun=plot)
daggregate(iris,'sepal*~I(as.numeric(species))|I(as.numeric(species))!=1,fun=summary)

dnumeric(iris) <- ~species
aggregate(iris,'sepal*~species.n|species.n!=1,fun=summary)
Derivatives of the bivariate normal cumulative distribution function

Description

Derivatives of the bivariate normal cumulative distribution function

Usage

\[
\text{dbvn}(p, \text{design} = \text{function}(p, \ldots) \{ \\
\text{return(list(mu = cbind(p[1], p[1]),} \\
\text{dmu = cbind(1, 1),} \\
S = \text{matrix(c(p[2], p[3], p[3], p[4]), ncol = 2),} \\
\text{dS = rbind(c(1, 0, 0, 0), c(0, 1, 1, 0), c(0, 0, 0, 1)) \}),} \\
Y = \text{cbind(0, 0))}
\]

Arguments

- \text{p}
  - Parameter vector
- \text{design}
  - Design function with defines mean, derivative of mean, variance, and derivative of variance with respect to the parameter \( p \)
- \text{Y}
  - Column vector where the CDF is evaluated

Author(s)

Klaus K. Holst

Calculate summary statistics grouped by

Description

Calculate summary statistics grouped by variable

Usage

\[
\text{dby(data, INPUT, \ldots, ID = NULL, ORDER = NULL, SUBSET = NULL, SORT = 0,} \\
\text{COMBINE = !REDUCE, NOCHECK = FALSE, ARGS = NULL, NAMES,} \\
\text{COLUMN = FALSE, REDUCE = FALSE, REGEX = mets.options()$regex,} \\
\text{ALL = TRUE)}
\]
Arguments

- **data**: Data.frame
- **INPUT**: Input variables (character or formula)
- **ID**: id variable
- **ORDER**: (optional) order variable
- **SUBSET**: (optional) subset expression
- **SORT**: sort order (id+order variable)
- **COMBINE**: If TRUE result is appended to data
- **NOCHECK**: No sorting or check for missing data
- **ARGS**: Optional list of arguments to functions (...)
- **NAMES**: Optional vector of column names
- **COLUMN**: If TRUE do the calculations for each column
- **REDUCE**: Reduce number of redundant rows
- **REGEX**: Allow regular expressions
- **ALL**: if FALSE only the subset will be returned

Details

Calculate summary statistics grouped by
dby2 for column-wise calculations

Author(s)

Klaus K. Holst and Thomas Scheike

Examples

```r
n <- 4
k <- c(3, rbinom(n-1, 3, 0.5)+1)
N <- sum(k)
d <- data.frame(y=rnorm(N), x=rnorm(N), id=rep(n, k), num=unlist(sapply(k, seq)))
d2 <- d[sample(nrow(d), )]

dby(d2, y~id, mean)
dby(d2, y~id + order(num), cumsum)

dby(d, y~id + order(num), dlag)
dby(d, y~id + order(num), dlag, ARGS=list(k=1:2))
dby(d, y~id + order(num), dlag, ARGS=list(k=1:2), NAMES=c("l1", "l2"))

dby(d, y~id + order(num), mean=mean, csum=cumsum, n=length)
dby(d2, y~id + order(num), a=cumsum, b=mean, N=length, l1=function(x) c(NA, x)[-length(x)])

dby(d, y~id + order(num), nn=seq_along, n=length)
```
dcor

summary, tables, and correlations for data frames

dby(d, y~id + order(num), nn=seq_along, n=length)

d <- d[1:4]
dby(d, x<0) <- list(z=mean)
d <- dby(d, is.na(z), z=1)

f <- function(x) apply(x,1,min)
dby(d, y+x~id, min=f)

dby(d, y+x~id+order(num), function(x) x)

f <- function(x) (cbind(cumsum(x[,1]),cumsum(x[,2]))/sum(x))
dby(d, y+x~id, f)

## column-wise
a <- d
dby2(a, mean, median, REGEX=TRUE) <- '^[y|x]'-id
a

## wildcards
a <- dby2(a,'y*'+x~id,mean)

## subset
a <- dby(d, x<0) <- list(z=NA)
d
a <- dby(d, y~id|x<-1, v=mean,z=1)
dby(d, y+x~id|x<-1, mean, median, COLUMN=TRUE)

dby2(d, y+x~id|x>0, mean, REDUCE=TRUE)

dby(d, y~id|x<0,mean,ALL=FALSE)

a <- iris
a <- dby(a,x=1)
dby(a,Species=="versicolor") <- list(y=2)
Arguments

- **data**: if `x` is formula or names for data frame then data frame is needed.
- **y**: name of variable, or formula, or names of variables on data frame.
- **x**: possible group variable
- **use**: how to handle missing values
- **...**: Optional additional arguments

Author(s)

Klaus K. Holst and Thomas Scheike

Examples

```r
data("sTRACE", package="timereg")
dt<- sTRACE
dt$time2 <- dt$time^2
dt$wmi2 <- dt$wmi^2
head(dt)

dcor(dt)

dcor(dt,"time+wmi")
dcor(dt,"time+wmi","vf+chf")
dcor(dt,"time+wmi","vf+chf")
dcor(dt,c("time","wmi","vf+chf"))
```

Description

Cut variables, if breaks are given these are used, otherwise cuts into using group size given by probs, or equispace groups on range. Default is equally sized groups if possible

Usage

```r
dcut(data, y = NULL, x = NULL, breaks = 4, probs = NULL, equi = FALSE,
     regex = mets.options()
\$regex, sep = NULL, na.rm = TRUE, labels = NULL, all = FALSE, ...)
```
Arguments

data if x is formula or names for data frame then data frame is needed.
y name of variable, or formula, or names of variables on data frame.
x name of variable, or formula, or names of variables on data frame.
breaks number of breaks, for variables or vector of break points,
probs groups defined from quantiles
equi for equi-spaced breaks
regex for regular expressions.
sep seperator for naming of cut names.
na.rm to remove NA for grouping variables.
labels to use for cut groups
all to do all variables, even when breaks are not unique
... Optional additional arguments

Author(s)

Klaus K. Holst and Thomas Scheike

Examples

data("sTRACE", package="timereg")
sTRACE$age2 <- sTRACE$age^2
sTRACE$age3 <- sTRACE$age^3

mm <- dcut(sTRACE,-age+wmi)
head(mm)

mm <- dcut(sTRACE,catage4+wmi4-age+wmi)
head(mm)

mm <- dcut(sTRACE,-age+wmi,breaks=c(2,4))
head(mm)

mm <- dcut(sTRACE,c("age","wmi"))
head(mm)

mm <- dcut(sTRACE,-.)
head(mm)

mm <- dcut(sTRACE,c("age","wmi"),breaks=c(2,4))
head(mm)

gx <- dcut(sTRACE$age)
head(gx)

## Removes all cuts variables with these names wildcards
### dermalridges

Data on dermal ridge counts in left and right hand in (nuclear) families

<table>
<thead>
<tr>
<th>dermalridges</th>
<th>Dermal ridges data (families)</th>
</tr>
</thead>
</table>

#### Description

Data on dermal ridge counts in left and right hand in (nuclear) families

#### Format

Data on 50 families with ridge counts in left and right hand for mother, father and each child. Family id in 'family' and gender and child number in 'sex' and 'child'.

### R code

```r
mm1 <- DRM(mm, c("*2", "*4"))
head(mmm)

## Wildcards, for age, age2, age4 and wmi
head(dcut(mm, c("a*", "?m*")))

## With direct assignment
drm(mm) <- c("*2", "*4")
head(mmm)

dcut(mm) <- c("age","*m")
dcut(mm) <- age1+wmi1-age+wmi
head(mmm)

###############################################
## Renaming
###############################################

head(mmm)
drename(mm, ~Age+Wmi) <- c("wmi","age")
head(mmm)
mm1 <- mm

## All names to lower
rename(mm1) <- ~.
head(mm1)

## A* to lower
mm2 <- drename(mm, c("A*","W*"))
head(mmm2)
drename(mm) <- "A*"
head(mmm)

dd <- data.frame(A_1=1:2,B_1=1:2)
funn <- function(x) gsub("_",".*",x)
drename(dd) <- ~.
drename(dd, fun=funn) <- ~.
names(dd)
```
Source


Examples

data(dermalridges)
fast.reshape(dermalridges, id="family", varying=c("child.left", "child.right", "sex"))

dermalridgesMZ  Dermal ridges data (monozygotic twins)

Description

Data on dermal ridge counts in left and right hand in (nuclear) families

Format

Data on dermal ridge counts (left and right hand) in 18 monozygotic twin pairs.

Source


Examples

data(dermalridgesMZ)
fast.reshape(dermalridgesMZ, id="id", varying=c("left", "right"))

divide.conquer  Split a data set and run function

Description

Split a data set and run function

Usage

divide.conquer(func = NULL, data, size, splits, id = NULL, ...)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>func</td>
<td>called function</td>
</tr>
<tr>
<td>data</td>
<td>data-frame</td>
</tr>
<tr>
<td>size</td>
<td>size of splits</td>
</tr>
<tr>
<td>splits</td>
<td>number of splits (ignored if size is given)</td>
</tr>
<tr>
<td>id</td>
<td>optional cluster variable</td>
</tr>
<tr>
<td>...</td>
<td>Additional arguments to lower level functions</td>
</tr>
</tbody>
</table>

Author(s)

Thomas Scheike, Klaus K. Holst

Examples

```r
library(timereg)
data(TRACE)
res <- divide.conquer(prop.odds, TRACE,
  formula=Event(time, status==9)-chf+vf+age, n.sim=0, size=200)
```

divide.conquer.timereg

*Split a data set and run function from timereg and aggregate*

Description

Split a data set and run function of cox-aalen type and aggregate results

Usage

`divide.conquer.timereg(func = NULL, data, size, ...)`

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>func</td>
<td>called function</td>
</tr>
<tr>
<td>data</td>
<td>data-frame</td>
</tr>
<tr>
<td>size</td>
<td>size of splits</td>
</tr>
<tr>
<td>...</td>
<td>Additional arguments to lower level functions</td>
</tr>
</tbody>
</table>

Author(s)

Thomas Scheike, Klaus K. Holst


Examples

```r
dlibrary(timereg)
data(TRACE)
a <- divide.conquer.timereg(prop.odds, TRACE,
formula=Event(time, status==9)~chf+vf+age, n.sim=0, size=200)
coef(a)
a2 <- divide.conquer.timereg(prop.odds, TRACE,
formula=Event(time, status==9)~chf+vf+age, n.sim=0, size=500)
coef(a2)

if (interactive()) {
par(mfrow=c(1,1))
plot(a, xlim=c(0,8), ylim=c(0,0.01))
par(new=TRUE)
plot(a2, xlim=c(0,8), ylim=c(0,0.01))
}
```

---

**dlag**

**Lag operator**

**Description**

Lag operator

**Usage**

```r
dlag(data, x, k = 1, combine = TRUE, simplify = TRUE, names, ...)
```

**Arguments**

- `data` : data.frame or vector
- `x` : optional column names or formula
- `k` : lag (vector of integers)
- `combine` : combine results with original data.frame
- `simplify` : Return vector if possible
- `names` : optional new column names
- `...` : additional arguments to lower level functions

**Examples**

```r
d <- data.frame(y=1:10, x=c(10:1))
dlag(d, k=1:2)
dlag(d, ~x, k=0:1)
dlag(d$x, k=1)
dlag(d$x, k=-1:2, names=letters[1:4])
```
dprint

list, head, print, tail

Description

listing for data frames

Usage

dprint(data, y = NULL, n = 0, ..., x = NULL)

Arguments

data
if x is formula or names for data frame then data frame is needed.
y
name of variable, or formula, or names of variables on data frame.
n
Index of observations to print (default c(1:nfirst, n-nlast:nlast)
...
Optional additional arguments (nfirst, nlast, and print options)
x
possible group variable

Author(s)

Klaus K. Holst and Thomas Scheike

Examples

n <- 20
m <- lava::lvm(letters)
d <- lava::sim(m,n)

dlist(d,-a+b+c)
dlist(d,-a+b+c|a<0 & b>0)
## listing all :
dlist(d,-a+b+c|a<0 & b>0,n=0)
dlist(d,a+b+c-I(d>0)|a<0 & b>0)
dlist(d,.-I(d>0)|a<0 & b>0)
dlist(d,-a+b+c|a<0 & b>0, nlast=0)
dlist(d,-a+b+c|a<0 & b>0, nfirst=3, nlast=3)
dlist(d,-a+b+c|a<0 & b>0, 1:5)
dlist(d,-a+b+c|a<0 & b>0, -(5:1))
dlist(d,-a+b+c|a<0 & b>0, list(1:5,50:55,(-(5:1)))
dprint(d,a+b+c ~ I(d>0) |a<0 & b>0, list(1:5,50:55,(-(5:1)))
drcumhaz

Rate for leaving HPN program for patients of Copenhagen

Description

Rate for leaving HPN program for patients of Copenhagen

Source

Estimated data

dreg

Regression for data frames with dutility call

Description

Regression for data frames with dutility call

Usage

dreg(data, y, x = NULL, z = NULL, x.oneatitime = TRUE,
   x.base.names = NULL, z.arg = c("clever", "base", "group", "condition"),
   fun. = lm, summary. = summary, regex = FALSE, convert = NULL,
   special = NULL, equal = TRUE, test = 1, ...)

Arguments

data   data frame
y      name of variable, or formula, or names of variables on data frame.
x      name of variable, or formula, or names of variables on data frame.
z      name of variable, or formula, or names of variables on data frame.
x.oneatitime  x's one at a time
x.base.names   base covarirates
z.arg         what is Z
fun.          function
summary.      summary to use
regex         regex
convert       convert
special       special's
equal         to do pairwise stuff
test          development argument
...            Additional arguments for fun
Author(s)
Klaus K. Holst, Thomas Scheike

Examples

```r
data(iris)
data=iris
drename(iris) <- ~
names(iris)
iris$time <- runif(nrow(iris))
iris$time1 <- runif(nrow(iris))
iris$status <- rbinom(nrow(iris),1,0.5)
iris$id1 <- with(iris,Surv(time,status))
iris$id2 <- with(iris,Surv(time1,status))
iris$id <- 1:nrow(iris)

mm <- dreg(iris,"*\.length"="*\.width"|I(species="setosa" & status==1))
mm <- dreg(iris,"*\.length"="*\.width"|species=status)
mm <- dreg(iris,"*\.length"="*\.width"|species)
mm <- dreg(iris,"*\.length"="*\.width"|species=status,z.arg="group")

### Reduce Ex.Timings
y <- "S*"="*\.width"
xr <- dreg(iris,y,fun=phreg)
xr <- dreg(iris,y,fun=survdiff)

### testing forskellige calls
y <- "S*"="*\.width"
xr <- dreg(iris,y,x,oneatime=FALSE,fun=phreg)

### with condition
y <- S1="*\.width"|I(species="setosa" & sepal.width>3)
xr <- dreg(iris,y,z.arg="condition",fun=phreg)
xr <- dreg(iris,y,fun=phreg)

### with baseline after |
y <- S1="*\.width"|sepal.length
xr <- dreg(iris,y,fun=phreg)

### by group by species, not working
y <- S1="*\.width"|species
ss <- split(iris,paste(iris$species,iris$status))
xr <- dreg(iris,y,fun=phreg)
```

## species as base, species is factor so assumes that this is grouping
y <- S1="* .width"|species
xs <- dreg(iris,y,z.arg="base",fun.=phreg)

## background var after | and then one of x's at at time
y <- S1="* .width"|status+"sepal"
xs <- dreg(iris,y,fun.=phreg)

## background var after | and then one of x's at at time
y <- S1="* .width"|status+"sepal"
xs <- dreg(iris,y,x.oneatatime=FALSE,fun.=phreg)
xs <- dreg(iris,y,fun.=phreg)

## background var after | and then one of x's at at time
y <- S1="* .width"+factor(species)
xs <- dreg(iris,y,fun.=phreg)
xs <- dreg(iris,y,fun.=phreg,x.oneatatime=FALSE)

y <- S1="* .width"|factor(species)
xs <- dreg(iris,y,z.arg="base",fun.=phreg)

y <- S1="* .width"|cluster(id)+factor(species)
xs <- dreg(iris,y,z.arg="base",fun.=phreg)
xs <- dreg(iris,y,z.arg="base",fun.=coxph)

## under condition with groups
y <- S1="* .width"|I(sepal.length>4)
xs <- dreg(subset(iris,species=="setosa"),y,z.arg="group",fun.=phreg)

## under condition with groups
y <- S1="* .width"+I(log(sepal.length))|I(sepal.length>4)
xs <- dreg(subset(iris,species=="setosa"),y,z.arg="group",fun.=phreg)

y <- S1="* .width"+I(dcut(sepal.length))|I(sepal.length>4)
xs <- dreg(subset(iris,species=="setosa"),y,z.arg="group",fun.=phreg)

ff <- function(formula,data,...) {
  ss <- survfit(formula,data,...)
  kmplot(ss,...)
  return(ss)
}

dcut(iris) <- "% .width"

y <- S1="*.4"|I(sepal.length>4)
par(mfrow=c(1,2))
xs <- dreg(iris,y,fun.=ff)
**Description**

levels shows levels for variables in data frame. `relevel` relevels a factor in data frame.

**Usage**

```r
drelevel(data, y = NULL, x = NULL, ref = NULL, newlevels = NULL,
         regex = mets.options()$regex, sep = NULL, overwrite = FALSE, ...)
```

**Arguments**

- **data**: if `x` is formula or names for data frame then data frame is needed.
- **y**: name of variable, or formula, or names of variables on data frame.
- **x**: name of variable, or formula, or names of variables on data frame.
- **ref**: new reference variable
- **newlevels**: to combine levels of factor in data frame
- **regex**: for regular expressions.
- **sep**: separator for naming of cut names.
- **overwrite**: to overwrite variable
- **...**: Optional additional arguments

**Author(s)**

Klaus K. Holst and Thomas Scheike

**Examples**

```r
data(mena)
dstr(mena)
dfactor(mena) <- ~twinnum
dnumeric(mena) <- ~twinnum.f

dstr(mena)

mena2 <- drelevel(mena,"cohort",ref="(1980,1982]")
mena2 <- drelevel(mena,"cohort",ref="(1980,1982]")
mena2 <- drelevel(mena,cohortII~cohort,ref="(1980,1982]")
dlevels(mena)
dlevels(mena2)
drelevel(mena,ref="(1975,1977]" ) <- ~cohort
drelevel(mena,ref="(1980,1982]" ) <- ~cohort
dlevels(mena,"coh*")
dtable(mena,"coh",level=1)

### level 1 of zyg as baseline for new variable

drelevel(mena,ref=1) <- ~zyg

drelevel(mena,ref=c("DZ","[1973,1975]")) <- ~zyg+cohort
drelevel(mena,ref=c("DZ","[1973,1975]")) <- zyg*+cohort.early~ zyg*+cohort
```
### level 2 of zyg and cohort as baseline for new variables

drelevel(mena, ref = 2) <- zyg + cohort
dlevels(mena)

#############################################################
# combining factor levels with newlevels argument

dcut(mena, labels = c("I", "II", "III", "IV")) <- cat4~agemena
dlevels(drelevel(mena, cat4, newlevels=1:3))
dlevels(drelevel(mena, ncat4~cat4, newlevels=3:2))
drelevel(mena, newlevels=3:2) <- ncat4~cat4
dlevels(mena)
dlevels(drelevel(mena, nca4~cat4, newlevels=list(c(1,4),2:3)))
drelevel(mena, newlevels=list(c(1,4),2:3)) <- nca4..2~cat4
dlevels(mena)
drelevel(mena, newlevels=list("I-III"=c("I","II","III"), "IV"="IV")) <- nca4..3~cat4
dlevels(mena)
drelevel(mena, newlevels=list("I-III"=c("I","II","III"))) <- nca4..4~cat4
dlevels(mena)
drelevel(mena, newlevels=list(group1=c("I","II","III"))) <- nca4..5~cat4
dlevels(mena)
drelevel(mena, newlevels=list(g1=c("I","II","III"), g2="IV")) <- nca4..6~cat4
dlevels(mena)

dsort

**Sort data frame**

**Description**

Sort data according to columns in data frame

**Usage**

dsort(data, x, ..., decreasing = FALSE, return.order = FALSE)

**Arguments**

data Data frame
x variable to order by
... additional variables to order by
decreasing sort order (vector of length x)
return.order return order
dspline

Value
data.frame

Examples

```r
data(data="hubble",package="lava")
dspline(data,"sigma")
dspline(hubble, hubble$sigma,"v")
dspline(hubble,-sigma+v)
dspline(hubble,-sigma-v)

## with direct assignment
dspline(hubble) <- -sigma-v
```

dpsline

Simple linear spline

Description

Constructs simple linear spline on a data frame using the formula syntax of dutils

Usage

```r
dpsline(data, y = NULL, x = NULL, breaks = 4, probs = NULL,
equi = FALSE, regex = mets.options()$regex, sep = NULL, na.rm = TRUE,
labels = NULL, all = FALSE, ...)
```

Arguments

data if x is formula or names for data frame then data frame is needed. y name of variable, or formula, or names of variables on data frame. x name of variable, or formula, or names of variables on data frame. breaks number of breaks, for variables or vector of break points, probs groups defined from quantiles equi for equi-spaced breaks regex for regular expressions. sep separator for naming of cut names. na.rm to remove NA for grouping variables. labels to use for cut groups all to do all variables, even when breaks are not unique ... Optional additional arguments

Author(s)

Thomas Scheike
Examples

data(TRACE)
TRACE <- dspline(TRACE,-wmi,breaks=c(1,1.3,1.7))
cca <- coxph(Surv(time,status==9)-age+vf+chf+wmi,data=TRACE)
cca2 <- coxph(Surv(time,status==9)-age+wmi+vf+chf+wmi.spline1+wmi.spline2+wmi.spline3,data=TRACE)
anova(cca,cca2)

nd=data.frame(age=50,vf=0,chf=0,wmi=seq(0.4,3,by=0.01))
nd <- dspline(nd,-wmi,breaks=c(1,1.3,1.7))
pl <- predict(cca2,newdata=nd)
plot(nd$wmi,pl,type="l")

---

dtable

Tables for data frames

Description

tables for data frames

Usage

dtable(data, y = NULL, x = NULL, ..., level = -1, response = NULL, flat = TRUE, total = FALSE, prop = FALSE, summary = NULL)

Arguments

data if x is formula or names for data frame then data frame is needed.
y name of variable, or formula, or names of variables on data frame.
x name of variable, or formula, or names of variables on data frame.
... Optional additional arguments
level 1 for all marginal tables, 2 for all 2 by 2 tables, and null for the full table, possible versus group variable
response For level=2, only produce tables with columns given by 'response' (index)
flat produce flat tables
total add total counts/proportions
prop Proportions instead of counts (vector of margins)
summary summary function

Author(s)
Klaus K. Holst and Thomas Scheike
dtransform

Transform that allows condition

Description

Defines new variables under condition for data frame

Usage

dtransform(data, ...)

Arguments

data is data frame

... new variable definitions including possible if condition
easy.binomial.twostage

Fits two-stage binomial for describing dependence in binomial data using marginals that are on logistic form using the binomial.twostage function, but call is different and easier and the data manipulation is build into the function. Useful in particular for family design data.

Description

If clusters contain more than two times, the algorithm uses a compososite likelihood based on the pairwise bivariate models.

Usage

easy.binomial.twostage(margbin = NULL, data = sys.parent(),
   score.method = "fisher.scoring", response = "response", id = "id",
   Nit = 60, detail = 0, silent = 1, weights = NULL, control = list(),
   theta = NULL, theta.formula = NULL, desnames = NULL, deshelp = 0,
   var.link = 1, iid = 1, step = 1, model = "plackett",
   marginal.p = NULL, strata = NULL, max.clust = NULL,
   se.clusters = NULL)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>margbin</td>
<td>Marginal binomial model</td>
</tr>
<tr>
<td>data</td>
<td>data frame</td>
</tr>
<tr>
<td>score.method</td>
<td>Scoring method</td>
</tr>
<tr>
<td>response</td>
<td>name of response variable in data frame</td>
</tr>
<tr>
<td>id</td>
<td>name of cluster variable in data frame</td>
</tr>
<tr>
<td>Nit</td>
<td>Number of iterations</td>
</tr>
<tr>
<td>detail</td>
<td>Detail for more output for iterations</td>
</tr>
<tr>
<td>silent</td>
<td>Debug information</td>
</tr>
<tr>
<td>weights</td>
<td>Weights for log-likelihood, can be used for each type of outcome in 2x2 tables.</td>
</tr>
<tr>
<td>control</td>
<td>Optimization arguments</td>
</tr>
<tr>
<td>theta</td>
<td>Starting values for variance components</td>
</tr>
</tbody>
</table>
The reported standard errors are based on the estimated information from the likelihood assuming that the marginals are known. This gives correct standard errors in the case of the Plackett distribution (OR model for dependence), but incorrect for the Clayton-Oakes types model. The OR model is often known as the ALR model. Our fitting procedures gives correct standard errors due to the orthogonality and is fast.

Examples

data(twinstut)
twinstut0 <- subset(twinstut, tvparnr<2300000)
twinstut <- twinstut0
twinstut$binstut <- (twinstut$stutter="yes")*1
theta.des <- model.matrix(~-1+factor(zyg), data=twinstut)
margbin <- glm(binstut~factor(sex)+age, data=twinstut, family=binomial())
bin <- binomial.twostage(margbin, data=twinstut, var.link=1,
                         clusters=twinstut$tvparnr, theta.des=theta.des, detail=0,
                         score.method="fisher.scoring")
summary(bin)
lava::estimate(coef=bin$theta, vcov=bin$var.theta, f=function(p) exp(p))

twinstut$age <- scale(twinstut$age)
theta.des <- model.matrix(~-1+factor(zyg)+factor(zyg)*cage, data=twinstut)
bina <- binomial.twostage(margbin, data=twinstut, var.link=1,
                         clusters=twinstut$tvparnr, theta.des=theta.des, detail=0)
summary(bina)

theta.des <- model.matrix(~-1+factor(zyg)+factor(zyg)*factor(zyg)*cage, data=twinstut)
bina <- binomial.twostage(margbin, data=twinstut, var.link=1,
                         clusters=twinstut$tvparnr, theta.des=theta.des)
summary(bina)

out <- easy.binomial.twostage(stutter~factor(sex)+age, data=twinstut,
                               response="binstut", id="tvparnr", var.link=1,
theta.formula<-1+factor(zyg1))
summary(out)

## refers to zygosity of first subject in each pair : zyg1
## could also use zyg2 (since zyg2=zyg1 within twinpair's))
desfs <- function(x,num1="zyg1",namesdes=c("mz","dz","os"))
c(x[num1]=="mz",x[num1]=="dz",x[num1]=="os")*1

out3 <- easy.binomial.twostage(binstut=factor(sex)+age,
data=twinstut, response="binstut",id="tvparnr",
var.link=1,theta.formula=desfs,
desnames=c("mz","dz","os"))
summary(out3)

## Reduce Ex.Timings
n <- 10000
set.seed(100)
dd <- simBinFam(n,beta=0.3)
binfam <- fast.reshape(dd,varying=c("age","x","y"))
## mother, father, children (ordered)
head(binfam)

simple analyses of binomial family data

##
## desfs <- function(x,num1="num1",num2="num2")
{
  pp <- 1*(((x[num1]=="m")*(x[num2]=="f"))|(x[num1]=="f")*(x[num2]=="m"))
  pc <- (x[num1]=="m" | x[num1]=="f")*(x[num2]=="b1" | x[num2]=="b2"))*1
  cc <- (x[num1]=="b1")*(x[num2]=="b1" | x[num2]=="b2")*1
  c(pp,pc,cc)
}

ud <- easy.binomial.twostage(y~1,data=binfam,
  response="y",id="id",
  theta.formula=desfs,desnames=c("pp","pc","cc"))
summary(ud)

udx <- easy.binomial.twostage(y~x,data=binfam,
  response="y",id="id",
  theta.formula=desfs,desnames=c("pp","pc","cc"))
summary(udx)

##
## now allowing parent child POR to be different for mother and father

##
## desfsi <- function(x,num1="num1",num2="num2")
{
  pp <- (x[num1]=="m")*(x[num2]=="f")*1
  mc <- (x[num1]=="m")*(x[num2]=="b1" | x[num2]=="b2")*1
  fc <- (x[num1]=="f")*(x[num2]=="b1" | x[num2]=="b2")*1
  cc <- (x[num1]=="b1")*(x[num2]=="b1" | x[num2]=="b2")*1


easy.survival.twostage

Wrapper for easy fitting of Clayton-Oakes or bivariate Plackett models for bivariate survival data
Description

Fits two-stage model for describing dependence in survival data using marginals that are on cox or aalen form using the twostage function, but call is different and easier and the data manipulation build into the function. Useful in particular for family design data.

Usage

easy.survival.twostage(margsurv = NULL, data = sys.parent(),
score.method = "nlminb", status = "status", time = "time",
entry = NULL, id = "id", Nit = 60, detail = 0, silent = 1,
weights = NULL, control = list(), theta = NULL, theta.formula = NULL,
desnames = NULL, deshelp = list(), theta.formula = NULL,
model = "plackett", marginal.surv = NULL, strata = NULL,
max.clust = NULL, se.clusters = NULL)

Arguments

margsurv  model
data  data frame
score.method  Scoring method
status  Status at exit time
time  Exit time
entry  Entry time
id  name of cluster variable in data frame
Nit  Number of iterations
detail  Detail for more output for iterations
silent  Debug information
weights  Weights for log-likelihood, can be used for each type of outcome in 2x2 tables.
control  Optimization arguments
theta  Starting values for variance components
theta.formula  design for dependence, either formula or design function
desnames  names for dependence parameters
deshelp  if 1 then prints out some data sets that are used, on on which the design function operates
var.link  Link function for variance (exp link)
iid  Calculate i.i.d. decomposition
step  Step size for newton-raphson
model  plackett or clayton-oakes model
marginal.surv  vector of marginal survival probabilities
strata  strata for fitting
max.clust  max clusters
se.clusters  clusters for iid decomposition for roubst standard errors
Details

If clusters contain more than two times, the algorithm uses a composite likelihood based on the pairwise bivariate models.

The reported standard errors are based on the estimated information from the likelihood assuming that the marginals are known.

Examples

```r
library("timereg")
library("survival")
data("prt",package="mets")
margp <- coxph(Surv(time,status==1)~factor(country),data=prt)
fitco <- survival.twostage(margp,data=prt,clusters=prt$id)
summary(fitco)

des <- model.matrix(~-1+factor(zyg),data=prt);
fitco <- survival.twostage(margp,data=prt,theta.des=des,clusters=prt$id)
summary(fitco)

dfam <- simSurvFam(1000)
dfam <- fast.reshape(dfam,vars=c("x","time","status"))

desfs <- function(x,num1="num1",num2="num2") {
  pp <- (x[num1]=="m")*(x[num2]=="f")*1  ## mother-father
  pc <- (x[num1]=="m" | x[num1]=="f")*(x[num2]=="b1" | x[num2]=="b2")*1  ## mother-child
  cc <- (x[num1]=="b1")*(x[num2]=="b1" | x[num2]=="b2")*1  ## child-child
  c(pp,pc,cc)
}
marg <- coxph(Surv(time,status)~factor(num),data=dfam)
out3 <- easy.survival.twostage(marg,data=dfam,time="time",status="status",id="id",deshelp=0,
score.method="fisher.scoring",theta.formula=desfs,
desnames=c("parent-parent","parent-child","child-child"))
summary(out3)
```

---

**Description**

Computes the relative risk for additive gamma model at time 0

**Usage**

`EVaddGam(theta, x1, x2, thetades, ags)`
Arguments

theta theta
x1 x1
x2 x2
thetades thetades
ags ags

Author(s)

Thomas Scheike

References


Examples

lam0 <- c(0.5, 0.3)
pars <- c(1,1,1,1,0,1)
## genetic random effects, cause1, cause2 and overall
parg <- pars[c(1,3,5)]
## environmental random effects, cause1, cause2 and overall
parc <- pars[c(2,4,6)]

## simulate competing risks with two causes with hazards 0.5 and 0.3
## ace for each cause, and overall ace
out <- simCompete.twin.ace(10000, parg, parc, 0, 2, lam0=lam0, overall=1, all.sum=1)

## setting up design for running the model
mm <- familycluster.index(out$cluster)
head(mm$familypairindex, n=10)
pairs <- matrix(mm$familypairindex, ncol=2, byrow=TRUE)
tail(pairs, n=12)

# kinship <- (out[pairs[,1],"zyg"]=='MZ') + (out[pairs[,1],"zyg"]=='DZ')*0.5

# dout <- make.pairwise.design.competing(pairs, kinship,
# type="ace", compete=length(lam0), overall=1)
# head(dout$ant.rvs)

## MZ
## dim(dout$theta.des)
## dout$random.design[,1]
## DZ
## dout$theta.des[,nrow(pairs)]
## dout$random.design[,nrow(pairs)]
#
# thetades <- dout$theta.des[,1]
## x <- dout$random.design[,1]
## x
eventpois

Extract survival estimates from lifetable analysis

Description

Summary for survival analyses via the 'lifetable' function

Usage

```r
eventpois(object, ..., timevar, time, int.len, confint = FALSE, level = 0.95, individual = FALSE, length.out = 25)
```

Arguments

- **object**: glm object (poisson regression)
- **...**: Contrast arguments
- **timevar**: Name of time variable
- **time**: Time points (optional)
- **int.len**: Time interval length (optional)
- **confint**: If TRUE confidence limits are supplied
- **level**: Level of confidence limits
- **individual**: Individual predictions
- **length.out**: Length of time vector

Details

Summary for survival analyses via the 'lifetable' function

Author(s)

Klaus K. Holst
familycluster.index  
Finds all pairs within a cluster (family)

Description
Finds all pairs within a cluster (family)

Usage
familycluster.index(clusters, index.type = FALSE, num = NULL, Rindex = 1)

Arguments
- clusters: list of indices
- index.type: argument of cluster index
- num: num
- Rindex: index starts with 1 in R, and 0 in C

Author(s)
Klaus Holst, Thomas Scheike

References
Cluster indeces

See Also
cluster.index familyclusterWithProbands.index

Examples
i<-c(1,1,2,2,1,3)
d<- familycluster.index(i)
print(d)
familyclusterWithProbands.index

*Finds all pairs within a cluster (family) with the proband (case/control)*

**Description**

second column of pairs are the probands and the first column the related subjects

**Usage**

`familyclusterWithProbands.index(clusters, probands, index.type = FALSE, num = NULL, Rindex = 1)`

**Arguments**

- `clusters` list of indeces giving the clusters (families)
- `probands` list of 0,1 where 1 specifies which of the subjects that are probands
- `index.type` argument passed to other functions
- `num` argument passed to other functions
- `Rindex` index starts with 1, in C is it is 0

**Author(s)**

Klaus Holst, Thomas Scheike

**References**

Cluster indeces

**See Also**

familycluster.index cluster.index

**Examples**

```r
i<-c(1,1,2,2,1,3)
p<-c(1,0,0,1,0,1)
d<- familyclusterWithProbands.index(i,p)
print(d)
```
**fast.approx**  
*Fast approximation*

**Description**
Fast approximation

**Usage**
fast.approx(time, new.time, equal = FALSE, type = c("nearest", "right", "left"), sorted = FALSE, ...)

**Arguments**
- **time**: Original ordered time points
- **new.time**: New time points
- **equal**: If TRUE a list is returned with additional element
- **type**: Type of matching, nearest index, nearest greater than or equal (right), number of elements smaller than y otherwise the closest value above new.time is returned.
- **sorted**: Set to true if new.time is already sorted
- **...**: Optional additional arguments

**Author(s)**
Klaus K. Holst

**Examples**
```r
id <- c(1,1,2,2,7,7,10,10)
fast.approx(unique(id), id)

t <- 0:6
n <- c(-1,0,0.1,0.9,1,1.1,1.2,6,6.5)
fast.approx(t,n,type="left")
```

---

**fast.pattern**  
*Fast pattern*

**Description**
Fast pattern

**Usage**
fast.pattern(x, y, categories = 2, ...)

---
Arguments

x Matrix (binary) of patterns. Optionally if y is also passed as argument, then the pattern matrix is defined as the elements agreeing in the two matrices.

y Optional matrix argument with same dimensions as x (see above)

categories Default 2 (binary)

... Optional additional arguments

Author(s)

Klaus K. Holst

Examples

X <- matrix(rbinom(100,1,0.5),ncol=4)
fast.pattern(X)

X <- matrix(rbinom(100,3,0.5),ncol=4)
fast.pattern(X, categories=4)

Description

Fast reshape/tranpose of data

Usage

fast.reshape(data, varying, id, num, sep = "", keep, idname = "id", numname = "num", factor = FALSE, idcombine = TRUE, labelnum = FALSE, labels, regex = mets.options()$regex, dropid = FALSE, ...)

Arguments

data data.frame or matrix

varying Vector of prefix-names of the time varying variables. Optional for Long->Wide reshaping.

id id-variable. If omitted then reshape Wide->Long.

num Optional number/time variable

sep String seperating prefix-name with number/time

keep Vector of column names to keep

idname Name of id-variable (Wide->Long)

numname Name of number-variable (Wide->Long)

factor If true all factors are kept (otherwise treated as character)
idcombine If TRUE and id is vector of several variables, the unique id is combined from all the variables. Otherwise the first variable is only used as identifier.

labelnum If TRUE varying variables in wide format (going from long->wide) are labeled 1,2,3,... otherwise use 'num' variable. In long-format (going from wide->long) varying variables matching 'varying' prefix are only selected if their postfix is a number.

labels Optional labels for the number variable

regex Use regular expressions

dropid Drop id in long format (default FALSE)

Author(s)

Thomas Scheike, Klaus K. Holst

Examples

library("lava")
m <- lvm(c(y1,y2,y3,y4)-x)
d <- sim(m,5)
d
fast.reshape(d,"y")
fast.reshape(fast.reshape(d,"y"),id="id")

##### From wide-format
(dd <- fast.reshape(d,"y"))
## Same with explicit setting new id and number variable/column names
## and sepator "" (default) and dropping x
fast.reshape(d,"y",idname="a",timevar="b",sep="",keep=c())
## Same with 'reshape' list-syntax
fast.reshape(d,list(c("y1","y2","y3","y4")),labelnum=TRUE)

##### From long-format
fast.reshape(dd,id="id")
## Restrict set up within-cluster varying variables
fast.reshape(dd,"y",id="id")
fast.reshape(dd,"y",id="id",keep="x",sep=".")

#####
x <- data.frame(id=c(5,5,6,6,7),y=1:5,x=1:5,tv=c(1,2,2,1,2))
x
(xw <- fast.reshape(x,id="id"))
(xl <- fast.reshape(xw,c("y","x"),idname="id2",keep=c()))
(xl <- fast.reshape(xw,c("y","x","tv")))
(xw2 <- fast.reshape(xl,id="id",num="num"))
fast.reshape(xw2,c("y","x"),idname="id")

### more generally:
### varying=list(c("ym","yf","yb1","yb2"), c("zm","zf","zb1","zb2"))
### varying=list(c("ym","yf","yb1","yb2"))
### Family cluster example

d <- mets::simBinFam(3)
d fast.reshape(d, var="y")
fast.reshape(d, varying=list(c("ym","yf","yb1","yb2")))

d <- sim(lvm\(-y1+y2+ya\), 10)
d (dd <- fast.reshape(d,"y"))
fast.reshape(dd, "y", labelnum=TRUE)
fast.reshape(dd, id="id", num="num")
fast.reshape(dd, id="id", num="num", labelnum=TRUE)
fast.reshape(d, c(a="y"), labelnum=TRUE) ## New column name

### Unbalanced data

m <- lvm(c(y1,y2,y3,y4)~ x+z1+z3+z5)
d <- sim(m, 3)
d fast.reshape(d, c("y","z"))

### not-varying syntax:

fast.reshape(d,-c("x"))

### Automatically define varying variables from trailing digits

fast.reshape(d)

### Prostate cancer example

data(prt)
head(prtw <- fast.reshape(prt, "cancer", id="id"))
ftable(cancer1~cancer2, data=prtw)
rm(prtw)

---

**gof.phreg**

**GOF for Cox PH regression**

### Description

Cumulative score process residuals for Cox PH regression p-values based on Lin, Wei, Ying resampling.

### Usage

```
## S3 method for class 'phreg'
gof(object, n.sim = 1000, silent = 1, ...)
```
Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>is phreg object</td>
</tr>
<tr>
<td>n.sim</td>
<td>number of simulations for score processes</td>
</tr>
<tr>
<td>silent</td>
<td>to show timing estimate will be produced for longer jobs</td>
</tr>
<tr>
<td>...</td>
<td>Additional arguments to lower level functions</td>
</tr>
</tbody>
</table>

Author(s)

Thomas Scheike and Klaus K. Holst

Examples

data(TRACE)

m1 <- phreg(Surv(time, status==9)~vf+chf+diabetes, data=TRACE)
gg <- gof(m1)
par(mfrow=c(1,3))
plot(gg)

m1 <- phreg(Surv(time, status==9)~strata(vf)+chf+diabetes, data=TRACE)
gg <- gof(m1)

gofG.phreg

Stratified baseline graphical GOF test for Cox covariates in PH regression

Description

Looks at stratified baseline in Cox model and plots all baselines versus each other to see if lines are straight, with 50 resample versions under the assumption that the stratified Cox is correct

Usage

gofG.phreg(x, sim = 0, silent = 1, lm = TRUE, ...)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>phreg object</td>
</tr>
<tr>
<td>sim</td>
<td>to simulate some variation from cox model to put on graph</td>
</tr>
<tr>
<td>silent</td>
<td>to keep it absolutely silent</td>
</tr>
<tr>
<td>lm</td>
<td>add line to plot, regressing the cumulatives on each other</td>
</tr>
<tr>
<td>...</td>
<td>Additional arguments to lower level functions</td>
</tr>
</tbody>
</table>

Author(s)

Thomas Scheike and Klaus K. Holst
Examples

```r
data(TRACE)
m1 <- phreg(Surv(time, status==9) ~ strata(vf) + chf + wmi, data=TRACE)
m2 <- phreg(Surv(time, status==9) ~ vf + strata(chf) + wmi, data=TRACE)
par(mfrow=c(2,2))
gofG.phreg(m1)
gofG.phreg(m2)

bplot(m1, log="y")
bplot(m2, log="y")
```

Description

Cumulative residuals after model matrix for Cox PH regression p-values based on Lin, Wei, Ying resampling.

Usage

```r
gofM.phreg(formula, data, offset = NULL, weights = NULL,
modelmatrix = NULL, n.sim = 1000, silent = 1, ...)
```

Arguments

- `formula`: formula for cox regression
- `data`: data for model
- `offset`: offset
- `weights`: weights
- `modelmatrix`: matrix for cumulating residuals
- `n.sim`: number of simulations for score processes
- `silent`: to keep it absolutely silent, otherwise timing estimate will be produced for longer jobs.
- `...`: Additional arguments to lower level functions

Author(s)

Thomas Scheike and Klaus K. Holst
Examples

data(TRACE)

dcut(TRACE) <- -.
mm <- model.matrix(-1+factor(wmict4),data=TRACE)
m1 <- gofM.phreg(Surv(time,status==9)-vf+chf+wmi,data=TRACE,modelmatrix=mm)
summary(m1)
par(mfrow=c(2,2))
plot(m1)

m1 <- gofM.phreg(Surv(time,status==9)-strata(vf)+chf+wmi,data=TRACE,modelmatrix=mm)
summary(m1)

## cumulative sums in covariates, via design matrix mm
mm <- cumContr(TRACE$wmi,breaks=10,equi=TRUE)
m1 <- gofM.phreg(Surv(time,status==9)-strata(vf)+chf+wmi,data=TRACE,
   modelmatrix=mm,silent=0)
summary(m1)

## cumulative sums in covariates, via design matrix mm
m1 <- gofZ.phreg(Surv(time,status==9)-strata(vf)+chf+wmi,data=TRACE,vars="wmi")
summary(m1)


Grandom.cif

Additive Random effects model for competing risks data for polygenic
  modelling

Description

Fits a random effects model describing the dependence in the cumulative incidence curves for sub-
jects within a cluster. Given the gamma distributed random effects it is assumed that the cumulative
incidence curves are independent, and that the marginal cumulative incidence curves are on additive
form

\[ P(T \leq t, cause = 1|x,z) = P_1(t,x) = 1 - \exp(-x^T A(t) - tz^T \beta) \]

Usage

Grandom.cif(cif, data, cause = NULL, cif2 = NULL, times = NULL,
   cause1 = 1, cause2 = 1, cens.code = NULL, cens.model = "KM",
   Nit = 40, detail = 0, clusters = NULL, theta = NULL,
   theta.des = NULL, weights = NULL, step = 1, sym = 0,
   same.cens = FALSE, censoring.weights = NULL, silent = 1, var.link = 0,
   score.method = "fisher.scoring", entry = NULL, estimator = 1,
   trunkp = 1, admin.cens = NULL, random.design = NULL, ...)

**Arguments**

cif

a model object from the comp.risk function with the marginal cumulative incidence of cause2, i.e., the event that is conditioned on, and whose odds the comparison is made with respect to.

data

a data.frame with the variables.

cause

specifies the causes related to the death times, the value cens.code is the censoring value.

cif2

specificies model for cause2 if different from cause1.

times

time points.

cause1

cause of first coordinate.

cause2

cause of second coordinate.

cens.code

specificies the code for the censoring if NULL then uses the one from the marginal cif model.

cens.model

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

Nit

number of iterations for Newton-Raphson algorithm.

detail

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

clusters

specifies the cluster structure.

theta

specifies starting values for the cross-odds-ratio parameters of the model.

theta.des

specifies a regression design for the cross-odds-ratio parameters.

weights

weights for score equations.

step

specifies the step size for the Newton-Raphson algorithm.

sym

1 for symmetri and 0 otherwise.

same.cens

if true then censoring within clusters are assumed to be the same variable, default is independent censoring.

censoring.weights

censoring probabilities

silent

debug information.

var.link

if var.link=1 then var is on log-scale.

score.method

default uses "nlminb" optimzer, alternatively, use the "fisher-scoring" algorithm.

entry

entry-age in case of delayed entry. Then two causes must be given.

estimator

estimator.

trunkp

gives probability of survival for delayed entry, and related to entry-ages given above.

admin.cens

Administrative censoring

random.design

specificies a regression design of 0/1’s for the random effects.

... extra arguments.
Details

We allow a regression structure for the independent gamma distributed random effects and their variances that may depend on cluster covariates.

random.design specifies the random effects for each subject within a cluster. This is a matrix of 1’s and 0’s with dimension n x d. With d random effects. For a cluster with two subjects, we let the random.design rows be $v_1$ and $v_2$. Such that the random effects for subject 1 is

$$v_1^T(Z_1, ..., Z_d)$$

for d random effects. Each random effect has an associated parameter ($\lambda_1, ..., \lambda_d$). By construction subjects 1’s random effect are Gamma distributed with mean $\lambda_1/v_1^T\lambda$ and variance $\lambda_1/(v_1^T\lambda)^2$. Note that the random effect $v_1^T(Z_1, ..., Z_d)$ has mean 1 and variance $1/(v_1^T\lambda)$.

The parameters ($\lambda_1, ..., \lambda_d$) are related to the parameters of the model by a regression construction $pard$ (d x k), that links the d $\lambda$ parameters with the (k) underlying $\theta$ parameters

$$\lambda = pard\theta$$

Value

returns an object of type ‘random.cif’. With the following arguments:

- theta: estimate of parameters of model.
- var.theta: variance for gamma.
- hess: the derivative of the used score.
- score: scores at final stage.
- theta.iid: matrix of iid decomposition of parametric effects.

Author(s)

Thomas Scheike

References

A Semiparametric Random Effects Model for Multivariate Competing Risks Data, Scheike, Zhang, Sun, Jensen (2010), Biometrika.

Cross odds ratio Modelling of dependence for Multivariate Competing Risks Data, Scheike and Sun (2013), Biostatistics.

Scheike, Holst, Hjelmborg (2014), LIDA, Estimating heritability for cause specific hazards based on twin data

Examples

```r
## Reduce Ex.Timings
library("timereg")
library("survival")
d <- simnordic.random(5000, delayed=TRUE, 
cordz=1.0, cormz=2, lam0=0.3, country=TRUE)
times <- seq(50, 90, by=10)
```
### ipw

**Inverse Probability of Censoring Weights**

**Description**

Internal function. Calculates Inverse Probability of Censoring Weights (IPCW) and adds them to a data.frame

**Usage**

```
ipw(formula, data, cluster, same.cens = FALSE, obs.only = TRUE,
```
weight.name = "w", trunc.prob = FALSE, weight.name2 = "wt",
indi.weight = "pr", cens.model = "aalen", pairs = FALSE,
theta.formula = ~1,...)

Arguments

formula          Formula specifying the censoring model
data             data frame
cluster          clustering variable
same.cens        For clustered data, should same censoring be assumed (bivariate probability calculated as minimum of the marginal probabilities)
obs.only         Return data with uncensored observations only
weight.name      Name of weight variable in the new data.frame
trunc.prob       If TRUE truncation probabilities are also calculated and stored in ’weight.name2’ (based on Clayton-Oakes gamma frailty model)
weight.name2     Name of truncation probabilities
indi.weight      Name of individual censoring weight in the new data.frame
cens.model       Censoring model (default Aalen’s additive model)
pairs            For paired data (e.g. twins) only the complete pairs are returned (With pairs=TRUE)
theta.formula    Model for the dependence parameter in the Clayton-Oakes model (truncation only)
...              Additional arguments to censoring model

Author(s)

Klaus K. Holst

Examples

## Not run:
data("prt",package="mets")
prt <- ipw(Surv(time,status==0)-country, data=prt[sample(nrow(prt),5000),],
cluster="id",weight.name="w")
plot(0,type="n",xlim=range(prt$time),ylim=c(0,1),xlab="Age",ylab="Probability")
count <- 0
for (l in unique(prt$country)) {
count <- count+1
prt <- prt[order(prt$time),]
with(subset(prt, country==l),
lines(time,w,col=count,lwd=2))
}
legend("topright",legend=unique(prt$country),col=1:4,pch=-1,lty=1)

## End(Not run)
Inverse Probability of Censoring Weights

Description

Internal function. Calculates Inverse Probability of Censoring and Truncation Weights and adds them to a data.frame

Usage

ipwR(data, times = NULL, entrytime = NULL, time = "time",
cause = "cause", same.cens = FALSE, cluster = NULL, pairs = FALSE,
strata = NULL, obs.only = TRUE, cens.formula = NULL, cens.code = 0,
pair.cweight = "pcw", pair.tweight = "ptw", pair.weight = "weights",
cname = "cweights", tname = "tweights", weight.name = "indi.weights",
prec.factor = 100)

Arguments

data  data frame

times  possible time argument for specifying a maximum value of time tau=max(times),
to specify when things are considered censored or not.

entrytime  name of entry-time for truncation.

time  name of time variable on data frame.

cause  name of cause indicator on data frame.

same.cens  For clustered data, should same censoring be assumed and same truncation (bivariate probability calculated as minimum of the marginal probabilities)

cluster  name of clustering variable

pairs  For paired data (e.g. twins) only the complete pairs are returned (With pairs=TRUE)

strata  name of strata variable to get weights stratified.

obs.only  Return data with uncensored observations only

cens.formula  model for Cox models for truncation and right censoring times.

cens.code  censoring.code

pair.cweight  Name of weight variable in the new data.frame for right censoring of pairs

pair.tweight  Name of weight variable in the new data.frame for left truncation of pairs

pair.weight  Name of weight variable in the new data.frame for right censoring and left truncation of pairs

cname  Name of weight variable in the new data.frame for right censoring of individuals

tname  Name of weight variable in the new data.frame for left truncation of individuals

weight.name  Name of weight variable in the new data.frame for right censoring and left truncation of individuals

prec.factor  To let tied censoring and truncation times come after the death times.

...  Additional arguments to censoring model
Author(s)

Thomas Scheike

Examples

```r
library("timereg")
d <- simnordic.random(3000, delayed=TRUE, ptrunc=0.7,
cordz=0.5, cormz=1, lamb0=0.3, country=FALSE)
d$strata <- as.numeric(d$country)+(d$zyg=="MZ")*4
times <- seq(60,100,by=10)
c1 <- comp.risk(Event(time, cause)-1+cluster(id), data=d, cause=1,
model="fg", times=times, max.clust=NULL, n.sim=0)
mm=model.matrix(~1+zyg, data=d)
out1<-random.cif(c1, data=d, cause1=1, cause2=1, same.cens=TRUE, theta.des=mm)
summary(out1)
pcl <- predict(c1, X=1, se=0)
plot(pcl)
dl <- d[!d$struncated,]
dl <- ipw2(dl, cluster="id", same.cens=TRUE, time="time", entrytime="entry", cause="cause",
strata="strata", prec.factor=100)
c1 <- comp.risk(Event(time, cause)-1+cluster(id),
data=dl, cause=1, model="fg",
weights=dl$indi.weights, cens.weights=rep(1,nrow(dl)),
times=times, max.clust=NULL, n.sim=0)
pcl <- predict(c1, X=1, se=0)
lines(pcl$time, pcl$p1, col=2)
mm=model.matrix(~1+factor(zyg), data=dl)
out2<-random.cif(c1, data=dl, cause1=1, cause2=1, theta.des=mm,
weights=dl$weights, censoring.weights=rep(1,nrow(dl))
summary(out2)
```

km

Kaplan-Meier with robust standard errors

Description

Kaplan-Meier with robust standard errors. Robust variance is default variance with the summary.

Usage

```r
km(formula, data = data, conf.type = "log", conf.int = 0.95,
   robust = TRUE)
```


**Arguments**

- `formula`: formula with 'Surv' outcome (see `coxph`)
- `data`: data frame
- `conf.type`: transformation
- `conf.int`: level of confidence intervals
- `robust`: for robust standard errors based on martingales
- `...`: Additional arguments to lower level functions

**Author(s)**

Thomas Scheike

**Examples**

```r
data(TRACE)
TRACE$cluster <- sample(1:100,1878,replace=TRUE)
out1 <- km(Surv(time,status==9)-strata(vf,chf),data=TRACE)
out2 <- km(Surv(time,status==9)-strata(vf,chf)+cluster(cluster),data=TRACE)

par(mfrow=c(1,2))
bplot(out1,se=TRUE)
bplot(out2,se=TRUE)
```

**Description**

Life-course plot for event life data with recurrent events

**Usage**

```r
lifecourse(formula, data, id = "id", group = NULL, type = "l", lty = 1, col = 1:10, alpha = 0.3, lwd = 1, recurrent.col = NULL, recurrent.lty = NULL, legend = NULL, pchlegend = NULL, by = NULL, status.legend = NULL, place.sl = "bottomright", xlab = "Time", ylab = "", add = FALSE, ...)
```

**Arguments**

- `formula`: Formula (Event(start,slut,status) ~ ...)
- `data`: data.frame
- `id`: Id variable
- `group`: group variable
- `type`: Type (line 'l', stair 's', ...)
lty Line type
col Colour
alpha transparency (0-1)
lwd Line width
recurrent.col col of recurrence type
recurrent.lty lty’s of of recurrence type
legend position of optional id legend
pchlegend point type legends
by make separate plot for each level in 'by' (formula, name of column, or vector)
status.legend Status legend
place.sl Placement of status legend
xlab Label of X-axis
ylab Label of Y-axis
add Add to existing device
... Additional arguments to lower level arguments

Author(s)
Thomas Scheike, Klaus K. Holst

Examples

data = data.frame(id=c(1,1,1,2,2),start=c(0,1,2,3,4),slut=c(1,2,4,4,7),
  type=c(1,2,3,2,3),status=c(0,1,2,1,2),group=c(1,1,1,2,2))
l1 = lifecourse(Event(start,slut,status)-id,data,id="id")
l1 = lifecourse(Event(start,slut,status)-id,data,id="id",recurrent.col="type")

ll = lifecourse(Event(start,slut,status)-id,data,id="id",group=-group,col=1:2)
op <- par(mfrow=c(1,2))
ll = lifecourse(Event(start,slut,status)-id,data,id="id",by=-group)
par(op)
legends=c("censored","pregnant","married")
l1 = lifecourse(Event(start,slut,status)-id,data,id="id",group=-group,col=1:2,status.legend=legends)

lifetable.matrix Life table

Description
Create simple life table
LinSpline

Usage

## S3 method for class 'matrix'
linSpline(x, strata = list(), breaks = c(),
    weights=NULL, confint = FALSE, ...)

## S3 method for class 'formula'
linSpline(x, data=parent.frame(), breaks = c(),
    weights=NULL, confint = FALSE, ...)

Arguments

- **x**: time formula (Surv) or matrix/data.frame with columns time,status or entry,exit,status
- **strata**: strata
- **breaks**: time intervals
- **weights**: weights variable
- **confint**: if TRUE 95% confidence limits are calculated
- **...**: additional arguments to lower level functions
- **data**: data.frame

Author(s)

Klaus K. Holst

Examples

library(timereg)
data(TRACE)

d <- with(TRACE, lifetable(Surv(time,status==0)-sex+vf,breaks=c(0,0.2,0.5,8.5)))
summary(glm(events ~ offset(log(atrisk))+factor(int.end)*vf + sex*vf,
           data=d,poisson))

LinSpline

Simple linear spline

Description

Simple linear spline

Usage

LinSpline(x, knots, num = TRUE, name = "Spline")
mets.options

Arguments

- `x` variable to make into spline
- `knots` cut points
- `num` to give names x1 x2 and so forth
- `name` name of spline expansion name.1 name.2 and so forth

Author(s)

Thomas Scheike

mena

Menarche data set

Description

Menarche data set

Source

Simulated data

mets.options

Set global options for mets

Description

Extract and set global parameters of mets.

Usage

mets.options(...)

Arguments

... Arguments

Details

- `regex`: If TRUE character vectors will be interpreted as regular expressions (dby, dcut, ...)
- `silent`: Set to FALSE to disable various output messages

Value

list of parameters
Examples

```r
## Not run:
mets.options(regex=TRUE)

## End(Not run)
```

---

migr  

**Migraine data**

---

**Description**

Migraine data

---

multcif  

**Multivariate Cumulative Incidence Function example data set**

---

**Description**

Multivariate Cumulative Incidence Function example data set

**Source**

Simulated data

---

np  

**np data set**

---

**Description**

np data set

**Source**

Simulated data

---

npc  

**For internal use**

---

**Description**

For internal use

**Author(s)**

Klaus K. Holst
phreg  

*Fast Cox PH regression*

**Description**

Fast Cox PH regression. Robust variance is default variance with the summary.

**Usage**

```r
phreg(formula, data, offset = NULL, weights = NULL, ...)
```

**Arguments**

- `formula`: formula with 'Surv' outcome (see coxph)
- `data`: data frame
- `offset`: offsets for cox model
- `weights`: weights for Cox score equations
- `...`: Additional arguments to lower level functions

**Author(s)**

Klaus K. Holst, Thomas Scheike

**Examples**

```r
data(TRACE)
dcut(TRACE) <- -.
out1 <- phreg(Surv(time, status==9)~vf+chf+strata(wmicat.4), data=TRACE)
## tracesim <- timereg::sim.cox(out1,1000)
## sout1 <- phreg(Surv(time, status==1)~vf+chf+strata(wmicat.4), data=tracesim)
## robust standard errors default
summary(out1)

par(mfrow=c(1,2))
bplot(out1)
## bplot(sout1,se=TRUE)

## computing robust variance for baseline
rob1 <- robust.phreg(out1)
bplot(rob1,se=TRUE,robust=TRUE)

## making iid decomposition of regression parameters
betaiid <- iid(out1)
```
plack.cif

plack Computes concordance for or.cif based model, that is Plackett random effects model

Description

.. content for description (no empty lines) ..

Usage

plack.cif(cif1, cif2, object)

Arguments

cif1 Cumulative incidence of first argument.
cif2 Cumulative incidence of second argument.
object or.cif object with dependence parameters.

Author(s)

Thomas Scheike

pmvn

pmvn Multivariate normal distribution function

Description

Multivariate normal distribution function

Usage

pmvn(lower, upper, mu, sigma, cor = FALSE)

Arguments

lower lower limits
upper upper limits
mu mean vector
sigma variance matrix or vector of correlation coefficients
cor if TRUE sigma is treated as standardized (correlation matrix)
Examples

```r
lower <- rbind(c(0,-Inf),c(-Inf,0))
upper <- rbind(c(Inf,0),c(0,Inf))
mu <- rbind(c(1,1),c(-1,1))
sigma <- diag(c(2,2)+1)
pmvn(lower=lower,upper=upper,mu=mu,sigma=sigma)
```

```r
predict.phreg  # Predictions from proportional hazards model
```

Description

Predictions from proportional hazards model

Usage

```r
## S3 method for class 'phreg'
predict(object, data, surv = FALSE, time = object$exit,
  X = object$X, strata = object$strata, ...)
```

Arguments

- `object`: phreg object
- `data`: data.frame
- `surv`: If TRUE predictions are provided on probability scale
- `time`: Time variable
- `X`: Design matrix
- `strata`: Strata variable
- `...`: Additional arguments to lower level functions

```r
print.casewise  # prints Concordance test
```

Description

prints Concordance test

Usage

```r
## S3 method for class 'casewise'
print(x, digits = 3, ...)
```
Arguments

- `x` output from casewise.test
- `digits` number of digits
- `...` Additional arguments to lower level functions

Author(s)

Thomas Scheike

---

**prob.exceed.recurrent** *Estimation of probability of more that k events for recurrent events process*

**Description**

Estimation of probability of more that k events for recurrent events process where there is terminal event, based on this also estimate of variance of recurrent events. The estimator is based on cumulative incidence of exceeding "k" events. In contrast the probability of exceeding k events can also be computed as a counting process integral, and this is implemented in `prob.exceedRecurrent`

**Usage**

```r
prob.exceed.recurrent(data, type, status = "status", death = "death",
start = "start", stop = "stop", id = "id", times = NULL,
exceed = NULL)
```

**Arguments**

- `data` data-frame
- `type` type of event (code) related to status
- `status` name of status
- `death` name of death indicator
- `start` start stop call of Hist() of prodlim
- `stop` start stop call of Hist() of prodlim
- `id` id
- `times` time at which to get probabilities P(N1(t) >= n)
- `exceed` n’s for which which to compute probabilities P(N1(t) >= n)
- `...` Additional arguments to lower level functions

Author(s)

Thomas Scheike
References

Scheike, Eriksson, Tribler (2018) The mean, variance and correlation for bivariate recurrent events with a terminal event, work in progress

Examples

```r
# getting some rates to mimick

data(base1cumhaz)
data(base4cumhaz)
data(drcumhaz)
dr <- drcumhaz
base1 <- base1cumhaz
base4 <- base4cumhaz

cor.mat <- corM <- rbnd(c(1.0, 0.6, 0.9), c(0.6, 1.0, 0.5), c(0.9, 0.5, 1.0))
rr <- simRecurrent(1000,base1,cumhaz2=base4,death.cumhaz=dr)
rr <- count.history(rr)
dtable(rr,~death+status)
oo <- prob.exceedRecurrent(rr,1)
bplot(o0)

par(mfrow=c(1,2))
with(o0,plot(time,mu,col=2,type="l"))
with(o0,plot(time,varN,type="l"))

## Bivariate probability of exceeding
oo <- prob.exceedBiRecurrent(rr,1,2,exceed1=c(1,5,10),exceed2=c(1,2,3))
with(oo, matplot(time,pe1e2,type="s"))
nc <- ncol(oo$pe1e2)
legend("topleft",legend=colnames(oo$pe1e2),lty=1:nc,col=1:nc)

## do not test to avoid dependence on prodlim
## now estimation based on cumulative incidence, but do not test to avoid dependence on prodlim
library(prodlim)
pp <- prob.exceed.recurrent(rr,1,status="status",death="death",start="entry",stop="time",id="id")
with(pp, matplot(times.prob,type="s"))
with(pp, matlines(times.se.lower,type="s"))
with(pp, matlines(times.se.upper,type="s"))
```
**Description**

Prostate data set

**Source**

Simulated data

**random.cif**

*Random effects model for competing risks data*

**Description**

Fits a random effects model describing the dependence in the cumulative incidence curves for subjects within a cluster. Given the gamma distributed random effects it is assumed that the cumulative incidence curves are independent, and that the marginal cumulative incidence curves are on the form

\[ P(T \leq t, \text{cause} = 1| x, z) = P_1(t, x, z) = 1 - \exp(-x^T A(t) \exp(x^T \beta)) \]

We allow a regression structure for the random effects variances that may depend on cluster covariates.

**Usage**

```r
random.cif(cif, data, cause = NULL, cif2 = NULL, cause1 = 1, cause2 = 1,
            cens.code = NULL, cens.model = "KM", Nit = 40, detail = 0,
            clusters = NULL, theta = NULL, theta.des = NULL, sym = 1, step = 1,
            same.cens = FALSE, var.link = 0, score.method = "fisher.scoring",
            entry = NULL, trunkp = 1, ...)```

**Arguments**

- **cif**
  - a model object from the `comp.risk` function with the marginal cumulative incidence of cause2, i.e., the event that is conditioned on, and whose odds the comparision is made with respect to
- **data**
  - a `data.frame` with the variables.
- **cause**
  - specifies the causes related to the death times, the value `cens.code` is the censoring value.
- **cif2**
  - specifies model for cause2 if different from cause1.
- **cause1**
  - cause of first coordinate.
- **cause2**
  - cause of second coordinate.
cens.code specifies the code for the censoring if NULL then uses the one from the marginal cif model.
cens.model specified which model to use for the ICPW, KM is Kaplan-Meier alternatively it may be "cox"
Nit number of iterations for Newton-Raphson algorithm.
detail if 0 no details are printed during iterations, if 1 details are given.
custers specifies the cluster structure.
theta specifies starting values for the cross-odds-ratio parameters of the model.
theta.des specifies a regression design for the cross-odds-ratio parameters.
sym 1 for symmetry 0 otherwise
step specifies the step size for the Newton-Raphson algorithm.
same.cens if true then censoring within clusters are assumed to be the same variable, default is independent censoring.
var.link if var.link=1 then var is on log-scale.
score.method default uses "nlminb" optimzer; alternatively, use the "fisher-scoring" algorithm.
entry entry-age in case of delayed entry. Then two causes must be given.
trunkp gives probability of survival for delayed entry, and related to entry-ages given above.
... extra arguments.

Value

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

theta estimate of proportional odds parameters of model.
var.theta variance for gamma.
hess the derivative of the used score.
score scores at final stage.
score scores at final stage.
theta.iid matrix of iid decomposition of parametric effects.

Author(s)

Thomas Scheike

References

A Semiparametric Random Effects Model for Multivariate Competing Risks Data, Scheike, Zhang, Sun, Jensen (2010), Biometrika.

Cross odds ratio Modelling of dependence for Multivariate Competing Risks Data, Scheike and Sun (2012), work in progress.
Examples

```
## Reduce Ex.Timings
library("timereg")
d <- simnordic.random(4000,delayed=TRUE,
cordz=0.5,cormz=2, lam0=0.3,country=TRUE)
times <- seq(50,90,by=10)
add1<-comp.risk(Event(time,cause)=const(country)+cluster(id),data=d,
times=times,cause=1,max.clust=NULL)

### making group indicator
mm <- model.matrix(~-1+factor(zyg),d)

out1<-random.cif(add1,data=d,cause1=1,cause2=1,theta=1,same.cens=TRUE)
summary(out1)

out2<-random.cif(add1,data=d,cause1=1,cause2=1,theta=1,theta.des=mm,same.cens=TRUE)
summary(out2)

###############################################################################
#### 2 different causes
###############################################################################

add2<-comp.risk(Event(time,cause)=const(country)+cluster(id),data=d,
times=times,cause=2,max.clust=NULL)
out3<-random.cif(add1,data=d,cause1=1,cause2=2,cif2=add2,sym=1,same.cens=TRUE)
summary(out3) ## negative dependence

out4<-random.cif(add1,data=d,cause1=1,cause2=2,cif2=add2,theta.des=mm,sym=1,same.cens=TRUE)
summary(out4) ## negative dependence
```

**Description**

Fast Marginal means of recurrent events. Using the Lin and Ghosh (2000) standard errors. Fitting two models for death and recurrent events these are combined to produce the estimator

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

the mean number of recurrent events, here

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

is the probability of survival for the baseline group, and

$$
dR(u|x = 0)
$$
is the hazard rate of an event among survivors for the baseline. Here

\[ S(u|x = 0) \]

is estimated by

\[ \exp(-\Lambda_d(u|x = 0)) \]

with

\[ \Lambda_d(u|x = 0) \]

being the cumulative baseline for death.

**Usage**

\[ \text{recurrentMarginal}(\text{recurrent, death, fixbeta = NULL, km = TRUE, ...}) \]

**Arguments**

- recurrent: phreg object with recurrent events
- death: phreg object with deaths
- fixbeta: to force the estimation of standard errors to think of regression coefficients as known/fixed
- km: if true then uses Kaplan-Meier for death, otherwise \( \exp(-\text{Nelson-Aalen}) \)
- ...: Additional arguments to lower level functions

**Details**

Assumes no ties in the sense that jump times needs to be unique, this is particularly so for the stratified version.

**Author(s)**

Thomas Scheike

**References**


**Examples**

```r
data(base1cumhaz)
data(base4cumhaz)
data(drcumhaz)
dr <- drcumhaz
base1 <- base1cumhaz
base4 <- base4cumhaz
rr <- simRecurrent(1000,base1,death.cumhaz=dr)
rr$x <- rnorm(nrow(rr))
rr$strata <- floor((rr$id-0.01)/500)
```
## to fit non-parametric models with just a baseline

```r
xr <- phreg(Surv(entry, time, status) ~ cluster(id), data=rr)
dr <- phreg(Surv(entry, time, death) ~ cluster(id), data=rr)
par(mfrow=c(1,3))
bplot(dr, se=TRUE)
title(main="death")
bplot(xr, se=TRUE)
```

### robust standard errors

```r
rnr <- robust.phreg(xr, fixbeta=1)
bplot(rnr, se=TRUE, robust=TRUE, add=TRUE, col=4)
```

## marginal mean of expected number of recurrent events

```r
out <- recurrentMarginal(xr, dr)
bplot(out, se=TRUE, ylab="marginal mean", col=2)
```

### with strata

```r
xr <- phreg(Surv(entry, time, status) ~ strata(strata) + cluster(id), data=rr)
dr <- phreg(Surv(entry, time, death) ~ strata(strata) + cluster(id), data=rr)
par(mfrow=c(1,3))
bplot(dr, se=TRUE)
title(main="death")
bplot(xr, se=TRUE)
rnr <- robust.phreg(xr, fixbeta=1)
bplot(rnr, se=TRUE, robust=TRUE, add=TRUE, col=1:2)
```

```r
out <- recurrentMarginal(xr, dr)
bplot(out, se=TRUE, ylab="marginal mean", col=1:2)
```

### cox case

```r
xr <- phreg(Surv(entry, time, status) ~ x + cluster(id), data=rr)
dr <- phreg(Surv(entry, time, death) ~ x + cluster(id), data=rr)
par(mfrow=c(1,3))
bplot(dr, se=TRUE)
title(main="death")
bplot(xr, se=TRUE)
rnr <- robust.phreg(xr)
bplot(rnr, se=TRUE, robust=TRUE, add=TRUE, col=1:2)
```

```r
out <- recurrentMarginal(xr, dr)
bplot(out, se=TRUE, ylab="marginal mean", col=1:2)
```

### CIF

```r
bmt$id <- 1:nrow(bmt)
xr <- phreg(Surv(time, cause==1) ~ cluster(id), data=bmt)
```

```r
dr <- phreg(Surv(time, cause!=0) ~ cluster(id), data=bmt)
```
out <- recurrentMarginal(xr,dr,km=TRUE)
bplot(out,se=TRUE,ylab="cumulative incidence")

---

simAalenFrailty Simulate from the Aalen Frailty model

Description

Simulate observations from Aalen Frailty model with Gamma distributed frailty and constant intensity.

Usage

simAalenFrailty(n = 5000, theta = 0.3, K = 2, beta0 = 1.5, beta = 1,
               cens = 1.5, cuts = 0, ...)

Arguments

- **n**: Number of observations in each cluster
- **theta**: Dependence parameter (variance of frailty)
- **K**: Number of clusters
- **beta0**: Baseline (intercept)
- **beta**: Effect (log hazard ratio) of covariate
- **cens**: Censoring rate
- **cuts**: time cuts
- **...**: Additional arguments

Author(s)

Klaus K. Holst
Description
Simulate observations from the Clayton-Oakes copula model with piecewise constant marginals.

Usage
```
simclaytonoakes(K, n, eta, beta, stoptime, left = 0, pairleft = 0,
trunc.prob = 0.5, same = 0)
```

Arguments
- **K**: Number of clusters
- **n**: Number of observations in each cluster
- **eta**: 1/variance
- **beta**: Effect (log hazard ratio) of covariate
- **stoptime**: Stopping time
- **left**: Left truncation
- **pairleft**: pairwise (1) left truncation or individual (0)
- **trunc.prob**: Truncation probability
- **same**: if 1 then left-truncation is same also for univariate truncation

Author(s)
Thomas Scheike and Klaus K. Holst

Description
Simulate observations from the Clayton-Oakes copula model with Weibull type baseline and Cox marginals.

Usage
```
simclaytonoakeswei(K, n, eta, beta, stoptime, weiscale = 1, weishape = 2,
left = 0, pairleft = 0)
```
Arguments

- **K** Number of clusters
- **n** Number of observations in each cluster
- **eta** 1/variance
- **beta** Effect (log hazard ratio) of covariate
- **stoptime** Stopping time
- **weiscale** Weibull scale parameter
- **weishape** Weibull shape parameter
- **left** Left truncation
- **pairleft** Pairwise (1) left truncation or individual (0)

Author(s)

Klaus K. Holst

---

**simRecurrent**

*Simulation of recurrent events data based on cumulative hazards*

Description

Simulation of recurrent events data based on cumulative hazards

Usage

```r
simRecurrent(n, cumhaz, death.cumhaz = NULL, cumhaz2 = NULL, gap.time = FALSE, max.recurrent = 100, dhaz = NULL, haz2 = NULL, dependence = 0, var.z = 2, cor.mat = NULL, ...)
```

Arguments

- **n** number of id’s
- **cumhaz** cumulative hazard of recurrent events
- **death.cumhaz** cumulative hazard of death
- **cumhaz2** cumulative hazard of recurrent events of type 2
- **gap.time** if true simulates gap-times with specified cumulative hazard
- **max.recurrent** limits number recurrent events to 100
- **dhaz** rate for death hazard if it is extended to time-range of first event
- **haz2** rate of second cause if it is extended to time-range of first event
- **dependence** =0 independence, =1 all share same random effect with variance var.z =2 random effect exp(normal) with correlation structure from cor.mat, first random effect is z1 and shared for a possible second cause, second random effect is for death
- **var.z** variance of random effects
- **cor.mat** correlation matrix for var.z variance of random effects
- **...** Additional arguments to lower level functions
**Details**

Must give hazard of death and recurrent events. Possible with two event types and their dependence can be specified but the two recurrent events need to share random effect. combined to produce the estimator

**Author(s)**

Thomas Scheike

**Examples**

```r
# getting some rates to mimic

data(base1cumhaz)
data(base4cumhaz)
data(drcumhaz)

dr <- drcumhaz
base1 <- base1cumhaz
base4 <- base4cumhaz

cor.mat <- corM <- rbind(c(1.0, 0.6, 0.9), c(0.6, 1.0, 0.5), c(0.9, 0.5, 1.0))

# simulating simple model that mimicks data
rr <- simRecurrent(5,base1,death.cumhaz=dr)
dlist(rr,~id,n=0)

rr <- simRecurrent(1000,base1,death.cumhaz=dr)
par(mfrow=c(1,3))
showfitsim(causes=1,rr,dr,base1,base1)

# simulating simple model that mimicks data
# now with two event types and second type has same rate as death rate

rr <- simRecurrent(1000,base1,death.cumhaz=dr,cumhaz2=base4)
dtable(rr,~death,status)
par(mfrow=c(2,2))
showfitsim(causes=2,rr,dr,base1,base4)

# simulating simple model
# random effect for all causes (Z shared for death and recurrent)

rr <- simRecurrent(1000,base1,

death.cumhaz=dr,dependence=1,var.gamma=0.4)
```
### Simulation of recurrent events data based on cumulative hazards II

#### Description
Simulation of recurrent events data based on cumulative hazards

#### Usage
```r
simRecurrentII(n, cumhaz, cumhaz2, death.cumhaz = NULL, gap.time = FALSE, 
max.recurrent = 100, dhaz = NULL, haz2 = NULL, dependence = 0, 
var.z = 0.22, cor.mat = NULL, cens = NULL, ...) 
```

#### Arguments
- `n` number of id’s
- `cumhaz` cumulative hazard of recurrent events
- `cumhaz2` cumulative hazard of recurrent events of type 2
- `death.cumhaz` cumulative hazard of death
- `gap.time` if true simulates gap-times with specified cumulative hazard
- `max.recurrent` limits number recurrent events to 100
- `dhaz` rate for death hazard if it is extended to time-range of first event
- `haz2` rate of second cause if it is extended to time-range of first event
- `dependence` 0:independence; 1:all share same random effect with variance var.z; 2:random effect exp(normal) with correlation structure from cor.mat; 3:additive gamma distributed random effects, \( z_1 = \frac{z_{11} + z_{12}}{2} \) such that mean is 1, \( z_2 = \frac{z_{11} \text{ cor.mat(1,2)} + z_{13}}{2} \), \( z_3 = \frac{z_{12} \text{ cor.mat(2,3)} + z_{13} \text{ cor.mat(1,3)}}{2} \), with \( z_{11}, z_{12}, z_{13} \) are gamma with mean and variance 1, first random effect is \( z_1 \) and for \( N_1 \) second random effect is \( z_2 \) and for \( N_2 \) third random effect is for death
- `var.z` variance of random effects
- `cor.mat` correlation matrix for var.z variance of random effects
- `cens` rate of censoring exponential distribution
- `...` Additional arguments to lower level functions

#### Details
Must give hazard of death and two recurrent events. Possible with two event types and their dependence can be specified but the two recurrent events need to share random effect. Based on drawing the from cumhaz and cumhaz2 and taking the first event rather the cumulative and then distributing it out. Key advantage of this is that there is more flexibility wrt random effects
summary.cor

Author(s)
Thomas Scheike

Examples

# getting some rates to mimick

data(base1cumhaz)
data(base4cumhaz)
data(drcumhaz)
dr <- drcumhaz
base1 <- base1cumhaz
base4 <- base4cumhaz
cor.mat <- corM <- rbind(c(1.0, 0.6, 0.9), c(0.6, 1.0, 0.5), c(0.9, 0.5, 1.0))

# simulating simple model that mimicks data
# now with two event types and second type has same rate as death rate

rr <- simRecurrentII(1000, base1, dr, death.cumhaz=base4)
dtable(rr, "death\+status")
par(mfrow=c(2,2))
showfitsim(causes=2, rr, dr, base1, base4)

summary.cor  Summary for dependence models for competing risks

Description
Computes concordance and casewise concordance for dependence models for competing risks models of the type cor.cif, rr.cif or or.cif for the given cumulative incidences and the different dependence measures in the object.

Usage

## S3 method for class 'cor'
summary(object, marg.cif = NULL, marg.cif2 = NULL,
digits = 3, ...)

Arguments

object object from cor.cif rr.cif or or.cif for dependence between competing risks data for two causes.
marg.cif a number that gives the cumulative incidence in one time point for which concordance and casewise concordance are computed.
marg.cif2 the cumulative incidence for cause 2 for concordance and casewise concordance are computed. Default is that it is the same as marg.cif.
digits digits in output.
... Additional arguments.

Value
prints summary for dependence model.
casewise gives casewise concordance that is, probability of cause 2 (related to cif2) given that cause 1 (related to cif1) has occurred.
concordance gives concordance that is, probability of cause 2 (related to cif2) and cause 1 (related to cif1).
cif1 cumulative incidence for cause1.
cif2 cumulative incidence for cause1.

Author(s)
Thomas Scheike

References
Cross odds ratio Modelling of dependence for Multivariate Competing Risks Data, Scheike and Sun (2012), Biostatistics to appear.
A Semiparametric Random Effects Model for Multivariate Competing Risks Data, Scheike, Zhang, Sun, Jensen (2010), Biometrika.

Examples
library("timereg")
data("multcif",package="mets") # simulated data
multcif$cause[multcif$cause==0] <- 2

times=seq(0.1,3,by=0.1) # to speed up computations use only these time-points
add<-comp.risk(Event(time,cause)-const(X)+cluster(id),data=multcif,
 n.sim=0,times=times,cause=1)
###
out1<-cor.cif(add,data=multcif,cause1=1,cause2=1,theta=log(2+1))
summary(out1)

pad <- predict(add,X=1,Z=0,se=0,uniform=0)
summary(out1,marg.cif=pad)
survival.iterative

Survival model for multivariate survival data

Description

Fits additive gamma frailty model with additive hazard conditional on the random effects

$$\lambda_{ij} = (V_{ij}^T Z)(X_{ij}^T \alpha(t))$$

The baseline $\alpha(t)$ is profiled out using marginal modelling adjusted for the random effects structure as in Eriksson and Scheike (2015). One advantage of the standard frailty model is that one can deal with competing risks for this model.

For all models the standard errors do not reflect this uncertainty of the baseline estimates, and might therefore be a bit too small. To remedy this one can do bootstrapping or use survival.twostage.fullse function when possible.

If clusters contain more than two times, the algorithm uses a composite likelihood based on the pairwise bivariate models. Can also fit an additive gamma random effects model described in detail below.

We allow a regression structure for the independent gamma distributed random effects and their variances that may depend on cluster covariates. So

$$\theta = z_j^T \alpha$$

where $z$ is specified by theta.des. The reported standard errors are based on the estimated information from the likelihood assuming that the marginals are known.

Can also fit a structured additive gamma random effects model, such as the ACE, ADE model for survival data.

Now random.design specifies the random effects for each subject within a cluster. This is a matrix of 1’s and 0’s with dimension $n \times d$. With $d$ random effects. For a cluster with two subjects, we let the random.design rows be $v_1$ and $v_2$. Such that the random effects for subject 1 is

$$v_1^T (Z_1, ..., Z_d)$$

for $d$ random effects. Each random effect has an associated parameter $(\lambda_1, ..., \lambda_d)$. By construction subjects 1’s random effect are Gamma distributed with mean $\lambda_j / v_1^T \lambda$ and variance $\lambda_j / (v_1^T \lambda)^2$.

Note that the random effect $v_1^T (Z_1, ..., Z_d)$ has mean 1 and variance $1 / (v_1^T \lambda)$. It is here assumed that $lamtot = v_1^T \lambda$ is fixed over all clusters as it would be for the ACE model below. The lamtot parameter may be specified separately for some sets of the parameter is the additive.gamma.sum (ags) matrix is specified and then lamtot for the j’th random effect is $ags_j^T \lambda$.

Based on these parameters the relative contribution (the heritability, $h$) is equivalent to the expected values of the random effects $\lambda_j / v_1^T \lambda$

The DEFAULT parametrization uses the variances of the random effects

$$\theta_j = \lambda_j / (v_1^T \lambda)^2$$

For alternative parametrizations one can specify how the parameters relate to $\lambda_j$ with the function
Given the random effects the survival distributions with a cluster are independent and on the form

\[ P(T > t | x, z) = \exp(-Z \cdot \text{Laplace}^{-1}(\text{lamtot}, \text{lamtot}, S(t|x))) \]

with the inverse laplace of the gamma distribution with mean 1 and variance lamtot.

The parameters \((\lambda_1, ..., \lambda_d)\) are related to the parameters of the model by a regression construction \(pard\) (d x k), that links the \(d\ \lambda\) parameters with the (k) underlying \(\theta\) parameters

\[ \lambda = \text{theta.des}\theta \]

here using theta.des to specify these low-dimension association. Default is a diagonal matrix.

The case.control option that can be used with the pair specification of the pairwise parts of the estimating equations. Here it is assumed that the second subject of each pair is the proband.

Usage

```
survival.iterative(margsurv, data = sys.parent(),
    scoreNmethod = "fisher.scoring", Nit = 60, detail = 0,
    clusters = NULL, silent = 1, weights = NULL, control = list(),
    theta = NULL, theta.des = NULL, var.link = 1, iid = 1, step = 0.5,
    notaylor = 0, model = "clayton.oakes", marginal.trunc = NULL,
    marginal.survival = NULL, marginal.status = NULL, strata = NULL,
    se.clusters = NULL, max.clust = NULL, numDeriv = 0,
    random.design = NULL, pairs = NULL, pairs.rvs = NULL,
    numDeriv.method = "simple", additive.gamma.sum = NULL, var.par = 1,
    cr.models = NULL, case.control = 0, ascertained = 0, shut.up = 0)
```

Arguments

- **margsurv**: Marginal model
- **data**: data frame
- **scoreNmethod**: Scoring method "fisher.scoring", "nlminb", "optimize", "nlm"
- **Nit**: Number of iterations
- **detail**: Detail
- **clusters**: Cluster variable
- **silent**: Debug information
- **weights**: Weights
- **control**: Optimization arguments
- **theta**: Starting values for variance components
- **theta.des**: design for dependence parameters, when pairs are given this is could be a (pairs)
  x (numet of parameters) x (max number random effects) matrix
- **var.link**: Link function for variance
- **iid**: Calculate i.i.d. decomposition
- **step**: Step size
- **notaylor**: Taylor expansion
model
marginal.trunc
marginal.survival
marginal.status
strata
se.clusters
max.clust
numDeriv
random.design
pairs
pairs.rvs
numDeriv.method
additive.gamma.sum
var.par
cr.models
case.control
ascertained
shut.up

Author(s)

Thomas Scheike
**Description**

Fits Clayton-Oakes or bivariate Plackett models for bivariate survival data using marginals that are on Cox or additive form. The dependence can be modelled via

1. Regression design on dependence parameter.
2. Random effects, additive gamma model.

For all models the standard errors do not reflect this uncertainty of the baseline estimates, and might therefore be a bit to small. To remedy this one can do bootstrapping or use survival.twostage.fullse function when possible.

If clusters contain more than two times, the algorithm uses a composite likelihood based on the pairwise bivariate models. Can also fit a additive gamma random effects model described in detail below.

We allow a regression structure for the independentgamma distributed random effects and their variances that may depend on cluster covariates. So

\[ \theta = z_j^T \alpha \]

where \( z \) is specified by theta.des

The reported standard errors are based on the estimated information from the likelihood assuming that the marginals are known.

Can also fit a structured additive gamma random effects model, such as the ACE, ADE model for survival data.

Given the gamma distributed random effects it is assumed that the survival functions are independent, and that the marginal survival functions are on additive form (or Cox form)

\[ P(T > t|x) = S(t|x) = \exp(-x^TA(t)) \]

Now random.design specifies the random effects for each subject within a cluster. This is a matrix of 1’s and 0’s with dimension \( n \times d \). With \( d \) random effects. For a cluster with two subjects, we let the random.design rows be \( v_1 \) and \( v_2 \). Such that the random effects for subject 1 is

\[ v_1^T(Z_1, ..., Z_d) \]

, for \( d \) random effects. Each random effect has an associated parameter \( (\lambda_1, ..., \lambda_d) \). By construction subjects 1’s random effect are Gamma distributed with mean \( \lambda_j/v_1^T \lambda \) and variance \( \lambda_j/(v_1^T \lambda)^2 \). Note that the random effect \( v_1^T(Z_1, ..., Z_d) \) has mean 1 and variance \( 1/(v_1^T \lambda) \). It is here assumed that \( lamtot = v_1^T \lambda \) is fixed over all clusters as it would be for the ACE model below. The lamtot parameter may be specified separately for some sets of the parameter is the additive.gamma.sum (ags) matrix is specified and then lamtot for the j’th random effect is \( ags_j^T \lambda \).

Based on these parameters the relative contribution (the heritability, \( h \)) is equivalent to the expected values of the random effects \( \lambda_j/v_1^T \lambda \)
The DEFAULT parametrization uses the variances of the random effects
\[ \theta_j = \frac{\lambda_j}{(v^T \lambda)^2} \]

For alternative parametrizations one can specify how the parameters relate to \( \lambda_j \) with the function. Given the random effects the survival distributions with a cluster are independent and on the form

\[ P(T > t | x, z) = \exp(-Z \cdot \text{Laplace}^{-1}(\text{lamtot}, \text{lamtot}, S(t|x))) \]

with the inverse laplace of the gamma distribution with mean 1 and variance lamtot.

The parameters \((\lambda_1, ..., \lambda_d)\) are related to the parameters of the model by a regression construction \( \text{pard} \) (d x k), that links the d \( \lambda \) parameters with the (k) underlying \( \theta \) parameters

\[ \lambda = \theta \text{.des} \theta \]

here using \( \theta \text{.des} \) to specify these low-dimension association. Default is a diagonal matrix.

The case.control option that can be used with the pair specification of the pairwise parts of the estimating equations. Here it is assumed that the second subject of each pair is the proband.

Usage

```r
survival.twostage(margsurv, data = sys.parent(),
    scoreNmethod = “fisher.scoring”, Nit = 60, detail = 0,
    clusters = NULL, silent = 1, weights = NULL, control = list(),
    theta = NULL, theta.des = NULL, var.link = 1, iid = 1, step = 0.5,
    notaylor = 0, model = “clayton.oakes”, marginal.trunc = NULL,
    marginal.survival = NULL, marginal.status = NULL, strata = NULL,
    se.clusters = NULL, max.clust = NULL, numDeriv = 0,
    random.design = NULL, pairs = NULL, pairs.rvs = NULL,
    numDeriv.method = “simple”, additive.gamma.sum = NULL, var.par = 1,
    cr.models = NULL, case.control = 0, ascertained = 0, shut.up = 0)
```

Arguments

- `margsurv`: Marginal model
- `data`: data frame
- `scoreNmethod`: Scoring method “fisher.scoring”, “nlminb”, “optimize”, “nlm”
- `Nit`: Number of iterations
- `detail`: Detail
- `clusters`: Cluster variable
- `silent`: Debug information
- `weights`: Weights
- `control`: Optimization arguments
- `theta`: Starting values for variance components
- `theta.des`: design for dependence parameters, when pairs are given this is could be a (pairs) x (numer of parameters) x (max number random effects) matrix
var.link | Link function for variance
iid | Calculate i.i.d. decomposition
step | Step size
notaylor | Taylor expansion
model | model
marginal.trunc | marginal left truncation probabilities
marginal.survival | optional vector of marginal survival probabilities
marginal.status | related to marginal survival probabilities
strata | strata for fitting, see example
se.clusters | for clusters for se calculation with iid
max.clust | max se.clusters for se calculation with iid
numDeriv | to get numDeriv version of second derivative, otherwise uses sum of squared score
random.design | random effect design for additive gamma model, when pairs are given this is a (pairs) x (2) x (max number random effects) matrix, see pairs.rvs below
pairs | matrix with rows of indeces (two-columns) for the pairs considered in the pairwise composite score, useful for case-control sampling when marginal is known.
pairs.rvs | for additive gamma model and random.design and theta.des are given as arrays, this specifies number of random effects for each pair.
numDeriv.method | uses simple to speed up things and second derivative not so important.
additive.gamma.sum | for two.stage=0, this is specification of the lamtot in the models via a matrix that is multiplied onto the parameters theta (dimensions=(number random effects x number of theta parameters)), when null then sums all parameters.
var.par | is 1 for the default parametrization with the variances of the random effects, var.par=0 specifies that the $\lambda_j$’s are used as parameters.
cr.models | competing risks models for two.stage=0, should be given as a list with models for each cause
case.control | assumes case control structure for "pairs" with second column being the probands, when this option is used the two stage model is profiled out via the paired estimating equations for the survival model.
ascertained | if the pair are sampled only when there is an event. This is in contrast to case.control sampling where a proband is given. This can be combined with control probands. Pair-call of two stage is needed and second column of pairs are the first jump time with an event for ascertained pairs, or time of control proband.
shut.up | to make the program more silent in the context of iterative procedures for case-control and ascertained sampling
survival.twostage

Author(s)

Thomas Scheike

References

Estimating heritability for cause specific mortality based on twins studies Scheike, Holst, Hjelmborg (2014), LIDA
Measuring early or late dependence for bivariate twin data Scheike, Holst, Hjelmborg (2015), LIDA
Twostage modelling of additive gamma frailty models for survival data. Scheike and Holst, working paper

Examples

```r
library("timereg")
library("survival")
data(diabetes)

# Marginal Cox model with treat as covariate
margph <- coxph(Surv(time,status)~treat,data=diabetes)
### Clayton-Oakes, from timereg
fitco1 <- two.stage(margph,data=diabetes,theta=1.0,detail=0,Nit=40,clusters=diabetes$id)
summary(fitco1)
### Plackett model
fitp <- survival.twostage(margph,data=diabetes,theta=3.0,Nit=40,
                           clusters=diabetes$id,link=1,model="plackett")
summary(fitp)
### Clayton-Oakes
fitco2 <- survival.twostage(margph,data=diabetes,theta=0.0,detail=0,
                            clusters=diabetes$id,link=1,model="clayton.oakes")
summary(fitco2)
fitco3 <- survival.twostage(margph,data=diabetes,theta=1.0,detail=0,
                            clusters=diabetes$id,link=0,model="clayton.oakes")
summary(fitco3)

### without covariates using Aalen for marginals
marg <- aalen(Surv(time,status)+1,data=diabetes,n.sim=0,max.clust=NULL,robust=1)
fitpa <- survival.twostage(marg,data=diabetes,theta=1.0,detail=0,Nit=40,
                           clusters=diabetes$id,method="optimize")
summary(fitpa)

fitcoa <- survival.twostage(marg,data=diabetes,theta=1.0,detail=0,Nit=40,clusters=diabetes$id,
                            link=1,model="clayton.oakes")
summary(fitcoa)
```
### Piecewise constant cross hazards ratio modelling

```r
d <- subset(simClaytonOakes(2000, 2, 0.5, 0, stoptime=2, left=0), !truncated)
udp <- piecewise.twostage(c(0,0.5,2), data=d, score.method="optimize",
  id="cluster", timevar="time",
  status="status", model="clayton.oakes", silent=0)

summary(udp)
```

## Reduce Ex. Timings

### Same model using the strata option, a bit slower

```r
# makes the survival pieces for different areas in the plane
#ud1=surv.boxarea(c(0,0),c(0.5,0.5),data=d,id="cluster",timevar="time",status="status")
#ud2=surv.boxarea(c(0,0.5),c(0.5,2),data=d,id="cluster",timevar="time",status="status")
#ud3=surv.boxarea(c(0.5,0),c(2,0.5),data=d,id="cluster",timevar="time",status="status")
#ud4=surv.boxarea(c(0.5,0.5),c(2,2),data=d,id="cluster",timevar="time",status="status")

# everything done in one call
ud <- piecewise.data(c(0,0.5,2),data=d,timevar="time",status="status",id="cluster")
ud$strata <- factor(ud$strata);
ud$intstrata <- factor(ud$intstrata)

# makes strata specific id variable to identify pairs within strata
# se's computed based on the id variable across strata "cluster"
ud$istrata <- ud$id+(as.numeric(ud$strata)-1)*2000

marg2 <- aalen(Surv(boxtime, status)-1+factor(num):factor(intstrata),
data=ud,n.sim=0,robust=0)
tdes <- model.matrix(~-1+factor(strata), data=ud)
fitp2 <- survival.twostage(marg2, data=ud, se.clusters=ud$cluster, clusters=ud$istrata,
  score.method="fisher.scoring", model="clayton.oakes",
  theta.des=tdes, step=0.5)

summary(fitp2)
```

### now fitting the model with symmetry, i.e. strata 2 and 3 same effect

```r
#ud$stratas <- ud$strata;
#ud$stratas[ud$strata=="0.5-2,0-0.5"] <- "0-0.5,0.5-2"
tdes2 <- model.matrix(~-1+factor(stratas), data=ud)
fitp3 <- survival.twostage(marg2, data=ud, clusters=ud$iddstrata, se.cluster=ud$cluster,
  score.method="fisher.scoring", model="clayton.oakes",
  theta.des=tdes2, step=0.5)

summary(fitp3)
```

### same model using strata option, a bit slower

```r
fitp4 <- survival.twostage(marg2, data=ud, clusters=ud$cluster, se.cluster=ud$cluster,
  score.method="fisher.scoring", model="clayton.oakes",
  theta.des=tdes2, step=0.5, strata=ud$strata)

summary(fitp4)
```

### structured random effects model additive gamma ACE
### simulate structured two-stage additive gamma ACE model

data <- simClaytonOakes.twin.ace(2000,2,1,0,3)
out <- twin.polygen.design(data,id="cluster")
pardes <- out$pardes
des.rv <- out$des.rv

aa <- aalen(Surv(time,status)+1,data=data,robust=0)
ts <- survival.twostage(aa,data=data,clusters=data$cluster,detail=0,
theta=c(2,1)/10,var.link=0,step=0.5,
random.design=des.rv,theta.des=pardes)
summary(ts)

### case control sampling of data, call via pairs

data2 <- fast.reshape(data,id="cluster")
ncontrol <- 400; ncase <- 100
controls <- which(data2$status==0)
cases <- which(data2$status==1)
controls<-sort(sample(controls,min(ncontrol,length(controls))))
cases <- sort(sample(cases, min(ncase,length(cases))))
clustco <- data2$cluster[controls]
clustca <- data2$cluster[cases]
ss <- data$cluster %in% c(clustco,clustca)
datacc <- data[ss,]

mm <- familycluster.index(datacc$cluster)
pairs <- mm$pairs
head(pairs)

## second column of pairs represent probands
kinship <- rep(1,nrow(pairs))
kinship[datacc$zyg[pairs[,1]]=="DZ"] <- 0.5
dout <- make.pairwise.design(pairs,kinship,type="ace")

## additive model specified via formula-list

# crNmodels <- list(Surv(time,status)+1)

tscce <- survival.twostage(NULL,data=datacc,clusters=datacc$cluster,
detail=0,theta=c(2,1)/10,var.link=0,step=1.0,
pairs=pairs,
random.design=dout$random.design,theta.des=dout$theta.des,
pairs.rvs=dout$ant.rvs,
case.control=1, marginal.status=datacc$status,
.inject.models=crNmodels)
summary(tscce)

### see also pairwise*.r demos under inst for frailty, competing risks and
### case control sampling

---

**Concordance test**

*Compares two concordance estimates*
tetrachoric

Description

Calculate tetrachoric correlation of probabilities from odds-ratio

Usage

tetrachoric(P, OR, approx = TRUE, ...)
twin.clustertrunc  

Estimation of twostage model with cluster truncation in bivariate situation

Description

Estimation of twostage model with cluster truncation in bivariate situation

Usage

twin.clustertrunc(survformula, data = sys.parent(), theta.des = NULL, clusters = NULL, var.link = 1, Nit = 10, final.fitting = FALSE, ...)

Arguments

- survformula: Formula with survival model aalen or cox.aalen, some limitation on model specification due to call of fast.reshape (so for example interactions and * and : do not work here, expand prior to call)
- data: Data frame
- theta.des: design for dependence parameters in two-stage model
- clusters: clustering variable for twins
- var.link: exp link for theta
- Nit: number of iteration
- final.fitting: TRUE to do final estimation with SE and ... arguments for marginal models
- ...: Additional arguments to lower level functions

Author(s)

Thomas Scheike

Examples

```r
library("timereg")
data(diabetes)
v <- diabetes$time*runif(nrow(diabetes))*rbinom(nrow(diabetes),1,0.5)
diabetes$v <- v

aout <- twin.clustertrunc(Surv(v, time, status)~1+treat+adult, data=diabetes, clusters="id")
aout$two  ## two stage output
par(mfrow=c(2,2))
plot(aout$marg)  ## marginal model output

out <- twin.clustertrunc(Surv(v, time, status)~1+prop(treat)+prop(adult), data=diabetes, clusters="id")
out$two   ## two stage output
plot(out$marg)  ## marginal model output
```
twinbmi

**BMI data set**

**Description**
BMI data set

**Format**
Self-reported BMI-values on 11,411 subjects
tvparnr: twin id bmi: BMI (m/kg^2) age: Age gender: (male/female) zyg: zygosity, MZ:=mz, DZ(same sex):=dz, DZ(opposite sex):=os

twinlm

**Classic twin model for quantitative traits**

**Description**
Fits a classical twin model for quantitative traits.

**Usage**

```
twinlm(formula, data, id, zyg, DZ, group = NULL, group.equal = FALSE,
strata = NULL, weights = NULL, type = c("ace"), twinnum = "twinnum",
binary = FALSE, ordinal = 0, keep = weights, estimator = NULL,
constrain = TRUE, control = list(), messages = 1, ...)
```

**Arguments**

- **formula**: Formula specifying effects of covariates on the response
- **data**: data.frame with one observation per row. In addition a column with the zygosity (DZ or MZ given as a factor) of each individual must be specified as well as a twin id variable giving a unique pair of numbers/factors to each twin pair
- **id**: The name of the column in the dataset containing the twin-id variable.
- **zyg**: The name of the column in the dataset containing the zygosity variable
- **DZ**: Character defining the level in the zyg variable corresponding to the dyzogotic twins. If this argument is missing, the reference level (i.e. the first level) will be interpreted as the dyzogotic twins
- **group**: Optional. Variable name defining group for interaction analysis (e.g., gender)
- **group.equal**: If TRUE marginals of groups are assumed to be the same
- **strata**: Strata variable name
- **weights**: Weights matrix if needed by the chosen estimator. For use with Inverse Probability Weights
type Character defining the type of analysis to be performed. Should be a subset of "aced" (additive genetic factors, common environmental factors, unique environmental factors, dominant genetic factors).

twinnum The name of the column in the dataset numbering the twins (1,2). If it does not exist in data it will automatically be created.

binary If TRUE a liability model is fitted. Note that if the right-hand-side of the formula is a factor, character vector, or logical variable, then the liability model is automatically chosen (wrapper of the bptwin function).

ordinal If non-zero (number of bins) a liability model is fitted.

keep Vector of variables from data that are not specified in formula, to be added to data.frame of the SEM

estimator Choice of estimator/model

constraint Development argument

control Control argument parsed on to the optimization routine

messages Control amount of messages shown

... Additional arguments parsed on to lower-level functions

Value

Returns an object of class twinlm.

Author(s)

Klaus K. Holst

See Also

bptwin, twinlm.time, twinlm.strata, twinsim

Examples

## Simulate data
set.seed(1)
d <- twinsim(1000, bl=c(1,-1), b2=c(), acde=c(1,1,0,1))
## E(y|z1,z2) = z1 - z2. var(A) = var(C) = var(E) = 1

## E.g to fit the data to an ACE-model without any confounders we simply write
ace <- twinlm(y ~ 1, data=d, DZ="DZ", zyg="zyg", id="id")
ace
## An AE-model could be fitted as
ae <- twinlm(y ~ 1, data=d, DZ="DZ", zyg="zyg", id="id", type="ae")
## LRT:
lava::compare(ae,ace)
## AIC
AIC(ae)-AIC(ace)
## To adjust for the covariates we simply alter the formula statement
ace2 <- twinlm(y - x1+x2, data=d, DZ="DZ", zyg="zyg", id="id", type="ace")
## Summary/GOF
## twinsim

### Simulate twin data

**Description**

Simulate twin data from a linear normal ACE/ADE/AE model.

**Usage**

```r
twinsim(nMZ = 100, nDZ = nMZ, b1 = c(), b2 = c(), mu = 0, acde = c(1, 1, 0, 1), randomslope = NULL, threshold = 0, cens = FALSE, wide = FALSE, ...)
```

**Arguments**

- `nMZ`: Number of monozygotic twin pairs
- `nDZ`: Number of dizygotic twin pairs
- `b1`: Effect of covariates (labelled `x1,x2,...`) of type 1. One distinct covariate value for each twin/individual.
- `b2`: Effect of covariates (labelled `g1,g2,...`) of type 2. One covariate value for each twin pair.
- `mu`: Intercept parameter.
- `acde`: Variance of random effects (in the order A,C,D,E)
- `randomslope`: Logical indicating whether to include random slopes of the variance components w.r.t. `x1,x2,...`
- `threshold`: Treshold used to define binary outcome `y0`
- `cens`: Logical variable indicating whether to censor outcome
- `wide`: Logical indicating if wide data format should be returned
- `...`: Additional arguments parsed on to lower-level functions

**Author(s)**

Klaus K. Holst

**See Also**

`twinlm`
Description
Based on nation-wide questionnaire answers from 33,317 Danish twins

Format
tparnr: twin-pair id  zyg: zygosity, MZ:=mz, DZ(same sex):=dz, DZ(opposite sex):=os  stutter: stutter status (yes/no)  age nr: number within twin-pair
Index

*Topic **binomial**
  binomial.twostage, 10
  easy.binomial.twostage, 52
*Topic **data**
  base1cumhaz, 6
  base44cumhaz, 6
  base4cumhaz, 6
  dermalridges, 39
  dermalridgesMZ, 40
  drcumhaz, 44
  mena, 78
  migr, 79
  multcif, 79
  np, 79
  prt, 85
  twinbmi, 108
  twinstut, 111
*Topic **models**
  blocksample, 16
  mets.options, 78
  twinlm, 108
  twinsim, 110
*Topic **package**
  mets.package, 4
*Topic **regression**
  binomial.twostage, 10
  easy.binomial.twostage, 52
  twinlm, 108
  twinsim, 110
*Topic **survival**
  cor.cif, 26
  easy.survival.twostage, 55
  Grandom.cif, 68
  LinSpline, 77
  random.cif, 85
  summary.cor, 95
  survival.iterative, 97
  survival.twostage, 100
*Topic **twostage**
  easy.survival.twostage, 55
*Topic **utilities**
  blocksample, 16
 npc, 79
  aalenfrailty, 4
  ace.family.design (npc), 79
  alpha2kendall (npc), 79
  alpha2spear (npc), 79
  ascertained.pairs (npc), 79
  back2timer (npc), 5
  basecif, 6
  base44cif, 6
  base4cif, 6
  basecif (basehazplot.phreg), 7
  basehazplot.phreg, 7
  bicomprisk, 8
  binomial.twostage, 10
  biprobit, 14
  blocksample, 16
  Bootcovariancerecurrence
    (covarianceRecurrent), 31
  BootcovariancerecurrenceS
    (covarianceRecurrent), 31
  bplot (basehazplot.phreg), 7
  bptwin, 18, 109
  casewise, 19
  casewise.bin (casewise.test), 20
  casewise.test, 20
  CCBinomial.twostage (npc), 79
  cif, 22
  ClaytonOakes, 23
  cluster.index, 24
  coarse.clust (npc), 79
  coefmat (npc), 79
  concordance, 25
  concordance.twin.ace (npc), 79
  concordance.twostage (npc), 79
INDEX

113
cor.cif, 26
corsim.prostate (npc), 79
count.history, 30
covarianceRecurrent, 31
covarianceRecurrentS
   (covarianceRecurrent), 31
covfr (predict.phreg), 82
covfridstrata (predict.phreg), 82
covfridstrataCov (predict.phreg), 82
covInth1dm1inthRdmR (simRecurrent), 92
cumContr (gofM.phreg), 67
cumsumidstratasum (predict.phreg), 82
cumsumidstratasumCov (predict.phreg), 82
cumsumstrata (predict.phreg), 82
cumsumstratasum (predict.phreg), 82
daggr (daggregate), 32
daggregate, 32
dbvn, 34
dby, 34
dby2 (dby), 34
dby2 <- (dby), 34
dby <- (dby), 34
dbyr (dby), 34
dcor, 36
dcount (dcor), 36
dcut, 37
dcut <- (dcut), 37
ddrop (dcut), 37
ddrop <- (dcut), 37
dermalridges, 39
dermalridgesMZ, 40
deval (dcor), 36
deval2 (dcor), 36
dfactor (drelevel), 46
dfactor <- (drelevel), 46
dhead (dprint), 43
divide.conquer, 40
divide.conquer.timeReg, 41
dkeep (dcut), 37
dkeep <- (dcut), 37
dlag, 42
dlag <- (dlag), 42
dlev (drelevel), 46
dlev <- (drelevel), 46
dlevel (drelevel), 46
dlevel <- (drelevel), 46
dlevels (drelevel), 46
dlist (dprint), 43
dmean (dcor), 36
dmeansd (dcor), 36
dmvn (pmvn), 81
dnames (dcut), 37
dnames <- (dcut), 37
dnumeric (drelevel), 46
dnumeric <- (drelevel), 46
dprint, 43
dquantile (dcor), 36
drcumhaz, 44
dreg, 44
drelev (drelevel), 46
drelev <- (drelevel), 46
drelevel, 46
drelevel <- (drelevel), 46
drename (dcut), 37
drename <- (dcut), 37
dreshape (fast.reshape), 63
drm (dcut), 37
drm <- (dcut), 37
dsapply (blocksample), 16
dscalar (dcor), 36
dsds (dcor), 36
dsort, 48
dsort <- (dsort), 48
dspline, 49
dspline <- (dspline), 49
dstr (dcor), 36
dsubset (dcor), 36
dsum (dcor), 36
dsummary (dcor), 36
dtab (dtable), 50
dtable, 50
dtail (dprint), 43
dtransform (dtransform), 51
dtransform <- (dtransform), 51
dtransform <- (dtransform), 51
dunique (dcut), 37
easy.binomial.twostage, 52
easy.survival.twostage, 55
EVaddGam, 57
eventpois, 59
familycluster.index, 60
familyclusterWithProbands.index, 61
fast.approx, 62
fast.cluster (npc), 79
fastNpattern, 62
fastNreshape, 63
fasterNreshape (npc), 79
folds (npc), 79
forceNsameNcens (npc), 79
gofNphreg, 65
gofGphreg, 66
gofMphreg, 67
gofZ.phreg (gofM.phreg), 67
Grandom.cif, 68
groupable (npc), 79
ilap (npc), 79
ipw, 71
ipw2, 73
jumpimes (npc), 79
kendall.ClaytonOakes.twin.ace (npc), 79
kendall.normal.twin.ace (npc), 79
km, 74
lifecourse, 75
lifetable(lifetable.matrix), 76
lifetable.matrix, 76
LinSpline, 77
loglikMVN (pmvn), 81
makeNpairwiseNdesign (npc), 79
matdoubleindex (predictNphreg), 82
matplot.mets.twostage (npc), 79
mena, 78
mets-package, 4
mets.options, 78
migr, 79
multcif, 79
nonparcuminc (npc), 79
np, 79
npc, 79
object.defined (npc), 79
or.cif (cor.cif), 26
or2prob (tetrachoric), 106
p11.binoial.twostage.RV (npc), 79
pbvn (pmvn), 81
pcif (eventpois), 59
phreg, 80
piecewise.data (npc), 79
piecewise.twostage (npc), 79
plack.cif, 81
plack.cif2 (plack.cif), 81
plot.covariate.recurrent
covarianceRecurrent), 31
plotcr (npc), 79
pmvn, 81
predict.phreg, 82
predictPairPlack (npc), 79
print.casewise, 82
prob.exceed.recurrent, 83
prob.exceedBiRecurrence,
(prob.exceed.recurrent), 83
prob.exceedRecurrence,
(prob.exceed.recurrent), 83
procform (npc), 79
procform3 (npc), 79
procformdata (npc), 79
prt, 85
random.cif, 85
recmarg (recurrentMarginal), 87
recurrentMarginal, 87
recurrentMarginalgam (simRecurrent), 92
revcumsum (predictNphreg), 82
revcumsumidstratasum (predictNphreg), 82
revcumsumidstratasumCov
(predictNphreg), 82
revcumsumstrata (predictNphreg), 82
revcumsumstratasum (predictNphreg), 82
rmvn (pmvn), 81
robust.basehaz.phreg (predictNphreg), 82
robust.phreg (phreg), 80
rr.cif (cor.cif), 26
scoreMVN (pmvn), 81
showfitsim (simRecurrent), 92
sim (npc), 79
simAalenFrailty, 90
simbinClaytonOakes.family.ace (npc), 79
simbinClaytonOakes.pairs (npc), 79
simbinClaytonOakes.twin.ace (npc), 79
simBinFam (npc), 79
simBinFam2 (npc), 79
simBinPlack (npc), 79
simClaytonOakes, 91
simClaytonOakes.Family.ace (npc), 79
simClaytonOakes.twin.ace (npc), 79
INDEX

simClaytonOakesWei, 91
simCompete.simple (npc), 79
simCompete.twin.ace (npc), 79
simCox (npc), 79
simFrailty.simple (npc), 79
simnordic (npc), 79
simRecurrent, 92
simRecurrentGamma (simRecurrent), 92
simRecurrentII, 94
simSurvFam (npc), 79
slopecs.process (casewise.test), 20
squareintHdM (simRecurrent), 92
summary.cor, 95
sumstrata (predict.phreg), 82
surv.boxarea (npc), 79
survival.iterative, 97
survival.twostage, 100
test.conc, 105
tetrachoric, 106
tie.breaker (recurrentMarginal), 87
twin.clustertrunc, 107
twin.polygen.design (npc), 79
twinnbmi, 108
twinnlm, 19, 108, 110
twinnlm.strata, 19, 109
twinnlm.time, 19, 109
twinnlm.time (bptwin), 18
twinstim, 19, 109, 110
twinstut, 111
twostage.aalen (survival.twostage), 100
twostage.cox.aalen (survival.twostage), 100
twostage.coxph (survival.twostage), 100
twostage.fullse (npc), 79
twostage.phreg (survival.twostage), 100