

Package ‘mets’

August 30, 2025

Type Package

Title Analysis of Multivariate Event Times

Version 1.3.7

Date 2025-08-30

Author Klaus K. Holst [aut, cre],
Thomas Scheike [aut]

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>. Modern methods for survival analysis, including regression modelling (Cox, Fine-Gray, Ghosh-Lin, Binomial regression) with fast computation of influence functions.

License GPL (>= 2)

LazyLoad yes

URL <https://kkholst.github.io/mets/>

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

Depends R (>= 3.5)

Imports Rcpp, lava (>= 1.8.1), methods, mvtnorm, numDeriv, splines,
survival (>= 2.43-1), timereg (>= 1.9.4)

Suggests cmprsk, icenReg, KernSmooth, knitr, mets, optimx, prodlm,
rmarkdown, tinytest (>= 1.4.1), ucminf

VignetteBuilder knitr

ByteCompile yes

LinkingTo Rcpp, RcppArmadillo, mvtnorm

Encoding UTF-8

RoxygenNote 7.3.2

NeedsCompilation yes

Repository CRAN

Date/Publication 2025-08-30 17:00:02 UTC

Contents

aalenfrailty	5
aalenMets	6
ACTG175	7
back2timereg	7
basehazplot.phreg	8
bicomprisk	9
BinAugmentCifstrata	11
binomial.twostage	13
binreg	18
binregATE	21
binregCasewise	23
binregG	24
binregRatio	25
binregTSR	28
biprobit	31
blocksample	33
bmt	34
Bootphreg	35
bptwin	36
calgb8923	38
casewise	39
casewise.test	40
cif	41
cifreg	42
ClaytonOakes	44
cluster.index	46
concordanceCor	47
cor.cif	48
count.history	53
CPH_HPN_CRBSI	54
daggregate	54
Dbvn	56
dby	56
dcor	59
dcut	60
dermalridges	62
dermalridgesMZ	62
diabetes	63
divide.conquer	64
divide.conquer.timereg	64
dlag	65
doubleFGR	66
dprint	68
dreg	69
drelevel	72
dsort	74

dspline	75
dtable	76
dtransform	78
easy.binomial.twostage	79
Effbinreg	83
EVaddGam	85
evalTerminal	86
Event	87
event.split	88
eventpois	89
EventSplit2	90
familycluster.index	91
familyclusterWithProbands.index	92
fast.approx	93
fast.pattern	94
fast.reshape	95
FG_AugmentCifstrata	97
glm_IPTW	99
gof.phreg	100
gofG.phreg	101
gofM.phreg	102
gofZ.phreg	104
Grandom.cif	105
haplo	109
haplo.surv.discrete	109
hfactioncpx12	112
iidBaseline	112
interval.logitsurv.discrete	113
ipw	115
ipw2	116
km	118
lifecourse	119
lifetable.matrix	121
LinSpline	122
logitSurv	122
mediatorSurv	123
medweight	125
melanoma	125
mena	126
mets.options	127
migr	127
mlogit	128
multcif	129
np	130
npc	130
phreg	130
phregR	131
phreg_IPTW	132

phreg_rct	134
plack.cif	136
plot.phreg	136
plot_twin	137
pmvn	138
predict.phreg	139
predictRisk	140
predictRisk.binreg	141
print.casewise	141
prob.exceed.recurrent	142
prt	143
random.cif	143
rchaz	146
rchazC	147
rcrisk	148
recreg	149
recurrentMarginal	151
resmean.phreg	153
resmeanATE	154
resmeanIPCW	155
rpch	157
sim.cause.cox	158
sim.cif	159
sim.cox	160
sim.recurrent	161
simAalenFrailty	162
simClaytonOakes	163
simClaytonOakesWei	164
simGLcox	165
simMultistate	166
simRecurrentII	168
simRecurrentTS	170
summary.cor	172
summaryGLM	173
survival.twostage	174
survivalG	179
test.conc	181
tetrachoric	181
TRACE	182
ttpd	183
twin.clustertrunc	183
twinbmi	184
twinlm	185
twinsim	187
twinstut	188
twostageMLE	189
WA_recurrent	190

aalenfrailty	<i>Aalen frailty model</i>
--------------	----------------------------

Description

Additive hazards model with (gamma) frailty

Usage

```
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

```
library("timereg")
dd <- simAalenFrailty(5000)
f <- ~1##+x
X <- model.matrix(f,dd) ## design matrix for non-parametric terms
system.time(out<-timereg::aalen(update(f,Surv(time,status)~.),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
##plot(out,spec=spec)
```

```
## plot(dd$time[dix],bb$B2[,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

## End(Not run)
```

aalenMets

Fast additive hazards model with robust standard errors

Description

Fast Lin-Ying additive hazards model with a possibly stratified baseline. Robust variance is default variance with the summary.

Usage

```
aalenMets(formula, data = data, no.baseline = FALSE, ...)
```

Arguments

formula	formula with 'Surv' outcome (see coxph)
data	data frame
no.baseline	to fit model without baseline hazard
...	Additional arguments to phreg

Details

influence functions (iid) will follow numerical order of given cluster variable so ordering after \$id will give iid in order of data-set.

Author(s)

Thomas Scheike

Examples

```
data(bmt); bmt$time <- bmt$time+runif(408)*0.001
out <- aalenMets(Surv(time,cause==1)~tcell+platelet+age,data=bmt)
summary(out)

## out2 <- timereg::aalen(Surv(time,cause==1)~const(tcell)+const(platelet)+const(age),data=bmt)
## summary(out2)
```

ACTG175*ACTG175, block randomized study from speff2trial package*

Description

Data from speff2trial

Format

Randomized study

Source

Hammer et al. 1996, speff2trial package.

Examples

```
data(ACTG175)
```

back2timereg*Convert to timereg object*

Description

convert to timereg object

Usage

```
back2timereg(obj)
```

Arguments

obj no use

Author(s)

Thomas Scheike

basehazplot.phreg	<i>Plotting the baselines of stratified Cox</i>
-------------------	---

Description

Plotting the baselines of stratified Cox

Usage

```
basehazplot.phreg(
  x,
  se = FALSE,
  time = NULL,
  add = FALSE,
  ylim = NULL,
  xlim = NULL,
  lty = NULL,
  col = NULL,
  lwd = NULL,
  legend = TRUE,
  ylab = NULL,
  xlab = NULL,
  polygon = TRUE,
  level = 0.95,
  stratas = NULL,
  robust = FALSE,
  conf.type = c("plain", "log"),
  ...
)
```

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
lwd	to specify lwd of components
legend	to specify col of components
ylab	to specify ylab
xlab	to specify xlab

<code>polygon</code>	to get standard error in shaded form
<code>level</code>	of standard errors
<code>stratas</code>	wich strata to plot
<code>robust</code>	to use robust standard errors if possible
<code>conf.type</code>	"plain" or "log" transformed
<code>...</code>	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))
plot(out1)
plot(out1,stratas=c(0,3))
plot(out1,stratas=c(0,3),col=2:3,lty=1:2,se=TRUE)
plot(out1,stratas=c(0),col=2,lty=2,se=TRUE,polygon=FALSE)
plot(out1,stratas=c(0),col=matrix(c(2,1,3),1,3),lty=matrix(c(1,2,3),1,3),se=TRUE,polygon=FALSE)
```

bicomprisk

Estimation of concordance in bivariate competing risks data

Description

Estimation of concordance in bivariate competing risks data

Usage

```
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 specyng 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 timereg::comp.risk call for iid decomposition, max.clust=NULL uses all clusters otherwise rougher grouping.
marg	marginal cumulative incidence to make stanard 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 additonal 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 esimators are the same in the case of stratified fitting.
messages	Control amount of output
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 timereg::comp.risk function

Author(s)

Thomas Scheike, Klaus K. Holst

References

Scheike, T. H.; Holst, K. K. & Hjelmberg, J. B. Estimating twin concordance for bivariate competing risks twin data Statistics in Medicine, Wiley Online Library, 2014 , 33 , 1193-204

Examples

```
library("timereg")

## Simulated data example
prt <- simnordic.random(2000,delayed=TRUE,ptrunc=0.7,
  cordz=0.5,cormz=2,lam0=0.3)
## Bivariate competing risk, concordance estimates
p11 <- bicomprisk(Event(time,cause)~strata(zyg)+id(id),data=prt,cause=c(1,1))

p11mz <- p11$model$"MZ"
p11dz <- p11$model$"DZ"
par(mfrow=c(1,2))
## Concordance
plot(p11mz,ylim=c(0,0.1));
plot(p11dz,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)

prt22 <- fast.reshape(prt2,id="id")

prt22$event <- (prt22$cause1==1)*(prt22$cause2==1)*1
prt22$time1 <- pmax(prt22$time1,prt22$time2)
ipwc <- timereg::comp.risk(Event(time1,event)~-1+factor(zyg1),
  data=prt22,cause=1,n.sim=0,model="rcif2",times=50:90,
  weights=prt22$weights1,cens.weights=rep(1,nrow(prt22)))

p11wmz <- ipwc$cum[,2]
p11wdz <- ipwc$cum[,3]
lines(ipwc$cum[,1],p11wmz,col=3)
lines(ipwc$cum[,1],p11wdz,col=3)
```

Description

Computes the augmentation term for each individual as well as the sum

$$A = \int_0^t H(u, X) \frac{1}{S^*(u, s)} \frac{1}{G_c(u)} dM_c(u)$$

with

$$H(u, X) = F_1^*(t, s) - F_1^*(u, s)$$

using a KM for

$$G_c(t)$$

and a working model for cumulative baseline related to

$$F_1^*(t, s)$$

and

$$s$$

is strata,

$$S^*(t, s) = 1 - F_1^*(t, s) - F_2^*(t, s)$$

.

Usage

```
BinAugmentCifstrata(
  formula,
  data = data,
  cause = 1,
  cens.code = 0,
  km = TRUE,
  time = NULL,
  weights = NULL,
  offset = NULL,
  ...
)
```

Arguments

formula	formula with 'Event', strata model for CIF given by strata, and strataC specifies censoring strata
data	data frame
cause	of interest
cens.code	code of censoring
km	to use Kaplan-Meier
time	of interest
weights	weights for estimating equations
offset	offsets for logistic regression
...	Additional arguments to binreg function.

Details

Standard errors computed under assumption of correct

$$G_c(s)$$

model.

Author(s)

Thomas Scheike

Examples

```
library(mets)
data(bmt)
dcut(bmt,breaks=2) <- ~age
out1<-BinAugmentCifstrata(Event(time,cause)~platelet+agecat.2+
  strata(platelet,agecat.2),data=bmt,cause=1,time=40)
summary(out1)

out2<-BinAugmentCifstrata(Event(time,cause)~platelet+agecat.2+
  strata(platelet,agecat.2)+strataC(platelet),data=bmt,cause=1,time=40)
summary(out2)
```

binomial.twostage	<i>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 compososite likelihood based on all pairwise bivariate models.</i>
-------------------	---

Description

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

Usage

```
binomial.twostage(
  margbin,
  data = parent.frame(),
  method = "nr",
  detail = 0,
  clusters = NULL,
  silent = 1,
  weights = NULL,
  theta = NULL,
  theta.des = NULL,
  var.link = 0,
```

```

var.par = 1,
var.func = NULL,
iid = 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,
dim.theta = NULL,
additive.gamma.sum = NULL,
pair.ascertained = 0,
case.control = 0,
no.opt = FALSE,
twostage = 1,
beta = NULL,
...
)

```

Arguments

<code>margbin</code>	Marginal binomial model
<code>data</code>	data frame
<code>method</code>	Scoring method "nr", for lava NR optimizer
<code>detail</code>	Detail
<code>clusters</code>	Cluster variable
<code>silent</code>	Debug information
<code>weights</code>	Weights for log-likelihood, can be used for each type of outcome in 2x2 tables.
<code>theta</code>	Starting values for variance components
<code>theta.des</code>	design for dependence parameters, when pairs are given the indices of the theta-design for this pair, is given in pairs as column 5
<code>var.link</code>	Link function for variance
<code>var.par</code>	parametrization
<code>var.func</code>	when alternative parametrizations are used this function can specify how the parameters are related to the λ_j 's.
<code>iid</code>	Calculate i.i.d. decomposition when <code>iid</code> >=1, when <code>iid</code> =2 then avoids adding the uncertainty for marginal parameters for additive gamma model (default).
<code>notaylor</code>	Taylor expansion
<code>model</code>	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 the indices of the pairs random.design rows are given as columns 3:4
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.
dim.theta	dimension of theta when pairs and pairs specific design is given. That is when pairs has 6 columns.
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)
no.opt	for not optimizing
twostage	default twostage=1, to fit MLE use twostage=0
beta	is starting value for beta for MLE version
...	for NR of lava

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 v_1 and v_2 . Such that the random effects for subject 1 is

$$v_1^T(Z_1, \dots, Z_d)$$

, for d random effects. Each random effect has an associated parameter $(\lambda_1, \dots, \lambda_d)$. 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 $v_1^T(Z_1, \dots, Z_d)$ 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 λ_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 $(\lambda_1, \dots, \lambda_d)$ are related to the parameters of the model by a regression construction *pard* (d x k), that links the d λ parameters with the (k) underlying θ parameters

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

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

```

data(twinstut)
twinstut0 <- subset(twinstut, tvparnr<4000)
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)
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)

## Reduce Ex.Timings
## 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]=="dz",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)

```

```

### make ace random effects design
out <- ace.family.design(data,member="type",id="cluster")
out$parides
head(out$des.rv)

bints <- binomial.twostage(margbin,data=data,
  clusters=data$cluster,detail=0,var.par=1,
  theta=c(2,1),var.link=0,
  random.design=out$des.rv,theta.des=out$parides)
summary(bints)

data <- simbinClaytonOakes.twin.ace(10000,2,1,beta=NULL,alpha=NULL)
out <- twin.polygen.design(data,id="cluster",zygname="zygosity")
out$parides
head(out$des.rv)
margbin <- glm(ybin~x,data=data,family=binomial())

bintwin <- binomial.twostage(margbin,data=data,
  clusters=data$cluster,var.par=1,
  theta=c(2,1),random.design=out$des.rv,theta.des=out$parides)
summary(bintwin)
concordanceTwinACE(bintwin)

```

binreg

Binomial Regression for censored competing risks data

Description

Simple version of comp.risk function of timereg for just one time-point thus fitting the model

$$P(T \leq t, \epsilon = 1 | X) = \text{expit}(X^T \text{beta})$$

Usage

```

binreg(
  formula,
  data,
  cause = 1,
  time = NULL,
  beta = NULL,
  type = c("II", "I"),
  offset = NULL,
  weights = NULL,
  cens.weights = NULL,
  cens.model = ~+1,
  se = TRUE,
  kaplan.meier = TRUE,

```

```

    cens.code = 0,
    no.opt = FALSE,
    method = "nr",
    augmentation = NULL,
    outcome = c("cif", "rmst", "rmtl"),
    model = c("default", "logit", "exp", "lin"),
    Ydirect = NULL,
    ...
)

```

Arguments

formula	formula with outcome (see coxph)
data	data frame
cause	cause of interest (numeric variable)
time	time of interest
beta	starting values
type	"II" adds augmentation term, and "I" classic binomial regression
offset	offsets for partial likelihood
weights	for score equations
cens.weights	censoring weights
cens.model	only stratified cox model without covariates
se	to compute se's based on IPCW
kaplan.meier	uses Kaplan-Meier for IPCW in contrast to exp(-Baseline)
cens.code	gives censoring code
no.opt	to not optimize
method	for optimization
augmentation	to augment binomial regression
outcome	can do CIF regression "cif"= $F(t X)$, "rmst"= $E(\min(T, t) X)$, or years-lost "rmtl"= $E(I(\text{epsilon} == \text{cause}) (t - \min(T, t)) X)$
model	link functions used, with defaults logit for cif, exp for rmst or rmtl, but can be logit, exp or lin (for identity link)
Ydirect	use this Y instead of outcome constructed inside the program (e.g. $I(T < t, \text{epsilon} = 1)$), then uses IPCW version of the Y, set outcome to "rmst" to fit using the model specified by model
...	Additional arguments to lower level funtions

Details

Based on binomial regression IPCW response estimating equation:

$$X(\Delta^{ipcw}(t)I(T \leq t, \epsilon = 1) - \text{expit}(X^T \text{beta})) = 0$$

where

$$\Delta^{ipcw}(t) = I((\min(t, T) < C)/G_c(\min(t, T)-)$$

is IPCW adjustment of the response

$$Y(t) = I(T \leq t, \epsilon = 1)$$

(type="I") solves this estimating equation using a stratified Kaplan-Meier for the censoring distribution. For (type="II") the default an additional censoring augmentation term

$$X \int E(Y(t)|T > s)/G_c(s)d\hat{M}_c$$

is added.

logitIPCW instead considers

$$XI(\min(T_i, t) < G_i)/G_c(\min(T_i, t))(I(T \leq t, \epsilon = 1) - \expit(X^T \beta)) = 0$$

a standard logistic regression with weights that adjust for IPCW.

The variance is based on the squared influence functions that are also returned as the iid component. naive.var is variance under known censoring model.

Censoring model may depend on strata (cens.model=~strata(gX)).

Author(s)

Thomas Scheike

Examples

```
library(mets)
data(bmt); bmt$time <- bmt$time+runif(408)*0.001
# logistic regresion with IPCW binomial regression
out <- binreg(Event(time,cause)~tcell+platelet,bmt,time=50)
summary(out)
head(iid(out))

predict(out,data.frame(tcell=c(0,1),platelet=c(1,1)),se=TRUE)

outs <- binreg(Event(time,cause)~tcell+platelet,bmt,time=50,cens.model=~strata(tcell,platelet))
summary(outs)

## glm with IPCW weights
outl <- logitIPCW(Event(time,cause)~tcell+platelet,bmt,time=50)
summary(outl)

#####
### risk-ratio of different causes #####
#####
data(bmt)
bmt$id <- 1:nrow(bmt)
bmt$status <- bmt$cause
```

```

bmt$strata <- 1
bmtdob <- bmt
bmtdob$strata <- 2
bmtdob <- dtransform(bmtdob, status=1, cause==2)
bmtdob <- dtransform(bmtdob, status=2, cause==1)
###
bmtdob <- rbind(bmt, bmtdob)
dtable(bmtdob, cause+status~strata)

cif1 <- cif(Event(time, cause)~+1, bmt, cause=1)
cif2 <- cif(Event(time, cause)~+1, bmt, cause=2)
plot(cif1)
plot(cif2, add=TRUE, col=2)

cifs1 <- binreg(Event(time, cause)~tcell+platelet+age, bmt, cause=2, time=50)
cifs2 <- binreg(Event(time, cause)~tcell+platelet+age, bmt, cause=2, time=50)
summary(cifs1)
summary(cifs2)

cifdob <- binreg(Event(time, status)~-1+factor(strata)+
  tcell*factor(strata)+platelet*factor(strata)+age*factor(strata)
  +cluster(id), bmtdob, cause=1, time=50, cens.model=~strata(strata))
summary(cifdob)

riskratio <- function(p) {
  Z <- rbind(c(1,0,1,1,0,0,0,0), c(0,1,1,1,0,1,1,0))
  lp <- c(Z %*% p)
  p <- lava::expit(lp)
  return(p[1]/p[2])
}

lava::estimate(cifdob, f=riskratio)

```

binregATE

*Average Treatment effect for censored competing risks data using Bi-
nomial Regression*

Description

Under the standard causal assumptions we can estimate the average treatment effect $E(Y(1) - Y(0))$. We need Consistency, ignorability ($Y(1), Y(0)$ indep A given X), and positivity.

Usage

```

binregATE(
  formula,
  data,
  cause = 1,
  time = NULL,

```

```

    beta = NULL,
    treat.model = ~+1,
    cens.model = ~+1,
    offset = NULL,
    weights = NULL,
    cens.weights = NULL,
    se = TRUE,
    type = c("II", "I"),
    kaplan.meier = TRUE,
    cens.code = 0,
    no.opt = FALSE,
    method = "nr",
    augmentation = NULL,
    outcome = c("cif", "rmst", "rmtl"),
    model = c("default", "logit", "exp", "lin"),
    Ydirect = NULL,
    ...
)

```

Arguments

formula	formula with outcome (see coxph)
data	data frame
cause	cause of interest
time	time of interest
beta	starting values
treat.model	logistic treatment model given covariates
cens.model	only stratified cox model without covariates
offset	offsets for partial likelihood
weights	for score equations
cens.weights	censoring weights
se	to compute se's with IPCW adjustment, otherwise assumes that IPCW weights are known
type	"II" adds augmentation term, and "I" classic binomial regression
kaplan.meier	uses Kaplan-Meier for IPCW in contrast to exp(-Baseline)
cens.code	gives censoring code
no.opt	to not optimize
method	for optimization
augmentation	for augment binomial regression
outcome	can do CIF regression "cif"= $F(t X)$, "rmst"= $E(\min(T, t) X)$, or $E(I(\text{epsilon} == \text{cause}) (t - \min(T, t)) X)$ depending on the number of the number of causes.
model	exp or linear model for $E(\min(T, t) X) = \exp(X^t \beta)$, or $E(I(\text{epsilon} == \text{cause}) (t - \min(T, t)) X) = \exp(X^t \beta)$
Ydirect	use this outcome Y with IPCW version
...	Additional arguments to lower level functions

Details

The first covariate in the specification of the competing risks regression model must be the treatment variable that should be coded as a factor. If the factor has more than two levels then it uses the mlogit for propensity score modelling. If there are no censorings this is the same as ordinary logistic regression modelling.

Estimates the ATE using the the standard binary double robust estimating equations that are IPCW censoring adjusted. Rather than binomial regression we also consider a IPCW weighted version of standard logistic regression `logitIPCWATE`.

Author(s)

Thomas Scheike

Examples

```
data(bmt)
dfactor(bmt) <- ~.

brs <- binregATE(Event(time,cause)~tcell.f+platelet+age,bmt,time=50,cause=1,
  treat.model=tcell.f~platelet+age)
summary(brs)

brsi <- binregATE(Event(time,cause)~tcell.f+tcell.f*platelet+tcell.f*age,bmt,time=50,cause=1,
  treat.model=tcell.f~platelet+age)
summary(brsi)
```

binregCasewise	<i>Estimates the casewise concordance based on Concordance and marginal estimate using binreg</i>
----------------	---

Description

Estimates the casewise concordance based on Concordance and marginal estimate using binreg

Usage

```
binregCasewise(concbreg, margbreg, zygs = c("DZ", "MZ"), newdata = NULL, ...)
```

Arguments

concbreg	Concordance
margbreg	Marginal estimate
zygs	order of zygosity for estimation of concordance and casewise.
newdata	to give instead of zygs.
...	to pass to estimate function

Details

Uses cluster iid for the two binomial-regression estimates standard errors better than those of case-wise that are often conservative.

Author(s)

Thomas Scheike

Examples

```
data(prt)
prt <- force.same.cens(prt,cause="status")

dd <- bicompriskData(Event(time, status)~strata(zyg)+id(id), data=prt, cause=c(2, 2))
newdata <- data.frame(zyg=c("DZ","MZ"),id=1)

## concordance
bcif1 <- binreg(Event(time,status)~-1+factor(zyg)+cluster(id), data=dd,
                time=80, cause=1, cens.model=~strata(zyg))
pconc <- predict(bcif1,newdata)

## marginal estimates
mbcif1 <- binreg(Event(time,status)~cluster(id), data=prt, time=80, cause=2)
mc <- predict(mbcif1,newdata)
mc

cse <- binregCasewise(bcif1,mbcif1)
cse
```

binregG

G-estimator for binomial regression model (Standardized estimates)

Description

Computes G-estimator

$$\hat{F}(t, A = a) = n^{-1} \sum_i \hat{F}(t, A = a, Z_i)$$

. Assumes that the first covariate is \$A\$. Gives influence functions of these risk estimates and SE's are based on these. If first covariate is a factor then all contrast are computed, and if continuous then considered covariate values are given by Avalues.

Usage

```
binregG(x, data, Avalues = c(0, 1), varname = NULL)
```

Arguments

x	binreg object
data	data frame for risk averaging
Avalues	values to compare for first covariate A
varname	if given then averages for this variable, default is first variable

Author(s)

Thomas Scheike

Examples

```
data(bmt); bmt$time <- bmt$time+runif(408)*0.001
bmt$event <- (bmt$cause!=0)*1

b1 <- binreg(Event(time,cause)~age+tcell+platelet,bmt,cause=1,time=50)
sb1 <- binregG(b1,bmt,Avalues=c(0,1,2))
summary(sb1)
```

binregRatio

Percentage of years lost due to cause regression

Description

Estimates the percentage of the years lost that is due to a cause and how covariates affects this percentage by doing ICPW regression.

Usage

```
binregRatio(
  formula,
  data,
  cause = 1,
  time = NULL,
  beta = NULL,
  type = c("II", "I"),
  offset = NULL,
  weights = NULL,
  cens.weights = NULL,
  cens.model = ~+1,
  se = TRUE,
  kaplan.meier = TRUE,
  cens.code = 0,
  no.opt = FALSE,
  method = "nr",
  augmentation = NULL,
```

```

outcome = c("cif", "rmtl"),
model = c("logit", "exp", "lin"),
Ydirect = NULL,
...
)

```

Arguments

formula	formula with outcome (see coxph)
data	data frame
cause	cause of interest (numeric variable)
time	time of interest
beta	starting values
type	"II" adds augmentation term, and "I" classical outcome IPCW regression
offset	offsets for partial likelihood
weights	for score equations
cens.weights	censoring weights
cens.model	only stratified cox model without covariates
se	to compute se's based on IPCW
kaplan.meier	uses Kaplan-Meier for IPCW in contrast to exp(-Baseline)
cens.code	gives censoring code
no.opt	to not optimize
method	for optimization
augmentation	to augment binomial regression
outcome	can do CIF regression "cif"= $F(t X)$, "rmtl"= $E(t - \min(T, t) X)$
model	logit, exp or lin(ear)
Ydirect	use this Y instead of outcome constructed inside the program, should be a matrix with two column for numerator and denominator.
...	Additional arguments to lower level funtions

Details

Let the years lost be

$$Y1 = t - \min(T, t)$$

and the years lost due to cause 1

$$Y2 = I(\epsilon == 1)(t - \min(T, t))$$

, then we model the ratio

$$\text{logit}(E(Y2|X)/E(Y1|X)) = X^T \beta$$

. Estimation is based on on binomial regresion IPCW response estimating equation:

$$X(\Delta^{ipcw}(t)Y2\expit(X^T \beta) - Y1) = 0$$

where

$$\Delta^{ipcw}(t) = I((\min(t, T) < C)/G_c(\min(t, T)-)$$

is IPCW adjustment of the response

$$Y(t) = I(T \leq t, \epsilon = 1)$$

.

(type="I") solves this estimating equation using a stratified Kaplan-Meier for the censoring distribution. For (type="II") the default an additional censoring augmentation term

$$X \int E(Y(t)|T > s)/G_c(s)d\hat{M}_c$$

is added.

The variance is based on the squared influence functions that are also returned as the iid component. naive.var is variance under known censoring model.

Censoring model may depend on strata (cens.model=~strata(gX)).

Author(s)

Thomas Scheike

Examples

```
library(mets)
data(bmt); bmt$time <- bmt$time+runif(408)*0.001

rmst30 <- rmstIPCW(Event(time,cause!=0)~platelet+tcell+age,bmt,time=30,cause=1)
rmst301 <- rmstIPCW(Event(time,cause)~platelet+tcell+age,bmt,time=30,cause=1)
rmst302 <- rmstIPCW(Event(time,cause)~platelet+tcell+age,bmt,time=30,cause=2)

estimate(rmst30)
estimate(rmst301)
estimate(rmst302)

## percentage of total cumulative incidence due to cause 1
rmstratioI <- rmstRatio(Event(time,cause)~platelet+tcell+age,bmt,time=30,
                        cause=1,outcome="rmst")
summary(rmstratioI)

pp <- predict(rmstratioI,bmt)
ppb <- cbind(pp,bmt)

## percentage of total cumulative incidence due to cause 1
cifratio <- binregRatio(Event(time,cause)~platelet+tcell+age,bmt,time=30,cause=1)
summary(cifratio)
pp <- predict(cifratio,bmt)

rmstratioI <- binregRatio(Event(time,cause)~platelet+tcell+age,bmt,
                        time=30,cause=1,outcome="rmst")
summary(rmstratioI)
```

```
pp <- predict(rmstratioI,bmt)
ppb <- cbind(pp,bmt)
```

binregTSR

2 Stage Randomization for Survival Data or competing Risks Data

Description

Under two-stage randomization we can estimate the average treatment effect $E(Y(i,j))$ of treatment regime (i,j) . The estimator can be augmented in different ways: using the two randomizations and the dynamic censoring augmentation. The treatment's must be given as factors.

Usage

```
binregTSR(
  formula,
  data,
  cause = 1,
  time = NULL,
  cens.code = 0,
  response.code = NULL,
  augmentR0 = NULL,
  treat.model0 = ~+1,
  augmentR1 = NULL,
  treat.model1 = ~+1,
  augmentC = NULL,
  cens.model = ~+1,
  estpr = c(1, 1),
  response.name = NULL,
  offset = NULL,
  weights = NULL,
  cens.weights = NULL,
  beta = NULL,
  kaplan.meier = TRUE,
  no.opt = FALSE,
  method = "nr",
  augmentation = NULL,
  outcome = c("cif", "rmst", "rmst-cause"),
  model = "exp",
  Ydirect = NULL,
  return.data = 0,
  pi0 = 0.5,
  pi1 = 0.5,
  cens.time.fixed = 1,
  outcome.iid = 1,
  meanCs = 0,
```

```
    ...  
)
```

Arguments

formula	formula with outcome (see coxph)
data	data frame
cause	cause of interest
time	time of interest
cens.code	gives censoring code
response.code	code of status of survival data that indicates a response at which 2nd randomization is performed
augmentR0	augmentation model for 1st randomization
treat.model0	logistic treatment model for 1st randomization
augmentR1	augmentation model for 2nd randomization
treat.model1	logistic treatment model for 2nd randomization
augmentC	augmentation model for censoring
cens.model	stratification for censoring model based on observed covariates
estpr	estimate randomization probabilities using model
response.name	can give name of response variable, otherwise reads this as first variable of treat.model1
offset	not implemented
weights	not implemented
cens.weights	can be given
beta	starting values
kaplan.meier	for censoring weights, rather than exp cumulative hazard
no.opt	not implemented
method	not implemented
augmentation	not implemented
outcome	can be c("cif", "rmst", "rmst-cause")
model	not implemented, uses linear regression for augmentation
Ydirect	use this Y instead of outcome constructed inside the program (e.g. $I(T < t, \epsilon_{\text{silon}} = 1)$), see binreg for more on this
return.dataw	to return weighted data for all treatment regimes
pi0	set up known randomization probabilities
pi1	set up known randomization probabilities
cens.time.fixed	to use time-dependent weights for censoring estimation using weights
outcome.iid	to get iid contribution from outcome model (here linear regression working models).
meanCs	(0) indicates that censoring augmentation is centered by CensAugment.times/n
...	Additional arguments to lower level functions

Details

The solved estimating equation is

$$(I(\min(T_i, t) < G_i)/G_c(\min(T_i, t))I(T \leq t, \epsilon = 1) - AUG_0 - AUG_1 + AUG_C - p(i, j)) = 0$$

where using the covariates from augmentR0

$$AUG_0 = \frac{A_0(i) - \pi_0(i)}{\pi_0(i)} X_0 \gamma_0$$

and using the covariates from augmentR1

$$AUG_1 = \frac{A_0(i)}{\pi_0(i)} \frac{A_1(j) - \pi_1(j)}{\pi_1(j)} X_1 \gamma_1$$

and the censoring augmentation is

$$AUG_C = \int_0^t \gamma_c(s)^T (e(s) - \bar{e}(s)) \frac{1}{G_c(s)} dM_c(s)$$

where

$$\gamma_c(s)$$

is chosen to minimize the variance given the dynamic covariates specified by augmentC.

In the observational case, we can use propensity score modelling and outcome modelling (using linear regression).

Standard errors are estimated using the influence function of all estimators and tests of differences can therefore be computed subsequently.

Author(s)

Thomas Scheike

Examples

```
library(mets)
ddf <- mets::gsim(200, covs=1, null=0, cens=1, ce=2)

bb <- binregTSR(Event(entry, time, status)~+1+cluster(id), ddf$datat, time=2, cause=c(1),
  cens.code=0, treat.model0=A0.f~+1, treat.model1=A1.f~A0.f,
  augmentR1=~X11+X12+TR, augmentR0=~X01+X02,
  augmentC=~A01+A02+X01+X02+A11t+A12t+X11+X12+TR,
  response.code=2)
summary(bb)
```

biprobit

*Bivariate Probit model***Description**

Bivariate Probit model

Usage

```

biprobit(
  x,
  data,
  id,
  rho = ~1,
  num = NULL,
  strata = NULL,
  eqmarg = TRUE,
  indep = FALSE,
  weights = NULL,
  weights.fun = function(x) ifelse(any(x <= 0), 0, max(x)),
  randomeffect = FALSE,
  vcov = "robust",
  pairs.only = FALSE,
  allmarg = !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
num	Optional name of order variable
strata	Strata
eqmarg	If TRUE same marginals are assumed (exchangeable)
indep	Independence
weights	Weights
weights.fun	Function defining the bivariate weight in each cluster

<code>randomeffect</code>	If TRUE a random effect model is used (otherwise correlation parameter is estimated allowing for both negative and positive dependence)
<code>vcov</code>	Type of standard errors to be calculated
<code>pairs.only</code>	Include complete pairs only?
<code>allmarg</code>	Should all marginal terms be included
<code>control</code>	Control argument parsed on to the optimization routine. Starting values may be parsed as 'start'.
<code>messages</code>	Control amount of messages shown
<code>constrain</code>	Vector of parameter constraints (NA where free). Use this to set an offset.
<code>table</code>	Type of estimation procedure
<code>p</code>	Parameter vector p in which to evaluate log-Likelihood and score function
<code>...</code>	Optional arguments

Examples

```
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
n <- 2e3
x <- sort(runif(n, -1, 1))
y <- rmvn(n, c(0,0), rho=cbind(tanh(x)))>0
d <- data.frame(y1=y[,1], y2=y[,2], x=x)
dd <- fast.reshape(d)

a <- biprobit(y~1+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"))

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

pp <- predict(a, data.frame(x=seq(-1,1,by=.1)), which=c(9)) ## rho
plot(pp[,1]~pp$x, type="l", xlab="x", ylab="Correlation", lwd=2, xaxs="i")
lava::confband(pp$x, pp[,2], pp[,3], polygon=TRUE, lty=0, col=lava::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+I(cut(x,breaks=xp)), data=dd, id="id")
pp2 <- predict(a2, data.frame(x=xp[-1]-delta/2), which=c(9)) ## rho
lava::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)

## End(Not run)
```

blocksample	<i>Block sampling</i>
-------------	-----------------------

Description

Sample blockwise from clustered data

Usage

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

Arguments

data	Data frame
size	Size of samples
idvar	Column defining the clusters
replace	Logical indicating wether 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

```
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:
blocksample(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+z~id|x>0,size=5)
```

bmt

The Bone Marrow Transplant Data

Description

Bone marrow transplant data with 408 rows and 5 columns.

Format

The data has 408 rows and 5 columns.

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

time a numeric vector. Survival time.

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

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

age a numeric vector code. Age of patient, scaled and centered $((\text{age}-35)/15)$.

Source

Simulated data

References

NN

Examples

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

Bootphreg

Wild bootstrap for Cox PH regression

Description

wild bootstrap for uniform bands for Cox models

Usage

```
Bootphreg(
  formula,
  data,
  offset = NULL,
  weights = NULL,
  B = 1000,
  type = c("exp", "poisson", "normal"),
  ...
)
```

Arguments

formula	formula with 'Surv' outcome (see coxph)
data	data frame
offset	offsets for cox model
weights	weights for Cox score equations
B	bootstraps
type	distribution for multiplier
...	Additional arguments to lower level funtions

Author(s)

Klaus K. Holst, Thomas Scheike

References

Wild bootstrap based confidence intervals for multiplicative hazards models, Dobler, Pauly, and Scheike (2018),

Examples

```

n <- 100
x <- 4*rnorm(n)
time1 <- 2*rexp(n)/exp(x*0.3)
time2 <- 2*rexp(n)/exp(x*(-0.3))
status <- ifelse(time1<time2,1,2)
time <- pmin(time1,time2)
rbin <- rbinom(n,1,0.5)
cc <- rexp(n)*(rbin==1)+(rbin==0)*rep(3,n)
status <- ifelse(time < cc,status,0)
time <- ifelse(time < cc,time,cc)
data <- data.frame(time=time,status=status,x=x)

b1 <- Bootphreg(Surv(time,status==1)~x,data,B=1000)
b2 <- Bootphreg(Surv(time,status==2)~x,data,B=1000)
c1 <- phreg(Surv(time,status==1)~x,data)
c2 <- phreg(Surv(time,status==2)~x,data)

### exp to make all bootstraps positive
out <- pred.cif.boot(b1,b2,c1,c2,gplot=0)

cif.true <- (1-exp(-out$time))*0.5
with(out,plot(time,cif,ylim=c(0,1),type="l"))
lines(out$time,cif.true,col=3)
with(out,plotConfRegion(time,band.EE,col=1))
with(out,plotConfRegion(time,band.EE.log,col=3))
with(out,plotConfRegion(time,band.EE.log.o,col=2))

```

bptwin

Liability model for twin data

Description

Liability-threshold model for twin data

Usage

```

bptwin(
  x,
  data,
  id,
  zyg,
  DZ,
  group = NULL,
  num = NULL,
  weights = NULL,
  weights.fun = function(x) ifelse(any(x <= 0), 0, max(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,
    varlink,
    ...
)

```

Arguments

x	Formula specifying effects of covariates on the response.
data	data.frame with one observation pr row. In addition a column with the zygosity (DZ or MZ given as a factor) of each individual much 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)
weights.fun	Function defining a single weight each individual/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 "acde" (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
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)
```

calgb8923	<i>CALGB 8923, twostage randomization SMART design</i>
-----------	--

Description

Data from CALGB 8923

Format

Data from smart design id: id of subject status : 1-death, 2-response for second randomization, 0-censoring A0 : treament at first randomization A1 : treament at second randomization At.f : treament given at record (A0 or A1) TR : time of response sex : 0-males, 1-females consent: 1 if agrees to 2nd randomization, censored if not R: 1 if response trt1: A0 trt2: A1

Source

https://github.com/ycchao/code_Joint_model_SMART

Examples

```
data(calgb8923)
```

casewise	<i>Estimates the casewise concordance based on Concordance and marginal estimate using prodlim but no testing</i>
----------	---

Description

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

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

```
## Reduce Ex.Timings
library(prodlim)
data(prt);
prt <- force.same.cens(prt,cause="status")

### marginal cumulative incidence of prostate cancer##'
outm <- prodlim(Hist(time,status)~+1,data=prt)

times <- 60:100
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$model$"DZ"
cmz <- cc$model$"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	<i>Estimates the casewise concordance based on Concordance and marginal estimate using timereg and performs test for independence</i>
---------------	---

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 and sometimes only the uncertainty of the concordance estimates. This works pretty well, alternatively one can use also the functions Casewise for a specific time point

Author(s)

Thomas Scheike

Examples

```
## Reduce Ex.Timings
library("timereg")
data("prt", package="mets");
prt <- force.same.cens(prt, cause="status")

prt <- prt[which(prt$id %in% sample(unique(prt$id), 7500)),]
### marginal cumulative incidence of prostate cancer
times <- seq(60, 100, by=2)
outm <- timereg::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
cc <- 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 <- timereg::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)
cc <- 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

Usage

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

Arguments

formula	formula with 'Event' outcome and strata (only!)
data	data frame
cause	NULL looks at all, otherwise specify which cause to consider
cens.code	censoring code "0" is default, and death is cens.code!=0
death.code	alternative to cens.code give codes of death
...	Additional arguments to lower level funtions

Author(s)

Thomas Scheike

Examples

```
data(bmt)
bmt$cluster <- sample(1:100,408,replace=TRUE)
out1 <- cif(Event(time,cause)~+1,data=bmt,cause=1)
out2 <- cif(Event(time,cause)~+1+cluster(cluster),data=bmt,cause=1)

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

cifreg

*CIF regression***Description**

CIF logistic-link for propodds=1 default and CIF Fine-Gray (cloglog) regression for propodds=NULL. The FG model can also be called using the cifregFG function that has propodds=NULL.

Usage

```
cifreg(
  formula,
  data,
  propodds = 1,
  cause = 1,
  cens.code = 0,
  no.codes = NULL,
  ...
)
```

Arguments

formula	formula with 'Event' outcome
data	data frame
propodds	to fit logit link model, and propodds=NULL to fit Fine-Gray model
cause	of interest
cens.code	code of censoring
no.codes	certain event codes to be ignored when finding competing causes
...	Additional arguments to recreg

Details

For FG model:

$$\int (X - E)Y_1(t)w(t)dM_1$$

is computed and summed over clusters and returned multiplied with inverse of second derivative as iid.naive. Here

$$w(t) = G(t)(I(T_i \wedge t < C_i)/G_c(T_i \wedge t))$$

and

$$E(t) = S_1(t)/S_0(t)$$

and

$$S_j(t) = \sum X_i^j Y_{i1}(t) w_i(t) \exp(X_i^T \beta)$$

.

The iid decomposition of the beta's, however, also have a censoring term that is also is computed and added (still scaled with inverse second derivative)

$$\int (X - E)Y_1(t)w(t)dM_1 + \int q(s)/p(s)dM_c$$

and returned as the iid

For logistic link standard errors are slightly to small since uncertainty from recursive baseline is not considered, so for smaller data-sets it is recommended to use the prop.odds.subdist of timereg that is also more efficient due to use of different weights for the estimating equations. Alternatively, one can also bootstrap the standard errors.

Author(s)

Thomas Scheike

Examples

```
## data with no ties
library(mets)
data(bmt,package="mets")
bmt$time <- bmt$time+runif(nrow(bmt))*0.01
bmt$id <- 1:nrow(bmt)

## logistic link OR interpretation
or=cifreg(Event(time,cause)~tcell+platelet+age,data=bmt,cause=1)
summary(or)
par(mfrow=c(1,2))
plot(or)
nd <- data.frame(tcell=c(1,0),platelet=0,age=0)
por <- predict(or,nd)
plot(por)

## Fine-Gray model
fg=cifregFG(Event(time,cause)~tcell+platelet+age,data=bmt,cause=1)
summary(fg)
plot(fg)
```

```

nd <- data.frame(tcell=c(1,0),platelet=0,age=0)
pfg <- predict(fg,nd,se=1)
plot(pfg,se=1)

## not run to avoid timing issues
## gofFG(Event(time,cause)~tcell+platelet+age,data=bmt,cause=1)

sfg <- cifregFG(Event(time,cause)~strata(tcell)+platelet+age,data=bmt,cause=1)
summary(sfg)
plot(sfg)

### predictions with CI based on iid decomposition of baseline and beta
### these are used in the predict function above
fg <- cifregFG(Event(time,cause)~tcell+platelet+age,data=bmt,cause=1)
Biid <- iidBaseline(fg,time=20)
pfg1 <- FGprediid(Biid,nd)
pfg1

```

ClaytonOakes

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

<code>cluster</code>	Variable defining the clustering (if not given in the formula)
<code>var.formula</code>	Formula specifying the variance component structure (if not given via the cluster special function in the formula) using a linear model with log-link.
<code>cuts</code>	Cut points defining the piecewise constant hazard
<code>type</code>	when equal to <code>two.stage</code> , the Clayton-Oakes-Glidden estimator will be calculated via the <code>timereg</code> package
<code>start</code>	Optional starting values
<code>control</code>	Control parameters to the optimization routine
<code>var.invlink</code>	Inverse link function for variance structure model
<code>...</code>	Additional arguments

Author(s)

Klaus K. Holst

Examples

```

set.seed(1)
d <- subset(simClaytonOakes(500,4,2,1,stoptime=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

d2 <- simClaytonOakes(500,4,2,1,stoptime=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=Aalens 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

e0 <- ClaytonOakes(survival::Surv(time,status)~cluster(~1+factor(z),cluster),
                  cuts=c(0,0.5,1,2),data=d2)
##ts0 <- ClaytonOakes(survival::Surv(time,status)~cluster(~1,cluster),
##                    data=d2,type="two.stage")
##plot(ts0)
plot(e0)

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	<i>Finds subjects related to same cluster</i>
---------------	---

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

See Also

familycluster.index familyclusterWithProbands.index

Examples

```
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)
```

concordanceCor

*Concordance Computes concordance and casewise concordance***Description**

Concordance for Twins

Usage

```
concordanceCor(
  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 wich model that is considered if object not given.
coefs	Specifies dependence parameters if object is not given.
...	Extra arguments, not used.

Details

The concordance is the probability that both twins have experienced the event of interest and is defined as

$$cor(t) = P(T_1 \leq t, \epsilon_1 = 1, T_2 \leq t, \epsilon_2 = 1)$$

Similarly, the casewise concordance is

$$casewise(t) = \frac{cor(t)}{P(T_1 \leq t, \epsilon_1 = 1)}$$

that is the probability that twin "2" has the event given that twins "1" has.

Author(s)

Thomas Scheike

References

Estimating twin concordance for bivariate competing risks twin data Thomas H. Scheike, Klaus K. Holst and Jacob B. Hjelmberg, Statistics in Medicine 2014, 1193-1204

Estimating Twin Pair Concordance for Age of Onset. Thomas H. Scheike, Jacob V B Hjelmberg, Klaus K. Holst, 2015 in Behavior genetics DOI:10.1007/s10519-015-9729-3

cor.cif

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 :

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

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

```
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

<code>cif</code>	a model object from the <code>timereg::comp.risk</code> function with the marginal cumulative incidence of <code>cause1</code> , i.e., the event of interest, and whose odds the comparison is compared to the conditional odds given <code>cause2</code>
<code>data</code>	a <code>data.frame</code> with the variables.
<code>cause</code>	specifies the causes related to the death times, the value <code>cens.code</code> is the censoring value. When missing it comes from marginal <code>cif</code> .
<code>times</code>	time-vector that specifies the times used for the estimating equations for the cross-odds-ratio estimation.
<code>cause1</code>	specifies the cause considered.
<code>cause2</code>	specifies the cause that is conditioned on.
<code>cens.code</code>	specifies the code for the censoring if <code>NULL</code> then uses the one from the marginal <code>cif</code> model.
<code>cens.model</code>	specified which model to use for the ICPW, KM is Kaplan-Meier alternatively it may be "cox"
<code>Nit</code>	number of iterations for Newton-Raphson algorithm.
<code>detail</code>	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
dipar	dipar
score.method	"nlminb", can also use "nr".
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.
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 (2012), Biostatistics.

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

Examples

```
## Reduce Ex.Timings
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 <- timereg::comp.risk(Event(time,cause)~1+cluster(id),data=multcif,cause=1,
  n.sim=0,times=times,model="fg",max.clust=NULL)
add2 <- timereg::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
#####
set.seed(100)
prt<-simnordic.random(2000, cordz=2, cormz=5)
prt$status <- prt$cause
table(prt$status)

times <- seq(40,100,by=10)
cifmod <- timereg::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]) +
    pardes %*% c( par[3]*(t-60)/12,par[4]*(t-60)/12)
```

```

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="nr",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)
pardz <- (pardes[,2]==1) * par1
dpar <- cbind( parmz,pardz)
dpar
}
head(dgparfunc(rep(0.1,6),50,theta.des))
head(gparfunc(rep(0.1,6),50,theta.des))

or1g <- or.cif(cifmod,data=prt,cause1=1,cause2=1,
              theta.des=theta.des, same.cens=TRUE,
              par.func=gparfunc,dpar.func=dgparfunc,
              dimpar=6,score.method="nr",detail=1)
summary(or1g)
names(or1g)

```

```
head(or1g$theta.iid)
```

count.history	<i>Counts the number of previous events of two types for recurrent events processes</i>
---------------	---

Description

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

Usage

```
count.history(  
  data,  
  status = "status",  
  id = "id",  
  types = 1,  
  names.count = "Count",  
  lag = TRUE,  
  multitype = FALSE,  
  marks = NULL  
)
```

Arguments

data	data-frame
status	name of status
id	id
types	types of the events (code) related to status (multiple values possible)
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
multitype	if multitype is true then counts when status "in" types, otherwise counts for each value of type, types=c(1,2)
marks	values related to status ("in" types), counts marks for types, only when multitype=TRUE

Author(s)

Thomas Scheike

Examples

```
data(hfactioncpx12)
hf <- hfactioncpx12
dtable(hf,~status)
rr <- count.history(hf,types=1:2,id="id",status="status")
dtable(rr,~"Count*"+status,level=1)
```

CPH_HPNCRBSI	<i>Rates for HPN program for patients of Copenhagen Cohort</i>
--------------	--

Description

Rates for HPN program for patients of Copenhagen Cohort

Format

crbsi: cumulative rate of catheter related bloodstream infection in HPN patients of Copenhagen
mechanical: cumulative rate of Mechanical (hole/defect) complication for catheter of HPN patients of Copenhagen
trombo: cumulative rate of Occlusion/Thrombosis complication for catheter of HPN patients of Copenhagen
terminal: rate of terminal event, patients leaving the HPN program

Source

Estimated data

daggregate	<i>aggregating for for data frames</i>
------------	--

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

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

Examples

```

data("sTRACE")
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, "*Length*", x="Species", fun=head)
daggregate(iris, "^e.al", x="Species", fun=tail, regex=TRUE)
daggregate(sTRACE, status~ diabetes, fun=table)
daggregate(sTRACE, status~ diabetes+sex, fun=table)
daggregate(sTRACE, status + diabetes+sex ~ vf+I(wmi>1.4), fun=table)
daggregate(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)
drename(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
daggregate(iris,'sepal* '~species.n|species.n!=1,fun=summary)
```

Dbvn	<i>Derivatives of the bivariate normal cumulative distribution function</i>
------	---

Description

Derivatives of the bivariate normal cumulative distribution function

Usage

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

Arguments

p	Parameter vector
design	Design function with defines mean, derivative of mean, variance, and derivative of variance with respect to the parameter p
Y	column vector where the CDF is evaluated

Author(s)

Klaus K. Holst

dby	<i>Calculate summary statistics grouped by</i>
-----	--

Description

Calculate summary statistics grouped by variable

Usage

```
dby(  
  data,  
  INPUT,  
  ...,  
  ID = NULL,  
  ORDER = NULL,  
  SUBSET = NULL,  
  SORT = 0,  
  COMBINE = !REDUCE,  
  NOCHECK = FALSE,  
  ARGS = NULL,  
  NAMES,  
  COLUMN = FALSE,  
  REDUCE = FALSE,  
  REGEX = mets.options()$regex,  
  ALL = TRUE  
)
```

Arguments

data	Data.frame
INPUT	Input variables (character or formula)
...	functions
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

```

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(seq(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)
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
dby2(a,'y*'+ 'x*'\~id,mean)

## subset
dby(d, x<0) <- list(z=NA)
d
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,y=1)
dby(a,Species=="versicolor") <- list(y=2)

```

dcor

summary, tables, and correlations for data frames

Description

summary, tables, and correlations for data frames

Usage

```
dcor(data, y = NULL, x = NULL, use = "pairwise.complete.obs", ...)
```

Arguments

data	if x is formula or names for data frame then data frame is needed.
y	name of variable, or fomula, 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

```

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)

```

dcut

Cutting, sorting, rm (removing), rename for data frames

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

```
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 fomula, or names of variables on data frame.
x	name of variable, or fomula, 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
mm1 <- drm(mm,c("*.2","*.4"))
head(mm1)

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

## with direct asignment
drm(mm) <- c("*.2","*.4")
head(mm)

dcut(mm) <- c("age","*m*")
dcut(mm) <- ageg1+wmi1~age+wmi
head(mm)

#####
## renaming
#####

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

## all names to lower

```

```
drename(mm1) <- ~.
head(mm1)

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

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

dermalridges	<i>Dermal ridges data (families)</i>
--------------	--------------------------------------

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 moter, father and each child. Family id in 'family' and gender and child number in 'sex' and 'child'.

Source

Sarah B. Holt (1952). Genetics of dermal ridges: bilateral asymmetry in finger ridge-counts. Annals of Eugenics 17 (1), pp.211–231. DOI: 10.1111/j.1469-1809.1952.tb02513.x

Examples

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

dermalridgesMZ	<i>Dermal ridges data (monozygotic twins)</i>
----------------	---

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

Sarah B. Holt (1952). Genetics of dermal ridges: bilateral asymmetry in finger ridge-counts. *Annals of Eugenics* 17 (1), pp.211–231. DOI: 10.1111/j.1469-1809.1952.tb02513.x

Examples

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

diabetes

The Diabetic Retinopathy Data

Description

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

Format

This data frame contains the following columns:

id a numeric vector. Patient code.

agedx a numeric vector. Age of patient at diagnosis.

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

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

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

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

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

Source

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

Examples

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

divide.conquer	<i>Split a data set and run function</i>
----------------	--

Description

Split a data set and run function

Usage

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

Arguments

func	called function
data	data-frame
size	size of splits
splits	number of splits (ignored if size is given)
id	optional cluster variable
...	Additional arguments to lower level functions

Author(s)

Thomas Scheike, Klaus K. Holst

Examples

```
## avoid dependency on timereg
## 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	<i>Split a data set and run function from timereg and aggregate</i>
------------------------	---

Description

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

Usage

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

Arguments

func	called function
data	data-frame
size	size of splits
...	Additional arguments to lower level functions

Author(s)

Thomas Scheike, Klaus K. Holst

Examples

```
## library(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

```
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

```
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])
```

doubleFGR

*Double CIF Fine-Gray model with two causes***Description**

Estimation based on derived hazards and recursive estimating equations. fits two parametrizations
1)

$$F_1(t, X) = 1 - \exp(-\exp(X^T \beta) \Lambda_1(t))$$

and

$$F_2(t, X_2) = 1 - \exp(-\exp(X_2^T \beta_2) \Lambda_2(t))$$

or restricted version 2)

$$F_1(t, X) = 1 - \exp(-\exp(X^T \beta) \Lambda_1(t))$$

and

$$F_2(t, X_2, X) = (1 - \exp(-\exp(X_2^T \beta_2) \Lambda_2(t)))(1 - F_1(\infty, X))$$

Usage

```
doubleFGR(formula, data, offset = NULL, weights = NULL, X2 = NULL, ...)
```

Arguments

formula	formula with 'Event'
data	data frame
offset	offsets for cox model
weights	weights for Cox score equations
X2	specifies the regression design for second CIF model
...	Additional arguments to lower level funtions

Author(s)

Thomas Scheike

Examples

```

res <- 0
data(bmt)
bmt$age2 <- bmt$age
newdata <- bmt[1:19,]
if (interactive()) par(mfrow=c(5,3))

## same X1 and X2
pr2 <- doubleFGR(Event(time,cause)~age+platelet,data=bmt,restrict=res)
if (interactive()) {
  bplotdFG(pr2,cause=1)
  bplotdFG(pr2,cause=2,add=TRUE)
}
pp21 <- predictdFG(pr2,newdata=newdata)
pp22 <- predictdFG(pr2,newdata=newdata,cause=2)
if (interactive()) {
  plot(pp21)
  plot(pp22,add=TRUE,col=2)
}
pp21 <- predictdFG(pr2)
pp22 <- predictdFG(pr2,cause=2)
if (interactive()) {
  plot(pp21)
  plot(pp22,add=TRUE,col=2)
}

pr2 <- doubleFGR(Event(time,cause)~strata(platelet),data=bmt,restrict=res)
if (interactive()) {
  bplotdFG(pr2,cause=1)
  bplotdFG(pr2,cause=2,add=TRUE)
}
pp21 <- predictdFG(pr2,newdata=newdata)
pp22 <- predictdFG(pr2,newdata=newdata,cause=2)
if (interactive()) {
  plot(pp21)
  plot(pp22,add=TRUE,col=2)
}
pp21 <- predictdFG(pr2)
pp22 <- predictdFG(pr2,cause=2)
if (interactive()) {
  plot(pp21)
  plot(pp22,add=TRUE,col=2)
}

## different X1 and X2
pr2 <- doubleFGR(Event(time,cause)~age+platelet+age2,data=bmt,X2=3,restrict=res)
if (interactive()) {
  bplotdFG(pr2,cause=1)
  bplotdFG(pr2,cause=2,add=TRUE)
}
pp21 <- predictdFG(pr2,newdata=newdata)
pp22 <- predictdFG(pr2,newdata=newdata,cause=2)

```

```

if (interactive()) {
  plot(pp21)
  plot(pp22,add=TRUE,col=2)
}
pp21 <- predictdFG(pr2)
pp22 <- predictdFG(pr2,cause=2)
if (interactive()) {
  plot(pp21)
  plot(pp22,add=TRUE,col=2)
}

### uden X1
pr2 <- doubleFGR(Event(time,cause)~age+platelet,data=bmt,X2=1:2,restrict=res)
if (interactive()) {
  bplotdFG(pr2,cause=1)
  bplotdFG(pr2,cause=2,add=TRUE)
}
pp21 <- predictdFG(pr2,newdata=newdata)
pp22 <- predictdFG(pr2,newdata=newdata,cause=2)
if (interactive()) {
  plot(pp21)
  plot(pp22,add=TRUE,col=2)
}
pp21 <- predictdFG(pr2)
pp22 <- predictdFG(pr2,cause=2)
if (interactive()) {
  plot(pp21)
  plot(pp22,add=TRUE,col=2)
}

### without X2
pr2 <- doubleFGR(Event(time,cause)~age+platelet,data=bmt,X2=0,restrict=res)
if (interactive()) {
  bplotdFG(pr2,cause=1)
  bplotdFG(pr2,cause=2,add=TRUE)
}
pp21 <- predictdFG(pr2,newdata=newdata)
pp22 <- predictdFG(pr2,newdata=newdata,cause=2)
if (interactive()) {
  plot(pp21)
  plot(pp22,add=TRUE,col=2)
}
pp21 <- predictdFG(pr2)
pp22 <- predictdFG(pr2,cause=2)
if (interactive()) {
  plot(pp21)
  plot(pp22,add=TRUE,col=2)
}

```

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 fomula, 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

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

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)))
```

dreg	<i>Regression for data frames with dutility call</i>
------	--

Description

Regression for data frames with dutility call

Usage

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

Arguments

<code>data</code>	data frame
<code>y</code>	name of variable, or fomula, or names of variables on data frame.
<code>x</code>	name of variable, or fomula, or names of variables on data frame.
<code>z</code>	name of variable, or fomula, or names of variables on data frame.
<code>x.oneatotime</code>	x's one at a time
<code>x.base.names</code>	base covarirates
<code>z.arg</code>	what is Z, c("clever","base","group","condition"), clever decides based on type of Z, base means that Z is used as fixed baseline covaraite for all X, group means the analyses is done based on groups of Z, and condition means that Z specifies a condition on the data
<code>fun.</code>	function lm is default
<code>summary.</code>	summary to use
<code>regex</code>	regex
<code>convert</code>	convert
<code>doSummary</code>	doSummary or not
<code>special</code>	special's
<code>equal</code>	to do pairwise stuff
<code>test</code>	development argument
<code>...</code>	Additional arguments for fun

Author(s)

Klaus K. Holst, Thomas Scheike

Examples

```
##'
data(iris)
dat <- iris
drename(dat) <- ~.
names(dat)
set.seed(1)
dat$time <- runif(nrow(dat))
dat$time1 <- runif(nrow(dat))
dat$status <- rbinom(nrow(dat),1,0.5)
dat$S1 <- with(dat, Surv(time,status))
dat$S2 <- with(dat, Surv(time1,status))
dat$id <- 1:nrow(dat)

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

## Reduce Ex.Timings
y <- "S*"~"*.*.width"
xs <- dreg(dat, y, fun.=phreg)
## xs <- dreg(dat, y, fun.=survdiff)

y <- "S*"~"*.*.width"
xs <- dreg(dat, y, x.oneatotime=FALSE, fun.=phreg)

## under condition
y <- S1~"*.*.width"|I(species=="setosa" & sepal.width>3)
xs <- dreg(dat, y, z.arg="condition", fun.=phreg)
xs <- dreg(dat, y, fun.=phreg)

## under condition
y <- S1~"*.*.width"|species=="setosa"
xs <- dreg(dat, y, z.arg="condition", fun.=phreg)
xs <- dreg(dat, y, fun.=phreg)

## with baseline after |
y <- S1~"*.*.width"|sepal.length
xs <- dreg(dat, y, fun.=phreg)

## by group by species, not working
y <- S1~"*.*.width"|species
ss <- split(dat, paste(dat$species, dat$status))

xs <- dreg(dat, y, fun.=phreg)

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

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

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

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

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

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

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

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

if (interactive()) {
  dcut(dat) <- ~"*.*.width"
  y <- S1~"*.*.4"|I(sepal.length>4)
  par(mfrow=c(1, 2))
  xs <- dreg(dat, y, fun.=ff)
}

```

drelevel

relev levels for data frames

Description

levels shows levels for variables in data frame, relelevel relelevels a factor in data.frame

Usage

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

Arguments

<code>data</code>	if <code>x</code> is formula or names for data frame then data frame is needed.
<code>y</code>	name of variable, or fomula, or names of variables on data frame.
<code>x</code>	name of variable, or fomula, or names of variables on data frame.
<code>ref</code>	new reference variable
<code>newlevels</code>	to combine levels of factor in data frame
<code>regex</code>	for regular expressions.
<code>sep</code>	seperator for naming of cut names.
<code>overwrite</code>	to overwrite variable
<code>...</code>	Optional additional arguments

Author(s)

Klaus K. Holst and Thomas Scheike

Examples

```
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]")) <- zygdz+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

Value

data.frame

Examples

```
data(data="hubble",package="lava")
dsort(hubble, "sigma")
dsort(hubble, hubble$sigma,"v")
dsort(hubble,~sigma+v)
dsort(hubble,~sigma-v)

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

dspline

Simple linear spline

Description

Constructs simple linear spline on a data frame using the formula syntax of dutils that is adds (x-cuti)*(x>cuti) to the data-set for each knot of the spline. The full spline is thus given by x and spline variables added to the data-set.

Usage

```
dspline(
  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 fomula, or names of variables on data frame.
x	name of variable, or fomula, 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)

Thomas Scheike

Examples

```
data(TRACE)
TRACE <- dspline(TRACE,~wmi,breaks=c(1,1.3,1.7))
cca <- survival::coxph(Surv(time,status==9)~age+vf+chf+wmi,data=TRACE)
cca2 <- survival::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))
p1 <- predict(cca2,newdata=nd)
plot(nd$wmi,p1,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

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

Author(s)

Klaus K. Holst and Thomas Scheike

Examples

```
data("sTRACE", package="timereg")

dtable(sTRACE, ~status)
dtable(sTRACE, ~status+vf)
dtable(sTRACE, ~status+vf, level=1)
dtable(sTRACE, ~status+vf, ~chf+diabetes)

dtable(sTRACE, c("*f*", "status"), ~diabetes)
dtable(sTRACE, c("*f*", "status"), ~diabetes, level=2)
dtable(sTRACE, c("*f*", "status"), level=1)

dtable(sTRACE, ~"*f*" + status, level=1)
dtable(sTRACE, ~"*f*" + status + I(wmi>1.4) | age>60, level=2)
```

```

dtable(sTRACE, "~*f*" + status ~ I(wmi > 0.5) | age > 60, level = 1)
dtable(sTRACE, status ~ dcut(age))

dtable(sTRACE, ~status + vf + sex | age > 60)
dtable(sTRACE, status + vf + sex ~ 1 | age > 60, level = 2)
dtable(sTRACE, . ~status + vf + sex | age > 60, level = 1)
dtable(sTRACE, status + vf + sex ~ diabetes | age > 60)
dtable(sTRACE, status + vf + sex ~ diabetes | age > 60, flat = FALSE)

dtable(sTRACE, status + vf + sex ~ diabetes | age > 60, level = 1)
dtable(sTRACE, status + vf + sex ~ diabetes | age > 60, level = 2)

dtable(sTRACE, status + vf + sex ~ diabetes | age > 60, level = 2, prop = 1, total = TRUE)
dtable(sTRACE, status + vf + sex ~ diabetes | age > 60, level = 2, prop = 2, total = TRUE)
dtable(sTRACE, status + vf + sex ~ diabetes | age > 60, level = 2, prop = 1:2, summary = summary)

```

dtransform	<i>Transform that allows condition</i>
------------	--

Description

Defines new variables under condition for data frame

Usage

```
dtransform(data, ...)
```

Arguments

data	is data frame
...	new variable definitions including possible if condition

Examples

```

data(mena)

xx <- dtransform(mena, ll = log(agemena) + twinnum)

xx <- dtransform(mena, ll = log(agemena) + twinnum, agemena < 15)
xx <- dtransform(xx, ll = 100 + agemena, ll2 = 1000, agemena > 15)
dsummary(xx, ll + ll2 ~ I(agemena > 15))

```

`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 = parent.frame(),  
  method = "nr",  
  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

<code>margbin</code>	Marginal binomial model
<code>data</code>	data frame
<code>method</code>	Scoring method
<code>response</code>	name of response variable in data frame
<code>id</code>	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
iid	Calculate i.i.d. decomposition
step	Step size
model	model
marginal.p	vector of marginal probabilities
strata	strata for fitting
max.clust	max clusters used for i.i.d. decomposition
se.clusters	clusters for iid decomposition for robust standard errors

Details

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<4000)
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,
                        method="nr")

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)+age,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)*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 zygoty of first subject in each pair : zyg1
## could also use zyg2 (since zyg2=zyg1 within twinpair's)
## do not run t save time
# 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 <- 1000
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
  c(pp,mc,fc,cc)
}

udi <- easy.binomial.twostage(y~+1,data=binfam,
  response="y",id="id",
  theta.formula=desfsi,desnames=c("pp","mother-child","father-child","cc"))
summary(udi)

##now looking to see if interactions with age or age influences marginal models
##converting factors to numeric to make all involved covariates numeric
##to use desfsai2 rather than desfsai that works on binfam

nbinfam <- binfam
nbinfam$num <- as.numeric(binfam$num)
head(nbinfam)

desfsai <- function(x,num1="num1",num2="num2")
{
  pp <- (x[num1]=="m")*(x[num2]=="f")*1
  ### av age for pp=1 i.e parent pairs
  agepp <- ((as.numeric(x["age1"])+as.numeric(x["age2"])))/2-30)*pp
  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
  agecc <- ((as.numeric(x["age1"])+as.numeric(x["age2"])))/2-12)*cc
  c(pp,agepp,mc,fc,cc,agecc)
}

desfsai2 <- function(x,num1="num1",num2="num2")
{
  pp <- (x[num1]==1)*(x[num2]==2)*1
  agepp <- (((x["age1"]+x["age2"]))/2-30)*pp ### av age for pp=1 i.e parent pairs
  mc <- (x[num1]==1)*(x[num2]==3 | x[num2]==4)*1
  fc <- (x[num1]==2)*(x[num2]==3 | x[num2]==4)*1
  cc <- (x[num1]==3)*(x[num2]==3 | x[num2]==4)*1
  agecc <- ((x["age1"]+x["age2"])/2-12)*cc ### av age for children
  c(pp,agepp,mc,fc,cc,agecc)
}

udxai2 <- easy.binomial.twostage(y~x+age,data=binfam,
  response="y",id="id",
  theta.formula=desfsai,
  desnames=c("pp","pp-age","mother-child","father-child","cc","cc-age"))
```

```
summary(udxai2)
```

Effbinreg

Efficient IPCW for binary data

Description

Simple version of comp.risk function of timereg for just one time-point thus fitting the model

$$E(T \leq t|X) = \expit(X^T \text{beta})$$

Usage

```
Effbinreg(
  formula,
  data,
  cause = 1,
  time = NULL,
  beta = NULL,
  offset = NULL,
  weights = NULL,
  cens.weights = NULL,
  cens.model = ~+1,
  se = TRUE,
  kaplan.meier = TRUE,
  cens.code = 0,
  no.opt = FALSE,
  method = "nr",
  augmentation = NULL,
  h = NULL,
  MCAugment = NULL,
  ...
)
```

Arguments

formula	formula with outcome (see coxph)
data	data frame
cause	cause of interest
time	time of interest
beta	starting values
offset	offsets for partial likelihood
weights	for score equations
cens.weights	censoring weights

<code>cens.model</code>	only stratified cox model without covariates
<code>se</code>	to compute se's based on IPCW
<code>kaplan.meier</code>	uses Kaplan-Meier for IPCW in contrast to exp(-Baseline)
<code>cens.code</code>	gives censoring code
<code>no.opt</code>	to not optimize
<code>method</code>	for optimization
<code>augmentation</code>	to augment binomial regression
<code>h</code>	h for estimating equation
<code>MCAugment</code>	iid of h and censoring model
<code>...</code>	Additional arguments to lower level funtions
<code>model</code>	exp or linear

Details

Based on binomial regression IPCW response estimating equation:

$$X(\Delta(T \leq t)/G_c(T_i-) - \text{expit}(X^T \text{beta})) = 0$$

for IPCW adjusted responses.

Based on binomial regression IPCW response estimating equation:

$$h(X)X(\Delta(T \leq t)/G_c(T_i-) - \text{expit}(X^T \text{beta})) = 0$$

for IPCW adjusted responses where h is given as an argument together with iid of censoring with h . By using appropriately the h argument we can also do the efficient IPCW estimator estimator this works the `prepsurv` and `prepcf` for survival or competing risks data. In this case also the censoring martingale should be given for variance calculation and this also comes out of the `prepsurv` or `prepcf` functions. (Experimental version at this stage).

Variance is based on

$$\sum w_i^2$$

also with IPCW adjustment, and `naive.var` is variance under known censoring model.

Censoring model may depend on strata.

Author(s)

Thomas Scheike

EVaddGam	<i>Relative risk for additive gamma model</i>
----------	---

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

Eriksson and Scheike (2015), Additive Gamma frailty models for competing risks data, Biometrics (2015)

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
##EVaddGam(rep(1,6),x[1,],x[3,],thetades,matrix(1,18,6))

# thetades <- dout$theta.des[,nrow(out)/2]
# x <- dout$random.design[,nrow(out)/2]
##EVaddGam(rep(1,6),x[1,],x[4,],thetades,matrix(1,18,6))

```

evalTerminal	<i>Evaluates piece constant covariates at $\min(D,t)$ where D is a terminal event</i>
--------------	---

Description

returns $X(\min(D,t))$ and $\min(D,t)$ and their ratio. for censored observation 0. to use with the IPCW models implemented.

Usage

```

evalTerminal(
  formula,
  data = data,
  death.code = 2,
  time = NULL,
  marks = NULL,
  mark.codes = NULL
)

```

Arguments

formula	formula with 'Event' outcome and X to evaluate at $\min(D,t)$
data	data frame
death.code	codes for death (terminating event, 2 default)
time	for evaluation
marks	for terminal events to add $\text{marks} * I(D \leq t, \text{epsilon "in" mark.codes})$ to $X(\min(D,t))$
mark.codes	gives death codes for which to add mark value

Author(s)

Thomas Scheike

Event

Event history object

Description

Constructur for Event History objects

Usage

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

Arguments

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

Details

... content for details

Value

Object of class Event (a matrix)

Author(s)

Klaus K. Holst and Thomas Scheike

Examples

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

event.split	<i>event.split (SurvSplit).</i>
-------------	---------------------------------

Description

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

Usage

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

Arguments

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

Author(s)

Thomas Scheike

Examples

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

```

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

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

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

#####
### same but for situation with multiple events along the time-axis
#####
d <- data.frame(event1=1:5+runif(5)*0.5,start=1:5,time=2*1:5,
status=rbinom(5,1,0.5),x=1:5,start0=0)
d$event2 <- d$event1+0.2
d$event2[4:5] <- NA
d

d0 <- event.split(d,cuts="event1",name.start="start",time="time",status="status")
d0
###
d00 <- event.split(d0,cuts="event2",name.start="start",time="time",status="status")
d00

```

eventpois

Extract survival estimates from lifetable analysis

Description

Summary for survival analyses via the 'lifetable' function

Usage

```

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

EventSplit2

Event split with two time-scales, time and gaptime

Description

Cuts time for two time-scales, as event.split

Usage

```
EventSplit2(
  data,
  time = "time",
  status = "status",
  entry = "start",
  cuts = "cuts",
  name.id = "id",
  gaptime = NULL,
  gaptime.entry = NULL,
  cuttime = c("time", "gaptime"),
  cens.code = 0,
  order.id = TRUE
)
```

Arguments

data	data to be split
time	time variable.
status	status variable.
entry	name of entry variable.
cuts	cuts variable or numeric cut (only one value)
name.id	name of id variable.
gaptime	gaptime variable.
gaptime.entry	name of entry variable for gaptime.
cuttime	to cut after time or gaptime
cens.code	code for the censoring.
order.id	order data after id and start.

Author(s)

Thomas Scheike

Examples

```
rr <- data.frame(time=c(500,1000),start=c(0,500),status=c(1,1),id=c(1,1))
rr$gaptime <- rr$time-rr$start
rr$gapstart <- 0

rr1 <- EventSplit2(rr,cuts=600,cuttime="time", gaptime="gaptime",gaptime.entry="gapstart")
rr2 <- EventSplit2(rr1,cuts=100,cuttime="gaptime",gaptime="gaptime",gaptime.entry="gapstart")

dlist(rr1,start-time+status+gapstart+gaptime~id)
dlist(rr2,start-time+status+gapstart+gaptime~id)
```

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 indeces
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

```
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

```
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)
```

fast.reshape

Fast reshape

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)
...	Optional additional arguments

Author(s)

Thomas Scheike, Klaus K. Holst

Examples

```
m <- lava::lvm(c(y1,y2,y3,y4)~x)
d <- lava::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 separator "" (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 <- lava::sim(lava::lvm(~y1+y2+ya),10)
d
(dd <- fast.reshape(d,"y"))
fast.reshape(d,"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 <- lava::lvm(c(y1,y2,y3,y4)~ x+z1+z3+z5)
d <- lava::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"))
fable(cancer1~cancer2,data=prtw)
rm(prtw)
```

FG_AugmentCifstrata	<i>Augmentation for Fine-Gray model based on stratified NPMLE Cif (Aalen-Johansen)</i>
---------------------	--

Description

Computes the augmentation term for each individual as well as the sum

$$A(\beta) = \int H(t, X, \beta) \frac{F_2^*(t, s)}{S^*(t, s)} \frac{1}{G_c(t)} dM_c$$

with

$$H(t, X, \beta) = \int_t^\infty (X - E(\beta, t)) G_c(t) d\Lambda_1^* i(t, s)$$

using a KM for

$$G_c(t)$$

and a working model for cumulative baseline related to

$$F_1^*(t, s)$$

and

$$s$$

is strata,

$$S^*(t, s) = 1 - F_1^*(t, s) - F_2^*(t, s)$$

, and

$$E(\beta^p, t)$$

is given. Assumes that no strata for baseline of ine-Gay model that is augmented.

Usage

```
FG_AugmentCifstrata(
  formula,
  data = data,
  E = NULL,
  cause = NULL,
  cens.code = 0,
  km = TRUE,
  case.weights = NULL,
  weights = NULL,
  offset = NULL,
  ...
)
```

Arguments

formula	formula with 'Event', strata model for CIF given by strata, and strataC specifies censoring strata
data	data frame
E	from FG-model
cause	of interest
cens.code	code of censoring
km	to use Kaplan-Meier
case.weights	weights for FG score equations (that follow dN_1)
weights	weights for FG score equations
offset	offsets for FG model
...	Additional arguments to lower level funtions

Details

After a couple of iterations we end up with a solution of

$$\int (X - E(\beta))Y_1(t)w(t)dM_1 + A(\beta)$$

the augmented FG-score.

Standard errors computed under assumption of correct

$$G_c$$

model.

Author(s)

Thomas Scheike

Examples

```
set.seed(100)
rho1 <- 0.2; rho2 <- 10
n <- 400
beta=c(0.0,-0.1,-0.5,0.3)
dats <- simul.cifs(n,rho1,rho2,beta,rc=0.2)
dtable(dats,~status)
dsort(dats) <- ~time
fg <- cifreg(Event(time,status)~Z1+Z2,data=dats,cause=1,propodds=NULL)
summary(fg)

fgaugS <- FG_AugmentCifstrata(Event(time,status)~Z1+Z2+strata(Z1,Z2),data=dats,cause=1,E=fg$E)
summary(fgaugS)
fgaugS2 <- FG_AugmentCifstrata(Event(time,status)~Z1+Z2+strata(Z1,Z2),data=dats,cause=1,E=fgaugS$E)
summary(fgaugS2)
```

glm_IPTW

IPTW GLM, Inverse Probability of Treatment Weighted GLM

Description

Fits GLM model with treatment weights

$$w(A) = \sum_a I(A = a) / P(A = a | X)$$

, computes standard errors via influence functions that are returned as the IID argument. Propensity scores are fitted using either logistic regression (glm) or the multinomial model (mlogit) when more than two categories for treatment. The treatment needs to be a factor and is identified on the rhs of the "treat.model".

Usage

```
glm_IPTW(  
  formula,  
  data,  
  treat.model = NULL,  
  family = binomial(),  
  id = NULL,  
  weights = NULL,  
  estpr = 1,  
  pi0 = 0.5,  
  ...  
)
```

Arguments

formula	for glm
data	data frame for risk averaging
treat.model	propensity score model (binary or multinomial)
family	of glm (logistic regression)
id	cluster id for standard errors
weights	may be given, and then uses weights*w(A) as the weights
estpr	to estimate propensity scores and get influence function contribution to uncertainty
pi0	fixed simple weights
...	arguments for glm call

Details

Also works with cluster argument.

Author(s)

Thomas Scheike

<i>gof.phreg</i>	<i>GOF for Cox PH regression</i>
------------------	----------------------------------

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, robust = NULL, ...)
```

Arguments

object	is phreg object
n.sim	number of simulations for score processes
silent	to show timing estimate will be produced for longer jobs
robust	to control whether robust $dM_i(t)$ or dN_i are used for simulations
...	Additional arguments to lower level functions

Author(s)

Thomas Scheike and Klaus K. Holst

Examples

```
library(mets)
data(sTRACE)

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

m1 <- phreg(Surv(time,status==9)~strata(vf)+chf+diabetes,data=sTRACE)
## to get Martingale ~ dN based simulations
gg <- gof(m1)
gg

## to get Martingale robust simulations, specify cluster in call
sTRACE$id <- 1:500
m1 <- phreg(Surv(time,status==9)~vf+chf+diabetes+cluster(id),data=sTRACE)
gg <- gof(m1)
gg

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

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

x	phreg object
sim	to simulate som variation from cox model to put on graph
silent	to keep it absolutely silent
lm	add line to plot, regressing the cumulatives on each other
...	Additional arguments to lower level funtions

Author(s)

Thomas Scheike and Klaus K. Holst

Examples

```
data(tTRACE)

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

gofG.phreg(m1)
gofG.phreg(m2)

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

gofM.phreg

GOF for Cox covariates in PH regression

Description

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

Usage

```
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

Details

That is, computes

$$U(t) = \int_0^t M^t d\hat{M}$$

and resamples its asymptotic distribution.

This will show if the residuals are consistent with the model. Typically, M will be a design matrix for the continuous covariates that gives for example the quartiles, and then the plot will show if for the different quartiles of the covariate the risk prediction is consistent over time (time x covariate interaction).

Author(s)

Thomas Scheike and Klaus K. Holst

Examples

```
library(mets)
data(TRACE)
set.seed(1)
TRACEsam <- blocksample(TRACE,idvar="id",replace=FALSE,100)

dcut(TRACEsam) <- ~.
mm <- model.matrix(~-1+factor(wmicat.4),data=TRACEsam)
m1 <- gofM.phreg(Surv(time,status==9)~vf+chf+wmi,data=TRACEsam,modelmatrix=mm)
summary(m1)
if (interactive()) {
  par(mfrow=c(2,2))
  plot(m1)
}

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

## cumulative sums in covariates, via design matrix mm
mm <- cumContr(TRACEsam$wmi,breaks=10,equi=TRUE)
```

```
m1 <- gofM.phreg(Surv(time,status==9)~strata(vf)+chf+wmi,data=TRACEsam,
  modelmatrix=mm,silent=0)
summary(m1)
```

gofZ.phreg

GOF for Cox covariates in PH regression

Description

That is, computes

$$U(z, \tau) = \int_0^\tau M(z)^t d\hat{M}$$

and resamples its asymptotic distribution.

Usage

```
gofZ.phreg(
  formula,
  data,
  vars = NULL,
  offset = NULL,
  weights = NULL,
  breaks = 50,
  equi = FALSE,
  n.sim = 1000,
  silent = 1,
  ...
)
```

Arguments

formula	formula for cox regression
data	data for model
vars	which variables to test for linearity
offset	offset
weights	weights
breaks	number of breaks for cumulatives in covariate direction
equi	equidistant breaks or not
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

Details

This will show if the residuals are consistent with the model evaluated in the z covariate. M is here chosen based on a grid (z_1, ..., z_m) and the different columns are $I(Z_i \leq z_l)$. for $l = 1, \dots, m$. The process in z is resampled to find extreme values. The time-points of evaluation is by default 50 points, chosen as 2

The p-value is valid but depends on the chosen grid. When the number of break points are high this will give the original test of Lin, Wei and Ying for linearity, that is also computed in the timereg package.

Author(s)

Thomas Scheike and Klaus K. Holst

Examples

```
library(mets)
data(TRACE)
set.seed(1)
TRACEsam <- blocksample(TRACE,idvar="id",replace=FALSE,100)

## cumulative sums in covariates, via design matrix mm
## Reduce Ex.Timings
m1 <- gofZ.phreg(Surv(time,status==9)~strata(vf)+chf+wmi+age,data=TRACEsam)
summary(m1)
plot(m1,type="z")
```

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 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 additive form

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

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 = "nr",
    entry = NULL,
    estimator = 1,
    trunkp = 1,
    admin.cens = NULL,
    random.design = NULL,
    ...
)

```

Arguments

<code>cif</code>	a model object from the <code>timereg::comp.risk</code> function with the marginal cumulative incidence of <code>cause2</code> , i.e., the event that is conditioned on, and whose odds the comparison is made with respect to
<code>data</code>	a <code>data.frame</code> with the variables.
<code>cause</code>	specifies the causes related to the death times, the value <code>cens.code</code> is the censoring value.
<code>cif2</code>	specifies model for <code>cause2</code> if different from <code>cause1</code> .
<code>times</code>	time points
<code>cause1</code>	cause of first coordinate.
<code>cause2</code>	cause of second coordinate.
<code>cens.code</code>	specifies the code for the censoring if <code>NULL</code> then uses the one from the marginal <code>cif</code> model.
<code>cens.model</code>	specified which model to use for the ICPW, KM is Kaplan-Meier alternatively it may be "cox"
<code>Nit</code>	number of iterations for Newton-Raphson algorithm.
<code>detail</code>	if 0 no details are printed during iterations, if 1 details are given.
<code>clusters</code>	specifies the cluster structure.
<code>theta</code>	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" optimizer, alternatively, use the "nr" 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	specifies 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 $n \times d$. 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

$$v_1^T (Z_1, \dots, Z_d)$$

, for d random effects. Each random effect has an associated parameter $(\lambda_1, \dots, \lambda_d)$. 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 $v_1^T (Z_1, \dots, Z_d)$ has mean 1 and variance $1/(v_1^T \lambda)$.

The parameters $(\lambda_1, \dots, \lambda_d)$ are related to the parameters of the model by a regression construction $pard$ ($d \times k$), that links the d λ parameters with the (k) underlying θ 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, Hjelmberg (2014), LIDA, Estimating heritability for cause specific hazards based on twin data

Examples

```
## Reduce Ex.Timings
d <- simnordic.random(5000,delayed=TRUE,
  cordz=1.0,cormz=2,lam0=0.3,country=TRUE)
times <- seq(50,90,by=10)
addm <- timereg::comp.risk(Event(time,cause)~-1+factor(country)+cluster(id),data=d,
  times=times,cause=1,max.clust=NULL)

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

out1m<-random.cif(addm,data=d,cause1=1,cause2=1,theta=1,
  theta.des=mm,same.cens=TRUE)
summary(out1m)

## this model can also be formulated as a random effects model
## but with different parameters
out2m<-Grandom.cif(addm,data=d,cause1=1,cause2=1,
  theta=c(0.5,1),step=1.0,
  random.design=mm,same.cens=TRUE)
summary(out2m)
1/out2m$theta
out1m$theta

#####
##### ACE modelling of twin data #####
#####
### assume that zygbin gives the zygosity of mono and dizygotic twins
### 0 for mono and 1 for dizygotic twins. We now formulate an AC model
zygbin <- d$zyg=="DZ"

n <- nrow(d)
### random effects for each cluster
des.rv <- cbind(mm,(zygbin==1)*rep(c(1,0)),(zygbin==1)*rep(c(0,1)),1)
### design making parameters half the variance for dizygotic components
pardes <- rbind(c(1,0), c(0.5,0),c(0.5,0), c(0.5,0), c(0,1))

outacem <-Grandom.cif(addm,data=d,cause1=1,cause2=1,
```

```
same.cens=TRUE,theta=c(0.35,0.15),
      step=1.0,theta.des=par.des,random.design=des.rv)
summary(outacem)
```

haplo	<i>haplo fun data</i>
-------	-----------------------

Description

haplo fun data

Format

hapfreqs : haplo frequencies
haploX: covariates and response for haplo survival
ghaplos: haplo-types for subjects of haploX data

Source

Estimated data

haplo.surv.discrete	<i>Discrete time to event haplo type analysis</i>
---------------------	---

Description

Can be used for logistic regression when time variable is "1" for all id.

Usage

```
haplo.surv.discrete(
  X = NULL,
  y = "y",
  time.name = "time",
  Haplos = NULL,
  id = "id",
  desnames = NULL,
  designfunc = NULL,
  beta = NULL,
  no.opt = FALSE,
  method = "NR",
  stderr = TRUE,
  designMatrix = NULL,
  response = NULL,
  idhap = NULL,
```

```

    design.only = FALSE,
    covnames = NULL,
    fam = binomial,
    weights = NULL,
    offsets = NULL,
    idhapweights = NULL,
    ...
)

```

Arguments

<code>X</code>	design matrix data-frame (sorted after id and time variable) with id time response and desnames
<code>y</code>	name of response (binary response with logistic link) from <code>X</code>
<code>time.name</code>	to sort after time for <code>X</code>
<code>Haplos</code>	(data.frame with id, haplo1, haplo2 (haplotypes (h)) and $p=P(h G)$) haplotypes given as factor.
<code>id</code>	name of id variable from <code>X</code>
<code>desnames</code>	names for design matrix
<code>designfunc</code>	function that computes design given haplotypes $h=(h1,h2)$ $x(h)$
<code>beta</code>	starting values
<code>no.opt</code>	optimization TRUE/FALSE
<code>method</code>	NR, nlm
<code>stderr</code>	to return only estimate
<code>designMatrix</code>	gives response and designMatrix directly not implemented (must contain: p, id, idhap)
<code>response</code>	gives response and design directly designMatrix not implemented
<code>idhap</code>	name of id-hap variable to specify different haplotypes for different id
<code>design.only</code>	to return only design matrices for haplo-type analyses.
<code>covnames</code>	names of covariates to extract from object for regression
<code>fam</code>	family of models, now binomial default and only option
<code>weights</code>	weights following id for GLM
<code>offsets</code>	following id for GLM
<code>idhapweights</code>	weights following id-hap for GLM (WIP)
<code>...</code>	Additional arguments to lower level functions lava::NR optimizer or nlm

Details

Cycle-specific logistic regression of haplo-type effects with known haplo-type probabilities. Given observed genotype G and unobserved haplotypes H we here mix out over the possible haplotypes using that $P(H|G)$ is provided.

$$S(t|x, G) = E(S(t|x, H)|G) = \sum_{h \in G} P(h|G)S(t|z, h)$$

so survival can be computed by mixing out over possible h given g .

Survival is based on logistic regression for the discrete hazard function of the form

$$\text{logit}(P(T = t|T \geq t, x, h)) = \alpha_t + x(h)\beta$$

where $x(h)$ is a regression design of x and haplotypes $h = (h_1, h_2)$

Likelihood is maximized and standard errors assumes that $P(H|G)$ is known.

The design over the possible haplotypes is constructed by merging X with Haplos and can be viewed by `design.only=TRUE`

Author(s)

Thomas Scheike

Examples

```
## some haplotypes of interest
types <- c("DCGCGCTCACG", "DTCCGCTGACG", "ITCAGTTGACG", "ITCCGCTGAGG")

## some haplotypes frequencies for simulations
data(haplo)
hapfreqs <- haplo$hapfreqs

www <- which(hapfreqs$haplotype %in% types)
hapfreqs$freq[www]

baseline=hapfreqs$haplotype[9]
baseline

designftypes <- function(x,sm=0) {# {{
  hap1=x[1]
  hap2=x[2]
  if (sm==0) y <- 1*( (hap1==types) | (hap2==types))
  if (sm==1) y <- 1*(hap1==types) + 1*(hap2==types)
  return(y)
}# }}

tcoef=c(-1.93110204,-0.47531630,-0.04118204,-1.57872602,-0.22176426,-0.13836416,
0.88830288,0.60756224,0.39802821,0.32706859)

ghaplos <- haplo$ghaplos
haploX <- haplo$haploX

haploX$time <- haploX$times
Xdes <- model.matrix(~factor(time),haploX)
colnames(Xdes) <- paste("X",1:ncol(Xdes),sep="")
X <- dkeep(haploX,~id+y+time)
X <- cbind(X,Xdes)
```

```
Haplos <- dkeep(ghaplos,~id+"haplo*"+p)
desnames=paste("X",1:6,sep="") # six X's related to 6 cycles
out <- haplo.surv.discrete(X=X,y="y",time.name="time",
  Haplos=Haplos,desnames=desnames,designfunc=designftypes)
names(out$coef) <- c(desnames,types)
out$coef
summary(out)
```

hfactioncp12	<i>hfaction, subset of block randomized study HF-ACTION from WA package</i>
--------------	---

Description

Data from HF-action trial slightly modified from WA package, consisting of 741 nonischemic patients with baseline cardiopulmonary test duration less than or equal to 12 minutes.

Format

Randomized study status : 1-event, 2-death, 0-censoring treatment : 1/0

Source

WA package, Connor et al. 2009

Examples

```
data(hfactioncp12)
```

iidBaseline	<i>Influence functions or IID decomposition of baseline for recrec/phreg/cifregFG</i>
-------------	---

Description

Influence functions or IID decomposition of baseline for recrec/phreg/cifregFG

Usage

```
iidBaseline(
  object,
  time = NULL,
  ft = NULL,
  fixbeta = NULL,
  beta.iid = NULL,
  tminus = FALSE,
  ...
)
```

Arguments

object	phreg/recreg/cifregFG object
time	for baseline IID
ft	function to compute IID of baseline integrated against f(t)
fixbeta	to fix the coefficients
beta.iid	to use these iid of beta
tminus	to get predictions in t-
...	additional arguments to lower level functions

Author(s)

Thomas Scheike

interval.logitsurv.discrete

Discrete time to event interval censored data

Description

We consider the cumulative odds model

$$P(T \leq t|x) = \frac{G(t) \exp(x\beta)}{1 + G(t)\exp(x\beta)}$$

or equivalently

$$\text{logit}(P(T \leq t|x)) = \log(G(t)) + x\beta$$

and we can thus also compute the probability of surviving

$$P(T > t|x) = \frac{1}{1 + G(t)\exp(x\beta)}$$

Usage

```
interval.logitsurv.discrete(
  formula,
  data,
  beta = NULL,
  no.opt = FALSE,
  method = "NR",
  stderr = TRUE,
  weights = NULL,
  offsets = NULL,
  exp.link = 1,
  increment = 1,
  ...
)
```

Arguments

formula	formula
data	data
beta	starting values
no.opt	optimization TRUE/FALSE
method	NR, nlm
stderr	to return only estimate
weights	weights following id for GLM
offsets	following id for GLM
exp.link	parametrize increments $\exp(\alpha) > 0$
increment	using increments $dG(t)=\exp(\alpha)$ as parameters
...	Additional arguments to lower level funtions lava::NR optimizer or nlm

Details

The baseline $G(t)$ is written as $\text{cumsum}(\exp(\alpha))$ and this is not the standard parametrization that takes log of $G(t)$ as the parameters. Note that the regression coefficients are describing the probability of dying before or at time t .

Input are intervals given by $[t_l, t_r]$ where t_r can be infinity for right-censored intervals. When truly discrete $[0, 1]$ will be an observation at 1, and $[j, j+1]$ will be an observation at $j+1$. Can be used for fitting the usual ordinal regression model (with logit link) that in contrast, however, describes the probability of surviving time t (thus leads to $-\beta$).

Likelihood is maximized:

$$\prod P(T_i > t_{il}|x) - P(T_i > t_{ir}|x)$$

Author(s)

Thomas Scheike

Examples

```
library(mets)
data(ttpd)
dtable(ttpd, ~entry+time2)

out <- interval.logitsurv.discrete(Interval(entry, time2)~X1+X2+X3+X4, ttpd)
summary(out)
head(iid(out))

pred <- predictlogitSurv(d(out, se=FALSE))
plotSurv(d(pred))

ttpd <- dfactor(ttpd, fentry~entry)
out <- cumoddsreg(fentry~X1+X2+X3+X4, ttpd)
summary(out)
head(iid(out))
```

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 = FALSE,
  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 Aalens 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")
prtw <- ipw(Surv(time, status==0)~country, data=prt[sample(nrow(prt), 5000), ],
            cluster="id", weight.name="w")
plot(0, type="n", xlim=range(prtw$time), ylim=c(0, 1), xlab="Age", ylab="Probability")
count <- 0
for (l in unique(prtw$country)) {
  count <- count+1
  prtw <- prtw[order(prtw$time), ]
  with(subset(prtw, country==l),
       lines(time, w, col=count, lwd=2))
}
legend("topright", legend=unique(prtw$country), col=1:4, pch=-1, lty=1)

## End(Not run)
```

ipw2

*Inverse Probability of Censoring Weights***Description**

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

Usage

```
ipw2(
  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 $\tau = \max(\text{times})$, 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 (bi-variate 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

```

library("timereg")
set.seed(1)
d <- simnordic.random(5000, delayed=TRUE, ptrunc=0.7,
  cordz=0.5, cormz=2, lam0=0.3, country=FALSE)
d$strata <- as.numeric(d$country)+(d$zyg=="MZ")*4

```

```

times <- seq(60,100,by=10)
## c1 <- timereg::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)
## pc1 <- predict(c1,X=1,se=0)
## plot(pc1)
##
## dl <- d[!d$truncated,]
## dl <- ipw2(dl,cluster="id",same.cens=TRUE,time="time",entrytime="entry",cause="cause",
##          strata="strata",prec.factor=100)
## cl <- timereg::comp.risk(Event(time,cause)~1+
## cluster(id),
## data=dl,cause=1,model="fg",
## weights=dl$indi.weights,cens.weights=rep(1,nrow(dl)),
## times=times,max.clust=NULL,n.sim=0)
## pcl <- predict(cl,X=1,se=0)
## lines(pcl$time,pcl$P1,col=2)
## mm=model.matrix(~1+factor(zyg),data=dl)
## out2<-random.cif(cl,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 and obtained from the predict call

Usage

```
km(formula, data = data, ...)
```

Arguments

formula	formula with 'Surv' 'Event' outcome
data	data frame
...	Additional arguments to phreg

Author(s)

Thomas Scheike

Examples

```

library(mets)
data(sTRACE)
sTRACE$cluster <- sample(1:100,500,replace=TRUE)
out1 <- km(Surv(time,status==9)~strata(vf,chf),data=sTRACE)
out2 <- km(Surv(time,status==9)~strata(vf,chf)+cluster(cluster),data=sTRACE)

summary(out1,times=1:3)
summary(out2,times=1:3)

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

```

lifecourse

*Life-course plot***Description**

Life-course plot for event life data with recurrent events

Usage

```

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

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

l1 = lifecourse(Event(start,slut,status)~id,data,id="id",group=~group,col=1:2)
op <- par(mfrow=c(1,2))
l1 = 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	<i>Life table</i>
------------------	-------------------

Description

Create simple life table

Usage

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

## S3 method for class 'formula'
lifetable(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==9)~sex+vf,breaks=c(0,0.2,0.5,8.5)))
lava::estimate(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")
```

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

 logitSurv

Proportional odds survival model

Description

Semiparametric Proportional odds model, that has the advantage that

$$\text{logit}(S(t|x)) = \log(\Lambda(t)) + x\beta$$

so covariate effects give OR of survival.

Usage

```
logitSurv(formula, data, offset = NULL, weights = NULL, ...)
```

Arguments

formula	formula with 'Surv' outcome (see coxph)
data	data frame
offset	offsets for exp(x beta) terms
weights	weights for score equations
...	Additional arguments to lower level funtions

Details

This is equivalent to using a hazards model

$$Z\lambda(t)\exp(x\beta)$$

where Z is gamma distributed with mean and variance 1.

Author(s)

Thomas Scheike

References

The proportional odds cumulative incidence model for competing risks, Eriksson, Frank and Li, Jianing and Scheike, Thomas and Zhang, Mei-Jie, Biometrics, 2015, 3, 687–695, 71,

Examples

```
data(TRACE)
dcut(TRACE) <- ~.
out1 <- logitSurv(Surv(time,status==9)~vf+chf+strata(wmicat.4),data=TRACE)
summary(out1)
gof(out1)
plot(out1)
```

mediatorSurv

Mediation analysis in survival context

Description

Mediation analysis in survival context with robust standard errors taking the weights into account via influence function computations. Mediator and exposure must be factors. This is based on numerical derivative wrt parameters for weighting. See vignette for more examples.

Usage

```
mediatorSurv(
  survmodel,
  weightmodel,
  data = data,
  wdata = wdata,
  id = "id",
  silent = TRUE,
  ...
)
```

Arguments

survmodel	with mediation model (binreg, aalenMets, phreg)
weightmodel	mediation model
data	for computations
wdata	weighted data expansion for computations
id	name of id variable, important for SE computations
silent	to be silent
...	Additional arguments to survival model

Author(s)

Thomas Scheike

Examples

```
library(mets)
n <- 400
dat <- kumarsimRCT(n, rho1=0.5, rho2=0.5, rct=2, censpar=c(0,0,0,0),
  beta = c(-0.67, 0.59, 0.55, 0.25, 0.98, 0.18, 0.45, 0.31),
  treatmodel = c(-0.18, 0.56, 0.56, 0.54), restrict=1)
dfactor(dat) <- dnr.f~dnr
dfactor(dat) <- gp.f~gp
drename(dat) <- ttt24~"ttt24*"
dat$id <- 1:n
dat$time <- 1

weightmodel <- fit <- glm(gp.f~dnr.f+preauto+ttt24, data=dat, family=binomial)
wdata <- medweight(fit, data=dat)

### fitting models with and without mediator
aaMss2 <- binreg(Event(time, status)~gp+dnr+preauto+ttt24+cluster(id), data=dat, time=50, cause=2)
aaMss22 <- binreg(Event(time, status)~dnr+preauto+ttt24+cluster(id), data=dat, time=50, cause=2)

### estimating direct and indirect effects (under strong strong assumptions)
aaMss <- binreg(Event(time, status)~dnr.f0+dnr.f1+preauto+ttt24+cluster(id),
  data=wdata, time=50, weights=wdata$weights, cause=2)
## to compute standard errors , requires numDeriv
library(numDeriv)
ll <- mediatorSurv(aaMss, fit, data=dat, wdata=wdata)
summary(ll)
## not run bootstrap (to save time)
## bll <- BootmediatorSurv(aaMss, fit, data=dat, k.boot=500)
```

medweight	<i>Computes mediation weights</i>
-----------	-----------------------------------

Description

Computes mediation weights for either binary or multinomial mediators. The important part is that the influence functions can be obtained to compute standard errors.

Usage

```
medweight(  
  fit,  
  data = data,  
  var = NULL,  
  name.weight = "weights",  
  id.name = "id",  
  ...  
)
```

Arguments

fit	either glm-binomial or mlogit (mets package)
data	data frame with data
var	is NULL reads mediator and exposure from formulae in the fit.
name.weight	name of weights
id.name	name of id variable, important for SE computations
...	Additional arguments to

Author(s)

Thomas Scheike

melanoma	<i>The Melanoma Survival Data</i>
----------	-----------------------------------

Description

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

Format

This data frame contains the following columns:

- no** a numeric vector. Patient code.
- status** a numeric vector code. Survival status. 1: dead from melanoma, 2: alive, 3: dead from other cause.
- days** a numeric vector. Survival time.
- ulc** a numeric vector code. Ulceration, 1: present, 0: absent.
- thick** a numeric vector. Tumour thickness (1/100 mm).
- sex** a numeric vector code. 0: female, 1: male.

Source

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

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

Examples

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

mena	<i>Menarche data set</i>
------	--------------------------

Description

Menarche data set

Source

Simulated data

mets.options	<i>Set global options for mets</i>
--------------	------------------------------------

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

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

migr	<i>Migraine data</i>
------	----------------------

Description

Migraine data

mlogit	<i>Multinomial regression based on phreg regression</i>
--------	---

Description

Fits multinomial regression model

$$P_i = \frac{\exp(X_i^\beta)}{\sum_{j=1}^K \exp(X_j^\beta)}$$

for

$$i = 1, \dots, K$$

where

$$\beta_1 = 0$$

, such that

$$\sum_j P_j = 1$$

using phreg function. Therefore the ratio

$$\frac{P_i}{P_1} = \exp(X_i^\beta)$$

Usage

mlogit(formula, data, offset = NULL, weights = NULL, fix.X = FALSE, ...)

Arguments

formula	formula with outcome (see coxph)
data	data frame
offset	offsets for partial likelihood
weights	for score equations
fix.X	to have same coefficients for all categories
...	Additional arguments to lower level functions

Details

Coefficients give log-Relative-Risk relative to baseline group (first level of factor, so that it can reset by relevel command). Standard errors computed based on sandwich form

$$DU^{-1} \sum U_i^2 DU^{-1}$$

.
Can also get influence functions (possibly robust) via iid() function, response should be a factor.
Can fit cumulative odds model as a special case of interval.logitsurv.discrete

Author(s)

Thomas Scheike

Examples

```
library(mets)
data(bmt)
bmt$id <- sample(200,408,replace=TRUE)
dfactor(bmt) <- cause1f~cause
drelevel(bmt,ref=3) <- cause3f~cause
dlevels(bmt)

mreg <- mlogit(cause1f~+1+cluster(id),bmt)
summary(mreg)
head(iid(mreg))
dim(iid(mreg))

mreg <- mlogit(cause1f~tcell+platelet,bmt)
summary(mreg)
head(iid(mreg))
dim(iid(mreg))

mreg3 <- mlogit(cause3f~tcell+platelet,bmt)
summary(mreg3)

## inverse information standard errors
lava::estimate(coef=mreg3$coef,vcov=mreg3$II)

## predictions based on seen response or not
newdata <- data.frame(tcell=c(1,1,1),platelet=c(0,1,1),cause1f=c("2","1","0"))
## all probabilities
predict(mreg,newdata,response=FALSE)
## only probability of seen response
predict(mreg,newdata)
```

multcif

Multivariate Cumulative Incidence Function example data set

Description

Multivariate Cumulative Incidence Function example data set

Source

Simulated data

np	<i>np data set</i>
----	--------------------

Description

np data set

Source

Simulated data

npc	<i>For internal use</i>
-----	-------------------------

Description

For internal use

Author(s)

Klaus K. Holst

phreg	<i>Fast Cox PH regression</i>
-------	-------------------------------

Description

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

Usage

```
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 funtions |

Details

influence functions (iid) will follow numerical order of given cluster variable so ordering after \$id will give iid in order of data-set.

Author(s)

Klaus K. Holst, Thomas Scheike

Examples

```
library(mets)
data(TRACE)
dcut(TRACE) <- ~.
out1 <- phreg(Surv(time,status==9)~vf+chf+strata(wmicat.4)+cluster(id),data=TRACE)
summary(out1)

par(mfrow=c(1,2))
plot(out1)

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

## iid decomposition, with scaled influence functions
## for regression parameters
head(iid(out1))
## making iid decomposition of baseline at a specific time-point
Aiiid <- iid(out1,time=30)
head(Aiiid)
## both iid decompositions
dd <- iidBaseline(out1,time=30)
head(dd$beta.iid)
head(dd$base.iid)
```

phregR

Fast Cox PH regression and calculations done in R to make play and adjustments easy

Description

Fast Cox PH regression with R implementation to play and adjust in R function: FastCoxPLstrataR

Usage

```
phregR(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 funtions

Details

Robust variance is default variance with the summary.

influence functions (iid) will follow numerical order of given cluster variable so ordering after \$id will give iid in order of data-set.

Author(s)

Klaus K. Holst, Thomas Scheike

phreg_IPTW

IPTW Cox, Inverse Probaibility of Treatment Weighted Cox regression

Description

Fits Cox model with treatment weights

$$w(A) = \sum_a I(A = a) / \pi(a|X)$$

, where

$$\pi(a|X) = P(A = a|X)$$

. Computes standard errors via influence functions that are returned as the IID argument. Propensity scores are fitted using either logistic regression (glm) or the multinomial model (mlogit) when there are than treatment categories. The treatment needs to be a factor and is identified on the rhs of the "treat.model". Recurrent events can be considered with start,stop structure and then cluster(id) must be specified. Robust standard errors are computed in all cases.

Usage

```
phreg_IPTW(
  formula,
  data,
  treat.model = NULL,
  treat.var = NULL,
  weights = NULL,
  estpr = 1,
  pi0 = 0.5,
  se.cluster = NULL,
  ...
)
```

Arguments

<code>formula</code>	for <code>phreg</code>
<code>data</code>	data frame for risk averaging
<code>treat.model</code>	propensity score model (binary or multinomial)
<code>treat.var</code>	a 1/0 variable that indicates when treatment is given and the propensity score is computed
<code>weights</code>	may be given, and then uses <code>weights*w(A)</code> as the weights
<code>estpr</code>	(=1, default) to estimate propensity scores and get influence function contribution to uncertainty
<code>pi0</code>	fixed simple weights
<code>se.cluster</code>	to compute GEE type standard errors when additional cluster structure is present
<code>...</code>	arguments for <code>phreg</code> call

Details

Time-dependent propensity score weights can also be computed when `treat.var` is used, it must be 1 at the time of first (A_0) and 2nd treatment (A_1), then uses weights

$$w_0(A_0) * w_1(A_1)^{t > T_r}$$

where

$$T_r$$

is time of 2nd randomization.

Author(s)

Thomas Scheike

Examples

```
library(mets)
data <- mets::simLT(0.7,100,beta=0.3,betac=0,ce=1,betao=0.3)
dfactor(data) <- Z.f~Z
out <- phreg_IPTW(Surv(time,status)~Z.f,data=data,treat.model=Z.f~X)
summary(out)
```

phreg_rct

*Lu-Tsiatis More Efficient Log-Rank for Randomized studies with baseline covariates***Description**

Efficient implementation of the Lu-Tsiatis improvement using baseline covariates, extended to competing risks and recurrent events. Results almost equivalent with the speffSurv function of the speff2trial function in the survival case. A dynamic censoring augmentation regression is also computed to gain even more from the censoring augmentation. Further, we also deal with twostage randomizations. The function was implemented to deal with recurrent events (start,stop) + cluster, and more examples in vignette.

Usage

```
phreg_rct(
  formula,
  data,
  cause = 1,
  cens.code = 0,
  typesR = c("R0", "R1", "R01"),
  typesC = c("C", "dynC"),
  weights = NULL,
  augmentR0 = NULL,
  augmentR1 = NULL,
  augmentC = NULL,
  treat.model = ~+1,
  RCT = TRUE,
  treat.var = NULL,
  km = TRUE,
  level = 0.95,
  cens.model = NULL,
  estpr = 1,
  pi0 = 0.5,
  base.augment = FALSE,
  return.augmentR0 = FALSE,
  mlogit = FALSE,
  ...
)
```

Arguments

formula	formula with 'Surv' or 'Event' outcome (see coxph) and treatment (randomization 0/1)
data	data frame
cause	to use for competing risks, recurrent events data

cens.code	to use for competing risks, recurrent events data
typesR	augmentations used for randomization
typesC	augmentations used for censoring
weights	weights for score equation
augmentR0	formula for the randomization augmentation (~age+sex)
augmentR1	formula for the randomization augmentation (~age+sex)
augmentC	formula for the censoring augmentation (~age+sex)
treat.model	propensity score model, default is ~+1, assuming an RCT study
RCT	if false will use propensity score adjustment for marginal model
treat.var	in case of twostage randomization, this variable is 1 for the treatment times, if start,stop then default assumes that only one treatment at first record
km	use Kaplan-Meier for the censoring weights (stratified on treatment)
level	of confidence intervals
cens.model	default is censoring model ~strata(treatment) but any model can be used to make censoring martingales
estpr	estimates propensity scores
pi0	possible fixed propensity scores for randomizations
base.augment	TRUE to covariate augment baselines (only for R0 augmentation)
return.augmentR0	to return augmentation data
mlogit	if TRUE then forces use of this function for propensity scores, default for binary treatment is glm
...	Additional arguments to phreg function

Author(s)

Thomas Scheike

References

- Lu, Tsiatis (2008), Improving the efficiency of the log-rank test using auxiliary covariates, *Biometrika*, 679–694
- Scheike et al. (2024), WIP, Two-stage randomization for recurrent events,

Examples

```
## Lu, Tsiatis simulation
data <- mets::simLT(0.7,100)
dfactor(data) <- Z.f~Z

out <- phreg_rct(Surv(time,status)~Z.f,data=data,augmentR0=~X,augmentC=~factor(Z):X)
summary(out)
```

plack.cif	<i>plack Computes concordance for or.cif based model, that is Plackett random effects model</i>
-----------	---

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

plot.phreg	<i>Plotting the baselines of stratified Cox</i>
------------	---

Description

Plotting the baselines of stratified Cox

Usage

```
## S3 method for class 'phreg'
plot(x, ...)
```

Arguments

x	phreg object
...	Additional arguments to baseplot funtion

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))
plot(out1)
plot(out1, stratas=c(0,3))
plot(out1, stratas=c(0,3), col=2:3, lty=1:2, se=TRUE)
plot(out1, stratas=c(0), col=2, lty=2, se=TRUE, polygon=FALSE)
plot(out1, stratas=c(0), col=matrix(c(2,1,3),1,3), lty=matrix(c(1,2,3),1,3), se=TRUE, polygon=FALSE)

```

plot_twin

*Scatter plot function***Description**

Scatterplot with contours of the (kernel) estimated density

Usage

```

plot_twin(
  data,
  marginal.args = list(),
  kernsmooth.args = list(),
  xlab,
  ylab,
  col = "black",
  col2 = "lightblue",
  alpha = 0.3,
  grid = TRUE,
  side.plot = TRUE,
  ...
)

```

Arguments

data	bivariate data to plot (data.frame or matrix with 2 columns)
marginal.args	arguments to marginal estimator ('density' continuous data, 'barplot' for categorical)
kernsmooth.args	arguments to 2d-kernel smoother
xlab	x-axis label
ylab	y-axis label
col	color of points
col2	color of contour / density plot

<code>alpha</code>	transparency level of points
<code>grid</code>	should grid be added to the plot
<code>side.plot</code>	If TRUE subplots of the marginal distributions are added to the plot
<code>...</code>	arguments to lower level plot functions

Author(s)

Klaus Kähler Holst

Examples

```
data("twinbmi", package="mets")
twinwide <- fast.reshape(twinbmi, id="tvparnr",varying=c("bmi"))
datamz <- log(subset(twinwide, zyg=="MZ")[,c("bmi1","bmi2")])

# continuous variables
plot_twin(datamz)

# categorical variables
datamz2 <- datamz
datamz2[, 1] <- cut(datamz[, 1], 4)
datamz2[, 2] <- cut(datamz[, 2], 4)
plot_twin(datamz2, color = TRUE)

# survival variables
cens1 <- rbinom(nrow(datamz), 1, 0.5)
cens2 <- rbinom(nrow(datamz), 1, 0.5)
datamz2[, 1] <- Event(datamz[, 1], cens1)
datamz2[, 2] <- suppressWarnings(Event(datamz[, 2], cens2))
plot_twin(datamz2)

rm(datamz, datamz2, cens1, cens2)
```

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

```
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(2)+1
pmvn(lower=lower,upper=upper,mu=mu,sigma=sigma)
```

predict.phreg	<i>Predictions from proportional hazards model</i>
---------------	--

Description

Predictions from proportional hazards model

Usage

```
## S3 method for class 'phreg'
predict(
  object,
  newdata,
  times = NULL,
  individual.time = FALSE,
  tminus = FALSE,
  se = TRUE,
  robust = FALSE,
  conf.type = "log",
  conf.int = 0.95,
  km = FALSE,
  ...
)
```

Arguments

object	phreg object
newdata	data.frame
times	Time where to predict variable, default is all time-points from the object sorted

<code>individual.time</code>	to use one (individual) time per subject, and then newdata and times have same length and makes only predictions for these individual times.
<code>tminus</code>	to make predictions in T- that is strictly before given times, useful for IPCW techniques
<code>se</code>	with standard errors and upper and lower confidence intervals.
<code>robust</code>	to get robust se's also default for most functions (uses <code>robse.cumhaz</code> otherwise <code>se.cumhaz</code>).
<code>conf.type</code>	transformation for survival estimates, default is log
<code>conf.int</code>	significance level
<code>km</code>	to use Kaplan-Meier product-limit for baseline
	$S_{s0}(t) = (1 - dA_{s0}(t))$
	, otherwise take exp of cumulative baseline.
<code>...</code>	Additional arguments to plot functions

predictRisk

Risk predictions to work with riskRegression package

Description

Risk predictions to work with riskRegression package

Usage

```
predictRisk(object, ...)
```

Arguments

<code>object</code>	phreg/binreg/cifreg object
<code>...</code>	additional arguments to lower level functions

Author(s)

Thomas Scheike

predictRisk.binreg	<i>Risk predictions to work with riskRegression package</i>
--------------------	---

Description

Risk predictions to work with riskRegression package

Usage

```
## S3 method for class 'binreg'  
predictRisk(object, newdata, cause, times = NULL, ...)
```

Arguments

object	phreg/binreg/cifreg object
newdata	data.frame on which to make new predictions
cause	cause (cif) to predict
times	times for predictions
...	additional arguments to lower level functions

print.casewise	<i>prints Concordance test</i>
----------------	--------------------------------

Description

prints Concordance test

Usage

```
## 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

```
prob.exceed.recurrent(
  formula,
  data,
  cause = 1,
  death.code = 2,
  cens.code = 0,
  exceed = NULL,
  marks = NULL,
  cifmets = TRUE,
  all.cifs = FALSE,
  ...
)
```

Arguments

formula	formula
data	data-frame
cause	of interest
death.code	for status
cens.code	censoring codes
exceed	values (if not given then all observed values)
marks	may be give for jump-times and then exceed values needs to be specified
cifmets	if true uses cif of mets package rather than prodlim
all.cifs	if true then returns list of all fitted objects in cif.exceed
...	Additional arguments to lower level funtions

Author(s)

Thomas Scheike

References

Scheike, Eriksson, Tribler (2019) The mean, variance and correlation for bivariate recurrent events with a terminal event, JRSS-C

Examples

```
data(hfactioncpx12)
dtable(hfactioncpx12,~status)

oo <- prob.exceed.recurrent(Event(entry,time,status)~cluster(id),
  hfactioncpx12,cause=1,death.code=2)
plot(oo)
```

prt	<i>Prostate data set</i>
-----	--------------------------

Description

Prostate data set

Source

Simulated data

random.cif	<i>Random effects model for competing risks data</i>
------------	--

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

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

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

Usage

```

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 = "nr",
  entry = NULL,
  trunkp = 1,
  ...
)

```

Arguments

<code>cif</code>	a model object from the <code>comp.risk</code> function with the marginal cumulative incidence of <code>cause2</code> , i.e., the event that is conditioned on, and whose odds the comparison is made with respect to
<code>data</code>	a <code>data.frame</code> with the variables.
<code>cause</code>	specifies the causes related to the death times, the value <code>cens.code</code> is the censoring value.
<code>cif2</code>	specifies model for <code>cause2</code> if different from <code>cause1</code> .
<code>cause1</code>	cause of first coordinate.
<code>cause2</code>	cause of second coordinate.
<code>cens.code</code>	specifies the code for the censoring if <code>NULL</code> then uses the one from the marginal <code>cif</code> model.
<code>cens.model</code>	specified which model to use for the ICPW, KM is Kaplan-Meier alternatively it may be "cox"
<code>Nit</code>	number of iterations for Newton-Raphson algorithm.
<code>detail</code>	if 0 no details are printed during iterations, if 1 details are given.
<code>clusters</code>	specifies the cluster structure.
<code>theta</code>	specifies starting values for the cross-odds-ratio parameters of the model.
<code>theta.des</code>	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" optimizer, alternatively, use the "nr" 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
d <- simnordic.random(5000, delayed=TRUE, cordz=0.5, cormz=2, lam0=0.3, country=TRUE)
times <- seq(50, 90, by=10)
add1 <- timereg::comp.risk(Event(time, cause) ~ -1 + factor(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 <- timereg::comp.risk(Event(time,cause)~-1+factor(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

```

rchaz

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

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

Usage

```

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

```

Arguments

cumhazard	cumulative hazard, or piece-constant rates for periods defined by first column of input.
rr	relative risk for simulations, alternatively when rr=1 specify n
n	number of simulation if rr not given
entry	delayed entry time for simulations.
cum.hazard	specifies wheter input is cumulative hazard or rates.
cause	name of cause
extend	to extend piecwise constant with constant rate. Default is average rate over time from cumulative (when TRUE), if numeric then uses given rate.

Details

For a piecewise linear cumulative hazard the inverse is easy to compute with and delayed entry x we compute

$$\Lambda^{-1}(\Lambda(x) + E/RR)$$

, where RR are the relative risks and E is exponential with mean 1. This quantity has survival function

$$P(T > t | T > x) = \exp(-RR(\Lambda(t) - \Lambda(x)))$$

.

Author(s)

Thomas Scheike

Examples

```
chaz <- c(0,1,1.5,2,2.1)
breaks <- c(0,10, 20, 30, 40)
cumhaz <- cbind(breaks,chaz)
n <- 10
X <- rbinom(n,1,0.5)
beta <- 0.2
rrcox <- exp(X * beta)

pctime <- rchaz(cumhaz,n=10)
pctimecox <- rchaz(cumhaz,rrcox,entry=runif(n))
```

rchazC

Piecewise constant hazard distribution

Description

Piecewise constant hazard distribution

Usage

```
rchazC(base1, rr, entry)
```

Arguments

base1	baseline
rr	relative risk terms
entry	entry times for left truncation

rcrisk	<i>Simulation of Piecewise constant hazard models with two causes (Cox).</i>
--------	--

Description

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

Usage

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

Arguments

cumhaz1	cumulative hazard of cause 1
cumhaz2	cumulative hazard of cause 1
rr1	number of simulations or vector of relative risk for simulations.
rr2	number of simulations or vector of relative risk for simulations.
n	number of simulation if rr not given
cens	to censor further , rate or cumumulative hazard
rrc	retlativ risk for censoring.
extend	to extend the cumulative hazards to largest end-point
...	arguments for rchaz

Author(s)

Thomas Scheike

Examples

```
data(bmt);
cox1 <- phreg(Surv(time,cause==1)~tcell+platelet,data=bmt)
cox2 <- phreg(Surv(time,cause==2)~tcell+platelet,data=bmt)
```

```

X1 <- bmt[,c("tcell","platelet")]
n <- 100
xid <- sample(1:nrow(X1),n,replace=TRUE)
Z1 <- X1[xid,]
Z2 <- X1[xid,]
rr1 <- exp(as.matrix(Z1) %*% cox1$coef)
rr2 <- exp(as.matrix(Z2) %*% cox2$coef)

d <- rcrisk(cox1$cum,cox2$cum,rr1,rr2)
dd <- cbind(d,Z1)

scox1 <- phreg(Surv(time,status==1)~tcell+platelet,data=dd)
scox2 <- phreg(Surv(time,status==2)~tcell+platelet,data=dd)
par(mfrow=c(1,2))
plot(cox1); plot(scox1,add=TRUE)
plot(cox2); plot(scox2,add=TRUE)
cbind(cox1$coef,scox1$coef,cox2$coef,scox2$coef)

```

recreg

*Recurrent events regression with terminal event***Description**

Fits Ghosh-Lin IPCW Cox-type model

Usage

```

recreg(
  formula,
  data,
  cause = 1,
  death.code = 2,
  cens.code = 0,
  cens.model = ~1,
  weights = NULL,
  offset = NULL,
  Gc = NULL,
  wcomp = NULL,
  marks = NULL,
  augmentation.type = c("lindyn.augment", "lin.augment"),
  ...
)

```

Arguments

formula	formula with 'Event' outcome
data	data frame

cause	of interest (1 default)
death.code	codes for death (terminating event, 2 default)
cens.code	code of censoring (0 default)
cens.model	for stratified Cox model without covariates
weights	weights for score equations
offset	offsets for model
Gc	censoring weights for time argument, default is to calculate these with a Kaplan-Meier estimator, should then give $G_c(T_i)$
wcomp	weights for composite outcome, so when $\text{cause}=c(1,3)$, we might have $\text{wcomp}=c(1,2)$.
marks	a mark value can be specified, this is vector from the data-frame where the mark value can be found at all events
augmentation.type	of augmentation when augmentation model is given
...	Additional arguments to lower level funtions

Details

For Cox type model :

$$E(dN_1(t)|X) = \mu_0(t)dt \exp(X^T \beta)$$

by solving Cox-type IPCW weighted score equations

$$\int (Z - E(t))w(t)dN_1(t)$$

where

$$w(t) = G(t)(I(T_i \wedge t < C_i)/G_c(T_i \wedge t))$$

and

$$E(t) = S_1(t)/S_0(t)$$

and

$$S_j(t) = \sum X_i^j w_i(t) \exp(X_i^T \beta)$$

.

The iid decomposition of the beta's are on the form

$$\int (Z - E)w(t)dM_1 + \int q(s)/p(s)dM_c$$

and returned as iid.

Events, deaths and censorings are specified via stop start structure and the Event call, that via a status vector and cause (code), censoring-codes (cens.code) and death-codes (death.code) indentifies these. See example and vignette.

Author(s)

Thomas Scheike

Examples

```
## data with no ties
library(mets)
data(hfactioncpx12)
hf <- hfactioncpx12
hf$x <- as.numeric(hf$treatment)
dd <- data.frame(treatment=levels(hf$treatment),id=1)

gl <- recreg(Event(entry,time,status)~treatment+cluster(id),data=hf,cause=1,death.code=2)
summary(gl)
pgl <- predict(gl,dd,se=1); plot(pgl,se=1)

## censoring stratified after treatment
gls <- recreg(Event(entry,time,status)~treatment+cluster(id),data=hf,
cause=1,death.code=2,cens.model=~strata(treatment))
summary(gls)

## IPCW at 2 years
l12 <- recregIPCW(Event(entry,time,status)~treatment+cluster(id),data=hf,
cause=1,death.code=2,time=2,cens.model=~strata(treatment))
summary(l12)
```

recurrentMarginal

Fast recurrent marginal mean when death is possible

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

```
recurrentMarginal(formula, data, cause = 1, ..., death.code = 2)
```

Arguments

formula	with Event object
data	data frame for computation
cause	of interest (1 default)
...	Additional arguments to lower level funtions
death.code	codes for death (terminating event, 2 default)

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

Cook, R. J. and Lawless, J. F. (1997) Marginal analysis of recurrent events and a terminating event. *Statist. Med.*, 16, 911–924. Ghosh and Lin (2002) Nonparametric Analysis of Recurrent events and death, *Biometrics*, 554–562.

Examples

```
library(mets)
data(hfactioncpx12)
hf <- hfactioncpx12
hf$x <- as.numeric(hf$treatment)

## to fit non-parametric models with just a baseline
xr <- phreg(Surv(entry,time,status==1)~cluster(id),data=hf)
dr <- phreg(Surv(entry,time,status==2)~cluster(id),data=hf)
par(mfrow=c(1,3))
plot(dr,se=TRUE)
title(main="death")
plot(xr,se=TRUE)
### robust standard errors
rxr <- robust.phreg(xr,fixbeta=1)
plot(rxr,se=TRUE,robust=TRUE,add=TRUE,col=4)

## marginal mean of expected number of recurrent events
## out <- recurrentMarginalPhreg(xr,dr)
## summary(out,times=1:5)

## marginal mean using formula
outN <- recurrentMarginal(Event(entry,time,status)~cluster(id),hf,cause=1,death.code=2)
```

```

plot(outN,se=TRUE,col=2,add=TRUE)
summary(outN,times=1:5)

#####
###   with strata   #####
#####
out <- recurrentMarginal(Event(entry,time,status)~strata(treatment)+cluster(id),
                          data=hf,cause=1,death.code=2)
plot(out,se=TRUE,ylab="marginal mean",col=1:2)

summary(out,times=1:5)

```

resmean.phreg	<i>Restricted mean for stratified Kaplan-Meier or Cox model with martingale standard errors</i>
---------------	---

Description

Restricted mean for stratified Kaplan-Meier or stratified Cox with martingale standard error. Standard error is computed using linear interpolation between standard errors at jump-times. Plots gives restricted mean at all times. Years lost can be computed based on this and decomposed into years lost for different causes using the `cif.yearslost` function that is based on integrating the cumulative incidence functions. One particular feature of these functions are that the restricted mean and years-lost are computed for all event times as functions and can be plotted/viewed. When times are given and beyond the last event time withn a strata the curves are extrapolated using the estimates of cumulative incidence.

Usage

```
resmean.phreg(x, times = NULL, covs = NULL, ...)
```

Arguments

<code>x</code>	phreg object
<code>times</code>	possible times for which to report restricted mean
<code>covs</code>	possible covariate for Cox model
<code>...</code>	Additional arguments to lower level funtions

Author(s)

Thomas Scheike

Examples

```
library(mets)
data(bmt); bmt$time <- bmt$time+runif(408)*0.001
out1 <- phreg(Surv(time,cause!=0)~strata(tcell,platelet),data=bmt)

rm1 <- resmean.phreg(out1,times=10*(1:6))
summary(rm1)
par(mfrow=c(1,2))
plot(rm1,se=1)
plot(rm1,years.lost=TRUE,se=1)

## comparing populations, can also be done using rmstIPCW via influence functions
rm1 <- resmean.phreg(out1,times=40)
e1 <- estimate(rm1)
e1
estimate(e1,rbind(c(1,-1,0,0)))

## years.lost decomposed into causes
drm1 <- cif.yearslost(Event(time,cause)~strata(tcell,platelet),data=bmt,times=10*(1:6))
par(mfrow=c(1,2)); plot(drm1,cause=1,se=1); plot(drm1,cause=2,se=1);
summary(drm1)

## comparing populations, can also be done using rmstIPCW via influence functions
drm1 <- cif.yearslost(Event(time,cause)~strata(tcell,platelet),data=bmt,times=40)
summary(drm1)
## first cause
e1 <- estimate(drm1)
estimate(e1,rbind(c(1,-1,0,0)))
```

resmeanATE	<i>Average Treatment effect for Restricted Mean for censored competing risks data using IPCW</i>
------------	--

Description

Under the standard causal assumptions we can estimate the average treatment effect $E(Y(1) - Y(0))$. We need Consistency, ignorability ($Y(1), Y(0)$ indep A given X), and positivity.

Usage

```
resmeanATE(formula, data, model = "exp", outcome = c("rmst", "rmtl"), ...)
```

Arguments

formula	formula with 'Event' outcome
data	data-frame
model	exp ("exp") or identity link ("lin")
outcome	restricted mean time (rmst) or restricted mean time lost (rmtl)
...	Additional arguments to pass to binregATE

Details

The first covariate in the specification of the competing risks regression model must be the treatment effect that is a factor. If the factor has more than two levels then it uses the mlogit for propensity score modelling. We consider the outcome $\min(T; \tau)$ or $I(\epsilon = \text{cause}) (t - \min(T; t))$ that gives years lost due to cause "cause" depending on the number of causes. The default model is the $\exp(X^T \beta)$ and otherwise a linear model is used.

Estimates the ATE using the the standard binary double robust estimating equations that are IPCW censoring adjusted.

Author(s)

Thomas Scheike

Examples

```
library(mets); data(bmt); bmt$event <- bmt$cause!=0; dfactor(bmt) <- tcell~tcell
out <- resmeanATE(Event(time,event)~tcell+platelet,data=bmt,time=40,treat.model=tcell~platelet)
summary(out)

out1 <- resmeanATE(Event(time,cause)~tcell+platelet,data=bmt,cause=1,time=40,
                    treat.model=tcell~platelet)
summary(out1)

ratioATE(out,out1,h=function(x) log(x))
```

resmeanIPCW

Restricted IPCW mean for censored survival data

Description

Simple and fast version for IPCW regression for just one time-point thus fitting the model

$$E(\min(T, t) | X) = \exp(X^T \beta)$$

or in the case of competing risks data

$$E(I(\epsilon = 1)(t - \min(T, t)) | X) = \exp(X^T \beta)$$

thus given years lost to cause, see binreg for the arguments.

Usage

```
resmeanIPCW(formula, data, outcome = c("rmst", "rmtl"), ...)
```

Arguments

formula	formula with outcome on Event form
data	data frame
outcome	can do either rmst regression ('rmst') or years-lost regression ('rmtl')
...	Additional arguments to binreg

Details

When the status is binary assumes it is a survival setting and default is to consider outcome $Y = \min(T, t)$, if status has more than two levels, then computes years lost due to the specified cause, thus using the response

$$Y = (t - \min(T, t))I(\text{status} = \text{cause})$$

Based on binomial regression IPCW response estimating equation:

$$X(\Delta(\min(T, t))Y/G_c(\min(T, t)) - \exp(X^T \text{beta})) = 0$$

for IPCW adjusted responses. Here

$$\Delta(\min(T, t)) = I(\min(T, t) \leq C)$$

is indicator of being uncensored. Concretely, the uncensored observations at time t will count those with an event (of any type) before t and those with a censoring time at t or further out. One should therefore be a bit careful when data has been constructed such that some of the event times T are equivalent to t .

Can also solve the binomial regression IPCW response estimating equation:

$$h(X)X(\Delta(\min(T, t))Y/G_c(\min(T, t)) - \exp(X^T \text{beta})) = 0$$

for IPCW adjusted responses where h is given as an argument together with iid of censoring with h .

By using appropriately the h argument we can also do the efficient IPCW estimator estimator.

Variance is based on

$$\sum w_i^2$$

also with IPCW adjustment, and `naive.var` is variance under known censoring model.

When `Ydirect` is given it solves :

$$X(\Delta(\min(T, t))Y_{\text{direct}}/G_c(\min(T, t)) - \exp(X^T \text{beta})) = 0$$

for IPCW adjusted responses.

The actual influence (type="II") function is based on augmenting with

$$X \int_0^t E(Y|T > s)/G_c(s) dM_c(s)$$

and alternatively just solved directly (type="I") without any additional terms.

Censoring model may depend on strata.

Author(s)

Thomas Scheike

Examples

```
library(mets)
data(bmt); bmt$time <- bmt$time+runif(nrow(bmt))*0.001
#  $E(\min(T;t) | X) = \exp(a+bX)$  with IPCW estimation
out <- resmeanIPCW(Event(time,cause!=0)~tcell+platelet+age,bmt,
                    time=50,cens.model=~strata(platelet),model="exp")
summary(out)

## weighted GLM version RMST
out2 <- logitIPCW(Event(time,cause!=0)~tcell+platelet+age,bmt,
                  time=50,cens.model=~strata(platelet),model="exp",outcome="rmst")
summary(out2)

### time-lost
outtl <- resmeanIPCW(Event(time,cause!=0)~tcell+platelet+age,bmt,
                     time=50,cens.model=~strata(platelet),model="exp",outcome="rmtl")
summary(outtl)

### same as Kaplan-Meier for full censoring model
bmt$int <- with(bmt,strata(tcell,platelet))
out <- resmeanIPCW(Event(time,cause!=0)~-1+int,bmt,time=30,
                  cens.model=~strata(platelet,tcell),model="lin")
estimate(out)
out1 <- phreg(Surv(time,cause!=0)~strata(tcell,platelet),data=bmt)
rm1 <- resmean.phreg(out1,times=30)
summary(rm1)

### years lost regression
outl <- resmeanIPCW(Event(time,cause!=0)~-1+int,bmt,time=30,outcome="years-lost",
                  cens.model=~strata(platelet,tcell),model="lin")
estimate(outl)

## competing risks years-lost for cause 1
out <- resmeanIPCW(Event(time,cause)~-1+int,bmt,time=30,cause=1,
                  cens.model=~strata(platelet,tcell),model="lin")
estimate(out)
## same as integrated cumulative incidence
rmc1 <- cif.yearslost(Event(time,cause)~strata(tcell,platelet),data=bmt,times=30)
summary(rmc1)
```

rpch

Piecewise constant hazard distribution

Description

Piecewise constant hazard distribution

Usage

```
rpch(n, lambda = 1, breaks = c(0, Inf))
```

Arguments

n	sample size
lambda	rate parameters
breaks	time cut-points

sim.cause.cox	<i>Simulation of cause specific from Cox models.</i>
---------------	--

Description

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

Usage

```
sim.cause.cox(coxs,n,data=NULL,cens=NULL,rrc=NULL,...)
```

Arguments

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

Author(s)

Thomas Scheike

Examples

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

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

```
par(mfrow=c(1,2))
plot(cox1); plot(scox1,add=TRUE);
plot(cox2); plot(scox2,add=TRUE);
```

sim.cif

Simulation of output from Cumulative incidence regression model

Description

Simulates data that looks like fit from fitted cumulative incidence model

Usage

```
sim.cif(cif,n,data=NULL,Z=NULL,drawZ=TRUE,cens=NULL,rrc=NULL,cumstart=c(0,0),...)
```

Arguments

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

Author(s)

Thomas Scheike

Examples

```
data(bmt)

scif <- cifreg(Event(time,cause)~tcell+platelet+age,data=bmt,cause=1,prop=NULL)
summary(scif)
plot(scif)
#####
# simulating several causes with specific cumulatives
#####
```

```

cif1 <- cifreg(Event(time,cause)~tcell+age,data=bmt,cause=1,prop=NULL)
cif2 <- cifreg(Event(time,cause)~tcell+age,data=bmt,cause=2,prop=NULL)
# dd <- sim.cifsRestrict(list(cif1,cif2),200,data=bmt)
dd <- sim.cifs(list(cif1,cif2),200,data=bmt)
scif1 <- cifreg(Event(time,cause)~tcell+age,data=dd,cause=1)
scif2 <- cifreg(Event(time,cause)~tcell+age,data=dd,cause=2)

par(mfrow=c(1,2))
plot(cif1); plot(scif1,add=TRUE,col=2)
plot(cif2); plot(scif2,add=TRUE,col=2)

```

sim.cox

*Simulation of output from Cox model.***Description**

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

Usage

```
sim.cox(cox,n,data=NULL,cens=NULL,rrc=NULL,entry=NULL,rr=NULL,Z=NULL,extend=TRUE,...)
```

Arguments

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

Author(s)

Thomas Scheike

Examples

```

data(sTRACE)
nsim <- 100
coxs <- phreg(Surv(time,status==9)~strata(chf)+vf+wmi,data=sTRACE)
sim3 <- sim.phreg(coxs,nsim,data=sTRACE)
cc <- phreg(Surv(time, status)~strata(chf)+vf+wmi,data=sim3)
cbind(coxs$coef,cc$coef)
plot(coxs,col=1); plot(cc,add=TRUE,col=2)

```

sim.recurrent	<i>Simulation of two-stage recurrent events data based on Cox/Cox or Cox/Ghosh-Lin structure</i>
---------------	--

Description

Simulation of two-stage recurrent events data based on Cox/Cox or Cox/Ghosh-Lin structure

Usage

```

sim.recurrent(cox1,cox=NULL,coxc=NULL,n=1,data=NULL,
type=c("default","cox-cox","gl-cox"),id="id",
varz=1,share=1,cens=0.001,scale1=1,scaled=1,dependence=NULL,...)

```

Arguments

cox1	cox/ghosh-lin for recurrent events
cox2	cox for terminal event
coxc	possible cox for censrong
n	number of id's
data	on which the models are fitted (to draw covariates)
type	to specify type of simulation, if not default
id	name of id variable
varz	dependence frailty
share	to fit patly shared random effects model
cens	censoring rate for exponential censoring
scale1	to scale baseline of recurrent events model
scaled	to scale baseline of terminal event
dependence	if dependence different from NULL, then uses simRecurrentList based on models given
...	Additional arguments to simGLcox, nmin, nmax regulates linear approximation grid

Details

Must specify two phreg objects, or a phreg and a recreg object, then simulates data from two-stage model

Author(s)

Thomas Scheike

References

Scheike (2024), Twostage recurrent events models, under review.

Examples

```
data(hfactioncpx12)
hf <- hfactioncpx12
hf$x <- as.numeric(hf$treatment)
n <- 100
xr <- phreg(Surv(entry,time,status==1)~x+cluster(id),data=hf)
dr <- phreg(Surv(entry,time,status==2)~x+cluster(id),data=hf)
simcoxcox <- sim.recurrent(xr,dr,n=n,data=hf)
recGL <- recreg(Event(entry,time,status)~x+cluster(id),hf,death.code=2)
simglcox <- sim.recurrent(recGL,dr,n=n,data=hf)
```

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 paramter (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

simClaytonOakes	<i>Simulate from the Clayton-Oakes frailty model</i>
-----------------	--

Description

Simulate observations from the Clayton-Oakes copula model with piecewise constant marginals.

Usage

```
simClaytonOakes(  
  K,  
  n,  
  eta,  
  beta,  
  stoptime,  
  lam = 1,  
  left = 0,  
  pairleft = 0,  
  trunc.prob = 0.5,  
  same = 0  
)
```

Arguments

K	Number of clusters
n	Number of observations in each cluster
eta	variance
beta	Effect (log hazard ratio) of covariate
stoptime	Stopping time

lam	constant hazard
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

simClaytonOakesWei	<i>Simulate from the Clayton-Oakes frailty model</i>
--------------------	--

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

simGLcox

Simulation of two-stage recurrent events data based on Cox/Cox or Cox/Ghosh-Lin structure

Description

Simulation of two-stage recurrent events data based on Cox/Cox or Cox/Ghosh-Lin structure. type=3 will generate Cox/Cox twostage mode, type=2 will generate Ghosh-Lin/Cox model. If the variance is var.z=0, then generates data without any dependence or frailty. If model="twostage" then default is to generate data from Ghosh-Lin/Cox model, and if type=3 then will generate data with marginal Cox models (Cox/Cox). Simulation based on linear aproximation of hazard for two-stage models based on grid on time-scale. Must be sufficiently fine.

Usage

```
simGLcox(
  n,
  base1,
  drcumhaz,
  var.z = 0,
  r1 = NULL,
  rd = NULL,
  rc = NULL,
  fz = NULL,
  fdz = NULL,
  model = c("twostage", "frailty", "shared"),
  type = NULL,
  share = 1,
  cens = NULL,
  nmin = 100,
  nmax = 1000
)
```

Arguments

n	number of id's
base1	baseline for cox/ghosh-lin models
drcumhaz	baseline for terminal event
var.z	variance of gamma frailty
r1	relative risk term for baseline
rd	relative risk term for terminal event
rc	relative risk term for censorings
fz	possible transformation (function) of frailty term
fdz	possible transformation (function) of frailty term for death

model	twostage, frailty, shared (partly shared two-stage model)
type	type of simulation, default is decided based on model
share	to fit patly shared random effects model
cens	censoring rate for exponential censoring
nmin	default 100, at least nmin or number of rows of the two-baselines max(nmin,nrow(base1),nrow(drcumhaz) points in time-grid from 0 to maximum time for base1
nmax	default 1000, at most nmax points in time-grid

Details

Must specify baselines of recurrent events and terminal event and possible covariate effects.

References

Scheike (2024), Twostage recurrent events models, under review.

simMultistate	<i>Simulation of illness-death model</i>
---------------	--

Description

Simulation of illness-death model

Usage

```
simMultistate(  
  n,  
  cumhaz,  
  cumhaz2,  
  death.cumhaz,  
  death.cumhaz2,  
  rr = NULL,  
  rr2 = NULL,  
  rd = NULL,  
  rd2 = NULL,  
  gap.time = FALSE,  
  max.recurrent = 100,  
  dependence = 0,  
  var.z = 0.22,  
  cor.mat = NULL,  
  cens = NULL,  
  ...  
)
```

Arguments

n	number of id's
cumhaz	cumulative hazard of going from state 1 to 2.
cumhaz2	cumulative hazard of going from state 2 to 1.
death.cumhaz	cumulative hazard of death from state 1.
death.cumhaz2	cumulative hazard of death from state 2.
rr	relative risk adjustment for cumhaz
rr2	relative risk adjustment for cumhaz2
rd	relative risk adjustment for death.cumhaz
rd2	relative risk adjustment for death.cumhaz2
gap.time	if true simulates gap-times with specified cumulative hazard
max.recurrent	limits number recurrent events to 100
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, $z1 = (z11 + z12)/2$ such that mean is 1, $z2 = (z11^{cor.mat(1,2)} + z13)/2$, $z3 = (z12^{cor.mat(2,3)} + z13^{cor.mat(1,3)})/2$, with $z11$ $z12$ $z13$ are gamma with mean and variance 1, first random effect is $z1$ and for N1 second random effect is $z2$ and for N2 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 funtions

Details

simMultistate with different death intensities from states 1 and 2

Must give cumulative hazards on some time-range

Author(s)

Thomas Scheike

Examples

```
#####
## getting some rates to mimick
#####
data(CPH_HPN_CRBSI)
dr <- CPH_HPN_CRBSI$terminal
base1 <- CPH_HPN_CRBSI$crbsi
base4 <- CPH_HPN_CRBSI$mechanical
dr2 <- scalecumhaz(dr,1.5)
cens <- rbind(c(0,0),c(2000,0.5),c(5110,3))

iddata <- simMultistate(1000,base1,base1,dr,dr2,cens=cens)
```

```

dlist(iddata,.~id|id<3,n=0)

### estimating rates from simulated data
c0 <- phreg(Surv(start,stop,status==0)~+1,iddata)
c3 <- phreg(Surv(start,stop,status==3)~+strata(from),iddata)
c1 <- phreg(Surv(start,stop,status==1)~+1,subset(iddata,from==2))
c2 <- phreg(Surv(start,stop,status==2)~+1,subset(iddata,from==1))
###
par(mfrow=c(2,3))
bplot(c0)
lines(cens,col=2)
bplot(c3,main="rates 1-> 3 , 2->3")
lines(dr,col=1,lwd=2)
lines(dr2,col=2,lwd=2)
###
bplot(c1,main="rate 1->2")
lines(base1,lwd=2)
###
bplot(c2,main="rate 2->1")
lines(base1,lwd=2)

```

simRecurrentII

*Simulation of recurrent events data based on cumulative hazards with
two types of recurrent events*

Description

Simulation of recurrent events data based on cumulative hazards

Usage

```

simRecurrentII(
  n,
  cumhaz,
  cumhaz2,
  death.cumhaz = NULL,
  r1 = NULL,
  r2 = NULL,
  rd = NULL,
  rc = NULL,
  dependence = 0,
  var.z = 1,
  cor.mat = NULL,
  cens = NULL,
  gap.time = FALSE,
  max.recurrent = 100,
  ...
)

```

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
r1	potential relative risk adjustment of rate
r2	potential relative risk adjustment of rate
rd	potential relative risk adjustment of rate
rc	potential relative risk adjustment of rate
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, $z1 = (z11 + z12)/2$ such that mean is 1, $z2 = (z11^{cor.mat(1,2)} + z13)/2$, $z3 = (z12^{cor.mat(2,3)} + z13^{cor.mat(1,3)})/2$, with $z11$ $z12$ $z13$ are gamma with mean and variance 1, first random effect is $z1$ and for $N1$ second random effect is $z2$ and for $N2$ 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
gap.time	if true simulates gap-times with specified cumulative hazard
max.recurrent	limits number recurrent events to 100
...	Additional arguments to lower level funtions

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

Author(s)

Thomas Scheike

Examples

```
#####
## getting some rates to mimick
#####
data(CPH_HPN_CRBSI)
dr <- CPH_HPN_CRBSI$terminal
base1 <- CPH_HPN_CRBSI$crbsi
base4 <- CPH_HPN_CRBSI$mechanical

#####
### simulating simple model that mimicks data
```

```
#####
rr <- simRecurrent(5,base1)
dlist(rr,~id,n=0)
rr <- simRecurrent(5,base1,death.cumhaz=dr)
dlist(rr,~id,n=0)

rr <- simRecurrent(100,base1,death.cumhaz=dr)
par(mfrow=c(1,3))
showfitsim(causes=1,rr,dr,base1,base1)
#####
### simulating simple model
### random effect for all causes (Z shared for death and recurrent)
#####
rr <- simRecurrent(100,base1,death.cumhaz=dr,dependence=1,var.gamma=0.4)
dtable(rr,~death+status)

#####
### simulating simple model that mimicks data
### now with two event types and second type has same rate as death rate
#####
set.seed(100)
rr <- simRecurrentII(100,base1,base4,death.cumhaz=dr)
dtable(rr,~death+status)
par(mfrow=c(2,2))
showfitsim(causes=2,rr,dr,base1,base4)

## general case, 3 recurrent events and 2 causes of death
set.seed(100)
cumhaz <- list(base1,base1,base4)
drl <- list(dr,base4)
rr <- simRecurrentList(100,cumhaz,death.cumhaz=drl,dependence=0)
dtable(rr,~death+status)
showfitsimList(rr,cumhaz,drl)
```

simRecurrentTS	<i>Simulation of recurrent events data based on cumulative hazards: Two-stage model</i>
----------------	---

Description

Simulation of recurrent events data based on cumulative hazards

Usage

```
simRecurrentTS(
  n,
  cumhaz,
  cumhaz2,
  death.cumhaz = NULL,
```

```

    nu = rep(1, 3),
    share1 = 0.3,
    vargamD = 2,
    vargam12 = 0.5,
    gap.time = FALSE,
    max.recurrent = 100,
    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
nu	powers of random effects where $\nu > -1/\text{shape}$
share1	how random effect for death splits into two parts
vargamD	variance of random effect for death
vargam12	shared random effect for N1 and N2
gap.time	if true simulates gap-times with specified cumulative hazard
max.recurrent	limits number recurrent events to 100
cens	rate of censoring exponential distribution
...	Additional arguments to lower level funtions

Details

Model is constructed such that marginals are on specified form by linear approximations of cumulative hazards that are on a specific form to make them equivalent to marginals after integrating out over survivors. Therefore $E(dN_1 | D > t) = \text{cumhaz}$, $E(dN_2 | D > t) = \text{cumhaz2}$, and hazard of death is death.cumhazard

Must give hazard of death and two recurrent events. Hazard of death is death.cumhazard two event types and their dependence can be specified but the two recurrent events need to share random effect.

Random effect for death $Z.\text{death} = (Zd1 + Zd2)$, $Z1 = (Zd1^{\nu_1}) Z12$, $Z2 = (Zd2^{\nu_2}) Z12^{\nu_3}$

$$Z.\text{death} = Zd1 + Zd2$$

gamma distributions

$$Zdj$$

gamma distribution with mean parameters (sharej), vargamD, share2=1-share1

$$Z12$$

gamma distribution with mean 1 and variance vargam12

Author(s)

Thomas Scheike

Examples

```
#####
## getting some rates to mimick
#####
data(CPH_HPN_CRBSI)
dr <- CPH_HPN_CRBSI$terminal
base1 <- CPH_HPN_CRBSI$crbsi
base4 <- CPH_HPN_CRBSI$mechanical

rr <- simRecurrentTS(1000,base1,base4,death.cumhaz=dr)
dtable(rr,~death+status)
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.

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 <- timereg::comp.risk(Event(time,cause)~+1+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,se=0,uniform=0)
## summary(out1,marg.cif=pad)
```

summaryGLM

Reporting OR (exp(coef)) from glm with binomial link and glm predictions

Description

Reporting OR from glm with binomial link and glm predictions

Usage

```
summaryGLM(object, id = NULL, fun = NULL, ...)
```

Arguments

object	glm output
id	possible id for cluster corrected standard errors
fun	possible function for non-standard predictions based on object
...	arguments of estimate of lava for example level=0.95

Author(s)

Thomas Scheike

Examples

```
data(sTRACE)
sTRACE$id <- sample(1:100,nrow(sTRACE),replace=TRUE)

model <- glm(I(status==9)~sex+factor(diabetes)+age,data=sTRACE,family=binomial)
summaryGLM(model)
summaryGLM(model,id=sTRACE$id)

nd <- data.frame(sex=c(0,1),age=67,diabetes=1)
predictGLM(model,nd)
```

survival.twostage

Twostage survival model for multivariate survival data

Description

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

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

If clusters contain more than two subjects, we use a composite likelihood based on the pairwise bivariate models, for full MLE see `twostageMLE`.

The two-stage model is constructed such that given the gamma distributed random effects it is assumed that the survival functions are independent, and that the marginal survival functions are on Cox form (or additive form)

$$P(T > t|x) = S(t|x) = \exp(-\exp(x^T \beta) A_0(t))$$

One possibility is to model the variance within clusters via a regression design, and then one can specify a regression structure for the independent gamma distributed random effect for each cluster, such that the variance is given by

$$\theta = h(z_j^T \alpha)$$

where z is specified by `theta.des`, and a possible link function `var.link=1` will use the exponential link $h(x) = \exp(x)$, and `var.link=0` the identity link $h(x) = x$. The reported standard errors are

based on the estimated information from the likelihood assuming that the marginals are known (unlike the twostageMLE and for the additive gamma model below).

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

$$v_1^T(Z_1, \dots, Z_d)$$

, for d random effects. Each random effect has an associated parameter $(\lambda_1, \dots, \lambda_d)$. 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 $v_1^T(Z_1, \dots, Z_d)$ has mean 1 and variance $1/(v_1^T \lambda)$. It is here assumed that $lamtot = v_1^T \lambda$ is fixed within clusters as it would be for the ACE model below.

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 (var.par=1) 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 λ_j with the argument var.par=0.

For both types of models the basic model assumptions are that given the random effects of the clusters the survival distributions within a cluster are independent and ' on the form

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

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

The parameters $(\lambda_1, \dots, \lambda_d)$ are related to the parameters of the model by a regression construction *pard* ($d \times k$), that links the d λ parameters with the (k) underlying θ parameters

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

here using theta.des to specify these low-dimension association. Default is a diagonal matrix. This can be used to make structural assumptions about the variances of the random-effects as is needed for the ACE model for example.

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.twostage(
  margsurv,
  data = parent.frame(),
  method = "nr",
  detail = 0,
  clusters = NULL,
  silent = 1,
  weights = NULL,
  theta = NULL,
```

```

    theta.des = NULL,
    var.link = 1,
    baseline.iid = 1,
    model = "clayton.oakes",
    marginal.trunc = NULL,
    marginal.survival = NULL,
    strata = NULL,
    se.clusters = NULL,
    numDeriv = 1,
    random.design = NULL,
    pairs = NULL,
    dim.theta = NULL,
    numDeriv.method = "simple",
    additive.gamma.sum = NULL,
    var.par = 1,
    no.opt = FALSE,
    ...
)

```

Arguments

margsurv	Marginal model
data	data frame
method	Scoring method "nr", for lava NR optimizer
detail	Detail
clusters	Cluster variable
silent	Debug information
weights	Weights
theta	Starting values for variance components
theta.des	design for dependence parameters, when pairs are given the indices of the theta-design for this pair, is given in pairs as column 5
var.link	Link function for variance: exp-link.
baseline.iid	to adjust for baseline estimation, using phreg function on same data.
model	model
marginal.trunc	marginal left truncation probabilities
marginal.survival	optional vector of marginal survival probabilities
strata	strata for fitting, see example
se.clusters	for clusters for se calculation with iid
numDeriv	to get numDeriv version of second derivative, otherwise uses sum of squared scores for each pair
random.design	random effect design for additive gamma model, when pairs are given the indices of the pairs random.design rows are given as columns 3:4

<code>pairs</code>	matrix with rows of indeces (two-columns) for the pairs considered in the pair-wise composite score, useful for case-control sampling when marginal is known.
<code>dim.theta</code>	dimension of the theta parameter for pairs situation.
<code>numDeriv.method</code>	uses simple to speed up things and second derivative not so important.
<code>additive.gamma.sum</code>	for <code>two.stage=0</code> , 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.
<code>var.par</code>	is 1 for the default parametrization with the variances of the random effects, <code>var.par=0</code> specifies that the λ_j 's are used as parameters.
<code>no.opt</code>	for not optimizng
<code>...</code>	Additional arguments to maximizer NR of lava. and ascertained sampling

Author(s)

Thomas Scheike

References

Twostage estimation of additive gamma frailty models for survival data. Scheike (2019), work in progress

Shih and Louis (1995) Inference on the association parameter in copula models for bivariate survival data, *Biometrics*, (1995).

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

Measuring early or late dependence for bivariate twin data Scheike, Holst, Hjelmberg (2015), *LIDA*

Estimating heritability for cause specific mortality based on twins studies Scheike, Holst, Hjelmberg (2014), *LIDA*

Additive Gamma frailty models for competing risks data, *Biometrics* (2015) Eriksson and Scheike (2015),

Examples

```
data(diabetes)

# Marginal Cox model with treat as covariate
margph <- phreg(Surv(time,status)~treat+cluster(id),data=diabetes)
### Clayton-Oakes, MLE
fitco1<-twostageMLE(margph,data=diabetes,theta=1.0)
summary(fitco1)

### Plackett model
mph <- phreg(Surv(time,status)~treat+cluster(id),data=diabetes)
fitp <- survival.twostage(mph,data=diabetes,theta=3.0,Nit=40,
                        clusters=diabetes$id,var.link=1,model="plackett")
summary(fitp)
```

```

### Clayton-Oakes
fitco2 <- survival.twostage(mph,data=diabetes,theta=0.0,detail=0,
                           clusters=diabetes$id,var.link=1,model="clayton.oakes")
summary(fitco2)
fitco3 <- survival.twostage(margph,data=diabetes,theta=1.0,detail=0,
                           clusters=diabetes$id,var.link=0,model="clayton.oakes")
summary(fitco3)

### without covariates but with stratified
marg <- phreg(Surv(time,status)~+strata(treat)+cluster(id),data=diabetes)
fitpa <- survival.twostage(marg,data=diabetes,theta=1.0,
                           clusters=diabetes$id,model="clayton.oakes")
summary(fitpa)

fitcoa <- survival.twostage(marg,data=diabetes,theta=1.0,clusters=diabetes$id,
                           model="clayton.oakes")
summary(fitcoa)

### Piecewise constant cross hazards ratio modelling
#####

d <- subset(simClaytonOakes(2000,2,0.5,0,stoptime=2,left=0),!truncated)
udp <- piecewise.twostage(c(0,0.5,2),data=d,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
#####
## 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$idstrata <- ud$id+(as.numeric(ud$strata)-1)*2000

marg2 <- timereg::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$cluster,clusters=ud$idstrata,
                           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
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$idstrata,se.cluster=ud$cluster,
                          model="clayton.oakes",theta.des=tdes2,step=0.5)
summary(fitp3)

### same model using strata option, a bit slower
fitp4 <- survival.twostage(marg2,data=ud,clusters=ud$cluster,se.cluster=ud$cluster,
                          model="clayton.oakes",theta.des=tdes2,step=0.5,strata=ud$strata)
summary(fitp4)

## Reduce Ex.Timings
### structured random effects model additive gamma ACE
### simulate structured two-stage additive gamma ACE model
data <- simClaytonOakes.twin.ace(4000,2,1,0,3)
out <- twin.polygen.design(data,id="cluster")
pardes <- out$pardes
pardes
des.rv <- out$des.rv
head(des.rv)
aa <- phreg(Surv(time,status)~x+cluster(cluster),data=data,robust=0)
ts <- survival.twostage(aa,data=data,clusters=data$cluster,detail=0,
                      theta=c(2,1),var.link=0,step=0.5,
                      random.design=des.rv,theta.des=pardes)
summary(ts)

```

survivalG

G-estimator for Cox and Fine-Gray model

Description

Computes G-estimator

$$\hat{S}(t, A = a) = n^{-1} \sum_i \hat{S}(t, A = a, Z_i)$$

for the Cox model based on phreg og the Fine-Gray model based on the cifreg function. Gives influence functions of these risk estimates and SE's are based on these. If first covariate is a factor then all contrast are computed, and if continuous then considered covariate values are given by Avalues.

Usage

```

survivalG(
  x,
  data,

```

```

    time = NULL,
    Avalues = c(0, 1),
    varname = NULL,
    same.data = TRUE,
    id = NULL,
    subdata = NULL
  )

```

Arguments

x	phreg or cifreg object
data	data frame for risk averaging
time	for estimate
Avalues	values to compare for first covariate A
varname	if given then averages for this variable, default is first variable
same.data	assumes that same data is used for fitting of survival model and averaging.
id	might be given to link to data to iid decomposition of survival data, must be coded as 1,2,...,
subdata	rows or TRUE/FALSE to select which part of the data that is used for the G-computation. Might be treated

Author(s)

Thomas Scheike

Examples

```

data(bmt); bmt$time <- bmt$time+runif(408)*0.001
bmt$event <- (bmt$cause!=0)*1
dfactor(bmt) <- tcell.f~tcell

fg1 <- cifreg(Event(time,cause)~tcell.f+platelet+age,bmt,cause=1,
              cox.prep=TRUE,propodds=NULL)
summary(survivalG(fg1,bmt,50))

ss <- phreg(Surv(time,event)~tcell.f+platelet+age,bmt)
summary(survivalG(ss,bmt,50))

ss <- phreg(Surv(time,event)~strata(tcell.f)+platelet+age,bmt)
summary(survivalG(ss,bmt,50))

sst <- survivalGtime(ss,bmt,n=50)
plot(sst)

fg1t <- survivalGtime(fg1,bmt,n=50)
plot(fg1t)

```

test.conc	<i>Concordance test Compares two concordance estimates</i>
-----------	--

Description

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

Usage

```
test.conc(conc1, conc2, same.cluster = FALSE)
```

Arguments

conc1	Concordance estimate of group 1
conc2	Concordance estimate of group 2
same.cluster	if FALSE then groups are independent, otherwise estimates are based on same data.

Author(s)

Thomas Scheike

tetrachoric	<i>Estimate parameters from odds-ratio</i>
-------------	--

Description

Calculate tetrachoric correlation of probabilities from odds-ratio

Usage

```
tetrachoric(P, OR, approx = 0, ...)
```

Arguments

P	Joint probabilities or marginals (if OR is given)
OR	Odds-ratio
approx	If TRUE an approximation of the tetrachoric correlation is used
...	Additional arguments

Examples

```
tetrachoric(0.3,1.25) # Marginal p1=p2=0.3, OR=2
P <- matrix(c(0.1,0.2,0.2,0.5),2)
prod(diag(P))/prod(lava::revdiag(P))
##mets:::assoc(P)
tetrachoric(P)
or2prob(2,0.1)
or2prob(2,c(0.1,0.2))
```

TRACE

The TRACE study group of myocardial infarction

Description

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

Format

This data frame contains the following columns:

id a numeric vector. Patient code.

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

time a numeric vector. Survival time in years.

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

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

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

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

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

age a numeric vector code. Age of patient.

Details

sTRACE is a subsample consisting of 300 patients.

tTRACE is a subsample consisting of 1000 patients.

Source

The TRACE study group.

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

Examples

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

ttpd	<i>ttpd discrete survival data on interval form</i>
------	---

Description

ttpd discrete survival data on interval form

Source

Simulated data

twin.clustertrunc	<i>Estimation of twostage model with cluster truncation in bivariate situation</i>
-------------------	--

Description

Estimation of twostage model with cluster truncation in bivariate situation

Usage

```
twin.clustertrunc(
  survformula,
  data = parent.frame(),
  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

```
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$two            ## twostage output
par(mfrow=c(2,2))
plot(aout$marg) ## marginal model output

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

twinbmi	<i>BMI data set</i>
---------	---------------------

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: zygoty, 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 pr row. In addition a column with the zygosity (DZ or MZ given as a factor) of each individual much 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. If this argument is missing, the reference level (i.e. the first level) will be interpreted as the dizygotic 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. Can be a subset of "aced" (additive genetic factors, common environmental factors, unique environmental factors, dominant genetic factors). Other choices are: • "0" (or "sat"): Saturated model where twin 1 and twin 2 within each twin pair may have a different marginal distribution. • "1" (or "flex", "zyg"): Within twin pairs the marginal distribution is the same, but the marginal distribution may differ between MZ and DZ twins. A free correlation structure within MZ and DZ twins. • "2" (or "u", "eqmarg"): All individuals have the same marginals but a free correlation structure within MZ and DZ twins. The default value is an additive polygenic model type="ace".
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
constrain	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, b1=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
summary(ace2)
## Reduce Ex.Timings
## An interaction could be analyzed as:
ace3 <- twinlm(y ~ x1+x2 + x1:I(x2<0), data=d, DZ="DZ", zyg="zyg", id="id", type="ace")
ace3
## Categorical variables are also supported
d2 <- transform(d,x2cat=cut(x2,3,labels=c("Low","Med","High")))
ace4 <- twinlm(y ~ x1+x2cat, data=d2, DZ="DZ", zyg="zyg", id="id", type="ace")

```

twinsim

Simulate twin data

Description

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

Usage

```

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 wether 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](#)

twinstut	<i>Stutter data set</i>
----------	-------------------------

Description

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

Format

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

twostageMLE

*Twostage survival model fitted by pseudo MLE***Description**

Fits Clayton-Oakes clustered survival data using marginals that are on Cox form in the likelihood for the dependence parameter as in Glidden (2000). The dependence can be modelled via a

1. Regression design on dependence parameter.

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

$$\theta = h(z_j^T \alpha)$$

where z is specified by `theta.des`. The link function can be the exp when `var.link=1`

Usage

```
twostageMLE(
  margsurv,
  data = parent.frame(),
  theta = NULL,
  theta.des = NULL,
  var.link = 0,
  method = "NR",
  no.opt = FALSE,
  weights = NULL,
  se.cluster = NULL,
  ...
)
```

Arguments

<code>margsurv</code>	Marginal model from <code>phreg</code>
<code>data</code>	data frame
<code>theta</code>	Starting values for variance components
<code>theta.des</code>	design for dependence parameters, when pairs are given this is could be a (pairs) x (number of parameters) x (max number random effects) matrix
<code>var.link</code>	Link function for variance if 1 then uses exp link
<code>method</code>	type of optimizer, default is Newton-Raphson "NR"
<code>no.opt</code>	to not optimize, for example to get score and iid for specific theta
<code>weights</code>	cluster specific weights, but given with length equivalent to data-set, weights for score equations
<code>se.cluster</code>	specifies how the influence functions are summed before squared when computing the variance. Note that the id from the marginal model is used to construct MLE, and then these scores can be summed with the <code>se.cluster</code> argument.
<code>...</code>	arguments to be passed to optimizer

Author(s)

Thomas Scheike

References

Measuring early or late dependence for bivariate twin data Scheike, Holst, Hjelmberg (2015), LIDA
Twostage modelling of additive gamma frailty models for survival data. Scheike and Holst, working paper

Shih and Louis (1995) Inference on the association parameter in copula models for bivariate survival data, Biometrics, (1995).

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

Examples

```
data(diabetes)
dd <- phreg(Surv(time,status==1)~treat+cluster(id),diabetes)
oo <- twostageMLE(dd,data=diabetes)
summary(oo)

theta.des <- model.matrix(~-1+factor(adult),diabetes)

oo <-twostageMLE(dd,data=diabetes,theta.des=theta.des)
summary(oo)
```

WA_recurrent

While-Alive estimands for recurrent events

Description

Considers the ratio of means

$$E(N(\min(D, t)))/E(\min(D, t))$$

and the the mean of the events per time unit

$$E(N(\min(D, t))/\min(D, t))$$

both based on IPCW etimation. RMST estimator equivalent to Kaplan-Meier based estimator.

Usage

```
WA_recurrent(
  formula,
  data,
  time = NULL,
  cens.code = 0,
```

```

    cause = 1,
    death.code = 2,
    trans = NULL,
    cens.formula = NULL,
    augmentR = NULL,
    augmentC = NULL,
    type = NULL,
    marks = NULL,
    ...
)

```

Arguments

formula	Event formula first covariate on rhs must be a factor giving the treatment
data	data frame
time	for estimation
cens.code	of censorings
cause	of events
death.code	of terminal events
trans	possible power for mean of events per time-unit
cens.formula	censoring model, default is to use strata(treatment)
augmentR	covariates for model of mean ratio
augmentC	covariates for censoring augmentation
type	augmentation for call of binreg, when augmentC is given default is "I" and otherwise "II"
marks	possible marks for composite outcome situation for model for counts with marks
...	arguments for binregATE

Author(s)

Thomas Scheike

Index

- * **binomial**
 - binomial.twostage, 13
 - easy.binomial.twostage, 79
- * **data**
 - ACTG175, 7
 - calgb8923, 38
 - CPH_HPN_CRBSI, 54
 - dermalridges, 62
 - dermalridgesMZ, 62
 - haplo, 109
 - hfactioncpx12, 112
 - mena, 126
 - migr, 127
 - multcif, 129
 - np, 130
 - prt, 143
 - ttpd, 183
 - twinbmi, 184
 - twinstut, 188
- * **models**
 - blocksample, 33
 - mets.options, 127
 - twinlm, 185
 - twinsim, 187
- * **package**
 - bmt, 34
 - diabetes, 63
 - melanoma, 125
 - TRACE, 182
- * **regression**
 - binomial.twostage, 13
 - easy.binomial.twostage, 79
 - twinlm, 185
 - twinsim, 187
- * **survival**
 - cor.cif, 48
 - event.split, 88
 - EventSplit2, 90
 - Grandom.cif, 105
 - LinSpline, 122
 - random.cif, 143
 - rchaz, 146
 - rcrisk, 148
 - sim.cause.cox, 158
 - sim.cif, 159
 - sim.cox, 160
 - summary.cor, 172
 - survival.twostage, 174
 - twostageMLE, 189
- * **utilities**
 - blocksample, 33
 - npc, 130
 - [.Event (Event), 87
- aalenfrailty, 5
- aalenMets, 6
- ace.family.design (npc), 130
- ACTG175, 7
- addCums (rchaz), 146
- alpha2kendall (npc), 130
- alpha2spear (npc), 130
- as.character.Event (Event), 87
- as.matrix.Event (Event), 87
- ascertained.pairs (npc), 130
- back2timereg, 7
- basecumhaz (plot.phreg), 136
- basehazplot.phreg, 8
- basehazplot0.phreg (plot.phreg), 136
- baseplot (plot.phreg), 136
- bicomprisk, 9
- bicompriskData (bicomprisk), 9
- BinAugmentCifstrata, 11
- binomial.twostage, 13
- binreg, 18
- binregATE, 21
- binregCasewise, 23
- binregG, 24
- binregRatio, 25

- binregt (binreg), 18
- binregTSR, 28
- biprobit, 31
- blocksample, 33
- bmt, 34
- BootmediatorSurv (mediatorSurv), 123
- Bootphreg, 35
- bplot (plot.phreg), 136
- bplotdFG (doubleFGR), 66
- bptwin, 36, 186
- calgb8923, 38
- casewise, 39
- casewise.bin (casewise.test), 40
- casewise.test, 40
- cause.pchazard.sim (rcrisk), 148
- CCbinomial.twostage (npc), 130
- cif, 41
- cif.yearslost (resmean.phreg), 153
- cifreg, 42
- cifregFG (cifreg), 42
- ClaytonOakes, 44
- cluster.index, 46
- coarse.clust (npc), 130
- coefmat (npc), 130
- concordance.cor (concordanceCor), 47
- concordanceCor, 47
- concordanceTwinACE (npc), 130
- concordanceTwostage (npc), 130
- conftype (phreg), 130
- construct_id (phreg), 130
- cor.cif, 48
- corsim.prostate (npc), 130
- count.history, 53
- count.historyVar (count.history), 53
- countID (cluster.index), 46
- covfr (predict.phreg), 139
- covfridstrata (predict.phreg), 139
- covfridstrataCov (predict.phreg), 139
- covIntH1dM1IntH2dM2 (simRecurrentII), 168
- CPH_HPN_CRBSI, 54
- cpred (fast.approx), 93
- cumContr (gofZ.phreg), 104
- cumoddsreg
 - (interval.logitsurv.discrete), 113
- cumsum2strata (predict.phreg), 139
- cumsumidstratasum (predict.phreg), 139
- cumsumstrata (predict.phreg), 139
- cumsumstratasum (predict.phreg), 139
- daggr (daggregate), 54
- daggregate, 54
- Dbvn, 56
- dby, 56
- dby2 (dby), 56
- dby2<- (dby), 56
- dby<- (dby), 56
- dbyr (dby), 56
- dcor, 59
- dcount (dcor), 59
- dcut, 60
- dcut<- (dcut), 60
- ddrop (dcut), 60
- ddrop<- (dcut), 60
- dermalridges, 62
- dermalridgesMZ, 62
- deval (dcor), 59
- deval2 (dcor), 59
- dfactor (drelevel), 72
- dfactor<- (drelevel), 72
- dhead (dprint), 69
- diabetes, 63
- diffstrata (cifreg), 42
- dInterval
 - (interval.logitsurv.discrete), 113
- divide.conquer, 64
- divide.conquer.timereg, 64
- dkeep (dcut), 60
- dkeep<- (dcut), 60
- dlag, 65
- dlag<- (dlag), 65
- dlev (drelevel), 72
- dlev<- (drelevel), 72
- dlevel (drelevel), 72
- dlevel<- (drelevel), 72
- dlevels (drelevel), 72
- dlist (dprint), 69
- dmean (dcor), 59
- dmeansd (dcor), 59
- dmvn (pmvn), 138
- dnames (dcut), 60
- dnames<- (dcut), 60
- dnumeric (drelevel), 72

- dnnumeric<- (drelevel), 72
- doubleFGR, 66
- dprint, 68
- dquantile (dcor), 59
- dreg, 69
- drelev (drelevel), 72
- drelev<- (drelevel), 72
- drelevel, 72
- drelevel<- (drelevel), 72
- drename (dcut), 60
- drename<- (dcut), 60
- dreshape (fast.reshape), 95
- drm (dcut), 60
- drm<- (dcut), 60
- drop.specials (npc), 130
- drop.strata (FG_AugmentCifstrata), 97
- dsample (blocksample), 33
- dscalar (dcor), 59
- dsd (dcor), 59
- dsort, 74
- dsort2 (dsort), 74
- dsort<- (dsort), 74
- dspline, 75
- dspline<- (dspline), 75
- dstr (dcor), 59
- dsubset (dcor), 59
- dsum (dcor), 59
- dsummary (dcor), 59
- dtab (dtable), 76
- dtable, 76
- dtail (dprint), 69
- dtrans (dtransform), 78
- dtrans<- (dtransform), 78
- dtransform, 78
- dtransform<- (dtransform), 78
- dunique (dcut), 60

- easy.binomial.twostage, 79
- Effbinreg, 83
- EVaddGam, 85
- evalTerminal, 86
- Event, 87
- event.split, 88
- eventpois, 89
- EventSplit2, 90
- extendCums (simMultistate), 166

- familycluster.index, 91
- familyclusterWithProband.index, 92

- fast.approx, 93
- fast.cluster (npc), 130
- fast.pattern, 94
- fast.reshape, 95
- FastCoxPLstrataR (phregR), 131
- faster.reshape (npc), 130
- FG_AugmentCifstrata, 97
- FGprediid (cifreg), 42
- folds (npc), 130
- force.same.cens (npc), 130
- format.Event (Event), 87

- glm_IPTW, 99
- GLprediid (recreg), 149
- gof.phreg, 100
- gofFG (cifreg), 42
- gofG.phreg, 101
- gofM.phreg, 102
- gofZ.phreg, 104
- Grandom.cif, 105
- grouptable (npc), 130

- haplo, 109
- haplo.surv.discrete, 109
- headstrata (predict.phreg), 139
- hfactioncpx12, 112

- iidBaseline, 112
- IIDrecreg (recreg), 149
- ilap (npc), 130
- indexstrata (fast.approx), 93
- indexstratarightR (cifreg), 42
- Interval (interval.logitsurv.discrete), 113
- interval.logitsurv.discrete, 113
- invsubdist (sim.cif), 159
- ipw, 115
- ipw2, 116

- jumptimes (npc), 130

- kendall.ClaytonOakes.twin.ace (npc), 130
- kendall.normal.twin.ace (npc), 130
- km, 118
- kmplot (plot.phreg), 136
- kumarsim (binregATE), 21
- kumarsimRCT (binregATE), 21

- lifecourse, 119
- lifetable (lifetable.matrix), 121

- lifetable.matrix, 121
- lin.approx (rchaz), 146
- LinSpline, 122
- logitATE (binregATE), 21
- logitIPCW (binreg), 18
- logitIPCWATE (binregATE), 21
- logitSurv, 122
- loglikMVN (pmvn), 138
- make.pairwise.design (npc), 130
- matdoubleindex (predict.phreg), 139
- matplot.mets.twostage (npc), 130
- mdi (predict.phreg), 139
- mediatorSurv, 123
- medweight, 125
- melanoma, 125
- mena, 126
- mets.options, 127
- migr, 127
- mlogit, 128
- multcif, 129
- mystrata (cluster.index), 46
- mystrata2index (cluster.index), 46
- namesortme (phreg), 130
- nonparcuminc (npc), 130
- normalATE (binregATE), 21
- np, 130
- npc, 130
- object.defined (npc), 130
- or.cif (cor.cif), 48
- or2prob (tetrachoric), 181
- p11.binomial.twostage.RV (npc), 130
- pairRisk (cluster.index), 46
- pbvn (pmvn), 138
- pcif (eventpois), 89
- phreg, 130
- phreg_IPTW, 132
- phreg_rct, 134
- phregR, 131
- piecewise.data (npc), 130
- piecewise.twostage (npc), 130
- plack.cif, 136
- plack.cif2 (plack.cif), 136
- plot.phreg, 136
- plot_twin, 137
- plotConfRegion (plot.phreg), 136
- plotConfRegionSE (plot.phreg), 136
- plotcr (npc), 130
- plot0.predictphreg (phreg), 130
- plotpredictphreg (phreg), 130
- plotstrata (plot.phreg), 136
- plotSurvd
 - (interval.logitsurv.discrete), 113
- pmvn, 138
- ppch (rpch), 157
- pre.cifs (sim.cif), 159
- pred.cif.boot (Bootphreg), 35
- predict (mlogit), 128
- predict.phreg, 139
- predictCumhaz (fast.approx), 93
- predictdFG (doubleFGR), 66
- predictGLM (summaryGLM), 173
- predictlogitSurvd
 - (interval.logitsurv.discrete), 113
- predict0.phreg (phreg), 130
- predictPairPlack (npc), 130
- predictrecreg (phreg), 130
- predictRisk, 140
- predictRisk.binreg, 141
- predictRisk.cifreg
 - (predictRisk.binreg), 141
- predictRisk.cifregFG
 - (predictRisk.binreg), 141
- predictRisk.phreg (predictRisk.binreg), 141
- predictRisk.recreg
 - (predictRisk.binreg), 141
- predictSurvd
 - (interval.logitsurv.discrete), 113
- predicttime (recreg), 149
- print.casewise, 141
- print.Event (Event), 87
- prob.exceed.recurrent, 142
- prob.exceedBiRecurrent
 - (prob.exceed.recurrent), 142
- prob.exceedBiRecurrentStrata
 - (prob.exceed.recurrent), 142
- prob.exceedRecurrent
 - (prob.exceed.recurrent), 142
- prob.exceedRecurrentStrata

- (prob.exceed.recurrent), 142
- procform (npc), 130
- procform3 (npc), 130
- procformdata (npc), 130
- prrt, 143
- random.cif, 143
- randomDes (survival.twostage), 174
- ratioATE (resmeanATE), 154
- rbind.Event (Event), 87
- rchaz, 146
- rchazC, 147
- rchazl (rcrisk), 148
- rcrisk, 148
- rcrisks (rcrisk), 148
- read.fit (sim.cox), 160
- read.phreg (sim.cox), 160
- readmargsurv (survival.twostage), 174
- readPhreg (phreg), 130
- recmarg (recurrentMarginal), 151
- recreg, 149
- recregIPCW (recreg), 149
- recurrentMarginal, 151
- recurrentMarginalAIPCW
 - (recurrentMarginal), 151
- recurrentMarginalPhreg
 - (recurrentMarginal), 151
- resmean.phreg, 153
- resmeanATE, 154
- resmeanIPCW, 155
- resmeanIPCWold (resmeanIPCW), 155
- revcumsum (predict.phreg), 139
- revcumsum2strata (predict.phreg), 139
- revcumsum2stratafdN (predict.phreg), 139
- revcumsumidstratasum (predict.phreg), 139
- revcumsumidstratasumCov
 - (predict.phreg), 139
- revcumsumstrata (predict.phreg), 139
- revcumsumstratasum (predict.phreg), 139
- rmst.phreg (resmean.phreg), 153
- rmstATE (resmeanATE), 154
- rmstIPCW (resmeanIPCW), 155
- rmstRatio (binregRatio), 25
- rmvn (pmvn), 138
- robust.basehaz.phreg (predict.phreg), 139
- robust.phreg (phreg), 130
- rpch, 157
- rr.cif (cor.cif), 48
- scalecumhaz (recreg), 149
- scoreMVN (pmvn), 138
- setup.cif (FG_AugmentCifstrata), 97
- setup.phreg (sim.cox), 160
- showfitsim (simRecurrentII), 168
- showfitsimList (simRecurrentII), 168
- sim (npc), 130
- sim.base (sim.cox), 160
- sim.cause.cox, 158
- sim.cif, 159
- sim.cifs (sim.cif), 159
- sim.cifsRestrict (sim.cif), 159
- sim.cox, 160
- sim.phreg (sim.cox), 160
- sim.phregs (sim.cox), 160
- sim.recurrent, 161
- simAalenFrailty, 162
- simbinClaytonOakes.family.ace (npc), 130
- simbinClaytonOakes.pairs (npc), 130
- simbinClaytonOakes.twin.ace (npc), 130
- simBinFam (npc), 130
- simBinFam2 (npc), 130
- simBinPlack (npc), 130
- simCens (rchaz), 146
- simClaytonOakes, 163
- simClaytonOakes.family.ace (npc), 130
- simClaytonOakes.twin.ace (npc), 130
- simClaytonOakesLam (simClaytonOakes), 163
- simClaytonOakesWei, 164
- simCompete.simple (npc), 130
- simCompete.twin.ace (npc), 130
- simCox (npc), 130
- simFrailty.simple (npc), 130
- simGLcox, 165
- simlogitSurv
 - (interval.logitsurv.discrete), 113
- simMultistate, 166
- simnordic (npc), 130
- simrchaz (rchaz), 146
- simRecurrent (simRecurrentII), 168
- simRecurrentII, 168
- simRecurrentList (simRecurrentII), 168
- simRecurrentTS, 170
- simsubdist (sim.cif), 159
- simSurvFam (npc), 130

simTTP(interval.logitsurv.discrete),
 113
 simul.cifs(FG_AugmentCifstrata), 97
 simulate.cox(sim.cox), 160
 slope.process(casewise.test), 40
 squareintHdM(simRecurrentII), 168
 sTRACE(TRACE), 182
 strataAugment(recreg), 149
 strataC(FG_AugmentCifstrata), 97
 subdist(sim.cif), 159
 summary.cor, 172
 summary.Event(Event), 87
 summarybase.phreg(phreg), 130
 summaryGLM, 173
 summaryRecurrentobject
 (prob.exceed.recurrent), 142
 summaryTimeobject
 (prob.exceed.recurrent), 142
 sumstrata(predict.phreg), 139
 surv.boxarea(npc), 130
 survival.twostage, 174
 survivalG, 179
 survivalGtime(survivalG), 179

 tailstrata(predict.phreg), 139
 test.conc, 181
 tetrachoric, 181
 tie.breaker(recurrentMarginal), 151
 TRACE, 182
 ttpd, 183
 tTRACE(TRACE), 182
 twin.clustertrunc, 183
 twin.polygen.design(npc), 130
 twinbmi, 184
 twinlm, 38, 185, 188
 twinlm.strata, 38, 186
 twinlm.time, 38, 186
 twinlm.time(bptwin), 36
 twinsim, 38, 186, 187
 twinstut, 188
 twostage.aalen(survival.twostage), 174
 twostage.cox.aalen(survival.twostage),
 174
 twostage.coxph(survival.twostage), 174
 twostage.phreg(survival.twostage), 174
 twostageMLE, 189
 twostageREC(recreg), 149

 vecAllStrata(cifreg), 42

 WA_recurrent, 190