Package 'stdReg2'

September 13, 2024

Type Package

Title Regression Standardization for Causal Inference

Version 1.0.1

URL `https://sachsmc.github.io/stdReg2/`

BugReports `https://github.com/sachsmc/stdReg2/issues/`

Date 2024-09-10

Description Contains more modern tools for causal inference using regression standardization. Four general classes of models are implemented; generalized linear models, conditional generalized estimating equation models, Cox proportional hazards models, and shared frailty gamma-Weibull models. Methodological details are described in Sjölander, A. (2016) <doi:10.1007/s10654-016-0157-3>. Also includes functionality for doubly robust estimation for generalized linear models in some special cases, and the ability to implement custom models.

License AGPL (>= 3)

Encoding UTF-8

Imports data.table, drgee, generics, survival

Suggests causaldata, AF, knitr, nnet, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

RoxygenNote 7.3.2

Depends R (>= 2.10)

NeedsCompilation no

Author Michael C Sachs [aut, cre],
Arvid Sjölander [aut],
Erin E Gabriel [aut],
Johan Sebastian Ohlendorff [aut],
Adam Brand [aut]

Maintainer Michael C Sachs <`sachsmc@gmail.com`>

Repository CRAN

Date/Publication 2024-09-13 17:50:02 UTC

Contents

stdReg2-package 2
parfrailty 3
plot.std_glm 5
plot.std_surv 6
print.std_surv 7
print.summary.parfrailty 7
sandwich 8
standardize 9
standardize_coxph 11
standardize_gee 15
standardize_glm 18
standardize_glm_dr 21
standardize_level 23
standardize_parfrailty 26
summary.parfrailty 29
tidy.std_glm 31
tidy.std_surv 32

Index **33**

`stdReg2-package` *stdReg2: Regression Standardization for Causal Inference*

Description

Contains more modern tools for causal inference using regression standardization. Four general classes of models are implemented; generalized linear models, conditional generalized estimating equation models, Cox proportional hazards models, and shared frailty gamma-Weibull models. Methodological details are described in Sjölander, A. (2016) doi:10.1007/s1065401601573. Also includes functionality for doubly robust estimation for generalized linear models in some special cases, and the ability to implement custom models.

Author(s)

Maintainer: Michael C Sachs <`sachsmc@gmail.com`>

Authors:

- Arvid Sjölander
- Erin E Gabriel
- Johan Sebastian Ohlendorff
- Adam Brand

See Also

Useful links:

- `https://sachsmc.github.io/stdReg2/`
- Report bugs at `https://github.com/sachsmc/stdReg2/issues/`

`parfrailty` *Fits shared frailty gamma-Weibull models*

Description

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

Usage

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

Arguments

`formula`	an object of class `"formula"`, in the same format as accepted by the coxph function.
`data`	a data frame containing the variables in the model.
`clusterid`	a 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

$$\eta t^{\eta-1}\alpha^{-\eta},$$

where η is the shape parameter and α 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 = ϕ. $h(X;\beta)$ is the regression function as specified by the `formula` argument, parameterized by a vector β. 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"` which is a list containing:

`est`	the Maximum Likelihood (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 Sjölander and Elisabeth Dahlqwist.

References

Dahlqwist E., Pawitan Y., Sjölander A. (2019). Regression standardization and attributable fraction estimation with between-within frailty models for clustered survival data. *Statistical Methods in Medical Research* **28**(2), 462-485.

Van den Berg G.J., Drepper B. (2016). Inference for shared frailty survival models with left-truncated data. *Econometric Reviews*, 35(6), 1075-1098.

Examples

```
require(survival)

# simulate data
set.seed(5)
n <- 200
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)
# reparameterize 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")
print(fit)
```

`plot.std_glm` *Plots regression standardization fit*

Description

This is a `plot` method for class `"std_glm"`.

Usage

```
## S3 method for class 'std_glm'
plot(
  x,
  plot_ci = TRUE,
  ci_type = "plain",
  ci_level = 0.95,
  transform = NULL,
  contrast = NULL,
  reference = NULL,
  summary_fun = "summary_std_glm",
  ...
)
```

Arguments

`x`	An object of class `"std_glm"`.
`plot_ci`	if `TRUE`, add the confidence intervals to the plot.
`ci_type`	A string, indicating the type of confidence intervals. Either "plain", which gives untransformed intervals, or "log", which gives log-transformed intervals.
`ci_level`	Coverage probability of confidence intervals.
`transform`	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`	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. If not `NULL`, a doubly robust estimator of the standardized estimator is used.
`reference`	If `contrast` is specified, the desired reference level.
`summary_fun`	For internal use only. Do not change.
`...`	Unused.

Value

None. Creates a plot as a side effect

Examples

```
# see standardize_glm
```

`plot.std_surv` *Plots regression standardization fit*

Description

This is a `plot` method for class `"std_surv"`.

Usage

```
## S3 method for class 'std_surv'
plot(
  x,
  plot_ci = TRUE,
  ci_type = "plain",
  ci_level = 0.95,
  transform = NULL,
  contrast = NULL,
  reference = NULL,
  legendpos = "bottomleft",
  summary_fun = "summary_std_coxph",
  ...
)
```

Arguments

`x`	An object of class `"std_surv"`.
`plot_ci`	if `TRUE`, add the confidence intervals to the plot.
`ci_type`	A string, indicating the type of confidence intervals. Either "plain", which gives untransformed intervals, or "log", which gives log-transformed intervals.
`ci_level`	Coverage probability of confidence intervals.
`transform`	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`	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. If not `NULL`, a doubly robust estimator of the standardized estimator is used.
`reference`	If `contrast` is specified, the desired reference level.
`legendpos`	position of the legend; see legend.
`summary_fun`	For internal use only. Do not change.
`...`	Unused.

Value

None. Creates a plot as a side effect

`print.std_surv` *Prints summary of regression standardization fit*

Description

Prints summary of regression standardization fit

Usage

```
## S3 method for class 'std_surv'
print(x, ...)

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

## S3 method for class 'std_custom'
print(x, ...)
```

Arguments

`x`	an object of class "`std_glm`", "`std_surv`" or "`std_custom`".
`...`	unused

Value

The object being printed, invisibly.

`print.summary.parfrailty` *Print method for parametric frailty fits*

Description

Print method for parametric frailty fits

Usage

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

Arguments

`x`	An object of class "parfrailty"
`digits`	Number of digits to print
`...`	Not used

Value

The object being printed, invisibly

`sandwich` *Compute the sandwich variance components from a model fit*

Description

Compute the sandwich variance components from a model fit

Usage

```
sandwich(fit, data, weights, t, fit.detail)
```

Arguments

`fit`	A fitted model object of class glm, coxph, ah, or survfit
`data`	The data used to fit the model
`weights`	Optional weights
`t`	Optional fixed time point for survival objects
`fit.detail`	For Cox models, the result of running coxph.detail on the model fit

Value

A list consisting of the Fisher information matrix (I) and the Score equations (U)

`standardize` *Get standardized estimates using the g-formula with a custom model*

Description

Get standardized estimates using the g-formula with a custom model

Usage

```
standardize(
  fitter,
  arguments,
  predict_fun,
  data,
  values,
  B = NULL,
  ci_level = 0.95,
  contrasts = NULL,
  reference = NULL,
  seed = NULL,
  times = NULL,
  transforms = NULL,
  progressbar = TRUE
)
```

Arguments

`fitter`	The function to call to fit the data.
`arguments`	The arguments to be used in the fitter function as a `list`.
`predict_fun`	The function used to predict the means/probabilities for a new data set on the response level. For survival data, this should be a matrix where each column is the time, and each row the data.
`data`	The data.
`values`	A named list or data.frame specifying the variables and values at which marginal means of the outcome will be estimated.
`B`	Number of nonparametric bootstrap resamples. Default is `NULL` (no bootstrap).
`ci_level`	Coverage probability of confidence intervals.
`contrasts`	A vector of contrasts in the following format: 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. Has to be `NULL` if no references are specified.
`reference`	A vector of reference levels in the following format: If `contrasts` is not `NULL`, the desired reference level(s). This must be a vector or list the same length as `contrasts`, and if not named, it is assumed that the order is as specified in contrasts.

`seed`	The seed to use with the nonparametric bootstrap.
`times`	For use with survival data. Set to `NULL` otherwise.
`transforms`	A vector of transforms in the following format: 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 the vector is `NULL`, then $\psi(x) = \theta(x)$.
`progressbar`	Logical, if TRUE will print bootstrapping progress to the console

Details

Let Y, X, and Z be the outcome, the exposure, and a vector of covariates, respectively. `standardize` uses a 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. With survival data, $Y = I(T > t)$, and a vector of different time points `times` (t) can be given, where T is the uncensored survival time.

Value

An object of class `std_custom`. This is a list with components estimates and fit for the outcome model.

References

Rothman K.J., Greenland S., Lash T.L. (2008). *Modern Epidemiology*, 3rd edition. Lippincott, Williams & Wilkins.

Sjölander A. (2016). Regression standardization with the R-package stdReg. *European Journal of Epidemiology* **31**(6), 563-574.

Sjölander A. (2016). Estimation of causal effect measures with the R-package stdReg. *European Journal of Epidemiology* **33**(9), 847-858.

Examples

```
set.seed(6)
n <- 100
Z <- rnorm(n)
X <- rnorm(n, mean = Z)
Y <- rbinom(n, 1, prob = (1 + exp(X + Z))^(-1))
dd <- data.frame(Z, X, Y)
prob_predict.glm <- function(...) predict.glm(..., type = "response")

x <- standardize(
  fitter = "glm",
  arguments = list(
    formula = Y ~ X * Z,
    family = "binomial"
  ),
  predict_fun = prob_predict.glm,
  data = dd,
  values = list(X = seq(-1, 1, 0.1)),
```

```
  B = 100,
  reference = 0,
  contrasts = "difference"
)
x

require(survival)
prob_predict.coxph <- function(object, newdata, times) {
  fit.detail <- suppressWarnings(basehaz(object))
  cum.haz <- fit.detail$hazard[sapply(times, function(x) max(which(fit.detail$time <= x)))]
  predX <- predict(object = object, newdata = newdata, type = "risk")
  res <- matrix(NA, ncol = length(times), nrow = length(predX))
  for (ti in seq_len(length(times))) {
    res[, ti] <- exp(-predX * cum.haz[ti])
  }
  res
}
set.seed(68)
n <- 500
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)
x <- standardize(
fitter = "coxph",
  arguments = list(
    formula = Surv(U, D) ~ X + Z + X * Z,
    method = "breslow",
    x = TRUE,
    y = TRUE
  ),
  predict_fun = prob_predict.coxph,
  data = dd,
  times = 1:5,
  values = list(X = c(-1, 0, 1)),
  B = 100,
  reference = 0,
  contrasts = "difference"
)
x
```

`standardize_coxph` *Regression standardization in Cox proportional hazards models*

Description

`standardize_coxph` 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. `standardize_coxph` fits a Cox proportional hazards model and the Breslow estimator of the baseline hazard in order to estimate the standardized survival function $\theta(t,x) = E\{S(t|X = x,Z)\}$ when `measure = "survival"` or the standardized restricted mean survival up to time t $\theta(t,x) = E\{\int_0^t S(u|X = x,Z)du\}$ when `measure = "rmean"`, 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

```
standardize_coxph(
  formula,
  data,
  values,
  times,
  measure = c("survival", "rmean"),
  clusterid,
  ci_level = 0.95,
  ci_type = "plain",
  contrasts = NULL,
  family = "gaussian",
  reference = NULL,
  transforms = NULL
)
```

Arguments

`formula`	The formula which is used to fit the model for the outcome.
`data`	The data.
`values`	A named list or data.frame specifying the variables and values at which marginal means of the outcome will be estimated.
`times`	A vector containing the specific values of T at which to estimate the standardized survival function.
`measure`	Either "survival" to estimate the survival function at times or "rmean" for the restricted mean survival up to the largest of times.
`clusterid`	An optional string containing the name of a cluster identification variable when data are clustered.
`ci_level`	Coverage probability of confidence intervals.
`ci_type`	A string, indicating the type of confidence intervals. Either "plain", which gives untransformed intervals, or "log", which gives log-transformed intervals.
`contrasts`	A vector of contrasts in the following format: 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. Has to be `NULL` if no references are specified.
`family`	The family argument which is used to fit the glm model for the outcome.

`reference`	A vector of reference levels in the following format: If `contrasts` is not `NULL`, the desired reference level(s). This must be a vector or list the same length as `contrasts`, and if not named, it is assumed that the order is as specified in contrasts.
`transforms`	A vector of transforms in the following format: 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 the vector is `NULL`, then $\psi(x) = \theta(x)$.

Details

`standardize_coxph` fits the Cox proportional hazards model

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

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 β to obtain estimates of the survival function $S(t|X = x, Z)$ if `measure = "survival"`:

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

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

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

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

If `measure = "rmean"`, then $\Lambda_0(t) = \int_0^t \lambda_0(u)du$ is used together with the partial likelihood estimate of β to obtain estimates of the restricted mean survival up to time t: $\int_0^t S(u|X = x, Z)du$ for each element of `times`. The estimation and inference is done using the method described in Chen and Tsiatis 2001. Currently, we can only estimate the difference in RMST for a single binary exposure. Two separate Cox models are fit for each level of the exposure, which is expected to be coded as 0/1.

Value

An object of class `std_surv`. This is basically a list with components estimates and covariance estimates in `res` Results for transformations, contrasts, references are stored in `res_contrasts`. The output contains estimates for contrasts and confidence intervals for all combinations of transforms and reference levels. Obtain numeric results in a data frame with the tidy function.

Note

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

`standardize_coxph/standardize_parfrailty` does not currently handle time-varying exposures or covariates.

`standardize_coxph/standardize_parfrailty` 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 `standardize_coxph` does not condition on the observed covariates $\bar{Z} = (Z_1, ..., Z_n)$. To see how this matters, note that

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

The usual parameter β 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 $var[E\{\hat{\beta}|\bar{Z}\}]$ in the corresponding variance decomposition for $var(\hat{\beta})$ 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 $var[E\{\hat{\theta}(t, x)|\bar{Z}\}]$ 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 Sjölander, Adam Brand, Michael Sachs

References

Chang I.M., Gelman G., Pagano M. (1982). Corrected group prognostic curves and summary statistics. *Journal of Chronic Diseases* **35**, 669-674.

Gail M.H. and Byar D.P. (1986). Variance calculations for direct adjusted survival curves, with applications to testing for no treatment effect. *Biometrical Journal* **28**(5), 587-599.

Makuch R.W. (1982). Adjusted survival curve estimation using covariates. *Journal of Chronic Diseases* **35**, 437-443.

Sjölander A. (2016). Regression standardization with the R-package stdReg. *European Journal of Epidemiology* **31**(6), 563-574.

Sjölander A. (2018). Estimation of causal effect measures with the R-package stdReg. *European Journal of Epidemiology* **33**(9), 847-858.

Chen, P. Y., Tsiatis, A. A. (2001). Causal inference on the difference of the restricted mean lifetime between two groups. *Biometrics*, **57**(4), 1030-1038.

Examples

```
require(survival)
set.seed(7)
n <- 300
Z <- rnorm(n)
Zbin <- rbinom(n, 1, .3)
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
fact <- factor(sample(letters[1:3], n, replace = TRUE))
U <- pmin(T, C) # time at risk
D <- as.numeric(T < C) # event indicator
dd <- data.frame(Z, Zbin, X, U, D, fact)
```

```
fit.std.surv <- standardize_coxph(
  formula = Surv(U, D) ~ X + Z + X * Z,
  data = dd,
  values = list(X = seq(-1, 1, 0.5)),
  times = 1:5
)
print(fit.std.surv)
plot(fit.std.surv)
tidy(fit.std.surv)

fit.std <- standardize_coxph(
  formula = Surv(U, D) ~ X + Zbin + X * Zbin + fact,
  data = dd,
  values = list(Zbin = 0:1),
  times = 1.5,
  measure = "rmean",
  contrast = "difference",
  reference = 0
)
print(fit.std)
tidy(fit.std)
```

`standardize_gee` *Regression standardization in conditional generalized estimating equations*

Description

`standardize_gee` 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. `standardize_gee` uses fitted fixed effects model, with cluster-specific intercept a_i (see `details`), to estimate the standardized mean $\theta(x) = E\{E(Y|i, X = x, Z)\}$, where x is a specific value of X, and the outer expectation is over the marginal distribution of (a_i, Z).

Usage

```
standardize_gee(
  formula,
  link = "identity",
  data,
  values,
  clusterid,
  case_control = FALSE,
  ci_level = 0.95,
  ci_type = "plain",
  contrasts = NULL,
```

```
  family = "gaussian",
  reference = NULL,
  transforms = NULL
)
```

Arguments

`formula`	A formula to be used with `"gee"` in the **drgee** package.
`link`	The link function to be used with `"gee"`.
`data`	The data.
`values`	A named list or data.frame specifying the variables and values at which marginal means of the outcome will be estimated.
`clusterid`	An optional string containing the name of a cluster identification variable when data are clustered.
`case_control`	Whether the data comes from a case-control study.
`ci_level`	Coverage probability of confidence intervals.
`ci_type`	A string, indicating the type of confidence intervals. Either "plain", which gives untransformed intervals, or "log", which gives log-transformed intervals.
`contrasts`	A vector of contrasts in the following format: 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. Has to be `NULL` if no references are specified.
`family`	The family argument which is used to fit the glm model for the outcome.
`reference`	A vector of reference levels in the following format: If `contrasts` is not `NULL`, the desired reference level(s). This must be a vector or list the same length as `contrasts`, and if not named, it is assumed that the order is as specified in contrasts.
`transforms`	A vector of transforms in the following format: 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 the vector is `NULL`, then $\psi(x) = \theta(x)$.

Details

`standardize_gee` assumes that a fixed effects model

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

has been fitted. The link function η is assumed to be the identity link or the log link. The conditional generalized estimating equation (CGEE) estimate of β is used to obtain estimates of the cluster-specific means:

$$\hat{a}_i = \sum_{j=1}^{n_i} r_{ij}/n_i,$$

where

$$r_{ij} = Y_{ij} - h(X_{ij}, Z_{ij}; \hat{\beta})$$

if η is the identity link, and

$$r_{ij} = Y_{ij} \exp\{-h(X_{ij}, Z_{ij}; \hat{\beta})\}$$

if η is the log link, and (X_{ij}, Z_{ij}) is the value of (X, Z) for subject j in cluster i, $j = 1, ..., n_i$, $i = 1, ..., n$. The CGEE estimate of β 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

$$\hat{\theta}(x) = \sum_{i=1}^{n} \sum_{j=1}^{n_i} \hat{E}(Y|i, X = x, Z_i)/N,$$

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 `std_glm`. This is basically a list with components estimates and covariance estimates in `res`. Results for transformations, contrasts, references are stored in `res_contrasts`. Obtain numeric results in a data frame with the tidy function.

Note

The variance calculation performed by `standardize_gee` does not condition on the observed covariates $\bar{Z} = (Z_{11}, ..., Z_{nn_i})$. To see how this matters, note that

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

The usual parameter β in a generalized linear 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 $var[E\{\hat{\beta}|\bar{Z}\}]$ in the corresponding variance decomposition for $var(\hat{\beta})$ becomes equal to 0. However, $\theta(x)$ depends on $\bar{Z}$ through the average over the sample distribution for Z, and thus the term $var[E\{\hat{\theta}(x)|\bar{Z}\}]$ is not 0, unless one conditions on $\bar{Z}$.

Author(s)

Arvid Sjölander.

References

Goetgeluk S. and Vansteelandt S. (2008). Conditional generalized estimating equations for the analysis of clustered and longitudinal data. *Biometrics* **64**(3), 772-780.

Martin R.S. (2017). Estimation of average marginal effects in multiplicative unobserved effects panel models. *Economics Letters* **160**, 16-19.

Sjölander A. (2019). Estimation of marginal causal effects in the presence of confounding by cluster. *Biostatistics* doi: 10.1093/biostatistics/kxz054

Examples

```
require(drgee)

set.seed(4)
n <- 300
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.std <- standardize_gee(
  formula = Y ~ X + Z + I(X^2),
  link = "identity",
  data = dd,
  values = list(X = seq(-3, 3, 0.5)),
  clusterid = "id"
)
print(fit.std)
plot(fit.std)
```

`standardize_glm` *Get regression standardized estimates from a glm*

Description

Get regression standardized estimates from a glm

Usage

```
standardize_glm(
  formula,
  data,
  values,
  clusterid,
  matched_density_cases,
  matched_density_controls,
  matching_variable,
  p_population,
  case_control = FALSE,
  ci_level = 0.95,
  ci_type = "plain",
  contrasts = NULL,
  family = "gaussian",
  reference = NULL,
  transforms = NULL
)
```

Arguments

`formula`	The formula which is used to fit the model for the outcome.
`data`	The data.
`values`	A named list or data.frame specifying the variables and values at which marginal means of the outcome will be estimated.
`clusterid`	An optional string containing the name of a cluster identification variable when data are clustered.
`matched_density_cases`	A function of the matching variable. The probability (or density) of the matched variable among the cases.
`matched_density_controls`	A function of the matching variable. The probability (or density) of the matched variable among the controls.
`matching_variable`	The matching variable extracted from the data set.
`p_population`	Specifies the incidence in the population when `case_control=TRUE`.
`case_control`	Whether the data comes from a case-control study.
`ci_level`	Coverage probability of confidence intervals.
`ci_type`	A string, indicating the type of confidence intervals. Either "plain", which gives untransformed intervals, or "log", which gives log-transformed intervals.
`contrasts`	A vector of contrasts in the following format: 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. Has to be `NULL` if no references are specified.
`family`	The family argument which is used to fit the glm model for the outcome.
`reference`	A vector of reference levels in the following format: If `contrasts` is not `NULL`, the desired reference level(s). This must be a vector or list the same length as `contrasts`, and if not named, it is assumed that the order is as specified in contrasts.
`transforms`	A vector of transforms in the following format: 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 the vector is `NULL`, then $\psi(x) = \theta(x)$.

Details

`standardize_glm` 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. `standardize_glm` 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.

Value

An object of class `std_glm`. This is basically a list with components estimates and covariance estimates in `res`. Results for transformations, contrasts, references are stored in `res_contrasts`. Obtain numeric results in a data frame with the tidy function.

References

Rothman K.J., Greenland S., Lash T.L. (2008). *Modern Epidemiology*, 3rd edition. Lippincott, Williams & Wilkins.

Sjölander A. (2016). Regression standardization with the R-package stdReg. *European Journal of Epidemiology* **31**(6), 563-574.

Sjölander A. (2016). Estimation of causal effect measures with the R-package stdReg. *European Journal of Epidemiology* **33**(9), 847-858.

Examples

```
# basic example
# needs to correctly specify the outcome model and no unmeasered confounders
# (+ standard causal assunmptions)
set.seed(6)
n <- 100
Z <- rnorm(n)
X <- rnorm(n, mean = Z)
Y <- rbinom(n, 1, prob = (1 + exp(X + Z))^(-1))
dd <- data.frame(Z, X, Y)
x <- standardize_glm(
  formula = Y ~ X * Z,
  family = "binomial",
  data = dd,
  values = list(X = 0:1),
  contrasts = c("difference", "ratio"),
  reference = 0
)
x
# different transformations of causal effects

# example from Sjölander (2016) with case-control data
# here the matching variable needs to be passed as an argument
singapore <- AF::singapore
Mi <- singapore$Age
m <- mean(Mi)
s <- sd(Mi)
d <- 5
standardize_glm(
  formula = Oesophagealcancer ~ (Everhotbev + Age + Dial + Samsu + Cigs)^2,
  family = binomial, data = singapore,
  values = list(Everhotbev = 0:1), clusterid = "Set",
  case_control = TRUE,
  matched_density_cases = function(x) dnorm(x, m, s),
  matched_density_controls = function(x) dnorm(x, m - d, s),
  matching_variable = Mi,
  p_population = 19.3 / 100000
)

# multiple exposures
set.seed(7)
n <- 100
```

```
Z <- rnorm(n)
X1 <- rnorm(n, mean = Z)
X2 <- rnorm(n)
Y <- rbinom(n, 1, prob = (1 + exp(X1 + X2 + Z))^(-1))
dd <- data.frame(Z, X1, X2, Y)
x <- standardize_glm(
  formula = Y ~ X1 + X2 + Z,
  family = "binomial",
  data = dd, values = list(X1 = 0:1, X2 = 0:1),
  contrasts = c("difference", "ratio"),
  reference = c(X1 = 0, X2 = 0)
)
x
tidy(x)

# continuous exposure
set.seed(2)
n <- 100
Z <- rnorm(n)
X <- rnorm(n, mean = Z)
Y <- rnorm(n, mean = X + Z + 0.1 * X^2)
dd <- data.frame(Z, X, Y)
x <- standardize_glm(
  formula = Y ~ X * Z,
  family = "gaussian",
  data = dd,
  values = list(X = seq(-1, 1, 0.1))
)

# plot standardized mean as a function of x
plot(x)
# plot standardized mean - standardized mean at x = 0 as a function of x
plot(x, contrast = "difference", reference = 0)
```

`standardize_glm_dr` *Get regression standardized doubly-robust estimates from a glm*

Description

Get regression standardized doubly-robust estimates from a glm

Usage

```
standardize_glm_dr(
  formula_outcome,
  formula_exposure,
  data,
  values,
  ci_level = 0.95,
```

```
  ci_type = "plain",
  contrasts = NULL,
  family_outcome = "gaussian",
  family_exposure = "binomial",
  reference = NULL,
  transforms = NULL
)
```

Arguments

`formula_outcome`	The formula which is used to fit the glm model for the outcome.
`formula_exposure`	The formula which is used to fit the glm model for the exposure. If not `NULL`, a doubly robust estimator of the standardized estimator is used.
`data`	The data.
`values`	A named list or data.frame specifying the variables and values at which marginal means of the outcome will be estimated.
`ci_level`	Coverage probability of confidence intervals.
`ci_type`	A string, indicating the type of confidence intervals. Either "plain", which gives untransformed intervals, or "log", which gives log-transformed intervals.
`contrasts`	A vector of contrasts in the following format: 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. Has to be `NULL` if no references are specified.
`family_outcome`	The family argument which is used to fit the glm model for the outcome.
`family_exposure`	The family argument which is used to fit the glm model for the exposure.
`reference`	A vector of reference levels in the following format: If `contrasts` is not `NULL`, the desired reference level(s). This must be a vector or list the same length as `contrasts`, and if not named, it is assumed that the order is as specified in contrasts.
`transforms`	A vector of transforms in the following format: 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 the vector is `NULL`, then $\psi(x) = \theta(x)$.

Details

`standardize_glm_dr` performs regression standardization in generalized linear models, see e.g., documentation for `standardize_glm_dr`. Specifically, this version uses a doubly robust estimator for standardization, meaning inference is valid when either the outcome regression or the exposure model is correctly specified and there is no unmeasured confounding.

Value

An object of class `std_glm`. This is basically a list with components estimates and covariance estimates in `res`. Results for transformations, contrasts, references are stored in `res_contrasts`. Obtain numeric results in a data frame with the tidy function.

References

Gabriel E.E., Sachs, M.C., Martinussen T., Waernbaum I., Goetghebeur E., Vansteelandt S., Sjölander A. (2024), Inverse probability of treatment weighting with generalized linear outcome models for doubly robust estimation. *Statistics in Medicine*, **43**(3):534–547.

Examples

```
# doubly robust estimator
# needs to correctly specify either the outcome model or the exposure model
# for confounding
# NOTE: only works with binary exposures
data <- AF::clslowbwt
x <- standardize_glm_dr(
  formula_outcome = bwt ~ smoker * (race + age + lwt) + I(age^2) + I(lwt^2),
  formula_exposure = smoker ~ race * age * lwt + I(age^2) + I(lwt^2),
  family_outcome = "gaussian",
  family_exposure = "binomial",
  data = data,
  values = list(smoker = c(0, 1)), contrasts = "difference", reference = 0
)

set.seed(6)
n <- 100
Z <- rnorm(n)
X <- rbinom(n, 1, prob = (1 + exp(Z))^(-1))
Y <- rbinom(n, 1, prob = (1 + exp(X + Z))^(-1))
dd <- data.frame(Z, X, Y)
x <- standardize_glm_dr(
  formula_outcome = Y ~ X * Z, formula_exposure = X ~ Z,
  family_outcome = "binomial",
  data = dd,
  values = list(X = 0:1), reference = 0,
  contrasts = c("difference"), transforms = c("odds")
)
```

`standardize_level` *Get standardized estimates using the g-formula with and separate models for each exposure level in the data*

Description

Get standardized estimates using the g-formula with and separate models for each exposure level in the data

Usage

```
standardize_level(
  fitter_list,
  arguments,
  predict_fun_list,
  data,
  values,
  B = NULL,
  ci_level = 0.95,
  contrasts = NULL,
  reference = NULL,
  seed = NULL,
  times = NULL,
  transforms = NULL,
  progressbar = TRUE
)
```

Arguments

`fitter_list`	The function to call to fit the data (as a list).
`arguments`	The arguments to be used in the fitter function as a `list`.
`predict_fun_list`	The function used to predict the means/probabilities for a new data set on the response level. For survival data, this should be a matrix where each column is the time, and each row the data (as a list).
`data`	The data.
`values`	A named list or data.frame specifying the variables and values at which marginal means of the outcome will be estimated.
`B`	Number of nonparametric bootstrap resamples. Default is `NULL` (no bootstrap).
`ci_level`	Coverage probability of confidence intervals.
`contrasts`	A vector of contrasts in the following format: 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. Has to be `NULL` if no references are specified.
`reference`	A vector of reference levels in the following format: If `contrasts` is not `NULL`, the desired reference level(s). This must be a vector or list the same length as `contrasts`, and if not named, it is assumed that the order is as specified in contrasts.
`seed`	The seed to use with the nonparametric bootstrap.
`times`	For use with survival data. Set to `NULL` otherwise.
`transforms`	A vector of transforms in the following format: 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 the vector is `NULL`, then $\psi(x) = \theta(x)$.
`progressbar`	Logical, if TRUE will print bootstrapping progress to the console

Details

See `standardize`. The difference is here that different models can be fitted for each value of `x` in `values`.

Value

An object of class `std_custom`. This is a list with components estimates and fit for the outcome model.

References

Rothman K.J., Greenland S., Lash T.L. (2008). *Modern Epidemiology*, 3rd edition. Lippincott, Williams & Wilkins.

Sjölander A. (2016). Regression standardization with the R-package stdReg. *European Journal of Epidemiology* **31**(6), 563-574.

Sjölander A. (2016). Estimation of causal effect measures with the R-package stdReg. *European Journal of Epidemiology* **33**(9), 847-858.

Examples

```
require(survival)
prob_predict.coxph <- function(object, newdata, times) {
  fit.detail <- suppressWarnings(basehaz(object))
 cum.haz <- fit.detail$hazard[sapply(times, function(x) max(which(fit.detail$time <= x)))]
  predX <- predict(object = object, newdata = newdata, type = "risk")
  res <- matrix(NA, ncol = length(times), nrow = length(predX))
  for (ti in seq_len(length(times))) {
    res[, ti] <- exp(-predX * cum.haz[ti])
  }
  res
}

set.seed(68)
n <- 500
Z <- rnorm(n)
X <- rbinom(n, 1, prob = 0.5)
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)
x <- standardize_level(
  fitter_list = list("coxph", "coxph"),
  arguments = list(
    list(
      formula = Surv(U, D) ~ X + Z + X * Z,
      method = "breslow",
      x = TRUE,
      y = TRUE
    ),
```

```
    list(
      formula = Surv(U, D) ~ X,
      method = "breslow",
      x = TRUE,
      y = TRUE
    )
  ),
  predict_fun_list = list(prob_predict.coxph, prob_predict.coxph),
  data = dd,
  times = seq(1, 5, 0.1),
  values = list(X = c(0, 1)),
  B = 100,
  reference = 0,
  contrasts = "difference"
)
print(x)
```

`standardize_parfrailty` *Regression standardization in shared frailty gamma-Weibull models*

Description

`standardize_parfrailty` 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. `standardize_parfrailty` fits a parametric frailty 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

```
standardize_parfrailty(
  formula,
  data,
  values,
  times,
  clusterid,
  ci_level = 0.95,
  ci_type = "plain",
  contrasts = NULL,
  family = "gaussian",
  reference = NULL,
  transforms = NULL
)
```

Arguments

`formula`	The formula which is used to fit the model for the outcome.
`data`	The data.
`values`	A named list or data.frame specifying the variables and values at which marginal means of the outcome will be estimated.
`times`	A vector containing the specific values of T at which to estimate the standardized survival function.
`clusterid`	An optional string containing the name of a cluster identification variable when data are clustered.
`ci_level`	Coverage probability of confidence intervals.
`ci_type`	A string, indicating the type of confidence intervals. Either "plain", which gives untransformed intervals, or "log", which gives log-transformed intervals.
`contrasts`	A vector of contrasts in the following format: 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. Has to be `NULL` if no references are specified.
`family`	The family argument which is used to fit the glm model for the outcome.
`reference`	A vector of reference levels in the following format: If `contrasts` is not `NULL`, the desired reference level(s). This must be a vector or list the same length as `contrasts`, and if not named, it is assumed that the order is as specified in contrasts.
`transforms`	A vector of transforms in the following format: 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 the vector is `NULL`, then $\psi(x) = \theta(x)$.

Details

`standardize_parfrailty` fits a shared frailty gamma-Weibull model

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

, with parameterization as described in the help section for parfrailty. Integrating out the gamma frailty gives the survival function

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

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 $S(t|X = x, Z)$:

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

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

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

The variance for $\hat{\theta}(t,x)$ is obtained by the sandwich formula.

Value

An object of class `std_surv`. This is basically a list with components estimates and covariance estimates in `res` Results for transformations, contrasts, references are stored in `res_contrasts`. The output contains estimates for contrasts and confidence intervals for all combinations of transforms and reference levels. Obtain numeric results in a data frame with the tidy function.

Note

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

`standardize_coxph/standardize_parfrailty` does not currently handle time-varying exposures or covariates.

`standardize_coxph/standardize_parfrailty` 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 `standardize_coxph` does not condition on the observed covariates $\bar{Z} = (Z_1, ..., Z_n)$. To see how this matters, note that

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

The usual parameter β 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 $var[E\{\hat{\beta}|\bar{Z}\}]$ in the corresponding variance decomposition for $var(\hat{\beta})$ 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 $var[E\{\hat{\theta}(t,x)|\bar{Z}\}]$ 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 Sjölander

References

Chang I.M., Gelman G., Pagano M. (1982). Corrected group prognostic curves and summary statistics. *Journal of Chronic Diseases* **35**, 669-674.

Dahlqwist E., Pawitan Y., Sjölander A. (2019). Regression standardization and attributable fraction estimation with between-within frailty models for clustered survival data. *Statistical Methods in Medical Research* **28**(2), 462-485.

Gail M.H. and Byar D.P. (1986). Variance calculations for direct adjusted survival curves, with applications to testing for no treatment effect. *Biometrical Journal* **28**(5), 587-599.

Makuch R.W. (1982). Adjusted survival curve estimation using covariates. *Journal of Chronic Diseases* **35**, 437-443.

Examples

```
require(survival)

# simulate data
set.seed(6)
n <- 300
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)
# reparameterize 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.std <- standardize_parfrailty(
  formula = Surv(L, T, D) ~ X,
  data = dd,
  values = list(X = seq(-1, 1, 0.5)),
  times = 1:5,
  clusterid = "id"
)
print(fit.std)
plot(fit.std)
```

`summary.parfrailty` *Summarizes parfrailty fit*

Description

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

Usage

```
## S3 method for class 'parfrailty'
summary(
  object,
  ci_type = "plain",
  ci_level = 0.95,
  digits = max(3L, getOption("digits") - 3L),
  ...
)
```

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.

Value

An object of class "summary.parfrailty", which is a list that contains relevant summary statistics about the fitted model

Author(s)

Arvid Sjölander and Elisabeth Dahlqwist.

See Also

`parfrailty`

Examples

```
## See documentation for parfrailty
```

`tidy.std_glm` *Provide tidy output from a std_glm object for use in downstream computations*

Description

Tidy summarizes information about the components of the standardized regression fit.

Usage

```
## S3 method for class 'std_glm'
tidy(x, ...)
```

Arguments

`x`	An object of class std_glm
`...`	Not currently used

Value

A data.frame

Examples

```
set.seed(6)
n <- 100
Z <- rnorm(n)
X <- rnorm(n, mean = Z)
Y <- rbinom(n, 1, prob = (1 + exp(X + Z))^(-1))
dd <- data.frame(Z, X, Y)
x <- standardize_glm(
  formula = Y ~ X * Z,
  family = "binomial",
  data = dd,
  values = list(X = 0:1),
  contrasts = c("difference", "ratio"),
  reference = 0
)
tidy(x)
```

`tidy.std_surv` *Provide tidy output from a std_surv object for use in downstream computations*

Description

Tidy summarizes information about the components of the standardized model fit.

Usage

```
## S3 method for class 'std_surv'
tidy(x, ...)
```

Arguments

`x`	An object of class std_surv
`...`	Not currently used

Value

A data.frame

Examples

```
require(survival)
set.seed(8)
n <- 300
Z <- rnorm(n)
X <- rnorm(n, mean = Z)
time <- rexp(n, rate = exp(X + Z + X * Z)) # survival time
C <- rexp(n, rate = exp(X + Z + X * Z)) # censoring time
U <- pmin(time, C) # time at risk
D <- as.numeric(time < C) # event indicator
dd <- data.frame(Z, X, U, D)
x <- standardize_coxph(
  formula = Surv(U, D) ~ X + Z + X * Z,
  data = dd, values = list(X = seq(-1, 1, 0.5)), times = c(2,3,4)
)

tidy(x)
```

Index

coxph, *3*, *8*
coxph.detail, *8*

glm, *8*

legend, *6*

parfrailty, 3, *27*, *30*
plot.std_glm, 5
plot.std_surv, 6
print.std_custom *(*print.std_surv*)*, 7
print.std_glm *(*print.std_surv*)*, 7
print.std_surv, 7
print.summary.parfrailty, 7

sandwich, 8
standardize, 9
standardize_coxph, 11
standardize_gee, 15
standardize_glm, 18
standardize_glm_dr, 21
standardize_level, 23
standardize_parfrailty, 26
stdReg2 *(*stdReg2-package*)*, 2
stdReg2-package, 2
summary.parfrailty, 29
survfit, *8*

tidy, *13*, *17*, *19*, *23*, *28*
tidy.std_glm, 31
tidy.std_surv, 32