Package 'depCensoring'

October 17, 2024

Type Package

Title Statistical Methods for Survival Data with Dependent Censoring

Version 0.1.3

Maintainer Negera Wakgari Deresa <negera.deresa@gmail.com>

Description Several statistical methods for analyzing survival data under various forms of dependent censoring are implemented in the package. In addition to accounting for dependent censoring, it offers tools to adjust for unmeasured confounding factors. The implemented approaches allow users to estimate the dependency between survival time and dependent censoring time, based solely on observed survival data. For more details on the methods, refer to Deresa and Van Keilegom (2021) <doi:10.1093/biomet/asaa095>, Czado and Van Keilegom (2023) <doi:10.1093/biomet/asac067>, Crommen et al. (2024) <doi:10.1007/s11749-023-00903-9> and Willems et al. (2024+) <doi:10.48550/arXiv.2403.11860>.

Imports survival, foreach, parallel, doParallel, pbivnorm, stats, MASS, nleqslv, OpenMx, mvtnorm, rafalib, rvinecopulib, matrixcalc, nloptr, stringr, numDeriv, SemiPar.depCens

License GPL-3

Encoding UTF-8

RoxygenNote 7.3.2

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

NeedsCompilation no

Author Ilias Willems [aut] (<https://orcid.org/0009-0001-9463-9942>),
Gilles Crommen [aut] (<https://orcid.org/0000-0001-8380-1900>),
Negera Wakgari Deresa [aut, cre]
(<https://orcid.org/0000-0002-1302-3725>),
Ingrid Van Keilegom [aut] (<https://orcid.org/0000-0001-8827-7642>),
Claudia Czado [aut] (<https://orcid.org/0000-0002-6329-5438>)

Repository CRAN

Date/Publication 2024-10-17 13:00:05 UTC

Contents

boot.nonparTrans	*Nonparametric bootstrap approach for a Semiparametric transformation model under dependent censpring*

Description

This function estimates the bootstrap standard errors for the finite-dimensional model parameters and for the non-parametric transformation function. Parallel computing using foreach has been used to speed up the estimation of standard errors.

Usage

```
boot.nonparTrans(init, resData, X, W, n.boot, n.iter, eps)
```

Arguments

init	Initial values for the finite dimensional parameters obtained from the fit of NonParTrans
resData	Data matrix with three columns; Z = the observed survival time, d1 = the censoring indicator of T and d2 = the censoring indicator of C.
X	Data matrix with covariates related to T
W	Data matrix with covariates related to C.
n.boot	Number of bootstraps to use in the estimation of bootstrap standard errors.
n.iter	Number of iterations; the default is n.iter = 15. The larger the number of iterations, the longer the computational time.
eps	Convergence error. This is set by the user in such away that the desired convergence is met; the default is eps = 1e-3

Value

Bootstrap standard errors for parameter estimates and for estimated cumulative hazard function.

Bvprob	*Compute bivariate survival probability*

Description

This function calculates a bivariate survival probability based on multivariate normal distribution.

Usage

```
Bvprob(lx, ly, rho)
```

Arguments

lx	The first lower bound of integration
ly	The second lower bound
rho	Association parameter

---

chol2par	*Transform Cholesky decomposition to covariance matrix*

Description

This function transforms the parameters of the Cholesky de- composition to the covariance matrix, represented as a the row-wise con- catenation of the upper-triangular elements.

Usage

```
chol2par(par.chol1)
```

Arguments

par.chol1	The vector of Cholesky parameters.

Value

Covariance matrix corresponding to the provided Cholesky decomposition.

---

chol2par.elem	*Transform Cholesky decomposition to covariance matrix parameter element.*

Description

This function transforms the parameters of the Cholesky de- composition to a covariance matrix element. This function is used in chol2par.R.

Usage

```
chol2par.elem(a, b, par.chol1)
```

Arguments

a	The row index of the covariance matrix element to be computed.
b	The column index of the covariance matrix element to be computed.
par.chol1	The vector of Cholesky parameters.

Value

Specified element of the covariance matrix resulting from the provided Cholesky decomposition.

---

cr.lik *Competing risk likelihood function.*

Description

This function implements the second step likelihood function of the competing risk model defined in Willems et al. (2024+).

Usage

```
cr.lik(
  n,
  s,
  Y,
  admin,
  cens.inds,
  M,
  Sigma,
  beta.mat,
  sigma.vct,
  rho.mat,
  theta.vct
)
```

Arguments

n	The sample size.
s	The number of competing risks.
Y	The observed times.
admin	Boolean value indicating whether or not administrative censoring should be taken into account.
cens.inds	matrix of censoring indicators (each row corresponds to a single observation).
M	Design matrix, consisting of [intercept, exo.cov, Z, cf]. Note that `cf` represents the multiple ways of 'handling' the endogenous covariate Z, see also the documentation of 'estimate.cmprsk.R'. When there is no confounding, `M` will be [intercept, exo.cov].
Sigma	The covariance matrix.
beta.mat	Matrix containing all of the covariate effects.
sigma.vct	Vector of standard deviations. Should be equal to `sqrt(diag(Sigma))`.
rho.mat	The correlation matrix.
theta.vct	Vector containing the parameters of the Yeo-Johnsontrans- formations.

Value

Evaluation of the log-likelihood function

References

Willems et al. (2024+). Flexible control function approach under competing risks (in preparation).

---

dat.sim.reg.comp.risks

Data generation function for competing risks data

Description

This function generates competing risk data that can be used in simulation studies.

Usage

```
dat.sim.reg.comp.risks(
  n,
  par,
  iseed,
  s,
  conf,
  Zbin,
  Wbin,
  type.cov,
  A.upper = 15
)
```

Arguments

n	The sample size of the generated data set.
par	List of parameter vectors for each component of the transformation model.
iseed	Random seed.
s	The number of competing risks. Note that the given parameter vector could specify the parameters for the model with more than s competing risks, but in this case only the first s sets of parameters will be considered.
conf	Boolean value indicating whether the data set should contain confounding.
Zbin	Indicator whether the confounded variable is binary Zbin = 1 or not Zbin = 0. If conf = FALSE, this variable is ignored.
Wbin	Indicator whether the instrument is binary (Zbin = 1) or not Zbin = 0.
type.cov	Vector of characters "c" and "b", indicating which exogenous covariates should be continuous ″c″ or binary ″b″.
A.upper	The upper bound on the support of the administrative censoring distribution. This can be used to control for the amount of administrative censoring in the data. Default is A.upper = 15. A.upper = NULL will return a data set without administrative censoring.

Value

A generated data set

---

dchol2par	*Derivative of transform Cholesky decomposition to covariance matrix.*

Description

This function defines the derivative of the transformation function that maps Cholesky parameters to the full covariance matrix.

Usage

```
dchol2par(par.chol1)
```

Arguments

par.chol1   The vector of Cholesky parameters.

Value

Derivative of the function that transforms the cholesky parameters to the full covariance matrix.

---

dchol2par.elem	*Derivative of transform Cholesky decomposition to covariance matrix element.*

Description

This function defines the derivative of the transformation function that maps Cholesky parameters to a covariance matrix element. This function is used in dchol2par.R.

Usage

```
dchol2par.elem(k, q, a, b, par.chol1)
```

Arguments

k	The row index of the parameter with respect to which to take the derivative.
q	the column index of the parameter with respect to which to take the derivative.
a	The row index of the covariance matrix element to be computed.
b	The column index of the covariance matrix element to be computed.
par.chol1	The vector of Cholesky parameters.

Value

Derivative of the function that transforms the cholesky parameters to the specified element of the covariance matrix, evaluated at the specified arguments.

Distance *Distance between vectors*

Description

This function computes distance between two vectors based on L2-norm

Usage

```
Distance(b, a)
```

Arguments

b	Second vector
a	First vector

DYJtrans *Derivative of the Yeo-Johnson transformation function*

Description

Evaluates the derivative of the Yeo-Johnson transformation at the provided argument.

Usage

```
DYJtrans(y, theta)
```

Arguments

y	The argument to be supplied to the derivative of the Yeo-Johnson transformation.
theta	The parameter of the Yeo-Johnson transformation. This should be a number in the range [0,2].

Value

The transformed value of y.

---

estimate.cf *Estimate the control function*

Description

This function estimates the control function for the endogenous variable based on the provided covariates. This function is called inside estimate.cmprsk.R.

Usage

```
estimate.cf(XandW, Z, Zbin, gammaest = NULL)
```

Arguments

XandW	Design matrix of exogenous covariates.
Z	Endogenous covariate.
Zbin	Boolean value indicating whether endogenous covariate is binary.
gammaest	Vector of pre-estimated parameter vector. If NULL, this function will first estimate gammaest. Default value is gammaest = NULL.

Value

List containing the vector of values for the control function and the regression parameters of the first step.

---

estimate.cmprsk *Estimate the competing risks model of Rutten, Willems et al. (20XX).*

Description

This function estimates the parameters in the competing risks model described in Willems et al. (2024+). Note that this model extends the model of Crommen, Beyhum and Van Keilegom (2024) and as such, this function also implements their methodology.

Usage

```
estimate.cmprsk(
  data,
  admin,
  conf,
  eoi.indicator.names = NULL,
  Zbin = NULL,
  inst = "cf",
  realV = NULL,
  eps = 0.001
)
```

Arguments

`data`	A data frame, adhering to the following formatting rules:

- The first column, named `"y"`, contains the observed times.
- The next columns, named `"delta1"`, `delta2`, etc. contain the indicators for each of the competing risks.
- The next column, named `da`, contains the censoring indicator (independent censoring).
- The next column should be a column of all ones (representing the intercept), names `x0`.
- The subsequent columns should contain the values of the covariates, named `x1`, `x2`, etc.
- When applicable, the next column should contain the values of the endogenous variable. This column should be named `z`.
- When `z` is provided and an instrument for `z` is available, the next column, named `w`, should contain the values for the instrument.

`admin`	Boolean value indicating whether the data contains administrative censoring.
`conf`	Boolean value indicating whether the data contains confounding and hence indicating the presence of `z` and, possibly, `w`.
`eoi.indicator.names`	
	Vector of names of the censoring indicator columns pertaining to events of interest. Events of interest will be modeled allowing dependence between them, whereas all censoring events (corresponding to indicator columns not listed in `eoi.indicator.names`) will be treated as independent of every other event. If `eoi.indicator.names == NULL`, all events will be modeled dependently.
`Zbin`	Indicator value indicating whether (`Zbin = TRUE`) or not `Zbin = FALSE` the endogenous covariate is binary. Default is `Zbin = NULL`, corresponding to the case when `conf == FALSE`.
`inst`	Variable encoding which approach should be used for dealing with the confounding. `inst = "cf"` indicates that the control function approach should be used. `inst = "W"` indicates that the instrumental variable should be used 'as is'. `inst = "None"` indicates that Z will be treated as an exogenous covariate. Finally, when `inst = "oracle"`, this function will access the argument `realV` and use it as the values for the control function. Default is `inst = "cf"`.
`realV`	Vector of numerics with length equal to the number of rows in `data`. Used to provide the true values of the instrumental function to the estimation procedure.
`eps`	Value that will be added to the diagonal of the covariance matrix during estimation in order to ensure strictly positive variances.

Value

A list of parameter estimates in the second stage of the estimation algorithm (hence omitting the estimates for the control function), as well as an estimate of their variance and confidence intervals.

References

Willems et al. (2024+). Flexible control function approach under competing risks (in preparation).

Crommen, G., Beyhum, J., and Van Keilegom, I. (2024). An instrumental variable approach under dependent censoring. Test, 33(2), 473-495.

Examples

```
n <- 200

# Set parameters
gamma <- c(1, 2, 1.5, -1)
theta <- c(0.5, 1.5)
eta1 <- c(1, -1, 2, -1.5, 0.5)
eta2 <- c(0.5, 1, 1, 3, 0)

# Generate exogenous covariates
x0 <- rep(1, n)
x1 <- rnorm(n)
x2 <- rbinom(n, 1, 0.5)

# Generate confounder and instrument
w <- rnorm(n)
V <- rnorm(n, 0, 2)
z <- cbind(x0, x1, x2, w) %*% gamma + V
realV <- z - (cbind(x0, x1, x2, w) %*% gamma)

# Generate event times
err <- MASS::mvrnorm(n, mu = c(0, 0), Sigma =
matrix(c(3, 1, 1, 2), nrow = 2, byrow = TRUE))
bn <- cbind(x0, x1, x2, z, realV) %*% cbind(eta1, eta2) + err
Lambda_T1 <- bn[,1]; Lambda_T2 <- bn[,2]
x.ind = (Lambda_T1>0)
y.ind <- (Lambda_T2>0)
T1 <- rep(0,length(Lambda_T1))
T2 <- rep(0,length(Lambda_T2))
T1[x.ind] = ((theta[1]*Lambda_T1[x.ind]+1)^(1/theta[1])-1)
T1[!x.ind] = 1-(1-(2-theta[1])*Lambda_T1[!x.ind])^(1/(2-theta[1]))
T2[y.ind] = ((theta[2]*Lambda_T2[y.ind]+1)^(1/theta[2])-1)
T2[!y.ind] = 1-(1-(2-theta[2])*Lambda_T2[!y.ind])^(1/(2-theta[2]))
# Generate adminstrative censoring time
C <- runif(n, 0, 40)

# Create observed data set
y <- pmin(T1, T2, C)
delta1 <- as.numeric(T1 == y)
delta2 <- as.numeric(T2 == y)
da <- as.numeric(C == y)
data <- data.frame(cbind(y, delta1, delta2, da, x0, x1, x2, z, w))
colnames(data) <- c("y", "delta1", "delta2", "da", "x0", "x1", "x2", "z", "w")

# Estimate the model
```

```
admin <- TRUE               # There is administrative censoring in the data.
conf <- TRUE                # There is confounding in the data (z)
eoi.indicator.names <- NULL # We will not impose that T1 and T2 are independent
Zbin <- FALSE               # The confounding variable z is not binary
inst <- "cf"                # Use the control function approach
# Since we don't use the oracle estimator, this argument is ignored anyway
realV <- NULL
estimate.cmprsk(data, admin, conf, eoi.indicator.names, Zbin, inst, realV)
```

IYJtrans   *Inverse Yeo-Johnson transformation function*

Description

Computes the inverse Yeo-Johnson transformation of the provided argument.

Usage

```
IYJtrans(y, theta)
```

Arguments

y	The argument to be supplied to the inverse Yeo-Johnson transformation.
theta	The parameter of the inverted Yeo-Johnson transformation. This should be a number in the range [0,2].

Value

The transformed value of y.

LikF.cmprsk   *Second step log-likelihood function.*

Description

This function defines the log-likelihood used to estimate the second step in the competing risks extension of the model described in Willems et al. (2024+).

Usage

```
LikF.cmprsk(par, data, admin, conf, cf)
```

Arguments

par	Vector of all second step model parameters, consisting of the regression parameters, variance-covariance matrix elements and transformation parameters.
data	Data frame resulting from the 'uniformize.data.R' function.
admin	Boolean value indicating whether the data contains administrative censoring.
conf	Boolean value indicating whether the data contains confounding and hence indicating the presence of Z and W.
cf	"Control function" to be used. This can either be the (i) estimated control function, (ii) the true control function, (iii) the instrumental variable, or (iv) nothing (cf = NULL). Option (ii) is used when comparing the two-step estimator to the oracle estimator, and option (iii) is used to compare the two-step estimator with the naive estimator.

Value

Log-likelihood evaluation of the second step.

References

Willems et al. (2024+). Flexible control function approach under competing risks (in preparation).

---

likF.cmprsk.Cholesky	*Wrapper implementing likelihood function using Cholesky factorization.*

Description

This function parametrizes the covariance matrix using its Cholesky decomposition, so that optimization of the likelihood can be done based on this parametrization, and positive-definiteness of the covariance matrix is guaranteed at each step of the optimization algorithm.

Usage

```
likF.cmprsk.Cholesky(par.chol, data, admin, conf, cf, eps = 0.001)
```

Arguments

par.chol	Vector of all second step model parameters, consisting of the regression parameters, Cholesky decomposition of the variance-covariance matrix elements and transformation parameters.
data	Data frame resulting from the 'uniformize.data.R' function.
admin	Boolean value indicating whether the data contains administrative censoring.
conf	Boolean value indicating whether the data contains confounding and hence indicating the presence of Z and W.

cf   "Control function" to be used. This can either be the (i) estimated control func-
                        tion, (ii) the true control function, (iii) the instrumental variable, or (iv) nothing
                        (cf = NULL). Option (ii) is used when comparing the two-step estimator to the
                        oracle estimator, and option (iii) is used to compare the two-step estimator with
                        the naive estimator.

eps   Minimum value for the diagonal elements in the covariance matrix. Default is
                        eps = 0.001.

Value

Log-likelihood evaluation of the second step.

LikGamma1   *First step log-likelihood function for Z continuous*

Description

This function defines the maximum likelihood used to estimate the control function in the case of a
continuous endogenous variable.

Usage

```
LikGamma1(gamma, Z, M)
```

Arguments

gamma   Vector of coefficients in the linear model used to estimate the control function.

Z   Endogenous covariate.

M   Design matrix, containing an intercept, the exogenous covariates and the instru-
                        mental variable.

Value

Returns the log-likelihood function corresponding to the data, evaluated at the point gamma.

LikGamma2   *First step log-likelihood function for Z binary.*

Description

This function defines the maximum likelihood used to estimate the control function in the case of a binary endogenous variable.

Usage

```
LikGamma2(gamma, Z, M)
```

Arguments

gamma   Vector of coefficients in the logistic model used to estimate the control function.

Z   Endogenous covariate.

M   Design matrix, containing an intercept, the exogenous covariates and the instrumental variable.

Value

Returns the log-likelihood function corresponding to the data, evaluated at the point gamma.

LikI.bis   *Second likelihood function needed to fit the independence model in the*
                             second step of the estimation procedure.

Description

This function defines the log-likelihood used in estimating the second step in the competing risks extension of the model described in Willems et al. (2024+). The results of this function will serve as starting values for subsequent optimizations (LikI.comprsk.R and LikF.cmprsk.R)

Usage

```
LikI.bis(par, data, admin, conf, cf)
```

Arguments

par   Vector of all second step model parameters, consisting of the regression parameters, variance-covariance matrix elements and transformation parameters.

data   Data frame resulting from the 'uniformize.data.R' function.

admin   Boolean value indicating whether the data contains administrative censoring.

conf   Boolean value indicating whether the data contains confounding and hence in-
                dicating the presence of Z and W

cf   "Control function" to be used. This can either be the (i) estimated control func-
                tion, (ii) the true control function, (iii) the instrumental variable, or (iv) nothing
                (cf = NULL). Option (ii) is used when comparing the two-step estimator to the
                oracle estimator, and option (iii) is used to compare the two-step estimator with
                the naive estimator.

Value

Starting values for subsequent optimization function used in the second step of the estimation pro-
cedure.

References

Willems et al. (2024+). Flexible control function approach under competing risks (in preparation).

LikI.cmprsk   *Second step log-likelihood function under independence assumption.*

Description

This function defines the log-likelihood used to estimate the second step in the competing risks
extension assuming independence of some of the competing risks in the model described in Willems
et al. (2024+).

Usage

```
LikI.cmprsk(par, data, eoi.indicator.names, admin, conf, cf)
```

Arguments

par   Vector of all second step model parameters, consisting of the regression param-
                eters, variance-covariance matrix elements and transformation parameters.

data   Data frame resulting from the 'uniformize.data.R' function.

eoi.indicator.names

                Vector of names of the censoring indicator columns pertaining to events of in-
                terest. Events of interest will be modeled allowing dependence between them,
                whereas all censoring events (corresponding to indicator columns not listed in
                eoi.indicator.names) will be treated as independent of every other event.

admin   Boolean value indicating whether the data contains administrative censoring.

conf   Boolean value indicating whether the data contains confounding and hence in-
                dicating the presence of Z and W

cf   "Control function" to be used. This can either be the (i) estimated control func-
                tion, (ii) the true control function, (iii) the instrumental variable, or (iv) nothing
                (cf = NULL). Option (ii) is used when comparing the two-step estimator to the
                oracle estimator, and option (iii) is used to compare the two-step estimator with
                the naive estimator.

Value

Log-likelihood evaluation for the second step in the esimation procedure.

References

Willems et al. (2024+). Flexible control function approach under competing risks (in preparation).

LikI.cmprsk.Cholesky   *Wrapper implementing likelihood function assuming independence between competing risks and censoring using Cholesky factorization.*

Description

This function does the same as LikI.cmprsk (in fact, it even calls said function), but it parametrizes the covariance matrix using its Cholesky decomposition in order to guarantee positive definiteness. This function is never used, might not work and could be deleted.

Usage

```
LikI.cmprsk.Cholesky(
  par.chol,
  data,
  eoi.indicator.names,
  admin,
  conf,
  cf,
  eps = 0.001
)
```

Arguments

par.chol
:   Vector of all second step model parameters, consisting of the regression parameters, Cholesky decomposition of the variance-covariance matrix elements and transformation parameters.

data
:   Data frame resulting from the 'uniformize.data.R' function.

eoi.indicator.names
:   Vector of names of the censoring indicator columns pertaining to events of interest. Events of interest will be modeled allowing dependence between them, whereas all censoring events (corresponding to indicator columns not listed in eoi.indicator.names) will be treated as independent of every other event.

admin
:   Boolean value indicating whether the data contains administrative censoring.

conf
:   Boolean value indicating whether the data contains confounding and hence indicating the presence of Z and W.

cf   "Control function" to be used. This can either be the (i) estimated control function, (ii) the true control function, (iii) the instrumental variable, or (iv) nothing (cf = NULL). Option (ii) is used when comparing the two-step estimator to the oracle estimator, and option (iii) is used to compare the two-step estimator with the naive estimator.

eps   Minimum value for the diagonal elements in the covariance matrix. Default is eps = 0.001.

Value

Log-likelihood evaluation for the second step in the estimation procedure.

---

likIFG.cmprsk.Cholesky

Full likelihood (including estimation of control function).

Description

This function defines the 'full' likelihood of the model. Specifically, it includes the estimation of the control function in the computation of the likelihood. This function is used in the estimation of the variance of the estimates (variance.cmprsk.R).

Usage

```
likIFG.cmprsk.Cholesky(
  parhatG,
  data,
  eoi.indicator.names,
  admin,
  conf,
  Zbin,
  inst
)
```

Arguments

parhatG   The full parameter vector.

data   Data frame.

eoi.indicator.names

                       Vector of names of the censoring indicator columns pertaining to events of interest. Events of interest will be modeled allowing dependence between them, whereas all censoring events (corresponding to indicator columns not listed in eoi.indicator.names) will be treated as independent of every other event. If eoi.indicator.names == NULL, all events will be modelled dependently.

admin   Boolean value indicating whether the data contains administrative censoring.

conf	Boolean value indicating whether the data contains confounding and hence indicating the presence of Z and W.
Zbin	Boolean value indicating whether the confounding variable is binary.
inst	Type of instrumental function to be used.

Value

Full model log-likelihood evaluation.

---

loglike.clayton.unconstrained

Log-likelihood function for the Clayton copula.

Description

This likelihood function is maximized to estimate the model parameters under the Clayton copula.

Usage

```
loglike.clayton.unconstrained(para, Y, Delta, Dist.T, Dist.C)
```

Arguments

para	Estimated parameter values/initial values.
Y	Follow-up time.
Delta	Censoring indicator.
Dist.T	The distribution to be used for the survival time T. This argument can take one of the values from c("lnorm", "weibull") and has to be the same as Dist.C.
Dist.C	The distribution to be used for the censoring time C. This argument can take one of the values from c("lnorm", "weibull") and has to be the same as Dist.T.

Value

Maximized log-likelihood value.

`loglike.frank.unconstrained`
Log-likelihood function for the Frank copula.

Description

This likelihood function is maximized to estimate the model parameters under the Frank copula.

Usage

```
loglike.frank.unconstrained(para, Y, Delta, Dist.T, Dist.C)
```

Arguments

para	Estimated parameter values/initial values.
Y	Follow-up time.
Delta	Censoring indicator.
Dist.T	The distribution to be used for the survival time T. This argument can take one of the values from c("lnorm", "weibull", "llogis").
Dist.C	The distribution to be used for the censoring time C. This argument can take one of the values from c("lnorm", "weibull", "llogis").

Value

Maximized log-likelihood value.

`loglike.gaussian.unconstrained`
Log-likelihood function for the Gaussian copula.

Description

This likelihood function is maximized to estimate the model parameters under the Gaussian copula.

Usage

```
loglike.gaussian.unconstrained(para, Y, Delta, Dist.T, Dist.C)
```

Arguments

para	Estimated parameter values/initial values.
Y	Follow-up time.
Delta	Censoring indicator.
Dist.T	The distribution to be used for the survival time T. This argument can only the value "lnorm".
Dist.C	The distribution to be used for the censoring time C. This argument can only the value "lnorm".

Value

Maximized log-likelihood value.

---

loglike.gumbel.unconstrained

Log-likelihood function for the Gumbel copula.

Description

This likelihood function is maximized to estimate the model parameters under the Gumbel copula.

Usage

```
loglike.gumbel.unconstrained(para, Y, Delta, Dist.T, Dist.C)
```

Arguments

para	Estimated parameter values/initial values.
Y	Follow-up time.
Delta	Censoring indicator.
Dist.T	The distribution to be used for the survival time T. This argument can take one of the values from c("lnorm", "weibull") and has to be the same as Dist.C.
Dist.C	The distribution to be used for the censoring time C. This argument can take one of the values from c("lnorm", "weibull") and has to be the same as Dist.T.

Value

Maximized log-likelihood value.

---

loglike.indep.unconstrained
                     Log-likelihood function for the independence copula.

Description

This likelihood function is maximized to estimate the model parameters under the independence copula.

Usage

```
loglike.indep.unconstrained(para, Y, Delta, Dist.T, Dist.C)
```

Arguments

para	Estimated parameter values/initial values.
Y	Follow-up time.
Delta	Censoring indicator.
Dist.T	The distribution to be used for the survival time T. This argument can take one of the values from c("lnorm", "weibull", "llogis").
Dist.C	The distribution to be used for the censoring time C. This argument can take one of the values from c("lnorm", "weibull", "llogis").

Value

Maximized log-likelihood value.

---

log_transform   *Logarithmic transformation function.*

Description

Computes the logarithm of a number.

Usage

```
log_transform(y)
```

Arguments

y	Numerical value of which the logarithm is computed.

Value

This function returns the logarithm of the provided argument y if it is greater than zero. If y is smaller than zero, it will return 0.

Longfun *Change H to long format*

Description

Change a nonparametric transformation function to long format

Usage

```
Longfun(Z, T1, H)
```

Arguments

Z	Observed survival time, which is the minimum of T, C and A, where A is the administrative censoring time.
T1	Distinct observed survival time
H	Nonparametric transformation function estimate

NonParTrans *Fit a semiparametric transformation model for dependent censoring*

Description

This function allows to estimate the dependency parameter along all other model parameters. First, estimates a non-parametric transformation function, and then at the second stage it estimates other model parameters assuming that the non-parametric function is known. The details for implementing the dependent censoring methodology can be found in Deresa and Van Keilegom (2021).

Usage

```
NonParTrans(
  resData,
  X,
  W,
  start = NULL,
  n.iter = 15,
  bootstrap = FALSE,
  n.boot = 50,
  eps = 0.001
)
```

Arguments

`resData`	Data matrix with three columns; Z = the observed survival time, d1 = the censoring indicator of T and d2 = the censoring indicator of C.
`X`	Data matrix with covariates related to T
`W`	Data matrix with covariates related to C
`start`	Initial values for the finite dimensional parameters. If `start` is NULL, the initial values will be obtained by fitting an Accelerated failure time models.
`n.iter`	Number of iterations; the default is `n.iter = 20`. The larger the number of iterations, the longer the computational time.
`bootstrap`	A boolean indicating whether to compute bootstrap standard errors for making inferences.
`n.boot`	Number of bootstrap samples to use in the estimation of bootstrap standard errors if `bootstrap = TRUE`. The default is n.boot = 50. But, higher values of `n.boot` are recommended for obtaining good estimates of bootstrap standard errors.
`eps`	Convergence error. This is set by the user in such away that the desired convergence is met; the default is `eps = 1e-3`.

Value

This function returns a fit of a semiparametric transformation model; parameter estimates, estimate of the non-parametric transformation function, bootstrap standard errors for finite-dimensional parameters, the nonparametric cumulative hazard function, etc.

References

Deresa, N. and Van Keilegom, I. (2021). On semiparametric modelling, estimation and inference for survival data subject to dependent censoring, Biometrika, 108, 965–979.

Examples

```
# Toy data example to illustrate implementation
n = 300
beta = c(0.5, 1); eta = c(1,1.5); rho = 0.70
sigma = matrix(c(1,rho,rho,1),ncol=2)
err = MASS::mvrnorm(n, mu = c(0,0) , Sigma=sigma)
err1 = err[,1]; err2 = err[,2]
x1 = rbinom(n,1,0.5); x2 = runif(n,-1,1)
X = matrix(c(x1,x2),ncol=2,nrow=n); W = X    # data matrix
T1 = X%*%beta+err1
C =  W%*%eta+err2
T1 = exp(T1); C = exp(C)
A = runif(n,0,8); Y = pmin(T1,C,A)
d1 = as.numeric(Y==T1)
d2 = as.numeric(Y==C)
resData = data.frame("Z" = Y,"d1" = d1, "d2" = d2)   # should be data frame
colnames(X) = c("X1", "X2")
colnames(W) = c("W1","W2")
```

```
# Bootstrap is false by default
output = NonParTrans(resData = resData, X = X, W = W, n.iter = 2)
output$parameterEstimates
```

---

optimlikelihood   *Fit the dependent censoring models.*

Description

Estimates the model parameters by maximizing the log-likelihood.

Usage

```
optimlikelihood(Y, Delta, Copula, Dist.T, Dist.C, start)
```

Arguments

Y	Follow-up time.
Delta	Censoring indicator.
Copula	The copula family. This argument can take values from c("frank","gumbel","clayton","gaussian",
Dist.T	The distribution to be used for the survival time T. This argument can take one of the values from c("lnorm", "weibull", "llogis").
Dist.C	The distribution to be used for the censoring time C. This argument can take one of the values from c("lnorm", "weibull", "llogis").
start	Starting values

Value

A list containing the minimized negative log-likelihood using the independence copula model, the estimated parameter values for the model with the independence copula, the minimized negative log-likelihood using the specified copula model and the estimated parameter values for the model with the specified copula.

ParamCop	*Estimation of a parametric dependent censoring model without co-variates.*

Description

Note that it is not assumed that the association parameter of the copula function is known, unlike most other papers in the literature. The details for implementing the methodology can be found in Czado and Van Keilegom (2023).

Usage

```
ParamCop(Y, Delta, Copula, Dist.T, Dist.C, start = c(1, 1, 1, 1))
```

Arguments

Y	Follow-up time.
Delta	Censoring indicator.
Copula	The copula family. This argument can take values from c("frank","gumbel","clayton","gaussian",
Dist.T	The distribution to be used for the survival time T. This argument can take one of the values from c("lnorm", "weibull", "llogis").
Dist.C	The distribution to be used for the censoring time C. This argument can take one of the values from c("lnorm", "weibull", "llogis").
start	Starting values

Value

A table containing the minimized negative log-likelihood using the independence copula model, the estimated parameter values for the model with the independence copula, the minimized negative log-likelihood using the specified copula model and the estimated parameter values for the model with the specified copula.

References

Czado and Van Keilegom (2023). Dependent censoring based on parametric copulas. Biometrika, 110(3), 721-738.

Examples

```
tau = 0.75
Copula = "frank"
Dist.T = "weibull"
Dist.C = "lnorm"
par.T = c(2,1)
par.C = c(1,2)
n=1000
```

```
simdata<-TCsim(tau,Copula,Dist.T,Dist.C,par.T,par.C,n)

Y = simdata[[1]]
Delta = simdata[[2]]

ParamCop(Y,Delta,Copula,Dist.T,Dist.C)
```

power_transform	*Power transformation function.*

Description

Computes a given power of a number.

Usage

```
power_transform(y, pw)
```

Arguments

y	The number which one wants to raise to a certain power pw.
pw	The power to which to raise y.

Value

This function returns the result of raising y to the power pw when y > 0. Otherwise, it will return 1.

ScoreEqn	*Score equations of finite parameters*

Description

This function computes the score vectors and the Jacobean matrix for finite model parameters.

Usage

```
ScoreEqn(theta, resData, X, W, H)
```

Arguments

theta	Vector of parameters in the semiparametric transformation model.
resData	Data matrix with three columns; Z = the observed survival time, d1 = the censoring indicator of T and d2 = the censoring indicator of C.
X	Data matrix with covariates related to T.
W	Data matrix with covariates related to C.
H	The estimated non-parametric transformation function for a given value of theta

---

SearchIndicate   *Search function*

Description

Function to indicate position of t in observed survival time

Usage

```
SearchIndicate(t, T1)
```

Arguments

t	fixed time t
T1	distinct observed survival time

SolveH   *Estimate a nonparametric transformation function*

Description

This function estimates the nonparametric transformation function H when the survival time and censoring time are dependent given covariates. The estimating equation of H was derived based on the martingale ideas. More details about the derivation of a nonparmaetric estimator of H and its estimation algorithm can be found in Deresa and Van Keilegom (2021).

Usage

```
SolveH(theta, resData, X, W)
```

Arguments

theta	Vector of parameters in the semiparametric transformation model.
resData	Data matrix with three columns; Z = the observed survival time, d1 = the censoring indicator of T and d2 = the censoring indicator of C.
X	Data matrix with covariates related to T.
W	Data matrix with covariates related to C.

Value

Returns the estimated transformation function H for a fixed value of parameters theta.

References

Deresa, N. and Van Keilegom, I. (2021). On semiparametric modelling, estimation and inference for survival data subject to dependent censoring, Biometrika, 108, 965–979.

SolveHt1 *Estimating equation for Ht1*

Description

This function obtains an estimating equation of H at the first observed survival time t1.

Usage

```
SolveHt1(Ht1, Z, nu, t, X, W, theta)
```

Arguments

Ht1	The solver solves for an optimal value of Ht1 by equating the estimating equation to zero.
Z	The observed survival time, which is the minimum of T, C and A.
nu	The censoring indicator for T or C
t	A fixed time point
X	Data matrix with covariates related to T.
W	Data matrix with covariates related to C.
theta	Vector of parameters

SolveScore *Estimate finite parameters based on score equations*

Description

This function estimates the model parameters

Usage

```
SolveScore(theta, resData, X, W, H, eps = 0.001)
```

Arguments

theta	Vector of parameters in the semiparametric transformation model.
resData	Data matrix with three columns; Z = the observed survival time, d1 = the censoring indicator of T and d2 = the censoring indicator of C.
X	Data matrix with covariates related to T.
W	Data matrix with covariates related to C.
H	The estimated non-parametric transformation function for a given value of theta.
eps	Convergence error.

TCsim *Function to simulate (Y,Delta) from the copula based model for (T,C).*

Description

Generates the follow-up time and censoring indicator according to the specified model.

Usage

```
TCsim(
  tau = 0,
  Copula = "frank",
  Dist.T = "lnorm",
  Dist.C = "lnorm",
  par.T = c(0, 1),
  par.C = c(0, 1),
  n = 10000
)
```

Arguments

tau	Value of Kendall's tau for (T,C). The default value is 0.
Copula	The copula family. This argument can take values from c("frank","gumbel","clayton","gaussian", The default copula model is "frank".
Dist.T	Distribution of the survival time T. This argument can take one of the values from c("lnorm", "weibull", "llogis"). The default distribution is "lnorm".
Dist.C	Distribution of the censoring time C. This argument can take one of the values from c("lnorm", "weibull", "llogis"). The default distribution is "lnorm".
par.T	Parameter values for the distribution of T.
par.C	Parameter values for the distribution of C.
n	Sample size.

Value

A list containing the generated follow-up times and censoring indicators.

Examples

```
tau = 0.5
Copula = "gaussian"
Dist.T = "lnorm"
Dist.C = "lnorm"
par.T = c(1,1)
par.C = c(2,2)
n=1000
```

```
simdata <- TCsim(tau,Copula,Dist.T,Dist.C,par.T,par.C,n)
Y = simdata[[1]]
Delta = simdata[[2]]
hist(Y)
mean(Delta)
```

---

uniformize.data *Standardize data format*

Description

Checks the required preconditions of the data and possibly restructures the data.

Usage

```
uniformize.data(
  data,
  admin = FALSE,
  conf = FALSE,
  comp.risks = FALSE,
  Zbin = NULL,
  Wbin = NULL
)
```

Arguments

data	A data frame that should contain columns named Y and delta (unless comp.risks = TRUE, see later).
admin	Boolean value indicating whether the provided data frame contains administrative (i.e. independent) censoring on top of the dependent censoring (in the column named delta). The default is admin = FALSE.
conf	Boolean value indicating whether the provided data frame contains a confounded variable and a corresponding instrument. If cond = TRUE, the provided data frame should contain columns named Z and W, corresponding to the confounded variable and instrument, respectively. Moreover, Zbin and Wbin should be specified. The default value is conf = FALSE.
comp.risks	Boolean value indicating whether the provided data frame contains competing risks. If comp.risks = TRUE, the given data frame should contain the columns delta1, delta2, etc., corresponding to the indicators $I(Y = T1)$, $I(Y = T2)$, etc. respectively. The default is comp.risks = FALSE.
Zbin	Boolean or integer value (0, 1) indicating whether the confounded variable is binary. Zbin = TRUE or Zbin = 1 means that Z is binary. Zbin = FALSE or Zbin = 0 means that Z is continuous.
Wbin	Boolean or integer value (0, 1) indicating whether the instrument is binary. Wbin = TRUE or Wbin = 1 means that W is binary. Wbin = FALSE or Wbin = 0 means that W is continuous.

Value

Returns the uniformized data set.

---

variance.cmprsk *Compute the variance of the estimates.*

Description

This function computes the variance of the estimates computed by the 'estimate.cmprsk.R' function.

Usage

```
variance.cmprsk(
  parhatc,
  gammaest,
  data,
  admin,
  conf,
  inst,
  cf,
  eoi.indicator.names,
  Zbin,
  use.chol,
  n.trans,
  totparl
)
```

Arguments

parhatc	Vector of estimated parameters, computed in the first part of `estimate.cmprsk.R`.
gammaest	Vector of estimated parameters in the regression model for the control function.
data	A data frame.
admin	Boolean value indicating whether the data contains administrative censoring.
conf	Boolean value indicating whether the data contains confounding and hence indicating the presence of z and, possibly, w.
inst	Variable encoding which approach should be used for dealing with the confounding. `inst = "cf"` indicates that the control function approach should be used. `inst = "W"` indicates that the instrumental variable should be used 'as is'. `inst = "None"` indicates that Z will be treated as an exogenous covariate. Finally, when `inst = "oracle"`, this function will access the argument realV and use it as the values for the control function. Default is `inst = "cf"`.
cf	The control function used to estimate the second step.

eoi.indicator.names

> Vector of names of the censoring indicator columns pertaining to events of interest. Events of interest will be modeled allowing dependence between them, whereas all censoring events (corresponding to indicator columns not listed in `eoi.indicator.names`) will be treated as independent of every other event. If `eoi.indicator.names == NULL`, all events will be modeled dependently.

Zbin   Indicator value indicating whether (`Zbin = TRUE`) or not `Zbin = FALSE` the endogenous covariate is binary. Default is `Zbin = NULL`, corresponding to the case when `conf == FALSE`.

use.chol   Boolean value indicating whether the cholesky decomposition was used in estimating the covariance matrix.

n.trans   Number of competing risks in the model (and hence, number of transformation models).

totparl   Total number of covariate effects (including intercepts) in all of the transformation models combined.

Value

Variance estimates of the provided vector of estimated parameters.

YJtrans   *Yeo-Johnson transformation function*

Description

Computes the Yeo-Johnson transformation of the provided argument.

Usage

```
YJtrans(y, theta)
```

Arguments

y   The argument to be supplied to the Yeo-Johnson transformation.

theta   The parameter of the Yeo-Johnson transformation. This should be a number in the range [0,2].

Value

The transformed value of y.

Index