Package 'LRErdd'

August 6, 2026

Type Package

Title Regression Discontinuity Designs as Local Randomized Experiments

Version 0.1.0

Description A set of functions for the design and analysis of Regression Discontinuity Designs as local randomized experiments within the potential outcome approach as formalized in Li, Mattei and Mealli (2015) <doi:10.1214/15-AOAS809>. A subset of functions implements the design phase of the study, where the focus is on the selection of suitable subpopulations for which valid causal inference can be drawn. These functions provide summary statistics of pre- and post-treatment variables by treatment status and select suitable subpopulations around the threshold where pre-treatment variables are well balanced between treatment groups, using randomization-based tests with adjustment for multiplicities. Functions for a visual inspection of the results are also provided. Finally, the package includes a set of functions for drawing inference on causal effects for the selected subpopulations using randomization-based modes of inference. Specifically, the Fisher Exact p-value and Neyman approaches are implemented for the analysis of both sharp and fuzzy Regression Discontinuity designs. The approach is illustrated in a study concerning the effects of university grants on student dropout.

License MIT + file LICENSE

Encoding UTF-8

Depends R (>= 3.5)

Imports ggplot2, cowplot, gtools, R6, shiny, stats

Suggests knitr, rmarkdown, bslib, shinycssloaders, DT, openxlsx, testthat (>= 3.0.0)

VignetteBuilder knitr

RoxygenNote 7.3.1

LazyData true

NeedsCompilation no

Author Ibon Tamayo [aut, cre], Alessandra Mattei [aut],

Fabrizia Mealli [aut],
Marie-Abele Bind [aut]

Maintainer Ibon Tamayo <itamuria@gmail.com>

Repository CRAN

Date/Publication 2026-08-06 10:20:08 UTC

Contents

`all_binom2num` *From binomial to numerical*

Description

all_binom2num() analizes a data frame, detects the binomial columns and convert them to numeric format.

Usage

```
all_binom2num(data)
```

Arguments

`data`	dataset

Details

In the shiny application we need to verify some conditions and this function helps us to check it.

Value

A new dataset. Convert every binomial column in the dataset as numerical to avoid errors

Examples

```
df <- data.frame(
gp = factor(rep(letters[1:3], each = 10)),
y = rnorm(30),
a = sample(c('0','1'), 30, replace = TRUE))

all_binom2num(df)
```

`cutting_range` *Selection of the cutting range*

Description

cutting_range() defines the values of the forcing values taking into account the threshold, the type of selection and the range value.

Usage

```
cutting_range(forcingvalues, threshold, typerange, range_value)
```

Arguments

`forcingvalues`	a vector with forcing variables that will be used to defined the cutting range
`threshold`	a numeric value that define the threshold to create the two groups
`typerange`	a character value that will define the type of selection that we want to carry out. It can be 'percentage', 'balunit' or 'unbalunit'. In the case or percentage, the reduction that we want to do is defined by the proportion of all the range. Instead of proportion, if we want to reduce by an accurate number we can use the other two options. In the case of balunit, the selection will be balanced to the two sides of the threshold. In the case of unbalunit, the selection can be unbalanced and we will need to give two numerical values in the range_value parameter.
`range_value`	a numerical (when typerange is percentage or balunit) and two numerica values (in the case of unbalunit) that define the value to create the range.

Value

a list with two values that define the lower and upper values to create the selection of the data

---

`da.bin` *Data Augmentation (DA) step of Markov Chain Monte Carlo (MCMC) algorithm to derive the posterior median of the Complier Average Causal Effect (CACE) for binary outcomes*

Description

da.bin() implements the Data Augmentation (DA) method to impute the compliance status of each unit for binary outcomes in the presence of one-sided noncompliance under exclusion restriction for never-takers

Usage

```
da.bin(theta, dat)
```

Arguments

theta	A list with the values of the parameters: pi.c = proportion of compliers; py.c0, py.c1 = probabilities that the outcome takes on value 1 for compliers under control and under treatment; py.nt = probability that the outcome takes on value 1 for never-takers
dat	Data frame

Value

A vector with the imputed compliance status for each unit in the data set

---

da.bin.h0 *Data Augmentation (DA) step of Markov Chain Monte Carlo (MCMC) algorithm to derive the posterior median of the Complier Average Causal Effect (CACE) for a binary outcome under the sharp null hypothesis of no effect*

Description

da.bin.h0() implements the Data Augmentation (DA) method to impute the compliance status of each unit for binary outcomes in the presence of one-sided noncompliance under exclusion restriction for never-takers and under the sharp null hypothesis of no effect

Usage

```
da.bin.h0(theta, dat)
```

Arguments

theta	A list with the values of the parameters: pi.c = proportion of compliers; py.c = probability that the outcome takes on value 1 for compliers; py.nt = probability that the outcome takes on value 1 for never-takers
dat	Data frame

Value

A vector with the imputed compliance status for each unit in the data set under the sharp null hypothesis of no effect

da.bin.h02 — *Data Augmentation (DA) step of Markov Chain Monte Carlo (MCMC) algorithm to derive the posterior median of the Complier Average Causal Effect (CACE) for a binary outcome under the sharp null hypothesis of no effect*

Description

da.bin.h02() implements the Data Augmentation (DA) method to impute the compliance status of each unit for binary outcomes in the presence of two-sided noncompliance under monotonicity (no defiers) and exclusion restrictions for both never-takers and always-takers and under the sharp null hypothesis of no effect

Usage

```
da.bin.h02(theta, dat)
```

Arguments

theta	A list with the values of the parameters: pi.c = proportion of compliers; py.c = probability that the outcome takes on value 1 for compliers; py.nt = probability that the outcome takes on value 1 for never-takers py.at = probability that the outcome takes on value 1 for always-takers
dat	Data frame

Value

A vector with the imputed compliance status for each unit in the data set under the sharp null hypothesis of no effect

da.bin2 — *Data Augmentation (DA) step of Markov Chain Monte Carlo (MCMC) algorithm to derive the posterior median of the Complier Average Causal Effect (CACE) for binary outcomes*

Description

da.bin2() implements the Data Augmentation (DA) method to impute the compliance status of each unit for binary outcomes in the presence of two-sided noncompliance under monotonicity (no defiers) and exclusion restrictions for both never-takers and always-takers

Usage

```
da.bin2(theta, dat)
```

Arguments

theta	A list with the values of the parameters: pi.c = proportion of compliers; py.c0, py.c1 = probabilities that the outcome takes on value 1 for compliers under control and under treatment; py.nt = probability that the outcome takes on value 1 for never-takers; py.at = probability that the outcome takes on value 1 for always-takers
dat	Data frame

Value

A vector with the imputed compliance status for each unit in the data set

---

da.gauss *Data Augmentation (DA) step of Markov Chain Monte Carlo (MCMC) algorithm to derive the posterior median of the Complier Average Causal Effect (CACE) for Gaussian outcomes*

Description

da.gauss() implements the Data Augmentation (DA) method to impute the compliance status of each unit for normally distributed outcomes in the presence of one-sided noncompliance under exclusion restriction for never-takers

Usage

```
da.gauss(theta, dat)
```

Arguments

theta	A list with the values of the parameters: pi.c = proportion of compliers; mu.c0, sigma2.c0, mu.c1, sigma2.c1 = means and variances of the Normal distributions of the outcome for compliers under control and under treatment; mu.nt, sigma2.nt = mean and variance of the Normal distribution of the outcome for never-takers
dat	Data frame

Value

A vector with the imputed compliance status for each unit in the data set

---

`da.gauss.h0` *Data Augmentation (DA) step of Markov Chain Monte Carlo (MCMC) algorithm to derive the posterior median of the Complier Average Causal Effect (CACE) for a Gaussian outcome under the sharp null hypothesis of no effect*

Description

da.gauss.h0() implements the Data Augmentation (DA) method to impute the compliance status of each unit for a continuous normally distributed outcome in the presence of one-sided noncompliance under exclusion restriction for never-takers and under the sharp null hypothesis of no effect

Usage

```
da.gauss.h0(theta, dat)
```

Arguments

`theta`	A list with the values of the parameters: pi.c = proportion of compliers; mu.c, sigma2.c = means and variances of the Normal distribution of the outcome for compliers; mu.nt, sigma2.nt = mean and variance of the Normal distribution of the outcome for never-takers
`dat`	Data frame

Value

A vector with the imputed compliance status for each unit in the data set under the sharp null hypothesis of no effect

---

`da.gauss.h02` *Data Augmentation (DA) step of Markov Chain Monte Carlo (MCMC) algorithm to derive the posterior median of the Complier Average Causal Effect (CACE) for a Gaussian outcome under the sharp null hypothesis of no effect*

Description

da.gauss.h02() implements the Data Augmentation (DA) method to impute the compliance status of each unit for a continuous normally distributed outcome in the presence of two-sided noncompliance under monotonicity (no defiers) and exclusion restrictions for both never-takers and always-takers and under the sharp null hypothesis of no effect

Usage

```
da.gauss.h02(theta, dat)
```

Arguments

theta	A list with the values of the parameters: pi.c = proportion of compliers; mu.c, sigma2.c = means and variances of the Normal distribution of the outcome for compliers; mu.nt, sigma2.nt = mean and variance of the Normal distribution of the outcome for never-takers; mu.at, sigma2.at = mean and variance of the Normal distribution of the outcome for always-takers
dat	Data frame

Value

A vector with the imputed compliance status for each unit in the data set under the sharp null hypothesis of no effect

---

da.gauss2 — *Data Augmentation (DA) step of Markov Chain Monte Carlo (MCMC) algorithm to derive the posterior median of the Complier Average Causal Effect (CACE) for Gaussian outcomes*

Description

da.gauss2() implements the Data Augmentation (DA) method to impute the compliance status of each unit for normally distributed outcomes in the presence of two-sided noncompliance under monotonicity (no defiers) and exclusion restrictions for both never-takers and always-takers

Usage

```
da.gauss2(theta, dat)
```

Arguments

theta	A list with the values of the parameters: pi.c = proportion of compliers; mu.c0, sigma2.c0, mu.c1, sigma2.c1 = means and variances of the Normal distributions of the outcome for compliers under control and under treatment; mu.nt, sigma2.nt = mean and variance of the Normal distribution of the outcome for never-takers; mu.at, sigma2.at = mean and variance of the Normal distribution of the outcome for always-takers
dat	Data frame

Value

A vector with the imputed compliance status for each unit in the data set

`E.step.bin` *Expectation step of the Expectation-Maximization (EM) algorithm to calculate the MLE of the Complier Average Causal Effect (CACE) for binary outcomes*

Description

E.step.bin() implements the Expectation step of the EM algorithm to calculate the MLE of CACE for binary outcomes in the presence of one-sided noncompliance under exclusion restriction for never-takers

Usage

```
E.step.bin(theta, dat)
```

Arguments

`theta`	A list with the values of the parameters: pi.c = proportion of compliers; py.c0, py.c1 = probabilities that the outcome takes on value 1 for compliers under control and under treatment; py.nt = probability that the outcome takes on value 1 for never-takers
`dat`	Data frame

Value

A vector with the expected compliance status for each unit in the data set

`E.step.bin2` *Expectation step of the Expectation-Maximization (EM) algorithm to calculate the MLE of the Complier Average Causal Effect (CACE) for binary outcomes*

Description

E.step.bin2() implements the Expectation step of the EM algorithm to calculate the MLE of CACE for binary outcomes in the presence of two-sided noncompliance under monotonicity (no defiers) and exclusion restrictions for both never-takers and always-takers

Usage

```
E.step.bin2(theta, dat)
```

Arguments

theta	A list with the values of the parameters: pi.c = proportion of compliers; py.c0, py.c1 = probabilities that the outcome takes on value 1 for compliers under control and under treatment; py.nt = probability that the outcome takes on value 1 for never-takers; py.at = probability that the outcome takes on value 1 for always-takers
dat	Data frame

Value

A vector with the expected compliance status for each unit in the data set

---

E.step.gauss *Expectation step of the Expectation-Maximization (EM) algorithm to calculate the MLE of the Complier Average Causal Effect (CACE) for Gaussian outcomes*

Description

E.step.gauss() implements the Expectation step of the EM algorithm to calculate the MLE of CACE for continuous normally distributed outcomes in the presence of one-sided noncompliance under exclusion restriction for never-takers

Usage

```
E.step.gauss(theta, dat)
```

Arguments

theta	A list with the values of the parameters: pi.c = proportion of compliers; mu.c0, sigma2.c0, mu.c1, sigma2.c1 = means and variances of the Normal distributions of the outcome for compliers under control and under treatment; mu.nt, sigma2.nt = mean and variance of the Normal distribution of the outcome for never-takers
dat	Data frame

Value

A vector with the expected compliance status for each unit in the data set

E.step.gauss2 — *Expectation step of the Expectation-Maximization (EM) algorithm to calculate the MLE of the Complier Average Causal Effect (CACE) for Gaussian outcomes*

Description

E.step.gauss2() implements the Expectation step of the EM algorithm to calculate the MLE of CACE for continuous normally distributed outcomes in the presence of two-sided noncompliance under monotonicity (no defiers) and exclusion restrictions for both never-takers and always-takers

Usage

```
E.step.gauss2(theta, dat)
```

Arguments

theta	A list with the values of the parameters: pi.c = proportion of compliers; mu.c0, sigma2.c0, mu.c1, sigma2.c1 = means and variances of the Normal distributions of the outcome for compliers under control and under treatment; mu.nt, sigma2.nt = mean and variance of the Normal distribution of the outcome for never-takers; mu.at, sigma2.at = mean and variance of the Normal distribution of the outcome for always-takers
dat	Data frame

Value

A vector with the expected compliance status for each unit in the data set

EM.bin — *Maximum Likelihood Estimate (MLE) of the Complier Average Causal Effect (CACE) for binary outcomes*

Description

EM.bin() uses the Expectation-Maximization (EM) algorithm to calculate the MLEs of CACE for binary outcomes in the presence of one-sided noncompliance under exclusion restriction for never-takers

Usage

```
EM.bin(dat, tol = 1e-04, maxit = 1000)
```

Arguments

dat	Data frame
tol	The convergence tolerance. Convergence is reached when the maximum absolute difference between the parameter values on consecutive step is not greater than @tol. Defaults to 1e-04
maxit	The maximum number of iterations. Defaults to 1000

Value

The value of the MLE of CACE for binary outcomes (at either convergence or the last iteration if the maximum number of iterations has been reached before convergence)

EM.bin2 — *Maximum Likelihood Estimate (MLE) of the Complier Average Causal Effect (CACE) for binary outcomes*

Description

EM.bin2() uses the Expectation-Maximization (EM) algorithm to calculate the MLEs of CACE for binary outcomes in the presence of two-sided noncompliance under monotonicity (no defiers) and exclusion restrictions for both never-takers and always-takers

Usage

```
EM.bin2(dat, tol = 1e-04, maxit = 1000)
```

Arguments

dat	Data frame
tol	The convergence tolerance. Convergence is reached when the maximum absolute difference between the parameter values on consecutive step is not greater than @tol. Defaults to 1e-04
maxit	The maximum number of iterations. The default is 1000

Value

The value of the MLE of CACE for binary outcomes (at either convergence or the last iteration if the maximum number of iterations has been reached before convergence)

EM.gauss — *Maximum Likelihood Estimate (MLE) of the Complier Average Causal Effect (CACE) for Gaussian outcomes*

Description

EM.gauss() uses the Expectation-Maximization (EM) algorithm to calculate the MLEs of CACE for continuous normally distributed outcomes in the presence of one-sided noncompliance under exclusion restriction for never-takers

Usage

```
EM.gauss(dat, tol = 1e-04, maxit = 1000)
```

Arguments

dat	Data frame
tol	The convergence tolerance Convergence is reached when the maximum absolute difference between the parameter values on consecutive step is not greater than @tol The default is 1e-04
maxit	The maximum number of iterations. The default is 1000

Value

The value of the MLE of CACE for continuous outcomes (at either convergence or the last iteration if the maximum number of iterations has been reached before convergence)

EM.gauss2 — *Maximum Likelihood Estimate (MLE) of the Complier Average Causal Effect (CACE) for Gaussian outcomes*

Description

EM.gauss2() uses the Expectation-Maximization (EM) algorithm to calculate the MLEs of CACE for continuous normally distributed outcomes in the presence of two-sided noncompliance under monotonicity (no defiers) and exclusion restrictions for both never-takers and always-takers

Usage

```
EM.gauss2(dat, tol = 1e-04, maxit = 1000)
```

Arguments

dat	Data frame
tol	The convergence tolerance Convergence is reached when the maximum absolute difference between the parameter values on consecutive step is not greater than @tol The default is 1e-04
maxit	The maximum number of iterations. The default is 1000

Value

The value of the MLE of CACE for continuous outcomes (at either convergence or the last iteration if the maximum number of iterations has been reached before convergence)

---

fuzzy_fep1sided *Fuzzy - FEP approach for binary one sided*

Description

Fuzzy - FEP approach for binary one sided

Usage

```
fuzzy_fep1sided(dataset, Y, W, Z, Y_name, M2 = 10)
```

Arguments

dataset	The data frame with the variables
Y	Y A numerical vector with outcome information
W	A vector that defines the treatment receipt status.
Z	A numerical vector with the forcing variable in binary format.
Y_name	A character format of the name of the outcome in the dataset
M2	number of iterations

Details

This is for the cases where the noncompliance is one sided and binary

Value

A list with values of pppv, Tsim, Tobs

fuzzy_fep2sided *Fuzzy - FEP binary two sided*

Description

Fuzzy - FEP binary two sided

Usage

```
fuzzy_fep2sided(dataset, Y, W, Z, Y_name, M2 = 10)
```

Arguments

dataset	The data frame with the variables
Y	Y A numerical vector with outcome information
W	A vector that defines the treatment receipt status.
Z	A numerical vector with the forcing variable in binary format.
Y_name	A character format of the name of the outcome in the dataset
M2	number of iterations

Details

This is for the cases where the noncompliance is two sided and binary

Value

A list with values of pppv, Tsim, Tobs

fuzzy_fep_bw *Bandwidth selection for Fuzzy - FEP bandwidth*

Description

fuzzy_fep() was defined to calculate just for one dataset. For the cases where the user wants to test several bandwidths this functions is applied.

Usage

```
fuzzy_fep_bw(
  dataset,
  forcing_var_name = "S",
  Y_name = "dropout",
  niter = 1000,
  W_name = "W",
  bandwidth = c(500, 1000, 1500),
  cut_value = 15000,
  M2 = 5,
  whichunder = 1,
  typemod = "binary",
  typesided = "onesided"
)
```

Arguments

dataset	The data frame with the variables
forcing_var_name	forcing_var_name
Y_name	A character format of the name of the outcome in the dataset
niter	Number of iteration to calculate the adjusted p-value.
W_name	The name that defines the treatment receipt status.
bandwidth	A numerical vector defining the ranges to subset the dataset and create the sub-populations of interest.
cut_value	Threshold value to separate the treatment status.
M2	number of iterations
whichunder	If the treatment status is lower (0) or higher (1) than the threshold
typemod	If the outcome is binary or numerical
typesided	If the treatment receipt status is onesided or twosided

Value

data frame with variable and value and bandwidth

---

fuzzy_fep_numeric1sided

Fuzzy - FEP 1 sided numerical

Description

Fuzzy - FEP 1 sided numerical

Usage

```
fuzzy_fep_numeric1sided(dataset, Y, W, Z, Y_name, M2 = 10)
```

Arguments

dataset	The data frame with the variables
Y	Y A numerical vector with outcome information
W	A vector that defines the treatment receipt status.
Z	A numerical vector with the forcing variable in binary format.
Y_name	A character format of the name of the outcome in the dataset
M2	number of iterations

Details

This is for the cases where the noncompliance is two sided and numerical

Value

A list with values of pppv, Tsim, Tobs

---

fuzzy_fep_numeric2sided

Fuzzy - FEP 2 sided numerical

Description

Fuzzy - FEP 2 sided numerical

Usage

```
fuzzy_fep_numeric2sided(dataset, Y, W, Z, Y_name, M2 = 10)
```

Arguments

dataset	The data frame with the variables
Y	Y A numerical vector with outcome information
W	A vector that defines the treatment receipt status.
Z	A numerical vector with the forcing variable in binary format.
Y_name	A character format of the name of the outcome in the dataset
M2	number of iterations

Details

This is for the cases where the noncompliance is one sided and numerical

Value

A list with values of pppv, Tsim, Tobs

`fuzzy_neyman` *Fuzzy - neyman approach*

Description

fuzzy_neyman() calculates the causal effect with the fuzzy neyman approach

Usage

```
fuzzy_neyman(Y, Wh, Z, cin = 0.05)
```

Arguments

`Y`	A numerical vector with outcome information
`Wh`	A vector that defines the treatment receipt status.
`Z`	A numerical vector with the forcing variable in binary format
`cin`	Alpha level for the confidence intervals (e.g. the default 0.05 yields 95% confidence intervals)

Details

The focus will be on the sharp nullhypothesis that disallows any effect of assignment to treatment versus control for compliersbelonging toUs0:H0:Yi(0) =Yi(1)for alli?\210\210 Us0withGi=c, which can be interpreted asthe null hypothesis of no effect of ?\200?receipt?\200\235 of treatment on outcome for the subpopulationof compliers belonging toUs.

Value

A vector with information about the effects and confidence intervals.

`fuzzy_neyman_bw` *Bandwidth selection for Fuzzy - neyman bandwidth*

Description

fuzzy_neyman() was defined to calculate just for one dataset. For the cases where the user wants to test several bandwidths this functions is applied.

Usage

```
fuzzy_neyman_bw(
  dataset,
  forcing_var_name = "S",
  Y_name = "dropout",
  niter = 1000,
  W = "W",
  bandwidth = c(500, 1000, 1500),
  cut_value = 15000,
  whichunder = 1,
  cin = 0.05
)
```

Arguments

dataset	The data frame with the variables
forcing_var_name	The name of the forcing variable in the dataset.
Y_name	A character format of the name of the outcome in the dataset
niter	Number of iteration to calculate the adjusted p-value.
W	The name that defines the treatment receipt status in the dataset.
bandwidth	A numerical vector defining the ranges to subset the dataset and create the sub-populations of interest.
cut_value	Threshold value to separate the treatment status.
whichunder	If the treatment status is lower (0) or higher (1) than the threshold
cin	Alpha level for the confidence intervals (e.g. the default 0.05 yields 95% confidence intervals)

Value

A data frame with effect and bandwidths

---

grants *Italian university grants and student dropout*

Description

Data from an Italian study concerning the effects of university grants on student dropout (Li, Mattei and Mealli, 2015). Eligibility for the grant is determined by a measure of the economic situation of the student's family falling below a threshold of 15,000 euros, which makes it a natural regression discontinuity design.

Usage

```
data(grants)
```

Format

A data frame with 15984 rows (students) and 21 columns:

S Forcing variable: measure of the family's economic situation (in euros); students with `S` below 15,000 are eligible for the grant

Z Eligibility indicator (1 = eligible, i.e. `S` below the threshold; 0 = not eligible)

A Grant application indicator (1 = applied)

W Treatment received: grant receipt indicator (1 = received the grant)

dropout Outcome: 1 if the student dropped out, 0 otherwise

sex Student's sex (binary indicator)

HSHumanity High school type: humanities (binary indicator)

HSScience High school type: science (binary indicator)

HSTech High school type: technical (binary indicator)

HSOther High school type: other (binary indicator)

hsgrade High school grade

Y2004 Cohort indicator: year 2004

Y2005 Cohort indicator: year 2005

Y2006 Cohort indicator: year 2006

University University indicator

Humanity Field of study: humanities (binary indicator)

Science Field of study: science (binary indicator)

Social.Science Field of study: social science (binary indicator)

BioMed Field of study: biology and medicine (binary indicator)

Tech Field of study: technical (binary indicator)

Other Field of study: other (binary indicator)

References

Li, F., Mattei, A. and Mealli, F. (2015). Bayesian inference for regression discontinuity designs with application to the evaluation of Italian university grants. *The Annals of Applied Statistics*, 9(4), 1906-1931. doi:10.1214/15AOAS809

`M.step.bin` *Maximization step of the Expectation-Maximization (EM) algorithm to calculate the MLE of the Complier Average Causal Effect (CACE) for binary outcomes*

Description

M.step.bin() implements the Maximization step of the EM algorithm to calculate the MLE of CACE for binary outcomes in the presence of one-sided noncompliance under exclusion restriction for never-takers

Usage

```
M.step.bin(G, dat)
```

Arguments

`G`	A vector with the expected compliance status for each unit in the data set
`dat`	Data frame

Value

A list with the values of the parameters: pi.c = proportion of compliers; py.c0, py.c1 = probabilities that the outcome takes on value 1 for compliers under control and under treatment; py.nt = probability that the outcome takes on value 1 for never-takers

`M.step.bin2` *Maximization step of the Expectation-Maximization (EM) algorithm to calculate the MLE of the Complier Average Causal Effect (CACE) for binary outcomes*

Description

M.step.bin2() implements the Maximization step of the EM algorithm to calculate the MLE of CACE for binary outcomes in the presence of two-sided noncompliance under monotonicity (no defiers) and exclusion restrictions for both never-takers and always-takers

Usage

```
M.step.bin2(G, dat)
```

Arguments

`G`	A vector with the expected compliance status for each unit in the data set
`dat`	Data frame

Value

A list with the values of the parameters: pi.c = proportion of compliers; py.c0, py.c1 = probabilities that the outcome takes on value 1 for compliers under control and under treatment; py.nt = probability that the outcome takes on value 1 for never-takers; py.at = probability that the outcome takes on value 1 for always-takers

---

`M.step.gauss` *Maximization step of the Expectation-Maximization (EM) algorithm to calculate the MLE of the Complier Average Causal Effect (CACE) for Gaussian outcomes*

Description

M.step.gauss() implements the Maximization step of the EM algorithm to calculate the MLE of CACE for continuous normally distributed outcomes in the presence of one-sided noncompliance under exclusion restriction for never-takers

Usage

```
M.step.gauss(G, dat)
```

Arguments

`G`	A vector with the expected compliance status for each unit in the data set
`dat`	Data frame

Value

A list with the values of the parameters: pi.c = proportion of compliers; mu.c0, sigma2.c0, mu.c1, sigma2.c1 = means and variances of the Normal distributions of the outcome for compliers under control and under treatment; mu.nt, sigma2.nt = mean and variance of the Normal distribution of the outcome for never-takers

---

`M.step.gauss2` *Maximization step of the Expectation-Maximization (EM) algorithm to calculate the MLE of the Complier Average Causal Effect (CACE) for Gaussian outcomes*

Description

M.step.gauss2() implements the Maximization step of the EM algorithm to calculate the MLE of CACE for continuous normally distributed outcomes in the presence of two-sided noncompliance under monotonicity (no defiers) and exclusion restrictions for both never-takers and always-takers

Usage

```
M.step.gauss2(G, dat)
```

Arguments

G	A vector with the expected compliance status for each unit in the data set
dat	Data frame

Value

A list with the values of the parameters: pi.c = proportion of compliers; mu.c0, sigma2.c0, mu.c1, sigma2.c1 = means and variances of the Normal distributions of the outcome for compliers under control and under treatment; mu.nt, sigma2.nt = mean and variance of the Normal distribution of the outcome for never-takers; mu.at, sigma2.at = mean and variance of the Normal distribution of the outcome for always-takers

---

mcmc.bin *Posterior median of the Complier Average Causal Effect (CACE) for binary outcomes*

Description

mcmc.bin() uses the Markov Chain Monte Carlo (MCMC) algorithm with the Data Augmentation (DA) method to derive the posterior median of CACE for binary outcomes in the presence of one-sided noncompliance under exclusion restriction for never-takers

Usage

```
mcmc.bin(n.iter, n.burn, dat)
```

Arguments

n.iter	A positive integer specifying the number of iterations (including burn-in)
n.burn	A positive integer specifying the number of burn-in iterations
dat	Data frame

Value

The value of the posterior median of CACE for binary outcomes

`mcmc.bin.h0` *Posterior distributions of the parameter vector and the compliance status of each unit for a binary outcome under the sharp null hypothesis of no effect*

Description

mcmc.bin.h0() uses the Markov Chain Monte Carlo (MCMC) algorithm with the Data Augmentation (DA) method to derive the posterior median of CACE for a binary outcome in the presence of one-sided noncompliance under exclusion restriction for never-takers and under the sharp null hypothesis of no effect

Usage

```
mcmc.bin.h0(n.iter, n.burn, dat)
```

Arguments

`n.iter`	A positive integer specifying the number of iterations (including burn-in)
`n.burn`	A positive integer specifying the number of burn-in iterations
`dat`	Data frame

Value

A list of two elements: THETA = a matrix with the simulated posterior distribution of the parameter vector for a binary outcome under the sharp null hypothesis of no effect Gstatus = a matrix with simulated posterior distribution of the compliance status of each unit under the sharp null hypothesis of no effect

`mcmc.bin.h02` *Posterior distributions of the parameter vector and the compliance status of each unit for a binary outcome under the sharp null hypothesis of no effect*

Description

mcmc.bin.h02() uses the Markov Chain Monte Carlo (MCMC) algorithm with the Data Augmentation (DA) method to derive the posterior median of CACE for a binary outcome in the presence of two-sided noncompliance under monotonicity (no defiers) and exclusion restrictions for both never-takers and always-takers and under the sharp null hypothesis of no effect

Usage

```
mcmc.bin.h02(n.iter, n.burn, dat)
```

Arguments

n.iter	A positive integer specifying the number of iterations (including burn-in)
n.burn	A positive integer specifying the number of burn-in iterations
dat	Data frame

Value

A list of two elements: THETA = a matrix with the simulated posterior distribution of the parameter vector for a binary outcome under the sharp null hypothesis of no effect Gstatus = a matrix with simulated posterior distribution of the compliance status of each unit under the sharp null hypothesis of no effect

---

mcmc.bin2 *Posterior median of the Complier Average Causal Effect (CACE) for binary outcomes*

Description

mcmc.bin2() uses the Markov Chain Monte Carlo (MCMC) algorithm with the Data Augmentation (DA) method to derive the posterior median of CACE for binary outcomes in the presence of two-sided noncompliance under monotonicity (no defiers) and exclusion restrictions for both never-takers and always-takers

Usage

```
mcmc.bin2(n.iter, n.burn, dat)
```

Arguments

n.iter	A positive integer specifying the number of iterations (including burn-in)
n.burn	A positive integer specifying the number of burn-in iterations
dat	Data frame

Value

The value of the posterior median of CACE for binary outcomes

mcmc.gauss — *Posterior median of the Complier Average Causal Effect (CACE) for Gaussian outcomes*

Description

mcmc.gauss() uses the Markov Chain Monte Carlo (MCMC) algorithm with the Data Augmentation (DA) method to derive the posterior median of CACE for normally distributed outcomes in the presence of one-sided noncompliance under exclusion restriction for never-takers

Usage

```
mcmc.gauss(n.iter, n.burn, dat)
```

Arguments

n.iter	A positive integer specifying the number of iterations (including burn-in)
n.burn	A positive integer specifying the number of burn-in iterations
dat	Data frame

Value

The value of the posterior median of CACE for Gaussian outcomes

mcmc.gauss.h0 — *Posterior distributions of the parameter vector and the compliance status of each unit for a Gaussian outcome under the sharp null hypothesis of no effect*

Description

mcmc.gauss.h0() uses the Markov Chain Monte Carlo (MCMC) algorithm with the Data Augmentation (DA) method to derive the posterior median of CACE for a normally distributed outcome in the presence of one-sided noncompliance under exclusion restriction for never-takers and under the sharp null hypothesis of no effect

Usage

```
mcmc.gauss.h0(n.iter, n.burn, dat)
```

Arguments

n.iter	A positive integer specifying the number of iterations (including burn-in)
n.burn	A positive integer specifying the number of burn-in iterations
dat	Data frame

Value

A list of two elements: THETA = a matrix with the simulated posterior distribution of the parameter vector for a Gaussian outcome under the sharp null hypothesis of no effect Gstatus = a matrix with simulated posterior distribution of the compliance status of each unit under the sharp null hypothesis of no effect

---

`mcmc.gauss.h02` *Posterior distributions of the parameter vector and the compliance status of each unit for a Gaussian outcome under the sharp null hypothesis of no effect*

Description

mcmc.gauss.h02() uses the Markov Chain Monte Carlo (MCMC) algorithm with the Data Augmentation (DA) method to derive the posterior median of CACE for a normally distributed outcome in the presence of two-sided noncompliance under monotonicity (no defiers) and exclusion restrictions for both never-takers and always-takers and under the sharp null hypothesis of no effect

Usage

```
mcmc.gauss.h02(n.iter, n.burn, dat)
```

Arguments

`n.iter`	A positive integer specifying the number of iterations (including burn-in)
`n.burn`	A positive integer specifying the number of burn-in iterations
`dat`	Data frame

Value

A list of two elements: THETA = a matrix with the simulated posterior distribution of the parameter vector for a Gaussian outcome under the sharp null hypothesis of no effect Gstatus = a matrix with simulated posterior distribution of the compliance status of each unit under the sharp null hypothesis of no effect

---

`mcmc.gauss2` *Posterior median of the Complier Average Causal Effect (CACE) for Gaussian outcomes*

Description

mcmc.gauss2() uses the Markov Chain Monte Carlo (MCMC) algorithm with the Data Augmentation (DA) method to derive the posterior median of CACE for normally distributed outcomes in the presence of two-sided noncompliance under monotonicity (no defiers) and exclusion restrictions for both never-takers and always-takers

Usage

```
mcmc.gauss2(n.iter, n.burn, dat)
```

Arguments

n.iter	A positive integer specifying the number of iterations (including burn-in)
n.burn	A positive integer specifying the number of burn-in iterations
dat	Data frame

Value

The value of the posterior median of CACE for Gaussian outcomes

---

open_LRErdd_framework *Example of 'LRErdd' in shiny version*

Description

With this function a shiny app will be opened with all the functionalities of 'LRErdd'

Usage

```
open_LRErdd_framework()
```

Value

A shiny app will be opened

Examples

```
if (interactive()) {
  open_LRErdd_framework()
}
```

rand_pajd *Randomization-based tests adjusted for multiple testing*

Description

rand_pajd() calculates, for each of the selected variables and in the selected subpopulation the mean difference and adjusted p-value.

Usage

```
rand_pajd(
  dataset,
  forcing_var_name = "S",
  covariates = c("sex", "HSHumanity", "HSTech", "HSOther", "hsgrade", "Y2004"),
  niter = 1000,
  bandwidth = 1000,
  cut_value = 15000,
  whichunder = 1
)
```

Arguments

dataset	The data frame with the variables
forcing_var_name	The name of the forcing variable in the dataset.
covariates	Covariates of the model.
niter	Number of iteration to calculate the adjusted p-value.
bandwidth	Selected range to subset the dataset and create the subpopulation of interest.
cut_value	Threshold value to separate the treatment status.
whichunder	If the treatment status is lower (0) or higher (1) than the threshold

Details

The table shows the mean difference of each variable between treatment groups, and the corresponding p-value derived using randomization-based tests adjusted for multiple testing.

Value

A data frame with the selected variables and the correponding mean difference with adjusted p-value.

Examples

```
data(grants)
rand_pajd(dataset = grants, forcing_var_name = 'S',
covariates = c('sex', 'HSHumanity', 'HSTech', 'HSOther',
'hsgrade', 'Y2004'), niter = 1000, bandwidth = 1000,
cut_value = 15000, whichunder = 1)
```

`rand_pajd_bw` *Bandwidth selection for Randomization-based tests adjusted for multiple testing*

Description

Randomization-based tests adjusted for multiple testing (rand_pajd function) was defined to calculate just for one bandwidth. For the cases where the user wants to test several bandwidths this functions is applied.

Usage

```
rand_pajd_bw(
  dataset,
  forcing_var_name = "S",
  covariates = c("sex", "HSHumanity", "HSTech", "HSOther", "hsgrade", "Y2004"),
  niter = 1000,
  bandwidth = c(500, 1000, 5000),
  cut_value = 15000,
  whichunder = 1
)
```

Arguments

`dataset`	The data frame with the variables
`forcing_var_name`	The name of the forcing variable in the dataset.
`covariates`	Covariates of the model.
`niter`	Number of iteration to calculate the adjusted p-value.
`bandwidth`	A numerical vector defining the ranges to subset the dataset and create the subpopulations of interest.
`cut_value`	Threshold value to separate the treatment status.
`whichunder`	If the treatment status is lower (0) or higher (1) than the threshold

Value

A data frame with the selected variables and bandwidths, and the correponding mean difference with adjusted p-value.

Examples

```
data(grants)
rand_pajd_bw(dataset = grants, forcing_var_name = 'S',
covariates = c('sex', 'HSHumanity', 'HSTech', 'HSOther',
'hsgrade', 'Y2004'), niter = 1000,
bandwidth = c(500, 1000, 5000), cut_value = 15000,
whichunder = 1)
```

`RegressionDiscontinuityClass` *Class providing object with RegressionDiscontinuityClass*

Description

Class providing object with RegressionDiscontinuityClass

Class providing object with RegressionDiscontinuityClass

Format

`R6::R6Class`

Value

Object of `R6::R6Class` with methods for Regression discontinuity analysis

Methods

`new(RegressionDiscontinuityClass)` This method is used to create object of this class with the next variables.

`initialize(data, forcing, threshold, Z1S1, covariates)` This method initialize the object with the corresponding information.

`full_print()` This method print a summary of the object.

`summary_statistics_mean()` This method a summary of the variables based on the mean.

`bandwidth_selection(typerange = "percentage", range_value = 50, num_it = 50, plot = TRUE)` This methods defines the type of bandwidth that will be considered for the analysis.

`distribution_plot(typerange = "percentage", range_value = 25, num_it = 50, covariate, typecov)` This method plots the selected variables and the bandwidths.

`summary_bandwidth(buffers = c(100, 500, 1000), num_it = 50)` This method create a table with the summary of the selected variables.

`causal_effect(method = 'Sharp FEP', buffers = c(100, 500, 1000), num_it = 50, outcome = "dropout", typeoutc` This method is the main function of the package, implementing the four modes of inference on causal effects.

Public fields

`data` data frame with needed variables

`forcing` Forcing variables name in data

`threshold` The value of the cutoff

`Z1S1` lower or upper will be the corresponding to the group that will have 1 value

`covariates` Covariates that will be used

`A` Name of the variable indicating eligibility status

`W` Name of the treatment received variable

Methods

Public methods:

- `RegressionDiscontinuityClass$new()`
- `RegressionDiscontinuityClass$full_print()`
- `RegressionDiscontinuityClass$summary_statistics_mean()`
- `RegressionDiscontinuityClass$bandwidth_selection()`
- `RegressionDiscontinuityClass$distribution_plot()`
- `RegressionDiscontinuityClass$summary_bandwidth()`
- `RegressionDiscontinuityClass$causal_effect()`
- `RegressionDiscontinuityClass$clone()`

Method `new()`:

Usage:

```
RegressionDiscontinuityClass$new(data, forcing, threshold, Z1S1, covariates)
```

Method `full_print()`:

Usage:

```
RegressionDiscontinuityClass$full_print()
```

Method `summary_statistics_mean()`:

Usage:

```
RegressionDiscontinuityClass$summary_statistics_mean()
```

Method `bandwidth_selection()`:

Usage:

```
RegressionDiscontinuityClass$bandwidth_selection(
  typerange = "percentage",
  range_value = 50,
  num_it = 50,
  plot = TRUE
)
```

Method `distribution_plot()`:

Usage:

```
RegressionDiscontinuityClass$distribution_plot(
  typerange = "percentage",
  range_value = 25,
  num_it = 50,
  covariate,
  typecov
)
```

Method `summary_bandwidth()`**:**

Usage:

```
RegressionDiscontinuityClass$summary_bandwidth(
  buffers = c(100, 500, 1000),
  num_it = 50
)
```

Method `causal_effect()`**:**

Usage:

```
RegressionDiscontinuityClass$causal_effect(
  method = "Sharp FEP",
  buffers = c(100, 500, 1000),
  num_it = 50,
  outcome = "dropout",
  typeoutcome,
  sided = "onesided",
  cin = 0.05,
  treatm_cov = self$W,
  num_it_fuzzy = 5,
  plot = TRUE
)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
RegressionDiscontinuityClass$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Examples

```
library (LRErdd)
library (R6)
library(ggplot2)

data(grants)

# Selection of the covariates of interest
cov <- c("HSTech", "hsgrade","Y2005")
# Create a RegressionDiscontinuityClass with the required information
rdo <- RegressionDiscontinuityClass$new( data = grants , forcing = "S",
```

```
                                        threshold = 15000 , Z1S1 = "lower",
                                        covariates = cov)

# New object structure
rdo
# Summary of the class
rdo$full_print()
# Summary of statistics
rdo$summary_statistics_mean()


rdo$bandwidth_selection(typerange="percentage",range_value=25,
                        num_it=50,plot=TRUE)

#Distributionpreandpostmatching
rdo$distribution_plot(typerange="percentage",range_value=25,
                      num_it=50,covariate="sex",typecov="binary")

# Summary of bandwidths
rdo$summary_bandwidth(buffers=c(100,500,1000),num_it=1000)


# Causal effect with the four methods :
# sharp FEP , sharp Neyman , Fuzzy Neyman and Fuzzy FEP
rdo$causal_effect(method='Sharp FEP',buffers=c(100,500,1000),
                  num_it=50,outcome="dropout",typeoutcome="binary",plot=TRUE)

rdo$causal_effect(method='Sharp Neyman',buffers=c(100,500,1000),
                  num_it=50,outcome="dropout",typeoutcome="binary",
                  sided="onesided",cin=0.05,plot=TRUE)

rdo$causal_effect(method='Fuzzy Neyman',buffers=c(100,500,1000),
                  num_it=50,outcome="dropout",typeoutcome="binary",
                  treatm_cov="W",plot=TRUE)

rdo$causal_effect(method='Fuzzy FEP',buffers=c(100,500,1000),
                  num_it=50,outcome="dropout",typeoutcome="binary",
                  sided="twosided",treatm_cov="W",num_it_fuzzy=5,
                  plot=TRUE)
```

---

sharp_fep *Calculate Fisher Exact p-value*

Description

sharp_fep implements the Fisher exact p?\210?value approach (Fisher 1925) for drawing inference on causal effects in the dataset

Usage

```
sharp_fep(dataset, forcing_bin_var_name = "Z", Y_name = "dropout", niter = 100)
```

Arguments

dataset	The data frame with the variables
forcing_bin_var_name	The name of the forcing variable in a binary format in the dataset.
Y_name	A character format of the name of the outcome in the dataset
niter	Number of iteration to calculate the adjusted p-value.

Details

This approach is randomization-based mode of inference because it assumes randomized experiments: it treats the potential outcome as fixed unknown quantities, viewing the assignment variable, Si, and thus, treatment assignment, Zi, as the only source of randomness. Randomization-based modes of inference, not relying on asymptotic approximations, are particularly attractive in RD settings where the analysis may depend on small samplesize. We implement the Fisher exactp?\210?value approach focusing on assessing the sharp (exact) null hypothesis of no effect of the treatment whatsoever, that is, the null hypothesis under which,for each unit in theUs0, both values of the potential outcomes are identical.

Value

A data frame the correponding effect of the selection.

Examples

```
data(grants)
sharp_fep(dataset = grants, forcing_bin_var_name = 'Z',
Y_name = 'dropout', niter = 100)
```

---

sharp_fep_bw *Bandwidth selection for Fisher Exact p-value*

Description

shar_fep() was defined to calculate just for one dataset. For the cases where the user wants to test several bandwidths this functions is applied.

Usage

```
sharp_fep_bw(
  dataset,
  forcing_var_name = "S",
  Y_name = "dropout",
  niter = 1000,
  bandwidth = c(500, 1000, 1500),
  cut_value = 15000,
  whichunder = 1
)
```

Arguments

dataset	The data frame with the variables
forcing_var_name	The name of the forcing variable in the dataset.
Y_name	A character format of the name of the outcome in the dataset
niter	Number of iteration to calculate the adjusted p-value.
bandwidth	A numerical vector defining the ranges to subset the dataset and create the sub-populations of interest.
cut_value	Threshold value to separate the treatment status.
whichunder	If the treatment status is lower (0) or higher (1) than the threshold

Value

A data frame with the selected bandwidths and the correponding effect.

---

sharp_neyman *SHARP RDD: RANDOMIZATION-BASED INFERENCE - NEYMAN APPROACH*

Description

sharp_neyman() implements the Neyman approach (Neyman 1923, 1990) for drawing inference on causal effects in the dataset

Usage

```
sharp_neyman(Y, Z, cin)
```

Arguments

Y	A numerical vector with the data about the outcome
Z	forcing binary variable
cin	Alpha level for the confidence intervals (e.g. the default 0.05 yields 95% confidence intervals)

Details

When the Neyman approach is used for inference, the function calculate the estimate of the average causal effect in this Equation (1) derived using the estimator proposed by Neyman,that is, the difference between the observed averages for units exposed to treatment and units exposed to control.

Value

A vector with values for the effect and the confidence intervals.

`sharp_neyman_bw` *Bandwidth selection for Sharp - neyman approach*

Description

sharp_neyman() was defined to calculate just for one dataset. For the cases where the user wants to test several bandwidths this functions is applied.

Usage

```
sharp_neyman_bw(
  dataset,
  forcing_var_name = "S",
  Y_name = "dropout",
  niter = 1000,
  bandwidth = c(500, 1000, 1500),
  cut_value = 15000,
  whichunder = 1,
  cin = 0.05
)
```

Arguments

`dataset`	The dataset with the variables
`forcing_var_name`	forcing_var_name
`Y_name`	A character format of the name of the outcome in the dataset
`niter`	Number of iteration to calculate the adjusted p-value.
`bandwidth`	A numerical vector defining the ranges to subset the dataset and create the subpopulations of interest.
`cut_value`	Threshold value to separate the treatment status.
`whichunder`	If the treatment status is lower (0) or higher (1) than the threshold
`cin`	Alpha level for the confidence intervals (e.g. the default 0.05 yields 95% confidence intervals)

Details

We focus on the sharp null hypothesis of no effect of assignment of treatment on outcome: H0:Yi(0) =Yi(1)for alli ?\210\210 Us0

Value

data frame with variable and value and bandwidth

Index