

# Package ‘mets’

October 13, 2022

**Type** Package

**Title** Analysis of Multivariate Event Times

**Version** 1.3.1

**Date** 2022-10-01

**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>. Also contains two-stage binomial modelling that can do pairwise odds-ratio dependence modelling based marginal logistic regression models. This is an alternative to the alternating logistic regression approach (ALR).

**License** GPL (>= 2)

**LazyLoad** yes

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

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

**Depends** R (>= 3.5), timereg (>= 1.9.4), lava (>= 1.6.6)

**Imports** mvtnorm, numDeriv, compiler, Rcpp, splines, survival (>= 2.43-1),

**Suggests** optimx, prodlim, cmprsk, testthat (>= 0.11), ucminf, knitr, rmarkdown, ggplot2, cowplot, icenReg

**VignetteBuilder** knitr

**ByteCompile** yes

**LinkingTo** Rcpp, RcppArmadillo, mvtnorm

**SystemRequirements** C++11

**Encoding** UTF-8

**RoxygenNote** 7.2.1

**NeedsCompilation** yes

**Repository** CRAN

**Date/Publication** 2022-10-02 15:50:02 UTC

## R topics documented:

mets-package	4
aalenfrailty	5
aalenMets	6
back2timereg	7
base1cumhaz	7
base44cumhaz	8
base4cumhaz	8
basehazplot.phreg	8
bicomprisk	10
BinAugmentCifstrata	12
binomial.twostage	14
binreg	18
binregATE	21
binregCasewise	23
biprobit	24
blocksample	26
Bootphreg	28
bptwin	29
casewise	31
casewise.test	32
cif	33
cifreg	34
ClaytonOakes	36
cluster.index	38
concordanceCor	39
cor.cif	40
count.history	45
covarianceRecurrent	46
daggregate	47
Dbvn	49
dby	50
dcor	52
dcut	53
dermalridges	55
dermalridgesMZ	55
divide.conquer	56
divide.conquer.timereg	57
dlag	58
doubleFGR	58
dprint	61
drcumhaz	62

dreg . . . . .	62
drelevel . . . . .	65
dsort . . . . .	67
dspline . . . . .	68
dtable . . . . .	69
dtransform . . . . .	71
easy.binomial.twostage . . . . .	71
easy.survival.twostage . . . . .	75
Effbinreg . . . . .	77
EVaddGam . . . . .	79
eventpois . . . . .	80
EventSplit . . . . .	81
familycluster.index . . . . .	83
familyclusterWithProbands.index . . . . .	84
fast.approx . . . . .	85
fast.pattern . . . . .	86
fast.reshape . . . . .	86
FG_AugmentCifstrata . . . . .	89
ghaplos . . . . .	91
gof.phreg . . . . .	91
gofG.phreg . . . . .	92
gofM.phreg . . . . .	93
gofZ.phreg . . . . .	95
Grandom.cif . . . . .	96
hapfreqs . . . . .	100
haplo.surv.discrete . . . . .	100
haploX . . . . .	103
interval.logitsurv.discrete . . . . .	103
ipw . . . . .	104
ipw2 . . . . .	106
km . . . . .	108
lifecourse . . . . .	109
lifetable.matrix . . . . .	111
LinSpline . . . . .	112
logitSurv . . . . .	112
mediatorSurv . . . . .	113
medweight . . . . .	115
mena . . . . .	115
mets.options . . . . .	116
migr . . . . .	116
mlogit . . . . .	117
multcif . . . . .	118
np . . . . .	118
npc . . . . .	119
phreg . . . . .	119
phregR . . . . .	120
plack.cif . . . . .	121
pmvn . . . . .	121

predict.phreg . . . . .	122
print.casewise . . . . .	123
prob.exceed.recurrent . . . . .	123
prt . . . . .	125
random.cif . . . . .	126
recreg . . . . .	128
recurrentMarginal . . . . .	130
resmean.phreg . . . . .	133
resmeanIPCW . . . . .	134
rpch . . . . .	136
simAalenFrailty . . . . .	137
simClaytonOakes . . . . .	138
simClaytonOakesWei . . . . .	139
simMultistate . . . . .	140
simRecurrent . . . . .	142
simRecurrentII . . . . .	143
simRecurrentTS . . . . .	145
summary.cor . . . . .	147
survival.iterative . . . . .	148
survival.twostage . . . . .	152
survival.twostageCC . . . . .	157
test.conc . . . . .	159
tetrachoric . . . . .	160
ttpd . . . . .	160
twin.clustertrunc . . . . .	161
twinbmi . . . . .	162
twinlm . . . . .	162
twinsim . . . . .	165
twinstut . . . . .	166
twostageMLE . . . . .	166

<b>Index</b>	<b>168</b>
--------------	------------

---

mets-package	<i>Analysis of Multivariate Events</i>
--------------	--

---

## Description

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

## Author(s)

Klaus K. Holst and Thomas Scheike

**Examples**

```
## To appear
```

---

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<-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, ...)
```

**Arguments**

formula	formula with 'Surv' outcome (see coxph)
data	data frame
...	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)
out <- aalenMets(Surv(time,cause==1)~tcell+platelet+age,data=bmt)
summary(out)
```

---

back2timereg                      *Convert to timereg object*

---

**Description**

convert to timereg object

**Usage**

```
back2timereg(obj)
```

**Arguments**

obj                      no use

**Author(s)**

Thomas Scheike

---

base1cumhaz                      *rate of CRBSI for HPN patients of Copenhagen*

---

**Description**

rate of CRBSI for HPN patients of Copenhagen

**Source**

Estimated data

---

base44cumhaz	<i>rate of Occlusion/Thrombosis complication for catheter of HPN patients of Copenhagen</i>
--------------	---

---

**Description**

rate of Occlusion/Thrombosis complication for catheter of HPN patients of Copenhagen

**Source**

Estimated data

---

base4cumhaz	<i>rate of Mechanical (hole/defect) complication for catheter of HPN patients of Copenhagen</i>
-------------	---

---

**Description**

rate of Mechanical (hole/defect) complication for catheter of HPN patients of Copenhagen

**Source**

Estimated data

---

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
polygon	to get standard error in shaded form
level	of standard errors
stratas	wich strata to plot
robust	to use robust standard errors if possible
conf.type	"plain" or "log" transformed
...	Additional arguments to lower level funtions

### Author(s)

Klaus K. Holst, Thomas Scheike

### Examples

```

data(TRACE)
dcut(TRACE) <- ~.
out1 <- phreg(Surv(time,status==9)~vf+chf+strata(wmicat.4),data=TRACE)

par(mfrow=c(2,2))
bplot(out1)
bplot(out1, stratas=c(0,3))
bplot(out1, stratas=c(0,3), col=2:3, lty=1:2, se=TRUE)
bplot(out1, stratas=c(0), col=2, lty=2, se=TRUE, polygon=FALSE)
bplot(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 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 additional weight used for paired competing risks data.
prodlim	prodlim to use prodlim estimator (Aalen-Johansen) rather than IPCW weighted estimator based on comp.risk function. These are equivalent in the case of no covariates. These 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 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 <- 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)

```

---

BinAugmentCifstrata    *Augmentation for Binomial regression based on stratified NPMLE Cif  
(Aalen-Johansen)*

---

## 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**

```
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 composite likelihood based on all pairwise bivariate models.</i>
-------------------	---

---

### Description

The 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**

margbin	Marginal binomial model
data	data frame
method	Scoring method "nr", for lava NR optimizer
detail	Detail
clusters	Cluster variable
silent	Debug information
weights	Weights for log-likelihood, can be used for each type of outcome in 2x2 tables.
theta	Starting values for variance components
theta.des	design for dependence parameters, when pairs are given the indeces of the theta-design for this pair, is given in pairs as column 5
var.link	Link function for variance
var.par	parametrization
var.func	when alternative parametrizations are used this function can specify how the paramters are related to the $\lambda_j$ 's.
iid	Calculate i.i.d. decomposition when iid>=1, when iid=2 then avoids adding the uncertainty for marginal paramters for additive gamma model (default).
notaylor	Taylor expansion
model	model
marginal.p	vector of marginal probabilities
beta.iid	iid decomposition of marginal probability estimates for each subject, if based on GLM model this is computed.
Dbeta.iid	derivatives of marginal model wrt marginal parameters, if based on GLM model this is computed.
strata	strata for fitting: considers only pairs where both are from same strata
max.clust	max clusters
se.clusters	clusters for iid decomposition for roubst standard errors
numDeriv	uses Fisher scoring aprox of second derivative if 0, otherwise numerical derivatives
random.design	random effect design for additive gamma model, when pairs are given the indeces of the pairs random.design rows are given as columns 3:4
pairs	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.
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. $Y1+Y2 \geq 1$ .
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  $n \times d$ . 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  $\text{lamtot} = v_1^T \lambda$  is fixed over all clusters as it would be for the ACE model below.

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

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

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

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



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

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

with the inverse laplace of the gamma distribution with mean 1 and variance  $\text{lamtot}$ .

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

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

here using  $\text{theta.des}$  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<11000)
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)

## refers to zygoty of first subject in eash 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 zygoty of first subject in eash 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$pardes
head(out$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$pardes)
summary(bints)

data <- simbinClaytonOakes.twin.ace(10000,2,1,beta=NULL,alpha=NULL)
out <- twin.polygen.design(data,id="cluster",zygname="zygosity")
out$pardes
head(out$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$pardes)
summary(bintwin)
concordanceTwinACE(bintwin)

```

**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,
  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,
  ...
)
```

**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
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
...	Additional arguments to lower level funtions

**Details**

Based on binomial regression IPCW response estimating equation:

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

for IPCW adjusted responses.

logitIPCW instead considers

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

a standard logistic regression with weights that adjust for IPCW.

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

### Examples

```
data(bmt)
# logistic regression with IPCW binomial regression
out <- binreg(Event(time,cause)~tcell+platelet,bmt,time=50)
summary(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)
bplot(cif1)
bplot(cif2,add=TRUE,col=2)
```

```

cifs1 <- binreg(Event(time,cause)~tcell+platelet+age,bmt,cause=1,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)+cluster(id))
summary(cifdob)

expit <- function(z) 1/(1+exp(-z))

riskratio <- function(p) {
  expit <- function(z) 1/(1+exp(-z)) ## expit
  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 <- expit(lp)
  return(p[1]/p[2])
}

estimate(cifdob,f=riskratio)

```

---

binregATE

*Average Treatment effect for censored competing risks data using Binomial 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,
  kaplan.meier = TRUE,
  cens.code = 0,
  no.opt = FALSE,

```

```

    method = "nr",
    augmentation = 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
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
...	Additional arguments to lower level funtions

### 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. If there are no censorings this is performing ordinary logistic regression modelling.

This is then model using a logistic regression using the standard binary double robust estimating equations that are then IPCW censoring adjusted using binomial regression.

Rather than binomial regression we also consider a IPCW weighted version of standard logistic regression `logitIPCWATE`.

The original version of the program with only binary treatment `binregATEbin` take binary-numeric as input for the treatment, and also computes the ATT and ATC, average treatment effect on the treated (ATT),  $E(Y(1) - Y(0) | A=1)$ , and non-treated, respectively. Experimental version.

### 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

```

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

### Examples

```

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$x, type="l", xlab="x", ylab="Concordance", lwd=2, xaxs="i")
confband(pp$x,pp[,2],pp[,3],polygon=TRUE,lty=0,col=Col(1))

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

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

*Block sampling*


---

## 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 whether to sample with replacement
...	additional arguments to lower level functions

**Details**

Original id is stored in the attribute 'id'

**Value**

data.frame

**Author(s)**

Klaus K. Holst

**Examples**

```
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)
```

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)
```

---

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	specififes 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 <- 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 <-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)

```

## Description

Cumulative incidence with robust standard errors

**Usage**

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

**Arguments**

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

**Author(s)**

Thomas Scheike

**Examples**

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

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

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

---

cifreg

*CIF regression*

---

**Description**

CIF logistic for propodds=1 default CIF Fine-Gray (cloglog) regression for propodds=NULL

**Usage**

```
cifreg(
  formula,
  data = data,
  cause = 1,
  cens.code = 0,
  cens.model = ~1,
  weights = NULL,
  offset = NULL,
  Gc = NULL,
```

```

propodds = 1,
...
)

```

### Arguments

formula	formula with 'Event' outcome
data	data frame
cause	of interest
cens.code	code of censoring
cens.model	for stratified Cox model without covariates
weights	weights for FG score equations
offset	offsets for FG model
Gc	censoring weights for time argument, default is to calculate these with a Kaplan-Meier estimator, should then give G_c(T_i-)
propodds	1 is logistic model, NULL is fine-gray model
...	Additional arguments to lower level funtions

### 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. 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

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

The iid decomposition of the beta's, however, also have a censoring term that is also is computed and added to UUiid (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 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
data(bmt,package="timereg")
bmt$time <- bmt$time+runif(nrow(bmt))*0.01
bmt$id <- 1:nrow(bmt)

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

## Fine-Gray model
fg=cifreg(Event(time,cause)~tcell+platelet+age,data=bmt,cause=1,propodds=NULL)
summary(fg)
plot(fg)
nd <- data.frame(tcell=c(1,0),platelet=0,age=0)
pfg <- predict(fg,nd)
plot(pfg)

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

### predictions with CI based on iid decomposition of baseline and beta
fg <- cifreg(Event(time,cause)~tcell+platelet+age,data=bmt,cause=1,propodds=NULL,cox.prep=TRUE)
Biid <- mets::iid.baseline.cifreg(fg,time=20)
FGprediid(Biid,bmt[1:5,])

```

---

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

### Author(s)

Klaus K. Holst

### Examples

```

set.seed(1)
d <- subset(simClaytonOakes(500,4,2,1,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
plot(ts)
plot(e2,add=TRUE)

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

---

cluster.index                    *Finds subjects related to same cluster*

---

### 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 which 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  $h(\cdot)$  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>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, <code>KM</code> is Kaplan-Meier alternatively it may be <code>"cox"</code>
<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**

```
library("timereg")
data(multcif);
multcif$cause[multcif$cause==0] <- 2
zyg <- rep(rbinom(200,1,0.5),each=2)
theta.des <- model.matrix(~-1+factor(zyg))

times=seq(0.05,1,by=0.05) # to speed up computations use only these time-points
add<-comp.risk(Event(time,cause)~+1+cluster(id),data=multcif,cause=1,
               n.sim=0,times=times,model="fg",max.clust=NULL)
add2<-comp.risk(Event(time,cause)~+1+cluster(id),data=multcif,cause=2,
                n.sim=0,times=times,model="fg",max.clust=NULL)

out1<-cor.cif(add,data=multcif,cause1=1,cause2=1)
summary(out1)

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

##out3<-cor.cif(add,data=multcif,cause1=1,cause2=2,cif2=add2)
##summary(out3)
#####
# investigating further models using parfunc and dparfunc
#####
## Reduce Ex.Timings
set.seed(100)
prt<-simnordic.random(2000, cordz=2, cormz=5)
prt$status <-prt$cause
table(prt$status)

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

parfunc <- function(par,t,pardes)
{
par <- pardes %*% c(par[1],par[2]) +
  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 = 1:2,
  names.count = "Count",
  lag = TRUE
)
```

### Arguments

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

### Author(s)

Thomas Scheike

### Examples

```
#####
## getting some rates to mimick
#####

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

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

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

---

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

---

### Description

Estimation of probability of more that k events for recurrent events process where there is terminal event

### Usage

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

### Arguments

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

**Author(s)**

Thomas Scheike

**References**

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

**Examples**

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

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

par(mfrow=c(1,3))
plot(covrp)

### with strata, each strata in matrix column, provides basis for fast Bootstrap
covrpS <- covarianceRecurrentS(rr,1,2,status="status",death="death",
                               start="entry",stop="time",strata="strata",id="id",names.count="Count")
```

---

daggregate

*aggregating for for data frames*


---

**Description**

aggregating for for data frames

**Usage**

```
daggregate(
  data,
  y = NULL,
  x = NULL,
  subset,
```

```

...,
fun = "summary",
regex = mets.options()$regex,
missing = FALSE,
remove.empty = FALSE,
matrix = FALSE,
silent = FALSE,
na.action = na.pass,
convert = NULL
)

```

### Arguments

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

### Examples

```

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

*Derivatives of the bivariate normal cumulative distribution function*


---

## 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 *Calculate summary statistics grouped by*

---

### 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"))
```

---

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**

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



---

`divide.conquer.timereg`*Split a data set and run function from timereg and aggregate*

---

**Description**

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

**Usage**

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

**Arguments**

<code>func</code>	called function
<code>data</code>	data-frame
<code>size</code>	size of splits
<code>...</code>	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))
}
```

---

<code>dlag</code>	<i>Lag operator</i>
-------------------	---------------------

---

**Description**

Lag operator

**Usage**

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

**Arguments**

<code>data</code>	data.frame or vector
<code>x</code>	optional column names or formula
<code>k</code>	lag (vector of integers)
<code>combine</code>	combine results with original data.frame
<code>simplify</code>	Return vector if possible
<code>names</code>	optional new column names
<code>...</code>	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])
```

---

<code>doubleFGR</code>	<i>Double CIF Fine-Gray model with two causes</i>
------------------------	---

---

**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)
}

```

---

dprint

*list, head, print, tail*


---

## Description

listing for data frames

## Usage

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

## Arguments

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

```

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

dlist(d, ~a+b+c)
dlist(d, ~a+b+c|a<0 & b>0)
## listing all :

```

```

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

```

---

drcumhaz

*Rate for leaving HPN program for patients of Copenhagen*


---

### Description

Rate for leaving HPN program for patients of Copenhagen

### Source

Estimated data

---

dreg

*Regression for data frames with dutility call*


---

### Description

Regression for data frames with dutility call

### Usage

```

dreg(
  data,
  y,
  x = NULL,
  z = NULL,
  x.oneatatime = 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**

data	data frame
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.
z	name of variable, or fomula, or names of variables on data frame.
x.oneatatime	x's one at a time
x.base.names	base covarirates
z.arg	what is Z, c("clever","base","group","condition"), clever decides based on type of Z, base means that Z is used as fixed baseline covaraites 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
fun.	function lm is default
summary.	summary to use
regex	regex
convert	convert
doSummary	doSummary or not
special	special's
equal	to do pairwise stuff
test	development argument
...	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.oneatime=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.=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**

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.
ref	new reference variable
newlevels	to combine levels of factor in data frame
regex	for regular expressions.
sep	separator for naming of cut names.
overwrite	to overwrite variable
...	Optional additional arguments

**Author(s)**

Klaus K. Holst and Thomas Scheike

**Examples**

```

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	separator for naming of cut names.
na.rm	to remove NA for grouping variables.
labels	to use for cut groups
all	to do all variables, even when breaks are not unique
...	Optional additional arguments

**Author(s)**

Thomas Scheike

**Examples**

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

nd = data.frame(age=50, vf=0, chf=0, wmi=seq(0.4, 3, by=0.01))
nd <- dspline(nd, ~wmi, breaks=c(1, 1.3, 1.7))
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**

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.
...	Optional additional arguments
level	1 for all marginal tables, 2 for all 2 by 2 tables, and null for the full table, possible versus group variable
response	For level=2, only produce tables with columns given by 'response' (index)
flat	produce flat tables
total	add total counts/proportions
prop	Proportions instead of counts (vector of margins)
summary	summary function

**Author(s)**

Klaus K. Holst and Thomas Scheike

**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 composite 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

margbin	Marginal binomial model
data	data frame
method	Scoring method
response	name of response variable in data frame
id	name of cluster variable in data frame
Nit	Number of iterations
detail	Detail for more output for iterations
silent	Debug information
weights	Weights for log-likelihood, can be used for each type of outcome in 2x2 tables.
control	Optimization arguments
theta	Starting values for variance components
theta.formula	design for dependence, either formula or design function
desnames	names for dependence parameters
deshelp	if 1 then prints out some data sets that are used, on on which the design function operates
var.link	Link function for variance
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<10000)
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$cage <- scale(twinstut$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,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 zygosity of first subject in eash pair : zyg1
## could also use zyg2 (since zyg2=zyg1 within twinpair's)
desfs <- function(x,num1="zyg1",namesdes=c("mz","dz","os"))
  c(x[num1]=="mz",x[num1]=="dz",x[num1]=="os")*1

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

summary(out3)

## Reduce Ex.Timings
n <- 5000
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 desfai2 rather than desfai 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)

```

---

easy.survival.twostage

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

---

## Description

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

## Usage

```

easy.survival.twostage(
  margsurv = NULL,
  data = parent.frame(),
  method = "nlminb",
  status = "status",
  time = "time",
  entry = NULL,
  id = "id",

```

```

Nit = 60,
detail = 0,
silent = 1,
weights = NULL,
control = list(),
theta = NULL,
theta.formula = NULL,
desnames = NULL,
deshelp = 0,
var.link = 1,
step = 0.5,
model = "plackett",
marginal.surv = NULL,
strata = NULL,
se.clusters = NULL
)

```

### Arguments

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

## Details

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

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

## Examples

```
library(mets)
data("prt", package="mets")
prtsam <- blocksample(prt, idvar="id", 1e3, replace=FALSE)
margp <- coxph(Surv(time, status==1)~factor(country), data=prtsam)
fitco <- survival.twestage(margp, data=prtsam, clusters=prtsam$id)
summary(fitco)

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

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

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

marg <- coxph(Surv(time, status)~factor(num), data=dfam)
out3 <- easy.survival.twestage(marg, data=dfam, time="time", status="status", id="id",
  deshelp=0,
  method="nr", theta.formula=desfs,
  model="plackett",
  desnames=c("parent-parent", "parent-child", "child-cild"))
summary(out3)
```

## Description

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

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

**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
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
h	h for estimating equation
MCAugment	iid of h and censoring model
...	Additional arguments to lower level funtions
model	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 prepcif 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 prepcif 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

*Relative risk for additive gamma model*

---

**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))
```

---

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

---

EventSplit

*Event split with two time-scales, time and gaptime*

---

**Description**

splits after cut times for the two time-scales.

**Usage**

```
EventSplit(
  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 <- EventSplit(rr,cuts=600,cuttime="time", gaptime="gaptime",gaptime.entry="gapstart")
rr2 <- EventSplit(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	<i>Fast approximation</i>
-------------	---------------------------

---

**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 separating 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

```

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

##### From wide-format
(dd <- fast.reshape(d,"y"))
## Same with explicit setting new id and number variable/column names
## and 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 <- sim(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 <- lvm(c(y1,y2,y3,y4)~ x+z1+z3+z5)

```



```

d <- sim(m,3)
d
fast.reshape(d,c("y","z"))

##### not-varying syntax:
fast.reshape(d,-c("x"))

##### Automatically define varying variables from trailing digits
fast.reshape(d)

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

```

---

FG\_AugmentCifstrata    *Augmentation for Fine-Gray model based on stratified NPMLE Cif (Aalen-Johansen)*

---

## 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

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

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 <sub>1</sub> )
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)

```

ghaplos

*ghaplos haplo-types for subjects of haploX data***Description**

ghaplos haplo-types for subjects of haploX data

**Source**

Simulated data

gof.phreg

*GOF for Cox PH regression***Description**

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

**Usage**

```

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

**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  $(z_1, \dots, z_m)$  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 polygenetic 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>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.

<code>theta.des</code>	specifies a regression design for the cross-odds-ratio parameters.
<code>weights</code>	weights for score equations.
<code>step</code>	specifies the step size for the Newton-Raphson algorithm.
<code>sym</code>	1 for symmetri and 0 otherwise
<code>same.cens</code>	if true then censoring within clusters are assumed to be the same variable, default is independent censoring.
<code>censoring.weights</code>	Censoring probabilities
<code>silent</code>	debug information
<code>var.link</code>	if <code>var.link=1</code> then <code>var</code> is on log-scale.
<code>score.method</code>	default uses "nlminb" optimizer, alternatively, use the "nr" algorithm.
<code>entry</code>	entry-age in case of delayed entry. Then two causes must be given.
<code>estimator</code>	estimator
<code>trunkp</code>	gives probability of survival for delayed entry, and related to entry-ages given above.
<code>admin.cens</code>	Administrative censoring
<code>random.design</code>	specifies a regression design of 0/1's for the random effects.
<code>...</code>	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  $d$   $\lambda$  parameters with the ( $k$ ) underlying  $\theta$  parameters

$$\lambda = pard\theta$$

### Value

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

<code>theta</code>	estimate of parameters of model.
<code>var.theta</code>	variance for gamma.
<code>hess</code>	the derivative of the used score.
<code>score</code>	scores at final stage.
<code>theta.iid</code>	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<-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 and 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=parides,random.design=des.rv)
summary(outacem)

```

---

hapfreqs	<i>hapfreqs data set</i>
----------	--------------------------

---

### Description

hapfreqs data set

### Source

Simulated 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

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

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

baseline=hapfreqs$haplotype[9]
baseline

designfntypes <- 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)

data(hHaplos)
data(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=designfntypes)
names(out$coef) <- c(desnames,types)
out$coef
summary(out)
```

---

haploX	<i>haploX covariates and response for haplo survival discrete survival</i>
--------	--

---

**Description**

haploX covariates and response for haplo survival discrete survival

**Source**

Simulated data

---

interval.logitsurv.discrete	<i>Discrete time to event interval censored data</i>
-----------------------------	--

---

**Description**

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

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

**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

This is thus also the cumulative odds model, since

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

The baseline  $G(t)$  is written as  $cumsum(\exp(\alpha))$  and this is not the standard parametrization that takes log of  $G(t)$  as the parameters.

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  $j+1$ .

Likelihood is maximized:

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

### Author(s)

Thomas Scheike

### Examples

```
data(tppd)
dtable(tppd, ~entry+time2)
out <- interval.logitsurv.discrete(Interval(entry, time2)~X1+X2+X3+X4, ttpd)
summary(out)

pred <- predictlogitSurvvd(out, se=FALSE)
plotSurvvd(pred)
```

### 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**

<code>formula</code>	Formula specifying the censoring model
<code>data</code>	data frame
<code>cluster</code>	clustering variable
<code>same.cens</code>	For clustered data, should same censoring be assumed (bivariate probability calculated as minimum of the marginal probabilities)
<code>obs.only</code>	Return data with uncensored observations only
<code>weight.name</code>	Name of weight variable in the new data.frame
<code>trunc.prob</code>	If TRUE truncation probabilities are also calculated and stored in 'weight.name2' (based on Clayton-Oakes gamma frailty model)
<code>weight.name2</code>	Name of truncation probabilities
<code>indi.weight</code>	Name of individual censoring weight in the new data.frame
<code>cens.model</code>	Censoring model (default Aalen's additive model)
<code>pairs</code>	For paired data (e.g. twins) only the complete pairs are returned (With pairs=TRUE)
<code>theta.formula</code>	Model for the dependence parameter in the Clayton-Oakes model (truncation only)
<code>...</code>	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	nam of entry-time for truncation.
time	name of time variable on data frame.
cause	name of cause indicator on data frame.
same.cens	For clustered data, should same censoring be assumed and same truncation (bivariate probability calculated as minimum of the marginal probabilities)
cluster	name of clustering variable
pairs	For paired data (e.g. twins) only the complete pairs are returned (With pairs=TRUE)
strata	name of strata variable to get weights stratified.
obs.only	Return data with uncensored observations only
cens.formula	model for Cox models for truncation and right censoring times.
cens.code	censoring.code
pair.cweight	Name of weight variable in the new data.frame for right censoring of pairs
pair.tweight	Name of weight variable in the new data.frame for left truncation of pairs
pair.weight	Name of weight variable in the new data.frame for right censoring and left truncation of pairs
cname	Name of weight variable in the new data.frame for right censoring of individuals
tname	Name of weight variable in the new data.frame for left truncation of individuals
weight.name	Name of weight variable in the new data.frame for right censoring and left truncation of individuals
prec.factor	To let tied censoring and truncation times come after the death times.
...	Additional arguments to censoring model

**Author(s)**

Thomas Scheike

**Examples**

```
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 <- 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)

d1 <- d[!d$truncated,]
d1 <- ipw2(d1,cluster="id",same.cens=TRUE,time="time",entrytime="entry",cause="cause",
          strata="strata",prec.factor=100)
c1 <- comp.risk(Event(time,cause)~+1+
               cluster(id),
               data=d1,cause=1,model="fg",
               weights=d1$indi.weights,cens.weights=rep(1,nrow(d1)),
               times=times,max.clust=NULL,n.sim=0)
pc1 <- predict(c1,X=1,se=0)
lines(pc1$time,pc1$P1,col=2)
mm=model.matrix(~-1+factor(zyg),data=d1)
out2<-random.cif(c1,data=d1,cause1=1,cause2=1,theta.des=mm,
                 weights=d1$weights,censoring.weights=rep(1,nrow(d1)))
summary(out2)

```

km

*Kaplan-Meier with robust standard errors***Description**

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

**Usage**

```

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

```

**Arguments**

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

**Author(s)**

Thomas Scheike

**Examples**

```

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

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

```

lifecourse

*Life-course plot***Description**

Life-course plot for event life data with recurrent events

**Usage**

```

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

```

**Arguments**

formula	Formula (Event(start,slut,status) ~ ...)
data	data.frame
id	Id variable

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

### Author(s)

Thomas Scheike, Klaus K. Holst

### Examples

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

ll = lifecourse(Event(start,slut,status)~id,data,id="id",group=~group,col=1:2)
op <- par(mfrow=c(1,2))
ll = lifecourse(Event(start,slut,status)~id,data,id="id",by=~group)
par(op)
legends=c("censored","pregnant","married")
ll = 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)))
summary(glm(events ~ offset(log(atrisk))+factor(int.end)*vf + sex*vf,
  data=d,poisson))
```

---

 LinSpline

*Simple linear spline*


---

**Description**

Simple linear spline

**Usage**

```
LinSpline(x, knots, num = TRUE, name = "Spline")
```

**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**

```
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

---

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

*Multinomial regression based on phreg regression*


---

**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**

```

data(bmt)
dfactor(bmt) <- cause1f~cause
drelevel(bmt,ref=3) <- cause3f~cause
dlevels(bmt)

mreg <- mlogit(cause1f~+1,bmt)
summary(mreg)

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

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

## inverse information standard errors
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"))
predictmlogit(mreg,newdata,response=FALSE)
predictmlogit(mreg,newdata)

```

---

multcif

*Multivariate Cumulative Incidence Function example data set*

---

**Description**

Multivariate Cumulative Incidence Function example data set

**Source**

Simulated data

---

np

*np data set*

---

**Description**

np data set

**Source**

Simulated data

---

npc *For internal use*

---

**Description**

For internal use

**Author(s)**

Klaus K. Holst

---

phreg *Fast Cox PH regression*

---

**Description**

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

**Usage**

```
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**

```

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

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

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

## making iid decomposition of regression parameters
betaiid <- iid(out1)

## making iid decomposition of baseline at a specific time-point
Aiiid <- mets::iid.baseline.phreg(out1,time=30)

```

---

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 functions

**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

---

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

---

**Description**

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

**Usage**

plack.cif(cif1, cif2, object)

**Arguments**

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

**Author(s)**

Thomas Scheike

---

pmvn	<i>Multivariate normal distribution function</i>
------	--

---

**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

*Predictions from proportional hazards model*


---

**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
individual.time	to use one (individual) time per subject, and then newdata and times have same length and makes only predictions for these individual times.
tminus	to make predictions in T- that is strictly before given times, useful for IPCW techniques
se	with standard errors and upper and lower confidence intervals.
robust	to get robust se's.
conf.type	transformation for survival estimates, default is log
conf.int	significance level

km                   to use Kaplan-Meier product-limit for baseline

$$S_{s0}(t) = (1 - dA_{s0}(t))$$

, otherwise take exp of cumulative baseline.

...                   Additional arguments to plot functions

---

print.casewise           *prints Concordance test*

---

### 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 than k events for recurrent events process*

---

### Description

Estimation of probability of more than 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(
  data,
  type,
  status = "status",
  death = "death",
  start = "start",
  stop = "stop",
  id = "id",
  times = NULL,
  exceed = NULL,
  cifmets = TRUE,
  strata = NULL,
  all.cifs = FALSE,
  ...
)

```

**Arguments**

<code>data</code>	data-frame
<code>type</code>	type of event (code) related to status
<code>status</code>	name of status
<code>death</code>	name of death indicator
<code>start</code>	start stop call of Hist() of prodlim
<code>stop</code>	start stop call of Hist() of prodlim
<code>id</code>	id
<code>times</code>	time at which to get probabilities $P(N_1(t) \geq n)$
<code>exceed</code>	$n$ 's for which to compute probabilities $P(N_1(t) \geq n)$
<code>cifmets</code>	if true uses cif of mets package rather than prodlim
<code>strata</code>	to stratify according to variable, only for cifmets=TRUE, when strata is given then only consider the output in the all.cifs
<code>all.cifs</code>	if true then returns list of all fitted objects in cif.exceed
<code>...</code>	Additional arguments to lower level functions

**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

```
#####
## getting some rates to mimick
#####

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

cor.mat <- corM <- rbind(c(1.0, 0.6, 0.9), c(0.6, 1.0, 0.5), c(0.9, 0.5, 1.0))
rr <- simRecurrentII(1000,base4,cumhaz2=base4,death.cumhaz=dr,cens=2/5000)
rr <- count.history(rr)
dtable(rr,~death+status)

oo <- prob.exceedRecurrent(rr,1)
bplot(oo)

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

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

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

---

prt

*Prostate data set*


---

## Description

Prostate data set

**Source**

Simulated data

random.cif

*Random effects model for competing risks data***Description**

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

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

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**

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

data	a data.frame with the variables.
cause	specifies the causes related to the death times, the value cens.code is the censoring value.
cif2	specificies model for cause2 if different from cause1.
cause1	cause of first coordinate.
cause2	cause of second coordinate.
cens.code	specificies the code for the censoring if NULL then uses the one from the marginal cif model.
cens.model	specified which model to use for the ICPW, KM is Kaplan-Meier alternatively it may be "cox"
Nit	number of iterations for Newton-Raphson algorithm.
detail	if 0 no details are printed during iterations, if 1 details are given.
clusters	specifies the cluster structure.
theta	specifies starting values for the cross-odds-ratio parameters of the model.
theta.des	specifies a regression design for the cross-odds-ratio parameters.
sym	1 for symmetry 0 otherwise
step	specifies the step size for the Newton-Raphson algorithm.m
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" optimzner, 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<-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<-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
```

---

recreg

*Recurrent events regression with terminal event*

---

## Description

Fits Ghosh-Lin IPCW Cox-type model

## Usage

```
recreg(
  formula,
  data = data,
```



```

    cause = 1,
    death.code = c(2),
    cens.code = 0,
    cens.model = ~1,
    weights = NULL,
    offset = NULL,
    Gc = NULL,
    ...
)

```

**Arguments**

formula	formula with 'EventCens' outcome
data	data frame
cause	of interest
death.code	codes for death (terminating event)
cens.code	code of censoring (1 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-)
...	Additional arguments to lower level funtions

**Details**

For Cox type model :

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

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.

**Author(s)**

Thomas Scheike

**Examples**

```

## data with no ties
data(base1cumhaz)
data(base4cumhaz)
data(drcumhaz)
Lam1 <- base1cumhaz; Lam2 <- base4cumhaz; LamD <- drcumhaz
## simulates recurrent events of types 1 and 2 and with terminal event D and censoring
rr <- simRecurrentII(1000,Lam1,cumhaz2=Lam2,death.cumhaz=LamD,cens=3/5000)
rr <- count.history(rr)
rr$cens <- 0
nid <- max(rr$id)
rr$revnr <- revcumsumstrata(rep(1,nrow(rr)),rr$id-1,nid)
rr$x <- rnorm(nid)[rr$id]
rr$statusG <- rr$status
rr <- dtransform(rr,statusG=3,death==1)
dtable(rr,~statusG+status+death)
dcut(rr) <- gx~x

ll <- recreg(Event(start,stop,statusG)~x+cluster(id),data=rr,cause=1,death.code=3)
summary(ll)

## censoring stratified after quartiles of x
lls <- recreg(Event(start, stop, statusG)~x+cluster(id),data=rr,cause=1,
              death.code=3,cens.model=~strata(gx))
summary(lls)

```

---

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(recurrent, death, fixbeta = NULL, km = TRUE, ...)
```

### Arguments

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

### 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

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

### Examples

```
data(base1cumhaz)
data(base4cumhaz)
data(drcumhaz)
dr <- drcumhaz
base1 <- base1cumhaz
base4 <- base4cumhaz
rr <- simRecurrent(1000,base1,death.cumhaz=dr)
rr$x <- rnorm(nrow(rr))
rr$strata <- floor((rr$id-0.01)/500)
```

```

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

## marginal mean of expected number of recurrent events
out <- recurrentMarginal(xr,dr)
bplot(out,se=TRUE,ylab="marginal mean",col=2)

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

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

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

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

#####
### CIF #####
#####
### use of function to compute cumulative incidence (cif) with robust standard errors
data(bmt)
bmt$id <- 1:nrow(bmt)
xr <- phreg(Surv(time,cause==1)~cluster(id),data=bmt)
dr <- phreg(Surv(time,cause!=0)~cluster(id),data=bmt)

```

```
out <- recurrentMarginal(xr,dr,km=TRUE)
bplot(out,se=TRUE,ylab="cumulative incidence")
```

---

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 viewed. When times are given and beyond the last event time within a strata the curves are extrapolated using the estimates of cumulative incidence.

## Usage

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

## Arguments

x	phreg object
times	possible times for which to report restricted mean
covs	possible covariate for Cox model
...	Additional arguments to lower level functions

## Author(s)

Thomas Scheike

## Examples

```
data(bmt)
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)

## years.lost decomposed into causes
drm1 <- cif.yearslost(Surv(time,cause!=0)~cause+strata(tcell,platelet),data=bmt,times=10*(1:6))
summary(drm1)
```

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

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

or in the case of competing risks data

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

thus given years lost to cause.

**Usage**

```
resmeanIPCW(
  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",
  model = "exp",
  augmentation = NULL,
  h = NULL,
  MCAugment = NULL,
  Ydirect = 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
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
model	exp or linear
augmentation	to augment binomial regression
h	h for estimating equation
MCAugment	iid of h and censoring model
Ydirect	to bypass the construction of the response $Y=\min(T,\tau)$ and use this instead
...	Additional arguments to lower level funtions

### 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 that particular cause, thus

Based on binomial regresion IPCW response estimating equation:

$$X(\Delta(\min(T, t))/G_c(\min(T_i, t)) - \exp(X^T \beta)) = 0$$

for IPCW adjusted responses. Here

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

is indicator of being uncensored.

Can also solve the binomial regression IPCW response estimating equation:

$$h(X)X(\Delta(\min(T, t))/G_c(\min(T_i, t)) - \exp(X^T \beta)) = 0$$

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 :

$$X(\Delta(\min(T, t))Ydirect/G_c(\min(T_i, t)) - \exp(X^T \beta)) = 0$$

for IPCW adjusted responses.

Censoring model may depend on strata.

**Author(s)**

Thomas Scheike

**Examples**

```

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

### 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)

## 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(Surv(time,cause!=0)~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



---

simAalenFrailty	<i>Simulate from the Aalen Frailty model</i>
-----------------	--

---

**Description**

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

**Usage**

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

**Arguments**

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

**Author(s)**

Klaus K. Holst

---

simClaytonOakes      *Simulate from the Clayton-Oakes frailty model*

---

**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     *Simulate from the Clayton-Oakes frailty model*

---

### 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

---

 simMultistate

*Simulation of illness-death model*


---

## 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 functions

### 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(base1cumhaz)
data(base4cumhaz)
data(drcumhaz)
dr <- drcumhaz
dr2 <- drcumhaz
dr2[,2] <- 1.5*drcumhaz[,2]
base1 <- base1cumhaz
base4 <- base4cumhaz
cens <- rbind(c(0,0),c(2000,0.5),c(5110,3))

iddata <- simMultistate(100,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)
```

---

simRecurrent

*Simulation of recurrent events data based on cumulative hazards*


---

## Description

Simulation of recurrent events data based on cumulative hazards

## Usage

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

## Arguments

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

**Details**

Must give hazard of death and recurrent events. Possible with two event types and their dependence can be specified but the two recurrent events need to have the same random effect, simRecurrentII more flexible !

**Author(s)**

Thomas Scheike

**Examples**

```
#####
## getting some rates to mimick
#####

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

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

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

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

rr <- simRecurrent(1000,base1,death.cumhaz=dr,dependence=1,var.gamma=0.4)
### marginals do fit after input after integrating out
par(mfrow=c(2,2))
showfitsim(causes=1,rr,dr,base1,base1)
```

---

simRecurrentII

*Simulation of recurrent events data based on cumulative hazards II*


---

**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,
  gap.time = FALSE,
  max.recurrent = 100,
  dhaz = NULL,
  haz2 = NULL,
  dependence = 0,
  var.z = 0.22,
  cor.mat = NULL,
  cens = NULL,
  ...
)

```

**Arguments**

n	number of id's
cumhaz	cumulative hazard of recurrent events
cumhaz2	cumulative hazard of recurrent events of type 2
death.cumhaz	cumulative hazard of death
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
gap.time	if true simulates gap-times with specified cumulative hazard
max.recurrent	limits number recurrent events to 100
dhaz	rate for death hazard if it is extended to time-range of first event
haz2	rate of second cause if it is extended to time-range of first event
dependence	0:independence; 1:all share same random effect with variance var.z; 2:random effect exp(normal) with correlation structure from cor.mat; 3:additive gamma distributed random effects, $z1 = (z11 + z12)/2$ 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 functions



**Details**

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

**Author(s)**

Thomas Scheike

**Examples**

```
#####
## getting some rates to mimick
#####

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

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

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

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

---

simRecurrentTS

*Simulation of recurrent events data based on cumulative hazards:  
Two-stage model*

---

**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  $\text{death.cumhazard}$

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

Random effect for death  $Z.\text{death}=(Zd1+Zd2)$ ,  $Z1=(Zd1^{\nu1}) Z12$ ,  $Z2=(Zd2^{\nu2}) Z12^{\nu3}$

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

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

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<-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)
```

---

survival.iterative      *Survival model for multivariate survival data*

---

**Description**

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

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

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

For all models the standard errors do not reflect this uncertainty of the baseline estimates, and might therefore be a bit too small. To remedy this one can do bootstrapping.

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

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

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

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

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

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

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

, 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 over all clusters as it would be for the ACE model below. The `lamtot` parameter may be specified separately for some sets of the parameter is the `additive.gamma.sum` (`ags`) matrix is specified and then `lamtot` for the  $j$ 'th random effect is  $ags_j^T \lambda$ .

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

The DEFAULT parametrization uses the variances of the random effects

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

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

$$P(T > t|x, z) = \exp(-ZA(t) \exp(Z^t \theta))$$

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  $d$   $\lambda$  parameters with the ( $k$ ) underlying  $\theta$  parameters

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

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

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

**Usage**

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

**Arguments**

margsurv	Marginal model
data	data frame
method	Scoring method "nr", "nlminb", "optimize", "nlm"
Nit	Number of iterations
detail	Detail
clusters	Cluster variable
silent	Debug information
weights	Weights

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

**Author(s)**

Thomas Scheike

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  $\lambda_j$  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

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

with the inverse laplace of the gamma distribution with mean 1 and variance  $1/\text{lamtot}$ .

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  $d$   $\lambda$  parameters with the ( $k$ ) underlying  $\theta$  parameters

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

here using `theta.des` to specify these low-dimension association. Default is a diagonal matrix. 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 indeces 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 indeces of the pairs random.design rows are given as columns 3:4
pairs	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.
dim.theta	dimension of the theta parameter for pairs situation.
numDeriv.method	uses simple to speed up things and second derivative not so important.
additive.gamma.sum	for two.stage=0, this is specification of the lamtot in the models via a matrix that is multiplied onto the parameters theta (dimensions=(number random effects x number of theta parameters), when null then sums all parameters.
var.par	is 1 for the default parametrization with the variances of the random effects, var.par=0 specifies that the $\lambda_j$ 's are used as parameters.
no.opt	for not optimizng
...	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 <- 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)

```

---

survival.twostageCC     *Twostage survival model for multivariate survival data*

---

### Description

older extended version of survival.twostage with more options and different maximizer.

### Usage

```

survival.twostageCC(
  margsurv,
  data = parent.frame(),
  method = "nr",
  Nit = 60,
  detail = 0,
  clusters = NULL,
  silent = 1,
  weights = NULL,
  control = list(),
  theta = NULL,
  theta.des = NULL,
  var.link = 1,
  baseline.iid = 1,
  step = 0.5,
  model = "clayton.oakes",
  marginal.trunc = NULL,
  marginal.survival = NULL,
  marginal.status = NULL,
  strata = NULL,
  se.clusters = NULL,
  numDeriv = 0,
  random.design = NULL,
  pairs = NULL,
  dim.theta = NULL,
  numDeriv.method = "simple",
  additive.gamma.sum = NULL,
  var.par = 1,
  cr.models = NULL,

```

```

    case.control = 0,
    ascertained = 0,
    shut.up = 0
)

```

### Arguments

margsurv	Marginal model
data	data frame
method	Scoring method "nr", "nlminb", "optimize", "nlm"
Nit	Number of iterations
detail	Detail
clusters	Cluster variable
silent	Debug information
weights	Weights
control	Optimization arguments
theta	Starting values for variance components
theta.des	design for dependence parameters, when pairs are given the indeces 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.
step	Step size
model	model
marginal.trunc	marginal left truncation probabilities
marginal.survival	optional vector of marginal survival probabilities
marginal.status	related to marginal survival probabilities
strata	strata for fitting, see example
se.clusters	for clusters for se calculation with iid
numDeriv	to get numDeriv version of second derivative, otherwise uses sum of squared score
random.design	random effect design for additive gamma model, when pairs are given the indeces of the pairs random.design rows are given as columns 3:4
pairs	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.
dim.theta	dimension of the theta parameter for pairs situation.
numDeriv.method	uses simple to speed up things and second derivative not so important.
additive.gamma.sum	for two.stage=0, this is specification of the lamtot in the models via a matrix that is multiplied onto the parameters theta (dimensions=(number random effects x number of theta parameters), when null then sums all parameters.

var.par	is 1 for the default parametrization with the variances of the random effects, var.par=0 specifies that the $\lambda_j$ 's are used as parameters.
cr.models	competing risks models for two.stage=0, should be given as a list with models for each cause
case.control	assumes case control structure for "pairs" with second column being the probands, when this options is used the twostage model is profiled out via the paired estimating equations for the survival model.
ascertained	if the pair are sampled only when there is an event. This is in contrast to case.control sampling where a proband is given. This can be combined with control probands. Pair-call of twostage is needed and second column of pairs are the first jump time with an event for ascertained pairs, or time of control proband.
shut.up	to make the program more silent in the context of iterative procedures for case-control

**Author(s)**

Thomas Scheike

test.conc

*Concordance test Compares two concordance estimates***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))
```

---

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

*BMI data set*


---

### Description

BMI data set

### Format

Self-reported BMI-values on 11,411 subjects

tvparnr: twin id bmi: BMI (m/kg<sup>2</sup>) 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	<p>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:</p> <ul style="list-style-type: none"> <li>• "0" (or "sat"): Saturated model where twin 1 and twin 2 within each twin pair may have a different marginal distribution.</li> <li>• "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.</li> <li>• "2" (or "u", "eqmarg"): All individuals have the same marginals but a free correlation structure within MZ and DZ twins.</li> </ul> <p>The default value is an additive polygenic model type="ace".</p>
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, og 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	Threshold 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	<i>Twostage survival model fitted by pseudo MLE</i>
-------------	---

---

**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**

margsurv	Marginal model from phreg
data	data frame
theta	Starting values for variance components
theta.des	design for dependence parameters, when pairs are given this is could be a (pairs) x (number of parameters) x (max number random effects) matrix
var.link	Link function for variance if 1 then uses exp link
method	type of optimizer, default is Newton-Raphson "NR"
no.opt	to not optimize, for example to get score and iid for specific theta
weights	cluster specific weights, but given with length equivalent to data-set, weights for score equations
se.cluster	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 se.cluster argument.
...	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)
```

# Index

- \* **binomial**
    - binomial.twostage, 14
    - easy.binomial.twostage, 71
  - \* **data**
    - base1cumhaz, 7
    - base44cumhaz, 8
    - base4cumhaz, 8
    - dermalridges, 55
    - dermalridgesMZ, 55
    - drcumhaz, 62
    - ghaplos, 91
    - hapfreqs, 100
    - haploX, 103
    - mena, 115
    - migr, 116
    - multcif, 118
    - np, 118
    - prt, 125
    - ttpd, 160
    - twinbmi, 162
    - twinstut, 166
  - \* **models**
    - blocksample, 26
    - mets.options, 116
    - twinlm, 162
    - twinsim, 165
  - \* **package**
    - mets-package, 4
  - \* **regression**
    - binomial.twostage, 14
    - easy.binomial.twostage, 71
    - twinlm, 162
    - twinsim, 165
  - \* **survival**
    - cor.cif, 40
    - easy.survival.twostage, 75
    - EventSplit, 81
    - Grandom.cif, 96
    - LinSpline, 112
    - random.cif, 126
    - summary.cor, 147
    - survival.iterative, 148
    - survival.twostage, 152
    - survival.twostageCC, 157
    - twostageMLE, 166
  - \* **twostage**
    - easy.survival.twostage, 75
  - \* **utilities**
    - blocksample, 26
    - npc, 119
- 
- aalenfrailty, 5
  - aalenMets, 6
  - ace.family.design (npc), 119
  - addCums (simRecurrent), 142
  - alpha2kendall (npc), 119
  - alpha2spear (npc), 119
  - ascertained.pairs (npc), 119
- 
- back2timereg, 7
  - base1cumhaz, 7
  - base44cumhaz, 8
  - base4cumhaz, 8
  - basecumhaz (basehazplot.phreg), 8
  - basehazplot.phreg, 8
  - bicomprisk, 10
  - bicompriskData (bicomprisk), 10
  - BinAugmentCifstrata, 12
  - binomial.twostage, 14
  - binreg, 18
  - binregATE, 21
  - binregATEbin (binregATE), 21
  - binregCasewise, 23
  - binregt (binreg), 18
  - biprobbit, 24
  - blocksample, 26
  - Bootcovariancerecurrence (covarianceRecurrent), 46



- BootcovarianceRecurrenceS
  - (covarianceRecurrent), 46
- BootmediatorSurv (mediatorSurv), 113
- Bootphreg, 28
- bplot (basehazplot.phreg), 8
- bplotdFG (doubleFGR), 58
- bptwin, 29, 164
- casewise, 31
- casewise.bin (casewise.test), 32
- casewise.test, 32
- CCbinomial.twostage (npc), 119
- cif, 33
- cif.yearslost (resmean.phreg), 133
- cifreg, 34
- ClaytonOakes, 36
- cluster.index, 38
- coarse.clust (npc), 119
- coefmat (npc), 119
- concordance.cor (concordanceCor), 39
- concordanceCor, 39
- concordanceTwinACE (npc), 119
- concordanceTwostage (npc), 119
- cor.cif, 40
- corsim.prostate (npc), 119
- count.history, 45
- count.historyVar (count.history), 45
- countID (cluster.index), 38
- covarianceRecurrent, 46
- covarianceRecurrentS
  - (covarianceRecurrent), 46
- covfr (predict.phreg), 122
- covfridstrata (predict.phreg), 122
- covfridstrataCov (predict.phreg), 122
- covIntH1dM1IntH2dM2 (simRecurrent), 142
- cumContr (gofZ.phreg), 95
- cumsum2strata (predict.phreg), 122
- cumsumidstratasum (predict.phreg), 122
- cumsumidstratasumCov (predict.phreg), 122
- cumsumstrata (predict.phreg), 122
- cumsumstratasum (predict.phreg), 122
- daggr (daggregate), 47
- daggregate, 47
- Dbvn, 49
- dby, 50
- dby2 (dby), 50
- dby2<- (dby), 50
- dby<- (dby), 50
- dbyr (dby), 50
- dcor, 52
- dcount (dcor), 52
- dcut, 53
- dcut<- (dcut), 53
- ddrop (dcut), 53
- ddrop<- (dcut), 53
- dermalridges, 55
- dermalridgesMZ, 55
- deval (dcor), 52
- deval2 (dcor), 52
- dfactor (drelevel), 65
- dfactor<- (drelevel), 65
- dhead (dprint), 61
- diffstrata (cifreg), 34
- dInterval
  - (interval.logitsurv.discrete), 103
- divide.conquer, 56
- divide.conquer.timereg, 57
- dkeep (dcut), 53
- dkeep<- (dcut), 53
- dlag, 58
- dlag<- (dlag), 58
- dlev (drelevel), 65
- dlev<- (drelevel), 65
- dlevel (drelevel), 65
- dlevel<- (drelevel), 65
- dlevels (drelevel), 65
- dlist (dprint), 61
- dmean (dcor), 52
- dmeansd (dcor), 52
- dmvn (pmvn), 121
- dnames (dcut), 53
- dnames<- (dcut), 53
- dnumeric (drelevel), 65
- dnumeric<- (drelevel), 65
- doubleFGR, 58
- dprint, 61
- dquantile (dcor), 52
- drcumhaz, 62
- dreg, 62
- drelev (drelevel), 65
- drelev<- (drelevel), 65
- drelevel, 65
- drelevel<- (drelevel), 65
- drename (dcut), 53

- drename<- (dcut), 53
- dreshape (fast.reshape), 86
- drm (dcut), 53
- drm<- (dcut), 53
- drop.specials (npc), 119
- drop.strata (FG\_AugmentCifstrata), 89
- dsample (blocksample), 26
- dscalar (dcor), 52
- dsd (dcor), 52
- dsort, 67
- dsort2 (dsort), 67
- dsort<- (dsort), 67
- dspline, 68
- dspline<- (dspline), 68
- dstr (dcor), 52
- dsubset (dcor), 52
- dsum (dcor), 52
- dsummary (dcor), 52
- dtab (dtable), 69
- dtable, 69
- dtail (dprint), 61
- dtrans (dtransform), 71
- dtrans<- (dtransform), 71
- dtransform, 71
- dtransform<- (dtransform), 71
- dunique (dcut), 53
  
- easy.binomial.twostage, 71
- easy.survival.twostage, 75
- Effbinreg, 77
- EVaddGam, 79
- eventpois, 80
- EventSplit, 81
- extendCums (simMultistate), 140
  
- familycluster.index, 83
- familyclusterWithProbands.index, 84
- fast.approx, 85
- fast.cluster (npc), 119
- fast.pattern, 86
- fast.reshape, 86
- FastCoxPLstrataR (phregR), 120
- faster.reshape (npc), 119
- FG\_AugmentCifstrata, 89
- FGprediid (cifreg), 34
- folds (npc), 119
- force.same.cens (npc), 119
  
- ghaplos, 91
  
- GLprediid (recreg), 128
- gof.phreg, 91
- gofG.phreg, 92
- gofM.phreg, 93
- gofZ.phreg, 95
- Grandom.cif, 96
- grouptable (npc), 119
  
- hapfreqs, 100
- haplo.surv.discrete, 100
- haploX, 103
- headstrata (predict.phreg), 122
  
- iid.baseline.cifreg (cifreg), 34
- ilap (npc), 119
- indexstrata (fast.approx), 85
- indexstratarightR (cifreg), 34
- Interval (interval.logitsurv.discrete), 103
- interval.logitsurv.discrete, 103
- ipw, 104
- ipw2, 106
  
- jumptimes (npc), 119
  
- kendall.ClaytonOakes.twin.ace (npc), 119
- kendall.normal.twin.ace (npc), 119
- km, 108
- kumarsim (binregATE), 21
- kumarsimRCT (binregATE), 21
  
- lifecourse, 109
- lifetable (lifetable.matrix), 111
- lifetable.matrix, 111
- LinSpline, 112
- logitATE (binregATE), 21
- logitIPCW (binreg), 18
- logitIPCWATE (binregATE), 21
- logitSurv, 112
- loglikMVN (pmvn), 121
  
- make.pairwise.design (npc), 119
- matdoubleindex (predict.phreg), 122
- matplot.mets.twostage (npc), 119
- mdi (predict.phreg), 122
- mediatorSurv, 113
- medweight, 115
- mena, 115
- mets-package, 4
- mets.options, 116

- migr, 116
- mlogit, 117
- multcif, 118
- mystrata (cluster.index), 38
- mystrata2index (cluster.index), 38
  
- nonparcuminc (npc), 119
- normalATE (binregATE), 21
- np, 118
- npc, 119
  
- object.defined (npc), 119
- or.cif (cor.cif), 40
- or2prob (tetrachoric), 160
  
- p11.binomial.twostage.RV (npc), 119
- pairRisk (cluster.index), 38
- pbvn (pmvn), 121
- pcif (eventpois), 80
- phreg, 119
- phregR, 120
- piecewise.data (npc), 119
- piecewise.twostage (npc), 119
- plack.cif, 121
- plack.cif2 (plack.cif), 121
- plot.covariace.recurrent  
    (covarianceRecurrent), 46
- plotConfRegion (basehazplot.phreg), 8
- plotConfRegionSE (basehazplot.phreg), 8
- plotcr (npc), 119
- plotstrata (basehazplot.phreg), 8
- plotSurv (haplo.surv.discrete), 100
- pmvn, 121
- ppch (rpch), 136
- pred.cif.boot (Bootphreg), 28
- predict.phreg, 122
- predictdFG (doubleFGR), 58
- predictlogitSurv  
    (interval.logitsurv.discrete),  
    103
- predictmlogit (mlogit), 117
- predictPairPlack (npc), 119
- predictSurv (haplo.surv.discrete), 100
- print.casewise, 123
- prob.exceed.recurrent, 123
- prob.exceedBiRecurrent  
    (prob.exceed.recurrent), 123
- prob.exceedBiRecurrentStrata  
    (prob.exceed.recurrent), 123
- prob.exceedRecurrent  
    (prob.exceed.recurrent), 123
- prob.exceedRecurrentStrata  
    (prob.exceed.recurrent), 123
- procform (npc), 119
- procform3 (npc), 119
- procformdata (npc), 119
- prr, 125
  
- random.cif, 126
- randomDes (survival.twostage), 152
- readmargsurv (survival.twostage), 152
- readPhreg (phreg), 119
- recmarg (recurrentMarginal), 130
- recreg, 128
- recurrentMarginal, 130
- recurrentMarginalAIPCW  
    (recurrentMarginal), 130
- recurrentMarginalAIPCWdata  
    (recurrentMarginal), 130
- recurrentMarginalIPCW  
    (recurrentMarginal), 130
- resmean.phreg, 133
- resmeanIPCW, 134
- revcumsum (predict.phreg), 122
- revcumsum2strata (predict.phreg), 122
- revcumsum2stratafdN (predict.phreg), 122
- revcumsumidstratasum (predict.phreg),  
    122
- revcumsumidstratasumCov  
    (predict.phreg), 122
- revcumsumstrata (predict.phreg), 122
- revcumsumstratasum (predict.phreg), 122
- rmvn (pmvn), 121
- robust.basehaz.phreg (predict.phreg),  
    122
- robust.phreg (phreg), 119
- rpch, 136
- rr.cif (cor.cif), 40
  
- scalecumhaz (recreg), 128
- scoreMVN (pmvn), 121
- setup.cif (FG\_AugmentCifstrata), 89
- showfitsim (simRecurrent), 142
- sim (npc), 119
- simAalenFrailty, 137
- simbinClaytonOakes.family.ace (npc), 119
- simbinClaytonOakes.pairs (npc), 119
- simbinClaytonOakes.twin.ace (npc), 119

simBinFam (npc), 119  
 simBinFam2 (npc), 119  
 simBinPlack (npc), 119  
 simClaytonOakes, 138  
 simClaytonOakes.family.ace (npc), 119  
 simClaytonOakes.twin.ace (npc), 119  
 simClaytonOakesLam (simClaytonOakes),  
     138  
 simClaytonOakesWei, 139  
 simCompete.simple (npc), 119  
 simCompete.twin.ace (npc), 119  
 simCox (npc), 119  
 simFrailty.simple (npc), 119  
 simlogitSurvd  
     (interval.logitsurv.discrete),  
     103  
 simMultistate, 140  
 simnordic (npc), 119  
 simRecurrent, 142  
 simRecurrentGamma (simRecurrent), 142  
 simRecurrentII, 143  
 simRecurrentTS, 145  
 simSurvFam (npc), 119  
 simTTP (haplo.surv.discrete), 100  
 simul.cifs (FG\_AugmentCifstrata), 89  
 slope.process (casewise.test), 32  
 squareintHdM (simRecurrent), 142  
 strataAugment (recreg), 128  
 strataC (FG\_AugmentCifstrata), 89  
 summary.cor, 147  
 summaryTimeobject  
     (prob.exceed.recurrent), 123  
 sumstrata (predict.phreg), 122  
 surv.boxarea (npc), 119  
 survival.iterative, 148  
 survival.twostage, 152  
 survival.twostageCC, 157  
  
 tailstrata (predict.phreg), 122  
 test.conc, 159  
 tetrachoric, 160  
 tie.breaker (recurrentMarginal), 130  
 ttpd, 160  
 twin.clustertrunc, 161  
 twin.polygen.design (npc), 119  
 twinbmi, 162  
 twinlm, 31, 162, 166  
 twinlm.strata, 31, 164  
 twinlm.time, 31, 164  
  
 twinlm.time (bptwin), 29  
 twinsim, 31, 164, 165  
 twinstut, 166  
 twostage.aalen (survival.twostage), 152  
 twostage.cox.aalen (survival.twostage),  
     152  
 twostage.coxph (survival.twostage), 152  
 twostage.phreg (survival.twostage), 152  
 twostageMLE, 166  
  
 vecAllStrata (cifreg), 34