Package ‘CICI’

January 7, 2026

Type Package

Title Causal Inference with Continuous (Multiple Time Point) Interventions

Version 0.9.8

Date 2025-12-19

Maintainer Michael Schomaker <michael.schomaker@stat.uni-muenchen.de>

Description Estimation of counterfactual outcomes for multiple values of continuous interventions at different time points, and plotting of causal dose-response curves. Details are given in Schomaker, McIlleron, Denti, Diaz (2024) <doi:10.48550/arXiv.2305.06645>.

Depends R (>= 4.0)

Imports mgcv, glmnet, ggplot2, parallel, doParallel, foreach, doRNG, rngtools, SuperLearner, survival

Suggests haldensify, hal9001

License GPL-2

NeedsCompilation no

Author Michael Schomaker [aut, cre],
Leo Fuhrhop [ctb],
Han Bao [ctb]

Repository CRAN

Date/Publication 2026-01-07 21:30:18 UTC

Contents

`CICI-package` *Causal Inference with Continuous (Multiple Time Point) Interventions*

Description

This package facilitates the estimation of counterfactual outcomes for multiple values of continuous interventions at different time points, and allows plotting of causal dose-response curves.

It offers implementations of both the (semi-)parametric *g*-formula and the sequential *g*-computation formula. Positivity violations can be detected with diagnostics, and addressed either through *feasible* intervention strategies, or outcome weights. Details are given in Schomaker et al. (2025) and Bao and Schomaker (2025), see references below.

Details

Package:	CICI
Type:	Package
Version:	0.9.8
Date:	2025-12-19
License:	GPL-2
Depends:	R (>= 4.0)
Imports:	mgcv, glmnet, ggplot2, parallel, doParallel, foreach, doRNG, rngtools, SuperLearner, survival
Suggests:	haldensify, hal9001

Author(s)

Michael Schomaker

Maintainer: Michael Schomaker <michael.schomaker@stat.uni-muenchen.de>

References

Schomaker M, McIlleron H, Denti P, Diaz I. (2024) *Causal Inference for Continuous Multiple Time Point Interventions*, Statistics in Medicine, 43:5380-5400, see also *https://arxiv.org/abs/2305.06645*.

Bao H, Schomaker M (2025) *Addressing Positivity Violations in Continuous Interventions through Data-Adaptive Strategies*, arXiv ePrint, *https://arxiv.org/abs/2502.14566*.

`calc.weights` *Calculate outcome weights to address positivity violations*

Description

The weights are calculated according to formula (14) in Schomaker et al. (2023).

Usage

```
calc.weights(X, Anodes = NULL, Ynodes = NULL, Lnodes = NULL, Cnodes = NULL,
             abar = NULL, times = length(Anodes), c = 0.01, screen = FALSE,
             survival = FALSE, eps = 1e-10, zero = 0,
          d.method = c("binning", "parametric", "hal_density", "hazardbinning"),
             z.method = c("density", "eps"), w.function = "gal_ga",
             for.sgf = TRUE,
             verbose = TRUE, ...)
```

Arguments

`X`	A data frame, following the time-ordering of the variables.
`Anodes`	A character string of column names in `X` of the intervention variable(s).
`Ynodes`	A character string of column names in `X` of the outcome variable(s).
`Lnodes`	A character string of column names in `X` of *all* confounders, both baseline and time-varying.
`Cnodes`	A character string of column names in `X` of the censoring variable(s).
`abar`	Numeric vector or matrix of intervention values. See Details.
`times`	Numeric value specifying for how many time points the weights should be calculated.
`c`	A numeric value (or vector) specifying the threshold(s) below which the weights correspond to the density ratios, rather than 1.
`screen`	Logical. If `TRUE`, variable screening with LASSO is performed prior to estimating the conditional densities for the weights.
`survival`	Logical. If `TRUE`, a survival setting is assumed and taken into account for model specification.
`eps`	A numeric value specifying epsilon if `z.method="eps"`. See details.
`zero`	A numeric value specifying which actual number is considered to be "zero" in the denominator.

`d.method`	A string specifying which method should be used to estimate the conditional density. One of `"binning"`, `"parametric"`, `"hal_density"`.
`z.method`	A string specifying the method which should be used if the denominator is zero: The default is `"density"`, which modifies the density according to formula (14) in Schomaker et al. (2023). Alternatively, `"eps"` replaces the denominator with a fixed value specified under `eps`.
`w.function`	A string specifying the weight function which specifies how the conditional densities from numerator and denominator as well as `c` should be combined. Currently, `"gal_ga"` equates to formula (14) of Schomaker et al. (2023); whereas `"gal_ga2"` equates to the same weights, but where the cutoff for `c` to define positivity violations is not based on the conditional treatment density alone, but relative to the density defined in the denominator.
`for.sgf`	Logical. If TRUE, weights are organized such that they fit the order required for `sgf`.
`verbose`	Logical. If `TRUE`, notes and warnings are printed.
`...`	Further arguments to be passed on.

Details

Calculates the outcome weights as described in formula (14) of Schomaker et al. (2023).

If `d.method="parametric"`, parametric conditional density estimation with generalized (additive models) is used. Under `d.method="binning"` the continuous intervention is categorized into `length(abar)` bins, and then logistic (additive) models are used to approximate the conditional treatment density. The method `d.method="hal_density"` estimates the conditional treatment density non-parametrically using highly adaptive LASSO density estimation, as implemented in **haldensify**. This option is experimental so far, and may take long, especially if the sample size is large.

In survival settings, past censoring and outcome nodes are omitted from the formulae. If censoring is present without a survival setting (e.g. Cnodes describe drop-outs and Y is a continuous outcome), then survival should be set as FALSE.

Value

An object of class `Yweights`. This is a named list of length of `c`; each list entry is another list of length(number of time points); each entry is a matrix of size n times I (n=sample size; I=number of Interventions).

Author(s)

Michael Schomaker

References

Schomaker M, McIlleron H, Denti P, Diaz I. (2023) *Causal Inference for Continuous Multiple Time Point Interventions*, ArXiv e-prints: *https://arxiv.org/abs/2305.06645*.

Examples

```
data(EFV)

w <- calc.weights(X=EFV, Lnodes  = c("sex", "metabolic",
                                     "log_age", "NRTI" ,"weight.0",
                                     "adherence.1","weight.1",
                                     "adherence.2","weight.2",
                                     "adherence.3","weight.3",
                                     "adherence.4","weight.4"),
                         Ynodes  = c("VL.0","VL.1","VL.2","VL.3","VL.4"),
                         Anodes  = c("efv.0","efv.1","efv.2","efv.3","efv.4"),
                         d.method="parametric", abar=seq(0,5,1), c=0.01)

summary(w)
# w can now be used under 'Yweights' in sgf()
```

---

contrast — *Counterfactual contrast from the parametric or sequential g-formula for continuous multiple time point interventions*

Description

Estimation of a contrast between counterfactual outcomes under different values of (continuous) interventions, or across different time points, using the parametric or sequential g-formula.

Usage

```
contrast(X, abar, nodes, contrastType = "difference", measure = mean,
         cond = NULL, cilevel = 0.95, ...)
```

Arguments

`X`	An object of class `gformula` produced by `gformula`, with option `ret` = `TRUE`, or `sgf`.
`abar`	Numeric vector or the string 'natural'. Specifies the intervention value(s) for the contrast. If two values are given, a contrast between these two intervention regimes is computed at the same outcome node. If a single value is given and `nodes` has two elements, a contrast between time points is computed under that intervention. If more than two entries are given, `contrastType` must be a custom function. See Details.
`nodes`	A character string vector specifying the variable(s) used in the contrast. If two values are given, a temporal contrast is computed (e.g., outcome change over time under the same intervention level). If more than two entries are given, `contrastType` must be a custom function. See Details.

`contrastType`	Type of contrast to compute between the counterfactual measures. Accepts one of 'difference', 'ratio', 'oddsratio', or a user-defined function taking `length(abar) * length(nodes)` numeric arguments and returning a numeric value. The default is 'difference'.
`measure`	Specifies the summary measure applied to the post-intervention counterfactual data. Defaults to `mean`.
`cond`	Optional filtering condition(s) applied to the post-intervention counterfactual data. Must be a quoted expression, e.g., `cond = quote(sex == 1)`, or a list of quoted expressions, e.g., `cond = list(quote(sex == 1), quote(sex == 0))`.
`cilevel`	Numeric value between 0 and 1 specifying the confidence level of the bootstrap confidence intervals. Defaults to 95%.
`...`	Additional arguments to be passed to `measure`.

Details

Causal effects are defined as contrasts between the distributions of counterfactual variables under different interventions, across different time points or across different covariate strata. The counterfactual distributions to compare must be uniquely determined, by either specifying two values of `abar` at a single `nodes` or two `nodes` at a single intervention level `abar` or the natural course scenario with `abar = 'natural'` or two covariate strata `cond`. If the natural course scenario is selected and two `nodes` are specified, the natural intervention is compared across the two nodes. If one `nodes` is specified, the natural and observed scenarios are compared at a single node.

By default, the difference between the expectations of the two counterfactual outcome distributions is calculated. The difference can be exchanged for a ratio, odds ratio or custom contrast in the `contrastType` argument, and expectations can be exchanged for custom measures in the `measure` argument. Conditional measures can be specified through the `cond` argument. Custom contrasts, including those comparing more than two counterfactuals, can be defined by passing a function to `contrastType`.

Confidence intervals are based on the nonparametric bootstrap with `B` samples.

Value

Returns a list of class `contrastResult`:

`counterfactuals`	The estimated `measures` of the counterfactual distributions.
`contrast`	The estimated contrast between the counterfactual measures.
`ciContrast`	The lower and upper bounds of the bootstrap confidence interval for the contrast.
`B`	The number of successful bootstrap samples. Will usually be equal to the input `B`.
`varContrast`	The estimated bootstrap variance of the contrast.

See Also

`gformula` and `sgf` for estimating expected counterfactual outcomes under multiple intervention values and `custom.measure` for measures other than expectations.

Examples

```
data(EFV)
gf1 <- gformula(
  X = EFV, Anodes = c("efv.0", "efv.1", "efv.2", "efv.3", "efv.4"),
  Ynodes = c("VL.0", "VL.1", "VL.2", "VL.3", "VL.4"),
  Lnodes = c("adherence.1", "weight.1", "adherence.2", "weight.2",
             "adherence.3", "weight.3", "adherence.4", "weight.4"),
  abar = seq(1, 5), B = 10, ret = TRUE
)

# compare outcomes at last time point under (1,...,1) and (5,...,5)
contrast(gf1, abar = c(1, 5), nodes = "VL.4")

# compare outcomes at different time points, for same intervention (2,...)
contrast(gf1, abar = 2, nodes = c("VL.3", "VL.2"))

# compare own measure (rel. risk reduction) instead of mean
# ... and conditional on subset
relativeRiskReduction <- function(k, l){(k - l) / k}

contrast(
  gf1, abar = c(1, 2), nodes = "VL.4",
  contrastType = relativeRiskReduction,
  cond = quote(sex == 1)
)

# Instead of the mean, any other measure can be taken,
# and - of course - applied also to counterfactual Lnodes
contrast(
  gf1, abar = 2, nodes = c("weight.3", "weight.2"),
  measure = median
)
```

`custom.measure` *Custom estimands after applying* `gformula`

Description

The default estimate returned by `gformula` is the **expected** outcome under the respective intervention strategies `abar`. `custom.measure` takes an object of class `gformula` and enables estimation of other estimands based on the counterfactual datasets produced by `gformula` (if the option `ret=TRUE` had been chosen), for example estimands conditional on baseline variables, quantiles instead of expectations, and others.

Usage

```
custom.measure(X, fun = NULL, cond = NULL, verbose = TRUE, with.se = FALSE,
                cilevel = 0.95, ...)
```

Arguments

X	An object of class gformula produced by gformula with option ret=TRUE.
fun	A function to be applied to the outcome(s) of the counterfactual data set.
cond	A string containing a condition to be applied to the counterfactual datasets.
verbose	Logical. TRUE if notes should be printed.
with.se	Logical. TRUE if standard deviation should be calculated and returned.
cilevel	Numeric value between 0 and 1 specifying the confidence level. Defaults to 95%.
...	other parameters to be passed to fun

Details

In settings with censoring, it will often be needed to pass on the option na.rm=T, e.g. for the mean, median, quantiles, and others.

Calculation of the bootstrap standard error (i.e., with.se=T) is typically not needed; but, for example, necessary for the calculations after multiple imputation and hence used by mi.boot.

Value

An object of class gformula. See gformula for details.

See Also

see also gformula

Examples

```
data(EFV)

est <- gformula(X=EFV,
                Lnodes  = c("adherence.1","weight.1",
                            "adherence.2","weight.2",
                            "adherence.3","weight.3",
                            "adherence.4","weight.4"
                ),
                Ynodes  = c("VL.0","VL.1","VL.2","VL.3","VL.4"),
                Anodes  = c("efv.0","efv.1","efv.2","efv.3","efv.4"),
                abar=seq(0,2,1), ret=TRUE
)

est
custom.measure(est, fun=prop,categ=1) # identical
custom.measure(est, fun=prop,categ=0)
custom.measure(est, fun=prop, categ=0, cond="sex==1")
# note: metabolic has been recoded internally (see output above)
custom.measure(est, fun=prop, categ=0, cond="metabolic==0")
# does not make sense here, just for illustration (useful for metric outcomes)
custom.measure(est, fun=quantile, probs=0.1)
```

`EFV` *Pharmacoepidemiological HIV treatment data*

Description

A hypothetical, simulated dataset which is line with the data-generating process of Schomaker et al. (2024) and inspired by the data of Bienczak et al. (2017); see references below.

Usage

```
data(EFV)
```

Format

A data frame with 5000 observations on the following variables:

`sex` The patient's sex

`metabolic` Metabolism status (slow, intermediate, extensive) related to the single nucleotide polymorphisms in the CYP2B6 gene, which is relevant for metabolizing evafirenz and directly affects its concentration in the body.

`log_age` log(age) at baseline

`NRTI` Nucleoside reverse transcriptase inhibitor (NRTI) component of HIV treatment, i.e. abacavir, stavudine or zidovudine.

`weight.0` log(weight) at time 0 (baseline)

`efv.0` Efavirenz concentration at time 0 (baseline)

`VL.0` Elevated viral load (viral failure) at time 0 (baseline)

`adherence.1` Adherence at time 1 (if 0, then signs of non-adherence)

`weight.1` log(weight) at time 1

`efv.1` Efavirenz concentration at time 1

`VL.1` Elevated viral load (viral failure) at time 1

`adherence.2` Adherence at time 2 (if 0, then signs of non-adherence)

`weight.2` log(weight) at time 2

`efv.2` Efavirenz concentration at time 2

`VL.2` Elevated viral load (viral failure) at time 2

`adherence.3` Adherence at time 3 (if 0, then signs of non-adherence)

`weight.3` log(weight) at time 3

`efv.3` Efavirenz concentration at time 3

`VL.3` Elevated viral load (viral failure) at time 3

`adherence.4` Adherence at time 4 (if 0, then signs of non-adherence)

`weight.4` log(weight) at time 4

`efv.4` Efavirenz concentration at time 4

`VL.4` Elevated viral load (viral failure) at time 4

References

Schomaker M, McIlleron H, Denti P, Diaz I. (2024) *Causal Inference for Continuous Multiple Time Point Interventions*, Statistics in Medicine, 43:5380-5400, see also *https://arxiv.org/abs/2305.06645*.

Bienczak et al. (2017) *Determinants of virological outcome and adverse events in African children treated with paediatric nevirapine fixed-dose-combination tablets*, AIDS, *31:905-915*

Examples

```
data(EFV)
str(EFV)
```

`EFVfull` *Pharmacoepidemiological HIV treatment data*

Description

A hypothetical, simulated dataset which is line with the data-generating process of Schomaker et al. (2024) and inspired by the data of Bienczak et al. (2017); see references below. Compared to the dataset `EFV`, it contains all variables of the DAG in Figure 3 of Schomaker et al. (2023), also those which are not needed for identification of the counterfactual quantity of interest; that is, the expected viral suppression (`VL`) under a specific intervention on efavirenz concentrations (`efv.0, efv.1, ...`).

Usage

```
data(EFVfull)
```

Format

A data frame with 5000 observations on the following variables:

`sex` The patient's sex

`metabolic` Metabolism status (slow, intermediate, extensive) related to the single nucleotide polymorphisms in the CYP2B6 gene, which is relevant for metabolizing evafirenz and directly affects its concentration in the body.

`log_age` log(age) at baseline

`NRTI` Nucleoside reverse transcriptase inhibitor (NRTI) component of HIV treatment, i.e. abacavir, stavudine or zidovudine.

`weight.0` log(weight) at time 0 (baseline)

`comorbidity.0` Presence of co-morbidities at time 0 (baseline)

`dose.0` Dose of efavirenz administered at time 0 (basline)

`efv.0` Efavirenz concentration at time 0 (baseline)

`VL.0` Elevated viral load (viral failure) at time 0 (baseline)

`adherence.1` Adherence at time 1 (if 0, then signs of non-adherence) `weight.1` log(weight) at time 1

`comorbidity.1` Presence of co-morbidities at time 1

`dose.1` Dose of efavirenz administered at time 1

`efv.1` Efavirenz concentration at time 1

`VL.1` Elevated viral load (viral failure) at time 1

`adherence.2` Adherence at time 2 (if 0, then signs of non-adherence)

`weight.2` log(weight) at time 2

`comorbidity.2` Presence of co-morbidities at time 2

`dose.2` Dose of efavirenz administered at time 2

`efv.2` Efavirenz concentration at time 2

`VL.2` Elevated viral load (viral failure) at time 2

`adherence.3` Adherence at time 3 (if 0, then signs of non-adherence)

`weight.3` log(weight) at time 3

`comorbidity.3` Presence of co-morbidities at time 3

`dose.3` Dose of efavirenz administered at time 3

`efv.3` Efavirenz concentration at time 3

`VL.3` Elevated viral load (viral failure) at time 3

`adherence.4` Adherence at time 4 (if 0, then signs of non-adherence)

`weight.4` log(weight) at time 4

`comorbidity.4` Presence of co-morbidities at time 4

`dose.4` Dose of efavirenz administered at time 4

`efv.4` Efavirenz concentration at time 4

`VL.4` Elevated viral load (viral failure) at time

References

Schomaker M, McIlleron H, Denti P, Diaz I. (2024) *Causal Inference for Continuous Multiple Time Point Interventions*, Statistics in Medicine, 43:5380-5400, see also *https://arxiv.org/abs/2305.06645*.

Bienczak et al. (2017) *Determinants of virological outcome and adverse events in African children treated with paediatric nevirapine fixed-dose-combination tablets*, AIDS, *31:905-915*

Examples

```
data(EFVfull)
str(EFVfull)
```

`feasible` *Estimate Feasible Intervention Strategies*

Description

Estimate a family of feasible intervention strategies for a continuous treatment (and optionally time-varying covariates). The method returns, for each intervention strategy, the corresponding "feasible" intervention values and a summary of overlap and positivity-violation diagnostics.

Usage

```
feasible(
  X, Anodes = NULL, Ynodes = NULL, Lnodes = NULL, Cnodes = NULL,
  abar = NULL,
  alpha = 0.95, grid.size = 0.5, tol = 1e-2,
  left.boundary = NULL, right.boundary = NULL,
  screen = FALSE, survival = FALSE,
  d.method = c("hazardbinning", "binning", "parametric", "hal_density"),
  verbose = TRUE, ...
)
```

Arguments

`X`	A data frame containing all nodes in temporal order. Columns must include the treatment, outcome, covariate and censoring nodes specified in `Anodes`, `Ynodes`, `Lnodes`, and `Cnodes`.
`Anodes`	Character vector giving the column names in `X` of the (possibly time-varying) treatment nodes. These define the treatment history.
`Ynodes`	Character vector giving the column names in `X` of the outcome nodes. At least one outcome node must be specified, and all of them must occur after the first treatment node in the column ordering of `X`.
`Lnodes`	Optional character vector of confounder nodes. May be `NULL` if there are no such nodes.
`Cnodes`	Optional character vector of censoring (or competing event) nodes. May be `NULL` if there is no censoring.
`abar`	Numeric vector or matrix specifying the target interventions. • If a *vector*, each element defines a static intervention that sets the treatment to that value at *all* time points. In this case, each strategy corresponds to a single scalar target value. • If a *matrix*, rows index intervention strategies and columns index time points (one column per element in `Anodes`). Then `abar[k, t]` is the target treatment value for strategy `k` at time `t`. The argument must be numeric; `NULL` is not allowed.

`alpha`	Numeric scalar in `(0, 1)` controlling the density-truncation level. For each observation and time point, the method finds the smallest density threshold f_α such that at most `alpha` of the total mass lies above this threshold. Cells with density below f_α are treated as “infeasible”. Default is `0.95`.
`grid.size`	Positive numeric scalar giving the spacing of the grid used to approximate the treatment density. If `NULL`, no internal grid is constructed and `abar` itself is used as the grid of evaluation points (this is only allowed when `abar` is a vector). Default is `0.5`.
`tol`	Non-negative numeric tolerance used when combining `abar` with the grid. Points closer than `tol` to an element of `abar` are considered duplicates and are dropped from the internal grid before merging with `abar`. Default is `1e-2`.
`left.boundary`	Optional numeric scalar setting the left boundary of the grid used to approximate the treatment density. If `NULL`, the minimum of the observed treatment values and `abar` is used.
`right.boundary`	Optional numeric scalar setting the right boundary of the density grid. If `NULL`, the maximum of the observed treatment values and `abar` is used.
`screen`	Logical; if `TRUE`, use variable-screening (via internal functions from **CICI**) for the treatment models, otherwise use the full treatment model formulae. Default is `FALSE`.
`survival`	Logical; indicates whether the outcome nodes correspond to a survival-type structure. Passed to the internal model-building function. Default is `FALSE`.
`d.method`	Character string specifying the density-estimation method used to estimate the conditional treatment density. Must be one of `"hazardbinning"`, `"binning"`, `"parametric"`, or `"hal_density"`.
`verbose`	Logical; if `TRUE`, print warnings about ignored arguments passed via `...` and other diagnostic messages. Default is `TRUE`.
`...`	Additional arguments passed to the underlying density-estimation function determined by `d.method`. For example, these may include `SL.library` for Super Learner-based methods, or tuning parameters for specific density estimators. Arguments not recognised by the chosen `d.method` are silently ignored when `verbose = FALSE` and produce a warning when `verbose = TRUE`.

Details

The main steps of the algorithm are:

1. **Model specification:** Treatment models for each time point are constructed via helper routines from **CICI**. If `screen = TRUE`, a screening step updates the treatment formulas before density estimation (only recommended to address computational constraints).
2. **Grid construction:** A grid of treatment values, `query_abar`, is formed by combining:
    - observed treatment values in `X[, Anodes]`,
    - the target values in `abar`, and
    - a regular grid from `left.boundary` to `right.boundary` with spacing `grid.size` (when `grid.size` is not `NULL`).

If `grid.size = NULL`, the grid is restricted to the unique values in `abar` (only allowed when `abar` is a vector).

3. **Density estimation:** For each time point, the conditional treatment density is evaluated on the grid for each observation using the specified `d.method`. The resulting matrices are normalised so that each row integrates to one over the grid (accounting for bin width).
4. **Feasibility threshold:** For each observation and time point, a density threshold f_α is computed such that the cumulative mass below the sorted densities first exceeds `1 - alpha`. Cells with density below f_α are flagged as "infeasible".
5. **Feasible mapping:** For each grid cell with density below f_α, the algorithm finds the closest grid cell with density at or above f_α (in terms of grid index) and maps its value to that cell. This defines a "feasible" intervention that avoids low-density regions.
6. **Summary:** For each time point t and each intervention strategy (row of `abar`), the method collects:
   - the mean feasible value across individuals (column `Feasible`),
   - the proportion of cells below the density threshold (column `Low`, interpreted as `%infeasible` in the associated S3 methods),
   - the corresponding target value `Abar` at time t, and
   - the strategy index `Strategy`.

   These are combined into a data frame stored as the `"summary"` attribute of the returned object.

Plotting and printing methods are available for visual and tabular diagnostics; see `plot.feasible`, `print.feasible`, and `summary.feasible`.

Value

An object of class `"feasible"` with the following components:

- `feasible`: a list of length equal to the number of strategies (rows of `abar`). Each element is a matrix with one column per time point, containing the feasible intervention values for that strategy and time point across observations.
- `low_matrix`: a list of length equal to the number of time points. Each element is a logical matrix indicating, on the internal grid, which cells were marked as below the density threshold.

The object additionally has a `"summary"` attribute, a data frame with at least the columns `time`, `Strategy`, `Abar`, `Feasible`, and `Low`, which is accessed and formatted by `summary.feasible` and `print.feasible`.

See Also

`plot.feasible`, `print.feasible`, `summary.feasible`

Examples

```
data(EFV)


Lnodes <- c("adherence.1","weight.1",
            "adherence.2","weight.2",
            "adherence.3","weight.3",
            "adherence.4","weight.4")
Ynodes <- c("VL.0","VL.1","VL.2","VL.3","VL.4")
Anodes <- c("efv.0","efv.1","efv.2","efv.3","efv.4")

## ------------------------------------------------------------------
## Example 1: Hazard binning with default grid
## Static grid of targets (vector abar) over the full support of efv.*
## ------------------------------------------------------------------

abar_static <- seq(0, 10, by = 1)

m_hazard <- feasible(
  X      = EFV,
  Anodes = Anodes,
  Lnodes = Lnodes,
  Ynodes = Ynodes,
  d.method      = "hazardbinning", # long computation, but appropriate
  abar          = abar_static,
  grid.size     = 0.5,
  left.boundary = 0,
  right.boundary = 10
)


## Individual-level feasible values (one matrix per strategy):
## rows = individuals, columns = time points
feasible_matrix <- m_hazard$feasible # pass on to gofrmula/sgf
lapply(feasible_matrix, head)

## Inspect feasability of strategies
m_hazard             # see also ?print.feasible
summary(m_hazard)    # see also ?summary.feasible


## ------------------------------------------------------------------
## Example 2: Parametric density, using abar as the grid
## Here grid.size = NULL, so only the target values are used as grid
## ------------------------------------------------------------------

abar_param <- seq(0, 10, by = 2)

m_param <- feasible(
  X      = EFV,
  Anodes = Anodes,
  Lnodes = Lnodes,
  Ynodes = Ynodes,
  # fast, but useful for reasonably symmetric distributions
  d.method      = "parametric",
  abar          = abar_param,
  grid.size     = NULL,
  left.boundary = 0,
  right.boundary = 10
```

```
)

## Inspect feasability of strategies
m_param             # see also ?print.feasible
summary(m_param)    # see also ?summary.feasible


## ------------------------------------------------------------------
## Example 3: Matrix abar with non-constant strategies over time
## Each row is a strategy, each column corresponds to efv.0, ..., efv.4
## ------------------------------------------------------------------

abar_matrix <- rbind(
  c(0, 2, 4, 6, 8),  # strategy 1
  c(9, 6, 2, 1, 0),  # strategy 2
  c(1, 3, 5, 7, 9)   # strategy 3
)

m_matrix <- feasible(
  X      = EFV,
  Anodes = Anodes,
  Lnodes = Lnodes,
  Ynodes = Ynodes,
  d.method      = "parametric",
  abar          = abar_matrix,
  grid.size     = 1,
  left.boundary = 0,
  right.boundary = 10
)

## Inspect feasability of strategies
m_matrix             # see also ?print.feasible
summary(m_matrix)    # see also ?summary.feasible
```

fit.updated.formulas *Fit models after screening*

Description

Fits the models that have been generated with screening using `model.formulas.update`.

Usage

```
fit.updated.formulas(formulas, X)
```

Arguments

`formulas`	An object returned by `model.formulas.update`
`X`	A data frame on which the model formulas should be evaluated

Details

Fits generalized (additive) linear models based on the screened model formula list generated by `model.formulas.update`.

Value

Returns a list of length 2:

`fitted.models`	A list of length 4, containing the fitted Y-/L-/C- and A-models.
`all.summaries`	A list of length 4, containing the summary of the fitted Y-/L-/C- and A-models.

See Also

`model.formulas.update`

Examples

```
data(EFV)

# first: generate generic model formulas
m <- make.model.formulas(X=EFV,
                         Lnodes  = c("adherence.1","weight.1",
                                     "adherence.2","weight.2",
                                     "adherence.3","weight.3",
                                     "adherence.4","weight.4"
                                    ),
                         Ynodes  = c("VL.0","VL.1","VL.2","VL.3","VL.4"),
                         Anodes  = c("efv.0","efv.1","efv.2","efv.3","efv.4"),
                         evaluate=FALSE)

# second: update these model formulas based on variable screening with LASSO
glmnet.formulas <-  model.formulas.update(m$model.names, EFV)
glmnet.formulas

# then: fit and inspect the updated models
fitted.models <- fit.updated.formulas(glmnet.formulas, EFV)
fitted.models$all.summaries
fitted.models$all.summaries$Ynames[1] # first outcome model
```

`gformula` *Parametric g-formula for continuous multiple time point interventions*

Description

Estimation of counterfactual outcomes for multiple values of continuous interventions at different time points using the g-formula.

Usage

```
gformula(X, Anodes, Ynodes, Lnodes = NULL, Cnodes = NULL,
         abar = NULL, cbar = "uncensored",
         survivalY = FALSE,
         Yform = "GLM", Lform = "GLM", Aform = "GLM", Cform = "GLM",
         calc.support = FALSE, B = 0, ret = FALSE, ncores = 1,
         verbose = TRUE, seed = NULL, prog = NULL, cilevel = 0.95, ...)
```

Arguments

`X`	A data frame, following the time-ordering of the nodes. Categorical variables with k categories should be a factor, with levels 0,...,k-1. Binary variables should be coded 0/1.
`Anodes`	A character string of column names in `X` of the intervention variable(s).
`Ynodes`	A character string of column names in `X` of the outcome variable(s).
`Lnodes`	A character string of column names in `X` of the time-dependent (post first treatment) variable(s).
`Cnodes`	A character string of column names in `X` of the censoring variable(s).
`abar`	Numeric vector or matrix of intervention values, or the string "natural". See Details.
`cbar`	Typically either the string "uncensored" or "natural", but a numeric vector or matrix of censoring values is not forbidden. See Details.
`survivalY`	Logical. If TRUE, then Y nodes are indicators of an event, and if Y at some time point is 1, then all following should be 1.
`Yform`	A string of either "GLM", "GAM" or of length 'number of Ynodes' with model formulas. See Details.
`Lform`	A string of either "GLM", "GAM" or of length 'number of Lnodes' with model formulas. See Details.
`Aform`	A string of either "GLM", "GAM" or of length 'number of Anodes' with model formulas. See Details.
`Cform`	A string of either "GLM", "GAM" or of length 'number of Cnodes' with model formulas. See Details.
`calc.support`	Logical. If `TRUE`, both crude and conditional support is estimated.
`B`	An integer specifying the number of bootstrap samples to be used, if any.
`ret`	Logical. If `TRUE`, the simulated post-intervention data is returned.
`ncores`	An integer for the number of threads/cores to be used. If >1, parallelization will be utilized.
`verbose`	Logical. If `TRUE`, notes and warnings are printed.
`seed`	An integer specifying the seed to be used to create reproducable results for parallel computing (i.e. when ncores>1).
`prog`	A character specifying a path where progress should be saved (typically, when `ncores>1`)
`cilevel`	Numeric value between 0 and 1 specifying the confidence level. Defaults to 95%.
`...`	Further arguments to be passed on.

Details

By default, expected counterfactual outcomes (specified under `Ynodes`) under the intervention `abar` are calculated. Other estimands can be specified via `custom.measure`.

If `abar` is a vector, then each vector component is used as the intervention value at each time point; that is, interventions which are constant over time are defined. If `abar` is a matrix (of size 'number interventions' x 'time points'), then each row of the length of `Anodes` refers to a particular time-varying intervention strategy. The natural intervention can be picked by setting `abar='natural'`.

The fitted outcome and confounder models are based on generalized additive models (GAMs) as implemented in the `mgcv` package. Model families are picked automatically and reported in the output if `verbose=TRUE` (see manual for modifications, though they hardly ever make sense). The model formulas are standard GLMs or GAMs (with penalized splines for continuous covariates), conditional on the past, unless specific formulae are given. It is recommended to use customized formulae to reduce the risk of model mis-specification and to ensure that the models make sense (e.g., not too many splines are used when this is computationally not meaningful). This can be best facilitated by using objects generated through `make.model.formulas`, followed by `model.formulas.update` and/or `model.update` (see examples for those functions).

For survival settings, it is required that i) `survivalY=TRUE` and ii) after a Cnode/Ynode is 1, every variable thereafter is set to `NA`. See manual for an example. By default, the package intervenes on Cnodes, i.e. calculates counterfactual outcomes under no censoring.

If `calc.support=TRUE`, conditional and crude support measures (i.e., diagnostics) are calculated as described in Section 3.4 of Schomaker et al. (2023). Another useful diagnostic for multiple time points is the natural course scenario, which can be evaluated under `abar='natural'` and `cbar='natural'`.

To parallelize computations automatically, it is sufficient to set `ncores>1`, as appropriate. To make estimates under parallelization reproducible, use the `seed` argument. To watch the progress of parallelized computations, set a path in the `prog` argument: then, a text file reports on the progress, which is particularly useful if lengthy bootstrapping computations are required.

Value

Returns an object of of `class` 'gformula':

`results`	data.frame of results. That is, the estimated counterfactual outcomes depending on the chosen intervention strategies, and time points.
`diagnostics`	list of diagnostics and weights based on the estimated support (if `calc.support=TRUE`)
`simulated.data`	list of counterfactual data sets related to the interventions defined through option `abar` (and `cbar`). Will be `NULL` is `ret=FALSE`.
`observed.data`	list of observed data (and bootstrapped observed data). Will be `NULL` is `ret=FALSE`.
`setup`	list of chosen setup parameters

Author(s)

Michael Schomaker

See Also

plot.gformula for plotting results as (causal) dose response curves, custom.measure for evaluating custom estimands and mi.boot for using gformula on multiply imputed data.

Examples

```
## Not run:
data(EFV)
est <- gformula(X=EFV,
                      Lnodes  = c("adherence.1","weight.1",
                                  "adherence.2","weight.2",
                                  "adherence.3","weight.3",
                                  "adherence.4","weight.4"
                        ),
                        Ynodes  = c("VL.0","VL.1","VL.2","VL.3","VL.4"),
                        Anodes  = c("efv.0","efv.1","efv.2","efv.3","efv.4"),
                        abar=seq(0,10,1)
)
est

## End(Not run)
```

---

make.model.formulas   *Compose appropriate model formulas*

Description

Function that generates generic model formulas for Y-/L-/A- and Cnodes, according to time ordering and to be used in gformula or model.formulas.update.

Usage

```
make.model.formulas(X, Ynodes = NULL, Lnodes = NULL, Cnodes = NULL, Anodes = NULL,
                      survival = FALSE, evaluate = FALSE)
```

Arguments

X	A data frame, following the time-ordering of the nodes.
Ynodes	A character string of column names in X of the outcome variable(s).
Lnodes	A character string of column names in X of time-dependent (post first treatment) variable(s).
Cnodes	A character string of column names in X of the censoring variable(s).
Anodes	A character string of column names in X of intervention variable(s).
survival	Logical. If TRUE, a survival setting is assumed and taken into account for model specification.
evaluate	Logical. TRUE if model formulas should model formulas be evaluated on X.

Details

This is a helper function to generate model formulas for Y-/L-/A- and Cnodes, according to the time ordering: i.e. to generate GLM/GAM model formulas for the respective nodes given all *past* variables. In survival settings, past censoring and outcome nodes are omitted from the formulae. If censoring is present without a survival setting (e.g. Cnodes describe drop-outs and Y is a continuous outcome), then survival should be set as FALSE.

Value

Returns a named list:

`model.names`	A list of length 4 containing strings of the actual formulas
`fitted.models`	A list of the fitted models (if `evaluate=TRUE`)
`fitted.model.summary`	A list of the summary of the fitted models (if `evaluate=TRUE`)

See Also

The generated generic model formulas can be updated manually with `model.update` or in an automated manner with screening using `model.formulas.update`.

Examples

```
data(EFV)

m <- make.model.formulas(X=EFV,
                         Lnodes  = c("adherence.1","weight.1",
                                     "adherence.2","weight.2",
                                     "adherence.3","weight.3",
                                     "adherence.4","weight.4"
                                    ),
                         Ynodes  = c("VL.0","VL.1","VL.2","VL.3","VL.4"),
                         Anodes  = c("efv.0","efv.1","efv.2","efv.3","efv.4"),
                         evaluate=FALSE) # set TRUE to see fitted models

m$model.names # all models potentially relevant for gformula(), given full past
```

`mi.boot` *Obtaining estimates from multiply imputed data*

Description

Combines `gformula` estimates obtained from multiple imputed data sets according to the *MI Boot* and *MI Boot pooled* methods decribed in Schomaker and Heumann (2018, see reference section below)

Usage

```
mi.boot(x, fun, cond = NULL, pooled = FALSE, cilevel = 0.95, ...)
```

Arguments

`x`	A list of objects of `class` 'gformula'
`fun`	A function to be applied to the outcome(s) of the counterfactual data set. For expected outcome, use `mean` and possibly pass on option `na.rm=TRUE`.
`cond`	A string containing a condition to be applied to the counterfactual datasets.
`pooled`	Logical. If TRUE, confidence interval estimation is based on the MI Boot pooled from Schomaker and Heumann (2018), otherwise on MI Boot.
`cilevel`	Numeric value between 0 and 1 specifying the confidence level. Defaults to 95%.
`...`	additional arguments to be passed on to `fun`

Value

An object of class `gformula`. See `gformula` for details.

Author(s)

Michael Schomaker

References

Schomaker, M., Heumann, C. (2018) *Bootstrap inference when using multiple imputation*, Statistics in Medicine, 37:2252-2266

Examples

```
data(EFV)

# suppose the following subsets were actually multiply imputed data (M=2)
EFV_1 <- EFV[1:2500,]
EFV_2 <- EFV[2501:5000,]

# first: conduct analysis on each imputed data set. Set ret=T.
m1 <- gformula(X=EFV_1,
               Lnodes  = c("adherence.1","weight.1",
                           "adherence.2","weight.2",
                           "adherence.3","weight.3",
                           "adherence.4","weight.4"
               ),
               Ynodes  = c("VL.0","VL.1","VL.2","VL.3","VL.4"),
               Anodes  = c("efv.0","efv.1","efv.2","efv.3","efv.4"),
               abar=seq(0,5,1), verbose=FALSE, ret=TRUE
        )

m2 <- gformula(X=EFV_2,
               Lnodes  = c("adherence.1","weight.1",
                           "adherence.2","weight.2",
                           "adherence.3","weight.3",
                           "adherence.4","weight.4"
               ),
               Ynodes  = c("VL.0","VL.1","VL.2","VL.3","VL.4"),
               Anodes  = c("efv.0","efv.1","efv.2","efv.3","efv.4"),
               abar=seq(0,5,1), verbose=FALSE, ret=TRUE
)

# second combine results
m_imp <- mi.boot(list(m1,m2), mean) # uses MI rules & returns 'gformula' object
plot(m_imp)

# custom estimand: evaluate probability of suppression (Y=0), among females
m_imp2 <- mi.boot(list(m1,m2), prop, categ=0, cond="sex==1")
plot(m_imp2)
```

model.formulas.update *Update model formulas based on variable screening*

Description

Wrapper function to facilitate variable screening on all models generated through `make.model.formulas` and return updated formulas in the appropriate format for `gformula`.

Usage

```
model.formulas.update(formulas, X, screening = screen.glmnet.cramer,
                      with.s = FALSE, by= NA, ...)
```

Arguments

`formulas`	A named list of length 4 containing model formulas for all Y-/L-/A- and Cnodes. These are likely formulas returned from `make.model.formulas`.
`X`	A data frame on which the model formulas are to be evaluated.
`screening`	A screening function. Default is `screen.glmnet.cramer`, see Details below.
`with.s`	Logical. If TRUE, a spline, i.e. s(), will be added to *all* continuous variables.
`by`	A character vector specifying the variables with which to multiply the smooth (if `with.s=TRUE`).
`...`	optional arguments to be passed to the screening algorithm

Details

The default screening algorithm uses LASSO for variable screening (and Cramer's V for the categorized version of all variables if LASSO fails). It is possible to provide user-specific screening algorithms. User-specific algorithms should take the data as first argument, *one* model formula (i.e. one entry of the list in `model.formulas`) as second argument and return a vector of strings, containing the variable names that remain after screening. Another screening algorithm available in the package is `screen.cramersv`, which categorizes all variables, calculates their association with the outcome based on Cramer's V and selects the 4 variables with strongest associations (can be changed with option `nscreen`). The manual provides more information.

The fitted models of the updated models can be evaluated with `fit.updated.formulas`.

Value

A list of length 4 containing the updated model formulas:

`Lnames`	A vector of strings containing updated model formulas for all L nodes.
`Ynames`	A vector of strings containing updated model formulas for all Y nodes.
`Anames`	A vector of strings containing updated model formulas for all A nodes.
`Cnames`	A vector of strings containing updated model formulas for all C nodes.

See Also

`make.model.formulas`, `model.update`, `fit.updated.formulas`

Examples

```
data(EFV)

# first: generate generic model formulas
m <- make.model.formulas(X=EFV,
                         Lnodes  = c("adherence.1","weight.1",
                                     "adherence.2","weight.2",
                                     "adherence.3","weight.3",
                                     "adherence.4","weight.4"
                                    ),
                         Ynodes  = c("VL.0","VL.1","VL.2","VL.3","VL.4"),
                         Anodes  = c("efv.0","efv.1","efv.2","efv.3","efv.4"),
                         evaluate=FALSE)

# second: update these model formulas based on variable screening with LASSO
glmnet.formulas <-  model.formulas.update(m$model.names, EFV)
glmnet.formulas


# third: use these models for estimation
est <- gformula(X=EFV,
                Lnodes  = c("adherence.1","weight.1",
                            "adherence.2","weight.2",
                            "adherence.3","weight.3",
                            "adherence.4","weight.4"
                ),
                Ynodes  = c("VL.0","VL.1","VL.2","VL.3","VL.4"),
                Anodes  = c("efv.0","efv.1","efv.2","efv.3","efv.4"),
                Yform=glmnet.formulas$Ynames, Lform=glmnet.formulas$Lnames,
                abar=seq(0,2,1)
)
est
```

`model.update` *Update GAM models*

Description

A wrapper to simplify the update of GAM models

Usage

```
model.update(gam.object, form)
```

Arguments

`gam.object`	A `gam` object produced with package **mgcv**.
`form`	A new model formula in the form `.~formula`

Details

The `gam` object needs to be fitted with the option control=list(keepData=T), otherwise the function can not access the data that is needed to update the model fit. Note that both `fit.updated.formulas` and `make.model.formulas` with option `evaluate=T` produce results that are based on this option.

Value

An object of `class` 'gam', 'glm' and 'lm'.

Examples

```
data(EFV)

m <- make.model.formulas(X=EFV,
                         Lnodes  = c("adherence.1","weight.1",
                                     "adherence.2","weight.2",
                                     "adherence.3","weight.3",
                                     "adherence.4","weight.4"
                         ),
                         Ynodes  = c("VL.0","VL.1","VL.2","VL.3","VL.4"),
                         Anodes  = c("efv.0","efv.1","efv.2","efv.3","efv.4"),
                         evaluate=TRUE) # set TRUE for model.update()

# update first confounder model of weight manually
model.update(m$fitted.models$fitted.L$m_weight.1, .~s(weight.0, by=sex))

# manual update of model formula
m$model.names$Lnames[2] <- "weight.1 ~ s(weight.0, by=sex)"
```

msm — *Marginal structural model from the parametric g-formula for continuous multiple time point interventions*

Description

Estimation of a marginal structural model using the parametric g-formula.

Usage

```
msm(X, formula, family = gaussian, se = NULL, cilevel = 0.95, abar=NULL)
```

Arguments

`X`	An object of class `gformula` produced by `gformula`, with option `ret = TRUE`.
`formula`	Form of the marginal structural model. Can be specified as a formula object, e.g., `formula = VL.4 ~ efv.4`, as a quoted expression, e.g., `formula = quote(VL.4 ~ efv.4)`, or as a character string, e.g., `formula = "VL.4 ~ efv.4"`.
`family`	A description of the error distribution and link function to be used in the model. See `family` for details of family functions.
`se`	A character string specifying the standard errors used to compute confidence intervals. One of `c('bootstrap', 'glm')`. See Details.
`cilevel`	Numeric value between 0 and 1 specifying the confidence level of the bootstrap confidence intervals. Defaults to 95%.
`abar`	Vector or matrix that is a subset of the intervention used in X. Can be used to fit an MSM on a subset of the stacked counterfactual data.

Details

The marginal structural model (MSM) is estimated as a GLM. Confidence intervals are calculated using GLM standard errors (if `se = 'glm'`) or nonparametric boostrap standard errors (if `se = 'bootstrap'` and `gformula` was run with `B > 0`.) By default: `se = 'bootstrap'` if `gformula` was run with `B > 0`, and `se = 'glm'` otherwise.

Value

Returns a list of class `msmResult`:

`MSM`	The fitted MSM of class `glm`.
`coefs`	The estimated coefficients of the MSM.
`CIlow`	Lower confidence interval bounds for each coefficient.
`CIup`	Upper confidence interval bounds for each coefficient.
`formula`	The ’formula’ input argument.
`se`	The ’se’ input argument.
`vcov`	Covariance matrix.

See Also

gformula for estimating expected counterfactual outcomes under multiple intervention values.

Examples

```
data(EFV)
gf <- gformula(
  X = EFV, Anodes = c("efv.0", "efv.1", "efv.2", "efv.3", "efv.4"),
  Ynodes = c("VL.0", "VL.1", "VL.2", "VL.3", "VL.4"),
  Lnodes = c("adherence.1", "weight.1", "adherence.2", "weight.2",
             "adherence.3", "weight.3", "adherence.4", "weight.4"),
  abar = seq(0, 5), B = 10, ret = TRUE
)

msm(gf, VL.4 ~ efv.4, se = "bootstrap") # default if B>0
msm(gf, VL.4 ~ efv.4, se = "glm")       # fast, but not valid (undercoverage)
```

`plot.feasible` *Plot Method for* `feasible` *objects*

Description

Generate diagnostic plots for objects of class "feasible", returned by feasible. One can display either (i) mean feasible vs. target interventions, or (ii) the non-overlap ratio.

Usage

```
## S3 method for class 'feasible'
plot(x, x.axis = c("strategy", "time"),
                        which  = c("feasible", "nonoverlap"),
                        facet  = c("none", "time", "strategy"), ...)
```

Arguments

`x`	An object of class "feasible" with a "summary" attribute (typically returned by feasible).
`x.axis`	A string specifying the x-axis: • If "strategy", and each strategy corresponds to the same target value at every time-point (i.e., this relationship is consistent across time), the method uses abar for the x-axis, otherwise the strategy index is used. • If "time", the x-axis shows discrete time-points and colors represent targets or strategies, depending on the context.
`which`	Which plot to show: • "feasible": mean feasible intervention values compared to original target intervention.

- `"nonoverlap"`: non-overlap ratio (proportion of mass below the density threshold).

`facet` Optional faceting to reduce overplotting:

- `"none"` (default): no faceting, all series in a single panel.
- `"time"`: one panel per time-point.
- `"strategy"`: one panel per intervention strategy.

Facet strips are labelled with variable name and value (via `label_both`).

`...` Additional arguments (currently unused). Included for method consistency.

Details

Both plot types are drawn with **ggplot2**. To reduce overplotting, lines and points use transparency (`alpha`) and slightly smaller widths/sizes by default. Faceting by `time` or `strategy` can further improve readability when many series are present.

The `"summary"` attribute of a `"feasible"` object is expected to contain (at least) the following columns:

- `time`: discrete time index.
- `Strategy`: strategy index (row index of the intervention design).
- `Abar`: target value (intervention level) for that strategy at that time.
- `Feasible`: mean feasible value under the estimated feasible intervention.
- `Low`: non-overlap ratio (proportion of mass below the density threshold).

Interpretation of `abar`:

- In `feasible`, the `abar` argument may be either a numeric vector (static grid of targets) or a numeric matrix (dynamic interventions).
- If `abar` is a vector, each distinct value defines a strategy that is constant over time; in this case each strategy represents the same target value at every time-point.
- If `abar` is a matrix, rows index intervention strategies and columns index time-points. In the summary, `Strategy` identifies the row, and `Abar` is the entry of that row at the corresponding time-point.

Plot types:

1. **Feasible vs Target** (`which = "feasible"`):
   - *Y-axis:* mean feasible intervention (`Feasible`).
   - *X-axis:* controlled by `x.axis`:
     - `x.axis = "time"`: x-axis shows time; colors represent Targets (`Abar`) when each strategy has the same target at all time-points, or represent strategies when targets vary over time within a strategy.
     - `x.axis = "strategy"`: x-axis shows strategy index; if each strategy corresponds to a single target value at all time-points and this relationship is consistent, the x-axis is relabelled to show `Abar` (Targets) instead.
   - *Reference line and ticks:*

   - When the x-axis is on the Target scale (strategies are constant over time with respect to `Abar`), the plot includes a dashed 1:1 reference line Feasible = Target and aligns the x- and y-axis limits to the range of `Abar`, when plausible (i.e., when the feasible values lie within the range of `Abar`).
   - When `x.axis = "time"`, short horizontal ticks at each time-point indicate the `Abar` values for each strategy (or Target when strategies are constant over time), using the same color mapping as the series.
   - When `x.axis = "strategy"` and strategies do not correspond to a single target over all time-points, ticks are drawn at each strategy to indicate the `Abar` values across time.

2. **Non-overlap Ratio** (`which = "nonoverlap"`):
   - *Y-axis:* non-overlap ratio `Low` (bounded between 0 and 1), plotted with fixed limits `c(0, 1)`.
   - *X-axis:* same choice of `x.axis` as for the feasible plot.

Terminology: Throughout the plots, "Target" refers to the intervention values passed as the `abar` argument to `feasible` (stored as column `Abar` in the object's summary). When strategies are constant over time with respect to `Abar` and this structure is consistent across time, each Target corresponds to an identical intervention pattern at all time-points. This is reflected in both the x-axis labelling and the legend.

Value

Invisibly returns the **ggplot2** object that is drawn (either the feasible plot or the non-overlap plot).

Author(s)

Han Bao, Michael Schomaker

See Also

`feasible`, `summary.feasible`

Examples

```
data(EFV)


Lnodes <- c("adherence.1","weight.1",
            "adherence.2","weight.2",
            "adherence.3","weight.3",
            "adherence.4","weight.4")
Ynodes <- c("VL.0","VL.1","VL.2","VL.3","VL.4")
Anodes <- c("efv.0","efv.1","efv.2","efv.3","efv.4")

## ------------------------------------------------------------------
## Example 1: Static grid of Targets (vector abar)
## Each strategy uses the same target value at every time-point
## ------------------------------------------------------------------

abar_static <- seq(0, 10, by = 2)
```

```
m_static <- feasible(X = EFV,
                     Anodes = Anodes,
                     Lnodes = Lnodes,
                     Ynodes = Ynodes,
                     d.method = "parametric",
                     abar = abar_static,
                     grid.size = NULL,
                     left.boundary = 0,
                     right.boundary = 10)

## Feasible vs Target with time on x-axis (default).
## Colors indicate Targets (Abar), and short ticks show Abar at each time.
plot(m_static, which = "feasible")

## Feasible vs Target with time on x-axis.
plot(m_static, x.axis = "time", which = "feasible")

## Non-overlap ratio
plot(m_static, which = "nonoverlap")

## Facet by time to reduce overplotting
plot(m_static, which = "feasible", facet = "time")


## ------------------------------------------------------------------
## Example 2: Non-constant intervention strategies (matrix abar)
## Strategies can have different target values at different time-points
## ------------------------------------------------------------------

## Here rows define strategies and columns define time-points.
abar_matrix <- rbind(
  c(0, 2, 4, 6, 8),  # strategy 1
  c(9, 6, 2, 1, 0),  # strategy 2
  c(1, 3, 5, 7, 9)   # strategy 3
)

set.seed(456)
m_matrix <- feasible(X = EFV,
                     Anodes = Anodes,
                     Lnodes = Lnodes,
                     Ynodes = Ynodes,
                     d.method = "parametric",
                     abar = abar_matrix,
                     grid.size = 1,
                     left.boundary = 0,
                     right.boundary = 10)

## Feasible vs Target with time on the x-axis.
## Colors represent strategies; short ticks at each time show
## the corresponding Abar for each strategy.
plot(m_matrix,
     x.axis = "time",
          which  = "feasible",
          facet  = "none")

    ## Feasible vs Target with strategy on the x-axis.
    ## Strategies no longer use the same target at all time-points,
    ## so the x-axis stays on the strategy index, and ticks at each
    ## strategy indicate the Abar values across time.
    plot(m_matrix,
          x.axis = "strategy",
          which  = "feasible",
          facet  = "none")

    ## Non-overlap ratio for these non-constant strategies,
    ## shown over time and faceted by strategy for clarity.
    plot(m_matrix,
          x.axis = "time",
          which  = "nonoverlap",
          facet  = "strategy")
```

`plot.gformula` *Plot dose-response curves*

Description

Function to plot dose-response curves based on results returned from `gformula`

Usage

```
## S3 method for class 'gformula'
plot(x, msm.method = c("line","loess", "gam", "none"),
                           CI = FALSE, time.points = NULL,
                           cols = NULL, weight = NULL, xaxis=NULL,
                           variable = "psi", difference = FALSE, ...)
```

Arguments

`x`	An object of class 'gformula'.
`msm.method`	A string specifying the method to connect individual estimates into a curve (marginal structural model). One of `"line"`,`"none"`,`"gam"` and `"loess"`.
`CI`	Logical. If TRUE, confidence bands are drawn; or confidence intervals for specific points if both `msm.method="none"` and appropriate.
`time.points`	A vector of time points for which the respective curves should be drawn. Default is all time points.
`cols`	A vector of strings specifying custom colours for each drawn curve.
`weight`	Weight vector of size "number of interventions times time points", that is used for the MSM if `msm.method="loess"` or `msm.method="gam"`.

`xaxis`	Either NULL or a string. If set to "time", then the x-axis is forced to represent time (unless this is impossible)
`variable`	A string specifying the variable to be plotted under the natural course scenario (i.e., if `abar"natural"` and `cbar="natural"` in the respective `gformula` object).
`difference`	Logical. If TRUE, differences of observed outcomes and outcomes under the natural intervention will be plotted (if `abar"natural"` and `cbar="natural"` in the respective `gformula` object.).
`...`	Further arguments to be passed on

Details

Time points and variable names should be specified according to the labeling of the results table returned by `gformula`.

Value

Draws an object of `class` 'ggplot'.

Examples

```
data(EFV)
est <- gformula(X=EFV,
                Lnodes  = c("adherence.1","weight.1",
                            "adherence.2","weight.2",
                            "adherence.3","weight.3",
                            "adherence.4","weight.4"
                ),
                Ynodes  = c("VL.0","VL.1","VL.2","VL.3","VL.4"),
                Anodes  = c("efv.0","efv.1","efv.2","efv.3","efv.4"),
                abar=seq(0,10,1)
)


plot(est)
plot(est, time.points=c(1,5))
```

`print.feasible` *Print method for* `feasible` *objects*

Description

Produces a concisey summary of a feasible intervention object. The printout summarizes information jointly over time and strategy, using tables with strategies as rows and time points as columns. Separate tables are printed for the proportion of infeasible mass (`%infeasible`) and the mean feasible value (`Feasible`).

Usage

```
## S3 method for class 'feasible'
print(x, digits = 3, strategies = "all", times = "all", ...)
```

Arguments

`x`	A `"feasible"` object returned by `feasible`.
`digits`	Integer; number of digits used when printing numeric values.
`strategies`	Either `"all"` (default) or a numeric vector of strategy indices to include in the printed summary. When a numeric vector is supplied, all summaries and tables are restricted to these strategies.
`times`	Either `"all"` (default) or a numeric vector of time indices to include in the printed summary. When a numeric vector is supplied, all summaries and tables are restricted to these time points.
`...`	Ignored; provided for S3 method compatibility.

Details

The method extracts the `data.frame` stored in the `"summary"` attribute of `x` and optionally restricts it to the selected `strategies` and `times`. All reported values are based on this restricted data.

The summary data typically contains at least the following columns:

- **time**: time index `t`.
- **Strategy**: index of the intervention strategy.
- **Abar**: target intervention value at time `t`.
- **%infeasible**: proportion of mass (on the 0–1 scale) falling below the estimated density threshold for the targeted `Abar`.
- **Feasible**: mean of the mapped feasible values (after replacing low-density bins) for the targeted bin.

The output consists of:

- A short header showing how many strategies and time points exist in the underlying object, and how many are being displayed after subsetting via `strategies` and `times`.
- **Table 1**: `%infeasible` summarized by strategy (rows) and time (columns), printed as percentage.
- **Table 2**: `Feasible` (mean feasible value) summarized by strategy (rows) and time (columns), printed on the original scale.
- A compact display of the `Abar` targets by strategy and time.

Within the selected subset, the method also checks whether each strategy uses the same `Abar` at every selected time point. If that is the case, the printout notes that each selected strategy corresponds to the same intervention pattern over time. Otherwise, differences in `Abar` across time are made visible by the `Abar`-by-time display.

Value

Invisibly returns x.

See Also

feasible, summary.feasible, plot.feasible

Examples

```
data(EFV)


Lnodes <- c("adherence.1","weight.1",
            "adherence.2","weight.2",
            "adherence.3","weight.3",
            "adherence.4","weight.4")
Ynodes <- c("VL.0","VL.1","VL.2","VL.3","VL.4")
Anodes <- c("efv.0","efv.1","efv.2","efv.3","efv.4")

## ------------------------------------------------------------------
## Example 1: Static grid of targets (vector abar)
## Each strategy uses the same target value at every time point
## ------------------------------------------------------------------

abar_static <- seq(0, 10, by = 2)

set.seed(123)
m_static <- feasible(X = EFV,
                     Anodes = Anodes,
                     Lnodes = Lnodes,
                     Ynodes = Ynodes,
                     d.method = "parametric",
                     abar = abar_static,
                     grid.size = NULL,
                     left.boundary = 0,
                     right.boundary = 10)

## Full time x strategy summary
print(m_static)

## Use fewer digits in the numeric summaries
print(m_static, digits = 2)

## Focus on a subset of strategies (e.g., 1 and 3)
print(m_static, strategies = c(1, 3))

## Focus on early time points only (e.g., times 1, 2)
print(m_static, times = c(1, 2))

## Combine selection: only strategies 1 and 3 over times 1, 2, 3
print(m_static, strategies = c(1, 3), times = 1:3)
```

```
## ----------------------------------------------------------------
## Example 2: Non-constant intervention strategies (matrix abar)
## Strategies can have different target values at different time points
## ----------------------------------------------------------------

## Rows define strategies, columns define time points.
## The first row increases over time, the second decreases, the third increases.
abar_matrix <- rbind(
  c(0, 2, 4, 6, 8),  # strategy 1
  c(9, 6, 2, 1, 0),  # strategy 2
  c(1, 3, 5, 7, 9)   # strategy 3
)

set.seed(456)
m_matrix <- feasible(X = EFV,
                     Anodes = Anodes,
                     Lnodes = Lnodes,
                     Ynodes = Ynodes,
                     d.method = "parametric",
                     abar = abar_matrix,
                     grid.size = 1,
                     left.boundary = 0,
                     right.boundary = 10)

## Time x strategy summary where targets vary over time within strategies
print(m_matrix)

## Focus on strategies 1 and 3 over a subset of time points
print(m_matrix, strategies = c(1, 3), times = 1:3)
```

---

sgf *Sequential g-formula for continuous multiple time point interventions*

Description

Estimation of counterfactual outcomes for multiple values of continuous interventions at different time points using the sequential (weighted) g-formula.

Usage

```
sgf(X, Anodes, Ynodes, Lnodes = NULL, Cnodes = NULL,
    abar = NULL, survivalY = FALSE,
    SL.library = "SL.glm", SL.export = NULL,
    Yweights = NULL, calc.support = FALSE, B = 0,
    ncores = 1, verbose = TRUE, seed = NULL, prog = NULL,
    cilevel = 0.95, ...)
```

Arguments

`X`	A data frame, following the time-ordering of the variables.
`Anodes`	A character string of column names in `X` of the intervention variable(s).
`Ynodes`	A character string of column names in `X` of the outcome variable(s).
`Lnodes`	A character string of column names in `X` of the time-dependent (post first treatment) variable(s).
`Cnodes`	A character string of column names in `X` of the censoring variable(s).
`abar`	Numeric vector or matrix of intervention values. See Details.
`survivalY`	Logical. If TRUE, then Y nodes are indicators of an event.
`SL.library`	Either a character vector of prediction algorithms or a list containing character vectors. See details.
`SL.export`	A string vector of user-written learning and screening algorithms that are not part of **SuperLearner**, but are part of the learning library. Only required if `ncores>1`. See details.
`Yweights`	A list of length of `Ynodes`, likely generated with `calc.weights`.
`calc.support`	Logical. If `TRUE`, both crude and conditional support is estimated.
`B`	An integer specifying the number of bootstrap samples to be used, if any.
`ncores`	An integer for the number of threads/cores to be used. If >1, parallelization will be utilized.
`verbose`	Logical. If `TRUE`, notes and warnings are printed.
`seed`	An integer specifying the seed to be used to create reproducable results for parallel computing (i.e. when ncores>1).
`prog`	A character specifying a path where progress should be saved (typically, when `ncores`>1).
`cilevel`	Numeric value between 0 and 1 specifying the confidence level. Defaults to 95%.
`...`	Further arguments to be passed on.

Details

The function calculates the expected counterfactual outcomes (specified under `Ynodes`) under the intervention `abar`.

If `abar` is a vector, then each vector component is used as the intervention value at each time point; that is, interventions which are constant over time are defined. If `abar` is a matrix (of size 'number interventions' x 'time points'), then each row of the length of `Anodes` refers to a particular time-varying intervention strategy.

The nested iterated outcome models are fitted using super learning. The specified prediction algorithms (possibly coupled with algorithms for prior variable screening) are passed on to package **SuperLearner**. See `?SuperLearner` for examples of permitted structures. Note: User-written prediction algorithms, corresponding `S3` prediction functions and screening algorithms need to be specified under `SL.export`, if parallelization is used.

For survival settings, it is required that i) `survivalY=TRUE` and ii) after a Cnode/Ynode is 1, every variable thereafter is set to `NA`. See manual for an example. The package intervenes on Cnodes, i.e. calculates counterfactual outcomes under no censoring.

If `calc.support=TRUE`, conditional and crude support measures (i.e., diagnostics) are calculated as described in Section 3.3.2 of Schomaker et al. (2023).

To parallelize computations automatically, it is sufficient to set `ncores>1`, as appropriate. No further customization or setup is needed, everything will be done by the package. To make estimates under parallelization reproducible, use the `seed` argument. To watch the progress of parallelized computations, set a path in the `prog` argument: then, a text file reports on the progress, which is particularly useful if lengthy bootstrapping computations are required.

Value

Returns an object of of `class` 'gformula':

`results`	matrix of results
`diagnostics`	list of diagnostics and weights based on the estimated support (if `calc.support=TRUE`)
`SL.weights`	matrix of average super learner weights, at each time point
`boot.results`	matrix of bootstrap results
`setup`	list of chosen setup parameters

Author(s)

Michael Schomaker

References

Schomaker M, McIlleron H, Denti P, Diaz I. (2024) *Causal Inference for Continuous Multiple Time Point Interventions*, ArXiv e-prints: *https://arxiv.org/abs/2305.06645*.

See Also

See gformula for parametric g-computation and `calc.weights` on generating outcome weights.

Examples

```
data(EFV)
est <- sgf(X=EFV,
                Lnodes  = c("adherence.1","weight.1",
                            "adherence.2","weight.2",
                            "adherence.3","weight.3",
                            "adherence.4","weight.4"
                ),
                Ynodes  = c("VL.0","VL.1","VL.2","VL.3","VL.4"),
                Anodes  = c("efv.0","efv.1","efv.2","efv.3","efv.4"),
                abar=seq(0,5,1)
)
```

```
est

# Note: replace sgf() with gformula() for parametric g-computation
```

`summary.feasible` *Summarize a* `feasible` *object*

Description

Displays the full summary of an object returned by `feasible`.

Usage

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

Arguments

`object`	An object of class `"feasible"` as returned by `feasible`.
`...`	Unused; included for S3 method compatibility.

Details

The method extracts the data frame stored in the `"summary"` attribute of the `feasible` object. This data frame contains (at least) the following columns:

- `time`: Time index `t`.
- `Strategy`: Index of the intervention strategy.
- `Abar`: The target intervention value at time `t`.
- `Feasible`: Mean of the mapped feasible values for the targeted bin.
- `%infeasible`: Proportion of observations falling below the estimated density threshold for the given `Abar` as targeted.

If the `"summary"` attribute is `NULL`, the method prints “No summary available.”

Value

A data frame containing the summary if available; otherwise `NULL`.

See Also

`feasible`, `plot.feasible`

Index