Package: gfoRmula (via r-universe)

September 16, 2024

Title Parametric G-Formula

Version 1.1.0

Description Implements the non-iterative conditional expectation (NICE) algorithm of the g-formula algorithm (Robins (1986) <doi:10.1016/0270-0255(86)90088-6>, Hernán and Robins (2024, ISBN:9781420076165)). The g-formula can estimate an outcome's counterfactual mean or risk under hypothetical treatment strategies (interventions) when there is sufficient information on time-varying treatments and confounders. This package can be used for discrete or continuous time-varying treatments and for failure time outcomes or continuous/binary end of follow-up outcomes. The package can handle a random measurement/visit process and a priori knowledge of the data structure, as well as censoring (e.g., by loss to follow-up) and two options for handling competing events for failure time outcomes. Interventions can be flexibly specified, both as interventions on a single treatment or as joint interventions on multiple treatments. See McGrath et al. (2020) <doi:10.1016/j.patter.2020.100008> for a guide on how to use the package.

Depends R (>= 3.5.0)

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

Imports data.table, ggplot2, ggpubr, grDevices, nnet, parallel, progress, stats, stringr, survival, truncnorm, truncreg, utils

Suggests Hmisc, knitr, randomForest, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

URL https://github.com/CausalInference/gfoRmula, https://doi.org/10.1016/j.patter.2020.100008

BugReports https://github.com/CausalInference/gfoRmula/issues

Config/testthat/edition 3

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

RemoteUrl https://github.com/causalinference/gformula

RemoteRef HEAD

RemoteSha d789dffd61efb536891621affa8ecd8100ae990e

Contents

`basicdata` *Example Dataset for a Survival Outcome with Censoring*

Description

A dataset consisting of 11,332 observations on 2,500 individuals over 7 time points. Each row in the dataset corresponds to the record of one individual at one time point. Individuals who are censored at time $k + 1$ only have a total of $k + 1$ records, which correspond to time indices $0, ..., k$.

Usage

```
basicdata
```

Format

A data table with 11,332 rows and 8 variables:

t0 Time index.

id Unique identifier for each individual.

L1 Binary time-varying covariate.

L2 Continuous time-varying covariate.

L3 Continuous baseline covariate. For each individual, the baseline values are repeated at each time point.

A Binary treatment variable.

D Competing event; time-varying indicator of failure.

Y Outcome of interest; time-varying indicator of failure.

`basicdata_nocomp` *Example Dataset for a Survival Outcome without Censoring*

Description

A dataset consisting of 13,170 observations on 2,500 individuals over 7 time points. Each row in the dataset corresponds to the record of one individual at one time point.

Usage

```
basicdata_nocomp
```

Format

A data table with 13,170 rows and 7 variables:

t0 Time index.

id Unique identifier for each individual.

L1 Binary covariate.

L2 Continuous covariate.

L3 Continuous baseline covariate. For each individual, the baseline values are repeated at each time point.

A Binary treatment variable.

Y Outcome of interest; time-varying indicator of failure.

binary_eofdata *Example Dataset for a Binary Outcome at End of Follow-Up*

Description

A dataset consisting of 17,500 observations on 2,500 individuals over 7 time points. Each row in the dataset corresponds to the record of one individual at one time point.

Usage

```
binary_eofdata
```

Format

A data table with 17,500 rows and 7 variables:

time Time index.

id_num Unique identifier for each individual.

cov1 Binary time-varying covariate.

cov2 Continuous time-varying covariate.

cov3 Continuous baseline covariate. For each individual, the baseline values are repeated at each time point.

treat Binary treatment variable.

outcome Binary outcome of interest. Because this outcome is only defined at the end of follow-up, values of NA are given in all other time points.

carry_forward *Carry Forward*

Description

This function assists the implemention of a restriction on a covariate in the date table newdf. A particular covariate is simulated only when some condition (usually a covariate representing whether a doctor's visit occurred or not) is TRUE. If the condition is FALSE, the covariate value is not simulated for that time point and the value is instead carried over from the previous time point.

Usage

```
carry_forward(newdf, pool, restriction, time_name, t, int_visit_type, intvar)
```

Arguments

newdf	Data table containing the simulated data at time t.
pool	Data table containing the simulated data at times before t.
restriction	List of vectors. Each vector contains as its first entry the covariate affected by the restriction; its second entry the condition that must be TRUE for the covariate to be modeled; its third entry a function that executes other specific actions based on the condition (in this case, this function); and its fourth entry some value used by the function (in this case, this entry is not used).
time_name	Character string specifying the name of the time variable in pool and newdf.
t	Integer specifying the current time index.
int_visit_type	Logical scalar specifying whether to carry forward the intervened value (rather than the natural value) of the treatment variables(s) when performing a carry forward restriction type
intvar	A vector specifying the name(s) of the variable(s) to be intervened on.

Value

No value is returned. The data table newdf is modified in place.

Examples

```
## Estimating the effect of static treatment strategies on risk of a
## failure event

id <- 'id'
time_points <- 7
time_name <- 't0'
covnames <- c('L1', 'L2', 'A')
outcome_name <- 'Y'
outcome_type <- 'survival'
covtypes <- c('binary', 'bounded normal', 'binary')
histories <- c(lagged, lagavg)
histvars <- list(c('A', 'L1', 'L2'), c('L1', 'L2'))
covparams <- list(covmodels = c(L1 ~ lag1_A + lag_cumavg1_L1 + lag_cumavg1_L2 +
                                  L3 + t0,
                                L2 ~ lag1_A + L1 + lag_cumavg1_L1 +
                                  lag_cumavg1_L2 + L3 + t0,
                                A ~ lag1_A + L1 + L2 + lag_cumavg1_L1 +
                                  lag_cumavg1_L2 + L3 + t0))
ymodel <- Y ~ A + L1 + L2 + L3 + lag1_A + lag1_L1 + lag1_L2 + t0
intervention1.A <- list(static, rep(0, time_points))
intervention2.A <- list(static, rep(1, time_points))
int_descript <- c('Never treat', 'Always treat')
nsimul <- 10000

# At t0 == 5, assign L1 its value at the previous time point
restrictions <- list(c('L2', 't0 != 5', carry_forward))

gform_basic <- gformula(obs_data = basicdata_nocomp, id = id,
```

```
                        time_points = time_points,
                        time_name = time_name, covnames = covnames,
                        outcome_name = outcome_name,
                        outcome_type = outcome_type, covtypes = covtypes,
                        covparams = covparams, ymodel = ymodel,
                        intervention1.A = intervention1.A,
                        intervention2.A = intervention2.A,
                        int_descript = int_descript,
                        restrictions = restrictions,
                        histories = histories, histvars = histvars,
                        basecovs = c('L3'), nsimul = nsimul,
                        seed = 1234)
gform_basic
```

`censor_data` *Example Dataset for a Survival Outcome with an Indicator of Censoring Variable*

Description

A dataset consisting of 118,725 observations on 20,000 individuals over 10 time points. Each row in the dataset corresponds to the record of one individual at one time point.

Usage

```
censor_data
```

Format

A data table with 22,500 rows and 7 variables:

t0 Time index.

id Unique identifier for each individual.

L Binary time-varying covariate.

A Continuous treatment variable.

C Censoring indicator.

Y Outcome of interest; time-varying indicator of failure.

coef.gformula *Coefficient method for objects of class "gformula"*

Description

This function extracts the coefficients of the fitted models for the time-varying covariates, outcome, and compevent event (if applicable).

Usage

```
## S3 method for class 'gformula'
coef(object, ...)
```

Arguments

object	Object of class "gformula".
...	Other arguments.

Value

If bootdiag was set to FALSE in gformula, this function returns a list of the coefficients of the fitted models to the observed data set. If bootstrapping was used and bootdiag was set to TRUE in gformula, this function returns a list described as follows. The first element (named 'Original sample') is a list of the coefficients of the fitted models to the observed data set. The kth element (named 'Bootstrap sample k-1') is a list of the coefficients of the fitted models corresponding to the k-1th bootstrap sample.

See Also

gformula

Examples

```
## Estimating the effect of static treatment strategies on risk of a
## failure event

id <- 'id'
time_points <- 7
time_name <- 't0'
covnames <- c('L1', 'L2', 'A')
outcome_name <- 'Y'
outcome_type <- 'survival'
covtypes <- c('binary', 'bounded normal', 'binary')
histories <- c(lagged, lagavg)
histvars <- list(c('A', 'L1', 'L2'), c('L1', 'L2'))
covparams <- list(covmodels = c(L1 ~ lag1_A + lag_cumavg1_L1 + lag_cumavg1_L2 +
                                  L3 + t0,
                                L2 ~ lag1_A + L1 + lag_cumavg1_L1 +
```

```
                                lag_cumavg1_L2 + L3 + t0,
                              A ~ lag1_A + L1 + L2 + lag_cumavg1_L1 +
                                lag_cumavg1_L2 + L3 + t0))
ymodel <- Y ~ A + L1 + L2 + L3 + lag1_A + lag1_L1 + lag1_L2 + t0
intervention1.A <- list(static, rep(0, time_points))
intervention2.A <- list(static, rep(1, time_points))
int_descript <- c('Never treat', 'Always treat')
nsimul <- 10000

gform_basic <- gformula(obs_data = basicdata_nocomp, id = id,
                        time_points = time_points,
                        time_name = time_name, covnames = covnames,
                        outcome_name = outcome_name,
                        outcome_type = outcome_type, covtypes = covtypes,
                        covparams = covparams, ymodel = ymodel,
                        intervention1.A = intervention1.A,
                        intervention2.A = intervention2.A,
                        int_descript = int_descript,
                        histories = histories, histvars = histvars,
                        basecovs = c('L3'), nsimul = nsimul,
                        seed = 1234)
coef(gform_basic)
```

`continuous_eofdata` *Example Dataset for a Continuous Outcome at End of Follow-Up*

Description

A dataset consisting of 17,500 observations on 2,500 individuals over 7 time points. Each row in the dataset corresponds to the record of one individual at one time point.

Usage

```
continuous_eofdata
```

Format

A data table with 17,500 rows and 7 variables:

t0 Time index.

id Unique identifier for each individual.

L1 Categorical time-varying covariate.

L2 Continuous time-varying covariate.

L3 Continuous baseline covariate. For each individual, the baseline values are repeated at each time point.

A Binary treatment variable.

Y Continuous outcome of interest. Because this outcome is only defined at the end of follow-up, values of NA are given in all other time points.

`continuous_eofdata_pb` *Example Dataset for a Continuous Outcome at End of Follow-Up with Pre-Baseline Times*

Description

A dataset consisting of 22,500 observations on 2,500 individuals over 2 pre-baseline time points and follow-up 7 time points. Each row in the dataset corresponds to the record of one individual at one time point.

Usage

```
continuous_eofdata_pb
```

Format

A data table with 22,500 rows and 7 variables:

t0 Time index.

id Unique identifier for each individual.

L1 Categorical time-varying covariate.

L2 Continuous time-varying covariate.

L3 Continuous baseline covariate. For each individual, the baseline values are repeated at each time point.

A Binary treatment variable.

Y Continuous outcome of interest. Because this outcome is only defined at the end of follow-up, values of NA are given in all other time points.

`gformula` *Estimation of Survival Outcome, Continuous End-of-Follow-Up Outcome, or Binary End-of-Follow-Up Outcome Under the Parametric G-Formula*

Description

Based on an observed data set, this function estimates the risk over time (for survival outcomes), outcome mean at end-of-follow-up (for continuous end-of-follow-up outcomes), or outcome probability at end-of-follow-up (for binary end-of-follow-up outcomes) under multiple user-specified interventions using the parametric g-formula. See McGrath et al. (2020) for further details concerning the application and implementation of the parametric g-formula.

Usage

```
gformula(
  obs_data,
  id,
  time_points = NULL,
  time_name,
  covnames,
  covtypes,
  covparams,
  covfits_custom = NA,
  covpredict_custom = NA,
  histvars = NULL,
  histories = NA,
  basecovs = NA,
  outcome_name,
  outcome_type,
  ymodel,
  ymodel_fit_custom = NULL,
  ymodel_predict_custom = NULL,
  compevent_name = NULL,
  compevent_model = NA,
  compevent_cens = FALSE,
  censor_name = NULL,
  censor_model = NA,
  intvars = NULL,
  interventions = NULL,
  int_times = NULL,
  int_descript = NULL,
  ref_int = 0,
  intcomp = NA,
  visitprocess = NA,
  restrictions = NA,
  yrestrictions = NA,
  compevent_restrictions = NA,
  baselags = FALSE,
  nsimul = NA,
  sim_data_b = FALSE,
  seed,
  nsamples = 0,
  parallel = FALSE,
  ncores = NA,
  ci_method = "percentile",
  threads,
  model_fits = FALSE,
  boot_diag = FALSE,
  show_progress = TRUE,
  ipw_cutoff_quantile = NULL,
  ipw_cutoff_value = NULL,
```

```
  int_visit_type = NULL,
  sim_trunc = TRUE,
  ...
)
```

Arguments

obs_data	Data table containing the observed data.
id	Character string specifying the name of the ID variable in obs_data.
time_points	Number of time points to simulate. By default, this argument is set equal to the maximum number of records that obs_data contains for any individual plus 1.
time_name	Character string specifying the name of the time variable in obs_data.
covnames	Vector of character strings specifying the names of the time-varying covariates in obs_data.
covtypes	Vector of character strings specifying the "type" of each time-varying covariate included in covnames. The possible "types" are: "binary", "normal", "categorical", "bounded normal", "zero-inflated normal", "truncated normal", "absorbing", "categorical time", and "custom".
covparams	List of vectors, where each vector contains information for one parameter used in the modeling of the time-varying covariates (e.g., model statement, family, link function, etc.). Each vector must be the same length as covnames and in the same order. If a parameter is not required for a certain covariate, it should be set to NA at that index.
covfits_custom	Vector containing custom fit functions for time-varying covariates that do not fall within the pre-defined covariate types. It should be in the same order covnames. If a custom fit function is not required for a particular covariate (e.g., if the first covariate is of type "binary" but the second is of type "custom"), then that index should be set to NA. The default is NA.
covpredict_custom	Vector containing custom prediction functions for time-varying covariates that do not fall within the pre-defined covariate types. It should be in the same order as covnames. If a custom prediction function is not required for a particular covariate, then that index should be set to NA. The default is NA.
histvars	List of vectors. The kth vector specifies the names of the variables for which the kth history function in histories is to be applied.
histories	Vector of history functions to apply to the variables specified in histvars. The default is NA.
basecovs	Vector of character strings specifying the names of baseline covariates in obs_data. These covariates are not simulated using a model but rather carry their value over all time points from the first time point of obs_data. These covariates should not be included in covnames. The default is NA.
outcome_name	Character string specifying the name of the outcome variable in obs_data.
outcome_type	Character string specifying the "type" of outcome. The possible "types" are: "survival", "continuous_eof", and "binary_eof".
ymodel	Model statement for the outcome variable.

ymodel_fit_custom
: Function specifying a custom outcome model. See the vignette "Using Custom Outcome Models in gfoRmula" for details. The default is NULL.

ymodel_predict_custom
: Function obtaining predictions from the custom outcome model specified in ymodel_fit_custom. See the vignette "Using Custom Outcome Models in gfoRmula" for details. The default is NULL.

compevent_name
: Character string specifying the name of the competing event variable in obs_data. Only applicable for survival outcomes.

compevent_model
: Model statement for the competing event variable. The default is NA. Only applicable for survival outcomes.

compevent_cens
: Logical scalar indicating whether to treat competing events as censoring events. This argument is only applicable for survival outcomes and when a competing even model is supplied (i.e., compevent_name and compevent_model are specified). If this argument is set to TRUE, the competing event model will only be used to construct inverse probability weights to estimate the natural course means / risk from the observed data. If this argument is set to FALSE, the competing event model will be used in the parametric g-formula estimates of the risk and will not be used to construct inverse probability weights. See "Details". The default is FALSE.

censor_name
: Character string specifying the name of the censoring variable in obs_data. Only applicable when using inverse probability weights to estimate the natural course means / risk from the observed data. See "Details".

censor_model
: Model statement for the censoring variable. Only applicable when using inverse probability weights to estimate the natural course means / risk from the observed data. See "Details".

intvars
: (Deprecated. See the ... argument) List, whose elements are vectors of character strings. The kth vector in intvars specifies the name(s) of the variable(s) to be intervened on in each round of the simulation under the kth intervention in interventions.

interventions
: (Deprecated. See the ... argument) List, whose elements are lists of vectors. Each list in interventions specifies a unique intervention on the relevant variable(s) in intvars. Each vector contains a function implementing a particular intervention on a single variable, optionally followed by one or more "intervention values" (i.e., integers used to specify the treatment regime).

int_times
: (Deprecated. See the ... argument) List, whose elements are lists of vectors. The kth list in int_times corresponds to the kth intervention in interventions. Each vector specifies the time points in which the relevant intervention is applied on the corresponding variable in intvars. When an intervention is not applied, the simulated natural course value is used. By default, this argument is set so that all interventions are applied in all time points.

int_descript
: Vector of character strings, each describing an intervention. It must be in same order as the specified interventions (see the ... argument).

ref_int
: Integer denoting the intervention to be used as the reference for calculating the risk ratio and risk difference. 0 denotes the natural course, while subsequent

integers denote user-specified interventions in the order that they are named (see the ... argument). The default is 0.

`intcomp` List of two numbers indicating a pair of interventions to be compared by a hazard ratio. The default is `NA`, resulting in no hazard ratio calculation.

`visitprocess` List of vectors. Each vector contains as its first entry the covariate name of a visit process; its second entry the name of a covariate whose modeling depends on the visit process; and its third entry the maximum number of consecutive visits that can be missed before an individual is censored. The default is `NA`.

`restrictions` List of vectors. Each vector contains as its first entry a covariate for which *a priori* knowledge of its distribution is available; its second entry a condition under which no knowledge of its distribution is available and that must be `TRUE` for the distribution of that covariate given that condition to be estimated via a parametric model or other fitting procedure; its third entry a function for estimating the distribution of that covariate given the condition in the second entry is false such that *a priori* knowledge of the covariate distribution is available; and its fourth entry a value used by the function in the third entry. The default is `NA`.

`yrestrictions` List of vectors. Each vector contains as its first entry a condition and its second entry an integer. When the condition is `TRUE`, the outcome variable is simulated according to the fitted model; when the condition is `FALSE`, the outcome variable takes on the value in the second entry. The default is `NA`.

`compevent_restrictions`
List of vectors. Each vector contains as its first entry a condition and its second entry an integer. When the condition is `TRUE`, the competing event variable is simulated according to the fitted model; when the condition is `FALSE`, the competing event variable takes on the value in the second entry. The default is `NA`. Only applicable for survival outcomes.

`baselags` Logical scalar for specifying the convention used for lagi and lag_cumavgi terms in the model statements when pre-baseline times are not included in `obs_data` and when the current time index, t, is such that $t < i$. If this argument is set to `FALSE`, the value of all lagi and lag_cumavgi terms in this context are set to 0 (for non-categorical covariates) or the reference level (for categorical covariates). If this argument is set to `TRUE`, the value of lagi and lag_cumavgi terms are set to their values at time 0. The default is `FALSE`.

`nsimul` Number of subjects for whom to simulate data. By default, this argument is set equal to the number of subjects in `obs_data`.

`sim_data_b` Logical scalar indicating whether to return the simulated data set. If bootstrap samples are used (i.e., `nsamples` is set to a value greater than 0), this argument must be set to `FALSE`. The default is `FALSE`.

`seed` Starting seed for simulations and bootstrapping.

`nsamples` Integer specifying the number of bootstrap samples to generate. The default is 0.

`parallel` Logical scalar indicating whether to parallelize simulations of different interventions to multiple cores.

`ncores` Integer specifying the number of CPU cores to use in parallel simulation. This argument is required when parallel is set to `TRUE`. In many applications, users may wish to set this argument equal to `parallel::detectCores() - 1`.

`ci_method` Character string specifying the method for calculating the bootstrap 95% confidence intervals, if applicable. The options are `"percentile"` and `"normal"`.

`threads` Integer specifying the number of threads to be used in `data.table`. See `setDTthreads` for further details.

`model_fits` Logical scalar indicating whether to return the fitted models. Note that if this argument is set to `TRUE`, the output of this function may use a lot of memory. The default is `FALSE`.

`boot_diag` Logical scalar indicating whether to return the parametric g-formula estimates as well as the coefficients, standard errors, and variance-covariance matrices of the parameters of the fitted models in the bootstrap samples. The default is `FALSE`.

`show_progress` Logical scalar indicating whether to print a progress bar for the number of bootstrap samples completed in the R console. This argument is only applicable when `parallel` is set to `FALSE` and bootstrap samples are used (i.e., `nsamples` is set to a value greater than 0). The default is `TRUE`.

`ipw_cutoff_quantile`
Percentile by which to truncate inverse probability weights. The default is `NULL` (i.e., no truncation). See "Details".

`ipw_cutoff_value`
Cutoff value by which to truncate inverse probability weights. The default is `NULL` (i.e., no truncation). See "Details".

`int_visit_type` Vector of logicals. The kth element is a logical specifying whether to carry forward the intervened value (rather than the natural value) of the treatment variables(s) when performing a carry forward restriction type for the kth intervention in `interventions`. When the kth element is set to `FALSE`, the natural value of the treatment variable(s) in the kth intervention in `interventions` will be carried forward. By default, this argument is set so that the intervened value of the treatment variable(s) is carried forward for all interventions.

`sim_trunc` Logical scalar indicating whether to truncate simulated covariates to their range in the observed data set. This argument is only applicable for covariates of type `"normal"`, `"bounded normal"`, `"truncated normal"`, and `"zero-inflated normal"`. The default is `TRUE`.

`...` Other arguments, including (a) those that specify the interventions and (b) those that are passed to the functions in `covpredict_custom`. To specify interventions, users can supply arguments with the following naming requirements

- Each intervention argument begins with a prefix of `intervention`.
- After the prefix, the intervention number is specified and followed by a period.
- After the period, the treatment variable name is specified.

Each intervention argument takes as input a list with the following elements:

- The first element specifies the intervention function.
- The subsequent elements specify any intervention values.
- (Optional) The named element `int_times` specifies the time points to apply the intervention. By default, all interventions are applied at all time points.

For example, an "always treat" intervention on A is given by
`intervention1.A = list(static, rep(1, time_points))`
See the vignette "A Simplified Approach for Specifying Interventions in gfoRmula" and "Examples" section for more examples.

Details

To assess model misspecification in the parametric g-formula, users can obtain inverse probability (IP) weighted estimates of the natural course risk and/or means of the time-varying covariates from the observed data. See Chiu et al. (2023) for details. In addition to the general requirements described in McGrath et al. (2020), the requirements for the input data set and the call to the gformula function for such analyses are described below.

Users need to include a column in `obs_data` with a time-varying censoring variable. Users need to indicate the name of the censoring variable and a model statement for the censoring variable with parameters `censor_name` and `censor_model`, respectively. When competing events are present, users need to include a column in `obs_data` with a time-varying indicator of the competing event variable and need to indicate the name of the competing event variable and the corresponding model statement with parameters `compevent_name` and `compevent_model`, respectively. Users need to indicate whether to treat competing events as censoring events with the `compevent_cens` parameter. Finally, users can specify how to truncate IP weights with the `ipw_cutoff_quantile` or `ipw_cutoff_value` parameters.

In addition to the package output described in McGrath et al. (2020), the output will display estimates of the "cumulative percent intervened on" and the "average percent intervened on". When using a custom intervention function, users need to specify whether each individual at that time point is eligible to contribute person-time to the percent intervened on calculations. Specifically, this must be specified in the `eligible_pt` column of `newdf`. By default, `eligible_pt` is set to `TRUE` for each individual at each time point in custom interventions.

Value

An object of class "gformula_survival" (for survival outcomes), "gformula_continuous_eof" (for continuous end-of-follow-up outcomes), or "gformula_binary_eof" (for binary end-of-follow-up outcomes). The object is a list with the following components:

`result` Results table. For survival outcomes, this contains the estimated risk, risk difference, and risk ratio for all interventions (inculding the natural course) at each time point. For continuous end-of-follow-up outcomes, this contains estimated mean outcome, mean difference, and mean ratio for all interventions (inculding natural course) at the last time point. For binary end-of-follow-up outcomes, this contains the estimated outcome probability, probability difference, and probability ratio for all interventions (inculding natural course) at the last time point. For all outcome types, this also contains the "cumulative percent intervened on" and the "average percent intervened on". If bootstrapping was used, the results table includes the bootstrap risk / mean / probability difference, ratio, standard error, and 95% confidence interval.

`coeffs` A list of the coefficients of the fitted models.

`stderrs` A list of the standard errors of the coefficients of the fitted models.

`vcovs` A list of the variance-covariance matrices of the parameters of the fitted models.

`rmses` A list of root mean square error (RMSE) values of the fitted models.

`hazardratio_val` Hazard ratio between two interventions (if applicable).

`fits` A list of the fitted models for the time-varying covariates, outcome, and competing event (if applicable). If `model_fits` is set to `FALSE`, a value of `NULL` is given.

`sim_data` A list of data tables of the simulated data. Each element in the list corresponds to one of the interventions. If the argument `sim_data_b` is set to `FALSE`, a value of `NA` is given.

`IP_weights` A numeric vector specifying the inverse probability weights. See "Details".

`bootests` A data.table containing the bootstrap replicates of the parametric g-formula estimates. If `boot_diag` is set to `FALSE`, a value of `NULL` is given.

`bootcoeffs` A list, where the kth element is a list containing the coefficients of the fitted models corresponding to the kth bootstrap sample. If `boot_diag` is set to `FALSE`, a value of `NULL` is given.

`bootstderrs` A list, where the kth element is a list containing the standard errors of the coefficients of the fitted models corresponding to the kth bootstrap sample. If `boot_diag` is set to `FALSE`, a value of `NULL` is given.

`bootvcovs` A list, where the kth element is a list containing the variance-covariance matrices of the parameters of the fitted models corresponding to the kth bootstrap sample. If `boot_diag` is set to `FALSE`, a value of `NULL` is given.

`...` Some additional elements.

The results for the g-formula simulation are printed with the `print.gformula_survival`, `print.gformula_continuous_eo` and `print.gformula_binary_eof` functions. To generate graphs comparing the mean estimated covariate values and risks over time and mean observed covariate values and risks over time, use the `plot.gformula_survival`, `plot.gformula_continuous_eof`, and `plot.gformula_binary_eof` functions.

References

Chiu YH, Wen L, McGrath S, Logan R, Dahabreh IJ, Hernán MA. Evaluating model specification when using the parametric g-formula in the presence of censoring. American Journal of Epidemiology. 2023;192:1887–1895.

McGrath S, Lin V, Zhang Z, Petito LC, Logan RW, Hernán MA, and JG Young. gfoRmula: An R package for estimating the effects of sustained treatment strategies via the parametric g-formula. Patterns. 2020;1:100008.

Robins JM. A new approach to causal inference in mortality studies with a sustained exposure period: application to the healthy worker survivor effect. Mathematical Modelling. 1986;7:1393–1512. [Errata (1987) in Computers and Mathematics with Applications 14, 917.-921. Addendum (1987) in Computers and Mathematics with Applications 14, 923-.945. Errata (1987) to addendum in Computers and Mathematics with Applications 18, 477.].

Examples

```
## Estimating the effect of static treatment strategies on risk of a
## failure event

id <- 'id'
time_points <- 7
time_name <- 't0'
covnames <- c('L1', 'L2', 'A')
outcome_name <- 'Y'
outcome_type <- 'survival'
covtypes <- c('binary', 'bounded normal', 'binary')
histories <- c(lagged, lagavg)
histvars <- list(c('A', 'L1', 'L2'), c('L1', 'L2'))
covparams <- list(covmodels = c(L1 ~ lag1_A + lag_cumavg1_L1 + lag_cumavg1_L2 +
                                  L3 + t0,
                                L2 ~ lag1_A + L1 + lag_cumavg1_L1 +
                                  lag_cumavg1_L2 + L3 + t0,
                                A ~ lag1_A + L1 + L2 + lag_cumavg1_L1 +
                                  lag_cumavg1_L2 + L3 + t0))
ymodel <- Y ~ A + L1 + L2 + L3 + lag1_A + lag1_L1 + lag1_L2 + t0
intervention1.A <- list(static, rep(0, time_points))
intervention2.A <- list(static, rep(1, time_points))
int_descript <- c('Never treat', 'Always treat')
nsimul <- 10000

gform_basic <- gformula(obs_data = basicdata_nocomp, id = id,
                        time_points = time_points,
                        time_name = time_name, covnames = covnames,
                        outcome_name = outcome_name,
                        outcome_type = outcome_type, covtypes = covtypes,
                        covparams = covparams, ymodel = ymodel,
                        intervention1.A = intervention1.A,
                        intervention2.A = intervention2.A,
                        int_descript = int_descript,
                        histories = histories, histvars = histvars,
                        basecovs = c('L3'), nsimul = nsimul,
                        seed = 1234)
gform_basic



## Estimating the effect of treatment strategies on risk of a failure event
## when competing events exist

id <- 'id'
time_points <- 7
time_name <- 't0'
covnames <- c('L1', 'L2', 'A')
outcome_name <- 'Y'
compevent_name <- 'D'
outcome_type <- 'survival'
covtypes <- c('binary', 'bounded normal', 'binary')
```

```
histories <- c(lagged, lagavg)
histvars <- list(c('A', 'L1', 'L2'), c('L1', 'L2'))
covparams <- list(covlink = c('logit', 'identity', 'logit'),
                  covmodels = c(L1 ~ lag1_A + lag_cumavg1_L1 + lag_cumavg1_L2 +
                                  L3 + as.factor(t0),
                                L2 ~ lag1_A + L1 + lag_cumavg1_L1 +
                                  lag_cumavg1_L2 + L3 + as.factor(t0),
                                A ~ lag1_A + L1 + L2 + lag_cumavg1_L1 +
                                  lag_cumavg1_L2 + L3 + as.factor(t0)))
ymodel <- Y ~ A + L1 + L2 + lag1_A + lag1_L1 + lag1_L2 + L3 + as.factor(t0)
compevent_model <- D ~ A + L1 + L2 + lag1_A + lag1_L1 + lag1_L2 + L3 + as.factor(t0)
intervention1.A <- list(static, rep(0, time_points))
intervention2.A <- list(static, rep(1, time_points))
int_descript <- c('Never treat', 'Always treat')
nsimul <- 10000

gform_basic <- gformula(obs_data = basicdata, id = id,
                        time_points = time_points,
                        time_name = time_name, covnames = covnames,
                        outcome_name = outcome_name,
                        outcome_type = outcome_type,
                        compevent_name = compevent_name,
                        covtypes = covtypes,
                        covparams = covparams, ymodel = ymodel,
                        compevent_model = compevent_model,
                        intervention1.A = intervention1.A,
                        intervention2.A = intervention2.A,
                        int_descript = int_descript,
                        histories = histories, histvars = histvars,
                        basecovs = c('L3'), nsimul = nsimul,
                        seed = 1234)
gform_basic



## Estimating the effect of treatment strategies on the mean of a continuous
## end of follow-up outcome

library('Hmisc')
id <- 'id'
time_name <- 't0'
covnames <- c('L1', 'L2', 'A')
outcome_name <- 'Y'
outcome_type <- 'continuous_eof'
covtypes <- c('categorical', 'normal', 'binary')
histories <- c(lagged)
histvars <- list(c('A', 'L1', 'L2'))
covparams <- list(covmodels = c(L1 ~ lag1_A + lag1_L1 + L3 + t0 +
                                  rcspline.eval(lag1_L2, knots = c(-1, 0, 1)),
                                L2 ~ lag1_A + L1 + lag1_L1 + lag1_L2 + L3 + t0,
                                A ~ lag1_A + L1 + L2 + lag1_L1 + lag1_L2 + L3 + t0))
ymodel <- Y ~ A + L1 + L2 + lag1_A + lag1_L1 + lag1_L2 + L3
intervention1.A <- list(static, rep(0, 7))
```

```
intervention2.A <- list(static, rep(1, 7))
int_descript <- c('Never treat', 'Always treat')
nsimul <- 10000

gform_cont_eof <- gformula(obs_data = continuous_eofdata,
                           id = id, time_name = time_name,
                           covnames = covnames, outcome_name = outcome_name,
                           outcome_type = outcome_type, covtypes = covtypes,
                           covparams = covparams, ymodel = ymodel,
                           intervention1.A = intervention1.A,
                           intervention2.A = intervention2.A,
                           int_descript = int_descript,
                           histories = histories, histvars = histvars,
                           basecovs = c("L3"), nsimul = nsimul, seed = 1234)
gform_cont_eof



## Estimating the effect of threshold interventions on the mean of a binary
## end of follow-up outcome

outcome_type <- 'binary_eof'
id <- 'id_num'
time_name <- 'time'
covnames <- c('cov1', 'cov2', 'treat')
outcome_name <- 'outcome'
histories <- c(lagged, cumavg)
histvars <- list(c('treat', 'cov1', 'cov2'), c('cov1', 'cov2'))
covtypes <- c('binary', 'zero-inflated normal', 'normal')
covparams <- list(covmodels = c(cov1 ~ lag1_treat + lag1_cov1 + lag1_cov2 +
                                  cov3 + time,
                                cov2 ~ lag1_treat + cov1 + lag1_cov1 +
                                  lag1_cov2 + cov3 + time,
                                treat ~ lag1_treat + cumavg_cov1 +
                                  cumavg_cov2 + cov3 + time))
ymodel <- outcome ~  treat + cov1 + cov2 + lag1_cov1 + lag1_cov2 + cov3
intervention1.treat <- list(static, rep(0, 7))
intervention2.treat <- list(threshold, 1, Inf)
int_descript <- c('Never treat', 'Threshold - lower bound 1')
nsimul <- 10000
ncores <- 2

gform_bin_eof <- gformula(obs_data = binary_eofdata,
                          outcome_type = outcome_type, id = id,
                          time_name = time_name, covnames = covnames,
                          outcome_name = outcome_name, covtypes = covtypes,
                          covparams = covparams, ymodel = ymodel,
                          intervention1.treat = intervention1.treat,
                          intervention2.treat = intervention2.treat,
                          int_descript = int_descript, histories = histories,
                          histvars = histvars, basecovs = c("cov3"),
                          seed = 1234, parallel = TRUE, nsamples = 5,
                          nsimul = nsimul, ncores = ncores)
```

```
gform_bin_eof


## Using IP weighting to estimate natural course risk
## Only the natural course intervention is included for simplicity

covnames <- c('L', 'A')
histories <- c(lagged)
histvars <- list(c('A', 'L'))
ymodel <- Y ~ L + A
covtypes <- c('binary', 'normal')
covparams <- list(covmodels = c(L ~ lag1_L + lag1_A,
                                A ~ lag1_L + L + lag1_A))
censor_name <- 'C'
censor_model <- C ~ L
res_censor <- gformula(obs_data = censor_data, id = 'id',
                       time_name = 't0', covnames = covnames,
                       outcome_name = 'Y', outcome_type = 'survival',
                       censor_name = censor_name, censor_model = censor_model,
                       covtypes = covtypes,
                       covparams = covparams, ymodel = ymodel,
                       histories = histories, histvars = histvars,
                       seed = 1234)
plot(res_censor)
```

`gformula_binary_eof` *Estimation of Binary End-of-Follow-Up Outcome Under the Parametric G-Formula*

Description

Based on an observed data set, this internal function estimates the outcome probability at end-of-follow-up under multiple user-specified interventions using the parametric g-formula. See McGrath et al. (2020) for further details concerning the application and implementation of the parametric g-formula.

Usage

```
gformula_binary_eof(
  obs_data,
  id,
  time_name,
  covnames,
  covtypes,
  covparams,
  covfits_custom = NA,
  covpredict_custom = NA,
```

```
  histvars = NULL,
  histories = NA,
  basecovs = NA,
  censor_name = NULL,
  censor_model = NA,
  outcome_name,
  ymodel,
  ymodel_fit_custom = NULL,
  ymodel_predict_custom = NULL,
  intvars = NULL,
  interventions = NULL,
  int_times = NULL,
  int_descript = NULL,
  ref_int = 0,
  visitprocess = NA,
  restrictions = NA,
  yrestrictions = NA,
  baselags = FALSE,
  nsimul = NA,
  sim_data_b = FALSE,
  seed,
  nsamples = 0,
  parallel = FALSE,
  ncores = NA,
  ci_method = "percentile",
  threads,
  model_fits = FALSE,
  boot_diag = FALSE,
  show_progress = TRUE,
  ipw_cutoff_quantile = NULL,
  ipw_cutoff_value = NULL,
  int_visit_type = NULL,
  sim_trunc = TRUE,
  ...
)
```

Arguments

obs_data	Data table containing the observed data.
id	Character string specifying the name of the ID variable in obs_data.
time_name	Character string specifying the name of the time variable in obs_data.
covnames	Vector of character strings specifying the names of the time-varying covariates in obs_data.
covtypes	Vector of character strings specifying the "type" of each time-varying covariate included in covnames. The possible "types" are: "binary", "normal", "categorical", "bounded normal", "zero-inflated normal", "truncated normal", "absorbing", "categorical time", and "custom".

covparams — List of vectors, where each vector contains information for one parameter used in the modeling of the time-varying covariates (e.g., model statement, family, link function, etc.). Each vector must be the same length as covnames and in the same order. If a parameter is not required for a certain covariate, it should be set to NA at that index.

covfits_custom — Vector containing custom fit functions for time-varying covariates that do not fall within the pre-defined covariate types. It should be in the same order covnames. If a custom fit function is not required for a particular covariate (e.g., if the first covariate is of type "binary" but the second is of type "custom"), then that index should be set to NA. The default is NA.

covpredict_custom — Vector containing custom prediction functions for time-varying covariates that do not fall within the pre-defined covariate types. It should be in the same order as covnames. If a custom prediction function is not required for a particular covariate, then that index should be set to NA. The default is NA.

histvars — List of vectors. The kth vector specifies the names of the variables for which the kth history function in histories is to be applied.

histories — Vector of history functions to apply to the variables specified in histvars. The default is NA.

basecovs — Vector of character strings specifying the names of baseline covariates in obs_data. These covariates are not simulated using a model but rather carry their value over all time points from the first time point of obs_data. These covariates should not be included in covnames. The default is NA.

censor_name — Character string specifying the name of the censoring variable in obs_data. Only applicable when using inverse probability weights to estimate the natural course means / risk from the observed data. See "Details".

censor_model — Model statement for the censoring variable. Only applicable when using inverse probability weights to estimate the natural course means / risk from the observed data. See "Details".

outcome_name — Character string specifying the name of the outcome variable in obs_data.

ymodel — Model statement for the outcome variable.

ymodel_fit_custom — Function specifying a custom outcome model. See the vignette "Using Custom Outcome Models in gfoRmula" for details. The default is NULL.

ymodel_predict_custom — Function obtaining predictions from the custom outcome model specified in ymodel_fit_custom. See the vignette "Using Custom Outcome Models in gfoRmula" for details. The default is NULL.

intvars — (Deprecated. See the ... argument) List, whose elements are vectors of character strings. The kth vector in intvars specifies the name(s) of the variable(s) to be intervened on in each round of the simulation under the kth intervention in interventions.

interventions — (Deprecated. See the ... argument) List, whose elements are lists of vectors. Each list in interventions specifies a unique intervention on the relevant variable(s) in intvars. Each vector contains a function implementing a particular

intervention on a single variable, optionally followed by one or more "intervention values" (i.e., integers used to specify the treatment regime).

`int_times` (Deprecated. See the `...` argument) List, whose elements are lists of vectors. The kth list in `int_times` corresponds to the kth intervention in `interventions`. Each vector specifies the time points in which the relevant intervention is applied on the corresponding variable in `intvars`. When an intervention is not applied, the simulated natural course value is used. By default, this argument is set so that all interventions are applied in all time points.

`int_descript` Vector of character strings, each describing an intervention. It must be in same order as the specified interventions (see the `...` argument).

`ref_int` Integer denoting the intervention to be used as the reference for calculating the end-of-follow-up mean ratio and mean difference. 0 denotes the natural course, while subsequent integers denote user-specified interventions in the order that they are named in `interventions`. The default is 0.

`visitprocess` List of vectors. Each vector contains as its first entry the covariate name of a visit process; its second entry the name of a covariate whose modeling depends on the visit process; and its third entry the maximum number of consecutive visits that can be missed before an individual is censored. The default is `NA`.

`restrictions` List of vectors. Each vector contains as its first entry a covariate for which *a priori* knowledge of its distribution is available; its second entry a condition under which no knowledge of its distribution is available and that must be `TRUE` for the distribution of that covariate given that condition to be estimated via a parametric model or other fitting procedure; its third entry a function for estimating the distribution of that covariate given the condition in the second entry is false such that *a priori* knowledge of the covariate distribution is available; and its fourth entry a value used by the function in the third entry. The default is `NA`.

`yrestrictions` List of vectors. Each vector contains as its first entry a condition and its second entry an integer. When the condition is `TRUE`, the outcome variable is simulated according to the fitted model; when the condition is `FALSE`, the outcome variable takes on the value in the second entry. The default is `NA`.

`baselags` Logical scalar for specifying the convention used for lagi and lag_cumavgi terms in the model statements when pre-baseline times are not included in `obs_data` and when the current time index, t, is such that $t < i$. If this argument is set to `FALSE`, the value of all lagi and lag_cumavgi terms in this context are set to 0 (for non-categorical covariates) or the reference level (for categorical covariates). If this argument is set to `TRUE`, the value of lagi and lag_cumavgi terms are set to their values at time 0. The default is `FALSE`.

`nsimul` Number of subjects for whom to simulate data. By default, this argument is set equal to the number of subjects in `obs_data`.

`sim_data_b` Logical scalar indicating whether to return the simulated data set. If bootstrap samples are used (i.e., `nsamples` is set to a value greater than 0), this argument must be set to `FALSE`. The default is `FALSE`.

`seed` Starting seed for simulations and bootstrapping.

`nsamples` Integer specifying the number of bootstrap samples to generate. The default is 0.

`parallel` Logical scalar indicating whether to parallelize simulations of different interventions to multiple cores.

`ncores` Integer specifying the number of CPU cores to use in parallel simulation. This argument is required when parallel is set to `TRUE`. In many applications, users may wish to set this argument equal to `parallel::detectCores() - 1`.

`ci_method` Character string specifying the method for calculating the bootstrap 95% confidence intervals, if applicable. The options are `"percentile"` and `"normal"`.

`threads` Integer specifying the number of threads to be used in `data.table`. See `setDTthreads` for further details.

`model_fits` Logical scalar indicating whether to return the fitted models. Note that if this argument is set to `TRUE`, the output of this function may use a lot of memory. The default is `FALSE`.

`boot_diag` Logical scalar indicating whether to return the parametric g-formula estimates as well as the coefficients, standard errors, and variance-covariance matrices of the parameters of the fitted models in the bootstrap samples. The default is `FALSE`.

`show_progress` Logical scalar indicating whether to print a progress bar for the number of bootstrap samples completed in the R console. This argument is only applicable when `parallel` is set to `FALSE` and bootstrap samples are used (i.e., `nsamples` is set to a value greater than 0). The default is `TRUE`.

`ipw_cutoff_quantile`
Percentile by which to truncate inverse probability weights. The default is `NULL` (i.e., no truncation). See "Details".

`ipw_cutoff_value`
Cutoff value by which to truncate inverse probability weights. The default is `NULL` (i.e., no truncation). See "Details".

`int_visit_type` Vector of logicals. The kth element is a logical specifying whether to carry forward the intervened value (rather than the natural value) of the treatment variables(s) when performing a carry forward restriction type for the kth intervention in `interventions`. When the kth element is set to `FALSE`, the natural value of the treatment variable(s) in the kth intervention in `interventions` will be carried forward. By default, this argument is set so that the intervened value of the treatment variable(s) is carried forward for all interventions.

`sim_trunc` Logical scalar indicating whether to truncate simulated covariates to their range in the observed data set. This argument is only applicable for covariates of type `"normal"`, `"bounded normal"`, `"truncated normal"`, and `"zero-inflated normal"`. The default is `TRUE`.

`...` Other arguments, including (a) those that specify the interventions and (b) those that are passed to the functions in `covpredict_custom`. To specify interventions, users can supply arguments with the following naming requirements

- Each intervention argument begins with a prefix of `intervention`.
- After the prefix, the intervention number is specified and followed by a period.
- After the period, the treatment variable name is specified.

Each intervention argument takes as input a list with the following elements:

- The first element specifies the intervention function.
- The subsequent elements specify any intervention values.
- (Optional) The named element `int_times` specifies the time points to apply the intervention. By default, all interventions are applied at all time points.

For example, an "always treat" intervention on `A` is given by
`intervention1.A = list(static, rep(1, time_points))`
See the vignette "A Simplified Approach for Specifying Interventions in gfoRmula" and "Examples" section for more examples.

Details

To assess model misspecification in the parametric g-formula, users can obtain inverse probability (IP) weighted estimates of the natural course means of the time-varying covariates from the observed data. See Chiu et al. (2023) for details. In addition to the general requirements described in McGrath et al. (2020), the requirements for the input data set and the call to the gformula function for such analyses are described below.

Users need to include a column in `obs_data` with a time-varying censoring variable. Users need to indicate the name of the censoring variable and a model statement for the censoring variable with parameters `censor_name` and `censor_model`, respectively. Finally, users can specify how to truncate IP weights with the `ipw_cutoff_quantile` or `ipw_cutoff_value` parameters.

In addition to the package output described in McGrath et al. (2020), the output will display estimates of the "cumulative percent intervened on" and the "average percent intervened on". When using a custom intervention function, users need to specify whether each individual at that time point is eligible to contribute person-time to the percent intervened on calculations. Specifically, this must be specified in the `eligible_pt` column of `newdf`. By default, `eligible_pt` is set to `TRUE` for each individual at each time point in custom interventions.

Value

An object of class "gformula_binary_eof". The object is a list with the following components:

`result`	Results table containing the estimated outcome probability for all interventions (inculding natural course) at the last time point as well as the "cumulative percent intervened on" and the "average percent intervened on". If bootstrapping was used, the results table includes the bootstrap end-of-follow-up mean ratio, standard error, and 95% confidence interval.
`coeffs`	A list of the coefficients of the fitted models.
`stderrs`	A list of the standard errors of the coefficients of the fitted models.
`vcovs`	A list of the variance-covariance matrices of the parameters of the fitted models.
`rmses`	A list of root mean square error (RMSE) values of the fitted models.
`fits`	A list of the fitted models for the time-varying covariates and outcome. If `model_fits` is set to `FALSE`, a value of `NULL` is given.
`sim_data`	A list of data tables of the simulated data. Each element in the list corresponds to one of the interventions. If the argument `sim_data_b` is set to `FALSE`, a value of `NA` is given.
`IP_weights`	A numeric vector specifying the inverse probability weights. See "Details".

bootests	A data.table containing the bootstrap replicates of the parametric g-formula estimates. If boot_diag is set to FALSE, a value of NULL is given.
bootcoeffs	A list, where the kth element is a list containing the coefficients of the fitted models corresponding to the kth bootstrap sample. If boot_diag is set to FALSE, a value of NULL is given.
bootstderrs	A list, where the kth element is a list containing the standard errors of the coefficients of the fitted models corresponding to the kth bootstrap sample. If boot_diag is set to FALSE, a value of NULL is given.
bootvcovs	A list, where the kth element is a list containing the variance-covariance matrices of the parameters of the fitted models corresponding to the kth bootstrap sample. If boot_diag is set to FALSE, a value of NULL is given.
...	Some additional elements.

The results for the g-formula simulation under various interventions for the last time point are printed with the print.gformula_binary_eof function. To generate graphs comparing the mean estimated and observed covariate values over time, use the plot.gformula_binary_eof function.

References

Chiu YH, Wen L, McGrath S, Logan R, Dahabreh IJ, Hernán MA. Evaluating model specification when using the parametric g-formula in the presence of censoring. American Journal of Epidemiology. 2023;192:1887–1895.

McGrath S, Lin V, Zhang Z, Petito LC, Logan RW, Hernán MA, and JG Young. gfoRmula: An R package for estimating the effects of sustained treatment strategies via the parametric g-formula. Patterns. 2020;1:100008.

Robins JM. A new approach to causal inference in mortality studies with a sustained exposure period: application to the healthy worker survivor effect. Mathematical Modelling. 1986;7:1393–1512. [Errata (1987) in Computers and Mathematics with Applications 14, 917.-921. Addendum (1987) in Computers and Mathematics with Applications 14, 923-.945. Errata (1987) to addendum in Computers and Mathematics with Applications 18, 477.].

See Also

gformula

Examples

```
## Estimating the effect of threshold interventions on the mean of a binary
## end of follow-up outcome

id <- 'id_num'
time_name <- 'time'
covnames <- c('cov1', 'cov2', 'treat')
outcome_name <- 'outcome'
histories <- c(lagged, cumavg)
histvars <- list(c('treat', 'cov1', 'cov2'), c('cov1', 'cov2'))
covtypes <- c('binary', 'zero-inflated normal', 'normal')
covparams <- list(covmodels = c(cov1 ~ lag1_treat + lag1_cov1 + lag1_cov2 + cov3 +
                                  time,
```

```
                                cov2 ~ lag1_treat + cov1 + lag1_cov1 + lag1_cov2 +
                                  cov3 + time,
                                treat ~ lag1_treat + cumavg_cov1 +
                                  cumavg_cov2 + cov3 + time))
ymodel <- outcome ~  treat + cov1 + cov2 + lag1_cov1 + lag1_cov2 + cov3
intervention1.treat <- list(static, rep(0, 7))
intervention2.treat <- list(threshold, 1, Inf)
int_descript <- c('Never treat', 'Threshold - lower bound 1')
nsimul <- 10000
ncores <- 2

gform_bin_eof <- gformula_binary_eof(obs_data = binary_eofdata, id = id,
                                     time_name = time_name,
                                     covnames = covnames,
                                     outcome_name = outcome_name,
                                     covtypes = covtypes,
                                     covparams = covparams,
                                     ymodel = ymodel,
                                     intervention1.treat = intervention1.treat,
                                     intervention2.treat = intervention2.treat,
                                     int_descript = int_descript,
                                     histories = histories, histvars = histvars,
                                     basecovs = c("cov3"), seed = 1234,
                                     parallel = TRUE, nsamples = 5,
                                     nsimul = nsimul, ncores = ncores)
gform_bin_eof
```

`gformula_continuous_eof` *Estimation of Continuous End-of-Follow-Up Outcome Under the Parametric G-Formula*

Description

Based on an observed data set, this internal function estimates the outcome mean at end-of-follow-up under multiple user-specified interventions using the parametric g-formula. See McGrath et al. (2020) for further details concerning the application and implementation of the parametric g-formula.

Usage

```
gformula_continuous_eof(
  obs_data,
  id,
  time_name,
  covnames,
  covtypes,
```

```
  covparams,
  covfits_custom = NA,
  covpredict_custom = NA,
  histvars = NULL,
  histories = NA,
  basecovs = NA,
  outcome_name,
  ymodel,
  ymodel_fit_custom = NULL,
  ymodel_predict_custom = NULL,
  censor_name = NULL,
  censor_model = NA,
  intvars = NULL,
  interventions = NULL,
  int_times = NULL,
  int_descript = NULL,
  ref_int = 0,
  visitprocess = NA,
  restrictions = NA,
  yrestrictions = NA,
  baselags = FALSE,
  nsimul = NA,
  sim_data_b = FALSE,
  seed,
  nsamples = 0,
  parallel = FALSE,
  ncores = NA,
  ci_method = "percentile",
  threads,
  model_fits = FALSE,
  boot_diag = FALSE,
  show_progress = TRUE,
  ipw_cutoff_quantile = NULL,
  ipw_cutoff_value = NULL,
  int_visit_type = NULL,
  sim_trunc = TRUE,
  ...
)
```

Arguments

obs_data	Data table containing the observed data.
id	Character string specifying the name of the ID variable in obs_data.
time_name	Character string specifying the name of the time variable in obs_data.
covnames	Vector of character strings specifying the names of the time-varying covariates in obs_data.
covtypes	Vector of character strings specifying the "type" of each time-varying covariate included in covnames. The possible "types" are: "binary", "normal",

"categorical", "bounded normal", "zero-inflated normal", "truncated normal", "absorbing", "categorical time", and "custom".

covparams — List of vectors, where each vector contains information for one parameter used in the modeling of the time-varying covariates (e.g., model statement, family, link function, etc.). Each vector must be the same length as covnames and in the same order. If a parameter is not required for a certain covariate, it should be set to NA at that index.

covfits_custom — Vector containing custom fit functions for time-varying covariates that do not fall within the pre-defined covariate types. It should be in the same order covnames. If a custom fit function is not required for a particular covariate (e.g., if the first covariate is of type "binary" but the second is of type "custom"), then that index should be set to NA. The default is NA.

covpredict_custom — Vector containing custom prediction functions for time-varying covariates that do not fall within the pre-defined covariate types. It should be in the same order as covnames. If a custom prediction function is not required for a particular covariate, then that index should be set to NA. The default is NA.

histvars — List of vectors. The kth vector specifies the names of the variables for which the kth history function in histories is to be applied.

histories — Vector of history functions to apply to the variables specified in histvars. The default is NA.

basecovs — Vector of character strings specifying the names of baseline covariates in obs_data. These covariates are not simulated using a model but rather carry their value over all time points from the first time point of obs_data. These covariates should not be included in covnames. The default is NA.

outcome_name — Character string specifying the name of the outcome variable in obs_data.

ymodel — Model statement for the outcome variable.

ymodel_fit_custom — Function specifying a custom outcome model. See the vignette "Using Custom Outcome Models in gfoRmula" for details. The default is NULL.

ymodel_predict_custom — Function obtaining predictions from the custom outcome model specified in ymodel_fit_custom. See the vignette "Using Custom Outcome Models in gfoRmula" for details. The default is NULL.

censor_name — Character string specifying the name of the censoring variable in obs_data. Only applicable when using inverse probability weights to estimate the natural course means / risk from the observed data. See "Details".

censor_model — Model statement for the censoring variable. Only applicable when using inverse probability weights to estimate the natural course means / risk from the observed data. See "Details".

intvars — (Deprecated. See the ... argument) List, whose elements are vectors of character strings. The kth vector in intvars specifies the name(s) of the variable(s) to be intervened on in each round of the simulation under the kth intervention in interventions.

`interventions` (Deprecated. See the `...` argument) List, whose elements are lists of vectors. Each list in `interventions` specifies a unique intervention on the relevant variable(s) in `intvars`. Each vector contains a function implementing a particular intervention on a single variable, optionally followed by one or more "intervention values" (i.e., integers used to specify the treatment regime).

`int_times` (Deprecated. See the `...` argument) List, whose elements are lists of vectors. The kth list in `int_times` corresponds to the kth intervention in `interventions`. Each vector specifies the time points in which the relevant intervention is applied on the corresponding variable in `intvars`. When an intervention is not applied, the simulated natural course value is used. By default, this argument is set so that all interventions are applied in all time points.

`int_descript` Vector of character strings, each describing an intervention. It must be in same order as the specified interventions (see the `...` argument).

`ref_int` Integer denoting the intervention to be used as the reference for calculating the end-of-follow-up mean ratio and mean difference. 0 denotes the natural course, while subsequent integers denote user-specified interventions in the order that they are named in `interventions`. The default is 0.

`visitprocess` List of vectors. Each vector contains as its first entry the covariate name of a visit process; its second entry the name of a covariate whose modeling depends on the visit process; and its third entry the maximum number of consecutive visits that can be missed before an individual is censored. The default is `NA`.

`restrictions` List of vectors. Each vector contains as its first entry a covariate for which *a priori* knowledge of its distribution is available; its second entry a condition under which no knowledge of its distribution is available and that must be `TRUE` for the distribution of that covariate given that condition to be estimated via a parametric model or other fitting procedure; its third entry a function for estimating the distribution of that covariate given the condition in the second entry is false such that *a priori* knowledge of the covariate distribution is available; and its fourth entry a value used by the function in the third entry. The default is `NA`.

`yrestrictions` List of vectors. Each vector contains as its first entry a condition and its second entry an integer. When the condition is `TRUE`, the outcome variable is simulated according to the fitted model; when the condition is `FALSE`, the outcome variable takes on the value in the second entry. The default is `NA`.

`baselags` Logical scalar for specifying the convention used for lagi and lag_cumavgi terms in the model statements when pre-baseline times are not included in `obs_data` and when the current time index, t, is such that $t < i$. If this argument is set to `FALSE`, the value of all lagi and lag_cumavgi terms in this context are set to 0 (for non-categorical covariates) or the reference level (for categorical covariates). If this argument is set to `TRUE`, the value of lagi and lag_cumavgi terms are set to their values at time 0. The default is `FALSE`.

`nsimul` Number of subjects for whom to simulate data. By default, this argument is set equal to the number of subjects in `obs_data`.

`sim_data_b` Logical scalar indicating whether to return the simulated data set. If bootstrap samples are used (i.e., `nsamples` is set to a value greater than 0), this argument must be set to `FALSE`. The default is `FALSE`.

`seed` Starting seed for simulations and bootstrapping.

`nsamples` Integer specifying the number of bootstrap samples to generate. The default is 0.

`parallel` Logical scalar indicating whether to parallelize simulations of different interventions to multiple cores.

`ncores` Integer specifying the number of CPU cores to use in parallel simulation. This argument is required when parallel is set to `TRUE`. In many applications, users may wish to set this argument equal to `parallel::detectCores() - 1`.

`ci_method` Character string specifying the method for calculating the bootstrap 95% confidence intervals, if applicable. The options are `"percentile"` and `"normal"`.

`threads` Integer specifying the number of threads to be used in `data.table`. See `setDTthreads` for further details.

`model_fits` Logical scalar indicating whether to return the fitted models. Note that if this argument is set to `TRUE`, the output of this function may use a lot of memory. The default is `FALSE`.

`boot_diag` Logical scalar indicating whether to return the parametric g-formula estimates as well as the coefficients, standard errors, and variance-covariance matrices of the parameters of the fitted models in the bootstrap samples. The default is `FALSE`.

`show_progress` Logical scalar indicating whether to print a progress bar for the number of bootstrap samples completed in the R console. This argument is only applicable when `parallel` is set to `FALSE` and bootstrap samples are used (i.e., `nsamples` is set to a value greater than 0). The default is `TRUE`.

`ipw_cutoff_quantile`
Percentile by which to truncate inverse probability weights. The default is `NULL` (i.e., no truncation). See "Details".

`ipw_cutoff_value`
Cutoff value by which to truncate inverse probability weights. The default is `NULL` (i.e., no truncation). See "Details".

`int_visit_type` Vector of logicals. The kth element is a logical specifying whether to carry forward the intervened value (rather than the natural value) of the treatment variables(s) when performing a carry forward restriction type for the kth intervention in `interventions`. When the kth element is set to `FALSE`, the natural value of the treatment variable(s) in the kth intervention in `interventions` will be carried forward. By default, this argument is set so that the intervened value of the treatment variable(s) is carried forward for all interventions.

`sim_trunc` Logical scalar indicating whether to truncate simulated covariates to their range in the observed data set. This argument is only applicable for covariates of type `"normal"`, `"bounded normal"`, `"truncated normal"`, and `"zero-inflated normal"`. The default is `TRUE`.

`...` Other arguments, including (a) those that specify the interventions and (b) those that are passed to the functions in `covpredict_custom`. To specify interventions, users can supply arguments with the following naming requirements

- Each intervention argument begins with a prefix of `intervention`.
- After the prefix, the intervention number is specified and followed by a period.

- After the period, the treatment variable name is specified.

Each intervention argument takes as input a list with the following elements:

- The first element specifies the intervention function.
- The subsequent elements specify any intervention values.
- (Optional) The named element `int_times` specifies the time points to apply the intervention. By default, all interventions are applied at all time points.

For example, an "always treat" intervention on `A` is given by
`intervention1.A = list(static, rep(1, time_points))`
See the vignette "A Simplified Approach for Specifying Interventions in gfoRmula" and "Examples" section for more examples.

Details

To assess model misspecification in the parametric g-formula, users can obtain inverse probability (IP) weighted estimates of the natural course means of the time-varying covariates from the observed data. See Chiu et al. (2023) for details. In addition to the general requirements described in McGrath et al. (2020), the requirements for the input data set and the call to the gformula function for such analyses are described below.

Users need to include a column in `obs_data` with a time-varying censoring variable. Users need to indicate the name of the censoring variable and a model statement for the censoring variable with parameters `censor_name` and `censor_model`, respectively. Finally, users can specify how to truncate IP weights with the `ipw_cutoff_quantile` or `ipw_cutoff_value` parameters.

In addition to the package output described in McGrath et al. (2020), the output will display estimates of the "cumulative percent intervened on" and the "average percent intervened on". When using a custom intervention function, users need to specify whether each individual at that time point is eligible to contribute person-time to the percent intervened on calculations. Specifically, this must be specified in the `eligible_pt` column of `newdf`. By default, `eligible_pt` is set to `TRUE` for each individual at each time point in custom interventions.

Value

An object of class "gformula_continuous_eof". The object is a list with the following components:

`result`	Results table containing the estimated mean outcome for all interventions (inculding natural course) at the last time point as well as the "cumulative percent intervened on" and the "average percent intervened on". If bootstrapping was used, the results table includes the bootstrap end-of-follow-up mean ratio, standard error, and 95% confidence interval.
`coeffs`	A list of the coefficients of the fitted models.
`stderrs`	A list of the standard errors of the coefficients of the fitted models.
`vcovs`	A list of the variance-covariance matrices of the parameters of the fitted models.
`rmses`	A list of root mean square error (RMSE) values of the fitted models.
`fits`	A list of the fitted models for the time-varying covariates and outcome. If `model_fits` is set to `FALSE`, a value of `NULL` is given.

sim_data	A list of data tables of the simulated data. Each element in the list corresponds to one of the interventions. If the argument sim_data_b is set to FALSE, a value of NA is given.
IP_weights	A numeric vector specifying the inverse probability weights. See "Details".
bootests	A data.table containing the bootstrap replicates of the parametric g-formula estimates. If boot_diag is set to FALSE, a value of NULL is given.
bootcoeffs	A list, where the kth element is a list containing the coefficients of the fitted models corresponding to the kth bootstrap sample. If boot_diag is set to FALSE, a value of NULL is given.
bootstderrs	A list, where the kth element is a list containing the standard errors of the coefficients of the fitted models corresponding to the kth bootstrap sample. If boot_diag is set to FALSE, a value of NULL is given.
bootvcovs	A list, where the kth element is a list containing the variance-covariance matrices of the parameters of the fitted models corresponding to the kth bootstrap sample. If boot_diag is set to FALSE, a value of NULL is given.
...	Some additional elements.

The results for the g-formula simulation under various interventions for the last time point are printed with the print.gformula_continuous_eof function. To generate graphs comparing the mean estimated and observed covariate values over time, use the print.gformula_continuous_eof function.

References

Chiu YH, Wen L, McGrath S, Logan R, Dahabreh IJ, Hernán MA. Evaluating model specification when using the parametric g-formula in the presence of censoring. American Journal of Epidemiology. 2023;192:1887–1895.

McGrath S, Lin V, Zhang Z, Petito LC, Logan RW, Hernán MA, and JG Young. gfoRmula: An R package for estimating the effects of sustained treatment strategies via the parametric g-formula. Patterns. 2020;1:100008.

Robins JM. A new approach to causal inference in mortality studies with a sustained exposure period: application to the healthy worker survivor effect. Mathematical Modelling. 1986;7:1393–1512. [Errata (1987) in Computers and Mathematics with Applications 14, 917.-921. Addendum (1987) in Computers and Mathematics with Applications 14, 923-.945. Errata (1987) to addendum in Computers and Mathematics with Applications 18, 477.].

See Also

gformula

Examples

```
## Estimating the effect of treatment strategies on the mean of a continuous
## end of follow-up outcome

library('Hmisc')
id <- 'id'
```

```
time_name <- 't0'
covnames <- c('L1', 'L2', 'A')
outcome_name <- 'Y'
covtypes <- c('categorical', 'normal', 'binary')
histories <- c(lagged)
histvars <- list(c('A', 'L1', 'L2'))
covparams <- list(covmodels = c(L1 ~ lag1_A + lag1_L1 + L3 + t0 +
                                  rcspline.eval(lag1_L2, knots = c(-1, 0, 1)),
                                L2 ~ lag1_A + L1 + lag1_L1 + lag1_L2 + L3 + t0,
                                A ~ lag1_A + L1 + L2 + lag1_L1 + lag1_L2 + L3 + t0))
ymodel <- Y ~ A + L1 + L2 + lag1_A + lag1_L1 + lag1_L2 + L3
intervention1.A <- list(static, rep(0, 7))
intervention2.A <- list(static, rep(1, 7))
int_descript <- c('Never treat', 'Always treat')
nsimul <- 10000

gform_cont_eof <- gformula_continuous_eof(obs_data = continuous_eofdata,
                                          id = id,
                                          time_name = time_name,
                                          covnames = covnames,
                                          outcome_name = outcome_name,
                                          covtypes = covtypes,
                                          covparams = covparams, ymodel = ymodel,
                                          intervention1.A = intervention1.A,
                                          intervention2.A = intervention2.A,
                                          int_descript = int_descript,
                                          histories = histories, histvars = histvars,
                                          basecovs = c("L3"),
                                          nsimul = nsimul, seed = 1234)
gform_cont_eof
```

`gformula_survival` *Estimation of Survival Outcome Under the Parametric G-Formula*

Description

Based on an observed data set, this internal function estimates the risk over time under multiple user-specified interventions using the parametric g-formula. See McGrath et al. (2020) for further details concerning the application and implementation of the parametric g-formula.

Usage

```
gformula_survival(
  obs_data,
  id,
  time_points = NULL,
  time_name,
  covnames,
```

```
  covtypes,
  covparams,
  covfits_custom = NA,
  covpredict_custom = NA,
  histvars = NULL,
  histories = NA,
  basecovs = NA,
  outcome_name,
  ymodel,
  ymodel_fit_custom = NULL,
  ymodel_predict_custom = NULL,
  compevent_name = NULL,
  compevent_model = NA,
  compevent_cens = FALSE,
  censor_name = NULL,
  censor_model = NA,
  intvars = NULL,
  interventions = NULL,
  int_times = NULL,
  int_descript = NULL,
  ref_int = 0,
  intcomp = NA,
  visitprocess = NA,
  restrictions = NA,
  yrestrictions = NA,
  compevent_restrictions = NA,
  baselags = FALSE,
  nsimul = NA,
  sim_data_b = FALSE,
  seed,
  nsamples = 0,
  parallel = FALSE,
  ncores = NA,
  ci_method = "percentile",
  threads,
  model_fits = FALSE,
  boot_diag = FALSE,
  show_progress = TRUE,
  ipw_cutoff_quantile = NULL,
  ipw_cutoff_value = NULL,
  int_visit_type = NULL,
  sim_trunc = TRUE,
  ...
)
```

Arguments

obs_data	Data table containing the observed data.

`id` Character string specifying the name of the ID variable in `obs_data`.

`time_points` Number of time points to simulate. By default, this argument is set equal to the maximum number of records that `obs_data` contains for any individual.

`time_name` Character string specifying the name of the time variable in `obs_data`.

`covnames` Vector of character strings specifying the names of the time-varying covariates in `obs_data`.

`covtypes` Vector of character strings specifying the "type" of each time-varying covariate included in `covnames`. The possible "types" are: `"binary"`, `"normal"`, `"categorical"`, `"bounded normal"`, `"zero-inflated normal"`, `"truncated normal"`, `"absorbing"`, `"categorical time"`, and `"custom"`.

`covparams` List of vectors, where each vector contains information for one parameter used in the modeling of the time-varying covariates (e.g., model statement, family, link function, etc.). Each vector must be the same length as `covnames` and in the same order. If a parameter is not required for a certain covariate, it should be set to `NA` at that index.

`covfits_custom` Vector containing custom fit functions for time-varying covariates that do not fall within the pre-defined covariate types. It should be in the same order `covnames`. If a custom fit function is not required for a particular covariate (e.g., if the first covariate is of type `"binary"` but the second is of type `"custom"`), then that index should be set to `NA`. The default is `NA`.

`covpredict_custom` Vector containing custom prediction functions for time-varying covariates that do not fall within the pre-defined covariate types. It should be in the same order as `covnames`. If a custom prediction function is not required for a particular covariate, then that index should be set to `NA`. The default is `NA`.

`histvars` List of vectors. The kth vector specifies the names of the variables for which the kth history function in `histories` is to be applied.

`histories` Vector of history functions to apply to the variables specified in `histvars`. The default is `NA`.

`basecovs` Vector of character strings specifying the names of baseline covariates in `obs_data`. These covariates are not simulated using a model but rather carry their value over all time points from the first time point of `obs_data`. These covariates should not be included in `covnames`. The default is `NA`.

`outcome_name` Character string specifying the name of the outcome variable in `obs_data`.

`ymodel` Model statement for the outcome variable.

`ymodel_fit_custom` Function specifying a custom outcome model. See the vignette "Using Custom Outcome Models in gfoRmula" for details. The default is `NULL`.

`ymodel_predict_custom` Function obtaining predictions from the custom outcome model specified in `ymodel_fit_custom`. See the vignette "Using Custom Outcome Models in gfoRmula" for details. The default is `NULL`.

`compevent_name` Character string specifying the name of the competing event variable in `obs_data`.

`compevent_model` Model statement for the competing event variable. The default is `NA`.

`compevent_cens` Logical scalar indicating whether to treat competing events as censoring events. This argument is only applicable for survival outcomes and when a competing even model is supplied (i.e., `compevent_name` and `compevent_model` are specified). If this argument is set to `TRUE`, the competing event model will only be used to construct inverse probability weights to estimate the natural course means / risk from the observed data. If this argument is set to `FALSE`, the competing event model will be used in the parametric g-formula estimates of the risk and will not be used to construct inverse probability weights. See "Details". The default is `FALSE`.

`censor_name` Character string specifying the name of the censoring variable in `obs_data`. Only applicable when using inverse probability weights to estimate the natural course means / risk from the observed data. See "Details".

`censor_model` Model statement for the censoring variable. Only applicable when using inverse probability weights to estimate the natural course means / risk from the observed data. See "Details".

`intvars` (Deprecated. See the `...` argument) List, whose elements are vectors of character strings. The kth vector in `intvars` specifies the name(s) of the variable(s) to be intervened on in each round of the simulation under the kth intervention in `interventions`.

`interventions` (Deprecated. See the `...` argument) List, whose elements are lists of vectors. Each list in `interventions` specifies a unique intervention on the relevant variable(s) in `intvars`. Each vector contains a function implementing a particular intervention on a single variable, optionally followed by one or more "intervention values" (i.e., integers used to specify the treatment regime).

`int_times` (Deprecated. See the `...` argument) List, whose elements are lists of vectors. The kth list in `int_times` corresponds to the kth intervention in `interventions`. Each vector specifies the time points in which the relevant intervention is applied on the corresponding variable in `intvars`. When an intervention is not applied, the simulated natural course value is used. By default, this argument is set so that all interventions are applied in all time points.

`int_descript` Vector of character strings, each describing an intervention. It must be in same order as the specified interventions (see the `...` argument).

`ref_int` Integer denoting the intervention to be used as the reference for calculating the risk ratio and risk difference. 0 denotes the natural course, while subsequent integers denote user-specified interventions in the order that they are named in `interventions`. The default is 0.

`intcomp` List of two numbers indicating a pair of interventions to be compared by a hazard ratio. The default is `NA`, resulting in no hazard ratio calculation.

`visitprocess` List of vectors. Each vector contains as its first entry the covariate name of a visit process; its second entry the name of a covariate whose modeling depends on the visit process; and its third entry the maximum number of consecutive visits that can be missed before an individual is censored. The default is `NA`.

`restrictions` List of vectors. Each vector contains as its first entry a covariate for which *a priori* knowledge of its distribution is available; its second entry a condition under which no knowledge of its distribution is available and that must be `TRUE` for the

distribution of that covariate given that condition to be estimated via a parametric model or other fitting procedure; its third entry a function for estimating the distribution of that covariate given the condition in the second entry is false such that *a priori* knowledge of the covariate distribution is available; and its fourth entry a value used by the function in the third entry. The default is NA.

`yrestrictions` List of vectors. Each vector contains as its first entry a condition and its second entry an integer. When the condition is TRUE, the outcome variable is simulated according to the fitted model; when the condition is FALSE, the outcome variable takes on the value in the second entry. The default is NA.

`compevent_restrictions`
List of vectors. Each vector contains as its first entry a condition and its second entry an integer. When the condition is TRUE, the competing event variable is simulated according to the fitted model; when the condition is FALSE, the competing event variable takes on the value in the second entry. The default is NA.

`baselags` Logical scalar for specifying the convention used for lagi and lag_cumavgi terms in the model statements when pre-baseline times are not included in `obs_data` and when the current time index, t, is such that $t < i$. If this argument is set to FALSE, the value of all lagi and lag_cumavgi terms in this context are set to 0 (for non-categorical covariates) or the reference level (for categorical covariates). If this argument is set to TRUE, the value of lagi and lag_cumavgi terms are set to their values at time 0. The default is FALSE.

`nsimul` Number of subjects for whom to simulate data. By default, this argument is set equal to the number of subjects in `obs_data`.

`sim_data_b` Logical scalar indicating whether to return the simulated data set. If bootstrap samples are used (i.e., `nsamples` is set to a value greater than 0), this argument must be set to FALSE. The default is FALSE.

`seed` Starting seed for simulations and bootstrapping.

`nsamples` Integer specifying the number of bootstrap samples to generate. The default is 0.

`parallel` Logical scalar indicating whether to parallelize simulations of different interventions to multiple cores.

`ncores` Integer specifying the number of CPU cores to use in parallel simulation. This argument is required when parallel is set to TRUE. In many applications, users may wish to set this argument equal to `parallel::detectCores() - 1`.

`ci_method` Character string specifying the method for calculating the bootstrap 95% confidence intervals, if applicable. The options are `"percentile"` and `"normal"`.

`threads` Integer specifying the number of threads to be used in `data.table`. See `setDTthreads` for further details.

`model_fits` Logical scalar indicating whether to return the fitted models. Note that if this argument is set to TRUE, the output of this function may use a lot of memory. The default is FALSE.

`boot_diag` Logical scalar indicating whether to return the parametric g-formula estimates as well as the coefficients, standard errors, and variance-covariance matrices of the parameters of the fitted models in the bootstrap samples. The default is FALSE.

show_progress Logical scalar indicating whether to print a progress bar for the number of bootstrap samples completed in the R console. This argument is only applicable when parallel is set to FALSE and bootstrap samples are used (i.e., nsamples is set to a value greater than 0). The default is TRUE.

ipw_cutoff_quantile
Percentile by which to truncate inverse probability weights. The default is NULL (i.e., no truncation). See "Details".

ipw_cutoff_value
Cutoff value by which to truncate inverse probability weights. The default is NULL (i.e., no truncation). See "Details".

int_visit_type Vector of logicals. The kth element is a logical specifying whether to carry forward the intervened value (rather than the natural value) of the treatment variables(s) when performing a carry forward restriction type for the kth intervention in interventions. When the kth element is set to FALSE, the natural value of the treatment variable(s) in the kth intervention in interventions will be carried forward. By default, this argument is set so that the intervened value of the treatment variable(s) is carried forward for all interventions.

sim_trunc Logical scalar indicating whether to truncate simulated covariates to their range in the observed data set. This argument is only applicable for covariates of type "normal", "bounded normal", "truncated normal", and "zero-inflated normal". The default is TRUE.

... Other arguments, including (a) those that specify the interventions and (b) those that are passed to the functions in covpredict_custom. To specify interventions, users can supply arguments with the following naming requirements

- Each intervention argument begins with a prefix of intervention.
- After the prefix, the intervention number is specified and followed by a period.
- After the period, the treatment variable name is specified.

Each intervention argument takes as input a list with the following elements:

- The first element specifies the intervention function.
- The subsequent elements specify any intervention values.
- (Optional) The named element int_times specifies the time points to apply the intervention. By default, all interventions are applied at all time points.

For example, an "always treat" intervention on A is given by
intervention1.A = list(static, rep(1, time_points))
See the vignette "A Simplified Approach for Specifying Interventions in gfoRmula" and "Examples" section for more examples.

Details

To assess model misspecification in the parametric g-formula, users can obtain inverse probability (IP) weighted estimates of the natural course risk and/or means of the time-varying covariates from the observed data. See Chiu et al. (2023) for details. In addition to the general requirements described in McGrath et al. (2020), the requirements for the input data set and the call to the gformula function for such analyses are described below.

Users need to include a column in obs_data with a time-varying censoring variable. Users need to indicate the name of the censoring variable and a model statement for the censoring variable with parameters censor_name and censor_model, respectively. When competing events are present, users need to include a column in obs_data with a time-varying indicator of the competing event variable and need to indicate the name of the competing event variable and the corresponding model statement with parameters compevent_name and compevent_model, respectively. Users need to indicate whether to treat competing events as censoring events with the compevent_cens parameter. Finally, users can specify how to truncate IP weights with the ipw_cutoff_quantile or ipw_cutoff_value parameters.

In addition to the package output described in McGrath et al. (2020), the output will display estimates of the "cumulative percent intervened on" and the "average percent intervened on". When using a custom intervention function, users need to specify whether each individual at that time point is eligible to contribute person-time to the percent intervened on calculations. Specifically, this must be specified in the eligible_pt column of newdf. By default, eligible_pt is set to TRUE for each individual at each time point in custom interventions.

Value

An object of class "gformula_survival". The object is a list with the following components:

result — Results table containing the estimated risk and risk ratio for all interventions (inculding the natural course) at each time point as well as the "cumulative percent intervened on" and the "average percent intervened on". If bootstrapping was used, the results table includes the bootstrap mean risk ratio, standard error, and 95% confidence interval.

coeffs — A list of the coefficients of the fitted models.

stderrs — A list of the standard errors of the coefficients of the fitted models.

vcovs — A list of the variance-covariance matrices of the parameters of the fitted models.

rmses — A list of root mean square error (RMSE) values of the fitted models.

hazardratio_val — Hazard ratio between two interventions (if applicable).

fits — A list of the fitted models for the time-varying covariates, outcome, and competing event (if applicable). If model_fits is set to FALSE, a value of NULL is given.

sim_data — A list of data tables of the simulated data. Each element in the list corresponds to one of the interventions. If the argument sim_data_b is set to FALSE, a value of NA is given.

IP_weights — A numeric vector specifying the inverse probability weights. See "Details".

bootests — A data.table containing the bootstrap replicates of the parametric g-formula estimates. If boot_diag is set to FALSE, a value of NULL is given.

bootcoeffs — A list, where the kth element is a list containing the coefficients of the fitted models corresponding to the kth bootstrap sample. If boot_diag is set to FALSE, a value of NULL is given.

bootstderrs — A list, where the kth element is a list containing the standard errors of the coefficients of the fitted models corresponding to the kth bootstrap sample. If boot_diag is set to FALSE, a value of NULL is given.

bootvcovs — A list, where the kth element is a list containing the variance-covariance matrices of the parameters of the fitted models corresponding to the kth bootstrap sample. If boot_diag is set to FALSE, a value of NULL is given.

... — Some additional elements.

The results for the g-formula simulation under various interventions only for the first and last time points are printed with the print.gformula_survival function. To generate graphs comparing the mean estimated covariate values and risks over time and mean observed covariate values and risks over time, use the plot.gformula_survival function.

References

Chiu YH, Wen L, McGrath S, Logan R, Dahabreh IJ, Hernán MA. Evaluating model specification when using the parametric g-formula in the presence of censoring. American Journal of Epidemiology. 2023;192:1887–1895.

McGrath S, Lin V, Zhang Z, Petito LC, Logan RW, Hernán MA, and JG Young. gfoRmula: An R package for estimating the effects of sustained treatment strategies via the parametric g-formula. Patterns. 2020;1:100008.

Robins JM. A new approach to causal inference in mortality studies with a sustained exposure period: application to the healthy worker survivor effect. Mathematical Modelling. 1986;7:1393–1512. [Errata (1987) in Computers and Mathematics with Applications 14, 917.-921. Addendum (1987) in Computers and Mathematics with Applications 14, 923-.945. Errata (1987) to addendum in Computers and Mathematics with Applications 18, 477.].

See Also

gformula

Examples

```
## Estimating the effect of static treatment strategies on risk of a
## failure event

id <- 'id'
time_points <- 7
time_name <- 't0'
covnames <- c('L1', 'L2', 'A')
outcome_name <- 'Y'
covtypes <- c('binary', 'bounded normal', 'binary')
histories <- c(lagged, lagavg)
histvars <- list(c('A', 'L1', 'L2'), c('L1', 'L2'))
covparams <- list(covmodels = c(L1 ~ lag1_A + lag_cumavg1_L1 + lag_cumavg1_L2 +
                                  L3 + t0,
                                L2 ~ lag1_A + L1 + lag_cumavg1_L1 +
                                  lag_cumavg1_L2 + L3 + t0,
                                A ~ lag1_A + L1 + L2 + lag_cumavg1_L1 +
                                  lag_cumavg1_L2 + L3 + t0))
ymodel <- Y ~ A + L1 + L2 + L3 + lag1_A + lag1_L1 + lag1_L2 + t0
intervention1.A <- list(static, rep(0, time_points))
intervention2.A <- list(static, rep(1, time_points))
```

```
int_descript <- c('Never treat', 'Always treat')
nsimul <- 10000

gform_basic <- gformula_survival(obs_data = basicdata_nocomp, id = id,
                                 time_points = time_points,
                                 time_name = time_name, covnames = covnames,
                                 outcome_name = outcome_name,
                                 covtypes = covtypes,
                                 covparams = covparams, ymodel = ymodel,
                                 intervention1.A = intervention1.A,
                                 intervention2.A = intervention2.A,
                                 int_descript = int_descript,
                                 histories = histories, histvars = histvars,
                                 basecovs = c('L3'), nsimul = nsimul,
                                 seed = 1234)
gform_basic



## Estimating the effect of treatment strategies on risk of a failure event
## when competing events exist

id <- 'id'
time_points <- 7
time_name <- 't0'
covnames <- c('L1', 'L2', 'A')
outcome_name <- 'Y'
compevent_name <- 'D'
covtypes <- c('binary', 'bounded normal', 'binary')
histories <- c(lagged, lagavg)
histvars <- list(c('A', 'L1', 'L2'), c('L1', 'L2'))
covparams <- list(covlink = c('logit', 'identity', 'logit'),
                  covmodels = c(L1 ~ lag1_A + lag_cumavg1_L1 + lag_cumavg1_L2 +
                                  L3 + as.factor(t0),
                                L2 ~ lag1_A + L1 + lag_cumavg1_L1 +
                                  lag_cumavg1_L2 + L3 + as.factor(t0),
                                A ~ lag1_A + L1 + L2 + lag_cumavg1_L1 +
                                  lag_cumavg1_L2 + L3 + as.factor(t0)))
ymodel <- Y ~ A + L1 + L2 + lag1_A + lag1_L1 + lag1_L2 + L3 + as.factor(t0)
compevent_model <- D ~ A + L1 + L2 + lag1_A + lag1_L1 + lag1_L2 + L3 + as.factor(t0)
intervention1.A <- list(static, rep(0, time_points))
intervention2.A <- list(static, rep(1, time_points))
int_descript <- c('Never treat', 'Always treat')
nsimul <- 10000

gform_basic <- gformula_survival(obs_data = basicdata, id = id,
                                 time_points = time_points,
                                 time_name = time_name, covnames = covnames,
                                 outcome_name = outcome_name,
                                 compevent_name = compevent_name,
                                 covtypes = covtypes,
                                 covparams = covparams, ymodel = ymodel,
                                 compevent_model = compevent_model,
```

```
                                intervention1.A = intervention1.A,
                                intervention2.A = intervention2.A,
                                int_descript = int_descript,
                                histories = histories, histvars = histvars,
                                basecovs = c('L3'), nsimul = nsimul,
                                seed = 1234)
gform_basic


## Using IP weighting to estimate natural course risk
## Only the natural course intervention is included for simplicity

covnames <- c('L', 'A')
histories <- c(lagged)
histvars <- list(c('A', 'L'))
ymodel <- Y ~ L + A
covtypes <- c('binary', 'normal')
covparams <- list(covmodels = c(L ~ lag1_L + lag1_A,
                                A ~ lag1_L + L + lag1_A))
censor_name <- 'C'
censor_model <- C ~ L
res_censor <- gformula(obs_data = censor_data, id = 'id',
                       time_name = 't0', covnames = covnames,
                       outcome_name = 'Y', outcome_type = 'survival',
                       censor_name = censor_name, censor_model = censor_model,
                       covtypes = covtypes,
                       covparams = covparams, ymodel = ymodel,
                       intvars = NULL, interventions = NULL, int_descript = NULL,
                       histories = histories, histvars = histvars,
                       seed = 1234)
plot(res_censor)
```

---

lagged *History functions*

Description

These functions create new columns in an input data table for covariate histories. Users must specify which covariates are to be used in the history functions.

Usage

```
lagged(
  pool,
  histvars,
  histvals,
  time_name,
  t,
```

```
  id_name,
  baselags,
  below_zero_indicator
)

cumavg(pool, histvars, time_name, t, id_name, below_zero_indicator)

lagavg(
  pool,
  histvars,
  histvals,
  time_name,
  t,
  id_name,
  baselags,
  below_zero_indicator
)
```

Arguments

pool	Data table containing all information prior to time t (t noninclusive).
histvars	Vector of character strings specifying the names of the variables for which history functions are to be applied.
histvals	For lagged, this argument is a vector specifying the lags used in the model statements (e.g., if lag1_varname and lag2_varname were included in the model statements, this vector would be c(1,2)). For lagavg, this argument is a numeric vector specifying the lag averages used in the model statements.
time_name	Character string specifying the name of the time variable in pool.
t	Integer specifying the current time index.
id_name	Character string specifying the name of the ID variable in pool.
baselags	Logical scalar for specifying the convention used for lagi and lag_cumavgi terms in the model statements when pre-baseline times are not included in obs_data and when the current time index, t, is such that $t < i$. If this argument is set to FALSE, the value of all lagi and lag_cumavgi terms in this context are set to 0 (for non-categorical covariates) or the reference level (for categorical covariates). If this argument is set to TRUE, the value of lagi and lag_cumavgi terms are set to their values at time 0. The default is FALSE.
below_zero_indicator	Logical scalar indicating whether the observed data set contains rows for time $t < 0$.

Details

lagged creates new columns for lagged versions of existing variables in the dataset. The user must specify which variables are to be lagged.

cumavg creates new columns for the cumulative average up until time t of existing variables in the dataset.

lagavg creates new columns for the "lagged cumulative average" (cumulative average up until time t, then lagged by one time unit) up until time t of existing variables in the dataset.

Value

No value is returned. The data table pool is modified in place.

Examples

```
## Estimating the effect of static treatment strategies on risk of a
## failure event

id <- 'id'
time_points <- 7
time_name <- 't0'
covnames <- c('L1', 'L2', 'A')
outcome_name <- 'Y'
outcome_type <- 'survival'
covtypes <- c('binary', 'bounded normal', 'binary')
histories <- c(lagged, lagavg)
histvars <- list(c('A', 'L1', 'L2'), c('L1', 'L2'))
covparams <- list(covmodels = c(L1 ~ lag1_A + lag_cumavg1_L1 + lag_cumavg1_L2 +
                                  L3 + t0,
                                L2 ~ lag1_A + L1 + lag_cumavg1_L1 +
                                  lag_cumavg1_L2 + L3 + t0,
                                A ~ lag1_A + L1 + L2 + lag_cumavg1_L1 +
                                  lag_cumavg1_L2 + L3 + t0))
ymodel <- Y ~ A + L1 + L2 + L3 + lag1_A + lag1_L1 + lag1_L2 + t0
intervention1.A <- list(static, rep(0, time_points))
intervention2.A <- list(static, rep(1, time_points))
int_descript <- c('Never treat', 'Always treat')
nsimul <- 10000

gform_basic <- gformula(obs_data = basicdata_nocomp, id = id,
                        time_points = time_points,
                        time_name = time_name, covnames = covnames,
                        outcome_name = outcome_name,
                        outcome_type = outcome_type, covtypes = covtypes,
                        covparams = covparams, ymodel = ymodel,
                        intervention1.A = intervention1.A,
                        intervention2.A = intervention2.A,
                        int_descript = int_descript,
                        histories = histories, histvars = histvars,
                        basecovs = c('L3'), nsimul = nsimul,
                        seed = 1234)
gform_basic
```

plot.gformula_binary_eof *Plot method for objects of class "gformula_binary_eof"*

Description

This function generates graphs of the mean simulated vs. observed values at each time point of the time-varying covariates under the natural course. For categorical covariates, the observed and simulated probability of each level are plotted at each time point.

Usage

```
## S3 method for class 'gformula_binary_eof'
plot(
  x,
  covnames = NULL,
  ncol = NULL,
  nrow = NULL,
  common.legend = TRUE,
  legend = "bottom",
  xlab = NULL,
  ylab_cov = NULL,
  ...
)
```

Arguments

x	Object of class "gformula_binary_eof".
covnames	Vector of character strings specifying the names of the time-varying covariates to be plotted. The ordering of covariates given here is used in the plot grid. Time-varying covariates of type "categorical time" cannot be included. By default, this argument is set equal to the covnames argument used in gformula_binary_eof, where covariates of type "categorical time" are removed.
ncol	Number of columns in the plot grid. By default, two columns are used when there is at least two plots.
nrow	Number of rows in the plot grid. By default, a maximum of six rows is used and additional plots are included in subsequent pages.
common.legend	Logical scalar indicating whether to include a legend. The default is TRUE.
legend	Character string specifying the legend position. Valid values are "top", "bottom", "left", "right", and "none". The default is "bottom".
xlab	Character string for the x axes of all plots. By default, this argument is set to the time_name argument specified in gformula_binary_eof.
ylab_cov	Vector of character strings for the y axes of the plots for the covariates. This argument must be the same length as covnames. The i-th element of this argument corresponds to the plot for the i-th element of covnames.
...	Other arguments, which are passed to ggarrange.

Value

An object of class "ggarrange". See documentation of ggarrange.

See Also

gformula_binary_eof

Examples

```
## Estimating the effect of threshold interventions on the mean of a binary
## end of follow-up outcome

outcome_type <- 'binary_eof'
id <- 'id_num'
time_name <- 'time'
covnames <- c('cov1', 'cov2', 'treat')
outcome_name <- 'outcome'
histories <- c(lagged, cumavg)
histvars <- list(c('treat', 'cov1', 'cov2'), c('cov1', 'cov2'))
covtypes <- c('binary', 'zero-inflated normal', 'normal')
covparams <- list(covmodels = c(cov1 ~ lag1_treat + lag1_cov1 + lag1_cov2 +
                                  cov3 + time,
                                cov2 ~ lag1_treat + cov1 + lag1_cov1 +
                                  lag1_cov2 + cov3 + time,
                                treat ~ lag1_treat + cumavg_cov1 +
                                  cumavg_cov2 + cov3 + time))
ymodel <- outcome ~  treat + cov1 + cov2 + lag1_cov1 + lag1_cov2 + cov3
intervention1.treat <- list(static, rep(0, 7))
intervention2.treat <- list(threshold, 1, Inf)
int_descript <- c('Never treat', 'Threshold - lower bound 1')
nsimul <- 10000
ncores <- 2

gform_bin_eof <- gformula(obs_data = binary_eofdata,
                          outcome_type = outcome_type, id = id,
                          time_name = time_name, covnames = covnames,
                          outcome_name = outcome_name, covtypes = covtypes,
                          covparams = covparams, ymodel = ymodel,
                          intervention1.treat = intervention1.treat,
                          intervention2.treat = intervention2.treat,
                          int_descript = int_descript, histories = histories,
                          histvars = histvars, basecovs = c("cov3"),
                          seed = 1234, parallel = TRUE, nsamples = 5,
                          nsimul = nsimul, ncores = ncores)
plot(gform_bin_eof)
```

plot.gformula_continuous_eof *Plot method for objects of class "gformula_continuous_eof"*

Description

This function generates graphs of the mean simulated vs. observed values at each time point of the time-varying covariates under the natural course. For categorical covariates, the observed and simulated probability of each level are plotted at each time point.

Usage

```
## S3 method for class 'gformula_continuous_eof'
plot(
  x,
  covnames = NULL,
  ncol = NULL,
  nrow = NULL,
  common.legend = TRUE,
  legend = "bottom",
  xlab = NULL,
  ylab_cov = NULL,
  ...
)
```

Arguments

x	Object of class "gformula_continuous_eof".
covnames	Vector of character strings specifying the names of the time-varying covariates to be plotted. The ordering of covariates given here is used in the plot grid. Time-varying covariates of type "categorical time" cannot be included. By default, this argument is set equal to the covnames argument used in gformula_continuous_eof, where covariates of type "categorical time" are removed.
ncol	Number of columns in the plot grid. By default, two columns are used when there is at least two plots.
nrow	Number of rows in the plot grid. By default, a maximum of six rows is used and additional plots are included in subsequent pages.
common.legend	Logical scalar indicating whether to include a legend. The default is TRUE.
legend	Character string specifying the legend position. Valid values are "top", "bottom", "left", "right", and "none". The default is "bottom".
xlab	Character string for the x axes of all plots. By default, this argument is set to the time_name argument specified in gformula_continuous_eof.
ylab_cov	Vector of character strings for the y axes of the plots for the covariates. This argument must be the same length as covnames. The i-th element of this argument corresponds to the plot for the i-th element of covnames.
...	Other arguments, which are passed to ggarrange.

Value

An object of class "ggarrange". See documentation of ggarrange.

See Also

gformula_continuous_eof

Examples

```
## Estimating the effect of treatment strategies on the mean of a continuous
## end of follow-up outcome

library('Hmisc')
id <- 'id'
time_name <- 't0'
covnames <- c('L1', 'L2', 'A')
outcome_name <- 'Y'
outcome_type <- 'continuous_eof'
covtypes <- c('categorical', 'normal', 'binary')
histories <- c(lagged)
histvars <- list(c('A', 'L1', 'L2'))
covparams <- list(covmodels = c(L1 ~ lag1_A + lag1_L1 + L3 + t0 +
                                  rcspline.eval(lag1_L2, knots = c(-1, 0, 1)),
                                L2 ~ lag1_A + L1 + lag1_L1 + lag1_L2 + L3 + t0,
                                A ~ lag1_A + L1 + L2 + lag1_L1 + lag1_L2 + L3 + t0))
ymodel <- Y ~ A + L1 + L2 + lag1_A + lag1_L1 + lag1_L2 + L3
intervention1.A <- list(static, rep(0, 7))
intervention2.A <- list(static, rep(1, 7))
int_descript <- c('Never treat', 'Always treat')
nsimul <- 10000

gform_cont_eof <- gformula(obs_data = continuous_eofdata,
                           id = id, time_name = time_name,
                           covnames = covnames, outcome_name = outcome_name,
                           outcome_type = outcome_type, covtypes = covtypes,
                           covparams = covparams, ymodel = ymodel,
                           intervention1.A = intervention1.A,
                           intervention2.A = intervention2.A,
                           int_descript = int_descript,
                           histories = histories, histvars = histvars,
                           basecovs = c("L3"), nsimul = nsimul, seed = 1234)
plot(gform_cont_eof)
```

---

plot.gformula_survival

Plot method for objects of class "gformula_survival"

Description

This function generates graphs of the mean simulated vs. observed values at each time point of the time-varying covariates, risk, and survival under the natural course. For categorical covariates, the observed and simulated probability of each level are plotted at each time point.

Usage

```
## S3 method for class 'gformula_survival'
plot(
  x,
  covnames = NULL,
  risk = TRUE,
  survival = FALSE,
  ncol = NULL,
  nrow = NULL,
  common.legend = TRUE,
  legend = "bottom",
  xlab = NULL,
  ylab_cov = NULL,
  ylab_risk = "risk",
  ylab_surv = "survival",
  pos_risk = NULL,
  pos_surv = NULL,
  ci_risk = FALSE,
  ...
)
```

Arguments

x	Object of class "gformula_survival".
covnames	Vector of character strings specifying the names of the time-varying covariates to be plotted. The ordering of covariates given here is used in the plot grid. Time-varying covariates of type "categorical time" cannot be included. To plot none of the time-varying covariates, set this argument to NA. By default, this argument is set equal to the covnames argument used in gformula_survival, where covariates of type 'categorical time' are removed.
risk	Logical scalar indicating whether to include a plot for the risk. The default is TRUE.
survival	Logical scalar indicating whether to include a plot for the survival. The default is FALSE.
ncol	Number of columns in the plot grid. By default, two columns are used when there is at least two plots.
nrow	Number of rows in the plot grid. By default, a maximum of six rows is used and additional plots are included in subsequent pages.
common.legend	Logical scalar indicating whether to include a legend. The default is TRUE.
legend	Character string specifying the legend position. Valid values are "top", "bottom", "left", "right", and "none". The default is "bottom".

xlab	Character string for the x axes of all plots. By default, this argument is set to the time_name argument specified in gformula_survival.
ylab_cov	Vector of character strings for the y axes of the plots for the covariates. This argument must be the same length as covnames. The i-th element of this argument corresponds to the plot for the i-th element of covnames.
ylab_risk	Character string for the y axis of the plot for the risk (if applicable). The default is "risk".
ylab_surv	Character string for the y axis of the plot for the survival (if applicable). The default is "survival".
pos_risk	Integer specifying the position at which to order the risk plot (if applicable). By default, this argument is set to the number of plots in the grid minus one (i.e., orders the risk plot second last).
pos_surv	Integer specifying the position at which to order the survival plot (if applicable). By default, this argument is set to the number of plots in the grid (i.e., orders the survival plot last).
ci_risk	Logical scalar specifying whether to include error bars for the 95% confidence intervals of the estimated risk under the natural course. This argument is only effective if the argument nsamples was set to a positive value in gformula_survival. The default is TRUE.
...	Other arguments, which are passed to ggarrange.

Value

An object of class "ggarrange". See documentation of ggarrange.

See Also

gformula_survival

Examples

```
## Estimating the effect of static treatment strategies on risk of a
## failure event

id <- 'id'
time_points <- 7
time_name <- 't0'
covnames <- c('L1', 'L2', 'A')
outcome_name <- 'Y'
outcome_type <- 'survival'
covtypes <- c('binary', 'bounded normal', 'binary')
histories <- c(lagged, lagavg)
histvars <- list(c('A', 'L1', 'L2'), c('L1', 'L2'))
covparams <- list(covmodels = c(L1 ~ lag1_A + lag_cumavg1_L1 + lag_cumavg1_L2 +
                                  L3 + t0,
                                L2 ~ lag1_A + L1 + lag_cumavg1_L1 +
                                  lag_cumavg1_L2 + L3 + t0,
                                A ~ lag1_A + L1 + L2 + lag_cumavg1_L1 +
                                  lag_cumavg1_L2 + L3 + t0))
```

```
ymodel <- Y ~ A + L1 + L2 + L3 + lag1_A + lag1_L1 + lag1_L2 + t0
intervention1.A <- list(static, rep(0, time_points))
intervention2.A <- list(static, rep(1, time_points))
int_descript <- c('Never treat', 'Always treat')
nsimul <- 10000

gform_basic <- gformula(obs_data = basicdata_nocomp, id = id,
                        time_points = time_points,
                        time_name = time_name, covnames = covnames,
                        outcome_name = outcome_name,
                        outcome_type = outcome_type, covtypes = covtypes,
                        covparams = covparams, ymodel = ymodel,
                        intervention1.A = intervention1.A,
                        intervention2.A = intervention2.A,
                        int_descript = int_descript,
                        histories = histories, histvars = histvars,
                        basecovs = c('L3'), nsimul = nsimul,
                        seed = 1234)
plot(gform_basic)
```

`print.gformula_survival` *Print and summary methods for "gformula" objects*

Description

Print and summary method for objects of class "gformula_survival", "gformula_continuous_eof", or "gformula_binary_eof".

Usage

```
## S3 method for class 'gformula_survival'
print(
  x,
  all_times = FALSE,
  coefficients = FALSE,
  stderrs = FALSE,
  rmses = FALSE,
  hazardratio = FALSE,
  fits = FALSE,
  ...
)

## S3 method for class 'gformula_continuous_eof'
print(
  x,
```

```
  coefficients = FALSE,
  stderrs = FALSE,
  rmses = FALSE,
  fits = FALSE,
  ...
)

## S3 method for class 'gformula_binary_eof'
print(
  x,
  coefficients = FALSE,
  stderrs = FALSE,
  rmses = FALSE,
  fits = FALSE,
  ...
)

## S3 method for class 'gformula'
summary(object, ...)

## S3 method for class 'summary.gformula'
print(
  x,
  all_times = TRUE,
  coefficients = FALSE,
  stderrs = FALSE,
  rmses = FALSE,
  hazardratio = FALSE,
  fits = TRUE,
  ...
)
```

Arguments

x	Object of class "gformula_survival", "gformula_continuous_eof", "gformula_binary_eof", or "summary.gformula" (for print).
all_times	Logical scalar indicating whether to print the results for all time points. This argument is only applicable to objects of class "gformula_survival". If this argument is set to FALSE, the results are only shown for the final time point. The default is FALSE for print and TRUE for summary.
coefficients	Logical scalar indicating whether to print the model coefficients. The default is FALSE.
stderrs	Logical scalar indicating whether to print the standard error of the model coefficients. The default is FALSE.
rmses	Logical scalar indicating whether to print the model root mean square errors (RMSEs). The default is FALSE.
hazardratio	Logical scalar indicating whether to print the hazard ratio between two interventions (if computed). If bootstrapping was used, 95% confidence intervals will be

given. This argument is only applicable to objects of class "gformula_survival". The default is FALSE.

fits — Logical scalar indicating whether to print summaries of the fitted models for the time-varying covariates, outcome, and competing event (if applicable). This argument is only effective if the argument model_fits was set to TRUE in gformula. The default is FALSE for print and TRUE for summary.

... — Other arguments.

object — Object of class "gformula" (for summary).

Value

No value is returned for the print functions. The summary function returns the object passed to it and adds the class "summary.gformula" to it.

See Also

gformula

Examples

```
## Estimating the effect of static treatment strategies on risk of a
## failure event

id <- 'id'
time_points <- 7
time_name <- 't0'
covnames <- c('L1', 'L2', 'A')
outcome_name <- 'Y'
outcome_type <- 'survival'
covtypes <- c('binary', 'bounded normal', 'binary')
histories <- c(lagged, lagavg)
histvars <- list(c('A', 'L1', 'L2'), c('L1', 'L2'))
covparams <- list(covmodels = c(L1 ~ lag1_A + lag_cumavg1_L1 + lag_cumavg1_L2 +
                                  L3 + t0,
                                L2 ~ lag1_A + L1 + lag_cumavg1_L1 +
                                  lag_cumavg1_L2 + L3 + t0,
                                A ~ lag1_A + L1 + L2 + lag_cumavg1_L1 +
                                  lag_cumavg1_L2 + L3 + t0))
ymodel <- Y ~ A + L1 + L2 + L3 + lag1_A + lag1_L1 + lag1_L2 + t0
intervention1.A <- list(static, rep(0, time_points))
intervention2.A <- list(static, rep(1, time_points))
int_descript <- c('Never treat', 'Always treat')
nsimul <- 10000

gform_basic <- gformula(obs_data = basicdata_nocomp, id = id,
                        time_points = time_points,
                        time_name = time_name, covnames = covnames,
                        outcome_name = outcome_name,
                        outcome_type = outcome_type, covtypes = covtypes,
                        covparams = covparams, ymodel = ymodel,
                        intervention1.A = intervention1.A,
```

```
                        intervention2.A = intervention2.A,
                        int_descript = int_descript,
                        histories = histories, histvars = histvars,
                        basecovs = c('L3'), nsimul = nsimul,
                        seed = 1234)
summary(gform_basic)
```

---

simple_restriction *Simple Restriction*

Description

This function assists the implementation of a restriction on a covariate in the data table newdf by setting lines where the covariate is restricted to a user-specified value.

Usage

```
simple_restriction(newdf, pool, restriction, time_name, t, ...)
```

Arguments

newdf	Data table containing the simulated data at time t.
pool	Data table containing the simulated data at times before t.
restriction	List of vectors. Each vector contains as its first entry the covariate affected by the restriction; its second entry the condition that must be TRUE for the covariate to be modeled; its third entry a function that executes other specific actions based on the condition (in this case, this function); and its fourth entry some value used by the function (in this case, the value the user desires to assign to the covariate when it is not modeled).
time_name	Character string specifying the name of the time variable in pool and newdf.
t	Integer specifying the current time index.
...	This argument is not used in this function.

Value

No value is returned. The data table newdf is modified in place.

Examples

```
## Estimating the effect of static treatment strategies on risk of a
## failure event with a simple restriction

id <- 'id'
time_points <- 7
time_name <- 't0'
```

```
covnames <- c('L1', 'L2', 'A')
outcome_name <- 'Y'
outcome_type <- 'survival'
covtypes <- c('binary', 'bounded normal', 'binary')
histories <- c(lagged, lagavg)
histvars <- list(c('A', 'L1', 'L2'), c('L1', 'L2'))
covparams <- list(covmodels = c(L1 ~ lag1_A + lag_cumavg1_L1 + lag_cumavg1_L2 +
                                  L3 + t0,
                                L2 ~ lag1_A + L1 + lag_cumavg1_L1 +
                                  lag_cumavg1_L2 + L3 + t0,
                                A ~ lag1_A + L1 + L2 + lag_cumavg1_L1 +
                                  lag_cumavg1_L2 + L3 + t0))
ymodel <- Y ~ A + L1 + L2 + L3 + lag1_A + lag1_L1 + lag1_L2 + t0
intervention1.A <- list(static, rep(0, time_points))
intervention2.A <- list(static, rep(1, time_points))
int_descript <- c('Never treat', 'Always treat')
nsimul <- 10000

# At t0 == 5, assume we have deterministic knowledge that L1 equals 0
restrictions <- list(c('L1', 't0 != 5', simple_restriction, 0))

gform_basic <- gformula(obs_data = basicdata_nocomp, id = id,
                        time_points = time_points,
                        time_name = time_name, covnames = covnames,
                        outcome_name = outcome_name,
                        outcome_type = outcome_type, covtypes = covtypes,
                        covparams = covparams, ymodel = ymodel,
                        intervention1.A = intervention1.A,
                        intervention2.A = intervention2.A,
                        restrictions = restrictions,
                        int_descript = int_descript,
                        histories = histories, histvars = histvars,
                        basecovs = c('L3'), nsimul = nsimul,
                        seed = 1234)
gform_basic
```

`static` *Static Intervention*

Description

This function implements a static intervention (i.e., either constant treatment or no treatment over all time points) for the specified intervention variable in the data table `newdf`.

Usage

```
static(newdf, pool, intvar, intvals, time_name, t)
```

Arguments

newdf	Data table containing the simulated data at time t.
pool	Data table containing the simulated data at times before t.
intvar	Character string specifying the name of the variable to be intervened on in each round of the simulation.
intvals	A list of length 1. The entry is the value of static treatment to be assigned to intvar.
time_name	Character string specifying the name of the time variable in pool and newdf.
t	Integer specifying the current time index.

Value

No value is returned. The data table newdf is modified in place.

Examples

```
## Estimating the effect of static treatment strategies on risk of a
## failure event

id <- 'id'
time_points <- 7
time_name <- 't0'
covnames <- c('L1', 'L2', 'A')
outcome_name <- 'Y'
outcome_type <- 'survival'
covtypes <- c('binary', 'bounded normal', 'binary')
histories <- c(lagged, lagavg)
histvars <- list(c('A', 'L1', 'L2'), c('L1', 'L2'))
covparams <- list(covmodels = c(L1 ~ lag1_A + lag_cumavg1_L1 + lag_cumavg1_L2 +
                                  L3 + t0,
                                L2 ~ lag1_A + L1 + lag_cumavg1_L1 +
                                  lag_cumavg1_L2 + L3 + t0,
                                A ~ lag1_A + L1 + L2 + lag_cumavg1_L1 +
                                  lag_cumavg1_L2 + L3 + t0))
ymodel <- Y ~ A + L1 + L2 + L3 + lag1_A + lag1_L1 + lag1_L2 + t0
intervention1.A <- list(static, rep(0, time_points))
intervention2.A <- list(static, rep(1, time_points))
int_descript <- c('Never treat', 'Always treat')
nsimul <- 10000

gform_basic <- gformula(obs_data = basicdata_nocomp, id = id,
                        time_points = time_points,
                        time_name = time_name, covnames = covnames,
                        outcome_name = outcome_name,
                        outcome_type = outcome_type, covtypes = covtypes,
                        covparams = covparams, ymodel = ymodel,
                        intervention1.A = intervention1.A,
                        intervention2.A = intervention2.A,
                        int_descript = int_descript,
                        histories = histories, histvars = histvars,
```

```
                        basecovs = c('L3'), nsimul = nsimul,
                        seed = 1234)
gform_basic
```

---

threshold *Threshold Intervention*

Description

This function implements a threshold intervention (i.e., once treatment bypasses a certain threshold, it remains at that threshold until end of follow-up) for the specified intervention variable in the data table newdf.

Usage

```
threshold(newdf, pool, intvar, intvals, time_name, t)
```

Arguments

newdf	Data table containing the simulated data at time t.
pool	Data table containing the simulated data at times before t.
intvar	Character string specifying the name of the variable to be intervened on in each round of the simulation.
intvals	A list of length 2. The first entry is lower bound of the threshold, and the second entry is the upper bound.
time_name	Character string specifying the name of the time variable in pool and newdf.
t	Integer specifying the current time index.

Value

No value is returned. The data table newdf is modified in place.

Examples

```
## Estimating the effect of threshold interventions on the mean of a binary
## end of follow-up outcome

id <- 'id_num'
time_name <- 'time'
covnames <- c('cov1', 'cov2', 'treat')
outcome_name <- 'outcome'
histories <- c(lagged, cumavg)
histvars <- list(c('treat', 'cov1', 'cov2'), c('cov1', 'cov2'))
covtypes <- c('binary', 'zero-inflated normal', 'normal')
covparams <- list(covmodels = c(cov1 ~ lag1_treat + lag1_cov1 + lag1_cov2 + cov3 +
                                  time,
```

```
                                cov2 ~ lag1_treat + cov1 + lag1_cov1 + lag1_cov2 +
                                  cov3 + time,
                                treat ~ lag1_treat + cumavg_cov1 +
                                  cumavg_cov2 + cov3 + time))
ymodel <- outcome ~  treat + cov1 + cov2 + lag1_cov1 + lag1_cov2 + cov3
intervention1.treat <- list(static, rep(0, 7))
intervention2.treat <- list(threshold, 1, Inf)
int_descript <- c('Never treat', 'Threshold - lower bound 1')
nsimul <- 10000
ncores <- 2

gform_bin_eof <- gformula_binary_eof(obs_data = binary_eofdata, id = id,
                                     time_name = time_name,
                                     covnames = covnames,
                                     outcome_name = outcome_name,
                                     covtypes = covtypes,
                                     covparams = covparams,
                                     ymodel = ymodel,
                                     intervention1.treat = intervention1.treat,
                                     intervention2.treat = intervention2.treat,
                                     int_descript = int_descript,
                                     histories = histories, histvars = histvars,
                                     basecovs = c("cov3"), seed = 1234,
                                     parallel = TRUE, nsamples = 5,
                                     nsimul = nsimul, ncores = ncores)
gform_bin_eof
```

`vcov.gformula` *Variance-covariance method for objects of class "gformula"*

Description

This function extracts the variance-covariance matrices of the parameters of the fitted models for the time-varying covariates, outcome, and competing event (if applicable).

Usage

```
## S3 method for class 'gformula'
vcov(object, ...)
```

Arguments

`object`	Object of class "gformula".
`...`	Other arguments.

Value

If bootdiag was set to FALSE in gformula, this function returns a list of the variance-covariance matrices of the parameters of the fitted models to the observed data set. If bootstrapping was used and bootdiag was set to TRUE in gformula, this function returns a list described as follows. The first element (named 'Original sample') is a list of the variance-covariance matrices of the parameters of the fitted models to the observed data set. The kth element (named 'Bootstrap sample k-1') is a list of the variance-covariance matrices of the parameters of the fitted models corresponding to the k-1th bootstrap sample.

See Also

gformula

Examples

```
## Estimating the effect of static treatment strategies on risk of a
## failure event

id <- 'id'
time_points <- 7
time_name <- 't0'
covnames <- c('L1', 'L2', 'A')
outcome_name <- 'Y'
outcome_type <- 'survival'
covtypes <- c('binary', 'bounded normal', 'binary')
histories <- c(lagged, lagavg)
histvars <- list(c('A', 'L1', 'L2'), c('L1', 'L2'))
covparams <- list(covmodels = c(L1 ~ lag1_A + lag_cumavg1_L1 + lag_cumavg1_L2 +
                                  L3 + t0,
                                L2 ~ lag1_A + L1 + lag_cumavg1_L1 +
                                  lag_cumavg1_L2 + L3 + t0,
                                A ~ lag1_A + L1 + L2 + lag_cumavg1_L1 +
                                  lag_cumavg1_L2 + L3 + t0))
ymodel <- Y ~ A + L1 + L2 + L3 + lag1_A + lag1_L1 + lag1_L2 + t0
intervention1.A <- list(static, rep(0, time_points))
intervention2.A <- list(static, rep(1, time_points))
int_descript <- c('Never treat', 'Always treat')
nsimul <- 10000

gform_basic <- gformula(obs_data = basicdata_nocomp, id = id,
                        time_points = time_points,
                        time_name = time_name, covnames = covnames,
                        outcome_name = outcome_name,
                        outcome_type = outcome_type, covtypes = covtypes,
                        covparams = covparams, ymodel = ymodel,
                        intervention1.A = intervention1.A,
                        intervention2.A = intervention2.A,
                        int_descript = int_descript,
                        histories = histories, histvars = histvars,
                        basecovs = c('L3'), nsimul = nsimul,
                        seed = 1234)
```

```
vcov(gform_basic)
```

Index