Package 'SpatialRDD'

May 7, 2026

Type Package

Title Conduct Multiple Types of Geographic Regression Discontinuity Designs

Version 0.1.0

Description Spatial versions of Regression Discontinuity Designs (RDDs) are becoming increasingly popular as tools for causal inference. However, conducting state-of-the-art analyses often involves tedious and time-consuming steps. This package offers comprehensive functionalities for executing all required spatial and econometric tasks in a streamlined manner. Moreover, it equips researchers with tools for performing essential placebo and balancing checks comprehensively. The fact that researchers do not have to rely on 'APIs' of external 'GIS' software ensures replicability and raises the standard for spatial RDDs.

Depends R (>= 3.5.0)

License GPL-3

Encoding UTF-8

LazyData true

Imports dplyr, sf, ggplot2, rdrobust, lmtest, sandwich, cowplot, magrittr, rlang, broom

RoxygenNote 7.2.3

Suggests knitr, tmap, rmarkdown, testthat, utils, kableExtra, lfe, stargazer

VignetteBuilder knitr

URL https://axlehner.github.io/SpatialRDD/

BugReports https://github.com/axlehner/SpatialRDD/issues

NeedsCompilation no

Author Alexander Lehner [aut, cre] (ORCID: <https://orcid.org/0000-0001-5885-5966>)

Maintainer Alexander Lehner <lehner@uchicago.edu>

Repository CRAN

Date/Publication 2023-08-08 15:30:05 UTC

Contents

`assign_treated` *Let the package know which observations were treated*

Description

Creates a vector with 0's and 1's to determine on which side of the cut-off each observation is. For this it is useful to have a polygon that fully describes the "treated area". If you do not have such a polygon there is a (preliminary and patchy) way implemented in the package via `points2line` and `cutoff2polygon` that lets you go from points to line to "treated polygon" in a very crude way.

Usage

```
assign_treated(data, polygon, id = NA)
```

Arguments

`data`	sf data frame containing point data (if you have polygons, convert first with sf::st_centroid())
`polygon`	sf object with polygon geometry that fully describes the area(s) that contain the treated points
`id`	string that represents the name of the column in the data that represents the unique identifier for each observation

Value

A vector of type factor with 0's and 1's. Convert with as.numeric() if you want real numbers/integers.

Note

This is essentially a wrapper of `sf::st_intersection`.

Examples

```
points_samp.sf <- sf::st_sample(polygon_full, 100) # create points
# make it an sf object bc st_sample only created the geometry list-column (sfc):
points_samp.sf <- sf::st_sf(points_samp.sf)
# add a unique ID to each observation:
points_samp.sf$id <- 1:nrow(points_samp.sf)
points_samp.sf$treated <- assign_treated(points_samp.sf, polygon_treated, id = "id")
```

`border_segment` *Border Segment Creation for FE-estimation*

Description

Creates `n` segments of a line (the RD cut-off) and assigns the closest border segment for each observation in the sf data frame. Computationally these tasks are quite demanding when the sample size is big and thus might take a few seconds to complete.

Usage

```
border_segment(data, cutoff, n = 10)
```

Arguments

`data`	sf data frame containing point data
`cutoff`	the RDD border in the form of a line (preferred) or borderpoints
`n`	the number of segments to be produced

Value

a vector with factors, each category representing one segment

Examples

```
points_samp.sf <- sf::st_sample(polygon_full, 100) # create points
# make it an sf object bc st_sample only created the geometry list-column (sfc):
points_samp.sf <- sf::st_sf(points_samp.sf)
points_samp.sf$segment10 <- border_segment(points_samp.sf, cut_off, 3)
```

create_placebos *Multiple placebocheks unified in just one list or coefplot*

Description

Unifies shift_border, cutoff2polygon, assign_treated in one function to carry out a myriad of placebo checks at once. The output is either a data.frame (with or without geometry of the respective placeboline) or a coefplot. Requires operations data.frame that contains all desired operations (columns shift.x, shift.y, scale, angle, orientation.1, orientation.2, endpoint.1, endpoint.2), if you don't need a certain operation just use default values (e.g. 0 for angle and 1 for scale), but the column has to be there.

Usage

```
create_placebos(
  data,
  cutoff,
  formula,
  operations,
  bw_dist,
  coefplot = FALSE,
  geometry = FALSE
)
```

Arguments

data	sf data.frame that contains all units of observation
cutoff	initial RD cutoff as an sj object
formula	provide the formula you want to use for OLS, omit the treatetment dummy (if you want a univariate regression just on "treated", then provide y ~ 1 as formula)
operations	container that has all the information in it on how to change the border for each placeboregression
bw_dist	what is the distance for the bandwith (in CRS units, thus ideally metres)
coefplot	provide coefplot instead of a data.frame
geometry	set to TRUE if you want to plot all the lines of the used placebo borders

Value

either a coefplot or data.frame containing results of placebo regressions

Examples

```
points_samp.sf <- sf::st_sample(polygon_full, 100) # create points
# make it an sf object bc st_sample only created the geometry list-column (sfc):
points_samp.sf <- sf::st_sf(points_samp.sf)
# add a unique ID to each observation:
```

```
points_samp.sf$id <- 1:nrow(points_samp.sf)
points_samp.sf$treated <- assign_treated(points_samp.sf, polygon_treated, id = "id")
operations.df <- data.frame(operation = c("shift"),
                            shift.x = c(0),
                            shift.y = c(0),
                            scale = 1,
                            angle = 0,
                            orientation.1 = c("west"),
                            orientation.2 = c("west"),
                            endpoint.1 = c(.8),
                            endpoint.2 = c(.2))
create_placebos(data = points_samp.sf, cutoff = cut_off,
formula = id ~ 1, operations = operations.df, bw_dist = 3000)
```

`cutoff2polygon` *Create (treated) polygon from line*

Description

Creates an approximation of a "treated/untreated polygon" to assign the status again to each observation after the border has been shifted. The function extends both ends of the provided cutoff to the edge of the (imaginary) bounding box of the provided data (this ensures all observations will be included). Key is that you provide a 2-tuple that indicates in which side of the bounding box each end should go (1st element is the one with lower x-coordinate, i.e. leftern most). Always check the output manually by plotting the polygon (e.g. with `tm_shape(your.polygon) + tm_polygons()`). If the output polygon looks odd, a first check should be to just switch the elements from the orientation vector around! See `vignette(shifting_borders)` for details and illustrative examples.

Usage

```
cutoff2polygon(data, cutoff, orientation = NA, endpoints = c(0, 0))
```

Arguments

`data`	study dataset to determine the bounding box (so that all observations are covered by the new polygons) in sf format
`cutoff`	sf object of the (placebo) cut-off
`orientation`	in which side of the bounding box does each of the extensions of the cutoff go into? First element refers to endpoint of border with smaller x-coordinate ("westernmost") (takes two of "north", "east", "south", "west" in a vector, e.g. `c("west", "north")`)
`endpoints`	at what position on the edge should each polygon end? (vector with two numbers between 0 and 1, where 0.5 e.g. means right in the middle of the respective edge)

Value

a polygon as an sf object

Examples

```
points_samp.sf <- sf::st_sample(polygon_full, 100) # create points
# make it an sf object bc st_sample only created the geometry list-column (sfc):
points_samp.sf <- sf::st_sf(points_samp.sf)
# add a unique ID to each observation:
points_samp.sf$id <- 1:nrow(points_samp.sf)
cutoff2polygon(data = points_samp.sf, cutoff = cut_off,
orientation = c("west", "west"), endpoints = c(.8, .2))
```

---

cut_off *Dataset with boundaries and polygons for the SpatialRDD vignette.*

Description

sf multilinestring representing a spatial RD cut-off

Usage

```
data(cut_off)
```

Format

A spatial data.frame of class sf

Source

Lehner, Alexander (2023) Culture, Institutions, and the Roots of Gender Inequality: 450 Years of Portuguese Colonialism in India

---

discretise_border *Split the RD cut-off into borderpoints*

Description

Takes in a border in the form of a polyline (or borderpoints) and converts it into point data. These points are later used to run separate non-parametric RD estimations which eventually allows to visualise potential heterogeneous treatment effects alongside the cut-off.

Usage

```
discretise_border(
  cutoff,
  n = 10,
  random = FALSE,
  range = FALSE,
  ymax = NA,
  ymin = NA,
  xmax = NA,
  xmin = NA
)
```

Arguments

cutoff	sf object of the RD cut-off in the form of a line (not preferred, but also boundarypoints are possible)
n	the number of borderpoints to be created
random	whether they are randomly chosen (not desireable in most cases)
range	default = FALSE, if there is a specific range (N-S or E-W) for which the points are to be drawn (useful in order to exclude sparse borderpoints with little/no oberservations around because the non-parametric RD estimation will fail)
ymax	if range = TRUE: y coordinates
ymin	if range = TRUE: y coordinates
xmax	if range = TRUE: x coordinates
xmin	if range = TRUE: x coordinates

Value

an sf object with selected (and evenly spaced) borderpoints

Examples

```
borderpoints <- discretise_border(cutoff = cut_off, n = 10)
```

---

plotspatialrd *Plot SpatialRD output*

Description

Produces plot of GRDDseries and optionally of a map that visualises every point estimate in space.

Usage

```
plotspatialrd(SpatialRDoutput, map = FALSE)
```

Arguments

SpatialRDoutput	spatial object that is produced by an estimation with spatialrd
map	TRUE/FALSE depending on whether mapplot is desired (make sure to set spatial.objcet = TRUE in the spatialrd function)

Value

plots produced with ggplot2

Examples

```
points_samp.sf <- sf::st_sample(polygon_full, 1000) # create points
# make it an sf object bc st_sample only created the geometry list-column (sfc):
points_samp.sf <- sf::st_sf(points_samp.sf)
# add a unique ID to each observation:
points_samp.sf$id <- 1:nrow(points_samp.sf)
# assign treatment:
points_samp.sf$treated <- assign_treated(points_samp.sf, polygon_treated, id = "id")
# first we define a variable for the number of "treated" and control
NTr <- length(points_samp.sf$id[points_samp.sf$treated == 1])
NCo <- length(points_samp.sf$id[points_samp.sf$treated == 0])
# the treated areas get a 10 percentage point higher literacy rate
points_samp.sf$education[points_samp.sf$treated == 1] <- 0.7
points_samp.sf$education[points_samp.sf$treated == 0] <- 0.6
# and we add some noise, otherwise we would obtain regression coeffictions with no standard errors
points_samp.sf$education[points_samp.sf$treated == 1] <- rnorm(NTr, mean = 0, sd = .1) +
  points_samp.sf$education[points_samp.sf$treated == 1]
points_samp.sf$education[points_samp.sf$treated == 0] <- rnorm(NCo, mean = 0, sd = .1) +
  points_samp.sf$education[points_samp.sf$treated == 0]

# create distance to cutoff
points_samp.sf$dist2cutoff <- as.numeric(sf::st_distance(points_samp.sf, cut_off))

points_samp.sf$distrunning <- points_samp.sf$dist2cutoff
# give the non-treated one's a negative score
points_samp.sf$distrunning[points_samp.sf$treated == 0] <- -1 *
 points_samp.sf$distrunning[points_samp.sf$treated == 0]

# create borderpoints
borderpoints.sf <- discretise_border(cutoff = cut_off, n = 10)
borderpoints.sf$id <- 1:nrow(borderpoints.sf)

# finally, carry out estimation alongside the boundary:
results <- spatialrd(y = "education", data = points_samp.sf, cutoff.points = borderpoints.sf,
treated = "treated", minobs = 20, spatial.object = FALSE)


plotspatialrd(results)
```

points2line *Convert borderpoints to a line*

Description

Small function that connects dots and makes them one line which can later be used as a cutoff for the RD.

Usage

```
points2line(borderpoints, crs)
```

Arguments

borderpoints	a set of points on a boundary
crs	set the coordinate reference system (CRS)

Value

a line as an sf object

Examples

```
points_samp.sf <- sf::st_sample(polygon_full, 2) # create points
# make it an sf object bc st_sample only created the geometry list-column (sfc):
points_samp.sf <- sf::st_sf(points_samp.sf)
points2line(points_samp.sf, crs = sf::st_crs(points_samp.sf))
```

polygon_full *Dataset with boundaries and polygons for the SpatialRDD vignette.*

Description

sf multipolygon

Usage

```
data(polygon_full)
```

Format

A spatial data.frame of class sf

Source

Lehner, Alexander (2023) Culture, Institutions, and the Roots of Gender Inequality: 450 Years of Portuguese Colonialism in India

polygon_treated *Dataset with boundaries and polygons for the SpatialRDD vignette.*

Description

sf multipolygon

Usage

```
data(polygon_treated)
```

Format

A spatial data.frame of class sf

Source

Lehner, Alexander (2023) Culture, Institutions, and the Roots of Gender Inequality: 450 Years of Portuguese Colonialism in India

printspatialrd *Print spatialrd output*

Description

Preliminary function, styling with e.g. kable and kableExtra has to be done by the user individually. You could also just use the package of your choice to print out columns of the output from `spatialrd`.

Usage

```
printspatialrd(SpatialRDoutput)
```

Arguments

`SpatialRDoutput`	output file from the `spatialrd` function

Value

A table with results from the `spatialrd` function

Examples

```
points_samp.sf <- sf::st_sample(polygon_full, 1000) # create points
# make it an sf object bc st_sample only created the geometry list-column (sfc):
points_samp.sf <- sf::st_sf(points_samp.sf)
# add a unique ID to each observation:
points_samp.sf$id <- 1:nrow(points_samp.sf)
# assign treatment:
points_samp.sf$treated <- assign_treated(points_samp.sf, polygon_treated, id = "id")
# first we define a variable for the number of "treated" and control
NTr <- length(points_samp.sf$id[points_samp.sf$treated == 1])
NCo <- length(points_samp.sf$id[points_samp.sf$treated == 0])
# the treated areas get a 10 percentage point higher literacy rate
points_samp.sf$education[points_samp.sf$treated == 1] <- 0.7
points_samp.sf$education[points_samp.sf$treated == 0] <- 0.6
# and we add some noise, otherwise we would obtain regression coeffictions with no standard errors
points_samp.sf$education[points_samp.sf$treated == 1] <- rnorm(NTr, mean = 0, sd = .1) +
  points_samp.sf$education[points_samp.sf$treated == 1]
points_samp.sf$education[points_samp.sf$treated == 0] <- rnorm(NCo, mean = 0, sd = .1) +
  points_samp.sf$education[points_samp.sf$treated == 0]

# create distance to cutoff
points_samp.sf$dist2cutoff <- as.numeric(sf::st_distance(points_samp.sf, cut_off))

points_samp.sf$distrunning <- points_samp.sf$dist2cutoff
# give the non-treated one's a negative score
points_samp.sf$distrunning[points_samp.sf$treated == 0] <- -1 *
 points_samp.sf$distrunning[points_samp.sf$treated == 0]

# create borderpoints
borderpoints.sf <- discretise_border(cutoff = cut_off, n = 10)
borderpoints.sf$id <- 1:nrow(borderpoints.sf)

# finally, carry out estimation alongside the boundary:
results <- spatialrd(y = "education", data = points_samp.sf, cutoff.points = borderpoints.sf,
treated = "treated", minobs = 20, spatial.object = FALSE)
printspatialrd(results)
```

`shift_border` *Shift, shrink/grow, and rotate borders around*

Description

This functions takes in a border and can either shift, shrink, or rotate it. All of them can be done together as well. This usually takes a bit of trial and error, so make sure to plot the result each time. For a detailed walk through check out the according vignette: `vignette(shifting_borders)`.

Usage

```
shift_border(
  border,
  operation = c("shift", "scale", "rotate"),
  shift = c(0, 0),
  scale = 1,
  angle = 0
)
```

Arguments

border	sf object with line geometry
operation	"shift", "rotate", "scale" - or a combination of them
shift	if operation = "shift", shift distance in CRS units (if UTM it is metres) for x and y coordinates as c(dist_x, dist_y)
scale	if operation = "scale", provide shrinkage/growth factor: e.g. .9 to shrink by 10perc. and 1.1 to increase by 10perc.
angle	if operation = "rotate", provide angle in degrees

Value

a new border in the form of an sf object

Examples

```
shift_border(border = cut_off, operation = c("shift", "scale"),
shift = c(-5000, -3000), scale = .85)

shift_border(border = cut_off, operation = "rotate", angle = 10)
```

---

spatialrd *non-parametric Spatial RD / GRD*

Description

This function loops over all boundary points and locally estimates a non-parametric RD (using local linear regression) using the rdrobust function from the rdrobust package from Calonico, Cattaneo, Titiunik (2014). It takes in the discretized cutoff point file (the RDcutoff, a linestring chopped into parts by the discretise_border function) and the sf object (which essentially is just a conventional data.frame with a geometry() column) containing all the observations (treated and untreated). The treated indicator variable has to be assigned before (potentially with assign_treated) and be part of the sf object as a column.

Usage

```
spatialrd(
  y,
  data,
  cutoff.points,
  treated,
  minobs = 50,
  bwfix_m = NA,
  sample = FALSE,
  samplesize = NA,
  sparse.exclusion = FALSE,
  store.CIs = FALSE,
  spatial.object = TRUE,
  ...
)
```

Arguments

y	The name of the dependent variable in the points frame in the form of a string
data	sf data.frame with points that describe the observations
cutoff.points	sf object of borderpoints (provided by user or obtained with discretise_border)
treated	column that contains the treated dummy (as string)
minobs	the minimum amount of observations in each estimation for the point estimate to be included (default is 50)
bwfix_m	fixed bandwidth in meters (in case you want to impose one yourself)
sample	draw a random sample of points (default is FALSE)
samplesize	if random, how many points
sparse.exclusion	in case we want to try to exclude sparse border points before the estimation (should reduce warnings)
store.CIs	set TRUE of confidence intervals should be stored
spatial.object	return a spatial object (deafult is TRUE, needed if you want to plot the point estimates on a map)?
...	in addition you can use all options in rdrobust

Details

This function nests rdrobust. All its options (aside from running variable x and cutoff c) are available here as well (e.g. bw selection, cluster level, kernel, weights). Check the documentation in the rdrobust package for details. (bandwidth selection default in rdrobust is bwselect = 'mserd')

To visualise the output, use plotspatialrd for a graphical representation. You can use printspatialrd (or an R package of your choice) for a table output. .

Value

a data.frame or spatial data.frame (sf object) in case spatial.object = TRUE (default)

References

Calonico, Cattaneo and Titiunik (2014): Robust Nonparametric Confidence Intervals for Regression-Discontinuity Designs, Econometrica 82(6): 2295-2326.

Examples

```
points_samp.sf <- sf::st_sample(polygon_full, 1000) # create points
# make it an sf object bc st_sample only created the geometry list-column (sfc):
points_samp.sf <- sf::st_sf(points_samp.sf)
# add a unique ID to each observation:
points_samp.sf$id <- 1:nrow(points_samp.sf)
# assign treatment:
points_samp.sf$treated <- assign_treated(points_samp.sf, polygon_treated, id = "id")
# first we define a variable for the number of "treated" and control
NTr <- length(points_samp.sf$id[points_samp.sf$treated == 1])
NCo <- length(points_samp.sf$id[points_samp.sf$treated == 0])
# the treated areas get a 10 percentage point higher literacy rate
points_samp.sf$education[points_samp.sf$treated == 1] <- 0.7
points_samp.sf$education[points_samp.sf$treated == 0] <- 0.6
# and we add some noise, otherwise we would obtain regression coeffictions with no standard errors
points_samp.sf$education[points_samp.sf$treated == 1] <- rnorm(NTr, mean = 0, sd = .1) +
  points_samp.sf$education[points_samp.sf$treated == 1]
points_samp.sf$education[points_samp.sf$treated == 0] <- rnorm(NCo, mean = 0, sd = .1) +
  points_samp.sf$education[points_samp.sf$treated == 0]

# create distance to cutoff
points_samp.sf$dist2cutoff <- as.numeric(sf::st_distance(points_samp.sf, cut_off))

points_samp.sf$distrunning <- points_samp.sf$dist2cutoff
# give the non-treated one's a negative score
points_samp.sf$distrunning[points_samp.sf$treated == 0] <- -1 *
 points_samp.sf$distrunning[points_samp.sf$treated == 0]

# create borderpoints
borderpoints.sf <- discretise_border(cutoff = cut_off, n = 10)
borderpoints.sf$id <- 1:nrow(borderpoints.sf)

# finally, carry out estimation alongside the boundary:
results <- spatialrd(y = "education", data = points_samp.sf, cutoff.points = borderpoints.sf,
treated = "treated", minobs = 20, spatial.object = FALSE)
```

Index