Package: mets (via r-universe)

October 28, 2026

Type Package

Title Analysis of Multivariate Event Times

Version 1.3.12

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

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

Description Implementation of various statistical models for multivariate event history data <doi:10.1007/s10985-013-9244-x>. Including multivariate cumulative incidence models <doi:10.1002/sim.6016>, and bivariate random effects probit models (Liability models) <doi:10.1016/j.csda.2015.01.014>. Modern methods for survival analysis, including regression modelling (Cox, Fine-Gray, Ghosh-Lin, Binomial regression) with fast computation of influence functions.

License Apache License (== 2.0)

LazyLoad yes

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

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

Depends R (>= 3.5)

Imports Rcpp, RcppArmadillo (>= 0.12.8.0), lava (>= 1.9.2), methods, mvtnorm, numDeriv, splines, survival (>= 2.43-1), timereg (>= 1.9.4)

Suggests cmprsk, icenReg, KernSmooth, knitr, optimx, prodlim, riskRegression, rmarkdown, tinytest (>= 1.4.1), ucminf

VignetteBuilder knitr

ByteCompile yes

LinkingTo Rcpp, RcppArmadillo (>= 0.12.8.0), mvtnorm

Encoding UTF-8

Config/roxygen2/version 8.0.0

NeedsCompilation yes

Repository https://cran.r-universe.dev

Date/Publication 2026-07-30 06:46:34 UTC

RemoteUrl https://github.com/cran/mets

RemoteRef HEAD

RemoteSha 4177a01298fce3e0019d94ff073c1f5b4a558c21

Contents

aalenMets . . . 5
ACTG175 . . . 6
bicomprisk . . . 7
binomial_twostage . . . 10
binreg . . . 14
binreg_IPTW . . . 18
binregATE . . . 20
binregCasewise . . . 23
binregG . . . 24
binregRatio . . . 26
binregStrata . . . 29
binregTSR . . . 32
biprobit . . . 35
blocksample . . . 37
bmt . . . 39
bptwin . . . 39
brier_binreg . . . 41
calgb8923 . . . 45
casewise . . . 46
casewise_bin . . . 47
cif . . . 48
cif_yearslost . . . 49
cifreg . . . 50
cifregFG . . . 53
ClaytonOakes . . . 53
cluster_index . . . 55
coarse_clust . . . 56
concordanceCor . . . 57
cor_cif . . . 58
count_history . . . 62
CPH_HPN_CRBSI . . . 64
cumoddsreg . . . 64
daggregate . . . 65
Dbvn . . . 67
dby . . . 67
dcor . . . 69
dcut . . . 70
dermalridges . . . 73

dermalridgesMZ . . . 73
diabetes . . . 74
divide_conquer . . . 74
dlag . . . 75
dprint . . . 76
dreg . . . 77
drelevel . . . 80
drop.specials . . . 82
dsort . . . 82
dspline . . . 83
dtable . . . 84
dtransform . . . 86
estimate.binregCV . . . 86
Event . . . 87
event_split . . . 88
event_split2 . . . 89
eventpois . . . 90
extendCums . . . 91
familycluster_index . . . 92
familyclusterWithProbands_index . . . 93
fast.approx . . . 94
fast.cluster . . . 95
fast.pattern . . . 95
fast.reshape . . . 96
faster.reshape . . . 98
folds . . . 99
force.same.cens . . . 99
gof.phreg . . . 100
gofM_phreg . . . 102
gofZ_phreg . . . 104
Grandom_cif . . . 106
grouptable . . . 109
haplo . . . 110
haplo_surv_discrete . . . 111
hfactioncpx12 . . . 114
IC.binreg . . . 114
IC.phreg . . . 116
iidBaseline . . . 117
ilap . . . 118
interval_logitsurv_discrete . . . 118
ipw . . . 121
ipw2 . . . 122
jumptimes . . . 124
km . . . 125
lifecourse . . . 126
lifetable.matrix . . . 128
LinSpline . . . 129
logitSurv . . . 129

mediatorSurv . . . 130
medweight . . . 132
melanoma . . . 132
mena . . . 133
mets.options . . . 134
migr . . . 134
mlogit . . . 135
multcif . . . 137
np . . . 137
p11_binomial_twostage_RV . . . 137
phreg . . . 141
phreg_IPTW . . . 142
phreg_rct . . . 144
phreg_weibull . . . 146
plack_cif . . . 148
plot.phreg . . . 148
plot_twin . . . 149
pmvn . . . 151
predict.mlogit . . . 151
predict.phreg . . . 152
print.casewise . . . 154
print.summary_binregCV_multi . . . 154
prob_exceed_recurrent . . . 155
prt . . . 157
random_cif . . . 157
ratioATE . . . 160
rchaz . . . 161
rchazl . . . 162
rcrisk . . . 163
recreg . . . 164
recregIPCW . . . 167
recurrent_marginal . . . 169
resmean_phreg . . . 171
resmeanATE . . . 172
resmeanIPCW . . . 174
robust.basehaz.phreg . . . 176
rr_cif . . . 177
rweibullcox . . . 181
sim_cif . . . 182
sim_ClaytonOakes . . . 184
sim_ClaytonOakesWei . . . 185
sim_GLcox . . . 186
sim_multistate . . . 187
sim_multistateII . . . 189
sim_phreg . . . 190
sim_phregs . . . 192
sim_recurrent . . . 193
sim_recurrent_ts . . . 195

sim_recurrentII 197
sim_recurrentTS 200
strata-numeric 202
summary.binregCV 203
summary.binregCV_list 204
summary.cor 204
summaryGLM 206
summaryTimeobject 206
surv_boxarea 207
survival-helpers 208
survival_twostage 210
survivalG 216
test_casewise 218
test_conc 220
test_logrankRecurrent 221
test_marginalMean 223
tetrachoric 225
tie_breaker 226
TRACE 228
ttpd 229
twinbmi 229
twinlm 230
twinsim 232
twinstut 233
twostageMLE 234
twostageREC 236
WA_recurrent 237
WA_reg 240

Index **242**

`aalenMets` *Fast Additive Hazards Model with Robust Standard Errors*

Description

Fits a fast Lin-Ying additive hazards model with a possibly stratified baseline. Robust variance is the default variance estimate in the summary.

Usage

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

Arguments

`formula`	Formula with a 'Surv' outcome (similar to `coxph`).
`data`	Data frame.
`no.baseline`	Logical; if `TRUE`, fits the model without baseline hazard estimation.
`...`	Additional arguments passed to `phreg`.

Details

Influence functions (IID) follow the numerical order of the given cluster variable. Ordering by `$id` aligns the IID terms with the dataset order.

Value

An object of class `"aalenMets"` (extends `"phreg"`) containing:

`coef`	Estimated coefficients.
`var`	Robust variance-covariance matrix.
`iid`	Influence functions.
`intZHZ`	Integrated ZHZ matrix.
`gamma`	Coefficient estimates.

Author(s)

Thomas Scheike

Examples

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

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

ACTG175 *ACTG175, block randomized study from speff2trial package*

Description

Data from speff2trial

Format

Randomized study

Source

Hammer et al. 1996, speff2trial package.

Examples

```
data(ACTG175)
```

`bicomprisk` *Estimation of Concordance in Bivariate Competing Risks Data*

Description

Estimates the bivariate cumulative incidence function (concordance) for paired data (e.g., twins, family members) in the presence of competing risks. The function handles both the IPCW (Inverse Probability of Censoring Weighting) estimator and the Aalen-Johansen estimator (via `prodlim`).

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 an `Event` object on the left-hand side. The right-hand side specifies the covariate structure, including `strata()` for grouping (e.g., MZ/DZ) and `id()` for pairing.
`data`	Data frame containing the variables.
`cause`	Vector of cause codes for which to estimate the bivariate cumulative incidence (default `c(1,1)`).

cens	Censoring code (default 0).
causes	Vector of all possible causes (optional, inferred from data if missing).
indiv	Variable indicating individual within a pair (optional, inferred from id).
strata	Variable for stratification (optional, can be specified in formula).
id	Clustering variable (pair ID). Required.
num	Variable for numbering individuals within pairs (optional, auto-generated if missing).
max.clust	Maximum number of clusters to use for IID decomposition in timereg::comp.risk. If NULL, uses all clusters. Useful for large datasets to speed up computation.
marg	Optional marginal cumulative incidence object (from comp.risk) to compute standard errors for same-cluster comparisons in subsequent casewise.test().
se.clusters	Vector of cluster indices or column name in data for standard error calculation. Defaults to the id variable.
wname	Name of an additional weight variable for paired competing risks data.
prodlim	Logical; if TRUE, uses the prodlim (Aalen-Johansen) estimator instead of the IPCW estimator based on comp.risk. These are equivalent in the absence of covariates.
messages	Control amount of output (0 = silent, 1 = messages).
model	Type of competing risk model for comp.risk (default "fg" for Fine-Gray).
return.data	If 1, returns the reshaped data; if 2, returns only the data; otherwise returns the model.
uniform	Logical; if TRUE, computes uniform standard errors based on resampling.
conservative	Logical; if TRUE, uses conservative standard errors (recommended for large datasets).
resample.iid	Logical; if TRUE, returns IID residual processes for further computations.
...	Additional arguments passed to timereg::comp.risk.

Details

The concordance function $C(t)$ is defined as the probability that both members of a pair experience the event of interest by time t:

$$C(t) = P(T_1 \leq t, T_2 \leq t, \epsilon_1 = k, \epsilon_2 = k)$$

where T_i is the event time and ϵ_i is the cause of failure for individual i.

The function supports:

- Stratified analysis (e.g., by zygosity in twin studies).
- Clustering for robust standard errors.
- Both IPCW and Aalen-Johansen estimation methods.
- Resampling for uniform standard errors.
- IID decomposition for further inference (e.g., casewise concordance tests).

Value

An object of class "bicomprisk" (or "bicomprisk.strata" if stratified) containing:

model	List of fitted models for each stratum (or a single model).
strata	Names of strata (if applicable).
N	Number of strata (if applicable).
time	Event times.
P1	Bivariate cumulative incidence estimates.
se.P1	Standard errors of the estimates.
P1.iid	IID decomposition (if resample.iid=TRUE).
clusters	Cluster assignments (if marg or se.clusters provided).

Author(s)

Thomas Scheike, Klaus K. Holst

References

Scheike, T. H.; Holst, K. K. & Hjelmborg, J. B. (2014). Estimating twin concordance for bivariate competing risks twin data. Statistics in Medicine, 33, 1193-1204.

See Also

test_casewise, casewise

Examples

```
library("timereg")

## Simulated data example
prt <- sim_nordic_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
prtl <-  prt[!prt$truncated,]
prt2 <- ipw2(prtl,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$cause2==1)*1
prt22$timel <- pmax(prt22$time1,prt22$time2)
ipwc <- timereg::comp.risk(Event(timel,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)
```

`binomial_twostage` *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 algoritm uses a compososite likelihood based on all pairwise bivariate models.*

Description

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

Usage

```
binomial_twostage(
  margbin,
  data = parent.frame(),
  method = "nr",
  detail = 0,
  clusters = NULL,
  silent = 1,
  weights = NULL,
  theta = NULL,
  theta.des = NULL,
  var.link = 0,
  var.par = 1,
  var.func = NULL,
  iid = 1,
  notaylor = 1,
  model = "plackett",
  marginal.p = NULL,
  beta.iid = NULL,
  Dbeta.iid = NULL,
  strata = NULL,
  max.clust = NULL,
  se.clusters = NULL,
```

```
  numDeriv = 0,
  random.design = NULL,
  pairs = NULL,
  dim.theta = NULL,
  additive.gamma.sum = NULL,
  pair.ascertained = 0,
  case.control = 0,
  no.opt = FALSE,
  twostage = 1,
  beta = NULL,
  ...
)
```

Arguments

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 λ_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 pairwise composite score, useful for case-control sampling when marginal is known.
dim.theta	dimension of theta when pairs and pairs specific design is given. That is when pairs has 6 columns.
additive.gamma.sum	this is specification of the lamtot in the models via a matrix that is multiplied onto the parameters theta (dimensions=(number random effects x number of theta parameters), when null then sums all parameters. Default is a matrix of 1's
pair.ascertained	if pairs are sampled only when there are events in the pair i.e. Y1+Y2>=1.
case.control	if data is case control data for pair call, and here 2nd column of pairs are probands (cases or controls)
no.opt	for not optimizing
twostage	default twostage=1, to fit MLE use twostage=0
beta	is starting value for beta for MLE version
...	for NR of lava

Details

The reported standard errors are based on a cluster corrected score equations from the pairwise likelihoods assuming that the marginals are known. This gives correct standard errors in the case of the Odds-Ratio model (Plackett distribution) for dependence, but incorrect standard errors for the Clayton-Oakes types model (that is also called "gamma"-frailty). For the additive gamma version of the standard errors are adjusted for the uncertainty in the marginal models via an iid deomposition using the iid() function of lava. For the clayton oakes model that is not speicifed 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 indpendent, and that the marginal survival functions are on logistic form

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$$\lambda = theta.des\theta$$

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

Author(s)

Thomas Scheike

References

Two-stage binomial modelling

Examples

```
library(mets)  ## do not delete this line
set.seed(123)  ## do not delete this line
data(twinstut)
twinstut0 <- subset(twinstut, tvparnr<4000)
twinstut <- twinstut0
twinstut$binstut <- (twinstut$stutter=="yes")*1
theta.des <- model.matrix( ~-1+factor(zyg),data=twinstut)
margbin <- glm(binstut~factor(sex)+age,data=twinstut,family=binomial())
bin <- binomial_twostage(margbin,data=twinstut,var.link=1,
         clusters=twinstut$tvparnr,theta.des=theta.des,detail=0)
summary(bin)

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

```
summary(bina)

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

### use of clayton oakes binomial additive gamma model
############################################################
 ## Reduce Ex.Timings
data <- sim_binClaytonOakes_family_ace(1000,2,1,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 <- sim_binClaytonOakes_twin_ace(1000,2,1,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)
```

`binreg` *Binomial Regression for Censored Competing Risks Data*

Description

Fits a binomial regression model for a specific time point in the presence of right-censored data and competing risks. This function implements the Inverse Probability of Censoring Weighted (IPCW) estimating equation approach.

Usage

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

Arguments

formula	A formula object specifying the outcome and covariates. The outcome must be an Event object (Event(time, cause)).
data	A data frame containing the variables in the formula.
cause	Numeric vector or scalar indicating the cause of interest for the competing risks.
time	Numeric scalar indicating the time point of interest for the cumulative incidence.
beta	Optional numeric vector of starting values for the coefficients. Defaults to zeros.
type	Character string. Either "I" (classic binomial regression) or "II" (adds augmentation term).
offset	Optional numeric vector of offsets for the linear predictor.
weights	Optional numeric vector of weights for the score equations.
cens.weights	Optional numeric vector of pre-calculated censoring weights. If NULL, they are estimated internally.
cens.model	A formula specifying the censoring model. Defaults to ~+1. Can include strata (e.g., ~strata(group)).
se	Logical. If TRUE, computes standard errors based on IPCW influence functions.
kaplan.meier	Logical. If TRUE, uses Kaplan-Meier estimator for IPCW weights; otherwise uses exponential baseline.
cens.code	Numeric code representing censored observations in the status variable. Defaults to 0.

no.opt	Logical. If TRUE, optimization is skipped and starting values are used.
method	Character string. Optimization method: "nr" (Newton-Raphson) or "nlm".
augmentation	Optional numeric vector for additional augmentation terms.
outcome	Character string. Outcome type: "cif" (Cumulative Incidence Function), "rmst" (Restricted Mean Survival Time), or "rmtl" (Restricted Mean Time Lost).
model	Character string. Link function: "default" (auto-selects based on outcome), "logit", "exp", or "lin" (identity).
Ydirect	Optional numeric vector. If provided, this outcome is used instead of constructing one from outcome. Useful for custom IPCW adjustments.
...	Additional arguments passed to lower-level optimization functions.

Details

The model estimates the probability:

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

Based on binomial regresion IPCW response estimating equation:

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

where

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

is IPCW adjustment of the response

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

. Two types of estimators are available:

- type="I": Solves the standard IPCW estimating equation.
- type="II": Adds a censoring augmentation term for efficiency gains, solving

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

.

Alternatively, logitIPCW performs a standard logistic regression with IPCW weights applied directly to the likelihood. Thus solving

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

a standard logistic regression with IPCW weights.

The variance estimation is based on squared influence functions, with options for naive variance (assuming known censoring) and robust variance (accounting for censoring model estimation).

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

Value

An object of class "binreg" containing coefficients, variance-covariance matrix, influence functions (iid), and model details.

References

- Blanche PF, Holt A, Scheike T (2022). "On logistic regression with right censored data, with or without competing risks, and its use for estimating treatment effects." *Lifetime data analysis*, 29, 441–482.
- Scheike TH, Zhang MJ, Gerds TA (2008). "Predicting cumulative incidence probability by direct binomial regression." *Biometrika*, 95(1), 205–220.

Outcome definition

The observed outcome Y_i constructed from `outcome` depends on whether competing risks are present. Competing risks are detected automatically: the data are considered to have competing risks when causes other than those specified in `cause` are observed (i.e. `any(!(Causes %in% cause))`, where `Causes` are all non-censoring event codes found in the data).

No competing risks (all observed events belong to `cause`):

- `"cif"`: $Y_i = I(T_i \leq t, \epsilon_i = 1)$. Cumulative incidence.
- `"rmst"`: $Y_i = \min(T_i, t)$. Restricted mean survival time.
- `"rmtl"`: $Y_i = t - \min(T_i, t)$. Restricted mean time lost.

Competing risks (causes beyond `cause` are observed in the data):

- `"cif"`: $Y_i = I(T_i \leq t, \epsilon_i \in \texttt{cause})$. Cumulative incidence for the cause(s) of interest.
- `"rmst"` or `"rmtl"`: $Y_i = I(\epsilon_i \in \texttt{cause})(t - \min(T_i, t))$. Cause-specific years lost, accumulated only for subjects who experience an event in `cause`.

The default link function (`model = "default"`) is `"logit"` for `"cif"` and `"exp"` for `"rmst"` and `"rmtl"`. If `Ydirect` is supplied, outcome construction is bypassed entirely.

Author(s)

Thomas Scheike

Examples

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

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

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

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

########################################
```

```
### risk-ratio of different causes #######
##########################################
data(bmt)
bmt$id <- 1:nrow(bmt)
bmt$status <- bmt$cause
bmt$strata <- 1
bmtdob <- bmt
bmtdob$strata <-2
bmtdob <- dtransform(bmtdob,status=1,cause==2)
bmtdob <- dtransform(bmtdob,status=2,cause==1)
###
bmtdob <- rbind(bmt,bmtdob)
dtable(bmtdob,cause+status~strata)

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

cifs1 <- binreg(Event(time,cause)~tcell+platelet+age,bmt,cause=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))
summary(cifdob)
head(iid(cifdob))

newdata <- data.frame(tcell=1,platelet=1,age=0,strata=1:2,id=1)
riskratio <- function(p) {
  cifdob$coef <- p
  p <- predict(cifdob,newdata,se=0)
  return(p[1]/p[2])
}
lava::estimate(cifdob,f=riskratio)

predict(cifdob,newdata)
(p1 <- predict(cifs1,newdata))
(p2 <- predict(cifs2,newdata))
p1[1,1]/p2[1,1]
```

`binreg_IPTW` *IPTW logistic regression, Inverse Probabibilty of Treatment Weighted binreg*

Description

Fits logistic regression model with treatment weights

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

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

Usage

```
binreg_IPTW(
  formula,
  data,
  treat.model = NULL,
  weights = NULL,
  estpr = 1,
  pi0 = 0.5,
  ...
)
```

Arguments

formula	for binreg
data	data-frame for estimation
treat.model	propensity score model (binary or multinomial)
weights	may be given, and then uses weights*w(A) as the weights
estpr	to estimate propensity scores and get infuence function contribution to uncertainty
pi0	fixed simple weights
...	arguments for binreg call

Details

Also works with cluster argument.

Author(s)

Thomas Scheike

Examples

```
data(bmt)
dfactor(bmt) <- platelet.f~platelet
## logistic modelling of cumulative incidence
gg <- binreg_IPTW(Event(time,cause)~platelet.f+age,bmt,
```

```
        treat.model=platelet.f~age,time=30)
summary(gg)
head(iid(gg))

## logistic modelling
gg <- binreg_IPTW(tcell~platelet.f+age,bmt,
        treat.model=platelet.f~age)
summary(gg)
```

---

binregATE *Average Treatment Effect for Censored Competing Risks Data using Binomial Regression*

Description

Estimates the average treatment effect (ATE) $E(Y(1) - Y(0))$ for censored competing risks data using binomial regression with Inverse Probability of Censoring Weighting (IPCW).

Usage

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

Arguments

formula	A formula object specifying the outcome and covariates (see coxph). The first covariate should be the treatment variable coded as a factor.
data	A data frame containing the variables in the formula.
cause	Numeric scalar indicating the cause of interest for competing risks.
time	Numeric scalar indicating the time point of interest.
beta	Optional numeric vector of starting values for the coefficients.
treat.model	A formula specifying the logistic treatment model given covariates (e.g., treatment ~ covariate1 + covariate2).
cens.model	A formula specifying the censoring model. Only stratified Cox models without covariates are supported (e.g., ~ strata(group)).
offset	Optional numeric vector of offsets for the partial likelihood.
weights	Optional numeric vector of weights for the score equations.
cens.weights	Optional numeric vector of pre-calculated censoring weights. If NULL, weights are estimated internally.
se	Logical. If TRUE, computes standard errors with IPCW adjustment. If FALSE, assumes IPCW weights are known.
type	Character string. Either "I" (classic binomial regression) or "II" (adds augmentation term for efficiency).
kaplan.meier	Logical. If TRUE, uses Kaplan-Meier for IPCW weights; otherwise uses $\exp(-\text{Baseline})$.
cens.code	Numeric code representing censored observations in the status variable.
no.opt	Logical. If TRUE, optimization is skipped and starting values are used.
method	Character string. Optimization method: "nr" (Newton-Raphson) or "nlm".
augmentation	Optional numeric vector for augmenting binomial regression.
outcome	Character string. Outcome type: "cif" (Cumulative Incidence Function, $F(t\|X)$), "rmst" (Restricted Mean Survival Time, $E(\min(T,t)\|X)$), or "rmtl" (Restricted Mean Time Lost, $E(I(\epsilon = \text{cause})(t - \min(T,t))\|X)$).
model	Character string. Link function for the outcome model: "exp" or "lin" (identity). For "cif", "logit" is typically used.
Ydirect	Optional numeric vector. Use this outcome Y with IPCW version instead of constructing one from outcome.
typeATE	Character string. Either "II" (censoring augmentation of the estimating equation) or "I" (standard).
...	Additional arguments passed to lower-level functions (e.g., binreg that fits the outcome model).

Details

Under standard causal assumptions, the ATE can be estimated. These assumptions include:

- **Consistency**: The observed outcome equals the potential outcome under the observed treatment.

- **Ignorability**: $(Y(1), Y(0)) \perp A|X$ (treatment assignment is independent of potential outcomes given covariates).
- **Positivity**: All treatment levels have non-zero probability given covariates.

The first covariate in the competing risks regression model must be the treatment variable, which should be coded as a factor. If the factor has more than two levels, multinomial logistic regression (`mlogit`) is used for propensity score modeling. In the absence of censoring, this reduces to ordinary logistic regression.

The ATE is estimated using standard doubly robust estimating equations that are IPCW-censoring adjusted. As an alternative to binomial regression, `logitIPCWATE` provides an IPCW-weighted version of standard logistic regression.

When `typeATE = "II"`, the estimating equation is augmented with:

$$(A/\pi(X)) \int E(O(t)|T \geq t, S(X))/G_c(t, S(X))d\hat{M}_c(s)$$

when estimating the mean outcome for the treated group.

Value

An object of class `c("binreg", "ATE")` containing:

`coef`	Estimated coefficients from the outcome model.
`riskDR`	Double-robust marginal risk estimates for each treatment level.
`riskG`	G-formula marginal risk estimates for each treatment level.
`difriskDR`	Difference in risks (ATE) using double-robust estimator.
`difriskG`	Difference in risks (ATE) using G-formula estimator.
`riskDR.iid`, `riskG.iid`	Influence functions for marginal risk estimates.
`var`, `var.riskDR`, `var.riskG`	Variance-covariance matrices.
`se.coef`, `se.riskDR`, `se.riskG`	Standard errors.

References

- Blanche PF, Holt A, Scheike T (2022). "On logistic regression with right censored data, with or without competing risks, and its use for estimating treatment effects." *Lifetime Data Analysis*, 29, 441–482.

Author(s)

Thomas Scheike

See Also

`binreg`, `logitIPCWATE`, `logitATE`, `binregG`

[kumarsim()] [kumarsimRCT()]

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)
head(brs$riskDR.iid)
head(brs$riskG.iid)

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)
head(brs$riskDR.iid)
head(brs$riskG.iid)
```

---

binregCasewise *Estimate Casewise Concordance Using Binomial Regression*

Description

Estimates the casewise concordance based on concordance and marginal estimates obtained from binreg objects. Uses cluster-based IID for standard errors, which are often better than those from casewise (which can be conservative).

Usage

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

Arguments

concbreg	Concordance object from binreg.
margbreg	Marginal estimate object from binreg.
zygs	Order of zygosity for estimation (e.g., c("DZ","MZ")).
newdata	Data frame to give instead of zygs.
...	Arguments passed to estimate.

Value

A list containing:

coef	Exponentiated coefficients (ratios).
logcoef	Log-scale coefficients and standard errors.

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)

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

`binregG` *G-Estimator for Binomial Regression Model (Standardized Estimates)*

Description

Computes the G-estimator (G-formula) for standardized risk estimates based on a fitted `binreg` object. The G-estimator standardizes predictions over the covariate distribution in the data:

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

Usage

```
binregG(x, data, Avalues = NULL, varname = NULL)
```

Arguments

`x`	An object of class `"binreg"` obtained from `binreg()` or `logitIPCW()`.
`data`	A data frame containing the covariates to be used for averaging the risk estimates. This should ideally be the same data used to fit the model, or a representative sample.
`Avalues`	Numeric or factor vector specifying the values of the first covariate (A) for which to compute standardized risks. • If the first covariate is a factor and `Avalues` is `NULL`, all levels of the factor are used. • If the first covariate is continuous, `Avalues` must be provided.
`varname`	Optional character string specifying the name of the variable to be treated as the treatment/exposure variable. If `NULL`, the first variable in the model formula is used.

Details

This function assumes that the first covariate in the original model formula represents the treatment or exposure variable (A). It calculates the marginal risk for specified values of A by averaging the conditional predictions over the observed covariate distribution Z.

The function returns influence functions for these risk estimates, allowing for the computation of standard errors and confidence intervals.

If the first covariate is a factor, contrasts between all levels are computed automatically. If it is continuous, specific values must be provided via `Avalues`.

Value

An object of class `"survivalG"` containing:

`risk`	A table of standardized risk estimates for each value of `Avalues`.
`risk.iid`	Influence functions for the standardized risk estimates.
`difference`	Pairwise differences in risks between levels of `A`.
`ratio`	Risk ratios between levels of `A`.
`vcov`	Variance-covariance matrix of the risk estimates.
`model`	The link function used in the original model.

References

- Blanche PF, Holt A, Scheike T (2022). "On logistic regression with right censored data, with or without competing risks, and its use for estimating treatment effects." *Lifetime Data Analysis*, 29, 441–482.

Author(s)

Thomas Scheike

See Also

`binreg`, `binregATE`

Examples

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

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

binregRatio *Percentage of Years Lost Due to a Cause Regression*

Description

Estimates the percentage of the restricted mean time lost (RMTL) that is attributable to a specific cause and models how covariates affect this percentage using IPCW regression.

Usage

```
binregRatio(
  formula,
  data,
  cause = 1,
  time = NULL,
  beta = NULL,
  type = c("III", "II", "I"),
  offset = NULL,
  weights = NULL,
  cens.weights = NULL,
  cens.model = ~+1,
  se = TRUE,
  relative.to.causes = NULL,
  kaplan.meier = TRUE,
  cens.code = 0,
  no.opt = FALSE,
  method = "nr",
  augmentation = NULL,
  outcome = c("rmtl", "cif"),
  model = c("logit", "exp", "lin"),
  Ydirect = NULL,
  ...
)
```

Arguments

formula	Formula with an outcome (see coxph). The first covariate on the RHS is typically the treatment or group indicator. Can include cluster(id).
data	Data frame containing the variables.
cause	Numeric code of the cause of interest.
time	Time point t for the analysis. Required.
beta	Starting values for optimization (default NULL, uses zeros).
type	Type of estimator: "I" (IPCW only), "II" (IPCW + augmentation), or "III" (IPCW + complex augmentation). Default is "III".
offset	Offsets for the partial likelihood.

`weights`	Weights for the score equations.
`cens.weights`	External censoring weights (if provided, `cens.model` is ignored).
`cens.model`	Formula for the censoring model (default `~+1`, stratified KM). Can include `strata()` for stratified censoring.
`se`	Logical; if TRUE, computes standard errors based on IPCW (default TRUE).
`relative.to.causes`	If not NULL, compares the RMTL of the specified `cause` to the RMTL of the causes in this vector (the denominator becomes the sum of these causes).
`kaplan.meier`	Logical; if TRUE, uses Kaplan-Meier for IPCW weights; if FALSE, uses $\exp(-\text{cumulative hazard})$.
`cens.code`	Censoring code (default 0).
`no.opt`	Logical; if TRUE, skips optimization and uses `beta` directly.
`method`	Optimization method: `"nr"` (Newton-Raphson) or `"nlm"`.
`augmentation`	Initial augmentation term (used for type "II" and "III").
`outcome`	Outcome type: `"rmtl"` (years lost) or `"cif"` (cumulative incidence).
`model`	Link function: `"logit"` (default), `"exp"`, or `"lin"`.
`Ydirect`	Matrix with two columns (numerator, denominator) to use directly as the response.
`...`	Additional arguments passed to lower-level functions.

Details

Let the total years lost be $Y = t - \min(T, t)$ and the years lost due to cause 1 be $Y_1 = I(\epsilon = 1)(t - \min(T, t))$. The function models the ratio:

$$\text{logit}\left(\frac{E(Y_1|X)}{E(Y|X)}\right) = X^T\beta$$

Estimation is based on a binomial regression IPCW response estimating equation:

$$X\left(\Delta^{\text{ipcw}}(t)\left(Y \cdot \text{expit}(X^T\beta) - Y_1\right)\right) = 0$$

where $\Delta^{\text{ipcw}}(t) = I(\min(t, T) < C)/G_c(\min(t, T))$ is the IPCW adjustment.

The function supports three types of estimators:

- `"I"`: Classical outcome IPCW regression (no augmentation).
- `"II"`: Adds a censoring augmentation term $X \int E(Z(t)|T > s)/G_c(s)d\hat{M}_c$ to improve efficiency (requires an initial estimate of β).
- `"III"`: Adds a more complex augmentation term separating the expectations of Y and Y_1 for further efficiency gains.

The variance is based on the squared influence functions (IID). A "naive" variance (assuming known censoring) is also provided for comparison.

Value

An object of class "binreg" and "ratio" containing:

coef	Coefficient estimates.
se.coef	Standard errors.
var	Variance-covariance matrix.
iid	Influence function decomposition (with censoring adjustment).
iidI	Influence function without censoring adjustment.
naive.var	Variance assuming known censoring.
time	Time point used.
cause	Cause of interest.
Causes	Set of causes considered in the denominator.
Yipcw	IPCW-adjusted response matrix.
coefI, varI	Results from the initial (type "I") fit.
augmentation	Final augmentation term used.

Author(s)

Thomas Scheike

References

Scheike, T. & Tanaka, S. (2025). Restricted mean time lost ratio regression: Percentage of restricted mean time lost due to specific cause. WIP.

See Also

resmeanIPCW, binreg

Examples

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

rmtl30 <- rmstIPCW(Event(time,cause!=0)~platelet+tcell+age, bmt, time=30, cause=1, outcome="rmtl")
rmtl301 <- rmstIPCW(Event(time,cause)~platelet+tcell+age, bmt, time=30, cause=1)
rmtl302 <- rmstIPCW(Event(time,cause)~platelet+tcell+age, bmt, time=30, cause=2)

estimate(rmtl30)
estimate(rmtl301)
estimate(rmtl302)

## Percentage of total RMTL due to cause 1
rmtlratioI <- rmtlRatio(Event(time,cause)~platelet+tcell+age, bmt, time=30, cause=1)
summary(rmtlratioI)

newdata <- data.frame(platelet=1, tcell=1, age=1)
pp <- predict(rmtlratioI, newdata)
```

```
pp

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

---

binregStrata *Stratified Binomial Regression with cross-validation option*

Description

Fits a stratified binomial regression model for the cumulative incidence function (CIF), restricted mean survival time (RMST), or restricted mean time lost (RMTL), using inverse probability of censoring weighting (IPCW). Each stratum may either be fit on its own observations or, when `cv.fold = TRUE`, on the complement of its observations (i.e. a leave-one-stratum-out / cross-validation style fit). Standard errors are computed via an iid decomposition that accounts for the estimation of the censoring distribution.

Usage

```
binregStrata(
  formula,
  data,
  cause = 1,
  time = NULL,
  beta = NULL,
  type = c("II", "I"),
  offset = NULL,
  weights = NULL,
  cens.weights = NULL,
  cens.model = ~+1,
  se = TRUE,
  kaplan.meier = TRUE,
  cens.code = 0,
  no.opt = FALSE,
  method = "nr",
  augmentation = NULL,
  outcome = c("cif", "rmst", "rmtl"),
  model = c("default", "logit", "exp", "lin"),
  Ydirect = NULL,
  strata = NULL,
  cv.fold = FALSE,
  low.memory = FALSE,
  ...
)
```

Arguments

formula	A formula with a Surv or Event object on the left-hand side, and covariates, optional cluster(), strata(), offset(), and weights() terms on the right-hand side.
data	A data.frame containing the variables in formula.
cause	Value(s) of the status variable that identify the event of interest (cause 1 by default).
time	The fixed time-point at which the CIF, RMST, or RMTL is evaluated. Required.
beta	Optional starting values for the regression coefficients. Either a vector (recycled across strata) or a matrix with one row per stratum.
type	Character; either "II" (default) or "I". When "II", an additional augmentation term and an extra component of the censoring-related iid decomposition are computed.
offset	Optional offset term; can also be specified via the formula.
weights	Optional case weights; can also be specified via the formula.
cens.weights	Optional pre-computed inverse probability of censoring weights. If supplied, the internal censoring model fit is skipped and se is forced to FALSE.
cens.model	A formula for the censoring model (right-hand side only, fitted via phreg). Defaults to the Kaplan-Meier (~+1).
se	Logical. If TRUE (default), standard errors are computed, including the contribution from estimating the censoring distribution.
kaplan.meier	Logical. If TRUE (default), uses the Kaplan-Meier estimator for censoring weights when the censoring model has no covariates; otherwise a Cox-type estimator is used internally when covariates are present.
cens.code	The code in the status variable denoting censoring (default 0).
no.opt	Logical. If TRUE, no optimization is performed and the object is evaluated at the supplied beta.
method	Optimization method. Currently only "nr" (Newton-Raphson via lava::NR) is supported.
augmentation	Optional augmentation term(s) added to the score equations, either a vector (recycled across strata) or a matrix with one row per stratum.
outcome	Character; the outcome scale to model: "cif" (default), "rmst", or "rmtl".
model	Character; the link function: "default" (chooses "logit" for CIF and "exp" for RMST/RMTL), "logit", "exp", or "lin".
Ydirect	Optional, directly supplied response vector, overriding the response otherwise constructed from outcome.
strata	Optional numeric vector defining strata, used when strata are not specified via strata() in the formula.
cv.fold	Logical. If TRUE, for each stratum s the regression coefficients are estimated using all observations *not* in stratum s (a cross-validation-style fit), rather than the observations within stratum s (the default, FALSE).

low.memory — Logical. If TRUE only saves key quanties, coef, var, iid and some design structures.

... — Additional arguments, currently passed as optimizer control settings (e.g. tol, stepsize) to the Newton-Raphson routine.

Details

When cv.fold = FALSE (the standard case), the coefficients for stratum s are estimated using only observations with strata == s. When cv.fold = TRUE, the coefficients for stratum s are instead estimated using observations with strata != s, i.e. the complement; this also affects the construction of the censoring-related iid terms (MGCiid) and the per-stratum subject-level influence functions used for the robust variance.

Censoring weights are computed via a Cox model (phreg) on cens.model, evaluated at pmin(exit, time), unless cens.weights is supplied directly. When se = TRUE, the additional variability coming from the estimation of the censoring distribution is incorporated into the standard errors via a martingale-type iid decomposition (MGCiid).

Value

An object of class c("binregStrata","binreg"), a list with components including:

coef — Estimated regression coefficients (stacked across strata).

iid — Subject-level iid decomposition of the coefficients, adjusted for censoring estimation when se = TRUE.

var, robvar — Robust (sandwich) variance-covariance matrix.

se.coef, se.robust — Standard errors derived from var.

hessian, ihessian — Per-stratum Hessians and their (pseudo-)inverses.

coef_list — Per-stratum coefficient vectors.

time, cause, cens.code, model, outcome — Echoed arguments.

Y, Yipcw, IPCW — The constructed response, IPCW-weighted response, and IPCW weights.

cens.weights, formC, cens.strata, cens.nstrata — Censoring model weights and related information (when internally estimated).

nstrata, strata_orig, strata_call, cv.fold — Stratification information and whether the cross-validation fold scheme was used.

n, nevent, ncluster, nid, id, name.id — Sample size and identifier information.

call, design — The matched call and the design object.

See Also

phreg, binreg

Examples

```
## Not run:
library(mets)
data(bmt)
out <- binregStrata(Event(time, cause) ~ tcell + strata(platelet),
                    data = bmt, cause = 1, time = 50)
summary(out)

## End(Not run)
```

---

binregTSR *Two-Stage Randomization for Survival or Competing Risks Data*

Description

Estimates the average treatment effect $E(Y(i,j))$ of treatment regime (i,j) under two-stage randomization. The estimator can be augmented using information from both randomizations and dynamic censoring augmentation to improve efficiency.

Usage

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

```
  Ydirect = NULL,
  return.dataw = 0,
  pi0 = 0.5,
  pi1 = 0.5,
  cens.time.fixed = 1,
  outcome.iid = 1,
  meanCs = 0,
  ...
)
```

Arguments

formula	Formula with outcome (see coxph), typically Event(entry,time,status)~+1+cluster(id).
data	Data frame containing all variables.
cause	Cause of interest for competing risks (default 1).
time	Time point for estimation.
cens.code	Censoring code (default 0).
response.code	Code of status indicating response at which 2nd randomization occurs.
augmentR0	Covariates for augmentation model of the first randomization.
treat.model0	Logistic treatment model for the first randomization.
augmentR1	Covariates for augmentation model of the second randomization.
treat.model1	Logistic treatment model for the second randomization.
augmentC	Covariates for censoring augmentation model.
cens.model	Stratification for censoring model based on observed covariates.
estpr	Logical; estimate randomization probabilities using model (default TRUE).
response.name	Name of response variable (reads from treat.model1 if NULL).
offset	Not implemented.
weights	Not implemented.
cens.weights	Can be provided externally.
beta	Starting values for optimization.
kaplan.meier	Logical; use Kaplan-Meier for censoring weights rather than exp cumulative hazard.
no.opt	Not implemented.
method	Not implemented.
augmentation	Not implemented.
outcome	Outcome type: "cif" (cumulative incidence), "rmst" (restricted mean survival time), or "rmst-cause" (restricted mean time lost for cause).
model	Not implemented, uses linear regression for augmentation.
Ydirect	Use this Y instead of outcome constructed inside the program.
return.dataw	Logical; return weighted data for all treatment regimes.
pi0	Known randomization probabilities for first randomization.

pi1	Known randomization probabilities for second randomization.
cens.time.fixed	Logical; use time-dependent weights for censoring estimation.
outcome.iid	Logical; get iid contribution from outcome model.
meanCs	Logical; indicates censoring augmentation is centered by CensAugment.times/n.
...	Additional arguments to lower-level functions.

Details

The method solves the estimating equation:

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

where:

- $AUG_0=\frac{A_0(i)-\pi_0(i)}{\pi_0(i)}X_0\gamma_0$ uses covariates from augmentR0
- $AUG_1=\frac{A_0(i)}{\pi_0(i)}\frac{A_1(j)-\pi_1(j)}{\pi_1(j)}X_1\gamma_1$ uses covariates from augmentR1
- $AUG_C=\int_0^t\gamma_c(s)^T(e(s)-\bar{e}(s))\frac{1}{G_c(s)}dM_c(s)$ is the censoring augmentation

Standard errors are estimated using the influence function of all estimators, enabling tests of differences to be computed subsequently. The method handles both survival data and competing risks data, and supports multiple treatment levels.

Value

An object of class "binregTSR" containing:

riskG	Simple estimator results (coefficient, SE).
riskG0	First randomization augmentation results.
riskG1	Second randomization augmentation results.
riskG01	Both randomizations augmentation results.
riskG.iid	Influence functions for all estimators.
varG	Variance-covariance matrices.
MGc	Censoring martingale contributions.
CensAugment.times	Censoring augmentation terms.
dynCens.coef	Dynamic censoring coefficients.
dataW	Weighted data (if return.dataw=TRUE).

Author(s)

Thomas Scheike

References

Scheike, T. H. (2024). Two-stage randomization analysis for survival data. mets package documentation.

See Also

binreg, phreg_rct

Examples

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

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

---

biprobit *Bivariate Probit model*

Description

Bivariate Probit model

Usage

```
biprobit(
  x,
  data,
  id,
  rho = ~1,
  num = NULL,
  strata = NULL,
  eqmarg = TRUE,
  indep = FALSE,
  weights = NULL,
  weights.fun = function(x) ifelse(any(x <= 0), 0, max(x)),
  randomeffect = FALSE,
  vcov = "robust",
  pairs.only = FALSE,
  allmarg = !is.null(weights),
  control = list(trace = 0),
  messages = 1,
  constrain = NULL,
  table = pairs.only,
```

```
  p = NULL,
  ...
)
```

Arguments

x	formula (or vector)
data	data.frame
id	The name of the column in the dataset containing the cluster id-variable.
rho	Formula specifying the regression model for the dependence parameter
num	Optional name of order variable
strata	Strata
eqmarg	If TRUE same marginals are assumed (exchangeable)
indep	Independence
weights	Weights
weights.fun	Function defining the bivariate weight in each cluster
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")
predict(a, newdata=lava::Expand(prt, zyg=c("MZ")))
## b <- biprobit(cancer~1+zyg, ~1+zyg, data=prt0, id="id",pairs.only=TRUE)
## predict(b,newdata=lava::Expand(prt,zyg=c("MZ","DZ")))

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

```
a <- biprobit(y~1+x,rho=~1+x,data=dd,id="id")
summary(a, mean.contrast=c(1,.5), cor.contrast=c(1,.5))
with(predict(a,data.frame(x=seq(-1,1,by=.1))), plot(p00~x,type="l"))

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

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

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




## Time
## Not run:
    a <- biprobit.time(cancer~1, rho=~1+zyg, id="id", data=prt, eqmarg=TRUE,
                       cens.formula=Surv(time,status==0)~1,
                       breaks=seq(75,100,by=3),fix.censweights=TRUE)

    a <- mets:::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 wether to sample with replacement
...	additional arguments to lower level functions

Details

Original id is stored in the attribute 'id'

Value

data.frame

Author(s)

Klaus K. Holst

Examples

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

## Not run:
blocksample(data.table::data.table(d),1e6,~id)

## End(Not run)


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

bmt *The Bone Marrow Transplant Data*

Description

Bone marrow transplant data with 408 rows and 5 columns.

Format

The data has 408 rows and 5 columns.

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

time a numeric vector. Survival time.

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

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

age a numeric vector code. Age of patient, scaled and centered ((age-35)/15).

Source

Simulated data

References

NN

Examples

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

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 dyzogitic 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") & tvparnr<5e3)
        ),
        id="tvparnr",zyg="zyg",DZ="dz",type="ae")
summary(b0)
```

---

brier_binreg *Cross-validated Brier score for competing risks, RMST and RMTL regression using stratified leave-fold-out fits (*binregStrata*)*

Description

Fits candidate models via binregStrata with cv.fold = TRUE and evaluates each using apparent (in-sample) and *k*-fold cross-validated IPCW Brier scores relative to a null model. Supports all outcome types available in binreg/binregStrata: cause-specific cumulative incidence (competing risks), restricted mean survival time (RMST), and restricted mean time lost (RMTL).

Usage

```
brier_binreg(
  formula,
  data,
  time,
  fold = 5,
  rhs = NULL,
  rhs0 = ~+1,
  cens.model = ~+1,
  brier.cens.model = ~+1,
  brier.args = list(),
  cv.split.index = NULL,
  ...
)
```

Arguments

formula	Two-sided formula of the form `Event(time, status) ~ covariates + cluster(id)` or `Event(time, status, cause) ~ covariates + cluster(id)`. The `cluster(id)` term is optional; if absent an internal identifier `id__` is added automatically assuming independent observations.
data	A `data.frame` containing all variables in `formula` and `rhs`.
time	Scalar or numeric vector of evaluation time points. When a vector is supplied the function loops over time points reusing the same fold structure and returns a named list of `binregCV` objects with class `"binregCV_list"`.
fold	Integer. Number of cross-validation folds / strata (default 5).
rhs	Named list of one-sided formulas specifying candidate models, e.g. `list(small = ~Z1, full = ~Z1 + Z2)`. If `NULL` (default), the right-hand side of `formula` (minus any `cluster()` term) is used as a single candidate named `"model1"`.
rhs0	One-sided formula for the null (intercept-only) model (default `~+1`).
cens.model	Censoring model passed to `binregStrata` for the CV fold fits (default `~+1`). Since folds are handled internally by `binregStrata`'s `strata/cv.fold` mechanism, this does *not* need to include `strata(fold)`.
brier.cens.model	Censoring model passed to `resmeanIPCW` when computing Brier scores. Default `~+1` gives Kaplan-Meier weights. Supply a stratified model, e.g. `~strata(Z1, Z2)`, for covariate-adjusted IPCW weights.
brier.args	Named list of additional arguments forwarded to every `resmeanIPCW` call (beyond `formula`, `data`, `time`, `Ydirect`, `cens.model`, and `model` which are set internally). Typically empty.
cv.split.index	List from `folds(n, fold)` or a vector of fold indices (1:fold) for each observation, to fix the cross-validation folds. `NULL` (default) generates folds internally via `folds(n, fold)`.
...	Additional arguments passed to every `binreg` / `binregStrata` call, e.g. `cause = 2`, `outcome = "rmst"`, or `offset`.

Details

Cross-validation is delegated directly to binregStrata via its strata and cv.fold = TRUE arguments: each stratum (fold) gets its own coefficient vector β_s, estimated on the complement of that fold. Out-of-fold predictions are obtained by routing each observation to the β_s corresponding to its own fold, which was trained on the other folds.

The fold membership is stored in an internal fold column (1-based) in data and passed to binregStrata as a 0-based strata vector. Censoring weights for the underlying competing-risks model are estimated according to cens.model as usual; since strata already define the folds, cens.model need not include strata(fold) explicitly.

Brier scores are computed on the **linear scale** via resmeanIPCW(..., model = "lin") and then log-transformed using lava::estimate(..., f = log) with the delta method. All stored coefficients, standard errors, and influence functions are therefore on the **log scale**. Use summary(fit, transform = exp) or estimate(fit, f = exp) to back-transform to the original Brier scale.

The delta-method correction for the cross-validated influence function accounts for the block structure of binregStrata's influence function ($n_{id} \times (\texttt{nstrata} \times p)$, one p-column block per stratum). For each stratum s, only observations belonging to fold s have a non-zero derivative of their out-of-fold prediction with respect to β_s, since β_s is exactly the coefficient used to predict fold s. The full gradient is therefore block-sparse, with one non-zero p-length block per stratum, and the correction reduces to a single matrix product against iid(outff_fit).

Left-truncation (delayed entry) is not supported and will raise an error. Specifically, Event(entry, time, status) is rejected; use Event(time, status) or Event(time, status, cause).

Value

An object of class "binregCV" (single time point) or "binregCV_list" (multiple time points), with the following structure:

null List with coef, se, and iid for the null model. coef and se are named vectors with entries bbrier (apparent, log scale) and bbrierCV (cross-validated, log scale). iid contains the influence-function matrix for bbrierCV with delta-method correction for the stratified CV prediction step and the log transform.

models Named list, one element per entry in rhs. Each element contains coef (brier, brierCV, log scale), se, dcoef (differences from null: log Brier ratios), sed, and iid (influence-function matrix for brierCV, log scale, with CV delta-method correction).

iid_mat Matrix of dimension $n \times (1 + \texttt{length(rhs)})$ with columns null and one per entry of rhs, each the log-scale CV Brier influence function for the corresponding model. Used by estimate.binregCV for joint inference.

cv.split.index List of fold index vectors used.

rhs, rhs0 Processed candidate and null formulas (cluster terms removed).

model Link function read from the binreg fit: "logit", "exp", or "lin".

call, time, fold, cens.model, brier.cens.model Call and settings for printing and downstream use.

Outcome types

The function supports all outcome types provided by `binreg`/`binregStrata`:

Competing risks CIF Default; use `Event(time, status, cause)` and pass `cause` via `...`. The Brier score measures squared error for the cause-specific CIF.

RMST Pass `outcome = "rmst"` via `...`. The Brier score measures squared error for the restricted mean survival time up to `time`.

RMTL Pass `outcome = "rmtl"` via `...`. The Brier score measures squared error for the restricted mean time lost.

Scale and inference

All output is on the log scale. To obtain Brier scores on the original scale with confidence intervals, use:

```
summary(fit, transform = exp)
estimate(fit, f = exp)
```

Differences on the log scale correspond to log Brier ratios; exponentiate to obtain Brier ratios. For absolute differences on the original scale, compute contrasts after `estimate(fit, f = exp)`.

See Also

`estimate.binregCV` for joint inference across models and time points; `summary.binregCV` and `print.summary_binregCV_multi` for formatted output; `binregStrata` for the underlying stratified regression model; `resmeanIPCW` for IPCW mean estimation; `estimate` for delta-method inference.

Examples

```
## Not run:
library(mets)
data(bmt)
bmt$id <- seq_len(nrow(bmt))

## --- competing risks CIF/Survival (default) ---
fit <- brier_binreg(
  Event(time, cause) ~ tcell + platelet + age + cluster(id),
  data = bmt, time = 50,
  rhs = list(small = ~age, full = ~tcell + platelet + age)
)
summary(fit)                        ## log scale
summary(fit, transform = exp)       ## Brier scale

## multiple time points
fitm <- brier_binreg(
  Event(time, cause) ~ tcell + platelet + age + cluster(id),
  data = bmt, time = c(50, 60),
  rhs = list(small = ~age, full = ~tcell + platelet + age)
)
```

```
summary(fitm, transform = exp)

## joint inference across models via lava::estimate
e <- estimate(fitm)
e
estimate(e, f = exp)              ## back-transform

## stratified IPCW weights for Brier score
fit_strat <- brier_binreg(
  Event(time, cause) ~ tcell + platelet + age + cluster(id),
  data = bmt, time = 50,
  rhs = list(small = ~age, full = ~tcell + platelet + age),
  brier.cens.model = ~strata(tcell)
)

## --- RMST --- survival case only
fit_rmst <- brier_binreg(
  Event(time, cause) ~ tcell + platelet + age + cluster(id),
  data = bmt, time = 50, cause = 1:2,
  rhs    = list(small = ~age, full = ~tcell + platelet + age),
  outcome = "rmst"
)
summary(fit_rmst, transform = exp)

## --- RMTL --- competing risks or survival (RMTL)
fit_rmtl <- brier_binreg(
  Event(time, cause) ~ tcell + platelet + age + cluster(id),
  data = bmt, time = 50,
  rhs    = list(small = ~age, full = ~tcell + platelet + age),
  outcome = "rmtl"
)
summary(fit_rmtl, transform = exp)

## End(Not run)
```

`calgb8923` *CALGB 8923, twostage randomization SMART design*

Description

Data from CALGB 8923

Format

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

Source

https://github.com/ycchao/code_Joint_model_SMART

Examples

```
data(calgb8923)
```

`casewise` *Estimate Casewise Concordance from prodlim Objects*

Description

Estimates the casewise concordance based on concordance and marginal estimates derived from `prodlim` objects. Unlike `test_casewise`, this function does not perform hypothesis testing but focuses on estimation and plotting.

Usage

```
casewise(conc, marg, cause.marg)
```

Arguments

`conc`	Concordance object from `prodlim` (output of `bicomprisk` with `prodlim=TRUE`).
`marg`	Marginal cumulative incidence object from `prodlim` (output of `prodlim`).
`cause.marg`	Specifies which cause should be used for the marginal CIF based on the `Event` object.

Value

An object of class `"casewise"` containing:

`casewise`	Matrix with time, casewise concordance, and standard errors.
`marg`	Matrix with time, marginal CIF, and standard errors.
`concordance`	Matrix with time, concordance, and standard errors.
`timer`	Time points used.
`P1`, `se.P1`	Extracted concordance values and SEs.

Author(s)

Thomas Scheike

See Also

`test_casewise`, `bicomprisk`

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"))
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_bin` *Casewise Concordance from Concordant/Discordant Counts*

Description

Computes casewise concordance probability and confidence interval from counts of concordant and discordant pairs using a binomial GLM.

Usage

```
casewise_bin(nc, nd)
```

Arguments

`nc`	number of concordant pairs.
`nd`	number of discordant pairs.

Value

A list with `p.casewise` (estimated probability) and `ci.casewise` (confidence interval).

`cif` *Cumulative Incidence with Robust Standard Errors*

Description

Computes cumulative incidence functions with robust standard errors.

Usage

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

Arguments

`formula`	Formula with 'Event' outcome and `strata` (only!).
`data`	Data frame.
`cause`	Cause of interest (default is `NULL`, which looks at all causes).
`cens.code`	Censoring code (default is `"0"`).
`death.code`	Alternative to `cens.code`; specifies codes of death.
`...`	Additional arguments passed to lower-level functions.

Value

An object of class `"cif"` (extends `"phreg"`) containing:

`cumhaz`	Cumulative incidence estimates.
`se.cumhaz`	Standard errors.
`cause`	Cause of interest.

Author(s)

Thomas Scheike

Examples

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

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

`cif_yearslost` *Restricted Mean Time Lost for Competing Risks*

Description

Computes the Restricted Mean Time Lost (RMTL) for competing risks based on the integrated Aalen-Johansen estimator.

Usage

```
cif_yearslost(formula, data = data, cens.code = 0, times = NULL, ...)
```

Arguments

`formula`	Formula for `phreg` object with `strata` to indicate strata, or `+1` if no strata.
`data`	Data frame for calculations.
`cens.code`	Censoring code (needed to separate event codes from censorings).
`times`	Possible times for which to report restricted mean. Summary displays estimates for these times.
`...`	Additional arguments passed to `phreg`.

Details

A set of time points can be given to be returned in the summary, but the function computes years-lost for all event times and can be plotted/viewed. The RMTL for a specific time-point can also be computed using the `rmstIPCW` function.

Value

An object of class `"resmean_phreg"` containing:

`cumhaz`	Matrix of cumulative hazards (years lost).
`se.cumhaz`	Standard errors.
`intF1times`	Years lost at specified times.
`causes`	Vector of cause codes.

Author(s)

Thomas Scheike

Examples

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

## Years lost decomposed into causes
drm1 <- cif_yearslost(Event(time, cause) ~ strata(tcell, platelet), data = bmt, times = c(40, 50))
par(mfrow = c(1, 2))
plot(drm1, cause = 1, se = 1)
plot(drm1, cause = 2, se = 1)
summary(drm1)
estimate(drm1, cause = 1)
estimate(drm1, cause = 2)

## Comparing populations
drm1 <- cif_yearslost(Event(time, cause) ~ strata(tcell, platelet), data = bmt, times = 40)
summary(drm1, contrast = list(1:4))
e1 <- estimate(drm1)
estimate(e1, rbind(c(1, -1, 0, 0)))
```

---

cifreg *Cumulative Incidence Function (CIF) Regression*

Description

Fits a regression model for the cumulative incidence function (CIF) in the presence of competing risks. Supports two link functions:

- `propodds=1` (default): Logistic link model (logit of CIF), providing Odds Ratio (OR) interpretations.
- `propodds=NULL`: Fine-Gray (cloglog) regression model, providing subdistribution hazard ratio interpretations.

Usage

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

Arguments

formula	Formula with an 'Event' outcome.
data	Data frame containing the variables.
propodds	Logical; if 1 (default), fits the logit link model. If NULL, fits the Fine-Gray model.
cause	Cause of interest (default is 1).
cens.code	Code for censoring (default is 0).
no.codes	Event codes to be ignored when identifying competing causes (useful for administrative censoring).
death.code	Codes for death (terminal events). If NULL, defaults to all remaining codes (excluding cause, cens.code, and no.codes).
...	Additional arguments passed to recreg.

Details

For the Fine-Gray model, the score equations are:

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

summed over clusters and returned as iid.naive (multiplied by the inverse of the second derivative). Here,

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

,

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

, and

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

.

The full influence function (IID decomposition) for the beta coefficients includes a censoring term:

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

which is returned as the iid component.

For the logistic link model, standard errors may be slightly underestimated because uncertainty from the recursive baseline estimation is not fully accounted for. For smaller datasets, it is recommended to use the prop.odds.subdist function from the **timereg** package (which uses more efficient weights) or to bootstrap the standard errors.

Value

An object of class "cifreg" (extending "phreg") containing:

coef	Estimated coefficients.
var	Robust variance-covariance matrix.
iid	Influence functions for the coefficients.
cumhaz	Cumulative incidence estimates.
propodds	Indicator of the link function used.

Author(s)

Thomas Scheike

See Also

cifregFG, recreg, gofFG

Examples

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

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

## approximate standard errors
por <-mets:::predict.phreg(or,nd)
plot(por,se=1)

## Fine-Gray model
fg=cifregFG(Event(time,cause)~tcell+platelet+age,data=bmt,cause=1)
summary(fg)
##fg=recreg(Event(time,cause)~tcell+platelet+age,data=bmt,cause=1,death.code=2)
##summary(fg)
plot(fg)
nd <- data.frame(tcell=c(1,0),platelet=0,age=0)
pfg <- predict(fg,nd,se=1)
plot(pfg,se=1)

## bt <- iidBaseline(fg,time=30)
## bt <- IIDrecreg(fg$cox.prep,fg,time=30)

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

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

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

pfg1

---

cifregFG *Fine-Gray Cumulative Incidence Function Regression*

Description

Convenience wrapper for cifreg that specifically fits the Fine-Gray model by setting propodds=NULL. This provides subdistribution hazard ratio interpretations.

Usage

```
cifregFG(formula, data, propodds = NULL, ...)
```

Arguments

formula	Formula with an 'Event' outcome.
data	Data frame.
propodds	Set to NULL to fit the Fine-Gray model (default behavior of this function).
...	Additional arguments passed to cifreg.

Value

An object of class "cifreg" (extending "phreg") with propodds=NULL.

Author(s)

Thomas Scheike

See Also

cifreg

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(sim_ClaytonOakes(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 <- sim_ClaytonOakes(500,4,2,1,stoptime=2,left=0)
```

```
d2$z <- rep(1,nrow(d2)); d2$z[d2$cluster%in%sample(d2$cluster,100)] <- 0
## Marginal=Cox Proportional Hazards model:
## ts <- ClaytonOakes(survival::Surv(time,status)~timereg::prop(x)+cluster(~1,cluster),
##                   data=d2,type="two.stage")
## Marginal=Aalens additive model:
## ts2 <- ClaytonOakes(survival::Surv(time,status)~x+cluster(~1,cluster),
##                    data=d2,type="two.stage")
## Marginal=Piecewise constant:
e2 <- ClaytonOakes(survival::Surv(time,status)~x+cluster(~-1+factor(z),cluster),
                   cuts=c(0,0.5,1,2),data=d2)
e2

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

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

`cluster_index` *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)
```

---

coarse_clust *Coarsen Cluster Identifiers*

Description

Reduces the number of unique clusters by binning into quantile groups.

Usage

```
coarse_clust(clusters, max.clust = 100)
```

Arguments

clusters	vector of cluster identifiers.
max.clust	maximum number of coarsened clusters.

Value

Integer vector of coarsened cluster indices (0-based).

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	Specfifies wich model that is considered if object not given.
coefs	Specfifies 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. Hjelmborg, Statistics in Medicine 2014, 1193-1204

Estimating Twin Pair Concordance for Age of Onset. Thomas H. Scheike, Jacob V B Hjelmborg, 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

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

and

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

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

Usage

```
cor_cif(
  cif,
  data,
  cause = NULL,
  times = NULL,
  cause1 = 1,
  cause2 = 1,
  cens.code = NULL,
  cens.model = "KM",
  Nit = 40,
  detail = 0,
  clusters = NULL,
  theta = NULL,
```

```
  theta.des = NULL,
  step = 1,
  sym = 0,
  weights = NULL,
  par.func = NULL,
  dpar.func = NULL,
  dimpar = NULL,
  score.method = "nlminb",
  same.cens = FALSE,
  censoring.weights = NULL,
  silent = 1,
  ...
)
```

Arguments

cif	a model object from the timereg::comp.risk function with the marginal cumulative incidence of cause1, i.e., the event of interest, and whose odds the comparision is compared to the conditional odds given cause2
data	a data.frame with the variables.
cause	specifies the causes related to the death times, the value cens.code is the censoring value. When missing it comes from marginal cif.
times	time-vector that specifies the times used for the estimating euqations for the cross-odds-ratio estimation.
cause1	specificies the cause considered.
cause2	specificies the cause that is conditioned on.
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.
step	specifies the step size for the Newton-Raphson algorithm.
sym	specifies if symmetry is used in the model.
weights	weights for estimating equations.
par.func	parfunc
dpar.func	dparfunc
dimpar	dimpar
score.method	"nlminb", can also use "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

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

This measure is numerically more stabile than the COR measure, and is symetric in i,k.

The RR dependence measure is given by

$$RR(i,k) = \log(\frac{P(T_1 \leq t, cause_1 = i, T_2 \leq t, cause_2 = k)}{P(T_1 \leq t, cause_1 = i) P(T_2 \leq t, cause_2 = k)}$$

This measure is numerically more stabile than the COR measure, and is symetric in i,k.

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

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

Value

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

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

Author(s)

Thomas Scheike

References

Cross odds ratio Modelling of dependence for Multivariate Competing Risks Data, Scheike and Sun (2012), Biostatistics.

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

Examples

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

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

out1 <- cor_cif(add,data=multcif,cause1=1,cause2=1)
summary(out1)

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

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

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

parfunc <- function(par,t,pardes)
{
par <- pardes %*% c(par[1],par[2]) +
       pardes %*% c( par[3]*(t-60)/12,par[4]*(t-60)/12)
par
}
head(parfunc(c(0.1,1,0.1,1),50,theta.des))

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

names(prt)
```

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

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

### piecewise contant 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` *Compute cumulative event counts as time-dependent covariates*

Description

For each subject and each row of `data`, counts the number of prior events of specified types in the recurrent event history. The resulting count columns can be used as time-dependent covariates in

subsequent models, e.g. to capture event-history dependence in the recurrent event rate.

Usage

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

Arguments

data	A data frame in counting-process format, with one row per event interval per subject.
status	Name of the column containing event status codes. Default is "status".
id	Name of the column containing subject identifiers. Default is "id".
types	Integer vector of status codes to count. Each value in types generates one new count column (when multitype = FALSE) or contributes to a single combined count (when multitype = TRUE). Default is 1.
names.count	Prefix for the names of the new count columns. The status code is appended, e.g. "Count1", "Count2". Default is "Count".
lag	Logical. If TRUE (default), the count at each row is the number of events strictly before the current time ($N(t-)$). If FALSE, events at the current time are included ($N(t)$).
multitype	Logical. If TRUE, events with status in types are aggregated into a single count column, optionally weighted by marks. If FALSE (default), a separate count column is created for each value in types.
marks	Optional numeric vector of weights applied to events when multitype = TRUE. If NULL (default), each event has weight 1.

Details

When lag = TRUE (default), the count at each row reflects events that occurred strictly before the current time point (i.e. $N(t-)$), making it suitable as a left-continuous covariate in counting-process models.

Value

The input data frame data with one or more new integer columns appended. With multitype = FALSE, columns are named paste0(names.count, k) for each k in types; with multitype = TRUE, a single column named paste0(names.count, types[1]) is added. An internal bookkeeping column lbnr__id is also added.

Author(s)

Thomas Scheike

Examples

```
data(hfactioncpx12)
hf <- hfactioncpx12
dtable(hf, ~status)

## Separate counts for event types 1 and 2
rr <- count_history(hf, types = 1:2, id = "id", status = "status")
dtable(rr, ~"Count*" + status, level = 1)
```

CPH_HPN_CRBSI *Rates for HPN program for patients of Copenhagen Cohort*

Description

Rates for HPN program for patients of Copenhagen Cohort

Format

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

Source

Estimated data

---

cumoddsreg *Cumulative Odds Regression for Discrete Time Data*

Description

A wrapper function for `interval_logitsurv_discrete` that simplifies the interface for discrete time-to-event data where the event time is observed exactly or as a factor level. It converts a factor response into an interval-censored format internally.

Usage

```
cumoddsreg(formula, data, ...)
```

Arguments

formula	Formula with a factor response on the left-hand side (representing the event time) and covariates on the right.
data	Data frame.
...	Arguments passed to interval_logitsurv_discrete.

Value

An object of class "cumoddsreg" with the same structure as interval_logitsurv_discrete.

Author(s)

Thomas Scheike

See Also

interval_logitsurv_discrete

---

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")
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+wmig1~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 <-  drename(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` *Dermal ridges data (families)*

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` *Dermal ridges data (monozygotic twins)*

Description

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

Format

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

Source

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

Examples

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

diabetes *The Diabetic Retinopathy Data*

Description

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

Format

This data frame contains the following columns:

id a numeric vector. Patient code.

agedx a numeric vector. Age of patient at diagnosis.

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

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

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

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

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

Source

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

Examples

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

divide_conquer *Split a data set and run function*

Description

Split a data set and run function

Usage

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

Arguments

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

Author(s)

Thomas Scheike, Klaus K. Holst

Examples

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

---

dlag *Lag operator*

Description

Lag operator

Usage

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

Arguments

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

Examples

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

`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

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

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

dreg *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.oneatatime=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.oneatatime=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.oneatatime=FALSE)

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

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

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

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

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

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

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

```
xs <- dreg(dat, y, fun.=ff)
}
```

`drelevel` *relev levels for data frames*

Description

levels shows levels for variables in data frame, relevel relevels 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`	seperator 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)
```

drop.specials *Remove Special Terms from a Formula*

Description

Removes terms such as `strata()` or `cluster()` from a formula/terms object.

Usage

```
drop.specials(x, components, ...)
```

Arguments

`x`	a formula object.
`components`	character vector of special term names to remove.
`...`	additional arguments passed to `lava::Specials`.

Value

The updated formula with an attribute `"variables"` containing the extracted variable names from removed specials.

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 asignment
dsort(hubble) <- ~sigma-v
```

`dspline` *Simple linear spline*

Description

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

Usage

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

Arguments

`data`	if x is formula or names for data frame then data frame is needed.
`y`	name of variable, or fomula, or names of variables on data frame.
`x`	name of variable, or fomula, or names of variables on data frame.
`breaks`	number of breaks, for variables or vector of break points,
`probs`	groups defined from quantiles
`equi`	for equi-spaced breaks
`regex`	for regular expressions.

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

Author(s)

Thomas Scheike

Examples

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

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

---

dtable *tables for data frames*

Description

tables for data frames

Usage

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

Arguments

data	if x is formula or names for data frame then data frame is needed.
y	name of variable, or 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 *Transform that allows condition*

Description

Defines new variables under condition for data frame

Usage

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

Arguments

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

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))
```

estimate.binregCV *Joint estimate across CV models for a binregCV object*

Description

Joint estimate across CV models for a binregCV object

Usage

```
## S3 method for class 'binregCV'
estimate(x, f = NULL, ...)
```

Arguments

x	A ‘binregCV‘ object.
f	Optional transformation function passed to ‘lava::estimate‘.
...	Not currently used.

`Event` *Event history object*

Description

Constructur for Event History objects

Usage

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

Arguments

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

Details

... content for details

Value

Object of class Event (a matrix)

Author(s)

Klaus K. Holst and Thomas Scheike

Examples

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

event_split *event_split (SurvSplit).*

Description

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

Usage

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

Arguments

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

Author(s)

Thomas Scheike

Examples

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

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

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

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

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

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

---

event_split2 *Event split with two time-scales, time and gaptime*

Description

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

Usage

```
event_split2(
  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 <- event_split2(rr,cuts=600,cuttime="time",  gaptime="gaptime",gaptime.entry="gapstart")
rr2 <- event_split2(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)
```

---

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

---

extendCums *Extend Cumulative Hazard Functions to Common Time Range*

Description

Extends a collection of cumulative hazard functions so that they all cover the same time range. Cumulative hazards that end before the maximum observed time are extrapolated linearly using either a user-supplied rate or the average hazard rate estimated from the existing cumulative hazard.

Usage

```
extendCums(cumA, cumB, extend = NULL)
```

Arguments

cumA	a cumulative hazard matrix (two columns: time, cumulative hazard) or a list of such matrices.
cumB	an optional second cumulative hazard matrix. If provided it is combined with cumA into a list.
extend	numeric vector of hazard rates used for extrapolation, one per cumulative hazard. If NULL (default), rates are estimated as the ratio of the last cumulative hazard value to the last time point.

Value

A list of cumulative hazard matrices extended to the common maximum time.

See Also

rchaz, rcrisk, mets-simulation

---

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 (famly) 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 specifices which of the subjects that are probands
`index.type`	argument passed to other functions
`num`	argument passed to other functions
`Rindex`	index starts with 1, in C is it is 0

Author(s)

Klaus Holst, Thomas Scheike

References

Cluster indeces

See Also

familycluster_index cluster_index

Examples

```
i<-c(1,1,2,2,1,3)
p<-c(1,0,0,1,0,1)
d<- familyclusterWithProbands_index(i,p)
print(d)
```

`fast.approx` *Fast approximation*

Description

Fast approximation

Usage

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

Arguments

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

Author(s)

Klaus K. Holst

Examples

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

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

fast.cluster *Fast Cluster Index Conversion*

Description

Converts a cluster variable to consecutive numeric indices using compiled code.

Usage

```
fast.cluster(x, ...)
```

Arguments

x	integer vector of cluster identifiers.
...	additional arguments (not used).

Value

Integer vector of 0-based cluster indices.

fast.pattern *Fast pattern*

Description

Fast pattern

Usage

```
fast.pattern(x, y, categories = 2, ...)
```

Arguments

x	Matrix (binary) of patterns. Optionally if y is also passed as argument, then the pattern matrix is defined as the elements agreeing in the two matrices.
y	Optional matrix argument with same dimensions as x (see above)
categories	Default 2 (binary)
...	Optional additional arguments

Author(s)

Klaus K. Holst

Examples

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

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

`fast.reshape` *Fast reshape*

Description

Fast reshape/tranpose of data

Usage

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

Arguments

`data`	data.frame or matrix
`varying`	Vector of prefix-names of the time varying variables. Optional for Long->Wide reshaping.
`id`	id-variable. If omitted then reshape Wide->Long.
`num`	Optional number/time variable
`sep`	String seperating prefix-name with number/time
`keep`	Vector of column names to keep
`idname`	Name of id-variable (Wide->Long)
`numname`	Name of number-variable (Wide->Long)

factor	If true all factors are kept (otherwise treated as character)
idcombine	If TRUE and id is vector of several variables, the unique id is combined from all the variables. Otherwise the first variable is only used as identifier.
labelnum	If TRUE varying variables in wide format (going from long->wide) are labeled 1,2,3,... otherwise use 'num' variable. In long-format (going from wide->long) varying variables matching 'varying' prefix are only selected if their postfix is a number.
labels	Optional labels for the number variable
regex	Use regular expressions
dropid	Drop id in long format (default FALSE)
...	Optional additional arguments

Author(s)

Thomas Scheike, Klaus K. Holst

Examples

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

##### From wide-format
(dd <- fast.reshape(d,"y"))
## Same with explicit setting new id and number variable/column names
## and seperator "" (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:::sim_BinFam(3)
d
fast.reshape(d,var="y")
fast.reshape(d,varying=list(c("ym","yf","yb1","yb2")))

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


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

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

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

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

`faster.reshape` *Fast Reshape from Long to Wide Format*

Description

Reshapes clustered long-format data to wide format efficiently using compiled code.

Usage

```
faster.reshape(data, clusters, index.type = FALSE, num = NULL, Rindex = 1)
```

Arguments

`data`	a matrix or data.frame to reshape.
`clusters`	vector of cluster identifiers, or column name in data.

index.type	logical; if TRUE, clusters are already 0-based indices.
num	optional within-cluster numbering variable.
Rindex	if 1, use R (1-based) indexing.

Value

A wide-format matrix.

---

folds *Generate Random Fold Indices for Cross-Validation*

Description

Splits n observations into random folds for cross-validation.

Usage

```
folds(n, folds = 10)
```

Arguments

n	number of observations.
folds	number of folds (default 10).

Value

A list of integer vectors, each containing indices for one fold.

---

force.same.cens *Force Same Censoring Within Clusters*

Description

Enforces the same censoring time within clusters (pairs) by censoring both subjects at the minimum censoring time when censoring occurs first.

Usage

```
force.same.cens(
  data,
  id = "id",
  time = "time",
  cause = "cause",
  entrytime = NULL,
  cens.code = 0
)

force_same_cens(
  data,
  id = "id",
  time = "time",
  cause = "cause",
  entrytime = NULL,
  cens.code = 0
)
```

Arguments

data	a data.frame in long format.
id	name of the cluster/pair identifier column.
time	name of the time variable.
cause	name of the cause/status variable.
entrytime	optional name of left-truncation time variable.
cens.code	value indicating censoring in the cause variable.

Value

A data.frame with enforced same censoring.

---

gof.phreg *Goodness-of-Fit for Cox PH Regression (Proportionality)*

Description

Performs cumulative score process residual tests for the proportional hazards (PH) assumption in Cox regression. The test statistics are based on the cumulative score process:

$$U(t) = \int_0^t (X_i - E(t)) d\hat{M}_i(s)$$

where $\hat{M}_i(s)$ are the martingale residuals.

Usage

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

Arguments

object	A fitted phreg object (from mets or survival).
n.sim	Number of simulations for the resampling procedure (default 1000).
silent	Logical; if TRUE, suppresses timing estimates for long jobs.
robust	Logical; if TRUE, uses robust martingale-based simulations. If NULL, defaults to TRUE if a cluster term is detected in the model call.
...	Additional arguments passed to lower-level functions.

Details

P-values are computed using the Lin, Wei, and Ying (1993) resampling method, which simulates the asymptotic distribution of the supremum of the score process under the null hypothesis of proportional hazards.

The function supports two types of simulation:

- **Standard**: Uses dN_i (counting process increments) for simulation.
- **Robust**: Uses $\hat{M}_i(t)$ (martingale residuals) adjusted for clustering if a `cluster()` term is present in the model.

Value

An object of class `"gof.phreg"` containing:

jumptimes	Event times used in the process.
supUsim	Matrix of simulated supremum values for each covariate.
res	Matrix with observed supremum (Sup|U(t)|) and p-values.
supU	Observed supremum values.
pvals	Vector of p-values for each covariate.
score	Cumulative score process values over time.
simUt	Simulated score processes.
type	Type of test performed ("prop").
robust	Logical flag indicating if robust simulation was used.

Author(s)

Thomas Scheike and Klaus K. Holst

References

Lin, D. Y., Wei, L. J., & Ying, Z. (1993). Checking the Cox model with cumulative sums of martingale-based residuals. Biometrika, 80(3), 557-572.

See Also

gofM_phreg, gofZ_phreg

Examples

```
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)
gg <- gof(m1)

## Robust simulations with cluster
sTRACE$id <- 1:500
m1 <- phreg(Surv(time,status==9)~vf+chf+diabetes+cluster(id), data=sTRACE)
gg <- gof(m1)
gg
```

---

gofM_phreg *Goodness-of-Fit for Cox Covariates (Model Matrix)*

Description

Tests the functional form of covariates in a Cox PH model by computing cumulative residuals against a user-specified model matrix. This helps detect non-linear effects or time-varying coefficients (interaction with time).

Usage

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

Arguments

formula	Formula for the Cox regression model.
data	Data frame.

offset	Offset vector.
weights	Weights vector.
modelmatrix	Matrix for cumulating residuals. Typically constructed using cumContr() or manually (e.g., quartiles of a continuous covariate).
n.sim	Number of simulations (default 1000).
silent	Logical; suppresses timing estimates if TRUE.
...	Additional arguments passed to phreg.

Details

The test statistic is:

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

where K is the model matrix (e.g., a set of basis functions for a continuous covariate) and $\hat{M}$ are the martingale residuals.

P-values are based on the Lin, Wei, and Ying (1993) resampling method. The plot shows whether the residuals are consistent with the model across the range of the covariate.

Value

An object of class "gof.phreg" containing:

jumptimes	Event times.
supUsim	Simulated supremum values.
res	Matrix with observed supremum and p-values for each column of modelmatrix.
score	Cumulative score process.
simUt	Simulated processes.
Utlast, pval.last	Supremum and p-value for the final time point (covariate direction).
type	Type of test ("modelmatrix").

Author(s)

Thomas Scheike and Klaus K. Holst

References

Lin, D. Y., Wei, L. J., & Ying, Z. (1993). Checking the Cox model with cumulative sums of martingale-based residuals. Biometrika, 80(3), 557-572.

See Also

gof.phreg, gofZ_phreg, cumContr

Examples

```
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)
}

## Cumulative sums in covariates via design matrix
mm <- mets:::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 *Goodness-of-Fit for Cox Covariates (Linearity)*

Description

Tests the functional form of continuous covariates in a Cox PH model to check for linearity. It computes cumulative residuals evaluated at a grid of covariate values z.

Usage

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

Arguments

formula	Formula for the Cox regression.
data	Data frame.
vars	Vector of variable names to test. If NULL, automatically detects continuous covariates with more than 2 levels.

offset	Offset vector.
weights	Weights vector.
breaks	Number of break points for the grid (default 50).
equi	Logical; if TRUE, uses equidistant breaks; if FALSE, uses quantiles.
n.sim	Number of simulations (default 1000).
silent	Logical; suppresses timing estimates.
...	Additional arguments passed to gofM_phreg.

Details

The test statistic is:

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

where $K(z)$ is a design matrix based on indicator functions $I(Z_i \leq z_l)$ for a grid of points z_l.

The p-value is valid but depends on the chosen grid. As the number of break points increases, this test converges to the original Lin, Wei, and Ying test for linearity.

Value

An object of class "gof.phreg" with type "Zmodelmatrix" containing:

res	Matrix of p-values for each tested variable.
Zres	List of gof.phreg objects, one for each variable.
type	Type of test ("Zmodelmatrix").

Author(s)

Thomas Scheike and Klaus K. Holst

See Also

gofM_phreg, cumContr

Examples

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

## Test linearity of continuous covariates
 ## 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 indpendent, and that the marginal cumulative incidence curves are on additive form

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

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

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

Details

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

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

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

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

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

$$\lambda = pard\theta$$

Value

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

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

Author(s)

Thomas Scheike

References

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

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

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

Examples

```
## Reduce Ex.Timings
d <- sim_nordic_random(1000,delayed=TRUE,
      cordz=1.0,cormz=2,lam0=0.3,country=TRUE)
times <- seq(50,90,by=10)
addm <- timereg::comp.risk(Event(time,cause)~-1+factor(country)+cluster(id),data=d,
times=times,cause=1,max.clust=NULL)
```

```
### making group indidcator
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=pardes,random.design=des.rv)
summary(outacem)
```

`grouptable` *Create Group Contingency Table from Clustered Data*

Description

Creates a contingency table by group from paired/clustered data, optionally combining lower and upper triangles.

Usage

```
grouptable(
  data,
  id,
```

```
  group,
  var,
  lower = TRUE,
  labels,
  order,
  group.labels,
  group.order,
  combine = " & ",
  ...
)
```

Arguments

data	a data.frame.
id	name of the cluster/pair identifier column.
group	name of the grouping variable (e.g., zygosity).
var	name of the outcome variable to tabulate.
lower	logical; if TRUE, fold upper triangle into lower.
labels	optional labels for levels of var.
order	optional ordering of factor levels.
group.labels	optional labels for the groups.
group.order	optional ordering of groups.
combine	separator for combining two groups (default " & ").
...	additional arguments.

Value

A table or list of tables.

---

haplo *haplo fun data*

Description

haplo fun data

Format

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

Source

Estimated data

haplo_surv_discrete *Discrete Time-to-Event Haplotype Analysis*

Description

Performs cycle-specific logistic regression to estimate haplotype effects on discrete time-to-event data, accounting for phase ambiguity. Given observed genotypes G and unobserved haplotypes H, the method integrates (mixes out) over the possible haplotype configurations using the conditional probabilities $P(H|G)$.

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 (must be sorted by id and time) containing the ID, time variable, binary response, and covariates.
y	Name of the response variable (binary, 0/1) in X.
time.name	Name of the time variable in X used for sorting and cycle definition.
Haplos	Data frame containing id, haplo1, haplo2 (haplotypes as factors), and p (probability $P(H|G)$).
id	Name of the ID variable in X and Haplos.

desnames	Names of the covariate columns in X to be used in the design matrix.
designfunc	Function that computes the design vector given haplotypes $h = (h_1, h_2)$ and covariates. Must return a vector or matrix compatible with the model.
beta	Starting values for the optimization (vector of length $p+k$, where p is the number of covariate effects and k is the number of time cycles).
no.opt	Logical; if TRUE, skips optimization and returns estimates based on the provided beta (useful for initialization or diagnostics).
method	Optimization method: "NR" (Newton-Raphson, default) or "nlm".
stderr	Logical; if FALSE, returns only the coefficient estimates.
designMatrix	Alternative to X and Haplos: provides the response and design matrix directly (not fully implemented).
response	Alternative to X: provides the response and design directly (not fully implemented).
idhap	Name of the ID-haplotype variable to specify different haplotypes for different IDs.
design.only	Logical; if TRUE, returns only the design matrices constructed for the analysis.
covnames	Names of covariates to extract from the object for regression output.
fam	Family of the model (default binomial, currently the only option).
weights	Weights following ID for the GLM component.
offsets	Offsets following ID for the GLM component.
idhapweights	Weights following ID-haplotype for the GLM component (Work in Progress).
...	Additional arguments passed to the optimizer (lava::NR or nlm).

Details

The survival function is computed by averaging over the possible haplotypes:

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

The discrete hazard function is modeled using logistic regression:

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

where α_t are time-specific intercepts (baseline hazards), $x(h)$ is the regression design constructed from covariates and haplotypes $h = (h_1, h_2)$, and β are the regression coefficients.

The likelihood is maximized numerically. Standard errors are computed assuming that $P(H|G)$ is known (i.e., ignoring the uncertainty in haplotype estimation).

The design matrix over possible haplotypes is constructed by merging the covariate data X with the haplotype probabilities $Haplos$ and applying a user-defined designfunc.

Value

An object of class "haplosurvd" containing:

coef	Estimated coefficients (baseline time effects and haplotype/covariate effects).
se	Standard errors of the coefficients.
var	Variance-covariance matrix.
se.robust	Robust standard errors (if available).
iid	Influence function (IID) decomposition.
ploglik	Log-likelihood at convergence.
gradient, hessian	Optimization results.
Xhap, X, Haplos	Data and design matrices used.
nid, nidhap	Number of IDs and ID-haplotype combinations.

Author(s)

Thomas Scheike

References

Scheike, T. H. (2024). Discrete time survival analysis with haplotype effects. mets package documentation.

Examples

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

## Some haplotype frequencies for simulations
data(haplo)
hapfreqs <- haplo$hapfreqs

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

baseline <- hapfreqs$haplotype[9]
baseline

## Design function: indicator for presence of any 'types' haplotype
designftypes <- function(x, sm=0) {
  hap1 <- x[1]
  hap2 <- x[2]
  if (sm == 0) y <- 1 * ((hap1 == types) | (hap2 == types))
  if (sm == 1) y <- 1 * (hap1 == types) + 1 * (hap2 == types)
  return(y)
}

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

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

haploX$time <- haploX$times
Xdes <- model.matrix(~ factor(time), haploX)
colnames(Xdes) <- paste("X", 1:ncol(Xdes), sep="")
X <- dkeep(haploX, ~ id + y + time)
X <- cbind(X, Xdes)
Haplos <- dkeep(ghaplos, ~ id + "haplo*" + p)
desnames <- paste("X", 1:6, sep="")   # Six X's related to 6 cycles
out <- haplo_surv_discrete(X=X, y="y", time.name="time",
         Haplos=Haplos, desnames=desnames, designfunc=designftypes)
names(out$coef) <- c(desnames, types)
out$coef
summary(out)
```

---

hfactioncpx12 — *hfaction, subset of block randomized study HF-ACtion from WA package*

Description

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

Format

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

Source

WA package, Connor et al. 2009

Examples

```
data(hfactioncpx12)
```

IC.binreg — *Influence curve components for binomial regression ATE*

Description

Extracts and scales influence function (IC) components from a fitted binomial regression ATE object or raw IC matrix. Depending on the object class and requested ‘type‘, returns one or more components combined by column-binding.

Usage

```
## S3 method for class 'binreg'
IC(x, type = c("coef", "DR", "G"), ...)
```

Arguments

x	An object of class ATE, or an object containing an iid component.
type	Character vector specifying which influence curve components to return. Possible values are: **coef** Influence curve for regression coefficients **DR** Doubly robust ATE influence curve **G** G-computation ATE influence curve If x inherits from ATE and type is missing, the default is c("DR", "G").
...	Currently unused.

Details

For objects of class ATE, the function extracts and rescales the stored influence curve components:

- x$iid for coefficient ICs
- x$riskDR.iid for doubly robust ATE ICs
- x$riskG.iid for G-computation ICs

Each component is multiplied by its sample size (NROW(.)).

For non-ATE objects, the function returns NROW(x$iid) * x$iid.

Value

A numeric matrix obtained by column-binding the selected influence curve components. Each column corresponds to one element of type, with column names preserved.

Examples

```
## Not run:
data(bmt)
dfactor(bmt)  <-  ~.

fit  <- binregATE(Event(time,cause)~tcell.f+platelet+age,bmt,time=50,cause=1,
  treat.model=tcell.f~platelet+age)
# default ATE behavior (DR + G)
head(IC(fit))

# only DR component
head(IC(fit, "DR"))

# multiple components
head(IC(fit, c("coef", "DR")))
```

```
## End(Not run)
```

`IC.phreg` *Influence Functions for phreg objects*

Description

Computes the influence functions (IID decomposition) for the regression coefficients and/or the baseline cumulative hazard at a specific time point.

Usage

```
## S3 method for class 'phreg'
IC(x, type = "robust", all = FALSE, time = NULL, baseline = NULL, ...)
```

Arguments

`x`	Object of class `"phreg"`.
`type`	Type of influence function: `"robust"` (default) or `"martingale"`.
`all`	Logical; if `TRUE`, returns both beta and baseline influence functions.
`time`	Time point for baseline influence function (required if baseline is requested).
`baseline`	Arguments for baseline estimation.
`...`	Additional arguments.

Value

A matrix of influence functions. If `all=TRUE`, columns correspond to regression coefficients and baseline cumulative hazard. Attributes include `coef` and `time`.

Author(s)

Thomas Scheike

See Also

`phreg`, `iidBaseline`

`iidBaseline` *Influence Functions or IID Decomposition of Baseline*

Description

Computes the influence functions for the baseline cumulative hazard (and optionally regression coefficients) for `phreg`, `recreg`, or `cifregFG` objects.

Usage

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

Arguments

`object`	Object of class `"phreg"`, `"recreg"`, or `"cifregFG"`.
`time`	Time point for baseline IID (required).
`ft`	Function to compute IID of baseline integrated against $f(t)$.
`fixbeta`	Logical; if `TRUE`, fixes the coefficients (useful for specific tests).
`beta.iid`	Optional matrix of beta influence functions to use.
`tminus`	Logical; if `TRUE`, computes predictions at $t-$ (strictly before t), useful for IPCW techniques.
`...`	Additional arguments passed to lower-level functions.

Details

The decomposition is based on the formula:

$$\sum_i \int_0^t \frac{f(s)}{S_0(s)} dM_{ki}(s) - P(t)\beta_k$$

where k denotes the stratum and i the cluster.

Value

An object of class `"iidBaseline"` containing:

`time`	Time point.
`base.iid`	Influence functions for the baseline.

`beta.iid`	Influence functions for the regression coefficients.
`cumhaz.time`	Cumulative hazard at the specified time.
`strata`	Strata indices.

Author(s)

Thomas Scheike

See Also

`phreg`, `recreg`, `cifreg`

`ilap` *Inverse Laplace Transform Helper*

Description

Computes the inverse Laplace transform for gamma frailty simulation.

Usage

```
ilap(theta, t)
```

Arguments

`theta`	frailty parameter.
`t`	value at which to evaluate.

Value

Numeric value of the inverse Laplace transform.

`interval_logitsurv_discrete` *Discrete Time-to-Event Analysis with Interval Censoring*

Description

Fits a cumulative odds model for discrete time-to-event data, handling interval censoring where the event time is known only to lie within an interval $(t_l, t_r]$. The model assumes:

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

where $G(t)$ is the baseline cumulative odds function and β are the regression coefficients. This is equivalent to:

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

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 with an Interval object (e.g., Interval(entry, time)) on the left-hand side and covariates on the right. Can include cluster() for correlated data.
data	Data frame containing the variables in the formula.
beta	Starting values for the optimization (vector of length $p+k$, where p is the number of covariates and k is the number of time intervals).
no.opt	Logical; if TRUE, skips optimization and returns estimates based on the provided beta (useful for initialization).
method	Optimization method: "NR" (Newton-Raphson, default) or "nlm".
stderr	Logical; if FALSE, returns only the coefficient estimates.
weights	Observation weights (follows ID).
offsets	Offsets (follows ID).
exp.link	Logical; if TRUE, parameterizes increments as $\exp(\alpha) > 0$.
increment	Logical; if TRUE, uses increments $dG(t) = \exp(\alpha)$ as parameters.
...	Additional arguments passed to the optimizer (lava::NR or nlm).

Details

The baseline $G(t)$ is parameterized as the cumulative sum of exponentials ($G(t) = \sum \exp(\alpha)$), ensuring positivity. The regression coefficients describe the log-odds of the event occurring by time t.

The likelihood is maximized over the observed intervals:

$$L = \prod_i [P(T_i > t_{il}|x_i) - P(T_i > t_{ir}|x_i)]$$

where t_{il} and t_{ir} are the left and right endpoints of the interval for subject i. Right-censored intervals have $t_{ir} = \infty$.

Value

An object of class "cumoddsreg" containing:

coef	Estimated coefficients (baseline time effects and covariate effects).
se.coef	Standard errors of the coefficients.
var	Variance-covariance matrix.
iid	Influence function (IID) decomposition for robust variance estimation.
ntimes	Number of distinct time intervals.
utimes	Unique time points.
ploglik	Log-likelihood at convergence.
gradient, hessian	Optimization results.
call	Original function call.

Author(s)

Thomas Scheike

References

Scheike, T. H. (2024). Discrete time survival analysis with interval censoring. mets package documentation.

See Also

cumoddsreg, predictlogitSurvd, simlogitSurvd

Examples

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

out <- interval_logitsurv_discrete(Interval(entry,time2)~X1+X2+X3+X4,ttpd)
summary(out)
head(iid(out))

pred <- predictlogitSurvd(out,se=FALSE)
plotSurvd(pred)

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

ipw *Inverse Probability of Censoring Weights*

Description

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

Usage

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

Arguments

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

Author(s)

Klaus K. Holst

Examples

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

## End(Not run)
```

---

ipw2 *Inverse Probability of Censoring Weights*

Description

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

Usage

```
ipw2(
  data,
  times = NULL,
  entrytime = NULL,
  time = "time",
  cause = "cause",
  same.cens = FALSE,
  cluster = NULL,
  pairs = FALSE,
  strata = NULL,
  obs.only = TRUE,
  cens.formula = NULL,
  cens.code = 0,
  pair.cweight = "pcw",
  pair.tweight = "ptw",
  pair.weight = "weights",
  cname = "cweights",
```

```
  tname = "tweights",
  weight.name = "indi.weights",
  prec.factor = 100,
  ...
)
```

Arguments

data	data frame
times	possible time argument for speciying a maximum value of time tau=max(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 mininum 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 censorig 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 <- sim_nordic_random(5000,delayed=TRUE,ptrunc=0.7,
      cordz=0.5,cormz=2,lam0=0.3,country=FALSE)
d$strata <- as.numeric(d$country)+(d$zyg=="MZ")*4
```

```
times <- seq(60,100,by=10)
## c1 <- timereg::comp.risk(Event(time,cause)~1+cluster(id),data=d,cause=1,
##  model="fg",times=times,max.clust=NULL,n.sim=0)
## mm=model.matrix(~-1+zyg,data=d)
## out1<-random_cif(c1,data=d,cause1=1,cause2=1,same.cens=TRUE,theta.des=mm)
## summary(out1)
## pc1 <- predict(c1,X=1,se=0)
## plot(pc1)
##
## dl <- d[!d$truncated,]
## dl <- ipw2(dl,cluster="id",same.cens=TRUE,time="time",entrytime="entry",cause="cause",
##            strata="strata",prec.factor=100)
## cl <- timereg::comp.risk(Event(time,cause)~+1+
##   cluster(id),
##    data=dl,cause=1,model="fg",
##   weights=dl$indi.weights,cens.weights=rep(1,nrow(dl)),
##           times=times,max.clust=NULL,n.sim=0)
## pcl <- predict(cl,X=1,se=0)
## lines(pcl$time,pcl$P1,col=2)
## mm=model.matrix(~-1+factor(zyg),data=dl)
## out2<-random_cif(cl,data=dl,cause1=1,cause2=1,theta.des=mm,
##                  weights=dl$weights,censoring.weights=rep(1,nrow(dl)))
## summary(out2)
```

`jumptimes` *Extract Event (Jump) Times*

Description

Extracts unique event times from survival data, optionally restricting to concordant pairs or specific causes.

Usage

```
jumptimes(
  time,
  status = TRUE,
  id,
  cause,
  sample,
  sample.all = TRUE,
  strata = NULL,
  num = NULL,
  ...
)
```

Arguments

time	vector of event/censoring times.
status	vector of status indicators (default TRUE = event).
id	optional cluster identifier for concordant pair times.
cause	optional cause value to select.
sample	optional maximum number of time points to return.
sample.all	logical; if TRUE and sampling, include remaining times.
strata	not used.
num	optional within-cluster numbering variable.
...	additional arguments.

Value

Sorted vector of event times.

---

km *Kaplan-Meier with Robust Standard Errors*

Description

Computes Kaplan-Meier estimates with robust standard errors. Robust variance is the default and is obtained from the predict call.

Usage

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

Arguments

formula	Formula with 'Surv' or 'Event' outcome.
data	Data frame.
km	Logical; if TRUE, returns Kaplan-Meier estimates; otherwise returns Nelson-Aalen based estimates.
...	Additional arguments passed to phreg.

Value

An object of class "km" (extends "predictphreg") containing:

surr	Survival probabilities.
se.surv	Standard errors.
lower, upper	Confidence intervals.

Author(s)

Thomas Scheike

Examples

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

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

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

---

lifecourse *Life-course plot*

Description

Life-course plot for event life data with recurrent events

Usage

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

Arguments

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` *Life table*

Description

Create simple life table

Usage

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

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

Arguments

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

Author(s)

Klaus K. Holst

Examples

```
library(timereg)
data(TRACE)

d <- with(TRACE,lifetable(Surv(time,status==9)~sex+vf,breaks=c(0,0.2,0.5,8.5)))
lava::estimate(glm(events ~ offset(log(atrisk))+factor(int.end)*vf + sex*vf,
            data=d,poisson))
```

LinSpline *Simple linear spline*

Description

Simple linear spline

Usage

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

Arguments

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

Author(s)

Thomas Scheike

logitSurv *Proportional Odds Survival Model*

Description

Fits a semiparametric proportional odds model where:

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

Thus, covariate effects represent the odds ratio (OR) of survival.

Usage

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

Arguments

formula	Formula with 'Surv' outcome (similar to coxph).
data	Data frame.
offset	Offsets for $\exp(x\beta)$ terms.
weights	Weights for score equations.
...	Additional arguments passed to lower-level functions.

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.

Value

An object of class "phreg" with propodds=1.

Author(s)

Thomas Scheike

References

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

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

```
 ## Reduce Ex.Timings
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$id <- 1:n
dat$ftime <- 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
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 *Computes mediation weights*

Description

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

Usage

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

Arguments

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

Author(s)

Thomas Scheike

melanoma *The Melanoma Survival Data*

Description

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

Format

This data frame contains the following columns:

no a numeric vector. Patient code.

status a numeric vector code. Survival status. 1: dead from melanoma, 2: alive, 3: dead from other cause.

days a numeric vector. Survival time.

ulc a numeric vector code. Ulceration, 1: present, 0: absent.

thick a numeric vector. Tumour thickness (1/100 mm).

sex a numeric vector code. 0: female, 1: male.

Source

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

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

Examples

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

---

mena *Menarche data set*

Description

Menarche data set

Source

Simulated data

mets.options *Set global options for* mets

Description

Extract and set global parameters of mets.

Usage

```
mets.options(...)
```

Arguments

...	Arguments

Details

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

Value

list of parameters

Examples

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

## End(Not run)
```

migr *Migraine data*

Description

Migraine data

`mlogit` *Multinomial Regression Based on phreg*

Description

Fits a multinomial regression model for a categorical outcome with K levels:

$$P_i = \frac{\exp(X^\top \beta_i)}{\sum_{j=1}^{K} \exp(X^\top \beta_j)}$$

for $i = 1, \ldots, K$, where $\beta_1 = 0$ (baseline category).

Usage

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

Arguments

`formula`	Formula with the outcome (similar to `coxph`). The outcome must be a factor.
`data`	Data frame containing the variables.
`offset`	Offsets for the partial likelihood.
`weights`	Weights for the score equations.
`fix.X`	Logical; if `TRUE`, forces the same coefficients for all categories (except intercepts).
`...`	Additional arguments passed to lower-level functions.

Details

This ensures that $\sum_j P_j = 1$. The model is fitted using the `phreg` function by expanding the data into a long format with strata for each category.

The coefficients represent the log-Relative-Risk (log-RR) relative to the baseline group (the first level of the factor, which can be reset using `relevel`).

Standard errors are computed based on the sandwich form:

$$DU^{-1} \left(\sum U_i^2 \right) DU^{-1}$$

where U is the score vector and D is the derivative matrix.

Influence functions (possibly robust) can be obtained via the `iid()` function. The response variable should be a factor.

Can also fit a cumulative odds model as a special case of `interval_logitsurv_discrete`.

Value

An object of class "mlogit" (extending "phreg") containing:

coef	Matrix of estimated coefficients (rows correspond to categories, columns to covariates).
var	Robust variance-covariance matrix.
iid	Influence functions for the coefficients.
nlev	Number of levels in the outcome.
px	Number of covariates.

Author(s)

Thomas Scheike

Examples

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

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

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

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

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

## predictions based on seen response or not
## all probabilities
head(predict(mreg,response=FALSE))
head(predict(mreg))
## using newdata
newdata <- data.frame(tcell=c(1,1,1),platelet=c(0,1,1),cause1f=c("2","2","0"))
## only probability of seen response
predict(mreg,newdata)
## without response
predict(mreg,newdata,response=FALSE)
## given indexx of P(Y=j)
predict(mreg,newdata,Y=c(1,2,3))
```

```
##  reponse not given
newdata <- data.frame(tcell=c(1,1,1),platelet=c(0,1,1))
predict(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

---

p11_binomial_twostage_RV
Concordance Probability from Twostage Model

Description

Computes concordance probability (joint probability of both subjects experiencing the event) given dependence parameters and random-effect variance structures from a twostage model.

Computes concordance probabilities for twin ACE/ADE models from a binomial twostage object.

Functions for constructing random-effects design matrices for twin and family models. These designs specify the genetic (A), dominance (D), common environment (C), and unique environment (E) variance components.

Usage

```
p11_binomial_twostage_RV(
  theta,
  rv1,
  rv2,
  p1,
  p2,
  pardes,
  ags = NULL,
  link = 0,
  i = 1,
  j = 1
)

concordanceTwostage(
  theta,
  p,
  rv1,
  rv2,
  theta.des,
  additive.gamma.sum = NULL,
  link = 0,
  var.par = 0,
  ...
)

concordanceTwinACE(
  object,
  rv1 = NULL,
  rv2 = NULL,
  xmarg = NULL,
  type = "ace",
  ...
)

kendall_ClaytonOakes_twin_ace(parg, parc, K = 10000, test = 0)

kendall.ClaytonOakes.twin.ace(x, y, ...)

kendall_normal_twin_ace(parg, parc, K = 10000)

ascertained_pairs(pairs, data, cr.models, bin = FALSE)

twin.polygen.design(x, ...)

ace_family_design(
  data,
  id = "id",
```

```
  member = "type",
  mother = "mother",
  father = "father",
  child = "child",
  child1 = "child",
  type = "ace",
  ...
)

make_pairwise_design(pairs, kinship, type = "ace")
```

Arguments

theta	dependence parameter vector.
rv1	random-effects design for subject 1.
rv2	random-effects design for subject 2.
p1	marginal probability for subject 1.
p2	marginal probability for subject 2.
pardes	parameter design matrix.
ags	additive gamma sum matrix (optional).
link	link function indicator (0 = identity, 1 = log).
i	index for subject 1.
j	index for subject 2.
p	matrix of marginal probabilities (n x 2).
theta.des	parameter design matrix linking theta to lambda parameters.
additive.gamma.sum	optional matrix for additive gamma sums.
var.par	if 1, parameters are rescaled by sum squared.
...	additional arguments.
object	a binomial twostage model object.
xmarg	optional covariate values for marginal probabilities.
type	model type: "ace", "ade", "ae", "de", "dce", or "un".
parg	genetic variance parameter (gamma shape for genetic component).
parc	common environment variance parameter (gamma shape for environment).
K	number of simulated twin pairs (multiplied by 2 internally).
test	if 1, prints diagnostic correlations.
x	passed as parg (alias wrapper).
y	passed as parc (alias wrapper).
pairs	matrix of pair indices (n x 2).
data	a data.frame with twin/family data.
cr.models	formula specifying time and status variables.

bin	logical; if TRUE uses binary (prevalence) ordering rather than time ordering.
id	character name of the cluster (pair) identifier column.
member	character name of the family member type column.
mother	value identifying mothers in the member column.
father	value identifying fathers in the member column.
child	value identifying children in the member column.
child1	column name distinguishing first child from second.
kinship	vector of kinship coefficients for each pair.

Details

twin_polygen_design creates a polygenic random-effects design for twin pairs, distinguishing MZ and DZ twins.

twin.polygen.design is an alias for twin_polygen_design.

ace_family_design creates designs for nuclear families (mother, father, children).

make_pairwise_design creates pairwise random-effects designs for arbitrary kinship structures.

concordanceTwostage computes concordance probabilities from a twostage model.

concordanceTwinACE computes concordance from a twin ACE model.

kendall_ClaytonOakes_twin_ace and kendall_normal_twin_ace compute Kendall's tau for Clayton-Oakes and normal-frailty twin ACE models respectively.

ascertained_pairs identifies ascertained (affected) pairs in clustered survival data.

p11_binomial_twostage_RV computes the joint probability P(T1<=t, T2<=t) for the additive gamma binary random effects model.

Value

A list of concordance tables, one per pair, each containing pmat (2x2 probability matrix), casewise (casewise concordance), and marg (marginal probabilities).

A list of concordance tables per zygosity group.

A list with components:

pardes	parameter design matrix linking random effects to variance parameters.
des.rv	random-effects design matrix for subjects.

Author(s)

Klaus K. Holst, Thomas Scheike

phreg *Fast Cox Proportional Hazards Regression*

Description

Fits a Cox proportional hazards model using a fast C++ backend. Robust variance (sandwich estimator) is the default variance estimate in the summary.

Usage

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

Arguments

formula	Formula with a 'Surv' or 'Event' outcome (similar to coxph).
data	Data frame containing the variables.
offset	Offsets for the Cox model linear predictor.
weights	Weights for the Cox score equations.
...	Additional arguments passed to lower-level functions (e.g., optimization controls).

Details

The influence functions (IID decomposition) follow the numerical order of the given cluster variable. Ordering the results by '$id' will align the IID terms with the original dataset order.

Value

An object of class "phreg" containing:

coef	Vector of estimated coefficients.
var	Robust variance-covariance matrix.
beta.iid	Influence functions for the regression coefficients.
cumhaz	Matrix of cumulative hazard estimates (time, cumhaz).
se.cumhaz	Matrix of standard errors for the cumulative hazard.
cox.prep	List containing preprocessed data for the Cox model.
opt	Optimization results (if optimization was performed).
call	The matched call.

Author(s)

Klaus K. Holst, Thomas Scheike

See Also

plot.phreg, predict.phreg, robust_phreg

Examples

```
data(TRACE)
dcut(TRACE) <- ~.
# Fit model with clustering
out1 <- phreg(Surv(time, status == 9) ~ wmi + age + strata(vf, chf) + cluster(id), data = TRACE)
summary(out1)

# Plotting baselines
par(mfrow = c(1, 2))
plot(out1)

# Computing robust variance for baseline
rob1 <- robust_phreg(out1)
plot(rob1, se = TRUE, robust = TRUE)

# IID decomposition for regression parameters
head(iid(out1))

# IID decomposition for baseline at a specific time-point
Aiiid <- iid(out1, time = 30)
head(Aiiid)

# Combined IID decomposition (beta and baseline)
dd <- iidBaseline(out1, time = 30)
head(dd$beta.iid)
head(dd$base.iid)

# Stratified model
outs <- phreg(Surv(time, status == 9) ~ strata(vf, wmicat.4) + cluster(id), data = TRACE)
summary(outs)
par(mfrow = c(1, 2))
plot(outs)
```

---

phreg_IPTW *IPTW Cox Regression (Inverse Probability of Treatment Weighted)*

Description

Fits a Cox model with treatment weights

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

, where

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

.

Usage

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

Arguments

formula	Formula for phreg.
data	Data frame for risk averaging.
treat.model	Propensity score model (binary or multinomial).
treat.var	A 1/0 variable indicating when treatment is given (for time-dependent weights).
weights	Optional weights to multiply with the IPTW weights.
estpr	(=1, default) to estimate propensity scores and include their uncertainty in the influence function.
pi0	Fixed simple weights (if estpr=0).
se.cluster	To compute GEE-type standard errors when additional cluster structure is present.
...	Arguments for phreg call.

Details

Standard errors are computed via influence functions that are returned as the IID argument. Propensity scores are fitted using either logistic regression (glm) or the multinomial model (mlogit) when there are more than two treatment categories.

The treatment variable must be a factor and is identified on the RHS of the treat.model. Recurrent events can be considered with a start-stop structure, requiring cluster(id). Robust standard errors are computed in all cases.

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

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

where

$$T_r$$

is time of 2nd randomization. The weights are constructed using a glm or mlogit model based on the data where treat.var=1. The propensity score can be constructed for any number of treatments in a similar manner.

Value

An object of class "phreg" with additional IPTW components:

IID	Influence functions including propensity score uncertainty.
iptw	IPTW weights used.
naive.var	Naive variance ignoring propensity score uncertainty.

Author(s)

Thomas Scheike

Examples

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

---

phreg_rct *Lu-Tsiatis More Efficient Log-Rank for Randomized Studies with Baseline Covariates*

Description

Efficient implementation of the Lu-Tsiatis improvement using baseline covariates, extended to competing risks and recurrent events. The results are almost equivalent to the speffSurv function of the speff2trial package in the survival case. A dynamic censoring augmentation regression is also computed to gain additional efficiency from the censoring augmentation. The function handles two-stage randomizations and recurrent events (start,stop) with cluster structure.

Usage

```
phreg_rct(
  formula,
  data,
  cause = 1,
  cens.code = 0,
  typesR = c("R0", "R1", "R01"),
  typesC = c("C", "dynC"),
  weights = NULL,
  augmentR0 = NULL,
  augmentR1 = NULL,
  augmentC = NULL,
  treat.model = ~+1,
  RCT = TRUE,
  treat.var = NULL,
  km = TRUE,
```

```
  level = 0.95,
  cens.model = NULL,
  estpr = 1,
  pi0 = 0.5,
  base.augment = FALSE,
  return.augmentR0 = FALSE,
  mlogit = FALSE,
  ...
)
```

Arguments

formula	Formula with Surv or Event outcome (see coxph) and treatment variable (randomization 0/1). The treatment variable must be the first covariate on the right-hand side.
data	Data frame containing all variables referenced in the formula.
cause	Numeric code for the event of interest in competing risks or recurrent events.
cens.code	Numeric code for censoring in competing risks or recurrent events.
typesR	Character vector specifying augmentation types for randomization (options: "R0" for baseline, "R1" for post-baseline, "R01" for both).
typesC	Character vector specifying augmentation types for censoring (options: "C" for static, "dynC" for dynamic).
weights	Weights to be used for phreg.
augmentR0	Formula for the first randomization augmentation (e.g., ~age+sex).
augmentR1	Formula for the second randomization augmentation (e.g., ~age+sex).
augmentC	Formula for the censoring augmentation (e.g., ~age+sex).
treat.model	Propensity score model formula (default is ~+1, assuming RCT).
RCT	Logical; if FALSE, uses propensity score adjustment for marginal model.
treat.var	Variable indicating treatment times in two-stage randomization.
km	Logical; use Kaplan-Meier for censoring weights (stratified on treatment).
level	Confidence level for intervals (default 0.95).
cens.model	Censoring model formula (default is ~strata(treatment)).
estpr	Numeric code (1/0); estimate propensity scores or not (default TRUE).
pi0	Fixed propensity scores for randomizations (if not estimating).
base.augment	Logical; covariate augment baselines (only for R0 augmentation).
return.augmentR0	Logical; return augmentation data.
mlogit	Logical; use multinomial logistic regression for propensity scores.
...	Additional arguments passed to phreg function.

Details

The method improves the efficiency of the log-rank test by utilizing auxiliary baseline covariates to reduce variance, particularly useful in randomized clinical trials (RCTs) where covariate adjustment can increase power.

Value

An object of class "phreg_rct" containing:

coefs	Coefficient estimates for the treatment effect.
iid	Influence function (IID) decomposition for variance estimation.
AugR0, AugR1, AugCdyn, AugClt	Augmentation terms for different strategies.
cumhaz	Cumulative hazards.
var	Variance-covariance matrix.
se	Standard errors.
call	Original function call.
formula	Formula used.
data	The data used (if requested).

The object includes results for different augmentation combinations (R0, R1, R01, C, dynC).

Author(s)

Thomas Scheike

References

Lu, T. and Tsiatis, A. A. (2008), Improving the efficiency of the log-rank test using auxiliary covariates, Biometrika, 95, 679–694.

Scheike, T. H., Nerstroem, C. and Martinussen, T. (2026), Randomized clinical trials and the proportional hazards model for recurrent events, TEST.

Examples

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

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

---

phreg_weibull *Weibull-Cox regression*

Description

Fits a Cox-Weibull with cumulative hazard given by

$$\Lambda(t) = \lambda \cdot t^s$$

where s is the shape parameter, and λ the rate parameter. We here allow a regression model for both parameters

$$\lambda := \exp(\beta^\top X)$$

$$s := \exp(\gamma^\top Z)$$

as defined by 'formula' and 'shape.formula' respectively.

Usage

```
phreg_weibull(
  formula,
  shape.formula = ~1,
  data,
  save.data = TRUE,
  control = list()
)
```

Arguments

formula	Formula for proportional hazards. The right-handside must be an [Event] or [Surv] object (with right-censoring and possibly delayed entry).
shape.formula	Formula for shape parameter
data	data.frame
save.data	if TRUE the data.frame is stored in the model object (for predictions and simulations)
control	control arguments to optimization routine [stats::nlmbin]

Details

The parametrization

Value

'phreg.par' object

Author(s)

Klaus Kähler Holst, Thomas Scheike

See Also

[mets::phreg()]

Examples

```
data(sTRACE, package="mets")
sTRACE$entry <- 0
fit1 <- phreg_weibull(Event(entry, time, status == 9) ~ age,
            shape.formula = ~age, data = sTRACE)
tt <- seq(0,10, length.out=100)
pr1 <- predict(fit1, newdata = sTRACE[1, ], times = tt)
fit2 <- phreg(Event(time, status == 9) ~ age, data = sTRACE)
pr2 <- predict(fit2, newdata = sTRACE[1, ], se = FALSE)
if (interactive()) {
   plot(pr2$times, pr2$surv, type="s")
   lines(tt, pr1[,1,1], col="red", lwd=2)
}
```

---

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

Description

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

Usage

```
plack_cif(cif1, cif2, object)
```

Arguments

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

Author(s)

Thomas Scheike

---

plot.phreg *Plotting the baselines of stratified Cox*

Description

Plotting the baselines of stratified Cox

Usage

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

Arguments

x	phreg object
...	Additional arguments to baseplot funtion

Author(s)

Klaus K. Holst, Thomas Scheike

Examples

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

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

---

plot_twin *Scatter plot function*

Description

Scatterplot with contours of the (kernel) estimated density

Usage

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

Arguments

data	bivariate data to plot (data.frame or matrix with 2 columns)
marginal.args	argumemts to marginal estimator (‘density‘ continuous data, ‘barplot‘ for categorical)
kernsmooth.args	arguments to 2d-kernel smoother
xlab	x-axis label
ylab	y-axis label
col	color of points
col2	color of contour / density plot
alpha	transparency level of points
grid	should grid be added to the plot
side.plot	If TRUE subplots of the marginal distributions are added to the plot
...	arguments to lower level plot functions

Author(s)

Klaus Kähler Holst

Examples

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

# continuous variables
plot_twin(datamz)

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

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

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

pmvn *Multivariate normal distribution function*

Description

Multivariate normal distribution function

Usage

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

Arguments

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

Examples

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

predict.mlogit *Predictions from Multinomial Regression*

Description

Computes predicted probabilities for the categorical outcome based on a `mlogit` object. Can return probabilities for all categories or only for the observed response.

Usage

```
## S3 method for class 'mlogit'
predict(
  object,
  newdata,
  se = TRUE,
  response = TRUE,
  Y = NULL,
  level = 0.95,
  ...
)
```

Arguments

object	Object of class "mlogit".
newdata	Data frame for prediction. If missing, predictions are made for the original data.
se	Logical; if TRUE, computes standard errors and confidence intervals.
response	Logical; if TRUE (default), returns probabilities only for the observed response (if newdata contains the response). If FALSE, returns probabilities for all categories.
Y	Vector of outcome levels to predict probabilities for (optional).
level	Confidence level for intervals (default 0.95).
...	Additional arguments.

Value

A matrix or data frame containing:

pred	Predicted probabilities.
se	Standard errors (if se=TRUE).
lower, upper	Confidence intervals (if se=TRUE).

If response=FALSE, columns correspond to the levels of the outcome factor.

Author(s)

Thomas Scheike

See Also

mlogit

---

predict.phreg *Predictions from Proportional Hazards Model*

Description

Computes predictions for survival probability, cumulative hazard, or risk at specified time points for new data or existing data. Includes standard errors and confidence intervals.

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`	Object of class `"phreg"`.
`newdata`	Data frame for prediction. If `NULL`, predictions are made for the original data.
`times`	Time points for prediction. Defaults to all unique event times in the model.
`individual.time`	Logical; if `TRUE`, uses one time per subject (requires `newdata` and `times` to be same length).
`tminus`	Logical; if `TRUE`, predicts at $t-$ (strictly before t).
`se`	Logical; if `TRUE`, computes standard errors and confidence intervals.
`robust`	Logical; if `TRUE`, uses robust standard errors (default for most functions).
`conf.type`	Transformation for survival estimates: `"log"` (default) or `"plain"`.
`conf.int`	Confidence level (default 0.95).
`km`	Logical; if `TRUE`, uses Kaplan-Meier product-limit for baseline; otherwise uses exponential of cumulative baseline.
`...`	Additional arguments for plotting functions.

Value

An object of class `"predictphreg"` containing:

`surr`	Matrix of survival probabilities.
`cumhaz`	Matrix of cumulative hazards.
`cif`	Matrix of cumulative incidence functions (if applicable).
`times`	Vector of time points.
`surv.upper`, `surv.lower`	Confidence bounds for survival.
`RR`	Relative risks.

Author(s)

Thomas Scheike

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

print.summary_binregCV_multi *Print method for summary_binregCV_multi objects*

Description

Print method for summary_binregCV_multi objects

Usage

```
## S3 method for class 'summary_binregCV_multi'
print(x, digits = 4, ...)
```

Arguments

x	A ‘summary_binregCV_multi‘ object.
digits	Number of digits to round to.
...	Not currently used.

prob_exceed_recurrent *Estimate the probability of exceeding k recurrent events by time t*

Description

Estimates $P(N(t) \geq k)$ as a function of time t, for a range of thresholds k, in the presence of a terminal event (death). The estimator is based on the cumulative incidence of "reaching k events", treating death as a competing risk. Confidence intervals are computed on the log or plain scale.

Usage

```
prob_exceed_recurrent(
  formula,
  data,
  cause = 1,
  death.code = 2,
  cens.code = 0,
  exceed = NULL,
  marks = NULL,
  all.cifs = FALSE,
  return.data = FALSE,
  conf.type = c("log", "plain"),
  level = 0.95,
  ...
)
```

Arguments

formula	A formula with an Event response giving the exit time and status (and optionally entry time). The right-hand side may include cluster() and strata().
data	A data frame containing all variables in formula.
cause	Integer code for the recurrent event of interest. Default is 1.
death.code	Integer code for the terminal event. Default is 2.
cens.code	Integer code for censoring. Default is 0.
exceed	Integer vector of thresholds k to evaluate. If NULL (default), all observed cumulative counts are used.
marks	Optional numeric vector of event weights. If non-NULL, cumulative counts are formed as weighted sums of events rather than simple counts. Must have the same length as nrow(data).
all.cifs	Logical. If TRUE, the fitted cif object for each threshold is returned in cif.exceed. Default is FALSE.
return.data	Logical. If TRUE, the constructed dataset for each threshold is returned in dataList. Default is FALSE.
conf.type	Type of confidence interval transformation: "log" (default) or "plain".
level	Confidence level. Default is 0.95.
...	Further arguments passed to cif.

Details

For each threshold k in `exceed`, the function identifies the first time each subject reaches k events, then fits a competing risks model (`cif`) with "reaching k events" as the event of interest and death as the competing event. Strata are supported. When `marks` is `NULL`, each event contributes equally; otherwise events are weighted by their mark values before cumulative counts are formed.

Value

An object of class `"exceed"` with the following components:

`time`	Vector of evaluation time points.
`prob`	Array of dimension `(length(time), length(exceed) + 1, nstrata)` containing $P(N(t) \geq k)$ for each threshold and stratum. The first column gives $P(N(t) <$ exceed$[1])$.
`se.prob`	Standard errors of `prob`.
`lower`, `upper`	Pointwise confidence interval bounds.
`meanN`	Estimated mean number of events $E(N(t))$ (single stratum only; `NULL` for stratified analyses).
`meanN2`, `varN`	Second moment and variance of $N(t)$ (single stratum only).
`exceed`	Thresholds evaluated (excluding zero).
`cif.exceed`	List of fitted `cif` objects (if `all.cifs = TRUE`).
`dataList`	List of datasets for each threshold (if `return.data = TRUE`).
`nstrata`, `strata.levels`, `strata.name`	Stratification information.

Use `plot()` and `summary()` methods for visualisation and tabulation.

Author(s)

Thomas Scheike

References

Scheike, T. H., Eriksson, L., and Tribler, P. (2019). The mean, variance and correlation for bivariate recurrent events with a terminal event. *Journal of the Royal Statistical Society, Series C*, 68(5).

See Also

`recurrent_marginal`, `cif`

Examples

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

oo <- prob_exceed_recurrent(Event(entry, time, status) ~ cluster(id),
                            hfactioncpx12, cause = 1, death.code = 2)
plot(oo)
summary(oo, times = c(1, 2, 5))
```

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 indpendent, and that the marginal cumulative incidence curves are on the form

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

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 comparision is made with respect to
data	a data.frame with the variables.
cause	specifies the causes related to the death times, the value cens.code is the censoring value.
cif2	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 algorith.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" optimzer, 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 <- sim_nordic_random(1000,delayed=TRUE,cordz=0.5,cormz=2,lam0=0.3,country=TRUE)
times <- seq(50,90,by=10)
add1 <- timereg::comp.risk(Event(time,cause)~-1+factor(country)+cluster(id),data=d,
times=times,cause=1,max.clust=NULL)

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

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

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

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

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

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

`ratioATE` *Ratio of Average Treatment Effects*

Description

Computes the ratio of two Average Treatment Effects (ATEs), typically comparing the ATE for a specific cause (e.g., RMTL due to cause 1) to the ATE for the total RMTL.

Usage

```
ratioATE(rmtl, rmtl1, h = NULL, null = 1)
```

Arguments

`rmtl`	Object containing the ATE for the total RMTL (from `resmeanATE`).
`rmtl1`	Object containing the ATE for the specific cause RMTL (from `resmeanATE`).
`h`	Transformation function (e.g., `log`) applied to the ratio. Default is identity.
`null`	Value under the null hypothesis for the ratio (default 1).

Details

The function transforms the estimates (e.g., using log) to compute the ratio and its standard error using the delta method and the joint influence functions.

Value

A list containing:

`ratioG`	Ratio based on the simple IPCW estimator.
`ratioDR`	Ratio based on the double robust estimator.

Author(s)

Thomas Scheike

See Also

`resmeanATE`

Examples

```
data(bmt); bmt$event <- bmt$cause!=0; dfactor(bmt) <- tcell~tcell
out <- resmeanATE(Event(time,event)~tcell+platelet, data=bmt, time=40, outcome="rmtl")
out1 <- resmeanATE(Event(time,cause)~tcell+platelet, data=bmt, cause=1, time=40, outcome="rmtl")
ratioATE(out, out1, h=log)
```

rchaz *Simulation of Piecewise Constant Hazard Model (Cox)*

Description

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

Usage

```
rchaz(cumhazard, rr, n = NULL, entry = NULL, cause = 1, extend = FALSE)
```

Arguments

cumhazard	Cumulative hazard matrix (columns: time, cumulative hazard), or piece-constant rates for periods defined by the first column of input.
rr	Relative risk vector for simulations, or alternatively when rr=1 specify n.
n	Number of simulations if rr not given.
entry	Delayed entry time for simulations (optional).
cause	Name/code of the event cause (default 1).
extend	To extend piecewise constant with constant rate beyond the last break point. Default is FALSE. If TRUE, extends with average rate over time from cumulative. If numeric, uses the given rate.

Details

For a piecewise linear cumulative hazard, the inverse is easy to compute. With delayed entry x, we compute:

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

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

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

Value

A data frame containing:

entry	Entry times.
time	Event/censoring times.
status	Event status (1=event, 0=censored).
rr	Relative risks used.

Attributes include:

cumhaz	The cumulative hazard used.
extend.rate	The extension rate if used.

Author(s)

Thomas Scheike

Examples

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

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

---

rchazl *Multiple Cause Piecewise Constant Hazard Simulation*

Description

Simulates data from multiple piecewise constant baseline hazards for competing risks. Takes the minimum of all cause-specific event times and assigns the corresponding cause.

Usage

```
rchazl(cumhaz, rr, causes = NULL, ...)
```

Arguments

cumhaz	List of cumulative hazard matrices, one for each cause.
rr	Matrix of relative risks (rows = subjects, columns = causes).
causes	Vector of cause codes to assign (default NULL, uses 1,2,...).
...	Additional arguments passed to rchaz.

Value

A data frame with event times and status indicating the cause of the first event.

Author(s)

Thomas Scheike

See Also

rchaz

`rcrisk` *Simulation of Piecewise constant hazard models with two causes (Cox).*

Description

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

Usage

```
rcrisk(
  cumA,
  cumB,
  rr1 = NULL,
  rr2 = NULL,
  n = NULL,
  cens = NULL,
  rrc = NULL,
  extend = TRUE,
  causes = NULL,
  ...
)
```

Arguments

`cumA`	cumulative hazard of cause 1, or list of multiple cumulative hazards
`cumB`	cumulative hazard of cause 2 or NULL when cumA is a list
`rr1`	number of simulations or vector of relative risk for simuations, or matrix with columns equal to number of hazards in list
`rr2`	number of simulations or vector of relative risk for simuations.
`n`	number of simulation if rr not given, must be given when rr is not given
`cens`	to censor further , rate or cumumlative hazard
`rrc`	retlativ risk for censoring.
`extend`	to extend the cumulative hazards to largest end-point
`causes`	to assign status values for each of the causes, vector of integers
`...`	arguments for rchaz

Author(s)

Thomas Scheike

Examples

```
data(bmt);
n <- 100
cox1 <- phreg(Surv(time,cause==1)~tcell+platelet,data=bmt)
cox2 <- phreg(Surv(time,cause==2)~tcell+platelet,data=bmt)

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

d <-  rcrisk(cox1$cum,cox2$cum,rr1,rr2,cens=2/70)
dd <- cbind(d,Z1)

d <-  rcrisk(cox1$cum,cox2$cum,rr1,rr2,cens=cbind(c(1,30,68),c(.01,1,3)))
dd <- cbind(d,Z1)

par(mfrow=c(1,3))
scox0 <- phreg(Surv(time,status==0)~tcell+platelet,data=dd)
plot(scox0); lines(cbind(c(1,30,68),c(.01,1,3)),col=2)
##
scox1 <- phreg(Surv(time,status==1)~tcell+platelet,data=dd)
scox2 <- phreg(Surv(time,status==2)~tcell+platelet,data=dd)
plot(cox1); plot(scox1,add=TRUE,col=2)
plot(cox2); plot(scox2,add=TRUE,col=2)
cbind(cox1$coef,scox1$coef,cox2$coef,scox2$coef)

# 3 causes and censoring
d3 <- rcrisk(list(cox1$cum,cox2$cum,cox1$cum),NULL,n=100,cens=cbind(c(1,30,68),c(.01,1,3)))
dtable(d3,~status)
```

---

recreg *Recurrent Events Regression with Terminal Event*

Description

Fits a Ghosh-Lin IPCW (Inverse Probability of Censoring Weighted) Cox-type model for recurrent events in the presence of a terminal event (e.g., death).

Usage

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

```
  cens.code = 0,
  cens.model = ~1,
  weights = NULL,
  offset = NULL,
  Gc = NULL,
  wcomp = NULL,
  marks = NULL,
  augmentation.type = c("lindyn.augment", "lin.augment"),
  ...
)
```

Arguments

formula	Formula with an 'Event' outcome.
data	Data frame containing the variables.
cause	Cause of interest (default is 1).
death.code	Codes for the terminal event/death (default is 2).
cens.code	Code for censoring (default is 0).
cens.model	Formula for a stratified Cox model without covariates used to estimate censoring probabilities.
weights	Weights for the score equations.
offset	Offsets for the model.
Gc	Censoring weights for the time argument. If NULL, these are calculated using a Kaplan-Meier estimator (should then provide $G_c(T_i-)$).
wcomp	Weights for composite outcomes (e.g., when cause=c(1,3), wcomp might be c(1,2)).
marks	A mark value vector from the data frame, specifying the mark value at all events.
augmentation.type	Type of augmentation when an augmentation model is given (options: "lindyn.augment", "lin.augment").
...	Additional arguments passed to lower-level functions.

Details

For the Cox-type model, the expectation is modeled as:

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

by solving Cox-type IPCW weighted score equations:

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

where

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

,

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

, and

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

.

The IID decomposition of the beta coefficients takes the form:

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

and is returned as the `iid` component.

Events, deaths, and censorings are specified via a start-stop structure and the `Event` call. The function identifies these via a status vector and cause codes, censoring codes (`cens.code`), and death codes (`death.code`). See examples and vignettes for details.

Value

An object of class `"recreg"` (extending `"phreg"`) containing:

`coef`	Estimated coefficients.
`var`	Robust variance-covariance matrix.
`iid`	Influence functions for the coefficients.
`cumhaz`	Cumulative hazard estimates.
`se.cumhaz`	Standard errors for cumulative hazard.

Author(s)

Thomas Scheike

See Also

`recregIPCW`

Examples

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

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

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

```
glss <- recreg(Event(entry,time,status)~strata(treatment)+cluster(id),data=hf,
cause=1,death.code=2,cens.model=~strata(treatment))
summary(glss)
plot(glss)

## IPCW at 2 years
ll2 <- recregIPCW(Event(entry,time,status)~treatment+cluster(id),data=hf,
cause=1,death.code=2,time=2,cens.model=~strata(treatment))
summary(ll2)

ll2i <- recregIPCW(Event(entry,time,status)~-1+treatment+cluster(id),data=hf,
cause=1,death.code=2,time=2,cens.model=~strata(treatment))
summary(ll2i)
```

`recregIPCW` *IPCW Estimator for Recurrent Events*

Description

Computes the Inverse Probability of Censoring Weighted (IPCW) estimator for the mean number of recurrent events. Supports various estimators including the Ghosh-Lin and Lawless-Cook estimators.

Usage

```
recregIPCW(
  formula,
  data = data,
  cause = 1,
  cens.code = 0,
  death.code = 2,
  cens.model = ~1,
  km = TRUE,
  times = NULL,
  beta = NULL,
  offset = NULL,
  type = c("II", "I"),
  marks = NULL,
  weights = NULL,
  model = c("exp", "lin"),
  no.opt = FALSE,
  augmentation = NULL,
  method = "nr",
  se = TRUE,
  ...
)
```

Arguments

formula	Formula with an 'Event' outcome.
data	Data frame.
cause	Cause of interest (default is 1).
cens.code	Censoring code (default is 0).
death.code	Death code (default is 2).
cens.model	Formula for the censoring model (default is ~1).
km	Logical; if TRUE, uses Kaplan-Meier for censoring weights; otherwise uses Cox model.
times	Time points for estimation (required).
beta	Initial values for coefficients (optional).
offset	Offsets.
type	Type of estimator: "II" (default) or "I".
marks	Mark values.
weights	Weights.
model	Model type for the mean: "exp" (default) or "lin".
no.opt	Logical; if TRUE, skips optimization.
augmentation	Augmentation terms.
method	Optimization method (default is "nr").
se	Logical; if TRUE, computes standard errors.
...	Additional arguments.

Value

An object of class "binreg" (extending "resmean") containing:

coef	Estimated coefficients.
var	Variance-covariance matrix.
iid	Influence functions.
times	Time points.
Y	Observed IPCW weighted counts (increment IPCW) $\int_0^t (1/G_c(s))dN(s)$

.

Author(s)

Thomas Scheike

See Also

recreg

`recurrent_marginal` *Marginal mean estimation for recurrent events with a terminal event*

Description

Estimates the marginal mean number of recurrent events over time in the presence of a competing terminal event (e.g. death), using the nonparametric estimator of Ghosh and Lin (2000). Two proportional hazards models are fitted internally—one for the recurrent event rate and one for the terminal event—and combined to form the estimator

$$\mu(t) = \int_0^t S(u-)\, dR(u),$$

where $S(u)$ is the marginal survival probability at the baseline covariate level and $dR(u)$ is the baseline recurrent event rate among survivors. Robust (sandwich) standard errors are computed via the influence-function approach of Ghosh and Lin (2000).

Usage

```
recurrent_marginal(formula, data, cause = 1, ..., death.code = 2, test = FALSE)

recurrentMarginal(formula, data, ...)
```

Arguments

`formula`	A formula with an `Event` response on the left-hand side, specifying entry time, exit time, and event status. The right-hand side may include `cluster()` to identify subjects and `strata()` for a stratified analysis. A `cluster()` term is required.
`data`	A data frame containing all variables in `formula`.
`cause`	Integer code(s) for the recurrent event of interest. Default is `1`.
`...`	Further arguments passed to `phreg`.
`death.code`	Integer code(s) for the terminal event. Default is `2`.
`test`	Logical. If `TRUE`, a logrank-type test comparing strata is computed and stored as an attribute of the result. Default is `FALSE`.

Details

Jump times must be unique within each stratum. If ties are present, use `tie_breaker` to resolve them before calling this function.

Value

An object of class `"recurrent"` with the following components:

`mu`	Estimated marginal mean $\mu(t)$ at each jump time.

`se.mu`	Robust standard error of `mu`.
`times`	Jump times at which estimates are computed.
`St`	Marginal survival estimate $S(t)$ at each jump time.
`cumhaz`	Two-column matrix of `(time, mu)`, suitable for plotting.
`se.cumhaz`	Two-column matrix of `(time, se.mu)`.

The object carries three attributes: `"logrank"` (the test result when `test = TRUE`, otherwise `NULL`), `"cause"`, and `"death.code"`.

Author(s)

Thomas Scheike

References

Cook, R. J. and Lawless, J. F. (1997). Marginal analysis of recurrent events and a terminating event. *Statistics in Medicine*, 16, 911–924.

Ghosh, D. and Lin, D. Y. (2000). Nonparametric analysis of recurrent events and death. *Biometrics*, 56, 554–562.

See Also

`test_logrankRecurrent`, `tie_breaker`, `prob_exceed_recurrent`

Examples

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

## Fit nonparametric baseline models for recurrent events and death
xr <- phreg(Surv(entry, time, status == 1) ~ cluster(id), data = hf)
dr <- phreg(Surv(entry, time, status == 2) ~ cluster(id), data = hf)

par(mfrow = c(1, 3))
plot(dr, se = TRUE); title(main = "Death")
plot(xr, se = TRUE); title(main = "Recurrent events")

## Compare naive and robust standard errors for the recurrent event rate
rxr <- robust_phreg(xr, fixbeta = 1)
plot(rxr, se = TRUE, robust = TRUE, add = TRUE, col = 4)

## Marginal mean via formula interface
outN <- recurrent_marginal(Event(entry, time, status) ~ cluster(id),
                           data = hf, cause = 1, death.code = 2)
plot(outN, se = TRUE, col = 2, add = TRUE)
summary(outN, times = 1:5)

## Stratified analysis with logrank test
out <- recurrent_marginal(Event(entry, time, status) ~ strata(treatment) + cluster(id),
```

```
                          data = hf, cause = 1, death.code = 2, test = TRUE)
plot(out, se = TRUE, ylab = "Marginal mean", col = 1:2)
attr(out, "logrank")
summary(out, times = 1:5)

## Influence-function (iid) decomposition at a fixed time point
head(iid(outN, time = 3))
```

---

resmean_phreg *Restricted Mean for Stratified Kaplan-Meier or Cox Model*

Description

Computes the Restricted Mean Survival Time (RMST) for stratified Kaplan-Meier or stratified Cox models with martingale standard errors.

Usage

```
resmean_phreg(x, times = NULL, covs = NULL, ...)
```

Arguments

x	Object of class "phreg".
times	Possible times for which to report restricted mean. If NULL, reports for all event times.
covs	Possible covariates for Cox model adjustment.
...	Additional arguments passed to lower-level functions.

Details

The standard error is computed using linear interpolation between standard errors at jump-times. This allows plotting the restricted mean as a function of time.

Years lost can be computed based on this and decomposed into years lost for different causes using the cif_yearslost function.

Value

An object of class "resmean_phreg" containing:

rmst	Matrix of restricted mean survival times.
se.rmst	Standard errors for RMST.
intkmtimes	Restricted mean at specified times.
years.lost	Years lost (if applicable).

Author(s)

Thomas Scheike

Examples

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

rm1 <- resmean_phreg(out1, times = 10 * (1:6))
summary(rm1)
e1 <- estimate(rm1)
par(mfrow = c(1, 2))
plot(rm1, se = 1)
plot(rm1, years.lost = TRUE, se = 1)

## Comparing populations
rm1 <- resmean_phreg(out1, times = 40)
e1 <- estimate(rm1)
estimate(e1, rbind(c(1, -1, 0, 0)))
```

---

resmeanATE *Average Treatment Effect for Restricted Mean Time*

Description

Estimates the Average Treatment Effect (ATE) for Restricted Mean Survival Time (RMST) or Restricted Mean Time Lost (RMTL) in censored competing risks data using IPCW.

Usage

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

Arguments

formula	Formula with an Event outcome. The first covariate must be the treatment factor.
data	Data frame.
model	Link function: "exp" (exponential) or "lin" (identity).
outcome	Outcome type: "rmst" or "rmtl".
...	Additional arguments passed to binregATE, such as time, treat.model, augmentR0, augmentC, etc.

Details

Under standard causal assumptions (Consistency, Ignorability, Positivity), the ATE is estimated as $E(Y(1) - Y(0))$, where $Y(a)$ is the potential outcome under treatment a. The method uses double robust estimating equations that are IPCW-adjusted for censoring.

The first covariate in the formula must be the treatment effect (a factor). If the factor has more than two levels, multinomial logistic regression (mlogit) is used for propensity score modeling.

Value

An object of class `"binregATE"` containing:

`riskG`	Simple IPCW estimator results.
`riskDR`	Double Robust estimator results.
`riskG.iid`, `riskDR.iid`	Influence functions.
`coef`	Treatment effect estimates.
`se`	Standard errors.

Author(s)

Thomas Scheike

References

Scheike, T. and Holst, K. K. (2024). Restricted mean time lost for survival and competing risks data using mets in R. WIP.

Scheike, T. and Tanaka, S. (2025). Restricted mean time lost ratio regression: Percentage of restricted mean time lost due to specific cause. WIP.

See Also

binregATE, ratioATE

Examples

```
data(bmt); bmt$event <- bmt$cause!=0; dfactor(bmt) <- tcell~tcell

out <- resmeanATE(Event(time,event)~tcell+platelet, data=bmt, time=40,
                  treat.model=tcell~platelet, outcome="rmtl")
summary(out)

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

`resmeanIPCW` *Restricted IPCW Mean for Censored Survival Data*

Description

Provides a fast implementation of Inverse Probability of Censoring Weighting (IPCW) regression for a single time point. It fits the model:

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

or, in the case of competing risks data:

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

which represents the "Years Lost Due to Cause" (RMTL).

Usage

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

Arguments

`formula`	Formula with an `Event` outcome (e.g., `Event(time, cause)`).
`data`	Data frame containing the variables.
`outcome`	Outcome type: `"rmst"` (Restricted Mean Survival Time) or `"rmtl"` (Restricted Mean Time Lost).
`...`	Additional arguments passed to `binreg`, such as `time`, `cause`, `cens.model`, `model`, `type`, etc.

Details

The method solves the binomial regression IPCW response estimating equation:

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

where $\Delta(\min(T,t)) = I(\min(T,t) \leq C)$ is the indicator of being uncensored at the time of interest.

When the status variable is binary, the outcome is assumed to be $Y = \min(T,t)$ (RMST). If the status has more than two levels (competing risks), the outcome is $Y = (t - \min(T,t))I(\text{status} = \text{cause})$ (RMTL for a specific cause).

The function supports:

- **IPCW Adjustment**: Weights by the inverse of the censoring survival probability.
- **Augmentation**: Can include an augmentation term (type="II" or "III") to improve efficiency and robustness (Double Robust estimation).
- **Variance Estimation**: Based on the influence function, including adjustments for the estimation of the censoring model.

Value

An object of class "binreg" containing:

coef	Coefficient estimates.
se	Standard errors.
var	Variance-covariance matrix.
iid	Influence function decomposition.
naive.var	Variance under known censoring model (if applicable).
time	Time point used.
outcome	Type of outcome analyzed.

Author(s)

Thomas Scheike

References

Scheike, T. and Holst, K. K. (2024). Restricted mean time lost for survival and competing risks data using mets in R. WIP.

See Also

binreg, resmeanATE, rmstIPCW

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)

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

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

### Same as Kaplan-Meier for full censoring model
bmt$int <- with(bmt, strata(tcell, platelet))
out <- resmeanIPCW(Event(time,cause!=0)~-1+int, bmt, time=30,
                              cens.model=~strata(platelet, tcell), model="lin")
estimate(out)
```

```
out1 <- phreg(Surv(time,cause!=0)~strata(tcell,platelet), data=bmt)
rm1 <- resmean_phreg(out1, times=30)
summary(rm1)

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

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

`robust.basehaz.phreg` *Robust Baseline Hazard Standard Errors*

Description

Computes robust (sandwich) standard errors for the cumulative baseline hazard from a `phreg` object.

Summarizes cumulative baseline hazard estimates from a phreg object, optionally with robust standard errors.

Computes confidence intervals using log or plain transformations, with optional restrictions to positive values or probability scale.

Usage

```
robust.basehaz.phreg(x, type = "robust", fixbeta = NULL, ...)

summarybase.phreg(object, robust = FALSE, ...)

conftype(
  x,
  std.err,
  conf.type = c("log", "plain"),
  restrict = c("positive", "prob", "none"),
  conf.int = 0.95
)
```

Arguments

`x`	point estimate(s).
`type`	type of standard error (default `"robust"`).

fixbeta	if non-NULL, fixes beta at given value.
...	additional arguments.
object	a phreg object.
robust	logical; if TRUE, uses robust standard errors.
std.err	standard error(s).
conf.type	type of transformation: "log" or "plain".
restrict	restriction: "positive", "prob", or "none".
conf.int	confidence level (default 0.95).

Value

A list with cumhaz, se.cumhaz, and strata.

An object of class "summary.recurrent".

A list with upper, lower, conf.type, and conf.int.

---

rr_cif *Non-parametric Cumulative Incidence Functions*

Description

Functions for computing and visualizing non-parametric cumulative incidence estimates, as well as dependence measures (odds ratio, relative risk) for bivariate competing risks data.

Usage

```
rr_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,
  step = 1,
  sym = 0,
  weights = NULL,
  same.cens = FALSE,
```

```
  censoring.weights = NULL,
  silent = 1,
  par.func = NULL,
  dpar.func = NULL,
  dimpar = NULL,
  score.method = "nlminb",
  entry = NULL,
  estimator = 1,
  trunkp = 1,
  admin.cens = NULL,
  ...
)

or_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,
  step = 1,
  sym = 0,
  weights = NULL,
  same.cens = FALSE,
  censoring.weights = NULL,
  silent = 1,
  par.func = NULL,
  dpar.func = NULL,
  dimpar = NULL,
  score.method = "nlminb",
  entry = NULL,
  estimator = 1,
  trunkp = 1,
  admin.cens = NULL,
  ...
)

random.cif(cif, ...)

Grandom.cif(cif, ...)
```

```
predictPairPlack(cif1, cif2, status1, status2, theta)

npc(T, cause, same.cens = TRUE, sep = FALSE)

nonparcuminc(t, status, cens = 0)

plotcr(
  x,
  col,
  lty,
  legend = TRUE,
  which = 1:2,
  cause = 1:2,
  ask = prod(par("mfcol")) < length(which) && dev.interactive(),
  ...
)
```

Arguments

cif	a cumulative incidence model object (from timereg).
data	a data.frame with the variables.
cause	causes to plot.
cif2	optional second CIF model if different from first.
times	time points for evaluation.
cause1	cause for first coordinate.
cause2	cause for second coordinate.
cens.code	censoring code value.
cens.model	censoring model type (default "KM").
Nit	maximum number of iterations.
detail	level of output detail.
clusters	cluster variable name or vector.
theta	dependence parameter(s).
theta.des	design matrix for theta.
step	step size for optimization.
sym	if 1, symmetric dependence structure.
weights	optional weights.
same.cens	logical; if TRUE, uses joint censoring weights.
censoring.weights	optional pre-computed censoring weights.
silent	verbosity level.
par.func	optional parameter function.

dpar.func	optional derivative of parameter function.
dimpar	dimension of parameter vector.
score.method	optimization method (default "nlminb").
entry	optional entry time variable.
estimator	estimator type.
trunkp	truncation probability.
admin.cens	administrative censoring time.
...	additional arguments.
cif1	CIF values for subject 1 (for predictPairPlack).
status1	status for subject 1.
status2	status for subject 2.
T	matrix with columns: time1, time2, status1, status2 (for npc).
sep	logical; if TRUE, uses separate censoring models for each subject.
t	vector of event/censoring times (for nonparcuminc).
status	vector of status codes (for nonparcuminc).
cens	censoring code (default 0).
x	data matrix or competing risks object.
col	colors for curves.
lty	line types for curves.
legend	logical; if TRUE, add legend.
which	which plots to show.
ask	logical; if TRUE, prompt before new page.

Details

npc computes bivariate non-parametric cumulative incidence using inverse-probability-of-censoring weights.

nonparcuminc computes univariate non-parametric cumulative incidence for multiple causes.

plotcr plots cumulative incidence curves for competing risks using the prodlim package.

or_cif fits an odds-ratio model for bivariate cumulative incidence.

rr_cif fits a relative-risk model for bivariate cumulative incidence.

random.cif and Grandom.cif are aliases for random_cif and Grandom_cif (random effects CIF models).

predictPairPlack predicts pairwise joint probabilities under a Plackett (odds-ratio) dependence model.

matplot.mets.twostage produces matrix-plots of concordance over time from a twostage object.

Value

For npc: matrix with columns (time, cumulative incidence). For nonparcuminc: matrix with time and cause-specific cumulative incidences.

Author(s)

Klaus K. Holst, Thomas Scheike

`rweibullcox` *Simulate observations from a Weibull distribution*

Description

Simulate observations from the model with cumulative hazard given by

$$\Lambda(t) = \lambda \cdot t^s$$

where λ is the *rate parameter* and s is the *shape parameter*.

Usage

```
rweibullcox(n, rate, shape)
```

Arguments

`n`	(integer) number of observations
`rate`	(numeric) rate parameter (can be a vector of size n)
`shape`	(numeric) shape parameter (can be a vector of size n)

Details

[stats::rweibull()] uses a different parametrization with cumulative hazard given by

$$H(t) = (t/b)^a,$$

i.e., the shape is the same $a := s$ but the scale paramter b is related to rate paramter r by

$$r := b^{-a}$$

See Also

[stats::rweibull()]

`sim_cif` *Simulation of Output from Cumulative Incidence Regression Model*

Description

Simulates data that looks like fit from a fitted cumulative incidence model (Fine-Gray or logistic).

Usage

```
sim_cif(
  cif,
  n,
  data = NULL,
  Z = NULL,
  rr = NULL,
  strata = NULL,
  drawZ = TRUE,
  cens = NULL,
  rrc = NULL,
  entry = NULL,
  Sentry = NULL,
  cumstart = c(0, 0),
  U = NULL,
  pU = NULL,
  type = NULL,
  extend = NULL,
  ...
)
```

Arguments

`cif`	Output from `prop.odds.subdist` or `ccr` (`cmprsk`), or call `invsubdist` with cumulative and linear predictor.
`n`	Number of simulations.
`data`	Data frame to extract covariates.
`Z`	Design matrix instead of data.
`rr`	Relative risks.
`strata`	Strata vector.
`drawZ`	Logical; randomly sample from Z.
`cens`	Censoring specification.
`rrc`	Relative risks for censoring.
`entry`	Delayed entry time.
`Sentry`	Survival related to delayed entry.
`cumstart`	Start cumulatives at time 0.

U	Uniforms for drawing timing.
pU	Uniforms for drawing event type.
type	Model type: "logistic", "cloglog", or "rr".
extend	Extend piecewise constant with constant rate.
...	Arguments for sim_subdist.

Value

Data frame with simulated event times, status, and covariates.

Author(s)

Thomas Scheike

See Also

[simul_cifs()]

Examples

```
data(bmt)
nsim <- 100

## logit cumulative incidence regression model
cif <- cifreg(Event(time,cause)~platelet+age,data=bmt,cause=1)
estimate(cif)
plot(cif,col=1)
simbmt <- sim_cif(cif,nsim,data=bmt)
dtable(simbmt,~cause)
scif <- cifreg(Event(time,cause)~platelet+age,data=simbmt,cause=1)
estimate(scif)
plot(scif,add=TRUE,col=2)

## Fine-Gray cloglog cumulative incidence regression model
cif <- cifregFG(Event(time,cause)~strata(tcell)+age,data=bmt,cause=1)
estimate(cif)
plot(cif,col=1)
simbmt <- sim_cif(cif,nsim,data=bmt)
scif <- cifregFG(Event(time,cause)~strata(tcell)+age,data=simbmt,cause=1)
estimate(scif)
plot(scif,add=TRUE,col=2)

###############################################################
#  simulating several causes with specific cumulatives
###############################################################
cif1 <-  cifreg(Event(time,cause)~strata(tcell)+age,data=bmt,cause=1)
cif2 <-  cifreg(Event(time,cause)~strata(platelet)+tcell+age,data=bmt,cause=2)
cifss <-  list(cif1,cif2)
simbmt <- sim_cifs(list(cif1,cif2),nsim,data=bmt,extend=0.005)
scif1 <-  cifreg(Event(time,cause)~strata(tcell)+age,data=simbmt,cause=1)
scif2 <-  cifreg(Event(time,cause)~strata(platelet)+tcell+age,data=simbmt,cause=2)
```

```
cbind(cif1$coef,scif1$coef)
## can be off due to restriction F1+F2<= 1
cbind(cif2$coef,scif2$coef)

par(mfrow=c(1,2))
## Cause 1 follows the model
plot(cif1); plot(scif1,add=TRUE,col=1:2,lwd=2)
# Cause 2:second cause is modified with restriction to satisfy F1+F2<= 1, so scaled down
plot(cif2); plot(scif2,add=TRUE,col=1:2,lwd=2)
```

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

Description

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

Usage

```
sim_ClaytonOakes(
  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

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

Description

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

Usage

```
sim_ClaytonOakesWei(
  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

`sim_GLcox` *Simulation of Two-Stage Recurrent Events Data*

Description

Simulates data based on Cox/Cox or Cox/Ghosh-Lin structures for recurrent events with a terminal event.

Usage

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

Arguments

`n`	Number of IDs.
`base1`	Baseline cumulative hazard for recurrent events.
`drcumhaz`	Baseline cumulative hazard for the terminal event.
`var.z`	Variance of the gamma frailty.
`r1`	Relative risk term for the recurrent event baseline.
`rd`	Relative risk term for the terminal event.
`rc`	Relative risk term for censoring.
`fz`	Function for transformation of the frailty term.
`fdz`	Function for transformation of the frailty term for death.
`model`	Model type: `"twostage"`, `"frailty"`, or `"shared"`.
`type`	Type of simulation (default depends on `model`).
`share`	Proportion of shared random effects (for partially shared models).
`cens`	Censoring rate for exponential censoring.
`nmin`	Minimum number of points in the time grid (default 100).
`nmax`	Maximum number of points in the time grid (default 1000).

Details

- `type=3`: Generates data from a Cox/Cox two-stage model.
- `type=2`: Generates data from a Ghosh-Lin/Cox model.

If `var.z=0`, data is generated without dependence or frailty. If `model="twostage"`, the default is to generate data from a Ghosh-Lin/Cox model. If `type=3`, data is generated with marginal Cox models (Cox/Cox).

Simulation is based on a linear approximation of the hazard for two-stage models on a time grid. The grid must be sufficiently fine.

Value

A data frame with simulated recurrent events and terminal events, including frailty terms.

Author(s)

Thomas Scheike

References

Scheike (2025), Two-stage recurrent events random effects models, LIDA, to appear.

---

sim_multistate *Simulation of Illness-Death Model*

Description

Simulates data from a full illness-death model with reversible transitions and multiple causes of death. Supports various dependence structures via shared frailties.

Usage

```
sim_multistate(
  n,
  cumhaz,
  cumhaz2,
  death.cumhaz,
  death.cumhaz2,
  rr12 = NULL,
  rr21 = NULL,
  rd13 = NULL,
  rd23 = NULL,
  rrc = NULL,
  gap.time = FALSE,
  max.recurrent = 100,
  dependence = 0,
  var.z = 0.5,
```

```
  cor.mat = NULL,
  cens = NULL,
  extend = TRUE,
  ...
)
```

Arguments

n	Number of IDs.
cumhaz	Cumulative hazard from state 1 to 2.
cumhaz2	Cumulative hazard from state 2 to 1.
death.cumhaz	Cumulative hazard of death from state 1.
death.cumhaz2	Cumulative hazard of death from state 2.
rr12	Relative risk for 1->2.
rr21	Relative risk for 2->1.
rd13	Relative risk for death 1->3.
rd23	Relative risk for death 2->3.
rrc	Relative risk for censoring.
gap.time	Gap time indicator. If true simulates gap-times with specified cumulative hazard.
max.recurrent	Maximum recurrent events.
dependence	Dependence structure (0-3).
var.z	Variance of random effects.
cor.mat	Correlation matrix.
cens	Censoring rate.
extend	Extend hazards.
...	Additional arguments.

Value

Data frame with multi-state event history.

Author(s)

Thomas Scheike

Examples

```
########################################
## getting some rates to mimick
########################################
data(CPH_HPN_CRBSI)
dr <- CPH_HPN_CRBSI$terminal
base1 <- CPH_HPN_CRBSI$crbsi
base4 <- CPH_HPN_CRBSI$mechanical
dr2 <- scalecumhaz(dr,1.5)
```

```
cens <- rbind(c(0,0),c(2000,0.5),c(5110,3))

iddata <- sim_multistate(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))
plot(c0)
lines(cens,col=2)
plot(c3,main="rates 1-> 3 , 2->3")
lines(dr,col=1,lwd=2)
lines(dr2,col=2,lwd=2)
###
plot(c1,main="rate 1->2")
lines(base1,lwd=2)
###
plot(c2,main="rate 2->1")
lines(base1,lwd=2)
```

`sim_multistateII` *Illness-Death Competing Risks with Two Causes of Death*

Description

Simulates data from an illness-death model with two causes of death from the illness state. Covariate effects can be introduced via relative risk terms.

Usage

```
sim_multistateII(
  cumhaz,
  death.cumhaz,
  death.cumhaz2,
  n = NULL,
  rr = NULL,
  rd = NULL,
  rd2 = NULL,
  gamma23 = 0,
  gamma24 = 0,
  early2 = 10000,
  gap.time = FALSE,
  max.recurrent = 100,
  cens = NULL,
```

```
  rrc = NULL,
  extend = TRUE,
  ...
)
```

Arguments

cumhaz	Cumulative hazard from state 1 to 2.
death.cumhaz	Cumulative hazard of death from state 1.
death.cumhaz2	Cumulative hazard of death from state 2.
n	Number of simulations.
rr	Relative risks.
rd	Relative risks for death from state 1.
rd2	Relative risks for death from state 2.
gamma23	Early effect parameters for death causes.
gamma24	Early effect parameters for death causes.
early2	Time threshold for early effect.
gap.time	Gap time indicator.
max.recurrent	Maximum recurrent events.
cens	Censoring specification.
rrc	Censoring relative risks.
extend	Extend hazards.
...	Additional arguments.

Value

Data frame with multi-state event history.

Author(s)

Thomas Scheike

---

sim_phreg *Simulation of Output from Cox Model*

Description

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

Usage

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

Arguments

cox	Output from coxph or phreg model fitting.
n	Number of simulations.
data	Data frame to extract covariates for simulations (draws from observed covariates).
Z	Design matrix instead of data.
rr	Vector of relative risks for Cox model.
strata	Vector of strata.
entry	Delayed entry variable for simulation.
extend	Extend possible stratified baselines to largest endpoint.
cens	Censoring specification (matrix = cumulative hazard, scalar = rate).
rrc	Relative risks for Cox-type censoring.
...	Arguments for rchaz (e.g., entry-time).

Value

Data frame with simulated event times, status, and covariates.

Author(s)

Thomas Scheike

Examples

```
data(sTRACE)
nsim <- 100
coxs <-  phreg(Surv(time,status==9)~strata(chf)+vf+wmi,data=sTRACE)
set.seed(100)
sim3 <- sim_phreg(coxs,nsim,data=sTRACE)
head(sim3)
```

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

Z <- sim3[,c("vf","chf","wmi")]
strata <- sim3[,c("chf")]
rr <- exp(as.matrix(Z[,-2]) %*% coef(coxs))
sim4 <- sim_phreg(coxs,nsim,data=NULL,rr=rr,strata=strata)
sim4 <- cbind(sim4,Z)
cc <-   phreg(Surv(time,status)~strata(chf)+vf+wmi,data=sim4)
cbind(coxs$coef,cc$coef)
plot(coxs,col=1); plot(cc,add=TRUE,col=2)
```

`sim_phregs` *Simulation of Cause-Specific Cox Models*

Description

Simulates data that looks like fit from cause-specific Cox models. Censors data automatically. When censoring is given in the list of causes, this provides censoring that looks like the data.

Usage

```
sim_phregs(
  coxs,
  n,
  data = NULL,
  rr = NULL,
  strata = NULL,
  entry = NULL,
  extend = TRUE,
  cens = NULL,
  rrc = NULL,
  ...
)
```

Arguments

`coxs`	List of Cox models.
`n`	Number of simulations.
`data`	Data frame to extract covariates.
`rr`	Relative risks.
`strata`	Strata vector.
`entry`	Delayed entry.
`extend`	Extend baselines to largest endpoint.

cens	Censoring specification.
rrc	Relative risks for censoring.
...	Arguments for rchaz.

Value

Data frame with simulated event times, status, and covariates.

Author(s)

Thomas Scheike

Examples

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

cox1 <- phreg(Surv(time,cause==1)~strata(tcell)+platelet+age,data=bmt)
cox2 <- phreg(Surv(time,cause==2)~tcell+strata(platelet),data=bmt)
coxs <- list(cox1,cox2)
## just calls sim_phregs !
dd <- sim_phregs(coxs,nsim,data=bmt,extend=c(0.001))
scox1 <- phreg(Surv(time,cause==1)~strata(tcell)+platelet+age,data=dd)
scox2 <- phreg(Surv(time,cause==2)~tcell+strata(platelet),data=dd)

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

---

sim_recurrent *Simulate recurrent events with a single event type and a terminal event*

Description

A convenience wrapper around sim_recurrentII for the common case of a single recurrent event type. Frailty and censoring options are passed through to sim_recurrent_list.

Usage

```
sim_recurrent(
  n,
  cumhaz,
  death.cumhaz = NULL,
  r1 = NULL,
  rd = NULL,
```

```
  rc = NULL,
  ...
)
```

Arguments

n	Number of subjects to simulate.
cumhaz	Two-column matrix (time, cumhaz) giving the cumulative hazard of the recurrent event.
death.cumhaz	Two-column matrix (time, cumhaz) giving the cumulative hazard of the terminal event. If NULL, no terminal event is included.
r1	Optional numeric vector of length n of subject-specific relative risks for the recurrent event.
rd	Optional numeric vector of length n of subject-specific relative risks for the terminal event.
rc	Optional numeric vector of length n of subject-specific multipliers for the exponential censoring rate.
...	Further arguments passed to sim_recurrent_list, including dependence, var.z, gap.time, and max.recurrent.

Author(s)

Thomas Scheike

Examples

```
data(CPH_HPN_CRBSI)
dr    <- CPH_HPN_CRBSI$terminal
base1 <- CPH_HPN_CRBSI$crbsi
base4 <- CPH_HPN_CRBSI$mechanical

## Single recurrent event type, with and without terminal event
rr <- sim_recurrent(5, base1)
dlist(rr, . ~ id, n = 0)

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

## Verify that estimated rates recover the true baselines (increase n for precision)
rr <- sim_recurrent(100, base1, death.cumhaz = dr)
par(mfrow = c(1, 3))
mets:::showfitsim(causes = 1, rr, dr, base1, base1)

## Shared frailty across all processes
rr <- sim_recurrent(100, base1, death.cumhaz = dr, dependence = 1, var.z = 0.4)
dtable(rr, ~death + status)

## Two event types; second type uses the mechanical complication rate
set.seed(100)
```

```
rr <- sim_recurrentII(100, base1, base4, death.cumhaz = dr)
dtable(rr, ~death + status)
par(mfrow = c(2, 2))
mets:::showfitsim(causes = 2, rr, dr, base1, base4)

## Three event types and two causes of death via sim_recurrent_list
set.seed(100)
cumhaz <- list(base1, base1, base4)
drl    <- list(dr, base4)
rr     <- sim_recurrent_list(100, cumhaz, death.cumhaz = drl, dependence = 0)
dtable(rr, ~death + status)
mets:::showfitsimList(rr, cumhaz, drl)
```

`sim_recurrent_ts` *Simulate recurrent events from a two-stage Cox or Ghosh-Lin model*

Description

Simulates recurrent event data from a fitted two-stage model, where the recurrent event process and the terminal event are each described by a separate fitted model. The recurrent event model may be either a Cox proportional hazards model (`phreg`) or a Ghosh-Lin marginal rate model (`recreg`); the terminal event model must be a Cox model (`phreg`).

Usage

```
sim_recurrent_ts(
  cox1,
  coxd = NULL,
  n = 1,
  data = NULL,
  type = c("default", "cox-cox", "gl-cox"),
  id = "id",
  varz = 1,
  share = 1,
  cens = 0.001,
  scale1 = 1,
  scaled = 1,
  dependence = NULL,
  r1 = NULL,
  rd = NULL,
  rc = NULL,
  strata1 = NULL,
  stratad = NULL,
  death.code = 3,
  ...
)
```

Arguments

cox1	A fitted phreg object for the recurrent event rate, or a fitted recreg (Ghosh-Lin) object. The model type is detected automatically from the class.
coxd	A fitted phreg object for the terminal event. May be NULL if no terminal event is modelled.
n	Number of subjects to simulate. Default is 1.
data	The data frame on which cox1 and coxd were fitted, used to draw covariate values for the simulated subjects. If NULL, covariates must be supplied via r1, rd, strata1, and stratad.
type	Simulation type: "default" (auto-detected from class of cox1), "cox-cox", or "gl-cox".
id	Name of the subject identifier variable in data. Default is "id".
varz	Variance of the frailty distribution in the two-stage model. Default is 1.
share	Proportion of the shared frailty assigned to the recurrent event process in the partial-sharing model. Default is 1.
cens	Rate of exponential censoring. Default is 0.001.
scale1	Scalar multiplier for the baseline cumulative hazard of the recurrent event process. Default is 1.
scaled	Scalar multiplier for the baseline cumulative hazard of the terminal event. Default is 1.
dependence	If non-NULL, falls back to sim_recurrent_list with this frailty structure (see sim_recurrentII for valid values). Default is NULL.
r1	Optional numeric vector of length n of subject-specific relative risks for the recurrent event, used when data = NULL.
rd	Optional numeric vector of length n of subject-specific relative risks for the terminal event, used when data = NULL.
rc	Optional numeric vector of length n of subject-specific censoring rate multipliers.
strata1	Optional integer vector of length n specifying the stratum index (0-based) for the recurrent event model, used when data = NULL.
stratad	Optional integer vector of length n specifying the stratum index (0-based) for the terminal event model, used when data = NULL.
death.code	Integer status code used for the terminal event in the output status column. Default is 3.
...	Further arguments passed to sim_GLcox, including nmin and nmax for the linear approximation grid.

Details

Covariates are drawn by bootstrap from data (if supplied), and subject-specific relative risks and strata are derived from the fitted model objects. Stratified baselines are fully supported. When dependence is NULL (default), the simulation uses the two-stage structure from sim_GLcox; setting dependence to an integer falls back to sim_recurrent_list with the corresponding frailty model.

Value

A data frame in counting-process format with one row per event interval per subject. Column names match those in the original model formula (entry, exit, and status variables). Additional columns include `id` and the covariates drawn from `data` (if supplied). The terminal event is coded as `death.code` in the status variable; recurrent events are coded as 1.

Author(s)

Thomas Scheike

References

Scheike, T. H. (2026). Two-stage recurrent events random effects models. *Lifetime Data Analysis*.

See Also

`recurrent_marginal`, `sim_recurrent_list`, `sim_GLcox`

Examples

```
data(hfactioncpx12)
hf    <- hfactioncpx12
hf$x  <- as.numeric(hf$treatment)
n     <- 100

## Cox-Cox two-stage model
xr <- phreg(Surv(entry, time, status == 1) ~ x + cluster(id), data = hf)
dr <- phreg(Surv(entry, time, status == 2) ~ x + cluster(id), data = hf)
simcoxcox <- sim_recurrent_ts(xr, dr, n = n, data = hf, death.code = 2)

## Ghosh-Lin/Cox two-stage model
recGL  <- recreg(Event(entry, time, status) ~ x + cluster(id), hf, death.code = 2)
simglcox <- sim_recurrent_ts(recGL, dr, n = n, data = hf, death.code = 2)
```

---

sim_recurrentII *Simulate recurrent events with two event types and a terminal event*

Description

Simulates recurrent event data with up to two distinct event types and an optional terminal event (death), based on user-supplied cumulative hazard functions. Dependence between processes can be introduced via shared or correlated gamma-distributed frailties.

Usage

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

Arguments

n	Number of subjects to simulate.
cumhaz	Two-column matrix (time, cumhaz) giving the cumulative hazard of the first type of recurrent event.
cumhaz2	Two-column matrix (time, cumhaz) giving the cumulative hazard of the second type of recurrent event.
death.cumhaz	Two-column matrix (time, cumhaz) giving the cumulative hazard of the terminal event. If NULL, no terminal event is simulated and follow-up ends at the end of cumhaz.
r1	Optional numeric vector of length n with subject-specific relative risk multipliers for the first event type.
r2	Optional numeric vector of length n with subject-specific relative risk multipliers for the second event type.
rd	Optional numeric vector of length n with subject-specific relative risk multipliers for the terminal event.
rc	Optional numeric vector of length n with subject-specific multipliers for the exponential censoring rate.
dependence	Integer specifying the frailty structure. One of 0 (independence), 1 (shared gamma frailty), or 4 (shared frailty for recurrent events only). See Details.
var.z	Variance of the gamma-distributed frailty. Default is 1.
cor.mat	Correlation matrix for the random effects. Used when dependence = 2 (in sim_recurrent_list).
cens	Rate of exponential censoring. If NULL (default), no additional censoring is applied.
gap.time	Logical. If TRUE, event times are drawn as gap times (time since the last event) rather than calendar times. Default is FALSE.

max.recurrent	Maximum number of recurrent events allowed per subject. Default is 100.
...	Further arguments passed to sim_recurrent_list.

Details

The simulation proceeds by sequentially drawing the next event time from the specified cumulative hazards, taking the minimum of the two recurrent event times, and stopping each subject at death or administrative censoring.

Dependence between processes is controlled by dependence:

0 Independence: all subjects have frailty fixed at 1.

1 Shared frailty: all processes share a single gamma-distributed random effect with mean 1 and variance var.z.

4 Recurrent-event frailty only: the two recurrent event processes share a gamma frailty but the terminal event is independent.

For more complex correlation structures across two event types and death, use sim_recurrentTS.

Value

A data frame in counting-process format (one row per event interval per subject) with columns:

id	Subject identifier.
start, entry	Interval start time.
stop, time	Interval end time (event or censoring time).
status	Event type at stop: 1 or 2 for a recurrent event of the corresponding type, 0 for censoring.
death	Indicator for a terminal event (1) or censoring/survival (0).

Attributes "cumhaz", "death.cumhaz", "rr", and "rd" store the inputs used for simulation.

Author(s)

Thomas Scheike

See Also

sim_recurrent, sim_recurrent_list, sim_recurrentTS

Examples

```
data(CPH_HPN_CRBSI)
dr    <- CPH_HPN_CRBSI$terminal
base1 <- CPH_HPN_CRBSI$crbsi
base4 <- CPH_HPN_CRBSI$mechanical

## Single recurrent event type, with and without terminal event
rr <- sim_recurrent(5, base1)
dlist(rr, . ~ id, n = 0)
```

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

## Verify that estimated rates recover the true baselines (increase n for precision)
rr <- sim_recurrent(100, base1, death.cumhaz = dr)
par(mfrow = c(1, 3))
mets:::showfitsim(causes = 1, rr, dr, base1, base1)

## Shared frailty across all processes
rr <- sim_recurrent(100, base1, death.cumhaz = dr, dependence = 1, var.z = 0.4)
dtable(rr, ~death + status)

## Two event types; second type uses the mechanical complication rate
set.seed(100)
rr <- sim_recurrentII(100, base1, base4, death.cumhaz = dr)
dtable(rr, ~death + status)
par(mfrow = c(2, 2))
mets:::showfitsim(causes = 2, rr, dr, base1, base4)

## Three event types and two causes of death via sim_recurrent_list
set.seed(100)
cumhaz <- list(base1, base1, base4)
drl    <- list(dr, base4)
rr     <- sim_recurrent_list(100, cumhaz, death.cumhaz = drl, dependence = 0)
dtable(rr, ~death + status)
mets:::showfitsimList(rr, cumhaz, drl)
```

`sim_recurrentTS` *Simulate recurrent events from a two-stage model with structured gamma frailties*

Description

Simulates recurrent event data with two event types and a terminal event, using a parametric two-stage frailty model. The construction ensures that the marginal rates are approximately correct: conditional on survival, $E(dN_j \mid D > t) \approx$ `cumhazj`, and the hazard of death equals `death.cumhaz`.

Usage

```
sim_recurrentTS(
  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 subjects to simulate.
cumhaz	Two-column matrix (time, cumhaz) giving the target marginal cumulative rate of the first recurrent event type.
cumhaz2	Two-column matrix (time, cumhaz) giving the target marginal cumulative rate of the second recurrent event type.
death.cumhaz	Two-column matrix (time, cumhaz) giving the cumulative hazard of the terminal event.
nu	Numeric vector of length 3: the powers (ν_1, ν_2, ν_3) applied to the frailty components (see Details). Must satisfy $\nu_j > -1/\text{shape}$. Default is rep(1, 3).
share1	Proportion of the total death frailty variance assigned to the first component Z_{d1}. The remainder goes to Z_{d2}. Must be in $(0, 1)$. Default is 0.3.
vargamD	Total variance of the death frailty Z_{death}. Default is 2.
vargam12	Variance of the shared recurrent-event frailty Z_{12}. Default is 0.5.
gap.time	Logical. If TRUE, event times are drawn as gap times rather than calendar times. Default is FALSE.
max.recurrent	Maximum number of recurrent events per subject. Default is 100.
cens	Rate of exponential censoring. If NULL (default), no administrative censoring is applied.
...	Further arguments passed to lower-level simulation functions.

Details

The frailty structure uses three gamma random variables Z_{d1}, Z_{d2}, Z_{12} to induce dependence:

$$Z_{\text{death}} = Z_{d1} + Z_{d2}, \quad Z_1 = Z_{d1}^{\nu_1} Z_{12}, \quad Z_2 = Z_{d2}^{\nu_2} Z_{12}^{\nu_3}.$$

The parameters share1 and vargamD control how the death frailty splits between the two components, and vargam12 controls the shared recurrent-event frailty. Setting $\nu = (1, 1, 1)$ with share1 = 0.5 gives a symmetric structure; varying ν allows asymmetric dependence.

Value

A data frame in counting-process format (one row per event interval per subject) with columns id, start, stop, entry, time, status, and death. Attributes store the (possibly adjusted) cumulative hazards used in simulation and the frailty parameters.

Author(s)

Thomas Scheike

See Also

sim_recurrentII, sim_recurrent_ts

Examples

```
data(CPH_HPN_CRBSI)
dr    <- CPH_HPN_CRBSI$terminal
base1 <- CPH_HPN_CRBSI$crbsi
base4 <- CPH_HPN_CRBSI$mechanical

rr <- sim_recurrentTS(1000, base1, base4, death.cumhaz = dr)
dtable(rr, ~death + status)
mets:::showfitsim(causes = 2, rr, dr, base1, base4)
```

---

strata-numeric *Stratified Cumulative and Summary Operations*

Description

Low-level helper functions for computing sums, cumulative sums, and reverse cumulative sums within strata groups, as well as matrix double-indexing.

Usage

```
sumstrata(x, strata, nstrata)

cumsumstrata(x, strata, nstrata)

revcumsumstrata(x, strata, nstrata)

revcumsum(x)

matdoubleindex(x, rows, cols, xvec = NULL)

mdi(x, ...)
```

Arguments

x	numeric vector (or matrix for matdoubleindex).
strata	integer vector of strata indices (0-based).
nstrata	number of distinct strata.
rows	row indices for matdoubleindex.
cols	column indices for matdoubleindex.
xvec	optional values to assign at indexed positions.
...	additional arguments (for mdi).

Value

Numeric vector of the same length as `x` (or modified matrix).

Author(s)

Klaus K. Holst, Thomas Scheike

`summary.binregCV` *Summary method for binregCV objects*

Description

Summary method for binregCV objects

Usage

```
## S3 method for class 'binregCV'
summary(
  object,
  transform = NULL,
  conf.level = 0.95,
  type = c("cv", "both"),
  ...
)
```

Arguments

`object`	A `binregCV` object, as returned by `brier_binreg` for a single evaluation time.
`transform`	Optional back-transformation function applied to Brier scores and confidence limits, e.g. `exp` to undo the internal log scale.
`conf.level`	Confidence level for the reported intervals (default 0.95).
`type`	Either `"cv"` (default) for cross-validated Brier scores only, or `"both"` to also include apparent (in-sample) scores.
`...`	Not currently used.

summary.binregCV_list *Summary method for binregCV_list objects*

Description

Summary method for binregCV_list objects

Usage

```
## S3 method for class 'binregCV_list'
summary(
  object,
  transform = NULL,
  conf.level = 0.95,
  type = c("cv", "both"),
  ...
)
```

Arguments

object	A binregCV_list object, as returned by brier_binreg when time has length greater than one.
transform	Optional back-transformation function applied to Brier scores and confidence limits, e.g. exp to undo the internal log scale.
conf.level	Confidence level for the reported intervals (default 0.95).
type	Either "cv" (default) for cross-validated Brier scores only, or "both" to also include apparent (in-sample) scores.
...	Not currently used.

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 occured.
concordance	gives concordance that is, probability of cause 2 (related to cif2) and cause 1 (related to cif1).
cif1	cumulative incidence for cause1.
cif2	cumulative incidence for cause1.

Author(s)

Thomas Scheike

References

Cross odds ratio Modelling of dependence for Multivariate Competing Risks Data, Scheike and Sun (2012), Biostatistics.

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

Examples

```
## library("timereg")
## data("multcif",package="mets") # simulated data
## multcif$cause[multcif$cause==0] <- 2
##
## times=seq(0.1,3,by=0.1) # to speed up computations use only these time-points
## add <- timereg::comp.risk(Event(time,cause)~+1+cluster(id),
##                           data=multcif,n.sim=0,times=times,cause=1)
###
## out1<- cor_cif(add,data=multcif,cause1=1,cause2=1,theta=log(2+1))
## summary(out1)
##
## pad <- predict(add,X=1,se=0,uniform=0)
## summary(out1,marg.cif=pad)
```

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

Description

Reporting OR from glm with binomial link and glm predictions

Usage

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

Arguments

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

Author(s)

Thomas Scheike

Examples

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

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

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

summaryTimeobject *Summarize a Time-Varying Estimate with Confidence Bands*

Description

Extracts estimates at specified time points with confidence intervals.

Usage

```
summaryTimeobject(
  mutimes,
  mu,
  se.mu = NULL,
  times = NULL,
  type = "log",
  level = 0.95,
  ...
)
```

Arguments

mutimes	vector of estimation time points.
mu	vector or matrix of estimates.
se.mu	vector or matrix of standard errors (optional).
times	time points at which to evaluate (default: mutimes).
type	confidence interval type: "log" or "plain".
level	confidence level (default 0.95).
...	additional arguments.

Value

A data.frame with columns: times, mean, se-mean, CI-2.5%, CI-97.5%.

---

surv_boxarea *Bivariate Survival Data on Rectangular Regions*

Description

Restricts bivariate survival data to a rectangular time region defined by left-truncation and right-censoring boundaries, for use with piecewise twostage models.

Usage

```
surv_boxarea(
  left.trunc,
  right.cens,
  data,
  timevar = "time",
  status = "status",
  id = "id",
  covars = NULL,
  covars.pairs = NULL,
  num = NULL,
```

```
  silent = 1,
  boxtimevar = "boxtime"
)
```

Arguments

left.trunc	vector of length 2 giving left truncation times.
right.cens	vector of length 2 giving right censoring times.
data	a data.frame with the survival data.
timevar	name of the time variable.
status	name of the status variable.
id	name of the cluster identifier.
covars	optional covariate names.
covars.pairs	optional pair-level covariate names.
num	within-cluster numbering variable name.
silent	verbosity level (1=silent).
boxtimevar	name for the created box-time variable.

Value

A data.frame in long format restricted to the specified region.

---

survival-helpers *Survival Twostage Helpers*

Description

Helper functions for the twostage survival dependence models.

Usage

```
survival.twostage(x, ...)

matplot.mets.twostage(object, ...)

alpha2spear(theta, link = 1)

alpha2kendall(theta, link = 0)

piecewise_twostage(
  cut1,
  cut2,
  data = parent.frame(),
  timevar = "time",
```

```
  status = "status",
  id = "id",
  covars = NULL,
  covars.pairs = NULL,
  num = NULL,
  method = "optimize",
  Nit = 100,
  detail = 0,
  silent = 1,
  weights = NULL,
  control = list(),
  theta = NULL,
  theta.des = NULL,
  var.link = 1,
  step = 0.5,
  model = "plackett",
  data.return = 0
)

piecewise_data(
  cut1,
  cut2,
  data = parent.frame(),
  timevar = "time",
  status = "status",
  id = "id",
  covars = NULL,
  covars.pairs = NULL,
  num = NULL,
  silent = 1
)
```

Arguments

x	a marginal model object (for survival.twostage).
...	additional arguments.
object	a twostage model object (for matplot method).
theta	initial dependence parameter values.
link	if 1, parameters are on log scale (for alpha2kendall/alpha2spear).
cut1	vector of cut points for the first time axis.
cut2	vector of cut points for the second time axis.
data	a data.frame with the survival data.
timevar	character name of the time variable.
status	character name of the status variable.
id	name of the cluster identifier column.
covars	optional character vector of covariate names.

covars.pairs	optional covariates at the pair level.
num	character name of the within-cluster number variable.
method	optimization method.
Nit	maximum number of iterations.
detail	level of detail in output.
silent	level of verbosity (1=silent).
weights	optional weights.
control	optimization control list.
theta.des	theta design matrix.
var.link	if 1, log-link for variance parameters.
step	step size for optimization.
model	dependence model: "plackett" or "clayton.oakes".
data.return	if 1, return data with model fits.

Details

survival.twostage is an alias for survival_twostage.

alpha2kendall converts the Clayton-Oakes alpha parameter to Kendall's tau.

alpha2spear converts the Clayton-Oakes alpha parameter to Spearman's rho.

piecewise_twostage fits twostage models on piecewise time intervals.

piecewise_data prepares data for piecewise twostage analysis.

matplot.mets.twostage produces matplot of twostage baseline estimates.

Author(s)

Klaus K. Holst, 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.

Usage

```
survival_twostage(
  margsurv,
  data = NULL,
  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 object.
data	Data frame (must be given).
method	Scoring method: "nr" for lava NR optimizer.
detail	Detail level for output.
clusters	Cluster variable.
silent	Debug information level.
weights	Weights for score equations.
theta	Starting values for variance components.
theta.des	Design for dependence parameters; when pairs are given, the indices of the theta-design for this pair are given in pairs as column 5.
var.link	Link function for variance: 1 for exponential link.
baseline.iid	To adjust for baseline estimation, using phreg function on same data.
model	Model type: "clayton.oakes" or "plackett".
marginal.trunc	Marginal left truncation probabilities.

`marginal.survival`
: Optional vector of marginal survival probabilities.

`strata`
: Strata for fitting (see examples).

`se.clusters`
: Clusters for SE calculation with IID.

`numDeriv`
: To get numDeriv version of second derivative; otherwise uses sum of squared scores for each pair.

`random.design`
: Random effect design for additive gamma model; when pairs are given, the indices of the pairs' random.design rows are given as columns 3:4.

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

`dim.theta`
: Dimension of the theta parameter for pairs situation.

`numDeriv.method`
: Method for numerical derivative (e.g., `"simple"` to speed up things).

`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 of random effects $\times$ number of theta parameters); when `NULL`, sums all parameters.

`var.par`
: Is 1 for the default parametrization with the variances of the random effects; `var.par=0` specifies that the λ_j's are used as parameters.

`no.opt`
: For not optimizing.

`...`
: Additional arguments to maximizer NR of lava.

Details

If clusters contain more than two subjects, a composite likelihood based on pairwise bivariate models is used (for full MLE see `twostageMLE`).

The two-stage model is constructed such that, given gamma distributed random effects, the survival functions are assumed independent, and the marginal survival functions are on Cox 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, specifying 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 `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 $n \times d$ (for d random effects).

For a cluster with two subjects, the `random.design` rows are v_1 and v_2. Such that the random effect for subject 1 is

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

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

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, specify how the parameters relate to λ_j with the argument `var.par=0`.

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

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

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

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

$$\lambda = theta.des \times \theta$$

here using `theta.des` to specify these low-dimension associations. 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 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.

Value

An object of class `"mets.twostage"` containing:

`theta`	Estimated dependence parameters.
`coef`	Coefficients.
`score`	Score vector.
`hess`	Hessian matrix.
`hessi`	Inverse Hessian matrix.
`var.theta`	Variance of theta parameters.
`robvar.theta`	Robust variance of theta parameters.
`loglike`	Log-likelihood value.
`theta.iid`	Influence functions for theta.
`model`	Model type used.

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*.
- Glidden (2000), A Two-Stage estimator of the dependence parameter for the Clayton Oakes model, LIDA.
- Measuring early or late dependence for bivariate twin data. Scheike, Holst, Hjelmborg (2015), LIDA.
- Estimating heritability for cause specific mortality based on twins studies. Scheike, Holst, Hjelmborg (2014), LIDA.
- Additive Gamma frailty models for competing risks data. Eriksson and Scheike (2015), *Biometrics*.

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)

 ## Reduce Ex.Timings
### 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 stratafied
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)


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

d <- subset(sim_ClaytonOakes(1000,2,0.5,0,stoptime=2,left=0),!truncated)
```

```
udp <- piecewise_twostage(c(0,0.5,2),data=d,method="optimize",
                          id="cluster",timevar="time",
                          status="status",model="clayton.oakes",silent=0)
summary(udp)


 ## Reduce Ex.Timings
### Same model using the strata option, a bit slower
########################################################
## makes the survival pieces for different areas in the plane
##ud1=surv_boxarea(c(0,0),c(0.5,0.5),data=d,id="cluster",timevar="time",status="status")
##ud2=surv_boxarea(c(0,0.5),c(0.5,2),data=d,id="cluster",timevar="time",status="status")
##ud3=surv_boxarea(c(0.5,0),c(2,0.5),data=d,id="cluster",timevar="time",status="status")
##ud4=surv_boxarea(c(0.5,0.5),c(2,2),data=d,id="cluster",timevar="time",status="status")

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

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

marg2 <- timereg::aalen(Surv(boxtime,status)~-1+factor(num):factor(intstrata),
               data=ud,n.sim=0,robust=0)
tdes <- model.matrix(~-1+factor(strata),data=ud)
fitp2 <- survival_twostage(marg2,data=ud,se.clusters=ud$cluster,clusters=ud$idstrata,
                model="clayton.oakes",theta.des=tdes,step=0.5)
summary(fitp2)

### now fitting the model with symmetry, i.e. strata 2 and 3 same effect
ud$stratas <- ud$strata;
ud$stratas[ud$strata=="0.5-2,0-0.5"] <- "0-0.5,0.5-2"
tdes2 <- model.matrix(~-1+factor(stratas),data=ud)
fitp3 <- survival_twostage(marg2,data=ud,clusters=ud$idstrata,se.cluster=ud$cluster,
                model="clayton.oakes",theta.des=tdes2,step=0.5)
summary(fitp3)

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


 ## Reduce Ex.Timings
### structured random effects model additive gamma ACE
### simulate structured two-stage additive gamma ACE model
data <- sim_ClaytonOakes_twin_ace(2000,2,1,0,3)
out <- twin_polygen_design(data,id="cluster")
pardes <- out$pardes
pardes
des.rv <- out$des.rv
```

```
head(des.rv)
aa <- phreg(Surv(time,status)~x+cluster(cluster),data=data,robust=0)
ts <- survival_twostage(aa,data=data,clusters=data$cluster,detail=0,
        theta=c(2,1),var.link=0,step=0.5,
        random.design=des.rv,theta.des=pardes)
summary(ts)
```

---

survivalG *G-Estimator for Cox and Fine-Gray Models*

Description

Computes the G-estimator (G-formula) for standardized survival or cumulative incidence estimates:

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

Usage

```
survivalG(
  x,
  data,
  time = NULL,
  Avalues = NULL,
  varname = NULL,
  same.data = TRUE,
  First = FALSE
)
```

Arguments

x	Object of class "phreg" or "cifreg".
data	Data frame for risk averaging. Must be part of the data used for fitting unless same.data=FALSE.
time	Time point for estimation.
Avalues	Values to compare for the first covariate A.
varname	Name of the variable to be treated as the treatment/exposure variable (default is the first variable).
same.data	Logical; assumes the same data is used for fitting and averaging.
First	Logical; if TRUE, uses only the first record for G-averaging (useful for start-stop structures).

Details

Based on a phreg or cifreg object. Provides influence functions for these risk estimates, allowing for standard error computation.

If the first covariate is a factor, contrasts between all levels are computed automatically. If it is continuous, specific values must be provided via Avalues.

Value

An object of class "survivalG" containing:

risk	Standardized risk estimates.
risk.iid	Influence functions for the risk estimates.
difference	Pairwise differences in risks.
ratio	Risk ratios.
survival.ratio	Survival ratios (for phreg).
survival.difference	Survival differences (for phreg).

Author(s)

Thomas Scheike

Examples

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

# Fine-Gray model
fg1 <- cifregFG(Event(time, cause) ~ tcell.f + platelet + age, bmt,
              cause = 1, cox.prep = TRUE)
summary(survivalG(fg1, bmt, 50))

 ## Reduce Ex.Timings
# Cox model
ss <- phreg(Surv(time, event) ~ tcell.f + platelet + age, bmt)
summary(survivalG(ss, bmt, 50))

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

# Time-varying G-estimates
sst <- survivalGtime(ss, bmt, n = 50)
plot(sst)

# Among treated (specify id to link influence functions)
```

```
ss <- phreg(Surv(time, event) ~ tcell.f + platelet + age + cluster(id), bmt)
summary(survivalG(ss, subset(bmt, tcell == 1), 50))
```

`test_casewise` *Test for Independence Using Casewise Concordance*

Description

Estimates the casewise concordance based on concordance and marginal estimates, and performs tests for the independence assumption. This is particularly useful in twin studies to assess genetic vs. environmental contributions to disease risk.

Usage

```
test_casewise(conc, marg, test = "no-test", p = 0.01)
```

Arguments

`conc`	An object containing concordance estimates (e.g., from `bicomprisk`). Must contain `time`, `P1`, `se.P1`, and ideally `P1.iid`.
`marg`	An object containing marginal cumulative incidence estimates (e.g., from `comp.risk` or `prodlim`). Must contain `time`, `P1`, `se.P1`, and ideally `P1.iid`.
`test`	Type of test: `"case"` (test on casewise concordance) or `"conc"` (test on concordance). Default is `"no-test"`.
`p`	Threshold for checking that marginal probability is greater than this value at some point (default 0.01). Used to avoid division by near-zero probabilities.

Details

The casewise concordance is defined as:

$$C(t) = \frac{P(T_1 \leq t, T_2 \leq t)}{P(T_1 \leq t)}$$

where the numerator is the joint probability of both twins having the event by time t, and the denominator is the marginal probability.

The function supports two types of tests:

- `"case"`: Tests on the casewise concordance scale (difference between observed and expected under independence).
- `"conc"`: Tests on the concordance scale (difference between observed concordance and squared marginal).

Standard errors are computed using cluster-based conservative estimates or influence functions (IID) if available from the input objects.

Value

An object of class "casewise" containing:

casewise	Matrix with time, casewise concordance, and standard errors.
marg	Matrix with time, marginal CIF, and standard errors.
conc	Matrix with time, concordance, and standard errors.
casewise.iid	Influence function decomposition for casewise concordance.
test	Test results (if test is specified): Pepe-Mori type test statistics, standard errors, z-values, and p-values.
mintime, maxtime	Time range used for the analysis.
same.cluster	Logical indicating if clusters were assumed identical.
testtype	Type of test performed.

Author(s)

Thomas Scheike

References

Scheike, T. H. (2024). Casewise concordance estimation and testing. mets package documentation.

See Also

bicomprisk, casewise, test_conc

Examples

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

prt <- prt[which(prt$id %in% sample(unique(prt$id),7500)),]
### marginal cumulative incidence of prostate cancer
times <- seq(60,100,by=2)
outm <- timereg::comp.risk(Event(time,status)~+1,data=prt,cause=2,times=times)

cifmz <- predict(outm,X=1,uniform=0,resample.iid=1)
cifdz <- predict(outm,X=1,uniform=0,resample.iid=1)

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

### To compute casewise cluster argument must be passed on,
###  here with a max of 100 to limit comp-time
```

```
outm <- timereg::comp.risk(Event(time,status)~+1,data=prt,
                 cause=2,times=times,max.clust=100)
cifmz <- predict(outm,X=1,uniform=0,resample.iid=1)
cc <- bicomprisk(Event(time,status)~strata(zyg)+id(id),data=prt,
               cause=c(2,2),se.clusters=outm$clusters)
cdz <- cc$model$"DZ"
cmz <- cc$model$"MZ"

cdz <- test_casewise(cdz,cifmz,test="case") ## test based on casewise
cmz <- test_casewise(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))
```

`test_conc` *Compare Two Concordance Estimates*

Description

Performs a Pepe-Mori type test to compare two concordance estimates (e.g., MZ vs DZ twins). The test evaluates whether the concordance functions differ significantly over time.

Usage

```
test_conc(conc1, conc2, same.cluster = FALSE)
```

Arguments

`conc1`	Concordance estimate of group 1.
`conc2`	Concordance estimate of group 2.
`same.cluster`	Logical; if FALSE, groups are assumed independent. If TRUE, estimates are based on the same data (e.g., paired twins).

Value

An object of class `"testconc"` containing:

`test`	Matrix with cumulative difference, standard error, z-value, and p-value.
`mintime`, `maxtime`	Time range used.
`same.cluster`	Logical flag.

Author(s)

Thomas Scheike

See Also

test_casewise

`test_logrankRecurrent` *Logrank-type test for comparing recurrent event marginal means between groups*

Description

Tests whether the marginal mean number of recurrent events differs across groups (strata), extending the classical logrank test to the setting of recurrent events with a competing terminal event. The test statistic is

$$z = \int_0^\tau w(s)\big[d\hat{\mu}_1(s) - d\hat{\mu}_2(s)\big],$$

where $w(s)$ is a weight function and $\hat{\mu}_j(s)$ is the estimated marginal mean for group j. Variance is estimated robustly via the influence functions of Ghosh and Lin (2000).

Usage

```
test_logrankRecurrent(
  recurrent,
  death,
  weight = c("I", "II", "III"),
  km = TRUE,
  start = 0,
  stop = NULL,
  at.risk = 5,
  cluster.id = NULL,
  ...
)
```

Arguments

`recurrent`	Either a `"recurrent"` object returned by recurrent_marginal, or a `"phreg"` object for the recurrent event model (in which case `death` must also be supplied).
`death`	A `"phreg"` object for the terminal event model. Required when `recurrent` is a `"phreg"` object; ignored otherwise.
`weight`	Character string specifying the weight scheme: `"I"`, `"II"`, or `"III"`. Default is `"I"`. For competing risks `"III"` gives Gray's test with robust standard errors.
`km`	Logical. If `TRUE` (default), the Kaplan-Meier estimator is used for the survival probability $S(t)$; otherwise the Nelson-Aalen estimator is used.
`start`	Left truncation time for the integration. Default is `0`.
`stop`	Right truncation time for the integration. Defaults to the last observed jump time.

`at.risk`	Minimum combined risk-set size below which the weight is set to zero. Default is 5.
`cluster.id`	Optional vector of cluster identifiers for aggregating influence functions across clusters before forming the test statistic.
`...`	Currently unused.

Details

Three weight schemes are available:

`"I"` (Default) $w(t) = R_1(t)R_2(t)/(R_1(t) + R_2(t))$, where $R_j(t) = Y_j(t)/\hat{S}_j(t-)$. Analogous to the standard logrank weight.

`"II"` $w(t) = Y_j(t)$, the raw risk-set size. Equivalent to using observed counts without survival adjustment.

`"III"` A modified weight incorporating the cumulative incidence, analogous to Gray's test for competing risks.

Value

An object of class `"estimate"` (from the **lava** package) with the following components:

`coef`	The weighted difference in marginal means between groups.
`se`	Robust standard error of the test statistic.
`lower`, `upper`	95% confidence interval bounds.
`p.value`	Two-sided p-value for the null hypothesis of no difference.

The object also carries an `iid` attribute containing the subject-level influence function decomposition of the test statistic, which can be used for further inference or combination with other estimators.

Author(s)

Thomas Scheike

References

Ghosh, D. and Lin, D. Y. (2000). Nonparametric analysis of recurrent events and death. *Biometrics*, 56, 554–562.

See Also

`recurrent_marginal`, `logrankRecurrentBase`

Examples

```
data(hfactioncpx12)
hf <- hfactioncpx12

## Test using two separate phreg models
xr <- phreg(Surv(entry, time, status == 1) ~ strata(treatment) + cluster(id), data = hf)
dr <- phreg(Surv(entry, time, status == 2) ~ strata(treatment) + cluster(id), data = hf)
out <- test_logrankRecurrent(xr, dr, stop = 5)
summary(out)

## Equivalently, using a recurrent_marginal object directly
outN <- recurrent_marginal(Event(entry, time, status) ~ strata(treatment) + cluster(id),
                           data = hf, cause = 1, death.code = 2)
test_logrankRecurrent(outN)
```

---

test_marginalMean *Pepe-Mori Test for Marginal Mean Comparison*

Description

Performs score-test type tests for proportionality of marginal means in competing risks data and recurrent events, as presented in Ghosh and Lin (2000). The test is based on an IPCW (Inverse Probability of Censoring Weighting) formulation.

Usage

```
test_marginalMean(
  formula,
  data,
  cause = 1,
  cens.code = 0,
  ...,
  death.code = 2,
  death.code.prop = NULL,
  time = NULL,
  beta = NULL
)
```

Arguments

formula	Formula with an Event object on the left-hand side and covariates (typically with strata() for group comparison) on the right. Can include cluster(id) for correlated data.
data	Data frame containing all variables referenced in the formula.
cause	Cause of interest (default 1).
cens.code	Censoring code (default 0).

`...`	Additional arguments passed to lower-level functions.
`death.code`	Code for death (terminating event, default 2).
`death.code.prop`	Code for other causes of death for Fine-Gray regression model.
`time`	Upper limit for Pepe-Mori and AUC integrals. If NULL, defaults to the maximum event time for the cause of interest.
`beta`	Starting values for the score test (default NULL, uses zeros).

Details

The function computes several tests:

1. **Pepe-Mori Test**: Tests for equality of marginal mean functions between groups.
2. **Ratio of AUC**: Compares the area under the curve of marginal means.
3. **Difference of AUC**: Tests for difference in areas under the curve.
4. **Score Test**: Tests for proportionality (equivalent to Gray's test for CIF).
5. **Proportionality Test**: Tests the proportional hazards assumption.

The Pepe-Mori test uses weights based on the number at risk in each group to construct a weighted integral of the difference in marginal means.

Value

An object of class `"marginalTest"` containing:

`pepe.mori`	Pepe-Mori test results with compare p-value.
`RatioAUC`	Ratio of AUC test results with compare p-value.
`difAUC`	Difference of AUC test results with compare p-value.
`prop.test`	Proportionality test results.
`score.test`	Score test results (equivalent to Gray's test).
`score.iid`	Influence function for the score test.
`time`	Upper time limit used.
`RAUC1`, `RAUCe`	Raw and transformed AUC estimates.

Author(s)

Thomas Scheike

References

Ghosh, D. and Lin, D. Y. (2000). Nonparametric Analysis of Recurrent Events and Death. Biometrics, 56, 554–562.

See Also

`test_logrankRecurrent`, `test_conc`

Examples

```
data(bmt,package="mets")
bmt$time <- bmt$time+runif(nrow(bmt))*0.01
bmt$id <- 1:nrow(bmt)
dcut(bmt) <- age.f~age

fg=cifregFG(Event(time,cause)~tcell,data=bmt,cause=1)

## computing tests for difference  for CIF
pmt <- test_marginalMean(Event(time,cause)~strata(tcell)+cluster(id),data=bmt,cause=1,
    death.code=1:2,death.code.prop=2,cens.code=0,time=40)
summary(pmt)

pmt$pepe.mori
pmt$RatioAUC
pmt$prop.test
## score test equialent to Gray's test but variance estimated differently
pmt$score.test

### age-groups
pmt <- test_marginalMean(Event(time,cause)~strata(age.f)+cluster(id),data=bmt,cause=1,
    death.code=1:2,death.code.prop=2,cens.code=0)
summary(pmt)

## having a look at the cumulative incidences
cifs <- cif(Event(time,cause)~strata(age.f)+cluster(id),data=bmt,cause=1)
plot(cifs)

## recurrent events
data(hfactioncpx12)
hf <- hfactioncpx12
pmt <- test_marginalMean(Event(entry,time,status)~strata(treatment)+cluster(id),data=hf,
    cause=1,death.code=2,cens.code=0)
summary(pmt)
```

`tetrachoric` *Estimate parameters from odds-ratio*

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))
```

---

tie_breaker *Break ties in event times for recurrent event data*

Description

Resolves tied event times in a counting-process dataset by adding a small random perturbation to duplicated exit times of event rows. This is a preprocessing step required by recurrent_marginal and related functions, which assume unique jump times within each stratum.

Usage

```
tie_breaker(
  data,
  stop = "time",
  start = "entry",
  status = "status",
  id = NULL,
  cause = NULL,
  cens.code = 0,
  exit.unique = TRUE,
  ddt = NULL,
  seed = NULL
)
```

Arguments

data	A data frame in counting-process format, sorted by subject and time.
stop	Name of the column containing interval exit (stop) times. Default is "time".
start	Name of the column containing interval entry (start) times, used to update the following row when id is supplied. Default is "entry".

status	Name of the column containing event status codes. Default is "status".
id	Name of the column containing subject identifiers. If supplied, the start time of the next interval for the same subject is adjusted to match the perturbed stop time, preserving interval continuity. Default is NULL (no adjustment made).
cause	Integer vector of status codes that identify events (non-censored rows). If NULL (default), all non-censoring status values are treated as events.
cens.code	Integer code(s) for censoring. Rows with this status are never perturbed. Default is 0.
exit.unique	Logical. If TRUE (default), an event time is considered tied whenever it coincides with *any* exit time in the data (including censored rows). If FALSE, only ties among event rows are resolved.
ddt	Maximum perturbation size. Tied event times are shifted by a uniform draw on $[0, \text{ddt}]$. If NULL (default), ddt is set to half the smallest positive gap between any two consecutive exit times in the data.
seed	Optional integer passed to set.seed() before drawing the perturbations, making results reproducible. Default is NULL (no seed set).

Details

A tie is defined as an event exit time (rows where status is a cause code) that coincides with another exit time in the dataset. When exit.unique = TRUE (default), a tie is flagged whenever an event time also appears among any other exit times (censored or event). When exit.unique = FALSE, only exact ties between two event rows are resolved.

Tied event times are perturbed by adding $U \cdot \delta$ where $U \sim \text{Uniform}(0, 1)$ and δ is ddt (defaulting to half the smallest observed positive gap between consecutive exit times). When subject IDs are provided via id, the corresponding interval start time of the immediately following row for the same subject is updated to maintain a valid counting-process structure, and a logical tiebreaker column is added to flag affected rows.

Value

The input data frame data with tied event exit times perturbed. If id is supplied, an additional logical column tiebreaker marks rows whose start time was adjusted as a consequence of a perturbation in the preceding row.

Author(s)

Thomas Scheike

See Also

recurrent_marginal, test_logrankRecurrent

Examples

```
data(hfactioncpx12)
hf <- hfactioncpx12
```

```
## Check for ties in event exit times
ev <- hf[hf$status == 1, ]
any(duplicated(ev$time))

## Resolve ties before fitting the marginal mean model
hf_clean <- tie_breaker(hf, stop = "time", start = "entry",
                        status = "status", id = "id",
                        cause = 1, cens.code = 0)

out <- recurrent_marginal(Event(entry, time, status) ~ cluster(id),
                          data = hf_clean, cause = 1, death.code = 2)
summary(out, times = 1:5)
```

TRACE *The TRACE study group of myocardial infarction*

Description

The TRACE data frame contains 1877 patients and is a subset of a data set consisting of approximately 6000 patients. It contains data relating survival of patients after myocardial infarction to various risk factors.

Format

This data frame contains the following columns:

id a numeric vector. Patient code.

status a numeric vector code. Survival status. 9: dead from myocardial infarction, 0: alive, 7: dead from other causes.

time a numeric vector. Survival time in years.

chf a numeric vector code. Clinical heart pump failure, 1: present, 0: absent.

diabetes a numeric vector code. Diabetes, 1: present, 0: absent.

vf a numeric vector code. Ventricular fibrillation, 1: present, 0: absent.

wmi a numeric vector. Measure of heart pumping effect based on ultrasound measurements where 2 is normal and 0 is worst.

sex a numeric vector code. 1: female, 0: male.

age a numeric vector code. Age of patient.

Details

sTRACE is a subsample consisting of 300 patients.

tTRACE is a subsample consisting of 1000 patients.

Source

The TRACE study group.

Jensen, G.V., Torp-Pedersen, C., Hildebrandt, P., Kober, L., F. E. Nielsen, Melchior, T., Joen, T. and P. K. Andersen (1997), Does in-hospital ventricular fibrillation affect prognosis after myocardial infarction?, European Heart Journal 18, 919–924.

Examples

```
data(TRACE)
names(TRACE)
```

---

ttpd *ttpd discrete survival data on interval form*

Description

ttpd discrete survival data on interval form

Source

Simulated data

---

twinbmi *BMI data set*

Description

BMI data set

Format

Self-reported BMI-values on 11,411 subjects

tvparnr: twin id bmi: BMI (m/kg^2) age: Age gender: (male/female) zyg: zygosity, MZ:=mz, DZ(same sex):=dz, DZ(opposite sex):=os

`twinlm` *Classic twin model for quantitative traits*

Description

Fits a classical twin model for quantitative traits.

Usage

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

Arguments

`formula`	Formula specifying effects of covariates on the response
`data`	`data.frame` with one observation 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 dyzogitic twins. If this argument is missing, the reference level (i.e. the first level) will be interpreted as the dyzogitic twins
`group`	Optional. Variable name defining group for interaction analysis (e.g., gender)
`group.equal`	If TRUE marginals of groups are asummed to be the same
`strata`	Strata variable name

weights	Weights matrix if needed by the chosen estimator. For use with Inverse Probability Weights
type	Character defining the type of analysis to be performed. Can be a subset of "aced" (additive genetic factors, common environmental factors, unique environmental factors, dominant genetic factors). Other choices are: • "0" (or "sat"): Saturated model where twin 1 and twin 2 within each twin pair may have a different marginal distribution. • "1" (or "flex","zyg"): Within twin pairs the marginal distribution is the same, but the marginal distribution may differ between MZ and DZ twins. A free correlation structure within MZ and DZ twins. • "2" (or "u", "eqmarg"): All individuals have the same marginals but a free correlation structure within MZ and DZ twins. The default value is an additive polygenic model `type="ace"`.
twinnum	The name of the column in the dataset numbering the twins (1,2). If it does not exist in `data` it will automatically be created.
binary	If `TRUE` a liability model is fitted. Note that if the right-hand-side of the formula is a factor, character vector, 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	Treshold used to define binary outcome y0
cens	Logical variable indicating whether to censor outcome
wide	Logical indicating if wide data format should be returned
...	Additional arguments parsed on to lower-level functions

Author(s)

Klaus K. Holst

See Also

twinlm

---

twinstut *Stutter data set*

Description

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

Format

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

twostageMLE *Twostage Survival Model Fitted by Pseudo MLE*

Description

Fits Clayton-Oakes clustered survival data using marginals that are on Cox form in the likelihood for the dependence parameter as in Glidden (2000).

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 could be a (pairs) $\times$ (number of parameters) $\times$ (max number random effects) matrix.
var.link	Link function for variance; if 1 then uses exponential 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.

Details

The dependence can be modelled via 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 exponential when `var.link=1`.

Value

An object of class `"mets.twostage"` containing:

`theta`	Estimated dependence parameters.
`coef`	Coefficients.
`var.theta`	Variance of theta parameters.
`robvar.theta`	Robust variance of theta parameters.
`theta.iid`	Influence functions for theta.
`theta.iid.naive`	Naive influence functions for theta.
`loglike`	Log-likelihood value.

Author(s)

Thomas Scheike

References

- Measuring early or late dependence for bivariate twin data. Scheike, Holst, Hjelmborg (2015), LIDA.
- Twostage modelling of additive gamma frailty models for survival data. Scheike and Holst, working paper.
- Shih and Louis (1995) Inference on the association parameter in copula models for bivariate survival data, *Biometrics*.
- Glidden (2000), A Two-Stage estimator of the dependence parameter for the Clayton Oakes model, LIDA.

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)
```

twostageREC *Fitting of Two-Stage Recurrent Events Random Effects Model*

Description

Fits a two-stage random effects model for recurrent events with a terminal event. Marginal models (Cox or Ghosh-Lin) are fitted first and passed to this function.

Usage

```
twostageREC(
  margsurv,
  recurrent,
  data = parent.frame(),
  theta = NULL,
  model = c("full", "shared", "non-shared"),
  ghosh.lin = NULL,
  theta.des = NULL,
  var.link = 0,
  method = "NR",
  no.opt = FALSE,
  weights = NULL,
  se.cluster = NULL,
  fnu = NULL,
  nufix = 0,
  nu = NULL,
  numderiv = 1,
  derivmethod = c("simple", "Richardson"),
  ...
)
```

Arguments

margsurv	Marginal model for the terminal event (object of class "phreg").
recurrent	Marginal model for recurrent events (object of class "phreg" or "recreg").
data	Data frame used for fitting.
theta	Starting value for total variance of gamma frailty.
model	Model type: "full" (fully shared), "shared" (partly shared), or "non-shared".
ghosh.lin	Logical; if TRUE, forces use of Ghosh-Lin marginals based on the recurrent model.
theta.des	Regression design for variance parameters.
var.link	Link function for variance (1 for exponential).
method	Optimization method (default "NR").
no.opt	Logical; if TRUE, skips optimization.

weights	Weights.
se.cluster	Clusters for SE calculation (GEE style).
fnu	Function to transform ν (amount shared).
nufix	Logical; if TRUE, fixes the amount shared.
nu	Starting value for the amount shared.
numderiv	Logical; if TRUE, uses numerical derivatives.
derivmethod	Method for numerical derivative.
...	Arguments for the optimizer.

Details

Supports:

- Cox/Cox marginals.
- Cox/Ghosh-Lin marginals.
- Fully shared, partly shared, or non-shared random effects.

Value

An object of class "twostageREC" containing:

coef	Estimated coefficients.
var	Variance-covariance matrix.
theta	Dependence parameters.
model	Model type.

Author(s)

Thomas Scheike

References

Scheike (2026), Two-stage recurrent events random effects models, LIDA, to appear.

WA_recurrent *While-Alive Estimands for Recurrent Events*

Description

Computes the "While-Alive" estimands for recurrent events in the presence of a terminal event (death). These estimands address the challenge of defining meaningful treatment effects when death prevents further observation of recurrent events.

Usage

```
WA_recurrent(
  formula,
  data,
  time = NULL,
  cens.code = 0,
  cause = 1,
  death.code = 2,
  trans = NULL,
  cens.formula = NULL,
  augmentR = NULL,
  augmentC = NULL,
  type = NULL,
  marks = NULL,
  ...
)
```

Arguments

formula	Formula with an Event object. The first covariate on the RHS must be a factor representing the treatment group. Can include cluster(id).
data	Data frame containing all variables referenced in the formula.
time	Time point t for estimation. If NULL, defaults to the maximum event time.
cens.code	Numeric code for censoring (default 0).
cause	Numeric code for the recurrent event of interest (default 1).
death.code	Numeric code for the terminal event/death (default 2).
trans	Power transformation for the mean of events per time-unit (default NULL, i.e., linear).
cens.formula	Formula for the censoring model. Default is ~strata(treatment).
augmentR	Formula for covariate augmentation in the randomization model (e.g., ~age+sex). Improves efficiency.
augmentC	Formula for covariate augmentation in the censoring model. Enables double robustness.
type	Type of augmentation for the binomial regression call. Default is "I" if augmentC is given, otherwise "II".
marks	Optional marks for composite outcome situations (e.g., distinguishing event types in a composite endpoint).
...	Additional arguments passed to binregATE.

Details

The function estimates two primary quantities:

1. **Ratio of Means**:

$$E(N(\min(D,t)))/E(\min(D,t))$$

The expected number of events up to time t (censored by death D) divided by the expected time alive up to t.

2. **Mean of Events per Time Unit**:

$$E(N(\min(D,t))/\min(D,t))$$

The expected rate of events per unit of time alive.

Estimation is based on Inverse Probability of Censoring Weighting (IPCW) to handle administrative censoring and death. The method can be augmented with covariates (double robust estimation) to improve efficiency and robustness.

Value

An object of class `"WA"` containing:

`RAW`	List of raw estimates: RMST, mean number of events, ratio of means, and their log-transformed versions with standard errors.
`ET`	List of estimated treatment effects: risk difference for the mean rate (`riskDR`) and optionally the augmented version (`riskDRC`).
`time`	The time point used for estimation.
`cause`, `death.code`, `cens.code`	Codes used.
`augmentR`, `augmentC`	Formulas used for augmentation.

The object includes influence functions (IID) for all estimators, allowing for further variance calculations or combination with other estimators.

Author(s)

Thomas Scheike

References

Ragni, A., Martinussen, T., & Scheike, T. H. (2023). Nonparametric estimation of the Patient Weighted While-Alive Estimand. arXiv preprint.

Mao, L. (2023). Nonparametric inference of general while-alive estimands for recurrent events. Biometrics, 79(3), 1749–1760.

Schmidli, H., Roger, J. H., & Akacha, M. (2023). Estimands for recurrent event endpoints in the presence of a terminal event. Statistics in Biopharmaceutical Research, 15(2), 238–248.

Examples

```
data(hfactioncpx12)
dtable(hfactioncpx12,~status)
dd <- WA_recurrent(Event(entry,time,status)~treatment+cluster(id),data=hfactioncpx12,
                   time=2,death.code=2)
summary(dd)
```

WA_reg *While-Alive Regression for Recurrent Events*

Description

Performs regression analysis for the "While-Alive" mean of events per time unit, defined as $Z(t) = N(\min(D,t))/\min(D,t)$. This function models how covariates affect the rate of recurrent events per unit of time alive.

Usage

```
WA_reg(
  formula,
  data,
  time = NULL,
  cens.code = 0,
  cause = 1,
  death.code = 2,
  marks = NULL,
  ...,
  trans = 1
)
```

Arguments

formula	Formula with regression design. The first covariate on the RHS must be the treatment factor. Can include other covariates and `cluster(id)`.
data	Data frame.
time	Time point t for estimation.
cens.code	Censoring code.
cause	Event cause code.
death.code	Death code.
marks	Marks for composite outcomes.
...	Additional arguments passed to `binreg`.
trans	Power transformation for the outcome (default 1).

Details

The estimation is based on IPCW (Inverse Probability of Censoring Weighting) and calls `binreg` after constructing the outcome variable. It supports double robust estimation if covariate augmentation is specified.

Value

An object of class `"binreg"` containing coefficient estimates, standard errors, confidence intervals, and influence functions for the regression of the event rate per time alive.

Author(s)

Thomas Scheike

References

Ragni, A., Martinussen, T., & Scheike, T. H. (2023). Nonparametric estimation of the Patient Weighted While-Alive Estimand. arXiv preprint.

Examples

```
data(hfactioncpx12)
hfactioncpx12$age <- rnorm(741)[hfactioncpx12$id]
dtable(hfactioncpx12,~status)
## exp-link regression
dd <- WA_reg(Event(entry,time,status)~treatment+age+cluster(id),data=hfactioncpx12,
                    time=2,death.code=2)
summary(dd)
```

Index

∗ **binomial**
binomial_twostage, 10
∗ **data**
ACTG175, 6
calgb8923, 45
CPH_HPN_CRBSI, 64
dermalridges, 73
dermalridgesMZ, 73
haplo, 110
hfactioncpx12, 114
mena, 133
migr, 134
multcif, 137
np, 137
prt, 157
ttpd, 229
twinbmi, 229
twinstut, 233
∗ **models**
blocksample, 37
mets.options, 134
twinlm, 230
twinsim, 232
∗ **package**
bmt, 39
diabetes, 74
melanoma, 132
TRACE, 228
∗ **regression**
binomial_twostage, 10
twinlm, 230
twinsim, 232
∗ **simulation**
rchaz, 161
sim_cif, 182
sim_multistate, 187
sim_multistateII, 189
sim_phreg, 190
sim_phregs, 192
∗ **survival**
cor_cif, 58
event_split, 88
event_split2, 89
Grandom_cif, 106
LinSpline, 129
random_cif, 157
rchaz, 161
rcrisk, 163
sim_cif, 182
sim_multistate, 187
sim_multistateII, 189
sim_phreg, 190
sim_phregs, 192
summary.cor, 204
survival_twostage, 210
twostageMLE, 234
∗ **utilities**
blocksample, 37
[.Event *(*Event*)*, 87

aalenMets, 5
ace_family_design
*(*p11_binomial_twostage_RV*)*, 137
ACTG175, 6
alpha2kendall *(*survival-helpers*)*, 208
alpha2spear *(*survival-helpers*)*, 208
as.character.Event *(*Event*)*, 87
as.matrix.Event *(*Event*)*, 87
ascertained_pairs
*(*p11_binomial_twostage_RV*)*, 137

basecumhaz *(*plot.phreg*)*, 148
baseplot *(*plot.phreg*)*, 148
bicomprisk, 7, *46*, *219*
bicompriskData *(*bicomprisk*)*, 7
binomial_twostage, 10
binomial_twostage_time
*(*binomial_twostage*)*, 10
binreg, 14, *22*, *25*, *28*, *31*, *35*, *175*

binreg_IPTW, 18
binregATE, 20, *25*, *173*
binregCasewise, 23
binregG, *22*, 24
binregRatio, 26
binregStrata, 29, *41*, *43, 44*
binregt *(*binreg*)*, 14
binregTSR, 32
biprobit, 35
blocksample, 37
bmt, 39
BootmediatorSurv *(*mediatorSurv*)*, 130
bplot *(*plot.phreg*)*, 148
bptwin, 39, *231*
brier_binreg, 41, *203, 204*

calgb8923, 45
casewise, *9*, 46, *219*
casewise_bin, 47
cif, 48, *155, 156*
cif-nonpar *(*rr_cif*)*, 177
cif_yearslost, 49
cifreg, 50, *53*, *118*
cifregFG, *52*, 53
ClaytonOakes, 53
cluster.index *(*cluster_index*)*, 55
cluster_index, 55
coarse_clust, 56
concordance.cor *(*concordanceCor*)*, 57
concordanceCor, 57
concordanceTwinACE
*(*p11_binomial_twostage_RV*)*, 137
concordanceTwostage
*(*p11_binomial_twostage_RV*)*, 137
conftype *(*robust.basehaz.phreg*)*, 176
cor_cif, 58
count_history, 62
countID *(*cluster_index*)*, 55
CPH_HPN_CRBSI, 64
cpred *(*fast.approx*)*, 94
cumContr, *103*, *105*
cumContr *(*gofZ_phreg*)*, 104
cumoddsreg, 64, *120*
cumsumstrata *(*strata-numeric*)*, 202

daggr *(*daggregate*)*, 65
daggregate, 65
Dbvn, 67
dby, 67
dby2 *(*dby*)*, 67
dby2<- *(*dby*)*, 67
dby<- *(*dby*)*, 67
dbyr *(*dby*)*, 67
dcor, 69
dcount *(*dcor*)*, 70
dcut, 70
dcut<- *(*dcut*)*, 70
ddrop *(*dcut*)*, 70
ddrop<- *(*dcut*)*, 70
dermalridges, 73
dermalridgesMZ, 73
deval *(*dcor*)*, 70
deval2 *(*dcor*)*, 70
dfactor *(*drelevel*)*, 80
dfactor<- *(*drelevel*)*, 80
dhead *(*dprint*)*, 76
diabetes, 74
diffstrata *(*cifreg*)*, 50
dInterval
*(*interval_logitsurv_discrete*)*,
118
divide_conquer, 74
dkeep *(*dcut*)*, 70
dkeep<- *(*dcut*)*, 70
dlag, 75
dlag<- *(*dlag*)*, 75
dlev *(*drelevel*)*, 80
dlev<- *(*drelevel*)*, 80
dlevel *(*drelevel*)*, 80
dlevel<- *(*drelevel*)*, 80
dlevels *(*drelevel*)*, 80
dlist *(*dprint*)*, 76
dmean *(*dcor*)*, 70
dmeansd *(*dcor*)*, 70
dmvn *(*pmvn*)*, 151
dnames *(*dcut*)*, 70
dnames<- *(*dcut*)*, 70
dnumeric *(*drelevel*)*, 80
dnumeric<- *(*drelevel*)*, 80
dprint, 76
dquantile *(*dcor*)*, 70
dreg, 77
drelev *(*drelevel*)*, 80
drelev<- *(*drelevel*)*, 80
drelevel, 80
drelevel<- *(*drelevel*)*, 80
drename *(*dcut*)*, 70

drename<- (dcut), 70
dreshape (fast.reshape), 96
drm (dcut), 70
drm<- (dcut), 70
drop.specials, 82
dsample (blocksample), 37
dscalar (dcor), 70
dsd (dcor), 70
dsort, 82
dsort2 (dsort), 82
dsort<- (dsort), 82
dspline, 83
dspline<- (dspline), 83
dstr (dcor), 70
dsubset (dcor), 70
dsum (dcor), 70
dsummary (dcor), 70
dtab (dtable), 84
dtable, 84
dtail (dprint), 76
dtrans (dtransform), 86
dtrans<- (dtransform), 86
dtransform, 86
dtransform<- (dtransform), 86
dunique (dcut), 70

estimate, *44*
estimate.binregCV, *43*, *44*, 86
Event, 87, *155*, *169*
event_split, 88
event_split2, 89
eventpois, 90
extendCums, 91

familycluster_index, 92
familyclusterWithProbands_index, 93
fast.approx, 94
fast.cluster, 95
fast.pattern, 95
fast.reshape, 96
faster.reshape, 98
FGprediid (cifreg), 50
folds, 99
force.same.cens, 99
force_same_cens (force.same.cens), 99
format.Event (Event), 87

GLprediid (recreg), 164
gof.phreg, 100, *103*
gofFG, *52*
gofFG (cifreg), 50
gofM_phreg, 102, *102*, *105*
gofZ_phreg, *102*, *103*, 104
Grandom.cif (rr_cif), 177
Grandom_cif, 106
grouptable, 109

haplo, 110
haplo_surv_discrete, 111
hfactioncpx12, 114

IC.binreg, 114
IC.phreg, 116
iidBaseline, *116*, 117
IIDrecreg (recreg), 164
iidRecurrent (recurrent_marginal), 169
ilap, 118
indexstrata (fast.approx), 94
indexstratarightR (cifreg), 50
Interval (interval_logitsurv_discrete), 118
interval_logitsurv_discrete, *65*, 118
invsubdist (sim_cif), 182
ipw, 121
ipw2, 122

jumptimes, 124

kendall.ClaytonOakes.twin.ace (p11_binomial_twostage_RV), 137
kendall_ClaytonOakes_twin_ace (p11_binomial_twostage_RV), 137
kendall_normal_twin_ace (p11_binomial_twostage_RV), 137
km, 125
kmplot (plot.phreg), 148

lifecourse, 126
lifetable (lifetable.matrix), 128
lifetable.matrix, 128
lin_approx (rchaz), 161
LinSpline, 129
logitATE, *22*
logitATE (binregATE), 20
logitIPCW (binreg), 14
logitIPCWATE, *22*
logitIPCWATE (binregATE), 20
logitSurv, 129

loglikMVN *(pmvn)*, 151
logrankRecurrentBase, *222*
logrankRecurrentBase *(test_logrankRecurrent)*, 221

make_pairwise_design *(p11_binomial_twostage_RV)*, 137
marks *(recreg)*, 164
matdoubleindex *(strata-numeric)*, 202
matplot.mets.twostage *(survival-helpers)*, 208
mdi *(strata-numeric)*, 202
mediatorSurv, 130
medweight, 132
melanoma, 132
mena, 133
mets.options, 134
migr, 134
mlogit, 135, *152*
multcif, 137
mystrata *(cluster_index)*, 55
mystrata2index *(cluster_index)*, 55

nonparcuminc *(rr_cif)*, 177
normalATE *(binregATE)*, 20
np, 137
npc *(rr_cif)*, 177

or.cif *(cor_cif)*, 58
or2prob *(tetrachoric)*, 225
or_cif *(rr_cif)*, 177

p11.binomial.twostage.RV *(p11_binomial_twostage_RV)*, 137
p11_binomial_twostage_RV, 137
pairRisk *(cluster_index)*, 55
pbvn *(pmvn)*, 151
pcif *(eventpois)*, 90
phreg, *30, 31*, *116*, *118*, 141, *169*
phreg.par *(phreg_weibull)*, 146
phreg_IPTW, 142
phreg_rct, *35*, 144
phreg_weibull, 146
piecewise_data *(survival-helpers)*, 208
piecewise_twostage *(survival-helpers)*, 208
plack.cif2 *(plack_cif)*, 148
plack_cif, 148
plot.phreg, *142*, 148
plot_twin, 149
plotConfRegion *(plot.phreg)*, 148
plotConfregion *(plot.phreg)*, 148
plotConfRegionSE *(plot.phreg)*, 148
plotcr *(rr_cif)*, 177
plotstrata *(plot.phreg)*, 148
plotSurvd *(interval_logitsurv_discrete)*, 118
pmvn, 151
predict *(mlogit)*, 135
predict.mlogit, 151
predict.phreg, *142*, 152
predictCumhaz *(fast.approx)*, 94
predictGLM *(summaryGLM)*, 206
predictlogitSurvd, *120*
predictlogitSurvd *(interval_logitsurv_discrete)*, 118
predictPairPlack *(rr_cif)*, 177
predictSurvd *(interval_logitsurv_discrete)*, 118
print.binregCV *(summary.binregCV)*, 203
print.casewise, 154
print.Event *(Event)*, 87
print.summary_binregCV_multi, *44*, 154
prob_exceed_recurrent, 155, *170*
prt, 157

random.cif *(rr_cif)*, 177
random_cif, 157
randomDes *(survival_twostage)*, 210
ratioATE, 160, *173*
rbind.Event *(Event)*, 87
rchaz, *92*, 161, *162*
rchazl, 162
rcrisk, *92*, 163
readmargsurv *(survival_twostage)*, 210
recreg, *52*, *118*, 164, *168*
recregIPCW, *166*, 167
recurrent_marginal, *156*, 169, *197*, *221*, *222*, *226*, *227*
recurrent_marginalAIPCW *(recurrent_marginal)*, 169
recurrentMarginal *(recurrent_marginal)*, 169
recurrentMarginalPhreg *(recurrent_marginal)*, 169

resmean_phreg, 171
resmeanATE, *160*, 172, *175*
resmeanIPCW, *28*, *44*, 174
revcumsum *(*strata-numeric*)*, 202
revcumsumstrata *(*strata-numeric*)*, 202
rmst_phreg *(*resmean_phreg*)*, 171
rmstATE *(*resmeanATE*)*, 172
rmstIPCW, *175*
rmstIPCW *(*resmeanIPCW*)*, 174
rmtlRatio *(*binregRatio*)*, 26
rmvn *(*pmvn*)*, 151
robust.basehaz.phreg, 176
robust_phreg, *142*
robust_phreg *(*phreg*)*, 141
rr.cif *(*cor_cif*)*, 58
rr_cif, 177
rweibullcox, 181

scalecumhaz *(*recreg*)*, 164
scoreMVN *(*pmvn*)*, 151
sim_cif, 182
sim_cifs *(*sim_cif*)*, 182
sim_ClaytonOakes, 184
sim_ClaytonOakesLam *(*sim_ClaytonOakes*)*, 184
sim_ClaytonOakesWei, 185
sim_GLcox, 186, *196, 197*
sim_multistate, 187
sim_multistateII, 189
sim_phreg, 190
sim_phregs, 192
sim_rchaz *(*rchaz*)*, 161
sim_recurrent, 193, *199*
sim_recurrent_list, *193, 194*, *196–199*
sim_recurrent_list *(*sim_recurrentII*)*, 197
sim_recurrent_ts, 195, *202*
sim_recurrentII, *193*, *196*, 197, *202*
sim_recurrentTS, *199*, 200
sim_subdist *(*sim_cif*)*, 182
simlogitSurvd, *120*
sTRACE *(*TRACE*)*, 228
strata-numeric, 202
subdist *(*sim_cif*)*, 182
summary.binregCV, *44*, 203
summary.binregCV_list, 204
summary.cor, 204
summary.Event *(*Event*)*, 87
summarybase.phreg *(*robust.basehaz.phreg*)*, 176
summaryGLM, 206
summaryTimeobject, 206
sumstrata *(*strata-numeric*)*, 202
Surv, *30*
surv_boxarea, 207
survival-helpers, 208
survival.twostage *(*survival-helpers*)*, 208
survival_twostage, 210
survivalG, 216
survivalGtime *(*survivalG*)*, 216

test_casewise, *9*, *46*, 218, *221*
test_conc, *219*, 220, *224*
test_logrankRecurrent, *170*, 221, *224*, *227*
test_marginalMean, 223
tetrachoric, 225
tie_breaker, *169, 170*, 226
TRACE, 228
ttpd, 229
tTRACE *(*TRACE*)*, 228
twin-design *(*p11_binomial_twostage_RV*)*, 137
twin.polygen.design *(*p11_binomial_twostage_RV*)*, 137
twin_polygen_design *(*p11_binomial_twostage_RV*)*, 137
twinbmi, 229
twinlm, *41*, 230, *233*
twinlm.strata, *41*, *231*
twinlm.time, *41*, *231*
twinlm.time *(*bptwin*)*, 39
twinsim, *41*, *231*, 232
twinstut, 233
twostage_aalen *(*survival_twostage*)*, 210
twostage_cox.aalen *(*survival_twostage*)*, 210
twostage_coxph *(*survival_twostage*)*, 210
twostage_phreg *(*survival_twostage*)*, 210
twostageMLE, 234
twostageREC, 236

vecAllStrata *(*cifreg*)*, 50

WA_recurrent, 237
WA_reg, 240