Package 'CICI'

November 1, 2024

Type Package

Title Causal Inference with Continuous (Multiple Time Point) Interventions

Version 0.9.3

Date 2024-10-29

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

License GPL-2

NeedsCompilation no

Author Michael Schomaker [aut, cre]

Repository CRAN

Date/Publication 2024-11-01 11:10:02 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 implements the standard g-methods approach using the (semi-)parametric *g*-formula, as described in the Schomaker et al. (2024) reference listed below. Weighted dose-response curves that address positivity violations, and are fitted via sequential *g*-computation, are currently only available on GitHub and are not (yet) integrated in this package.

The main function of the package is currently `gformula`.

Details

Package:	CICI
Type:	Package
Version:	0.9.3
Date:	2024-10-29
License:	GPL-2
Depends:	R (>= 4.0)
Imports:	mgcv, glmnet, ggplot2, parallel, doParallel, foreach, doRNG, rngtools

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, in press, see also *https://arxiv.org/abs/2305.06645*.

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, ...)
```

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.
...	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, quantilesn, 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. (2023) 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. (2023) *Causal Inference for Continuous Multiple Time Point Interventions*, ArXiv e-prints: *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. (2023) 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. (2023) *Causal Inference for Continuous Multiple Time Point Interventions*, ArXiv e-prints: *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)
```

---

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, ...)
```

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)
...	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`, ii) the data are in a format where a Ynode stays 1, after it jumps to 1 and ii) after a Cnode/Ynode is 1, every variable thereafter is set to `NA` (except a Ynode which is already 1). 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 3.3.2 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. 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`)
`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

```
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
```

`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, ...)
```

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.
`...`	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)"
```

---

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,
                         survival = FALSE,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".
survival	Logical. If TRUE, time refers to the x-axis (under the condition that it is indeed a survival setting).
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))
```

Index