Package ‘CausalModels’

January 20, 2025

Type Package

Title Causal Inference Modeling for Estimation of Causal Effects

Version 0.2.0

Maintainer Joshua Anderson <jwanderson198@gmail.com>

Description Provides an array of statistical models common in causal inference such as standardization, IP weighting, propensity matching, outcome regression, and doubly-robust estimators. Estimates of the average treatment effects from each model are given with the standard error and a 95% Wald confidence interval (Hernan, Robins (2020) <https://www.hsph.harvard.edu/miguel-hernan/causal-inference-book/>).

License GPL-3

URL https://github.com/ander428/CausalModels

BugReports https://github.com/ander428/CausalModels/issues

Encoding UTF-8

RoxygenNote 7.1.2

Language en-US

Imports stats, causaldata, boot, multcomp, geepack

NeedsCompilation no

Author Joshua Anderson [aut, cre, cph],
Cyril Rakovski [rev],
Yesha Patel [rev],
Erin Lee [rev]

Repository CRAN

Date/Publication 2022-11-23 21:30:02 UTC

Contents

`doubly_robust` *Doubly Robust Model*

Description

``doubly_robust`` trains both an outcome model and a propensity model to generate predictions for the outcome and probability of treatment respectively. By default, the model uses `standardization` and `propensity_scores` to form a doubly-robust model between standardization and IP weighting. Alternatively, any outcome and treatment models can be provided instead, but must be compatible with the `predict` generic function in R. Since many propensity models may not predict probabilities without additional arguments into the predict function, the predictions themselves can be given for both the outcome and propensity scores.

Usage

```
doubly_robust(
  data,
  out.mod = NULL,
  p.mod = NULL,
  f = NA,
  family = gaussian(),
  simple = pkg.env$simple,
  scores = NA,
  p.f = NA,
  p.simple = pkg.env$simple,
  p.family = binomial(),
  p.scores = NA,
  n.boot = 50,
  ...
)
```

Arguments

`data`	a data frame containing the variables in the model. This should be the same data used in `init_params`.
`out.mod`	(optional) a regression model that predicts the outcome. NOTE: the model given must be compatible with the `predict` generic function.
`p.mod`	(optional) a propensity model that predicts the probability of treatment. NOTE: the model given must be compatible with the `predict` generic function.

`f`	(optional) an object of class "formula" that overrides the default parameter
`family`	the family to be used in the general linear model. By default, this is set to `gaussian`.
`simple`	a boolean indicator to build default formula with interactions. If true, interactions will be excluded. If false, interactions will be included. By default, simple is set to false.
`scores`	(optional) use calculated outcome estimates.
`p.f`	(optional) an object of class "formula" that overrides the default formula for the denominator of the IP weighting function.
`p.simple`	a boolean indicator to build default formula with interactions for the propensity models. If true, interactions will be excluded. If false, interactions will be included. By default, simple is set to false. NOTE: if this is changed, the coefficient for treatment may not accurately represent the average causal effect.
`p.family`	the family to be used in the underlying propensity model. By default, this is set to `binomial`.
`p.scores`	(optional) use calculated propensity scores.
`n.boot`	an integer value that indicates number of bootstrap iterations to calculate standard error.
`...`	additional arguments that may be passed to the underlying `glm` model.

Value

`doubly_robust` returns an object of `class` "doubly_robust".

The functions `print`, `summary`, and `predict` can be used to interact with the underlying `glm` model.

An object of class `"doubly_robust"` is a list containing the following:

`out.call`	the matched call of the outcome model.
`p.call`	the matched call of the propensity model.
`out.model`	the underlying outcome model.
`p.model`	the underlying propensity model.
`y_hat`	the estimated outcome values.
`p.scores`	the estimated propensity scores.
`ATE`	the estimated average treatment effect (risk difference).
`ATE.summary`	a data frame containing the ATE, SE, and 95% CI of the ATE.
`data`	the data frame used to train the model.

Examples

```
library(causaldata)
data(nhefs)
nhefs.nmv <- nhefs[which(!is.na(nhefs$wt82)), ]
nhefs.nmv$qsmk <- as.factor(nhefs.nmv$qsmk)

confounders <- c(
  "sex", "race", "age", "education", "smokeintensity",
  "smokeyrs", "exercise", "active", "wt71"
)

init_params(wt82_71, qsmk,
  covariates = confounders,
  data = nhefs.nmv
)

# model using all defaults
model <- doubly_robust(data = nhefs.nmv)
summary(model)

# use alternative outcome model
out.mod <- propensity_matching(data = nhefs.nmv)
db.model <- doubly_robust(
  out.mod = out.mod,
  data = nhefs.nmv
)
db.model

# give calculated outcome predictions and give formula for propensity scores
db.model <- doubly_robust(
  scores = predict(out.mod),
  p.f = qsmk ~ sex + race + age,
  data = nhefs.nmv
)
db.model
```

gestimation *One Parameter G-Estimation of Structural Nested Mean Models*

Description

‘gestimation‘ uses the propensity_scores function to generate inverse probability weights. The weights can either be standardized weights or non-standardized weights. A grid search is done on α to construct the best β coefficient in the structural nested mean model. Alternatively, a linear mean model can be used for a closed form estimator.

Usage

```
gestimation(
  data,
  grid,
  ids = list(),
  f = NA,
  family = binomial(),
  simple = pkg.env$simple,
  p.f = NA,
  p.simple = pkg.env$simple,
  p.family = binomial(),
  p.scores = NA,
  SW = TRUE,
  n.boot = 100,
  type = "one.grid",
  ...
)
```

Arguments

`data`	a data frame containing the variables in the model. This should be the same data used in `init_params`.
`grid`	a list of possible β values that will be used in the grid search.
`ids`	(optional) see documentation for `geeglm`. By default rownames of the data will be used.
`f`	(optional) an object of class "formula" that overrides the default parameter. NOTE: for g-estimation this should be a propensity formula.
`family`	the family to be used in the general linear model. By default, this is set to `gaussian`.
`simple`	(optional) a boolean indicator to build default formula with interactions for the g-estimation model. If true, interactions will be excluded. If false, interactions will be included. By default, simple is set to false. NOTE: β will be appended to the end of the formula
`p.f`	(optional) an object of class "formula" that overrides the default formula for the denominator of the IP weighting function.
`p.simple`	(optional) a boolean indicator to build default formula with interactions for the propensity models. If true, interactions will be excluded. If false, interactions will be included. By default, simple is set to false. NOTE: if this is changed, the coefficient for treatment may not accurately represent the average causal effect.
`p.family`	the family to be used in the underlying propensity model. By default, this is set to `binomial`.
`p.scores`	(optional) use calculated propensity scores for the weights. If using standardized weights, the numerator will still be modeled.
`SW`	a boolean indicator to indicate the use of standardized weights. By default, this is set to true.
`n.boot`	(optional) an integer value that indicates number of bootstrap iterations to calculate standard error. If no value is given, the standard error from the underlying linear model will be used. NOTE: when type is 'one.grid' bootstrapping is not performed. By default, this is set to 100.
`type`	the type of g-estimation to perform. It must be one of `"one.grid"` or `"one.linear"` for a one parameter grid and linear mean model estimation respectively.
`...`	additional arguments that may be passed to the underlying model.

Value

gestimation returns an object of class "gestimation".

The functions print, summary, and predict can be used to interact with the underlying glm or geeglm model.

An object of class "gestimation" is a list containing the following:

call	the matched call.
formula	the formula used in the model.
model	the underlying glm model. If the model performed a grid search, this will be renamed 'best.model'
weights	the estimated IP weights.
type	returns the value used for the 'type' parameter.
ATE	the estimated average treatment effect (risk difference).
ATE.summary	a data frame containing the ATE, SE, and 95% CI of the ATE.

Examples

```
library(causaldata)
data(nhefs)
nhefs.nmv <- nhefs[which(!is.na(nhefs$wt82)), ]
nhefs.nmv$qsmk <- as.factor(nhefs.nmv$qsmk)

confounders <- c(
  "sex", "race", "age", "education", "smokeintensity",
  "smokeyrs", "exercise", "active", "wt71"
)

init_params(wt82_71, qsmk,
  covariates = confounders,
  data = nhefs.nmv
)

gest.model <- gestimation(nhefs.nmv, type = "one.linear", n.boot = 150)
gest.model$ATE.summary
```

---

init_params *Initialize CausalModels Package*

Description

This function is required to be run first before any other function can run. This will set within the package the global outcome, treatment, and covariate functions for each model to use.

Usage

```
init_params(outcome, treatment, covariates, data, simple = FALSE)
```

Arguments

`outcome`	the outcome variable of interest (must be continuous).
`treatment`	the treatment with the causal effect of interest on the outcome.
`covariates`	a list/vector of covariate names to be use for confounding adjustment.
`data`	a data frame containing the variables in the model.
`simple`	a boolean indicator to build default formula with interactions. If true, interactions will be excluded. If false, interactions will be included. By default, simple is set to false.

Value

No return value. Called for parameter initialization.

`ipweighting` *Parametric IP Weighting*

Description

'ipweighting' uses the `propensity_scores` function to generate inverse probability weights. The weights can either be standardized weights or non-standardized weights. The weights are used to train a general linear model whose coefficient for treatment represents the average treatment effect on the additive scale.

Usage

```
ipweighting(
  data,
  f = NA,
  family = gaussian(),
  p.f = NA,
  p.simple = pkg.env$simple,
  p.family = binomial(),
  p.scores = NA,
  SW = TRUE,
  n.boot = 0,
  ...
)
```

Arguments

`data`	a data frame containing the variables in the model. This should be the same data used in `init_params`.
`f`	(optional) an object of class "formula" that overrides the default parameter
`family`	the family to be used in the general linear model. By default, this is set to `gaussian`.

`p.f`	(optional) an object of class "formula" that overrides the default formula for the denominator of the IP weighting function.
`p.simple`	a boolean indicator to build default formula with interactions for the propensity models. If true, interactions will be excluded. If false, interactions will be included. By default, simple is set to false. NOTE: if this is changed, the coefficient for treatment may not accurately represent the average causal effect.
`p.family`	the family to be used in the underlying propensity model. By default, this is set to `binomial`.
`p.scores`	(optional) use calculated propensity scores for the weights. If using standardized weights, the numerator will still be modeled.
`SW`	a boolean indicator to indicate the use of standardized weights. By default, this is set to true.
`n.boot`	(optional) an integer value that indicates number of bootstrap iterations to calculate standard error. If no value is given, the standard error from the underlying linear model will be used.
`...`	additional arguments that may be passed to the underlying `glm` model.

Value

`ipweighting` returns an object of `class` `"ipweighting"`.

The functions `print`, `summary`, and `predict` can be used to interact with the underlying `glm` model.

An object of class `"ipweighting"` is a list containing the following:

`call`	the matched call.
`formula`	the formula used in the model.
`model`	the underlying glm model.
`weights`	the estimated IP weights.
`ATE`	the estimated average treatment effect (risk difference).
`ATE.summary`	a data frame containing the ATE, SE, and 95% CI of the ATE.

Examples

```
library(causaldata)
data(nhefs)
nhefs.nmv <- nhefs[which(!is.na(nhefs$wt82)), ]
nhefs.nmv$qsmk <- as.factor(nhefs.nmv$qsmk)

confounders <- c(
  "sex", "race", "age", "education", "smokeintensity",
  "smokeyrs", "exercise", "active", "wt71"
)

init_params(wt82_71, qsmk,
  covariates = confounders,
  data = nhefs.nmv
)
```

```
# model using all defaults
model <- ipweighting(data = nhefs.nmv)
summary(model)

# Model using calculated propensity scores and manual outcome formula
p.scores <- propensity_scores(nhefs.nmv)$p.scores
model <- ipweighting(wt82_71 ~ qsmk, p.scores = p.scores, data = nhefs.nmv)
summary(model)
```

---

iv_est *Standard Instrumental Variable Estimator*

Description

'iv_est' calculates the standard IV estimand using the conditional means on a given instrumental variable.

Usage

```
iv_est(IV, data, n.boot = 50)
```

Arguments

`IV`	the instrumental variable to be used in the conditional means. Must be a factor with no more than 2 levels. It is assumed the second level is the positive level, i.e., the binary equivalent of the second factor level should be 1 and the first should be 0.
`data`	a data frame containing the variables in the model. This should be the same data used in `init_params`.
`n.boot`	an integer value that indicates number of bootstrap iterations to calculate standard error.

Value

`iv_est` returns a data frame containing the standard IV estimate, standard error, and Wald 95

Examples

```
library(causaldata)
data(nhefs)
nhefs.nmv <- nhefs[which(!is.na(nhefs$wt82)), ]
nhefs.nmv$qsmk <- as.factor(nhefs.nmv$qsmk)

confounders <- c(
  "sex", "race", "age", "education", "smokeintensity",
  "smokeyrs", "exercise", "active", "wt71"
)
```

```
nhefs.iv <- nhefs[which(!is.na(nhefs$wt82) & !is.na(nhefs$price82)), ]
nhefs.iv$highprice <- as.factor(ifelse(nhefs.iv$price82 >= 1.5, 1, 0))
nhefs.iv$qsmk <- as.factor(nhefs.iv$qsmk)
init_params(wt82_71, qsmk,
            covariates = confounders,
            data = nhefs.iv)

iv_est("highprice", nhefs.iv)
```

`outcome_regression` *Outcome Regression*

Description

‘outcome_regression‘ builds a linear model using all covariates. The treatment effects are stratified within the levels of the covariates. The model will automatically provide all discrete covariates in a contrast matrix. To view estimated change in treatment effect from continuous variables, a list called `contrasts`, needs to be given with specific values to estimate. A vector of values can be given for any particular continuous variable.

Usage

```
outcome_regression(
  data,
  f = NA,
  simple = pkg.env$simple,
  family = gaussian(),
  contrasts = list(),
  ...
)
```

Arguments

`data`	a data frame containing the variables in the model. This should be the same data used in `init_params`.
`f`	(optional) an object of class "formula" that overrides the default parameter
`simple`	a boolean indicator to build default formula with interactions. If true, interactions will be excluded. If false, interactions will be included. By default, simple is set to false.
`family`	the family to be used in the general linear model. By default, this is set to `gaussian`. NOTE: if this is changed, the assumptions about the model output may be incorrect and may not provide accurate treatment effects.
`contrasts`	a list of continuous covariates and values in the model to be included in the contrast matrix (e.g. `list(age = c(18, 25, 40), weight = c(90, 159))`).
`...`	additional arguments that may be passed to the underlying `glht` model.

Value

`outcome_regression` returns an object of `class` `"outcome_regression"`

The functions `print`, `summary`, and `predict` can be used to interact with the underlying `glht` model.

An object of class `"outcome_regression"` is a list containing the following:

`call`	the matched call.
`formula`	the formula used in the model.
`model`	the underlying glht model.
`ATE`	a data frame containing the ATE, SE, and 95% CI of the ATE.
`ATE.summary`	a more detailed summary of the ATE estimations from glht.

Examples

```
library(causaldata)
library(multcomp)

data(nhefs)
nhefs.nmv <- nhefs[which(!is.na(nhefs$wt82)), ]
nhefs.nmv$qsmk <- as.factor(nhefs.nmv$qsmk)

confounders <- c(
  "sex", "race", "age", "education", "smokeintensity",
  "smokeyrs", "exercise", "active", "wt71"
)

init_params(wt82_71, qsmk,
  covariates = confounders,
  data = nhefs.nmv
)

out.mod <- outcome_regression(nhefs.nmv, contrasts = list(
  age = c(21, 55),
  smokeintensity = c(5, 20, 40)
))
print(out.mod)
summary(out.mod)
head(data.frame(preds = predict(out.mod)))
```

`propensity_matching` *Propensity Matching*

Description

'propensity_matching' uses either stratification or standardization to model an outcome conditional on the propensity scores. In stratification, the model will break the propensity scores into groups and output a `glht` model based off a contrast matrix which estimates the change in average causal effect within groups of propensity scores. In standardization, the model will output a `standardization` model that conditions on the propensity strata rather than the covariates. The model can also predict the expected outcome.

Usage

```
propensity_matching(
  data,
  f = NA,
  simple = pkg.env$simple,
  p.scores = NA,
  p.simple = pkg.env$simple,
  type = "strata",
  grp.width = 0.1,
  quant = TRUE,
  ...
)
```

Arguments

`data`	a data frame containing the variables in the model. This should be the same data used in `init_params`.
`f`	(optional) an object of class "formula" that overrides the default parameter
`simple`	a boolean indicator to build default formula with interactions. If true, interactions will be excluded. If false, interactions will be included. By default, simple is set to false.
`p.scores`	(optional) use calculated propensity scores for matching. Otherwise, propensity scores will be automatically modeled.
`p.simple`	a boolean indicator to build default formula with interactions for the propensity models. If true, interactions will be excluded. If false, interactions will be included. By default, simple is set to false.
`type`	a string representing the type of propensity model to be used. By default, the function will stratify. Standardization with propensity scores may also be used. The value given for `type` must be in `c("strata", "stdm")`.
`grp.width`	a decimal value to specify the range to stratify the propensity scores. If option `quant` is set to true, this will represent the spread of percentiles. If false, it will represent the spread of raw values of propensity scores. Must be a decimal between 0 and 1. By default, this is set to 0.1. This option is ignored for standardization.
`quant`	a boolean indicator to specify the type of stratification. If true (default), the model will stratify by percentiles. If false, the scores will be grouped by a range of their raw values. This option is ignored for standardization.

`...`	additional arguments that may be passed to the underlying `propensity_scores` function.

Value

`propensity_matching` returns an object of `class` `"propensity_matching"`

The functions `print`, `summary`, and `predict` can be used to interact with the underlying `glht` or `standardization` model.

An object of class `"propensity_matching"` is a list containing the following:

`call`	the matched call.
`formula`	the formula used in the model.
`model`	either the underlying `glht` or `standardization` model.
`p.scores`	the estimated propensity scores
`ATE`	a data frame containing the ATE, SE, and 95% CI of the ATE.
`ATE.summary`	either a data frame containing the `glht` or `standardization` summary.

Examples

```
library(causaldata)
library(multcomp)

data(nhefs)
nhefs.nmv <- nhefs[which(!is.na(nhefs$wt82)), ]
nhefs.nmv$qsmk <- as.factor(nhefs.nmv$qsmk)

confounders <- c(
  "sex", "race", "age", "education", "smokeintensity",
  "smokeyrs", "exercise", "active", "wt71"
)

init_params(wt82_71, qsmk,
  covariates = confounders,
  data = nhefs.nmv
)

pm.model <- propensity_matching(nhefs.nmv)
pm.model$ATE.summary
summary(pm.model)
head(data.frame(preds = predict(pm.model)))
```

propensity_scores *Propensity Scores*

Description

‘propensity_scores‘ builds a logistic regression with the target as the treatment variable and the covariates as the independent variables.

Usage

```
propensity_scores(
  data,
  f = NA,
  simple = pkg.env$simple,
  family = binomial(),
  ...
)
```

Arguments

`data`	a data frame containing the variables in the model. This should be the same data used in `init_params`.
`f`	(optional) an object of class "formula" that overrides the default parameter
`simple`	a boolean indicator to build default formula with interactions. If true, interactions will be excluded. If false, interactions will be included. By default, simple is set to false.
`family`	the family to be used in the general linear model. By default, this is set to `binomial` NOTE: if this is changed, the outcome of the model may not be the probabilities and the results will not be valid.
`...`	additional arguments that may be passed to the underlying `glm` model.

Value

`propensity_scores` returns an object of `class` `"propensity_scores"`

The functions `print`, `summary`, and `predict` can be used to interact with the underlying `glm` model.

An object of class `"propensity_scores"` is a list containing the following:

`call`	the matched call.
`formula`	the formula used in the model.
`model`	the underlying glm model.
`p.scores`	the estimated propensity scores.

Examples

```
library(causaldata)
data(nhefs)
nhefs.nmv <- nhefs[which(!is.na(nhefs$wt82)), ]
nhefs.nmv$qsmk <- as.factor(nhefs.nmv$qsmk)

confounders <- c(
  "sex", "race", "age", "education", "smokeintensity",
  "smokeyrs", "exercise", "active", "wt71"
)

init_params(wt82_71, qsmk,
  covariates = confounders,
  data = nhefs.nmv
)

p.score <- propensity_scores(nhefs.nmv)
p.score
```

`standardization` *Parametric Standardization*

Description

‘standardization‘ uses a standard `glm` linear model to perform parametric standardization by adjusting bias through including all confounders as covariates. The model will calculate during training both the risk difference and the risk ratio. Both can be accessed from the model as well as estimates of the counterfactuals of treatment.

Usage

```
standardization(
  data,
  f = NA,
  family = gaussian(),
  simple = pkg.env$simple,
  n.boot = 50,
  ...
)
```

Arguments

`data`	a data frame containing the variables in the model. This should be the same data used in `init_params`.
`f`	(optional) an object of class "formula" that overrides the default parameter
`family`	the family to be used in the general linear model. By default, this is set to `gaussian`.

`simple`	a boolean indicator to build default formula with interactions. If true, interactions will be excluded. If false, interactions will be included. By default, simple is set to false. NOTE: if this is changed, the coefficient for treatment may not accurately represent the average causal effect.
`n.boot`	an integer value that indicates number of bootstrap iterations to calculate standard error.
`...`	additional arguments that may be passed to the underlying `glm` model.

Value

`standardization` returns an object of `class` `"standardization"`.

The functions `print`, `summary`, and `predict` can be used to interact with the underlying `glm` model.

An object of class `"standardization"` is a list containing the following:

`call`	the matched call.
`formula`	the formula used in the model.
`model`	the underlying glm model.
`ATE`	a data frame containing estimates of the treatment effect of the observed, counterfactuals, and risk metrics.
`ATE.summary`	a data frame containing the ATE, SE, and 95% CI of the ATE.

Examples

```
library(causaldata)

data(nhefs)
nhefs.nmv <- nhefs[which(!is.na(nhefs$wt82)), ]
nhefs.nmv$qsmk <- as.factor(nhefs.nmv$qsmk)

confounders <- c(
  "sex", "race", "age", "education", "smokeintensity",
  "smokeyrs", "exercise", "active", "wt71"
)

init_params(wt82_71, qsmk,
  covariates = confounders,
  data = nhefs.nmv
)

# model using all defaults
model <- standardization(data = nhefs.nmv)
print(model)
summary(model)
print(model$ATE.summary)
print(model$ATE.summary$Estimate[[2]] -
  model$ATE.summary$Estimate[[3]]) # manually calculate risk difference
```

Index