Package: gcomputation (via r-universe)

May 12, 2026

Type Package

Title Causal Inference by using G-Computation

Version 0.34

Depends R (>= 4.0.0), survival, hdnom, glmnet, MASS, mice

Imports graphics, utils, methods, grDevices, stats

Description Several functions and S3 methods for G-computation and emulation of clinical trials. It allows for flexible estimation of the outcome model, especially penalized regressions (Lasso, Ridge, or Elasticnet) for binary, continuous, counting, or right-censored time-to-event outcomes. Average treatment effect among the entire population (ATE) or among the treated population (ATT) can be estimated. The method for time-to-events is described by Chatton et al. (2020) <doi:10.1038/s41598-020-65917-x>. For a binary outcome, details are available in the paper proposed by Chatton et al. (2022) <doi:10.1177/09622802211047345>.

License GPL (>=2)

Encoding UTF-8

LazyLoad yes

NeedsCompilation no

Maintainer Yohann Foucher <yohann.foucher@univ-poitiers.fr>

BugReports https://github.com/chupverse/gcomputation/issues

RoxygenNote 7.3.3

Config/pak/sysreqs cmake make libicu-dev libx11-dev zlib1g-dev

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

Date/Publication 2026-05-06 13:16:09 UTC

RemoteUrl https://github.com/chupverse/gcomputation

RemoteRef HEAD

RemoteSha 4e2d75b738cfa007c5245a77f24b9c862b645680

Contents

dataCOHORT . . . 2
dataPROPHYVAP . . . 3
gc_binary . . . 5
gc_continuous . . . 8
gc_count . . . 11
gc_times . . . 14
plot.gcbinary . . . 18
plot.gccontinuous . . . 19
plot.gccount . . . 20
plot.gctimes . . . 21
print.gcbinary . . . 23
print.gccontinuous . . . 24
print.gccount . . . 24
print.gctimes . . . 25
summary.gcbinary . . . 26
summary.gccontinuous . . . 27
summary.gccount . . . 29
summary.gctimes . . . 30
transport . . . 31

Index **34**

`dataCOHORT` *Simulated Real Word Data Similar to the PROPHYVAP Study*

Description

This dataset `dataCOHORT` is a simulated observational cohort designed to reflect a real-world data similar to the PROPHYVAP clinical trial (see `dataPROPHYVAP`).

Usage

```
data(dataCOHORT)
```

Format

A data frame with 2000 observations for the following variables:

`GROUP` This character vector represents the treatment group (1 for Ceftriaxone and 0 otherwise)

`AGE` This numeric vector represents the patient age in years

`SEX` This character vector represents the gender (F=female/M=male)

`BMI` This numeric vector represents the body mass index in kg/m2

`DIABETES` This character vector represents the diabetes status (Yes/No)

`ALCOHOL` This character vector represents the alcohol consumption (Yes/No)

`SMOKING` This character vector represents the tabaco status (Yes/No)

`INJURY` This character vector represents the cause of the injury in 4 classes

`GLASGOW` This character vector represents the Glasgow scale in 3 classes

`PAO2FIO2` This character vector represents the PAO2-FIO2 ratio in 2 classes

`LEUKO` This character vector represents leukocytosis at admission per mm3 in 2 classes

`TIME_INTUBATION` This numeric vector represents the time to intubation in hours

`VAP` This character vector represents ventilatory associated pneumonia (1 for event and 0 otherwise)

`TIME_DEATH` This numeric vector represents the time to death in days (follow-up of 60 days)

`DEATH` This character vector represents status at follow-up end (1 for event and 0 otherwise)

References

Dahyot-Fizelier et al. Ceftriaxone to prevent early ventilator-associated pneumonia in patients with acute brain injury: a multicentre, randomised, double-blind, placebo-controlled, assessor-masked superiority trial. Lancet Respir Med, 12:375-385, 2024. <doi:10.1016/S2213-2600(23)00471-X>.

Examples

```
data(dataCOHORT)

### Kaplan and Meier estimation of the survival at day 60
plot(survfit(Surv(TIME_DEATH, DEATH) ~ 1, data = dataCOHORT),
 xlab="Time in days", ylab="Patient survival", conf.int=TRUE,
 mark.time=FALSE)
```

---

dataPROPHYVAP *A Simulated Randomized Clinical Trial from the PROPHYVAP Study*

Description

Ventilator-associated pneumonia (VAP) is the first cause of healthcare-associated infections in intensive care units. The PROPHYVAP is a French multicenter, randomized, double-blind, placebo-controlled, clinical trial. The main objective of this study was to determine whether a single dose of Ceftriaxone within the 12 hours post-intubation can decrease the risk of early-onset VAP and mortality at 60 days in patients admitted for severe brain injury. All variables in this dataset were simulated according to the original trial. The number of episodes of severe hypotension was not observed and completely generated by Poisson regression for illustrative purposes of this package.

Usage

```
data(dataPROPHYVAP)
```

Format

A data frame with 319 observations for the following variables:

`GROUP` This character vector represents the treatment (1 for Ceftriaxone and 0 for Placebo)

`AGE` This numeric vector represents the patient age in years

`SEX` This character vector represents the gender (F=female/M=male)

`BMI` This numeric vector represents the body mass index in kg/m2

`DIABETES` This character vector represents the diabetes status (Yes/No)

`ALCOHOL` This character vector represents the alcohol consumption (Yes/No)

`SMOKING` This character vector represents the tabaco status (Yes/No)

`INJURY` This character vector represents the cause of the injury in 4 classes

`GLASGOW` This character vector represents the Glasgow scale in 3 classes

`PAO2FIO2` This character vector represents the PAO2-FIO2 ratio in 2 classes

`LEUKO` This character vector represents leukocytosis at admission per mm3 in 2 classes

`TIME_INTUBATION` This numeric vector represents the time to intubation in hours

`TIME_TRT` This numeric vector represents the time to treatment in hours

`VAP` This character vector represents ventilatory associated pneumonia (1 for event and 0 otherwise)

`TIME_DEATH` This numeric vector represents the time to death in days (follow-up of 60 days)

`DEATH` This character vector represents status at follow-up end (1 for event and 0 otherwise)

`HYPOTENSION` This numeric vector represents the number of episodes of severe hypotension.

References

Dahyot-Fizelier et al. Ceftriaxone to prevent early ventilator-associated pneumonia in patients with acute brain injury: a multicentre, randomised, double-blind, placebo-controlled, assessor-masked superiority trial. Lancet Respir Med, 12:375-385, 2024. <doi:10.1016/S2213-2600(23)00471-X>.

Examples

```
data(dataPROPHYVAP)

### Kaplan and Meier estimation of the survival at day 60
plot(survfit(Surv(TIME_DEATH, DEATH) ~ GROUP, data = dataPROPHYVAP),
 xlab="Post-randomization time in days", ylab="Patient survival",
 mark.time=FALSE, col=c("red3","blue3"), lty=c(2,1))

legend("bottomleft", c("Placebo", "Ceftriaxone"), col=c("red3","blue3"), lty=1:2)
```

gc_binary *G-Computation to Estimate a Marginal Effect for a Binary Outcome*

Description

This function computes G-computation (GC) with different working models or algorithms (Q-models) for a binary outcome and a 2-class exposure/treatment.

Usage

```
gc_binary(formula, data, group, effect="ATE", model,
            param.tune=NULL, cv=30, boot.type="bcv", boot.number=500,
            boot.tune=FALSE, progress=TRUE, seed=NULL, boot.mi=FALSE, m=5, ...)
```

Arguments

formula	A regression formula related to the Q-model with the variable group among the predictors. The outcome must have two modalities encoded as 0 for non-events and 1 for events.
data	A data frame in which to look for the variables related to the outcome, the studied (group) and the predictors included in the previous model.
group	The name of the variable related to the exposure/treatment. This variable shall have only two modalities encoded 0 for the untreated/unexposed subjects and 1 for the treated/exposed ones.
effect	The type of the marginal effect to be estimated. Three types are possible: "ATE" (by default), "ATT" and "ATU". See details.
model	The modelling method used to create the Q-model. Current implemented methods are: "all", "lasso", "ridge", "elasticnet", "aic" and "bic". See details.
param.tune	An optional argument to specify the tuning parameter(s) for the previous modelling method. If NULL (the default), the tuning parameter(s) is(are) estimated by cv-fold cross-validation. Otherwise, the user can propose a tuning grid. See details.
cv	The number of splits for cross-validation. The default value is 30.
boot.type	The type of bootstrap to use. Two types are possible: "bcv" for bootstrap cross-validation (by default) and "boot" usual bootstrap. See details.
boot.number	The number of bootstrap resamples. The default value is 500.
boot.tune	A logical value to determine whether the tuning parameter(s) should be estimated inside of each bootstrap iteration. See details.
progress	A logical value to print a progress bar. The default is TRUE
seed	A random seed to ensure reproducibility during the cv process. If NULL, a seed is randomly assigned.
boot.mi	A logical value to apply multiple imputation using the mice package. See details.

m	Number of imputations to perform if boot.mi is TRUE.
...	Additional arguments to be passed directly to the mice function for customizing the multiple imputation process (e.g., method, maxit, diagnostics).

Details

The option effect="ATE" corresponds to the Average Treatment effect on the Entire population, i.e. the marginal effect if the entire sample were treated versus untreated. The "ATT" modality allows the estimation of the Average Treatment effect on the Treated, i.e. the marginal effect if the treated subjects (group = 1) would have been untreated. The "ATU" modality allows the estimation of the Average Treatment effect on the Untreated, i.e. the marginal effect if the untreated subjects (group = 0) would have been treated.

Several modelling methods can be used for the Q-model estimation:

"all"	A logistic regression, all the covariates in the formula being used.
"lasso"	L1 regularized logistic regression allowing predictors' selection.
"ridge"	L2 regularized logistic regression allowing highly correlated predictors.
"elasticnet"	A logistic regression which combines both L1 and L2 regularizations.
"aic"	A logistic regression with a AIC-based forward selection.
"bic"	A logistic regression with a BIC-based forward selection.

The param.tune argument allows users to specify tuning parameters of penalized regression. If NULL (the default), the tuning parameters of each algorithm are estimated by cv-fold cross-validation and the default grid of the glmnet package is used. Otherwise, the user can propose a specific grid. For "lasso" and "ridge" penalizations, it should be a vector representing the lambda penalization parameter. For "elasticnet", it should be a list or a vector of length 2, containing lambda (penalization parameter) and alpha (mixing parameter between L1 and L2 regularizations) values. The alpha value typically ranges from 0 to 1. The user may propose a single value of each tuning parameter if she/he aims to define her/his own penalty.

The boot.tune argument is logical value which determines whether the tuning parameter should be estimated inside of each bootstrap iteration. If FALSE (the default), the tuning parameter will be estimated once on the complete dataset.

The boot.type argument controls how bootstrap estimates are computed. With boot.type = "boot", each iteration fits the Q-model on a sampled dataset and predicts on the same patients. With boot.type = "bcv", the Q-model is fitted on the sampled patients, and predictions are made on patients not included in that sample.

If boot.mi = TRUE, multiple imputation is considered by using the MI-BOOT approach proposed by Schomaker & Heumann (2018). The dataset is first imputed using the mice function to create m complete datasets. The bootstrap samples are then generated from each imputed dataset. The Q-model and the G-computation is performed for each m x boot.number samples, and all runs are finally concatenated.

Value

qmodel.fit	The fitted Q-model.
predictions	The outcome predictions obtained on the complete dataset.

`tuning.parameters`
: The estimated tuning parameters for the Q-model. For "aic" and "bic" methods, this represents the final model.

`data`
: The data frame with individual with no missing data in the formula parameters.

`formula`
: The formula provided by the user.

`model`
: The method used for the Q-model.

`cv`
: The number of splits used for cross-validation.

`penalty.factor`
: The penalty factors used for the penalized regression. The variable `group` has a penalty factor of 0 (no penalization) and 1 for the others.

`missing`
: The number of observations that were removed from the original dataset due to missing values.

`boot.number`
: The total number of bootstrap resamples.

`boot.type`
: The type of bootstrap.

`group`
: A character string specifying the name of the variable related to the exposure or treatment.

`n`
: The sample size of the dataset after missing data removal (if `boot.mi = FALSE`).

`nevent`
: The total number of events in the dataset.

`adjusted.results`
: A data frame containing the adjusted results for each bootstrap sample, including: `p1` (proportion in experimental group), `p0` (proportion in control group), `delta` (risk difference), `ratio` (risk ratio), and `OR` (odds ratio).

`unadjusted.results`
: A data frame containing the unadjusted results for each bootstrap sample, including: `p1` (proportion in experimental group), `p0` (proportion in control group), `delta` (risk difference), `ratio` (risk ratio), and `OR` (odds ratio).

`call`
: The function call that generated the `gcbinary` object.

`m`
: The number of multiple imputations performed (only present if `mi.boot = TRUE`).

`initial.data`
: The original dataset provided by the user, before any imputation (only present if `mi.boot = TRUE`).

`nimput`
: The number of observations with missing values that were removed from the dataset prior to imputation (only present if `mi.boot = TRUE`).

`seed`
: The random seed used.

References

Joe de Keizer et al. G-computation for increasing performances of clinical trials with individual randomization and binary response. ArXiv 2024-11. <doi:10.48550/arXiv.2411.10089>.

Schomaker M, Heumann C. Bootstrap inference when using multiple imputation. Statistics in medicine. 2018;37(14):2252-2266. <doi:10.1002/sim.7654>.

Examples

```
data("dataPROPHYVAP")

.f <- formula(VAP ~ GROUP * AGE + SEX + ALCOHOL + BMI + DIABETES + GLASGOW + INJURY)

### In practice use larger values for cv and boot.number
### We set boot.number and cv at 10 for speed in CRAN checks
### cv=30 and boot.number=1000 are more appropriate values
gc1 <- gc_binary(formula=.f, model="lasso", data=dataPROPHYVAP,
              group="GROUP", param.tune=NULL, cv=10, boot.type="bcv",
              boot.number=10, effect="ATE", boot.tune=TRUE)

summary(gc1)
```

`gc_continuous` *G-Computation to Estimate a Marginal Effect with a Continuous Outcome*

Description

This function computes G-computation (GC) with different working models or algorithms (Q-models) for a continuous outcome and a 2-class exposure/treatment.

Usage

```
gc_continuous(formula, data, group, effect="ATE", model,
            param.tune=NULL, cv=30, boot.type="bcv", boot.number=500,
            boot.tune=FALSE, progress=TRUE, seed=NULL, boot.mi=FALSE, m=5, ...)
```

Arguments

`formula`	A regression formula related to the Q-model with the variable `group` among the predictors.
`data`	A data frame in which to look for the variables related to the outcome, the studied (`group`) and the predictors included in the previous model.
`group`	The name of the variable related to the exposure/treatment. This variable shall have only two modalities encoded 0 for the untreated/unexposed subjects and 1 for the treated/exposed ones.
`effect`	The type of the marginal effect to be estimated. Three types are possible: "ATE" (by default), "ATT" and "ATU". See details.
`model`	The modelling method used to create the Q-model. Current implemented methods are: "all", "lasso", "ridge", "elasticnet", "aic" and "bic". See details.
`param.tune`	An optional argument to specify the tuning parameter(s) for the previous modelling method. If `NULL` (the default), the tuning parameter(s) is(are) estimated by `cv`-fold cross-validation. Otherwise, the user can propose a tuning grid. See details.

cv	The number of splits for cross-validation. The default value is 30.
boot.type	The type of bootstrap to use. Two types are possible: "bcv" for bootstrap cross-validation (by default) and "boot" usual bootstrap. See details.
boot.number	The number of bootstrap resamples. The default value is 500.
boot.tune	A logical value to determine whether the tuning parameter(s) should be estimated inside of each bootstrap iteration. See details.
progress	A logical value to print a progress bar. The default is TRUE
seed	A random seed to ensure reproducibility during the cv process. If NULL, a seed is randomly assigned.
boot.mi	A logical value to apply multiple imputation using the mice package. See details.
m	Number of imputations to perform if boot.mi is TRUE.
...	Additional arguments to be passed directly to the mice function for customizing the multiple imputation process (e.g., method, maxit, diagnostics).

Details

The option methods="ATE" corresponds to the Average Treatment effect on the Entire population, i.e. the marginal effect if the entire sample were treated versus untreated. The "ATT" modality allows the estimation of the Average Treatment effect on the Treated, i.e. the marginal effect if the treated subjects (group = 1) would have been untreated. The "ATU" modality allows the estimation of the Average Treatment effect on the Untreated, i.e. the marginal effect if the untreated subjects (group = 0) would have been treated.

Several modelling methods can be used for the Q-model estimation:

"all"	A linear regression, all the covariates in the formula being used.
"lasso"	L1 regularized linear regression allowing predictors' selection.
"ridge"	L2 regularized linear regression allowing highly correlated predictors.
"elasticnet"	A linear regression which combines both L1 and L2 regularizations.
"aic"	A linear regression with a AIC-based forward selection.
"bic"	A linear regression with a BIC-based forward selection.

The param.tune argument allows users to specify tuning parameters of penalized regression. If NULL (the default), the tuning parameters of each algorithm are estimated by cv-fold cross-validation and the default grid of the glmnet package is used. Otherwise, the user can propose a specific grid. For "lasso" and "ridge" penalizations, it should be a vector representing the lambda penalization parameter. For "elasticnet", it should be a list or a vector of length 2, containing lambda (penalization parameter) and alpha (mixing parameter between L1 and L2 regularizations) values. The alpha value typically ranges from 0 to 1. The user may propose a single value of each tuning parameter if she/he aims to define her/his own penalty.

The boot.tune argument is logical value which determines whether the tuning parameter should be estimated inside of each bootstrap iteration. If FALSE (the default), the tuning parameter will be estimated once on the complete dataset.

The boot.type argument controls how bootstrap estimates are computed. With boot.type = "boot", each iteration fits the Q-model on a sampled dataset and predicts on the same patients.

With boot.type = "bcv", the Q-model is fitted on the sampled patients, and predictions are made on patients not included in that sample.

If boot.mi = TRUE, multiple imputation is considered by using the MI-BOOT approach proposed by Schomaker & Heumann (2018). The dataset is first imputed using the mice function to create m complete datasets. The bootstrap samples are then generated from each imputed dataset. The Q-model and the G-computation is performed for each m x boot.number samples, and all runs are finally concatenated.

Value

qmodel.fit — The fitted Q-model.

predictions — The outcome predictions obtained on the complete dataset.

tuning.parameters — The estimated tuning parameters for the Q-model. For "aic" and "bic" methods, this represents the final model.

data — The data frame with individual with no missing data in the formula parameters.

formula — The formula provided by the user.

model — The method used for the Q-model.

cv — The number of splits used for cross-validation.

penalty.factor — The penalty factors used for the penalized regression. The variable group has a penalty factor of 0 (no penalization) and 1 for the others.

missing — The number of observations that were removed from the original dataset due to missing values.

boot.number — The total number of bootstrap resamples.

boot.type — The type of bootstrap.

group — A character string specifying the name of the variable related to the exposure or treatment.

n — The sample size of the dataset after missing data removal (if boot.mi = FALSE).

adjusted.results — A data frame containing the adjusted results for each bootstrap sample: m1 (mean experimental), m0 (mean control), delta (difference), and ratio (mean ratio).

unadjusted.results — A data frame containing the unadjusted results for each bootstrap sample: m1 (mean experimental), m0 (mean control), delta (difference), and ratio (mean ratio).

call — The function call that generated the gccontinuous object.

m — The number of multiple imputations performed (only present if mi.boot = TRUE).

initial.data — The original dataset provided by the user, before any imputation (only present if mi.boot = TRUE).

nimput — The number of observations with missing values that were removed from the dataset prior to imputation (only present if mi.boot = TRUE).

seed — The random seed used.

References

Joe de Keizer et al. G-computation for increasing performances of clinical trials with individual randomization and binary response. ArXiv 2024-11. <doi:10.48550/arXiv.2411.10089>.

Schomaker M, Heumann C. Bootstrap inference when using multiple imputation. Statistics in medicine. 2018;37(14):2252-2266. <doi:10.1002/sim.7654>.

Examples

```
data("dataPROPHYVAP")

.f <- formula(HYPOTENSION ~ GROUP * AGE + SEX + ALCOHOL + BMI + DIABETES + GLASGOW + INJURY)

### In practice use larger values for cv and boot.number
### We set boot.number and cv at 10 for speed in CRAN checks
### cv=30 and boot.number=1000 are more appropriate values
gc1 <- gc_continuous(formula=.f, model="lasso", data=dataPROPHYVAP,
              group="GROUP", param.tune=NULL, cv=10, boot.type="bcv",
              boot.number=10, effect="ATE", boot.tune=TRUE)

summary(gc1)
```

`gc_count` *G-Computation to Estimate a Marginal Effect with a Counting Outcome*

Description

This function computes G-computation (GC) with different working models or algorithms (Q-models) for a counting outcome and a 2-class exposure/treatment.

Usage

```
gc_count(formula, data, group, effect="ATE", model,
            param.tune=NULL, cv=30, boot.type="bcv", boot.number=500,
            boot.tune=FALSE, progress=TRUE, seed=NULL, boot.mi=FALSE, m=5, ...)
```

Arguments

`formula`	A regression formula related to the Q-model with the variable `group` among the predictors.
`data`	A data frame in which to look for the variables related to the outcome, the studied (`group`) and the predictors included in the previous model.
`group`	The name of the variable related to the exposure/treatment. This variable shall have only two modalities encoded 0 for the untreated/unexposed subjects and 1 for the treated/exposed ones.
`effect`	The type of the marginal effect to be estimated. Three types are possible: "ATE" (by default), "ATT" and "ATU". See details.

model	The modelling method used to create the Q-model. Current implemented methods are: "all", "lasso", "ridge", "elasticnet", "aic" and "bic". See details.
param.tune	An optional argument to specify the tuning parameter(s) for the previous modelling method. If NULL (the default), the tuning parameter(s) is(are) estimated by cv-fold cross-validation. Otherwise, the user can propose a tuning grid. See details.
cv	The number of splits for cross-validation. The default value is 30.
boot.type	The type of bootstrap to use. Two types are possible: "bcv" for bootstrap cross-validation (by default) and "boot" usual bootstrap. See details.
boot.number	The number of bootstrap resamples. The default value is 500.
boot.tune	A logical value to determine whether the tuning parameter(s) should be estimated inside of each bootstrap iteration. See details.
progress	A logical value to print a progress bar. The default is TRUE
seed	A random seed to ensure reproducibility during the cv process. If NULL, a seed is randomly assigned.
boot.mi	A logical value to apply multiple imputation using the mice package. See details.
m	Number of imputations to perform if boot.mi is TRUE.
...	Additional arguments to be passed directly to the mice function for customizing the multiple imputation process (e.g., method, maxit, diagnostics).

Details

The option methods="ATE" corresponds to the Average Treatment effect on the Entire population, i.e. the marginal effect if the entire sample were treated versus untreated. The "ATT" modality allows the estimation of the Average Treatment effect on the Treated, i.e. the marginal effect if the treated subjects (group = 1) would have been untreated. The "ATU" modality allows the estimation of the Average Treatment effect on the Untreated, i.e. the marginal effect if the untreated subjects (group = 0) would have been treated.

Several modelling methods can be used for the Q-model estimation:

"all"	A Poission regression, all the covariates in the formula being used.
"lasso"	L1 regularized Poission regression allowing predictors' selection.
"ridge"	L2 regularized Poission regression allowing highly correlated predictors.
"elasticnet"	A Poissionregression which combines both L1 and L2 regularizations.
"aic"	A Poission regression with a AIC-based forward selection.
"bic"	A Poission regression with a BIC-based forward selection.

The param.tune argument allows users to specify tuning parameters of penalized regression. If NULL (the default), the tuning parameters of each algorithm are estimated by cv-fold cross-validation and the default grid of the glmnet package is used. Otherwise, the user can propose a specific grid. For "lasso" and "ridge" penalizations, it should be a vector representing the lambda penalization parameter. For "elasticnet", it should be a list or a vector of length 2, containing lambda (penalization parameter) and alpha (mixing parameter between L1 and L2 regularizations) values. The alpha

value typically ranges from 0 to 1. The user may propose a single value of each tuning parameter if she/he aims to define her/his own penalty.

The boot.tune argument is logical value which determines whether the tuning parameter should be estimated inside of each bootstrap iteration. If FALSE (the default), the tuning parameter will be estimated once on the complete dataset.

The boot.type argument controls how bootstrap estimates are computed. With boot.type = "boot", each iteration fits the Q-model on a sampled dataset and predicts on the same patients. With boot.type = "bcv", the Q-model is fitted on the sampled patients, and predictions are made on patients not included in that sample.

If boot.mi = TRUE, multiple imputation is considered by using the MI-BOOT approach proposed by Schomaker & Heumann (2018). The dataset is first imputed using the mice function to create m complete datasets. The bootstrap samples are then generated from each imputed dataset. The Q-model and the G-computation is performed for each m x boot.number samples, and all runs are finally concatenated.

Value

qmodel.fit	The fitted Q-model.
predictions	The outcome predictions obtained on the complete dataset.
tuning.parameters	The estimated tuning parameters for the Q-model. For "aic" and "bic" methods, this represents the final model.
data	The data frame with individual with no missing data in the formula parameters.
formula	The formula provided by the user.
model	The method used for the Q-model.
cv	The number of splits used for cross-validation.
penalty.factor	The penalty factors used for the penalized regression. The variable group has a penalty factor of 0 (no penalization) and 1 for the others.
missing	The number of observations that were removed from the original dataset due to missing values.
boot.number	The total number of bootstrap resamples.
boot.type	The type of bootstrap.
group	A character string specifying the name of the variable related to the exposure or treatment.
n	The sample size of the dataset after missing data removal (if boot.mi = FALSE).
adjusted.results	A data frame containing the adjusted results for each bootstrap sample: c1 (mean count experimental), c0 (mean count control), delta (difference), and ratio (rate ratio).
unadjusted.results	A data frame containing the unadjusted results for each bootstrap sample: c1 (mean count experimental), c0 (mean count control), delta (difference), and ratio (rate ratio).
call	The function call that generated the gccontinuous object.

m	The number of multiple imputations performed (only present if mi.boot = TRUE).
initial.data	The original dataset provided by the user, before any imputation (only present if mi.boot = TRUE).
nimput	The number of observations with missing values that were removed from the dataset prior to imputation (only present if mi.boot = TRUE).
seed	The random seed used.

References

Joe de Keizer et al. G-computation for increasing performances of clinical trials with individual randomization and binary response. ArXiv 2024-11. <doi:10.48550/arXiv.2411.10089>.

Schomaker M, Heumann C. Bootstrap inference when using multiple imputation. Statistics in medicine. 2018;37(14):2252-2266. <doi:10.1002/sim.7654>.

Examples

```
data("dataPROPHYVAP")

.f <- formula(HYPOTENSION ~ GROUP * AGE + SEX + ALCOHOL + BMI + DIABETES + GLASGOW + INJURY)

### In practice use larger values for cv and boot.number
### We set boot.number and cv at 10 for speed in CRAN checks
### cv=30 and boot.number=1000 are more appropriate values
gc1 <- gc_count(formula=.f, model="lasso", data=dataPROPHYVAP,
              group="GROUP", param.tune=NULL, cv=10, boot.type="bcv",
              boot.number=10, effect="ATE", boot.tune=TRUE)

summary(gc1)
```

---

gc_times *G-Computation to Estimate a Marginal Effect for Time-to-Event Outcome*

Description

This function computes G-computation (GC) with different working models or algorithms (Q-models) for a time-to-event outcome and a 2-class exposure/treatment.

Usage

```
gc_times(formula, data, group, pro.time=NULL, effect="ATE", model,
            param.tune=NULL, cv=30, boot.type="bcv", boot.number=500,
            boot.tune=FALSE, progress=TRUE, seed=NULL, boot.mi=FALSE, m=5, ...)
```

Arguments

formula	A formula object, with the response on the left of a ~ operator, and the terms on the right. The response must be a survival object as returned by the Surv function.
data	A data frame in which to look for the variables related to the outcome, the studied (group) and the predictors included in the previous model.
group	The name of the variable related to the exposure/treatment. This variable shall have only two modalities encoded 0 for the untreated/unexposed subjects and 1 for the treated/exposed ones.
pro.time	An optional value for censoring the follow-up times and obtaining survival curves and related restricted mean survival times up to pro.time. If not specified, it is automatically set to the time point at which 10% of subjects remain at risk, or to the max follow-up time if survival never drops below 10%.
effect	The type of the marginal effect to be estimated. Three types are possible: "ATE" (by default), "ATT" and "ATU". See details.
model	The modelling method used to create the Q-model. Current implemented methods are: "all", "lasso", "ridge", "elasticnet", "aic" and "bic". See details.
param.tune	An optional argument to specify the tuning parameter(s) for the previous modelling method. If NULL (the default), the tuning parameter(s) is(are) estimated by cv-fold cross-validation. Otherwise, the user can propose a tuning grid. See details.
cv	The number of splits for cross-validation. The default value is 30.
boot.type	The type of bootstrap to use. Two types are possible: "bcv" for bootstrap cross-validation (by default) and "boot" usual bootstrap. See details.
boot.number	The number of bootstrap resamples. The default value is 500.
boot.tune	A logical value to determine whether the tuning parameter(s) should be estimated inside of each bootstrap iteration. See details.
progress	A logical value to print a progress bar. The default is TRUE
seed	A random seed to ensure reproducibility during the cv process. If NULL, a seed is randomly assigned.
boot.mi	A logical value to apply multiple imputation using the mice package. See details.
m	Number of imputations to perform if boot.mi is TRUE.
...	Additional arguments to be passed directly to the mice function for customizing the multiple imputation process (e.g., method, maxit, diagnostics).

Details

The option effect="ATE" corresponds to the Average Treatment effect on the Entire population, i.e. the marginal effect if the entire sample were treated versus untreated. The "ATT" modality allows the estimation of the Average Treatment effect on the Treated, i.e. the marginal effect if the treated subjects (group = 1) would have been untreated. The "ATU" modality allows the estimation of the Average Treatment effect on the Untreated, i.e. the marginal effect if the untreated subjects (group = 0) would have been treated.

Several modelling methods can be used for the Q-model estimation:

"all"	A proportional hazard regression, the baseline hazard function being estimated by using the Breslow estimator, all the covariates in the linear predictor.
"lasso"	The same PH model with L1 regularization allowing predictors' selection.
"ridge"	The same PH model with L2 regularization allowing highly correlated predictors.
"elasticnet"	The same PH model which combines both L1 and L2 regularizations.
"aic"	The same PH model with a AIC-based forward selection.
"bic"	The same PH model with a BIC-based forward selection.

The param.tune argument allows users to specify tuning parameters of penalized regression. If NULL (the default), the tuning parameters of each algorithm are estimated by cv-fold cross-validation and the default grid of the glmnet package is used. Otherwise, the user can propose a specific grid. For "lasso" and "ridge" penalizations, it should be a vector representing the lambda penalization parameter. For "elasticnet", it should be a list or a vector of length 2, containing lambda (penalization parameter) and alpha (mixing parameter between L1 and L2 regularizations) values. The alpha value typically ranges from 0 to 1. The user may propose a single value of each tuning parameter if she/he aims to define her/his own penalty.

The boot.tune argument is logical value which determines whether the tuning parameter should be estimated inside of each bootstrap iteration. If FALSE (the default), the tuning parameter will be estimated once on the complete dataset.

The boot.type argument controls how bootstrap estimates are computed. With boot.type = "boot", each iteration fits the Q-model on a sampled dataset and predicts on the same patients. With boot.type = "bcv", the Q-model is fitted on the sampled patients, and predictions are made on patients not included in that sample.

If boot.mi = TRUE, multiple imputation is considered by using the MI-BOOT approach proposed by Schomaker & Heumann (2018). The dataset is first imputed using the mice function to create m complete datasets. The bootstrap samples are then generated from each imputed dataset. The Q-model and the G-computation is performed for each m x boot.number samples, and all runs are finally concatenated.

Value

qmodel.fit	The fitted Q-model.
calibration	A list containing the time points (time) with the related cumulative hazards (cumhaz), survival rates (surv), and baseline cumulative hazard (H0.multi); the linear predictors (lp) obtained on the complete dataset; survival probabilities for group 0 (surv0); and survival probabilities for group 1 (surv1)
tuning.parameters	The estimated tuning parameters for the Q-model. For "aic" and "bic" methods, this represents the final model.
data	The data frame with individual with no missing data in the formula parameters.
formula	The formula provided by the user.
model	The method used for the Q-model (e.g., "lasso", "ridge", "aic").
cv	The number of splits used for cross-validation.

`penalty.factor` The penalty factors used for the penalized regression. The variable `group` has a penalty factor of 0 (no penalization) and 1 for the others.

`missing` The number of observations that were removed from the original dataset due to missing values.

`pro.time` The time point up to which RMST and survival probabilities are estimated.

`boot.number` The total number of bootstrap resamples.

`boot.type` The type of bootstrap.

`group` A character string specifying the name of the variable related to the exposure or treatment.

`n` The sample size of the dataset after missing data removal (if `boot.mi = FALSE`).

`nevent` The total number of events in the dataset.

`adjusted.results`
A data frame containing the adjusted results for each bootstrap sample, including: `AHR` (average hazard ratio), `RMST0` and `RMST1` (restricted mean survival times), `deltaRMST` (difference in RMST), `s0` and `s1` (survival probabilities at pro.time), and `delta` (difference in survival probabilities).

`unadjusted.results`
A data frame containing the unadjusted results for each bootstrap sample, including: `AHR` (average hazard ratio), `RMST0` and `RMST1` (restricted mean survival times), `deltaRMST` (difference in RMST), `s0` and `s1` (survival probabilities at pro.time), and `delta` (difference in survival probabilities).

`call` The complete function call that generated the `gctimes` object.

`m` The number of multiple imputations performed (only present if `mi.boot = TRUE`).

`initial.data` The original dataset provided by the user, before any imputation (only present if `mi.boot = TRUE`).

`nimput` The number of observations with missing values that were removed from the dataset prior to imputation (only present if `mi.boot = TRUE`).

`seed` The random seed used.

References

Chatton et al. G-computation and doubly robust standardisation for continuous-time data: A comparison with inverse probability weighting. Stat Methods Med Res. 31(4):706-718. 2022. <doi:10.1177/09622802211047345>.

Examples

```
data(dataPROPHYVAP)

.f <- formula(Surv(TIME_DEATH, DEATH) ~ GROUP + AGE +
              SEX + BMI + DIABETES)

### In practice use larger values for cv and boot.number
### We set boot.number and cv at 10 for speed in CRAN checks
### cv=30 and boot.number=1000 are more appropriate values
gc1 <- gc_times(formula=.f, model="lasso", data=dataPROPHYVAP,
```

```
                group="GROUP", cv=10, boot.type="bcv",
                boot.number=10, effect="ATE", progress=TRUE , pro.time=20,
                boot.tune=FALSE)

summary(gc1)
```

`plot.gcbinary` *S3 Method for Plotting a 'gcbinary' Object*

Description

Provides a calibration plot for an object returned by the function `gc_binary`. This method assesses how well the Q-model's predicted values align with the observed binary outcomes.

Usage

```
## S3 method for class 'gcbinary'
plot(x, n.groups=5, smooth=FALSE, ...)
```

Arguments

`x`	An object returned by the function `gc_binary`.
`n.groups`	An integer for the number of groups to divide the predicted means into. The default is 5. Note: If `n.groups` is set too high, some groups may have too few observations. An error can also occur if the Q-model is singular.
`smooth`	A logical value, by default `FALSE`. If `boot.mi` is TRUE, enables plotting a single smooth calibration curve averaged over all `m` values instead of separate curves.
`...`	Additional graphical parameters that can be passed to the underlying plot function.

Details

The function visualizes the calibration of the Q-model by dividing predicted probabilities into `n.groups` quantiles. For each group, the average predicted probability is plotted against the observed proportion of events, including 95% confidence intervals for the observed values. An identity line is provided for reference and perfect calibration is indicated when the observed points are directly along this line.

Value

No return value for this S3 method.

Examples

```
data("dataPROPHYVAP")

.f <- formula(VAP ~ GROUP * AGE + SEX + BMI + DIABETES)

### In practice use larger values for cv and boot.number
### We set boot.number and cv at 10 for speed in CRAN checks
### cv=30 and boot.number=1000 are more appropriate values
gc1 <- gc_binary(formula=.f, model="ridge", data=dataPROPHYVAP, group="GROUP",
                 param.tune=NULL, boot.type="bcv", cv=10, boot.number=10,
                 effect="ATE", progress=TRUE, boot.tune=TRUE)

### Plot the calibration curve
plot(gc1, n.groups=3, col="red3")
```

---

plot.gccontinuous *S3 Method for Plotting a 'gccontinuous' Object*

Description

Provides a calibration plot for an object returned by the function gc_continuous. This method assesses how well the Q-model's predicted values align with the observed continuous outcomes.

Usage

```
## S3 method for class 'gccontinuous'
plot(x, n.groups=5, smooth=FALSE, ...)
```

Arguments

x	An object returned by the function gc_continuous.
n.groups	An integer for the number of groups to divide the predicted means into. The default is 5. Note: If n.groups is set too high, some groups may have too few observations. An error can also occur if the Q-model is singular.
smooth	A logical value, by default FALSE. If boot.mi is TRUE, enables plotting a single smooth calibration curve averaged over all m values instead of separate curves.
...	Additional graphical parameters that can be passed to the underlying plot function.

Details

The function visualizes the calibration of the Q-model by dividing predicted probabilities into n.groups quantiles. For each group, the average predicted probability is plotted against the observed proportion of events, including 95% confidence intervals for the observed values. An identity line is provided for reference and perfect calibration is indicated when the observed points are directly along this line.

Value

No return value for this S3 method.

Examples

```
data("dataPROPHYVAP")

.f <- formula(VAP ~ GROUP * (AGE + SEX + ALCOHOL + BMI + DIABETES))

### In practice use larger values for cv and boot.number
### We set boot.number and cv at 10 for speed in CRAN checks
### cv=30 and boot.number=1000 are more appropriate values
gc1 <- gc_continuous(formula=.f, model="all", data=dataPROPHYVAP, group="GROUP",
                     cv=10, boot.type="boot", boot.number=10, boot.tune=TRUE,
                     effect="ATE", progress=TRUE, seed=5192)

### Plot the calibration curve
plot(gc1, n.groups=5, col="red3")
```

---

plot.gccount *S3 Method for Plotting a 'gccount' Object*

Description

Provides a calibration plot for an object returned by the function gc_count. This method assesses how well the Q-model's predicted values align with the observed counting outcomes.

Usage

```
## S3 method for class 'gccount'
plot(x, n.groups=5, smooth=FALSE, ...)
```

Arguments

x	An object returned by the function gc_count.
n.groups	An integer for the number of groups to divide the predicted means into. The default is 5. Note: If n.groups is set too high, some groups may have too few observations. An error can also occur if the Q-model is singular.
smooth	A logical value, by default FALSE. If boot.mi is TRUE, enables plotting a single smooth calibration curve averaged over all m values instead of separate curves.
...	Additional graphical parameters that can be passed to the underlying plot function.

Details

The function visualizes the calibration of the Q-model by dividing predicted probabilities into `n.groups` quantiles. For each group, the average predicted probability is plotted against the observed proportion of events, including 95% confidence intervals for the observed values. An identity line is provided for reference and perfect calibration is indicated when the observed points are directly along this line.

Value

No return value for this S3 method.

Examples

```
data("dataPROPHYVAP")

.f <- formula(VAP ~ GROUP * (AGE + SEX + ALCOHOL + BMI + DIABETES))

### In practice use larger values for cv and boot.number
### We set boot.number and cv at 10 for speed in CRAN checks
### cv=30 and boot.number=1000 are more appropriate values
gc1 <- gc_count(formula=.f, model="all", data=dataPROPHYVAP, group="GROUP",
                cv=10, boot.type="boot", boot.number=10, boot.tune=TRUE,
                effect="ATE", progress=TRUE, seed=5192)

### Plot the calibration curve
plot(gc1, n.groups=5, col="red3")
```

---

plot.gctimes *S3 Method for Plotting a 'gctimes' Object*

Description

Provides calibration and survival plots for an object returned by the function `gc_times`. This method assesses how well the Q-model's predicted survival aligns with observed time-to-event outcomes.

Usage

```
## S3 method for class 'gctimes'
plot(x, method="calibration", n.groups=5, pro.time=NULL, smooth=FALSE, ...)
```

Arguments

`x`	An object returned by the function `gc_times`.
`method`	A character string specifying the type of plot. Options are: `"calibration"` (default) for calibration at `pro.time`, `"calibration2"` for Kaplan-Meier survival vs. mean predicted survival, `"survival"` for predicted survival curves by treatment/exposure group.

n.groups	An integer for the number of quantiles to divide the predicted probabilities into. The default is 5. Note: If n.groups is set too high, some groups may have too few events to reliably estimate survivals. An error can also occur if the Q-model is singular.
pro.time	A numeric value specifying the time at which calibration or RMST is evaluated. Defaults to the pro.time used in gc_times.
smooth	A logical value, by default FALSE. If TRUE, applies smoothing to the calibration curves.
...	Additional graphical parameters that can be passed to the underlying plot functions.

Details

Methods:

- "calibration": Visualizes calibration at pro.time by dividing predicted probabilities into n.groups quantiles. For each group, the average predicted probability is plotted against the observed proportion of events at pro.time, including 95% confidence intervals. An identity line is provided for reference, and perfect calibration is indicated when the observed points lie directly along this line.
- "calibration2": Plots the Kaplan-Meier estimations of the overall survival against the mean of the survival predictions derived from the Q-model. Note that this method cannot be used when boot.mi=TRUE.
- "survival": Plots the predicted survival curves derived from the Q-model for each treatment/exposure group.

Survival predictions are computed using the linear predictor from the fitted model and the baseline cumulative hazard estimated via Breslow's method. Bootstrap iterations are used in gc_times to compute adjusted and unadjusted survival curves, RMST, and differences between groups.

Value

No return value for this S3 method.

Examples

```
data("dataPROPHYVAP")

.f <- formula(Surv(TIME_DEATH, DEATH) ~ GROUP * (AGE + SEX + ALCOHOL + BMI + DIABETES))

### In practice use larger values for cv and boot.number
### We set boot.number and cv at 10 for speed in CRAN checks
### cv=30 and boot.number=1000 are more appropriate values
gc1 <- gc_times(formula=.f, model="lasso", data=dataPROPHYVAP, group="GROUP",
                cv=10, boot.type="boot", boot.number=10, boot.tune=TRUE,
                effect="ATE", progress=TRUE, pro.time=20, seed=5192)

### Calibration plot at 20 days
plot(gc1, method="calibration", n.groups=5, pro.time=10)
```

```
### Kaplan-Meier vs mean predicted survival
plot(gc1, method="calibration2")

### Predicted survival curves for treatment groups
plot(gc1, method="survival", col=c("red3","blue3"))
legend("bottomleft", c("Placebo", "Ceftriaxone"), col=c("red3","blue3"), lty=1)
```

`print.gcbinary` *S3 Method for Printing an 'gcbinary' Object*

Description

Print a summary of an object returned from `gc_binary` function.

Usage

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

Arguments

`x`	An object returned by the function `gc_binary`.
`digits`	An optional integer for the number of digits to print when printing numeric values.
`...`	For future methods.

Value

No return value for this S3 method.

Examples

```
data(dataPROPHYVAP)

.f <- formula(VAP ~ GROUP * AGE + SEX + BMI + DIABETES)

### In practice use larger values for cv and boot.number
### We set boot.number and cv at 10 for speed in CRAN checks
gc1 <- gc_binary(formula=.f, model="ridge", data=dataPROPHYVAP,
              group="GROUP", param.tune=NULL, boot.type="bcv", cv=10,
              boot.number=10, effect="ATE", progress=TRUE, boot.tune=TRUE)

print(gc1)
```

print.gccontinuous *S3 Method for Printing an 'gccontinuous' Object*

Description

Print a summary of and object returned from gc_continuous function.

Usage

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

Arguments

x	An object returned by the function gc_continuous.
digits	An optional integer for the number of digits to print when printing numeric values.
...	For future methods.

Value

No return value for this S3 method.

Examples

```
data("dataPROPHYVAP")

.f <- formula(VAP ~ GROUP * (AGE + SEX + ALCOHOL + BMI + DIABETES))

### In practice use larger values of boot.number (e.g., 500)
### We set boot.number and cv at 10 for speed in CRAN checks
gc1 <- gc_continuous(formula=.f, model="all", data=dataPROPHYVAP, group="GROUP",
                     cv=10, boot.type="boot", boot.number=10, boot.tune=TRUE,
                     effect="ATE", progress=TRUE, seed=5192)

print(gc1)
```

print.gccount *S3 Method for Printing an 'gccontinuous' Object*

Description

Print a summary of and object returned from gc_count function.

Usage

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

Arguments

x	An object returned by the function gc_count.
digits	An optional integer for the number of digits to print when printing numeric values.
...	For future methods.

Value

No return value for this S3 method.

Examples

```
data("dataPROPHYVAP")

.f <- formula(VAP ~ GROUP * (AGE + SEX + ALCOHOL + BMI + DIABETES))

### In practice use larger values of boot.number (e.g., 500)
### We set boot.number and cv at 10 for speed in CRAN checks
gc1 <- gc_count(formula=.f, model="all", data=dataPROPHYVAP, group="GROUP",
                    cv=10, boot.type="boot", boot.number=10, boot.tune=TRUE,
                    effect="ATE", progress=TRUE, seed=5192)

print(gc1)
```

---

print.gctimes *S3 Method for Printing an 'gctimes' Object*

Description

Print a summary of the gctimes object returned from gc_times function

Usage

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

Arguments

x	An object returned by the function gc_times.
digits	An optional integer for the number of digits to print when printing numeric values.
...	For future methods.

Value

No return value for this S3 method.

Examples

```
data(dataPROPHYVAP)

.f <- formula(Surv(TIME_DEATH, DEATH) ~ GROUP * AGE + SEX +
              BMI + DIABETES)

### In practice use larger values for cv and boot.number
### We set boot.number and cv at 10 for speed in CRAN checks
gc1 <- gc_times(formula=.f, model="lasso", data=dataPROPHYVAP,
group="GROUP", param.tune=NULL, boot.type="bcv", cv=10, boot.number=10,
              effect="ATE", progress=TRUE , pro.time=10, boot.tune=FALSE)

print(gc1)
```

---

summary.gcbinary *S3 Method for Summarizing an 'gcbinary' Object*

Description

Summarize an object returned by the function gc_binary.

Usage

```
## S3 method for class 'gcbinary'
summary(object, digits=4, ci.type=NULL, ci.level=0.95,
                          unadjusted=TRUE, ...)
```

Arguments

object	An object returned by the function gc_binary.
digits	An optional integer for the number of digits to print when summarizing numeric values.
ci.type	The type of confidence intervals. Two types are possible: "norm" (assumed the Normal distribution) and "perc" (non-parametric estimation by percentiles).
ci.level	The confidence level required. Default is 95%.
unadjusted	A logical value to print the unadjusted results. The default is TRUE
...	For future methods.

Value

adjusted	The data frame of the G-computation summary results.
unadjusted	The data frame of the unadjusted summary results.
model	The method used for the Q-model.
formula	The formula provided by the user.
tuning.parameters	The estimated tuning parameters for the Q-model. For "aic" and "bic" methods, this represents the final model.
n	The sample size of the dataset after missing data removal (if boot.mi = FALSE).
nevent	The total number of events in the dataset.
nimput	The number of imputations to perform if boot.mi is TRUE.
missing	The number of observations that were removed from the original dataset due to missing values.
m	The number of multiple imputations performed (only present if mi.boot = TRUE).
digits	The number of digits to print.
unadjusted.flag	A logical value to indicate if the unadjusted results are provided.

Examples

```
data(dataPROPHYVAP)

.f <- formula(VAP ~ GROUP * AGE + SEX + BMI + DIABETES)

### In practice use larger values for cv and boot.number
### We set boot.number and cv at 10 for speed in CRAN checks
### cv=30 and boot.number=1000 are more appropriate values
gc1 <- gc_binary(formula=.f, model="ridge", data=dataPROPHYVAP,
              group="GROUP", param.tune=NULL, boot.type="bcv", cv=10,
              boot.number=10, effect="ATE", progress=TRUE, boot.tune=TRUE)

summary(gc1, ci.type="norm", ci.level=0.95)
```

---

summary.gccontinuous *S3 Method for Summarizing an 'gccontinuous' Object*

Description

Summarize an object returned by the function gc_continuous.

Usage

```
## S3 method for class 'gccontinuous'
summary(object, digits=4, ci.type=NULL, ci.level=0.95,
                          unadjusted=TRUE, ...)
```

Arguments

object	An object returned by the function gc_continuous.
digits	An optional integer for the number of digits to print when summarizing numeric values.
ci.type	The type of confidence intervals. Two types are possible: "norm" (assumed the Normal distribution) and "perc" (non-parametric estimation by percentiles).
ci.level	The confidence level required. Default is 95%.
unadjusted	A logical value to print the unadjusted results. The default is TRUE
...	For future methods.

Value

adjusted	The data frame of the G-computation summary results.
unadjusted	The data frame of the unadjusted summary results.
model	The method used for the Q-model.
formula	The formula provided by the user.
tuning.parameters	The estimated tuning parameters for the Q-model. For "aic" and "bic" methods, this represents the final model.
n	The sample size of the dataset after missing data removal (if boot.mi = FALSE).
nimput	The number of imputations to perform if boot.mi is TRUE.
missing	The number of observations that were removed from the original dataset due to missing values.
m	The number of multiple imputations performed (only present if mi.boot = TRUE).
digits	The number of digits to print.
unadjusted.flag	A logical value to indicate if the unadjusted results are provided.

Examples

```
data("dataPROPHYVAP")

.f <- formula(VAP ~ GROUP * (AGE + SEX + ALCOHOL + BMI + DIABETES))

### In practice use larger values for cv and boot.number
### We set boot.number and cv at 10 for speed in CRAN checks
### cv=30 and boot.number=1000 are more appropriate values
gc1 <- gc_continuous(formula=.f, model="all", data=dataPROPHYVAP, group="GROUP",
                     cv=10, boot.type="boot", boot.number=10, boot.tune=TRUE,
                     effect="ATE")

summary(gc1, ci.type="norm", ci.level=0.99)
```

summary.gccount *S3 Method for Summarizing an 'gccount' Object*

Description

Summarize an object returned by the function gc_count.

Usage

```
## S3 method for class 'gccount'
summary(object, digits=4, ci.type=NULL, ci.level=0.95,
                          unadjusted=TRUE, ...)
```

Arguments

object	An object returned by the function gc_count.
digits	An optional integer for the number of digits to print when summarizing numeric values.
ci.type	The type of confidence intervals. Two types are possible: "norm" (assumed the Normal distribution) and "perc" (non-parametric estimation by percentiles).
ci.level	The confidence level required. Default is 95%.
unadjusted	A logical value to print the unadjusted results. The default is TRUE
...	For future methods.

Value

adjusted	The data frame of the G-computation summary results.
unadjusted	The data frame of the unadjusted summary results.
model	The method used for the Q-model.
formula	The formula provided by the user.
tuning.parameters	The estimated tuning parameters for the Q-model. For "aic" and "bic" methods, this represents the final model.
n	The sample size of the dataset after missing data removal (if boot.mi = FALSE).
nimput	The number of imputations to perform if boot.mi is TRUE.
missing	The number of observations that were removed from the original dataset due to missing values.
m	The number of multiple imputations performed (only present if mi.boot = TRUE).
digits	The number of digits to print.
unadjusted.flag	A logical value to indicate if the unadjusted results are provided.

Examples

```
data("dataPROPHYVAP")

.f <- formula(VAP ~ GROUP * (AGE + SEX + ALCOHOL + BMI + DIABETES))

### In practice use larger values for cv and boot.number
### We set boot.number and cv at 10 for speed in CRAN checks
### cv=30 and boot.number=1000 are more appropriate values
gc1 <- gc_count(formula=.f, model="all", data=dataPROPHYVAP, group="GROUP",
                    cv=10, boot.type="boot", boot.number=10, boot.tune=TRUE,
                    effect="ATE")

summary(gc1, ci.type="norm", ci.level=0.99)
```

`summary.gctimes` *S3 Method for Summarizing an 'gctimes' Object*

Description

Summarize an object returned by the function `gc_times`.

Usage

```
## S3 method for class 'gctimes'
summary(object, digits=4, ci.type=NULL, ci.level=0.95,
                        unadjusted=TRUE, ...)
```

Arguments

`object`	An object returned by the function `gc_times`.
`digits`	An optional integer for the number of digits to print when summarizing numeric values.
`ci.type`	The type of confidence intervals. Two types are possible: "norm" (assumed the Normal distribution) and "perc" (non-parametric estimation by percentiles).
`ci.level`	The confidence level required. Default is 95%.
`unadjusted`	A logical value to print the unadjusted results. The default is `TRUE`
`...`	For future methods.

Value

`adjusted`	The data frame of the G-computation summary results.
`unadjusted`	The data frame of the unadjusted summary results.
`model`	The method used for the Q-model.
`formula`	The formula provided by the user.

`tuning.parameters`
: The estimated tuning parameters for the Q-model. For "aic" and "bic" methods, this represents the final model.

`n`
: The sample size of the dataset after missing data removal (if `boot.mi = FALSE`).

`nevent`
: The total number of events in the dataset.

`nimput`
: The number of imputations to perform if boot.mi is `TRUE`.

`missing`
: The number of observations that were removed from the original dataset due to missing values.

`m`
: The number of multiple imputations performed (only present if `mi.boot = TRUE`).

`pro.time`
: The time point of interest for the evaluation.

`digits`
: The number of digits to print.

`unadjusted.flag`
: A logical value to indicate if the unadjusted results are provided.

Examples

```
data(dataPROPHYVAP)

formula <- formula(Surv(TIME_DEATH, DEATH) ~ GROUP * AGE +
              SEX + BMI + DIABETES)

### In practice use larger values for cv and boot.number
### We set boot.number and cv at 10 for speed in CRAN checks
### cv=30 and boot.number=1000 are more appropriate values
gc1 <- gc_times(formula=formula, model="lasso", data=dataPROPHYVAP,
              group="GROUP", param.tune=NULL, boot.type="bcv", cv=10,
              boot.number=10, effect="ATE", pro.time=20, boot.tune=TRUE)

summary(gc1, ci.type="norm", ci.level=0.99)
```

---

transport *Transport the Marginal Effect for Another Populations*

Description

Applies an already fitted Q-model to estimate marginal effects for a new population (i.e., with differentially distributed baseline characteristics used as predictors in the Q-Model).

Usage

```
transport(object, newdata, n.sim=500, seed=NULL)
```

Arguments

object	An object returned by the function gc_binary, gc_continuous, gc_count or gc_times.
newdata	The new data frame with all the covariates in the formula of the object.
n.sim	The number of Monte Carlo simulations to perform for estimating standard errors and confidence intervals of the marginal effects on the newdata. This is only applicable for unpenalized models. The default value is 500.
seed	A random seed to ensure reproducibility during the simulation process. If NULL (the default), a seed is randomly assigned.

Details

The function employs two distinct estimation strategies:

- **Monte Carlo Simulation:** For unpenalized models (likelihood-based models like "all", "aic", or "bic"), the function uses the variance-covariance matrix of the Q-model regression coefficients to simulate Q-models assuming a multivariate normal distribution. This provides a distribution of adjusted results and the related confidence intervals.
- **Single Point Estimate:** For penalized models ("lasso", "ridge", "elasticnet") and survival models (gctimes), the function does not estimate the distribution of adjusted results and the related confidence intervals.

Value

The function returns an object of the same class with the same arguments as the input object, but with updated results for the newdata.

References

Pearl J, Bareinboim E. External validity: From do-calculus to transportability across populations. Statistical Science. 2012;29(4):579-595.

Examples

```
### For a binary outcome
data("dataPROPHYVAP")

.f <- formula(VAP ~ GROUP * AGE + SEX + BMI + DIABETES)

### In practice use larger values of boot.number (e.g., 500)
### We set boot.number and cv at 10 for speed in CRAN checks
gc1 <- gc_binary(formula=.f, model="all", data=dataPROPHYVAP,
              group="GROUP", param.tune=NULL, boot.type="bcv", cv=10,
              boot.number=10, effect="ATE", progress=TRUE, boot.tune=FALSE)

summary(gc1, ci.type="norm", ci.level=0.95)

### New dataset (for example a subset of younger patients)
newdata <- subset(dataPROPHYVAP, AGE <= 50)
```

```
### transport the gc_binary model to the new dataset
gc2 <- transport(object=gc1, newdata=newdata, n.sim=10)

summary(gc2, ci.type="perc", ci.level=0.95)
```

Index

∗ **Binary outcome**
  gc_binary, 5
  plot.gcbinary, 18
  print.gcbinary, 23
  summary.gcbinary, 26
∗ **Calibration**
  plot.gctimes, 21
∗ **Continuous outcome**
  gc_continuous, 8
  plot.gccontinuous, 19
  summary.gccontinuous, 27
  summary.gccount, 29
∗ **Counting outcome**
  gc_count, 11
  plot.gccount, 20
∗ **Dataset**
  dataCOHORT, 2
  dataPROPHYVAP, 3
∗ **G-computation**
  gc_binary, 5
  gc_continuous, 8
  gc_count, 11
  gc_times, 14
∗ **Plotting**
  plot.gcbinary, 18
  plot.gccontinuous, 19
  plot.gccount, 20
  plot.gctimes, 21
∗ **Printing**
  print.gcbinary, 23
  print.gccontinuous, 24
  print.gccount, 24
  print.gctimes, 25
∗ **Summarizing**
  summary.gcbinary, 26
  summary.gccontinuous, 27
  summary.gccount, 29
  summary.gctimes, 30
∗ **Survival analysis**
  plot.gctimes, 21
∗ **Time-to-event outcome**
  gc_times, 14
  plot.gctimes, 21
  print.gctimes, 25
  summary.gctimes, 30
∗ **Transportation**
  transport, 31

dataCOHORT, 2
dataPROPHYVAP, *2*, 3

gc_binary, 5, *32*
gc_continuous, 8, *32*
gc_count, 11, *32*
gc_times, 14, *32*

mice, *5, 6*, *9, 10*, *12, 13*, *15, 16*

plot.gcbinary, 18
plot.gccontinuous, 19
plot.gccount, 20
plot.gctimes, 21
print.gcbinary, 23
print.gccontinuous, 24
print.gccount, 24
print.gctimes, 25
print.summary.gcbinary
  *(*summary.gcbinary*)*, 26
print.summary.gccontinuous
  *(*summary.gccontinuous*)*, 27
print.summary.gccount
  *(*summary.gccount*)*, 29
print.summary.gctimes
  *(*summary.gctimes*)*, 30

summary.gcbinary, 26
summary.gccontinuous, 27
summary.gccount, 29
summary.gctimes, 30

transport, 31