Package 'hgwrr'

July 8, 2024

Type Package

Title Hierarchical and Geographically Weighted Regression

Version 0.4-0

Date 2024-07-04

Author Yigong Hu, Richard Harris, Richard Timmerman

Maintainer Yigong Hu <yigong.hu@bristol.ac.uk>

Description This model divides coefficients into three types,
i.e., local fixed effects, global fixed effects, and random effects (Hu et al., 2022)<doi:10.1177/23998083211063885>.
If data have spatial hierarchical structures (especially are overlapping on some locations),
it is worth trying this model to reach better fitness.

License GPL (>= 2)

URL https://github.com/HPDell/hgwrr/, https://hpdell.github.io/hgwrr/

Imports Rcpp (>= 1.0.8)

LinkingTo Rcpp, RcppArmadillo

Depends R (>= 3.5.0), sf, stats, utils

NeedsCompilation yes

Suggests knitr, rmarkdown, testthat (>= 3.0.0),

SystemRequirements GNU make

RoxygenNote 7.2.3

VignetteBuilder knitr

Config/Needs/website tidyverse, ggplot2, tmap, lme4, spdep, GWmodel

Repository CRAN

Date/Publication 2024-07-08 15:40:02 UTC

Contents

`hgwrr-package` *HGWR: Hierarchical and Geographically Weighted Regression*

Description

An R and C++ implementation of Hierarchical and Geographically Weighted Regression (HGWR) model is provided in this package. This model divides coefficients into three types: local fixed effects, global fixed effects, and random effects. If data have spatial hierarchical structures (especially are overlapping on some locations), it is worth trying this model to reach better fitness.

Details

Package:	hgwrr
Type:	Package
Title:	Hierarchical and Geographically Weighted Regression
Version:	0.4-0
Date:	2024-07-04
Author:	Yigong Hu, Richard Harris, Richard Timmerman
Maintainer:	Yigong Hu <yigong.hu@bristol.ac.uk>
Description:	This model divides coefficients into three types, i.e., local fixed effects, global fixed effects, and rand
License:	GPL (>= 2)
URL:	https://github.com/HPDell/hgwrr/, https://hpdell.github.io/hgwrr/
Imports:	Rcpp (>= 1.0.8)
LinkingTo:	Rcpp, RcppArmadillo
Depends:	R (>= 3.5.0), sf, stats, utils
NeedsCompilation:	yes
Suggests:	knitr, rmarkdown, testthat (>= 3.0.0),
SystemRequirements:	GNU make
Roxygen:	list(markdown = TRUE)
RoxygenNote:	7.2.3
VignetteBuilder:	knitr
Config/Needs/website:	tidyverse, ggplot2, tmap, lme4, spdep, GWmodel

Note

Acknowledgement: We gratefully acknowledge support from China Scholarship Council.

Author(s)

Yigong Hu, Richard Harris, Richard Timmerman

References

Hu, Y., Lu, B., Ge, Y., Dong, G., 2022. Uncovering spatial heterogeneity in real estate prices via combined hierarchical linear model and geographically weighted regression. Environment and Planning B: Urban Analytics and City Science. doi:10.1177/23998083211063885

`coef.hgwrm` *Get estimated coefficients.*

Description

Get estimated coefficients.

Usage

```
## S3 method for class 'hgwrm'
coef(object, ...)
```

Arguments

`object`	An `hgwrm` object returned by `hgwr()`.
`...`	Parameter received from other functions.

Value

A `DataFrame` object consists of all estimated coefficients.

See Also

`hgwr()`, `summary.hgwrm()`, `fitted.hgwrm()` and `residuals.hgwrm()`.

`fitted.hgwrm` *Get fitted response.*

Description

Get fitted response.

Usage

```
## S3 method for class 'hgwrm'
fitted(object, ...)
```

Arguments

`object`	An `hgwrm` object returned by `hgwr()`.
`...`	Parameter received from other functions.

Value

A vector consists of fitted response values.

See Also

`hgwr()`, `summary.hgwrm()`, `coef.hgwrm()` and `residuals.hgwrm()`.

`hgwr` *Hierarchical and Geographically Weighted Regression*

Description

A Hierarchical Linear Model (HLM) with local fixed effects.

Usage

```
hgwr(
  formula,
  data,
  ...,
  bw = "CV",
  kernel = c("gaussian", "bisquared"),
  alpha = 0.01,
  eps_iter = 1e-06,
  eps_gradient = 1e-06,
  max_iters = 1e+06,
  max_retries = 1e+06,
```

```
  ml_type = c("D_Only", "D_Beta"),
  verbose = 0
)

## S3 method for class 'sf'
hgwr(
  formula,
  data,
  ...,
  bw = "CV",
  kernel = c("gaussian", "bisquared"),
  alpha = 0.01,
  eps_iter = 1e-06,
  eps_gradient = 1e-06,
  max_iters = 1e+06,
  max_retries = 1e+06,
  ml_type = c("D_Only", "D_Beta"),
  verbose = 0
)

## S3 method for class 'data.frame'
hgwr(
  formula,
  data,
  ...,
  coords,
  bw = "CV",
  kernel = c("gaussian", "bisquared"),
  alpha = 0.01,
  eps_iter = 1e-06,
  eps_gradient = 1e-06,
  max_iters = 1e+06,
  max_retries = 1e+06,
  ml_type = c("D_Only", "D_Beta"),
  verbose = 0
)

hgwr_fit(
  formula,
  data,
  coords,
  bw = "CV",
  kernel = c("gaussian", "bisquared"),
  alpha = 0.01,
  eps_iter = 1e-06,
  eps_gradient = 1e-06,
  max_iters = 1e+06,
  max_retries = 1e+06,
```

```
  ml_type = c("D_Only", "D_Beta"),
  verbose = 0
)
```

Arguments

formula	A formula. Its structure is similar to lmer function in **lme4** package. Models can be specified with the following form: `response ~ L(local.fixed) + global.fixed + (random | group)` For more information, please see the formula subsection in details.
data	The data.
...	Further arguments for the specified type of data.
bw	A numeric value. It is the value of bandwidth or "CV". In this stage this function only support adaptive bandwidth. And its unit must be the number of nearest neighbours. If "CV" is specified, the algorithm will automatically select an optimized bandwidth value.
kernel	A character value. It specify which kernel function is used in GWR part. Possible values are gaussian Gaussian kernel function $k(d) = \exp\left(-\frac{d^2}{b^2}\right)$ bisquared Bi-squared kernel function. If $d < b$ then $k(d) = \left(1 - \frac{d^2}{b^2}\right)^2$ else $k(d) = 0$
alpha	A numeric value. It is the size of the first trial step in maximum likelihood algorithm.
eps_iter	A numeric value. Terminate threshold of back-fitting.
eps_gradient	A numeric value. Terminate threshold of maximum likelihood algorithm.
max_iters	An integer value. The maximum of iteration.
max_retries	An integer value. If the algorithm tends to be diverge, it stops automatically after trying *max_retires* times.
ml_type	An integer value. Represent which maximum likelihood algorithm is used. Possible values are: D_Only Only D is specified by maximum likelihood. D_Beta Both D and $beta$ is specified by maximum likelihood.
verbose	An integer value. Determine the log level. Possible values are: **0** no log is printed. **1** only logs in back-fitting are printed. **2** all logs are printed.
coords	A 2-column matrix. It consists of coordinates for each group.

Details

Effect Specification in Formula:

In the HGWR model, there are three types of effects specified by the `formula` argument:

Local fixed effects Effects wrapped by functional symbol `L`.

Random effects Effects specified outside the functional symbol `L` but to the left of symbol `|`.

Global fixed effects Other effects

For example, the following formula in the example of this function below is written as

```
y ~ L(g1 + g2) + x1 + (z1 | group)
```

where `g1` and `g2` are local fixed effects, `x1` is the global fixed effects, and `z1` is the random effects grouped by the group indicator `group`. Note that random effects can only be specified once!

Value

A list describing the model with following fields.

`gamma` Coefficients of local fixed effects.

`beta` Coefficients of global fixed effects.

`mu` Coefficients of random effects.

`D` Variance-covariance matrix of random effects.

`sigma` Variance of errors.

`effects` A list including names of all effects.

`call` Calling of this function.

`frame` The DataFrame object sent to this call.

`frame.parsed` Variables extracted from the data.

`groups` Unique group labels extracted from the data.

Functions

- `hgwr_fit()`: Fit a HGWR model

Examples

```
data(multisampling)
hgwr(formula = y ~ L(g1 + g2) + x1 + (z1 | group),
     data = multisampling$data,
     coords = multisampling$coords,
     bw = 10)
```

logLik.hgwrm *Log likelihood function*

Description

Log likelihood function

Usage

```
## S3 method for class 'hgwrm'
logLik(object, ...)
```

Arguments

object	An hgwrm object.
...	Additional arguments.

Value

An logLik instance used for S3 method logLik().

make.dummy *Make Dummy Variables*

Description

Function make.dummy converts categorical variables in a data frame to dummy variables.

Function make.dummy.extract converts a column to dummy variables if necessary and assign appropriate names. See the "detail" section for further information. Users can define their own functions to allow the model deal with some types of variables properly.

Usage

```
make.dummy(data)

make.dummy.extract(col, name)

## S3 method for class 'character'
make.dummy.extract(col, name)

## S3 method for class 'factor'
make.dummy.extract(col, name)

## S3 method for class 'logical'
make.dummy.extract(col, name)
```

```
## Default S3 method:
make.dummy.extract(col, name)
```

Arguments

data	The data frame from which dummy variables need to be extracted.
col	A vector to extract dummy variables.
name	The vector's name.

Details

If col is a character vector, the function will get unique values of its elements and leave out the last one. Then, all the unique values are combined with the name argument as names of new columns.

If col is a factor vector, the function will get its levels and leave out the last one. Then, all level labels are combined with the name argument as names of new columns.

If col is a logical vector, the function will convert it to a numeric vector with value TRUE mapped to 1 and FALSE to 0.

If col is of other types, the default behaviour for extracting dummy variables is just to copy the original value and try to convert it to numeric values.

Value

The data frame with extracted dummy variables.

Examples

```
make.dummy(iris["Species"])

make.dummy.extract(iris$Species, "Species")

make.dummy.extract(c("top", "mid", "low", "mid", "top"), "level")

make.dummy.extract(factor(c("far", "near", "near")), "distance")

make.dummy.extract(c(TRUE, TRUE, FALSE), "sold")
```

---

multisampling *Simulated Spatial Multisampling Data (DataFrame)*

Description

A simulation data of spatial hierarchical structure and samples overlapping on certain locations.

Usage

```
data(multisampling)
```

Format

A list of two items called "data" and "coord". Item "data" is a data frame with 484 observations at 16 locations on the following 6 variables.

`y` a numeric vector, dependent variable y

`g1` a numeric vector, group level independent variable g_1

`g2` a numeric vector, group level independent variable g_2

`z1` a numeric vector, sample level independent variable z_1

`x1` a numeric vector, sample level independent variable x_1

`group` a numeric vector, group id of each sample

where `g1` and `g2` are used to estimate local fixed effects; `x1` is used to estimate global fixed effects and `z1` is used to estimate random effects.

Author(s)

Yigong Hu <yigong.hu@bristol.ac.uk>

Examples

```
data(multisampling)
hgwr(formula = y ~ L(g1 + g2) + x1 + (z1 | group),
     data = multisampling$data,
     coords = multisampling$coords,
     bw = 10)
```

---

multisampling.large *Large Scale Simulated Spatial Multisampling Data (DataFrame)*

Description

A large scale simulation data of spatial hierarchical structure and samples overlapping on certain locations.

Usage

```
data(multisampling)
```

Format

A list of three items called "data", "coords" and "beta". Item "data" is a data frame with 13862 observations at 200 locations and the following 6 variables.

`y` a numeric vector, dependent variable y

`g1` a numeric vector, group level independent variable g_1

`g2` a numeric vector, group level independent variable g_2

`z1` a numeric vector, sample level independent variable z_1

`x1` a numeric vector, sample level independent variable x_1

`group` a numeric vector, group id of each sample

where `g1` and `g2` are used to estimate local fixed effects; `x1` is used to estimate global fixed effects and `z1` is used to estimate random effects.

Author(s)

Yigong Hu <yigong.hu@bristol.ac.uk>

Examples

```
## Not run:
data(multisampling.large)
hgwr(formula = y ~ L(g1 + g2) + x1 + (z1 | group),
     data = multisampling.large$data,
     coords = multisampling.large$coords,
     bw = 32, kernel = "bisquared")

## End(Not run)
```

`print.hgwrm` *Print description of a* `hgwrm` *object.*

Description

Print description of a `hgwrm` object.

Usage

```
## S3 method for class 'hgwrm'
print(x, decimal.fmt = "%.6f", ...)
```

Arguments

`x`	An `hgwrm` object returned by `hgwr()`.
`decimal.fmt`	The format string passing to `base::sprintf()`.
`...`	Arguments passed on to `print.table.md` `col.sep` Column separator. Default to `""`. `header.sep` Header separator. Default to `"-"`. `row.begin` Character at the beginning of each row. Default to `col.sep`. `row.end` Character at the ending of each row. Default to `col.sep`. `table.style` Name of pre-defined style. Possible values are `"plain"`, `"md"` or `"latex"`. Default to `"plain"`.

Value

No return.

See Also

summary.hgwrm(), print.table.md().

Examples

```
data(multisampling)
model <- hgwr(formula = y ~ L(g1 + g2) + x1 + (z1 | group),
              data = multisampling$data,
              coords = multisampling$coords,
              bw = 10)
print(model)
print(model, table.style = "md")
```

---

print.summary.hgwrm *Print summary of an* hgwrm *object.*

Description

Print summary of an hgwrm object.

Usage

```
## S3 method for class 'summary.hgwrm'
print(x, decimal.fmt = "%.6f", ...)
```

Arguments

x	An object returned from summary.hgwrm().
decimal.fmt	The format string passing to base::sprintf().
...	Arguments passed on to print.table.md col.sep Column separator. Default to "". header.sep Header separator. Default to "-". row.begin Character at the beginning of each row. Default to col.sep. row.end Character at the ending of each row. Default to col.sep. table.style Name of pre-defined style. Possible values are "plain", "md" or "latex". Default to "plain".

Value

No return.

See Also

summary.hgwrm(), print.table.md().

Examples

```
data(multisampling)
model <- hgwr(formula = y ~ L(g1 + g2) + x1 + (z1 | group),
              data = multisampling$data,
              coords = multisampling$coords,
              bw = 10)
summary(model)
```

`print.table.md` *Print a character matrix as a table.*

Description

Print a character matrix as a table.

Usage

```
## S3 method for class 'table.md'
print(
  x,
  col.sep = "",
  header.sep = "",
  row.begin = "",
  row.end = "",
  table.style = c("plain", "md", "latex"),
  ...
)
```

Arguments

`x`	A character matrix.
`col.sep`	Column separator. Default to `""`.
`header.sep`	Header separator. Default to `"-"`.
`row.begin`	Character at the beginning of each row. Default to `col.sep`.
`row.end`	Character at the ending of each row. Default to `col.sep`.
`table.style`	Name of pre-defined style. Possible values are `"plain"`, `"md"` or `"latex"`. Default to `"plain"`.
`...`	Additional style control arguments.

Details

When `table.style` is specified, `col.sep`, `header.sep`, `row.begin` and `row.end` would not take effects. Because this function will automatically set their values. For each possible value of `table.style`, its corresponding style settings are shown in the following table.

	plain	md	latex
col.sep	""	"|"	"&"
header.sep	""	"-"	""
row.begin	""	"|"	""
row.end	""	"|"	"\\"

In this function, characters are right padded by spaces.

Value

No return.

See Also

`print.hgwrm()`, `summary.hgwrm()`.

`residuals.hgwrm` *Get residuals.*

Description

Get residuals.

Usage

```
## S3 method for class 'hgwrm'
residuals(object, ...)
```

Arguments

`object`	An `hgwrm` object returned by `hgwr()`.
`...`	Parameter received from other functions.

Value

A vector consists of residuals.

See Also

`hgwr()`, `summary.hgwrm()`, `coef.hgwrm()` and `fitted.hgwrm()`.

summary.hgwrm *Summary an* hgwrm *object.*

Description

Summary an hgwrm object.

Usage

```
## S3 method for class 'hgwrm'
summary(object, ..., test_hetero = FALSE)
```

Arguments

object	An hgwrm object returned from hgwr().
...	Other arguments passed from other functions.
test_hetero	Logical/list value. Whether to test the spatial heterogeneity of local fixed effects. If it is set to FALSE, the test will not be executed. If it is set to TRUE, the test will be executed with default parameters (see details below). It accepts a list to enable the test with specified parameters.

Details

The parameters used to perform test of spatial heterogeneity are

bw Bandwidth (unit: number of nearest neighbours) used to make spatial kernel density estimation. Default: 10.

poly The number of polynomial terms used in the local polynomial estimation. Default: 2.

resample Total resampling times. Default: 5000.

Value

A list containing summary informations of this hgwrm object with the following fields.

diagnostic A list of diagnostic information.

random.stddev The standard deviation of random effects.

random.corr The correlation matrix of random effects.

residuals The residual vector.

See Also

hgwr().

wuhan.hp *Wuhan Second-hand House Price and POI Data (DataFrame)*

Description

A data set of second-hand house price in Wuhan, China collected in 2018.

Usage

```
data(multisampling)
```

Format

A list of two items called "data" and "coords". Item "data" is a data frame with 13862 second-hand properties at 779 neighbourhoods and the following 22 variables.

`Price` House price per square metre.

`Floor.High` 1 if a property is on a high floor, otherwise 0.

`Floor.Low` 1 if a property is on a low floor, otherwise 0.

`Decoration.Fine` 1 if a property is well decorated, otherwise 0.

`PlateTower` 1 if a property is of the plate-tower type, otherwise 0.

`Steel` 1 if a property is of 'steel' structure, otherwise 0.

`BuildingArea` Building area in square metres.

`Fee` Management fee per square meter per month.

`d.Commercial` Distance to the nearest commercial area.

`d.Greenland` Distance to the nearest green land.

`d.Water` Distance to the nearest river or lake.

`d.University` Distance to the nearest university.

`d.HighSchool` Distance to the nearest high school.

`d.MiddleSchool` Distance to the nearest middle school.

`d.PrimarySchool` Distance to the nearest primary school.

`d.Kindergarten` Distance to the nearest kindergarten.

`d.SubwayStation` Distance to the nearest subway station.

`d.Supermarket` Distance to the nearest supermarket.

`d.ShoppingMall` Distance to the nearest shopping mall.

`lon` Longitude coordinates (Projected CRS: EPSG 3857).

`lat` Latitude coordinates (Projected CRS: EPSE 3857).

`group` Group id of each sample.

The following variables are group level:

- Fee - d.Commercial - d.Greenland - d.Water - d.University - d.HighSchool - d.MiddleSchool - d.PrimarySchool - d.Kindergarten - d.SubwayStation - d.Supermarket - d.ShoppingMall

The following variables are sample level:

- Price - Floor.High - Floor.Low - Decoration.Fine - PlateTower - Steel - BuildingArea

Item "coords" is a 779-by-2 matrix of coordinates of all neighbourhoods.

Author(s)

Yigong Hu <yigong.hu@bristol.ac.uk>

Examples

```
## Not run:
data(wuhan.hp)
hgwr(
  formula = Price ~ L(d.Water + d.Commercial + d.PrimarySchool +
            d.Kindergarten + Fee) + BuildingArea + (Floor.High | group),
  data = wuhan.hp$data,
  coords = wuhan.hp$coords, bw = 50, kernel = "bisquared")

## End(Not run)
```

Index