Package: sfdepmap (via r-universe)

November 23, 2026

Title Moran, LISA and Getis-Ord Maps for Teaching Spatial Statistics

Version 0.3.0

Description Functions written for Penn State spatial statistics courses. Each function takes an 'sf' dataset, a column name, a neighbour list and spatial weights, and makes one finished plot: a Moran scatterplot, a LISA (Local Moran's I) map, Getis-Ord Gi and Gi* hot spot maps, or a four-map summary figure. The maps are built with 'tmap' and are sized to fit R Markdown and Quarto reports.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

Depends R (>= 4.1.0)

Imports graphics, grDevices, grid, sf, sfdep, spdep, stats, tmap (>= 4.0), utils

Suggests knitr, testthat (>= 3.0.0)

Config/testthat/edition 3

Config/roxygen2/version 8.1.0

Config/pak/sysreqs libabsl-dev cmake libgdal-dev gdal-bin libgeos-dev make libpng-dev libuv1-dev libxml2-dev libssl-dev libproj-dev libsqlite3-dev libudunits2-dev zlib1g-dev

Repository https://psu-spatial.r-universe.dev

Date/Publication 2026-11-23 08:04:13 UTC

RemoteUrl https://github.com/psu-spatial/sfdepmap

RemoteRef HEAD

RemoteSha 3b59ae9008425e0a7927aefa416a42b08cccb35f

Contents

autocorrelation_summary 2
getisstar_map 3

lisa_map 6
lisa_scatterplot 7
moran_scatterplot 9
print.autocorrelation_map 10

Index **11**

`autocorrelation_summary`

Four maps in one figure: your data, LISA, Gi and Gi*

Description

Makes one figure with four maps and one caption underneath: A. your data as a choropleth map (five groups, Jenks natural breaks), B. LISA clusters, C. Gi* hot and cold spots, D. Gi hot and cold spots. In R Markdown, a chunk with `fig.width=9, fig.height=11` gives it enough room.

Usage

```
autocorrelation_summary(
  data,
  variable,
  label,
  neighbours,
  weights,
  title = variable,
  data_source = NULL,
  mode = "plot",
  basemap = FALSE,
  significance = 0.05,
  nsim = 499,
  seed = NULL,
  explanation = TRUE,
  global_stats = TRUE,
  units = "Value",
  adjust_p = FALSE
)
```

Arguments

`data`	Your sf object, for example `Census.sf`.
`variable`	The NAME of the column you want to study, in "quotes".
`label`	The NAME of a column that labels each row, in "quotes" (for example `"GEOID"`).
`neighbours`	The neighbours you made, for example with `sfdep::st_contiguity()`.
`weights`	The weights you made with `sfdep::st_weights()`.
`title`	A description of your variable, used on the plots.

data_source	A line of text saying where your data came from, shown under the map. Leave it out (or use FALSE) for no line.
mode	"plot" for a static map for a report (the default), or "view" for a map you can zoom and click on.
basemap	FALSE for no background map (the default), TRUE for tmap's standard background map, or the name of a background map such as "OpenStreetMap". Background maps need the internet.
significance	The p-value cut-off. The default is 0.05.
nsim	How many random shuffles are used for the p-values. The default is 499.
seed	The random seed. Leave it out and R picks a new one each time. Give a whole number to get the same map every time.
explanation	FALSE leaves out the lines of the caption that explain maps A to D.
global_stats	FALSE leaves out the global Moran's I, Geary's C and General G.
units	What your numbers are, for example "Percentage". It is used as the title of the legend on map A.
adjust_p	TRUE adjusts the Getis-Ord p-values (maps C and D) for the number of tests (Benjamini-Hochberg). The LISA map is not adjusted.

Value

A figure. Type its name to see it.

Examples

```
nc <- sf::st_read(system.file("shape/nc.shp", package = "sf"), quiet = TRUE)
nc <- sf::st_transform(nc, 32019)   # a flat (projected) map of North Carolina
NeighbourMatrix <- sfdep::st_contiguity(nc)
WeightsMatrix   <- sfdep::st_weights(NeighbourMatrix, style = "W", allow_zero = TRUE)
Summary <- autocorrelation_summary(nc, "BIR74", "NAME", NeighbourMatrix, WeightsMatrix,
                                   title = "Births in 1974", units = "Births", seed = 364)
```

---

getisstar_map *Getis-Ord hot spot maps: Gi* and Gi*

Description

Both give every area a z-score and a p-value. A positive z-score with a small p-value is a hot spot (high values clustered together) and a negative z-score with a small p-value is a cold spot. The p-values come from random shuffles of the data, using `sfdep::local_g_perm()` and `sfdep::local_gstar_perm()`. Areas are put into 7 groups: 90%, 95% and 99% confidence mean p <= 0.10, 0.05 and 0.01.

Usage

```
getisstar_map(
  data,
  variable,
  label,
  neighbours,
  weights,
  title = variable,
  data_source = NULL,
  mode = "plot",
  palette = 1,
  basemap = FALSE,
  nsim = 499,
  seed = NULL,
  adjust_p = FALSE,
  silent = FALSE,
  return_data = FALSE
)

getis_map(
  data,
  variable,
  label,
  neighbours,
  weights,
  title = variable,
  data_source = NULL,
  mode = "plot",
  palette = 1,
  basemap = FALSE,
  nsim = 499,
  seed = NULL,
  adjust_p = FALSE,
  silent = FALSE,
  return_data = FALSE
)
```

Arguments

`data`	Your sf object, for example `Census.sf`.
`variable`	The NAME of the column you want to study, in "quotes".
`label`	The NAME of a column that labels each row, in "quotes" (for example `"GEOID"`).
`neighbours`	The neighbours you made, for example with `sfdep::st_contiguity()`.
`weights`	The weights you made with `sfdep::st_weights()`.
`title`	A description of your variable, used on the plots.
`data_source`	A line of text saying where your data came from, shown under the map. Leave it out (or use `FALSE`) for no line.

mode	"plot" for a static map for a report (the default), or "view" for a map you can zoom and click on.
palette	Which colours to use: a palette number (1 to 5), a palette name ("red_yellow_blue", "brewer_RdYlBu", "brewer_RdYlBu_DarkSig", "brewer_PuOr_DarkSig", "brewer_PuOr"), or 7 colours of your own, from cold spot 99% to hot spot 99%.
basemap	FALSE for no background map (the default), TRUE for tmap's standard background map, or the name of a background map such as "OpenStreetMap". Background maps need the internet.
nsim	How many random shuffles are used for the p-values. The default is 499.
seed	The random seed. Leave it out and R picks a new one each time. Give a whole number to get the same map every time.
adjust_p	TRUE adjusts the p-values for the number of tests (Benjamini-Hochberg). This is stricter, so expect fewer hot and cold spots. The default is FALSE.
silent	TRUE means do not print the table of groups.
return_data	TRUE gives back the results as an sf data object, in place of the map. It holds your label column, the column you studied, the geometry and the new columns (getisstar.z, getisstar.pvalue and getisstar.cluster, or getis.z, getis.pvalue and getis.cluster).

Details

getisstar_map() makes a Gi* ("G-i-star") map: each area is counted TOGETHER WITH its neighbours. getis_map() makes a Gi map: each area's own value is LEFT OUT and only its neighbours are used. The map changes slightly each time unless you give a seed. The seed that was used is always written under the map.

Value

A map. Type its name to see it.

Examples

```
nc <- sf::st_read(system.file("shape/nc.shp", package = "sf"), quiet = TRUE)
nc <- sf::st_transform(nc, 32019)   # a flat (projected) map of North Carolina
NeighbourMatrix <- sfdep::st_contiguity(nc)
WeightsMatrix   <- sfdep::st_weights(NeighbourMatrix, style = "W", allow_zero = TRUE)
GETISSTARmap <- getisstar_map(nc, "BIR74", "NAME", NeighbourMatrix, WeightsMatrix,
                              title = "Births in 1974", seed = 364)
GETISmap     <- getis_map(nc, "BIR74", "NAME", NeighbourMatrix, WeightsMatrix,
                          title = "Births in 1974", seed = 364, adjust_p = TRUE)
```

`lisa_map` *LISA map (Local Moran's I)*

Description

Puts each area into a group (High-High, High-Low, Low-High, Low-Low or not significant), prints a table of how many areas are in each group, and makes a finished map. The p-values come from random shuffles of the data, so the map changes slightly each time unless you give a `seed`. The seed that was used is always written under the map.

Usage

```
lisa_map(
  data,
  variable,
  label,
  neighbours,
  weights,
  title = variable,
  data_source = NULL,
  mode = "plot",
  significance = 0.05,
  nsim = 499,
  global_stats = FALSE,
  palette = 1,
  basemap = FALSE,
  seed = NULL,
  silent = FALSE,
  return_data = FALSE
)
```

Arguments

`data`	Your sf object, for example `Census.sf`.
`variable`	The NAME of the column you want to study, in "quotes".
`label`	The NAME of a column that labels each row, in "quotes" (for example `"GEOID"`).
`neighbours`	The neighbours you made, for example with `sfdep::st_contiguity()`.
`weights`	The weights you made with `sfdep::st_weights()`.
`title`	A description of your variable, used on the plots.
`data_source`	A line of text saying where your data came from, shown under the map. Leave it out (or use `FALSE`) for no line.
`mode`	`"plot"` for a static map for a report (the default), or `"view"` for a map you can zoom and click on.
`significance`	The p-value cut-off. The default is 0.05.

nsim	How many random shuffles are used for the p-values. The default is 499.
global_stats	TRUE adds the global Moran's I, Geary's C and Getis-Ord General G to the notes under the map.
palette	Which colours to use: a palette number (1 to 6), a palette name ("red_blue", "pastel", "bright", "black_gold", "green_blue", "brown_teal"), or 5 colours of your own in the order High-High, High-Low, Low-High, Low-Low, not significant.
basemap	FALSE for no background map (the default), TRUE for tmap's standard background map, or the name of a background map such as "OpenStreetMap". Background maps need the internet.
seed	The random seed. Leave it out and R picks a new one each time. Give a whole number to get the same map every time.
silent	TRUE means do not print the table of groups.
return_data	TRUE gives back the results as an sf data object, in place of the map. It holds your label column, the column you studied, the geometry and the new columns (LocalMoran, LocalMoran_pvalue, LocalMoran_Quadrant, Quadrant_sig).

Value

A map. Type its name to see it. In R Markdown, a chunk with `fig.width=7, fig.height=6` gives it enough room.

Examples

```
nc <- sf::st_read(system.file("shape/nc.shp", package = "sf"), quiet = TRUE)
nc <- sf::st_transform(nc, 32019)   # a flat (projected) map of North Carolina
NeighbourMatrix <- sfdep::st_contiguity(nc)
WeightsMatrix   <- sfdep::st_weights(NeighbourMatrix, style = "W", allow_zero = TRUE)
LISAmap <- lisa_map(nc, "BIR74", "NAME", NeighbourMatrix, WeightsMatrix,
                    title = "Births in 1974", seed = 364)
```

---

lisa_scatterplot *LISA scatterplot*

Description

The same plot as moran_scatterplot(), with every point coloured by its LISA group (High-High, High-Low, Low-High, Low-Low). Points that are not significant at your chosen p-value are grey. The groups and p-values are the ones lisa_map() uses, so with the same significance, nsim and seed the plot matches the map. The seed that was used is always written under the plot.

Usage

```
lisa_scatterplot(
  data,
  variable,
  label,
  neighbours,
  weights,
  title = variable,
  standardise = FALSE,
  annotate = FALSE,
  palette = 1,
  significance = 0.05,
  nsim = 499,
  seed = NULL
)
```

Arguments

`data`	Your sf object, for example `Census.sf`.
`variable`	The NAME of the column you want to study, in "quotes".
`label`	The NAME of a column that labels each row, in "quotes" (for example `"GEOID"`).
`neighbours`	The neighbours you made, for example with `sfdep::st_contiguity()`.
`weights`	The weights you made with `sfdep::st_weights()`.
`title`	A description of your variable, used on the plots.
`standardise`	`TRUE` turns the values into z-scores and makes the plot square.
`annotate`	`TRUE` colours in the four quadrants, names them and labels the two mean lines.
`palette`	The colours for the four LISA groups. These are the same palettes as `lisa_map()`: a palette number (1 to 6), a palette name, or colours of your own in the order High-High, High-Low, Low-High, Low-Low.
`significance`	The p-value cut-off. The default is 0.05.
`nsim`	How many random shuffles are used for the p-values. The default is 499.
`seed`	The random seed. Leave it out and R picks a new one each time.

Details

A point's LISA group can differ from the quadrant it sits in when it is very close to one of the dashed lines, because the LISA calculation measures the neighbours' average against the overall mean and the dashed line shows the mean of the neighbours' averages.

Value

The plot is drawn. The numbers behind it are returned invisibly, with a `lisa_group` column.

Examples

```
nc <- sf::st_read(system.file("shape/nc.shp", package = "sf"), quiet = TRUE)
nc <- sf::st_transform(nc, 32019)   # a flat (projected) map of North Carolina
NeighbourMatrix <- sfdep::st_contiguity(nc)
WeightsMatrix   <- sfdep::st_weights(NeighbourMatrix, style = "W", allow_zero = TRUE)
lisa_scatterplot(nc, "BIR74", "NAME", NeighbourMatrix, WeightsMatrix,
                 title = "Births in 1974", seed = 364)
```

`moran_scatterplot` *Moran's scatterplot*

Description

Plots the value in each area (x axis) against the average value in that area's neighbours, the "spatial lag" (y axis). The most unusual points are labelled using your label column.

Usage

```
moran_scatterplot(
  data,
  variable,
  label,
  neighbours,
  weights,
  title = variable,
  standardise = FALSE,
  annotate = FALSE,
  palette = 1
)
```

Arguments

`data`	Your sf object, for example `Census.sf`.
`variable`	The NAME of the column you want to study, in "quotes".
`label`	The NAME of a column that labels each row, in "quotes" (for example `"GEOID"`).
`neighbours`	The neighbours you made, for example with `sfdep::st_contiguity()`.
`weights`	The weights you made with `sfdep::st_weights()`.
`title`	A description of your variable, used on the plots.
`standardise`	`TRUE` turns the values into z-scores (the number of standard deviations above/below the mean) and makes the plot square. The slope of the line stays the same. The default is `FALSE`.
`annotate`	`TRUE` colours in the four quadrants, names them (High-High, High-Low, Low-High, Low-Low) and labels the two mean lines. The default is `FALSE`.
`palette`	The colours used by `annotate`. These are the same palettes as `lisa_map()`: a palette number (1 to 6), a palette name, or colours of your own in the order High-High, High-Low, Low-High, Low-Low.

Details

Areas with no neighbours, or with a missing value, are left off the plot, and a message says which ones. With row-standardised weights (`style = "W"`), the slope of the line is Moran's I for the areas on the plot.

Value

The plot is drawn. The numbers behind it are returned invisibly: the x value, the spatial lag (`wx`), whether the point was labelled (`is_inf`) and the label.

Examples

```
nc <- sf::st_read(system.file("shape/nc.shp", package = "sf"), quiet = TRUE)
nc <- sf::st_transform(nc, 32019)   # a flat (projected) map of North Carolina
NeighbourMatrix <- sfdep::st_contiguity(nc)
WeightsMatrix   <- sfdep::st_weights(NeighbourMatrix, style = "W", allow_zero = TRUE)
moran_scatterplot(nc, "BIR74", "NAME", NeighbourMatrix, WeightsMatrix,
                  title = "Births in 1974")
moran_scatterplot(nc, "BIR74", "NAME", NeighbourMatrix, WeightsMatrix,
                  title = "Births in 1974", standardise = TRUE, annotate = TRUE)
```

---

print.autocorrelation_map

Show a map made by this package

Description

These run for you when you type the name of a map, or when a map is printed in R Markdown or Quarto. You do not need to call them yourself.

Usage

```
## S3 method for class 'autocorrelation_map'
print(x, ...)

## S3 method for class 'autocorrelation_map'
knit_print(x, ...)
```

Arguments

`x`	A map made by `lisa_map()`, `getisstar_map()`, `getis_map()` or `autocorrelation_summary()`.
`...`	Not used.

Value

The map, invisibly.

Index

autocorrelation_summary, 2
autocorrelation_summary(), *10*

getis_map *(*getisstar_map*)*, 3
getis_map(), *10*
getisstar_map, 3
getisstar_map(), *10*

knit_print.autocorrelation_map
        *(*print.autocorrelation_map*)*, 10

lisa_map, 6
lisa_map(), *7–10*
lisa_scatterplot, 7

moran_scatterplot, 9
moran_scatterplot(), *7*

print.autocorrelation_map, 10