Package ‘stdReg’

February 22, 2018

Type Package
Title Regression Standardization
Version 3.0.0
Date 2018-02-22
Author Arvid Sjolander and Elisabeth Dahlqwist
Maintainer Arvid Sjolander <arvid.sjolander@ki.se>
Description Contains functionality for regression standardization.
   Four general classes of models are allowed; generalized linear models,
   Conditional generalized estimating equation models,
   Cox proportional hazards models and shared frailty gamma-Weibull models.
License LGPL (>= 3)
Imports graphics, stats, survival, data.table, numDeriv, drgee
NeedsCompilation no
RoxygenNote 6.0.1
Repository CRAN
Date/Publication 2018-02-22 17:34:28 UTC

\r topics documented:

stdReg-package .................................................... 2
confint.stdCoxph .................................................. 4
confint.stdGee ..................................................... 5
confint.stdGlm .................................................... 6
confint.stdParfrailty ............................................. 7
parfrailty .......................................................... 8
plot.stdCoxph ..................................................... 10
plot.stdGee ....................................................... 11
plot.stdGlm ....................................................... 12
plot.stdParfrailty ................................................. 13
print.summary.parfrailty ....................................... 14
print.summary.stdCoxph ......................................... 14
print.summary.stdGee ............................................ 15
Description

Contains functionality for regression standardization. Four general classes of models are allowed; generalized linear models, Conditional generalized estimating equation models, Cox proportional hazards models and shared frailty gamma-Weibull models.

Details

The DESCRIPTION file:

Package: stdReg
Type: Package
Title: Regression Standardization
Version: 3.0.0
Date: 2018-02-22
Author: Arvid Sjolander and Elisabeth Dahlqwist
Maintainer: Arvid Sjolander <arvid.sjolander@ki.se>
Description: Contains functionality for regression standardization. Four general classes of models are allowed; generalized linear models, Conditional generalized estimating equation models, Cox proportional hazards models and shared frailty gamma-Weibull models.
License: LGPL (>=3)
Imports: graphics, stats, survival, data.table, numDeriv, drgee
NeedsCompilation: no
Packaged: 2018-02-21 12:58:49 UTC; arvsjo
RoxygenNote: 6.0.1

Index of help topics:

confint.stdCoxph  Confidence interval
confint.stdGee    Confidence interval
confint.stdGlm    Confidence interval
confint.stdParfrailty  Confidence interval
parfrailty        Fits shared frailty gamma-Weibull models
plot.stdCoxph  Plots Cox regression standardization fit
plot.stdGee   Plots GEE regression standardization fit
plot.stdGlm   Plots GLM regression standardization fit
plot.stdParfrailty Plots parfrailty standardization fit
print.summary.parfrailty Prints summary of parfrailty fit
print.summary.stdCoxph Prints summary of Cox regression standardization fit
print.summary.stdGee Prints summary of GEE regression standardization fit
print.summary.stdGlm Prints summary of GLM regression standardization fit
print.summary.stdParfrailty Prints summary of Frailty standardization fit
stdCoxph Regression standardization in Cox proportional hazards models
stdGee  Regression standardization in conditional generalized estimating equations
stdGlm   Regression standardization in generalized linear models
stdParfrailty Regression standardization in shared frailty gamma-Weibull models
stdReg-package Regression Standardization
summary.parfrailty Summarizes parfrailty fit
summary.stdCoxph Summarizes Cox regression standardization fit
summary.stdGee Summarizes GEE regression standardization fit
summary.stdParfrailty Summarizes Frailty standardization fit

~~ An overview of how to use the package, including the most important ~~ ~~ functions ~~

Author(s)

Arvid Sjolander and Elisabeth Dahlqwist

Maintainer: Arvid Sjolander <arvid.sjolander@ki.se>

References


confint.stdCoxph

Description

This is a confint method for class "stdCoxph".

Usage

## S3 method for class 'stdCoxph'

confint(object, parm, level = 0.95, fun, type="plain", ...)

Arguments

- **object**: an object of class "stdCoxph".
- **parm**: not used.
- **level**: the coverage probability of the confidence intervals.
- **fun**: a function of one matrix argument with q rows and p columns, which returns a vector of length q.
- **type**: a string specifying the type of confidence interval; plain (for untransformed) or log (for log-transformed).
- **...**: not used.

Details

confint.stdCoxph extracts the est element from object, and inputs this to fun. It then uses the delta method to compute a confidence interval for the output of fun.

Value

A matrix with q rows and 2 columns, containing the computed confidence interval.

Author(s)

Arvid Sjolander.
Description

This is a `confint` method for class "stdGee".

Usage

```r
## S3 method for class 'stdGee'
confint(object, parm, level = 0.95, fun, type="plain", ...)
```

Arguments

- `object`: an object of class "stdGee".
- `parm`: not used.
- `level`: the coverage probability of the confidence intervals.
- `fun`: a function of one vector argument of length p, which returns a scalar.
- `type`: a string specifying the type of confidence interval; `plain` (for untransformed) or `log` (for log-transformed).
- `...`: not used.

Details

`confint.stdGee` extracts the `est` element from `object`, and inputs this to `fun`. It then uses the delta method to compute a confidence interval for the output of `fun`.

Value

- a matrix with 1 row and 2 columns, containing the computed confidence interval.

Author(s)

Arvid Sjolander.
Description

This is a confint method for class "stdGlm".

Usage

```r
## S3 method for class 'stdGlm'
confint(object, parm, level = 0.95, fun = "plain", ...)  
```

Arguments

- `object`: an object of class "stdGlm".
- `parm`: not used.
- `level`: the coverage probability of the confidence intervals.
- `fun`: a function of one vector argument of length p, which returns a scalar.
- `type`: a string specifying the type of confidence interval; plain (for untransformed) or log (for log-transformed).
- `...`: not used.

Details

`confint.stdGlm` extracts the `est` element from `object`, and inputs this to `fun`. It then uses the delta method to compute a confidence interval for the output of `fun`.

Value

A matrix with 1 row and 2 columns, containing the computed confidence interval.

Author(s)

Arvid Sjolander.
Description

This is a `confint` method for class "stdParfrailty".

Usage

```r
## S3 method for class 'stdParfrailty'
confint(object, parm = NULL, level = 0.95, fun = NULL, type = "plain", ...)
```

Arguments

- `object`: an object of class "stdParfrailty".
- `parm`: not used.
- `level`: the coverage probability of the confidence intervals.
- `fun`: a function of one matrix argument with q rows and p columns, which returns a vector of length q.
- `type`: a string specifying the type of confidence interval; "plain" (for untransformed) or "log" (for log-transformed).
- `...`: not used.

Details

`confint.stdParfrailty` extracts the `est` element from `object`, and inputs this to `fun`. It then uses the delta method to compute a confidence interval for the output of `fun`.

Value

a matrix with q rows and 2 columns, containing the computed confidence interval.

Author(s)

Arvid Sjolander.
**Description**

`parfrailty` fits shared frailty gamma-Weibull models. It is specifically designed to work with the function `stdParfrailty`, which performs regression standardization in shared frailty gamma-Weibull models.

**Usage**

```r
parfrailty(formula, data, clusterid, init)
```

**Arguments**

- **formula**: an object of class "formula", on the same format as accepted by the `coxph` function in the `survival` package.
- **data**: a data frame containing the variables in the model.
- **clusterid**: an string containing the name of a cluster identification variable.
- **init**: an optional vector of initial values for the model parameters.

**Details**

`parfrailty` fits the shared frailty gamma-Weibull model

\[
\lambda(t_{ij}|C_{ij}) = \lambda(t_{ij}; \alpha, \eta) U_i \exp\{h(C_{ij}; \beta)\},
\]

where \( t_{ij} \) and \( C_{ij} \) are the survival time and covariate vector for subject \( j \) in cluster \( i \), respectively. \( \lambda(t; \alpha, \eta) \) is the Weibull baseline hazard function

\[
\eta \eta^{\alpha - 1} \alpha^{-\eta},
\]

where \( \eta \) is the shape parameter and \( \alpha \) is the scale parameter. \( U_i \) is the unobserved frailty term for cluster \( i \), which is assumed to have a gamma distribution with scale = 1/shape = \( \phi \). \( h(X; \beta) \) is the regression function as specified by the `formula` argument, parametrized by a vector \( \beta \). The ML estimates \( \{\log(\hat{\alpha}), \log(\hat{\eta}), \log(\hat{\phi}), \hat{\beta}\} \) are obtained by maximizing the marginal (over \( U \)) likelihood.

**Value**

An object of class "parfrailty" is a list containing:

- **est**: the ML estimates \( \{\log(\hat{\alpha}), \log(\hat{\eta}), \log(\hat{\phi}), \hat{\beta}\} \).
- **vcov**: the variance-covariance vector of the ML estimates.
- **score**: a matrix containing the cluster-specific contributions to the ML score equations.
Note

If left truncation is present, it is assumed that it is strong left truncation. This means that, even if the truncation time may be subject-specific, the whole cluster is unobserved if at least one subject in the cluster dies before his/her truncation time. If all subjects in the cluster survive beyond their subject-specific truncation times, then the whole cluster is observed (Van den Berg and Drepper, 2016).

Author(s)

Arvid Sjolander and Elisabeth Dahlqwist.

References


Examples

```r
require(survival)

# simulate data
n <- 1000
m <- 3
alpha <- 1.5
eta <- 1
phi <- 0.5
beta <- 1
id <- rep(1:n, each=m)
U <- rep(rgamma(n, shape=1/phi, scale=phi), each=m)
X <- rnorm(n*m)
# reparametrize scale as in rweibull function
weibull.scale <- alpha/(U*exp(beta*X))^(1/eta)
T <- rweibull(n*m, shape=eta, scale=weibull.scale)

c <- rnorm(n*m, 0, 10)
D <- as.numeric(T<=C)
T <- pmin(T, C)

# strong left-truncation
L <- runif(n*m, 0, 2)
incl <- T>L
incl <- ave(x=incl, id, FUN=sum)==m
dd <- data.frame(L, T, D, X, id)
fit <- parfrailty(formula=Surv(L, T, D)~X, data=dd, clusterid="id")
```
print(summary(fit))

---

**plot.stdCoxph**

**Plots Cox regression standardization fit**

**Description**

This is a plot method for class "stdCoxph".

**Usage**

```r
## S3 method for class 'stdCoxph'
plot(x, plot.CI = TRUE, CI.type = "plain", CI.level = 0.95,
     transform = NULL, contrast = NULL, reference = NULL, legendpos="bottomleft", ...)
```

**Arguments**

- **x**: an object of class "stdCoxph".
- **plot.CI**: logical, indicating whether confidence intervals should be added to the plot.
- **CI.type**: string, indicating the type of confidence intervals. Either "plain", which gives untransformed intervals, or "log", which gives log-transformed intervals.
- **CI.level**: desired coverage probability of confidence intervals, on decimal form.
- **transform**: a string. If set to "log", "logit", or "odds", the standardized survival function \( \psi(t,x) = \log\{\theta(t,x)\} \), \( \psi(t,x) = \log[\theta(t,x)/(1-\theta(t,x))] \), or \( \psi(t,x) = \theta(t,x)/(1-\theta(t,x)) \), respectively. If left unspecified, \( \psi(t,x) = \theta(t,x) \).
- **contrast**: a string. If set to "difference" or "ratio", then \( \psi(t,x)-\psi(t,x_0) \) or \( \psi(t,x)/\psi(t,x_0) \) are constructed, where \( x_0 \) is a reference level specified by the reference argument.
- **reference**: must be specified if contrast is specified.
- **legendpos**: position of the legend; see help for legend.
- **...**: further arguments passed on to plot.default.

**Author(s)**

Arvid Sjolander

**See Also**

- stdCoxph

**Examples**

```r
##See documentation for stdCoxph
```
Description

This is a plot method for class "stdGee".

Usage

```r
# S3 method for class 'stdGee'
plot(x, CI.type = "plain", CI.level = 0.95,
     transform = NULL, contrast = NULL, reference = NULL, ...)
```

Arguments

- `x` an object of class "stdGee".
- `CI.type` string, indicating the type of confidence intervals. Either "plain", which gives untransformed intervals, or "log", which gives log-transformed intervals.
- `CI.level` desired coverage probability of confidence intervals, on decimal form.
- `transform` a string. If set to "log", "logit", or "odds", the standardized mean \( \theta(x) \) is transformed into \( \psi(x) = \log(\theta(x)) \), \( \psi(x) = \log(\theta(x)/(1-\theta(x))) \), or \( \psi(x) = \theta(x)/(1-\theta(x)) \), respectively. If left unspecified, \( \psi(x) = \theta(x) \).
- `contrast` a string. If set to "difference" or "ratio", then \( \psi(x) - \psi(x_0) \) or \( \psi(x)/\psi(x_0) \) are constructed, where \( x_0 \) is a reference level specified by the reference argument.
- `reference` must be specified if contrast is specified.
- `...` further arguments passed on to plot.default.

Author(s)

Arvid Sjolander

See Also

`stdGee`

Examples

```r
# See documentation for stdGee
```
plot.stdGlm

Plots GLM regression standardization fit

Description

This is a plot method for class "stdGlm".

Usage

```r
## S3 method for class 'stdGlm'
plot(x, CI.type = "plain", CI.level = 0.95,
     transform = NULL, contrast = NULL, reference = NULL, ...)
```

Arguments

- `x` an object of class "stdGlm".
- `CI.type` string, indicating the type of confidence intervals. Either "plain", which gives untransformed intervals, or "log", which gives log-transformed intervals.
- `CI.level` desired coverage probability of confidence intervals, on decimal form.
- `transform` a string. If set to "log", "logit", or "odds", the standardized mean \( \theta(x) \) is transformed into \( \psi(x) = \log(\theta(x)) \), \( \psi(x) = \log(\theta(x)/(1 - \theta(x))) \), or \( \psi(x) = \theta(x)/(1 - \theta(x)) \), respectively. If left unspecified, \( \psi(x) = \theta(x) \).
- `contrast` a string. If set to "difference" or "ratio", then \( \psi(x) - \psi(x_0) \) or \( \psi(x)/\psi(x_0) \) are constructed, where \( x_0 \) is a reference level specified by the `reference` argument.
- `reference` must be specified if `contrast` is specified.
- `...` further arguments passed on to `plot.default`.

Author(s)

Arvid Sjolander

See Also

`stdGlm`

Examples

```r
## See documentation for stdGlm
```
plot.stdParfrailty

Plots parfrailty standardization fit

Description

This is a plot method for class "stdParfrailty".

Usage

## S3 method for class 'stdParfrailty'
plot(x, plot.CI = TRUE, CI.type = "plain", CI.level = 0.95,
transform = NULL, contrast = NULL, reference = NULL, legendpos="bottomleft", ...)

Arguments

x an object of class "stdParfrailty".
plot.CI logical, indicating whether confidence intervals should be added to the plot.
CI.type string, indicating the type of confidence intervals. Either "plain", which gives
untransformed intervals, or "log", which gives log-transformed intervals.
CI.level desired coverage probability of confidence intervals, on decimal form.
transform a string. If set to "log", "logit", or "odds", the standardized survival function
\( \theta(t, x) \) is transformed into \( \psi(t, x) = \log(\theta(t, x)) \), \( \psi(t, x) = \log[\theta(t, x)/(1 - \theta(t, x))] \), or \( \psi(t, x) = \theta(t, x)/(1 - \theta(t, x)) \), respectively. If left unspecified,
\( \psi(t, x) = \theta(t, x) \).
contrast a string. If set to "difference" or "ratio", then \( \psi(t, x) - \psi(t, x_0) \) or \( \psi(t, x)/\psi(t, x_0) \)
are constructed, where \( x_0 \) is a reference level specified by the reference argument.
reference must be specified if contrast is specified.
legendpos position of the legend; see help for legend.
... further arguments passed on to plot.default.

Author(s)

Arvid Sjolander

See Also

stdParfrailty

Examples

##See documentation for stdParfrailty
print.summary.parfrailty

Prints summary of parfrailty fit

Description

This is a print method for class "summary.parfrailty".

Usage

## S3 method for class 'summary.parfrailty'
print(x, digits = max(3L, getOption("digits") - 3L),
      ...)

Arguments

x

an object of class "summary.parfrailty".

digits

the number of significant digits to use when printing.

... not used.

Author(s)

Arvid Sjolander and Elisabeth Dahlqwist

See Also

parfrailty

Examples

## See documentation for frailty

print.summary.stdCoxph

Prints summary of Cox regression standardization fit

Description

This is a print method for class "summary.stdCoxph".

Usage

## S3 method for class 'summary.stdCoxph'
print(x, ...)

Description

This is a print method for class "summary.stdGee".

Usage

## S3 method for class 'summary.stdGee'
print(x, ...)

Arguments

x an object of class "summary.stdGee".
... not used.

Author(s)

Arvid Sjolander

See Also

stdGee

Examples

##See documentation for stdGee
print.summary.stdGlm  
*Prints summary of GLM regression standardization fit*

**Description**

This is a print method for class "summary.stdGlm".

**Usage**

```r
## S3 method for class 'summary.stdGlm'
print(x, ...)
```

**Arguments**

- `x`: an object of class "summary.stdGlm".
- `...`: not used.

**Author(s)**

Arvid Sjolander

**See Also**

- `stdGlm`

**Examples**

```r
## See documentation for stdGlm
```

print.summary.stdParfrailty  
*Prints summary of Frailty standardization fit*

**Description**

This is a print method for class "summary.stdParfrailty".

**Usage**

```r
## S3 method for class 'summary.stdParfrailty'
print(x, ...)
```
stdCoxph

Arguments

- `x`: an object of class "summary.stdParfrailty".
- `...`: not used.

Author(s)

Arvid Sjolander

See Also

stdParfrailty

Examples

```r
# See documentation for stdParfrailty
```

---

**Description**

stdCoxph performs regression standardization in Cox proportional hazards models, at specified values of the exposure, over the sample covariate distribution. Let $T$, $X$, and $Z$ be the survival outcome, the exposure, and a vector of covariates, respectively. stdCoxph uses a fitted Cox proportional hazards model to estimate the standardized survival function $\theta(t, x) = E\{S(t|X = x, Z)\}$, where $t$ is a specific value of $T$, $x$ is a specific value of $X$, and the expectation is over the marginal distribution of $Z$.

**Usage**

```r
stdCoxph(fit, data, X, x, t, clusterid, subsetnew)
```

**Arguments**

- `fit`: an object of class "coxph", as returned by the coxph function in the survival package, but without special terms strata, cluster or tt. Only breslow method for handling ties is allowed. If arguments weights and/or subset are used when fitting the model, then the same weights and subset are used in stdGlm.
- `data`: a data frame containing the variables in the model. This should be the same data frame as was used to fit the model in `fit`.
- `X`: a string containing the name of the exposure variable $X$ in `data`.  

---
stdCoxph

x an optional vector containing the specific values of \( X \) at which to estimate the
standardized survival function. If \( X \) is binary (0/1) or a factor, then \( x \) defaults
to all values of \( X \). If \( X \) is numeric, then \( x \) defaults to the mean of \( X \). If \( x \) is set
to \( NA \), then \( X \) is not altered. This produces an estimate of the marginal survival
function \( S(t) = E\{S(t|X, Z)\} \).

t an optional vector containing the specific values of \( T \) at which to estimate the
standardized survival function. It defaults to all the observed event times in

clusterid an optional string containing the name of a cluster identification variable when
data are clustered.

subsetnew an optional logical statement specifying a subset of observations to be used in
the standardization. This set is assumed to be a subset of the subset (if any) that
was used to fit the regression model.

Details

stdCoxph assumes that a Cox proportional hazards model

\[
\lambda(t|X, Z) = \lambda_0(t) \exp\{h(X, Z; \beta)\}
\]

has been fitted. Breslow’s estimator of the cumulative baseline hazard \( \Lambda_0(t) = \int_0^t \lambda_0(u)du \) is
used together with the partial likelihood estimate of \( \beta \) to obtain estimates of the survival function

\[
\hat{S}(t|X = x, Z) = \exp[-\hat{\Lambda}_0(t) \exp\{h(X = x, Z; \hat{\beta})\}]
\]

For each \( t \) in the \( t \) argument and for each \( x \) in the \( x \) argument, these estimates are averaged across
all subjects (i.e. all observed values of \( Z \)) to produce estimates

\[
\hat{\theta}(t, x) = \frac{1}{n} \sum_{i=1}^n \hat{S}(t|X = x, Z_i)/n,
\]

where \( Z_i \) is the value of \( Z \) for subject \( i, i = 1, \ldots, n \). The variance for \( \hat{\theta}(t, x) \) is obtained by the
sandwich formula.

Value

An object of class "stdCoxph" is a list containing

est a matrix with length(t) rows and length(x) columns, where the element on
row i and column j is equal to \( \hat{\theta}(t[i], x[j]) \).

vcov a list with length(t) elements. Each element is a square matrix with length(x)
rows. In the \( k \):th matrix, the element on row i and column j is the (estimated)
covariance of \( \hat{\theta}(t[k], x[i]) \) and \( \hat{\theta}(t[k], x[j]) \).

Note

Standardized survival functions are sometimes referred to as (direct) adjusted survival functions in
the literature.

stdCoxph does not currently handle time-varying exposures or covariates.
stdCoxph internally loops over all values in the t argument. Therefore, the function will usually be considerably faster if length(t) is small.

The variance calculation performed by stdCoxph does not condition on the observed covariates \( \bar{Z} = (Z_1, \ldots, Z_n) \). To see how this matters, note that

\[
\text{var}\{\hat{\theta}(t, x)\} = E[\text{var}\{\hat{\theta}(t, x)|\bar{Z}\}] + \text{var}[E\{\hat{\theta}(t, x)|\bar{Z}\}].
\]

The usual parameter \( \beta \) in a Cox proportional hazards model does not depend on \( \bar{Z} \). Thus, \( E(\hat{\beta}|\bar{Z}) \) is independent of \( \bar{Z} \) as well (since \( E(\hat{\beta}|\bar{Z}) = \beta \)), so that the term \( \text{var}\{E\{\hat{\beta}|\bar{Z}\}\} \) in the corresponding variance decomposition for \( \text{var}(\hat{\beta}) \) becomes equal to 0. However, \( \theta(t, x) \) depends on \( \bar{Z} \) through the average over the sample distribution for \( Z \), and thus the term \( \text{var}[E\{\hat{\theta}(t, x)|\bar{Z}\}] \) is not 0, unless one conditions on \( \bar{Z} \). The variance calculation by Gail and Byar (1986) ignores this term, and thus effectively conditions on \( \bar{Z} \).

**Author(s)**

Arvid Sjolander

**References**


**Examples**

```r
require(survival)

n <- 1000
Z <- rnorm(n)
X <- rnorm(n, mean=Z)
T <- rexp(n, rate=exp(X+Z+X*Z)) #survival time
C <- rexp(n, rate=exp(X+Z+X*Z)) #censoring time
U <- pmin(T, C) #time at risk
D <- as.numeric(T < C) #event indicator
dd <- data.frame(Z, X, U, D)
fit <- coxph(formula=Surv(U, D)-X+Z+X+Z, data=dd, method="breslow")
fit.std <- stdCoxph(fit=fit, data=dd, X="X", x=seq(-1,1,0.5), t=1:5)
print(summary(fit.std, t=3))
plot(fit.std)
```
stdGee performs regression standardization in linear and log-linear fixed effects models, at specified values of the exposure, over the sample covariate distribution. Let $Y$, $X$, and $Z$ be the outcome, the exposure, and a vector of covariates, respectively. It is assumed that data are clustered with a
cluster indicator $i$. stdGee uses fitted fixed effects model, with cluster-specific intercept $a_i$ (see details), to estimate the standardized mean $\theta(x) = E\{E(Y|X=x,Z)\}$, where $x$ is a specific value of $X$, and the outer expectation is over the marginal distribution of $(a_i, Z)$.

**Usage**

```r
stdGee(fit, data, x, x, clusterid, subsetnew)
```

**Arguments**

- `fit`: an object of class "gee", with argument cond = TRUE, as returned by the gee function in the drgee package. If arguments weights and/or subset are used when fitting the model, then the same weights and subset are used in stdGee.
- `data`: a data frame containing the variables in the model. This should be the same data frame as was used to fit the model in `fit`.
- `x`: a string containing the name of the exposure variable $X$ in `data`.
- `x`: an optional vector containing the specific values of $X$ at which to estimate the standardized mean. If $X$ is binary (0/1) or a factor, then $x$ defaults to all values of $X$. If $X$ is numeric, then $x$ defaults to the mean of $X$. If $x$ is set to NA, then $X$ is not altered. This produces an estimate of the marginal mean $E(Y) = E\{E(Y|X,Z)\}$.
- `clusterid`: an mandatory string containing the name of a cluster identification variable. Must be identical to the clusterid variable used in the gee call.
- `subsetnew`: an optional logical statement specifying a subset of observations to be used in the standardization. This set is assumed to be a subset of the subset (if any) that was used to fit the regression model.

**Details**

stdGee assumes that a fixed effects model

$$\eta\{E(Y|X,Z)\} = a_i + h(X,Z; \beta)$$

has been fitted. The link function $\eta$ is assumed to be the identity link or the log link. The conditional generalized estimating equation (CGGE) estimate of $\beta$ is used to obtain estimates of the cluster-specific means:

$$\hat{a}_i = \frac{\sum_{j=1}^{n_i} r_{ij}}{n_i},$$
where
\[ r_{ij} = Y_{ij} - h(X_{ij}, Z_{ij}; \hat{\beta}) \]
if \( \eta \) is the identity link, and
\[ r_{ij} = Y_{ij} \exp\{-h(X_{ij}, Z_{ij}; \hat{\beta})\} \]
if \( \eta \) is the log link, and \((X_{ij}, Z_{ij})\) is the value of \((X, Z)\) for subject \( j \) in cluster \( i \), \( j = 1, \ldots, n_i \), \( i = 1, \ldots, n \). The CGEE estimate of \( \beta \) and the estimate of \( a_i \) are used to estimate the mean \( E(Y|i, X = x, Z) \):
\[
\hat{E}(Y|i, X = x, Z) = \eta^{-1}\{\hat{a}_i + h(X = x, Z; \hat{\beta})\}.
\]
For each \( x \) in the \( x \) argument, these estimates are averaged across all subjects (i.e. all observed values of \( Z \) and all estimated values of \( a_i \)) to produce estimates
\[
\hat{\theta}(x) = \frac{\sum_{i=1}^n \sum_{j=1}^{n_i} \hat{E}(Y|i, X = x, Z_i)}{N},
\]
where \( N = \sum_{i=1}^n n_i \). The variance for \( \hat{\theta}(x) \) is obtained by the sandwich formula.

**Value**

An object of class "stdGee" is a list containing
- `est` a vector with length equal to \( \text{length}(x) \), where element \( j \) is equal to \( \hat{\theta}(x[j]) \).
- `vcov` a square matrix with \( \text{length}(x) \) rows, where the element on row \( i \) and column \( j \) is the (estimated) covariance of \( \hat{\theta}(x[i]) \) and \( \hat{\theta}(x[j]) \).

**Note**

The variance calculation performed by `stdGee` does not condition on the observed covariates \( \bar{Z} = (Z_{11}, \ldots, Z_{nn_i}) \). To see how this matters, note that
\[
\text{var}\{\hat{\theta}(x)\} = \text{E}[\text{var}\{\hat{\theta}(x)|\bar{Z}\}] + \text{var}[\text{E}\{\hat{\theta}(x)|\bar{Z}\}].
\]
The usual parameter \( \beta \) in a generalized linear model does not depend on \( \bar{Z} \). Thus, \( \text{E}(\hat{\beta}|\bar{Z}) \) is independent of \( \bar{Z} \) as well (since \( \text{E}(\hat{\beta}|\bar{Z}) = \beta \)), so that the term \( \text{var}[\text{E}\{\hat{\theta}(x)|\bar{Z}\}] \) in the corresponding variance decomposition for \( \text{var}(\hat{\beta}) \) becomes equal to 0. However, \( \theta(x) \) depends on \( \bar{Z} \) through the average over the sample distribution for \( Z \), and thus the term \( \text{var}[\text{E}\{\hat{\theta}(x)|\bar{Z}\}] \) is not 0, unless one conditions on \( \bar{Z} \).

**Author(s)**

Arvid Sjolander.

**References**


Examples

```r
require(drgee)

n <- 1000
ni <- 2
id <- rep(1:n, each=ni)
ai <- rep(rnorm(n), each=ni)
Z <- rnorm(n*ni)
X <- rnorm(n*ni, mean=ai+Z)
Y <- rnorm(n*ni, mean=ai+X+Z+0.1*X^2)
dd <- data.frame(id, Z, X, Y)
fit <- gee(formula=Y~X+Z*I(X^2), data=dd, clusterid="id", link="identity",
          cond=TRUE)
fit.std <- stdgee(fit=fit, data=dd, X="X", x=seq(-3,3,0.5), clusterid="id")
print(summary(fit.std, contrast="difference", reference=2))
plot(fit.std)
```

**stdGlm**

*Regression standardization in generalized linear models*

**Description**

stdGlm performs regression standardization in generalized linear models, at specified values of the exposure, over the sample covariate distribution. Let $Y$, $X$, and $Z$ be the outcome, the exposure, and a vector of covariates, respectively. stdGlm uses a fitted generalized linear model to estimate the standardized mean $\theta(x) = E\{E(Y|X = x, Z)\}$, where $x$ is a specific value of $X$, and the outer expectation is over the marginal distribution of $Z$.

**Usage**

`stdGlm(fit, data, X, x, clusterid, case.control = FALSE, subsetnew)`

**Arguments**

- `fit` an object of class "glm", as returned by the glm function in the stats package. If arguments weights and/or subset are used when fitting the model, then the same weights and subset are used in stdGlm.
- `data` a data frame containing the variables in the model. This should be the same data frame as was used to fit the model in `fit`.
- `X` a string containing the name of the exposure variable $X$ in `data`.
- `x` an optional vector containing the specific values of $X$ at which to estimate the standardized mean. If $X$ is binary (0/1) or a factor, then `x` defaults to all values of $X$. If $X$ is numeric, then `x` defaults to the mean of $X$. If `x` is set to NA, then $X$ is not altered. This produces an estimate of the marginal mean $E(Y) = E\{E(Y|X, Z)\}$. 

clusterid an optional string containing the name of a cluster identification variable when data are clustered.
case.control logical. Do data come from a case-control study? Defaults to FALSE.
subsetnew an optional logical statement specifying a subset of observations to be used in the standardization. This set is assumed to be a subset of the subset (if any) that was used to fit the regression model.

Details
stdGlm assumes that a generalized linear model
\[ \eta \{ E(Y|X,Z) \} = h(X,Z; \beta) \]
has been fitted. The maximum likelihood estimate of \( \beta \) is used to obtain estimates of the mean \( E(Y|X=x,Z) \):
\[ \hat{E}(Y|X=x,Z) = \eta^{-1}\{ h(X=x,Z; \hat{\beta}) \} \].
For each \( x \) in the \( x \) argument, these estimates are averaged across all subjects (i.e. all observed values of \( Z \)) to produce estimates
\[ \hat{\theta}(x) = \frac{1}{n} \sum_{i=1}^{n} \hat{E}(Y|X=x,Z_i) \]
where \( Z_i \) is the value of \( Z \) for subject \( i, i = 1, \ldots, n \). The variance for \( \hat{\theta}(x) \) is obtained by the sandwich formula.

Value
An object of class "stdGlm" is a list containing
est a vector with length equal to length(\( x \)), where element \( j \) is equal to \( \hat{\theta}(x[j]) \).
vcov a square matrix with length(\( x \)) rows, where the element on row \( i \) and column \( j \) is the (estimated) covariance of \( \hat{\theta}(x[i]) \) and \( \hat{\theta}(x[j]) \).

Note
The variance calculation performed by stdGlm does not condition on the observed covariates \( \bar{Z} = (Z_1, \ldots, Z_n) \). To see how this matters, note that
\[ \text{var}\{\hat{\theta}(x)\} = \text{E}[\text{var}\{\hat{\theta}(x)|\bar{Z}\}] + \text{var}[\text{E}\{\hat{\theta}(x)|\bar{Z}\}] \].
The usual parameter \( \beta \) in a generalized linear model does not depend on \( \bar{Z} \). Thus, \( \text{E}(\hat{\beta}|\bar{Z}) \) is independent of \( \bar{Z} \) as well (since \( \text{E}(\hat{\beta}|\bar{Z}) = \beta \)), so that the term \( \text{var}[\text{E}\{\hat{\beta}|\bar{Z}\}] \) in the corresponding variance decomposition for \( \text{var}(\hat{\beta}) \) becomes equal to 0. However, \( \hat{\theta}(x) \) depends on \( \bar{Z} \) through the average over the sample distribution for \( Z \), and thus the term \( \text{var}[\text{E}\{\hat{\theta}(x)|\bar{Z}\}] \) is not 0, unless one conditions on \( \bar{Z} \).

Author(s)
Arvid Sjolander.
References


Examples

```r
## Example 1: continuous outcome
n <- 1000
Z <- rnorm(n)
X <- rnorm(n, mean=Z)
Y <- rnorm(n, mean=X+Z+0.1*X^2)
dd <- data.frame(Z, X, Y)
fit <- glm(formula=Y~X+Z, data=dd)
fit.std <- stdglm(fit, data=dd, X="X", x=seq(-3,3,0.5))
print(summary(fit.std))
plot(fit.std)
```

```r
## Example 2: binary outcome
n <- 1000
Z <- rnorm(n)
X <- rnorm(n, mean=Z)
Y <- rbinom(n, 1, prob=(1+exp(X+Z))/(1+1))
dd <- data.frame(Z, X, Y)
fit <- glm(formula=Y~X+Z, data=dd, family="binomial")
fit.std <- stdglm(fit, data=dd, X="X", x=seq(-3,3,0.5))
print(summary(fit.std))
plot(fit.std)
```

**stdParfrailty**

*Regression standardization in shared frailty gamma-Weibull models*

**Description**

stdParfrailty performs regression standardization in shared frailty gamma-Weibull models, at specified values of the exposure, over the sample covariate distribution. Let $T$, $X$, and $Z$ be the survival outcome, the exposure, and a vector of covariates, respectively. stdParfrailty uses a fitted Cox proportional hazards model to estimate the standardized survival function $\theta(t, x) = E[S(t|X = x, Z)]$, where $t$ is a specific value of $T$, $x$ is a specific value of $X$, and the expectation is over the marginal distribution of $Z$.

**Usage**

```r
stdParfrailty(fit, data, X, x, t, clusterid, subsetnew)
```
stdParfrailty

Arguments

fit an object of class "parfrailty", as returned by the parfrailty function in the stdReg package.
data a data frame containing the variables in the model. This should be the same data frame as was used to fit the model in fit.
x a string containing the name of the exposure variable X in data.
x an optional vector containing the specific values of X at which to estimate the standardized survival function. If X is binary (0/1) or a factor, then x defaults to all values of X. If X is numeric, then x defaults to the mean of X. If x is set to NA, then X is not altered. This produces an estimate of the marginal survival function \( S(t) = E\{S(t|X,Z)\} \).
t an optional vector containing the specific values of T at which to estimate the standardized survival function. It defaults to all the observed event times in data.
clusterid a string containing the name of the cluster identification variable.
subsetnew an optional logical statement specifying a subset of observations to be used in the standardization. This set is assumed to be a subset of the subset (if any) that was used to fit the regression model.

Details

stdParfrailty assumes that a shared frailty gamma-Weibull model

\[
\lambda(t_{ij}|X_{ij}, Z_{ij}) = \lambda(t_{ij}; \alpha, \eta) U_i \exp\{h(X_{ij}, Z_{ij}; \beta)\}
\]

has been fitted, with parametrization as described in the help section for parfrailty. Integrating out the gamma frailty gives the survival function

\[
S(t|X, Z) = \left[1 + \phi \Lambda_0(t; \alpha, \eta) \exp\{h(X, Z; \beta)\}\right]^{-1/\phi},
\]

where \( \Lambda_0(t; \alpha, \eta) \) is the cumulative baseline hazard

\[
(t/\alpha)^\eta.
\]

The ML estimates of \( (\alpha, \eta, \phi, \beta) \) are used to obtain estimates of the survival function \( S(t|X = x, Z) \):

\[
\hat{S}(t|x, Z) = \left[1 + \hat{\phi} \Lambda_0(t; \hat{\alpha}, \hat{\eta}) \exp\{h(X, Z; \hat{\beta})\}\right]^{-1/\hat{\phi}}.
\]

For each t in the t argument and for each x in the x argument, these estimates are averaged across all subjects (i.e. all observed values of Z) to produce estimates

\[
\hat{\theta}(t, x) = \frac{1}{n} \sum_{i=1}^{n} \hat{S}(t|x, Z_i) / n.
\]

The variance for \( \hat{\theta}(t, x) \) is obtained by the sandwich formula.
An object of class "stdParfrailty" is a list containing

- `est`: a matrix with `length(t)` rows and `length(x)` columns, where the element on row `i` and column `j` is equal to \( \hat{\theta}(t[i], x[j]) \).

- `vcov`: a list with `length(t)` elements. Each element is a square matrix with `length(x)` rows. In the `k`th matrix, the element on row `i` and column `j` is the (estimated) covariance of \( \hat{\theta}(t[k], x[i]) \) and \( \hat{\theta}(t[k], x[j]) \).

**Note**

Standardized survival functions are sometimes referred to as (direct) adjusted survival functions in the literature.

`stdParfrailty` does not currently handle time-varying exposures or covariates.

`stdParfrailty` internally loops over all values in the `t` argument. Therefore, the function will usually be considerably faster if `length(t)` is small.

The variance calculation performed by `stdParfrailty` does not condition on the observed covariates \( \bar{Z} = (Z_1, ..., Z_n) \). To see how this matters, note that

\[
\text{var}\{\hat{\theta}(t, x)\} = E[\text{var}\{\hat{\theta}(t, x)|\bar{Z}\}] + \text{var}[E\{\hat{\theta}(t, x)|\bar{Z}\}].
\]

The usual parameter \( \beta \) in a Cox proportional hazards model does not depend on \( \bar{Z} \). Thus, \( E(\hat{\beta}|\bar{Z}) \) is independent of \( \bar{Z} \) as well (since \( E(\hat{\beta}|\bar{Z}) = \beta \)), so that the term \( \text{var}[E\{\hat{\theta}(t, x)|\bar{Z}\}] \) in the corresponding variance decomposition for \( \text{var}(\hat{\beta}) \) becomes equal to 0. However, \( \hat{\theta}(t, x) \) depends on \( \bar{Z} \) through the average over the sample distribution for \( Z \), and thus the term \( \text{var}[E\{\hat{\theta}(t, x)|\bar{Z}\}] \) is not 0, unless one conditions on \( \bar{Z} \). The variance calculation by Gail and Byar (1986) ignores this term, and thus effectively conditions on \( \bar{Z} \).

**Author(s)**

Arvid Sjolander

**References**


Examples

```
require(survival)

# simulate data
n <- 1000
m <- 3
alpha <- 1.5
eta <- 1
phi <- 0.5
beta <- 1
id <- rep(1:n, each=m)
U <- rep(rgamma(n, shape=1/phi, scale=phi), each=m)
X <- rnorm(n*m)

# reparametrize scale as in rweibull function
weibull.scale <- alpha/(U*exp(beta*X))^(1/eta)
T <- rweibull(n*m, shape=eta, scale=weibull.scale)

# right censoring
C <- runif(n*m, 0, 10)
D <- as.numeric(T<C)
T <- pmin(T, C)

# strong left truncation
L <- runif(n*m, 0, 2)
incl <- T>L
incl <- ave(x=incl, id, FUN=sum)==m
dd <- data.frame(L, T, D, X, id)
dd <- dd[incl, ]

fit <- parfrailty(formula=Surv(L, T, D)-X, data=dd, clusterid="id")
fit.std <- stdParfrailty(fit, data=dd, X="X", x=seq(-1,1,0.5), t=1:5, clusterid="id")
print(summary(fit.std, t=3))
plot(fit.std)
```
Arguments

- **object**
  - an object of class "parfrailty".
- **CI.type**
  - string, indicating the type of confidence intervals. Either "plain", which gives untransformed intervals, or "log", which gives log-transformed intervals.
- **CI.level**
  - desired coverage probability of confidence intervals, in decimal form.
- **digits**
  - the number of significant digits to use when printing.
- **...**
  - not used.

Author(s)

Arvid Sjolander and Elisabeth Dahlqwist.

See Also

- parfrailty

Examples

```r
# See documentation for frailty
```

---

**summary.stdCoxph**

**Summarizes Cox regression standardization fit**

Description

This is a summary method for class "stdCoxph".

Usage

```r
## S3 method for class 'stdCoxph'
summary(object, t, CI.type = "plain", CI.level = 0.95,
         transform = NULL, contrast = NULL, reference = NULL, ...)
```

Arguments

- **object**
  - an object of class "stdCoxph".
- **t**
  - numeric, indicating the times at which to summarize. It defaults to the specified value(s) of the argument t in the stdCox function.
- **CI.type**
  - string, indicating the type of confidence intervals. Either "plain", which gives untransformed intervals, or "log", which gives log-transformed intervals.
- **CI.level**
  - desired coverage probability of confidence intervals, on decimal form.
transform a string. If set to "log", "logit", or "odds", the standardized survival function \( \theta(t, x) \) is transformed into \( \psi(t, x) = \log(\theta(t, x)/\{1 - \theta(t, x)\}) \), or \( \psi(t, x) = \theta(t, x)/\{1 - \theta(t, x)\} \), respectively. If left unspecified, \( \psi(t, x) = \theta(t, x) \).

contrast a string. If set to "difference" or "ratio", then \( \psi(t, x) - \psi(t, x_0) \) or \( \psi(t, x)/\psi(t, x_0) \) are constructed, where \( x_0 \) is a reference level specified by the reference argument.

reference must be specified if contrast is specified.

... not used.

Author(s)
Arvid Sjolander

See Also
stdCoxph

Examples

##See documentation for stdCoxph

summary.stdGee  
\[ \text{Summarizes GEE regression standardization fit} \]

Description
This is a summary method for class "stdGee".

Usage

### S3 method for class 'stdGee'

\[ \text{summary(object, CI.type = "plain", CI.level = 0.95, transform = NULL, contrast = NULL, reference = NULL, ...)} \]

Arguments

- **object** an object of class "stdGee".
- **CI.type** string, indicating the type of confidence intervals. Either "plain", which gives untransformed intervals, or "log", which gives log-transformed intervals.
- **CI.level** desired coverage probability of confidence intervals, on decimal form.
- **transform** a string. If set to "log", "logit", or "odds", the standardized mean \( \theta(x) \) is transformed into \( \psi(x) = \log(\theta(x)), \psi(x) = \log(\theta(x)/\{1 - \theta(x)\}) \), or \( \psi(x) = \theta(x)/\{1 - \theta(x)\} \), respectively. If left unspecified, \( \psi(x) = \theta(x) \).
contrast is a string. If set to "difference" or "ratio", then \( \psi(x) - \psi(x_0) \) or \( \psi(x)/\psi(x_0) \) are constructed, where \( x_0 \) is a reference level specified by the reference argument.

reference must be specified if contrast is specified.

Author(s)
Arvid Sjolander

See Also
stdGee

Examples

```r
## See documentation for stdGee
```

---

**summary.stdParfrailty**  
**Summarizes Frailty standardization fit**

**Description**

This is a summary method for class "stdParfrailty".

**Usage**

```r
## S3 method for class 'stdParfrailty'
summary(object, t, CI.type = "plain", CI.level = 0.95,
        transform = NULL, contrast = NULL, reference = NULL, ...)
```

**Arguments**

- **object**: an object of class "stdParfrailty".
- **t**: numeric, indicating the times at which to summarize. It defaults to the specified value(s) of the argument t in the stdCox function.
- **CI.type**: string, indicating the type of confidence intervals. Either "plain", which gives untransformed intervals, or "log", which gives log-transformed intervals.
- **CI.level**: desired coverage probability of confidence intervals, on decimal form.
- **transform**: a string. If set to "log", "logit", or "odds", the standardized survival function \( \hat{\theta}(t,x) \) is transformed into \( \psi(t,x) = \log\{\theta(t,x)\} \), \( \psi(t,x) = \log[\theta(t,x)/(1 - \theta(t,x))] \), or \( \psi(t,x) = \theta(t,x)/(1 - \theta(t,x)) \), respectively. If left unspecified, \( \psi(t,x) = \hat{\theta}(t,x) \).
contrast

A string. If set to "difference" or "ratio", then $\psi(t, x) - \psi(t, x_0)$ or $\psi(t, x)/\psi(t, x_0)$ are constructed, where $x_0$ is a reference level specified by the reference argument.

reference

Must be specified if contrast is specified.

... not used.

Author(s)

Arvid Sjolander

See Also

stdParfrailty

Examples

# See documentation for stdParfrailty
Index

*Topic package
  stdReg-package, 2

confint.stdCoxph, 4
confint.stdGee, 5
confint.stdGlm, 6
confint.stdParfrailty, 7
parfrailty, 8, 14, 28
plot.stdCoxph, 10
plot.stdGee, 11
plot.stdGlm, 12
plot.stdParfrailty, 13
print.summary.parfrailty, 14
print.summary.stdCoxph, 14
print.summary.stdGee, 15
print.summary.stdGlm, 16
print.summary.stdParfrailty, 16
stdCoxph, 10, 15, 17, 29
stdGee, 11, 15, 20, 30
stdGlm, 12, 16, 22
stdParfrailty, 13, 17, 24, 31
stdReg(stdReg-package), 2
stdReg-package, 2
summary.parfrailty, 27
summary.stdCoxph, 28
summary.stdGee, 29
summary.stdParfrailty, 30