Package ‘GWlasso’

July 21, 2025

Title Geographically Weighted Lasso

Version 1.0.1

Description Performs geographically weighted Lasso regressions. Find optimal bandwidth, fit a geographically weighted lasso or ridge regression, and make predictions.
These methods are specially well suited for ecological inferences. Bandwidth selection algorithm is from A. Comber and P. Harris (2018) <doi:10.1007/s10109-018-0280-7>.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.1

Imports dplyr, ggplot2, ggside, glmnet, GWmodel, lifecycle, magrittr, methods, progress, rlang, sf, tidyr

NeedsCompilation no

Author Matthieu Mulot [aut, cre, cph] (ORCID: <https://orcid.org/0000-0002-8039-5078>),
Sophie Erb [aut] (ORCID: <https://orcid.org/0000-0002-0700-283X>)

Maintainer Matthieu Mulot <matthieu.mulot@gmail.com>

Suggests knitr, maps, rmarkdown

VignetteBuilder knitr

Depends R (>= 3.5.0)

LazyData true

URL https://github.com/nibortolum/GWlasso,

https://nibortolum.github.io/GWlasso/

BugReports https://github.com/nibortolum/GWlasso/issues

Repository CRAN

Date/Publication 2024-11-22 09:30:07 UTC

Contents

`Amesbury` *Amesbury Testate Amoebae dataset*

Description

Dataset from Amesbury (2016) in Development of a new pan-European testate amoeba transfer function for reconstructing peatland palaeohydrology

Usage

```
Amesbury
```

Format

`Amesbury`**:**

This dataset contains the data from Amesbury (2016). In essence, it's a Testate amoebae community table (45 broad TA taxa and 1103 samples)

spe.df A species x sites dataframe with stites as rows and species in column

WTD A vector od Water table depth associated with each samples

coords a dataframe containing the coordinates of each sample

Source

doi:10.1016/j.quascirev.2016.09.024

References

Matthew J. Amesbury, Graeme T. Swindles, Anatoly Bobrov, Dan J. Charman, Joseph Holden, Mariusz Lamentowicz, Gunnar Mallon, Yuri Mazei, Edward A.D. Mitchell, Richard J. Payne, Thomas P. Roland, T. Edward Turner, Barry G. Warner, *Development of a new pan-European testate amoeba transfer function for reconstructing peatland palaeohydrology. Quaternary Science Reviews, vol. 152, 2016, pages 132-151*. doi:10.1016/j.quascirev.2016.09.024.

`compute_distance_matrix` *Compute distance matrix*

Description

`compute_distance_matrix()` is a small helper function to help you compute a distance matrix. For the geographically method to work, is is important that distances between points are not zero. This function allows to add a small random noise to avoid zero distances.

Usage

```
compute_distance_matrix(data, method = "euclidean", add.noise = FALSE)
```

Arguments

`data`	A dataframe or matrix containing at least two numerical columns.
`method`	method to compute the distance matrix. Ultimately passed to `stats::dist()`. Can be `euclidean`, `maximum`, `manhattan`, `canberra`, `binary` or `minkowski`.
`add.noise`	TRUE/FALSE set to TRUE to add a small noise to the distance matrix. Noise U is generated as $U \sim (1 \times 10^{-6}, 5 \times 10^{-6})$. Noise is added only for pairs for which distance is zero.

Value

a distance matrix, usable in `gwl_bw_estimation()`

Examples

```
coords <- data.frame("Lat" = rnorm(200), "Long" = rnorm(200))
distance_matrix <- compute_distance_matrix(coords)
```

`gwl_bw_estimation` *Bandwidth estimation for Geographically Weighted Lasso*

Description

This function performs a bruteforce selection of the optimal bandwidth for the selected kernel to perform a geographically weighted lasso. The user should be aware that this function could be really long to run depending of the settings. We recommend starting with `nbw = 5` and `nfolds = 5` at first to ensure that the function is running properly and producing the desired output.

Usage

```
gwl_bw_estimation(
  x.var,
  y.var,
  dist.mat,
  adaptive = TRUE,
  adptbwd.thresh = 0.1,
  kernel = "bisquare",
  alpha = 1,
  progress = TRUE,
  nbw = 100,
  nfolds = 5
)
```

Arguments

`x.var`	input matrix, of dimension nobs x nvars; each row is an observation vector. `x` should have 2 or more columns.
`y.var`	response variable for the lasso
`dist.mat`	a distance matrix. can be generated by `compute_distance_matrix()`
`adaptive`	TRUE or FALSE Whether to perform an adaptive bandwidth search or not. A fixed bandwidth means that than samples are selected if they fit a determined fixed radius around a location. in a aptative bandwidth , the radius around a location varies to gather a fixed number of samples around the investigated location
`adptbwd.thresh`	the lowest fraction of samples to take into account for local regression. Must be 0 < `adptbwd.thresh` < 1
`kernel`	the geographical kernel shape to compute the weight. passed to `GWmodel::gw.weight()` Can be `gaussian`, `exponential`, `bisquare`, `tricube`, `boxcar`
`alpha`	the elasticnet mixing parameter. set 1 for lasso, 0 for ridge. see `glmnet::glmnet()`
`progress`	if TRUE, print a progress bar
`nbw`	the number of bandwidth to test
`nfolds`	the number f folds for the glmnet cross validation

Value

a `gwlest` object. It is a list with `rmspe` (the RMSPE of the model with the associated badwidth), `NA` (the number of NA in the dataset), `bw` (the optimal bandwidth), `bwd.vec` (the vector of tested bandwidth)

References

A. Comber and P. Harris. *Geographically weighted elastic net logistic regression (2018). Journal of Geographical Systems, vol. 20, no. 4, pages 317–341*. doi:10.1007/s1010901802807.

Examples

```
predictors <- matrix(data = rnorm(2500), 50,50)
y_value <- sample(1:1000, 50)
coords <- data.frame("Lat" = rnorm(50), "Long" = rnorm(50))
distance_matrix <- compute_distance_matrix(coords)


  myst.est <- gwl_bw_estimation(x.var = predictors,
                                y.var = y_value,
                                dist.mat = distance_matrix,
                                adaptive = TRUE,
                                adptbwd.thresh = 0.5,
                                kernel = "bisquare",
                                alpha = 1,
                                progress = TRUE,
                                n=10,
                                nfolds = 5)


  myst.est
```

---

gwl_fit *Fit a geographically weighted lasso with the selected bandwidth*

Description

Fit a geographically weighted lasso with the selected bandwidth

Usage

```
gwl_fit(
  bw,
  x.var,
  y.var,
  kernel,
  dist.mat,
  alpha,
  adaptive,
  progress = TRUE,
  nfolds = 5
)
```

Arguments

bw	Bandwidth

x.var	input matrix, of dimension nobs x nvars; each row is an observation vector. x should have 2 or more columns.
y.var	response variable for the lasso
kernel	the geographical kernel shape to compute the weight. passed to `GWmodel::gw.weight()` Can be `gaussian`, `exponential`, `bisquare`, `tricube`, `boxcar`
dist.mat	a distance matrix. can be generated by `compute_distance_matrix()`
alpha	the elasticnet mixing parameter. set 1 for lasso, 0 for ridge. see `glmnet::glmnet()`
adaptive	TRUE or FALSE Whether to perform an adaptive bandwidth search or not. A fixed bandwidth means that samples are selected if they fit a determined fixed radius around a location. In a adaptive bandwidth, the radius around a location varies to gather a fixed number of samples around the investigated location
progress	TRUE/FALSE whether to display a progress bar or not
nfolds	the number f folds for the glmnet cross validation

Value

a `gwlfit` object containing a fitted Geographically weighted Lasso.

Examples

```
predictors <- matrix(data = rnorm(2500), 50,50)
y_value <- sample(1:1000, 50)
coords <- data.frame("Lat" = rnorm(50), "Long" = rnorm(50))
distance_matrix <- compute_distance_matrix(coords)

my.gwl.fit <- gwl_fit(bw = 20,
                      x.var = predictors,
                      y.var = y_value,
                      kernel = "bisquare",
                      dist.mat = distance_matrix,
                      alpha = 1,
                      adaptive = TRUE,
                      progress = TRUE,
                      nfolds = 5)

my.gwl.fit
```

`plot.gwlfit` *Plot method for gwlfit object*

Description

Plot method for gwlfit object

Usage

```
## S3 method for class 'gwlfit'
plot(x, ...)
```

Arguments

x	a gwlfit object returned by gwl_fit()
...	ellipsis for S3 method compatibility

Value

a ggplot

Examples

```
predictors <- matrix(data = rnorm(2500), 50,50)
y_value <- sample(1:1000, 50)
coords <- data.frame("Lat" = rnorm(50), "Long" = rnorm(50))
distance_matrix <- compute_distance_matrix(coords)

my.gwl.fit <- gwl_fit(bw = 20,
                      x.var = predictors,
                      y.var = y_value,
                      kernel = "bisquare",
                      dist.mat = distance_matrix,
                      alpha = 1,
                      adaptive = TRUE,
                      progress = TRUE,
                      nfolds = 5)

plot(my.gwl.fit)
```

---

plot_gwl_map — *Plot a map of beta coefficient for gwlfit object*

Description

this function plots a map of the beta coefficients for a selected column (aka species). For this function to work, the coordinates supplied to gwl_fit() must be named "Lat" and "Long". The function is not bulletproof yet but is added here to reproduce the maps from the original publication.

[Experimental]

Usage

```
plot_gwl_map(x, column, crs = 4326)
```

Arguments

x	a gwlfit object returned by gwl_fit().
column	the name of a variable to be plotted on the map. Must be quoted. for instance "NEB.MIN"
crs	the crs projection for the map (default is mercator WGS84). See sf::st_crs()

Value

a ggplot object

Examples

```
data(Amesbury)

distance_matrix <- compute_distance_matrix(Amesbury$coords[1:30,], add.noise = TRUE)

my.gwl.fit <- gwl_fit(bw= 20,
                      x.var = Amesbury$spe.df[1:30,],
                      y.var = Amesbury$WTD[1:30],
                      dist.mat = distance_matrix,
                      adaptive = TRUE,
                      kernel = "bisquare",
                      alpha = 1,
                      progress = TRUE)

if(requireNamespace("maps")){
    plot_gwl_map(my.gwl.fit, column = "NEB.MIN")
  }
```

---

predict.gwlfit *Predict method for gwlfit objects*

Description

Predict method for gwlfit objects

Usage

```
## S3 method for class 'gwlfit'
predict(object, newdata, newcoords, type = "response", verbose = FALSE, ...)
```

Arguments

object	Object of class inheriting from "gwlfit"
newdata	a data.frame or matrix with the same columns as the training dataset
newcoords	a dataframe or matrix of coordinates of the new data

type	the type of response. see glmnet::predict.glmnet()
verbose	TRUE to print info about the execution of the function (useful for very large predictions)
...	ellipsis for S3 compatibility. Not used in this function.

Value

a vector of predicted values

Examples

```
predictors <- matrix(data = rnorm(2500), 50,50)
y_value <- sample(1:1000, 50)
coords <- data.frame("Lat" = rnorm(50), "Long" = rnorm(50))
distance_matrix <- compute_distance_matrix(coords)

my.gwl.fit <- gwl_fit(bw = 20,
                      x.var = predictors,
                      y.var = y_value,
                      kernel = "bisquare",
                      dist.mat = distance_matrix,
                      alpha = 1,
                      adaptive = TRUE,
                      progress = TRUE,
                      nfolds = 5)

my.gwl.fit

new_predictors <- matrix(data = rnorm(500), 10,50)
new_coords <- data.frame("Lat" = rnorm(10), "Long" = rnorm(10))

predicted_values <- predict(my.gwl.fit,
                             newdata = new_predictors,
                             newcoords = new_coords)
```

---

print.gwlest *Printing gwlest objects*

Description

Printing gwlest objects

Usage

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

Arguments

`x`	an object of class `gwlest`
`...`	ellipsis for S3 method compatibility

Value

this function print key elements of a `gwlest` object

Examples

```
predictors <- matrix(data = rnorm(2500), 50,50)
y_value <- sample(1:1000, 50)
coords <- data.frame("Lat" = rnorm(50), "Long" = rnorm(50))
distance_matrix <- compute_distance_matrix(coords)


myst.est <- gwl_bw_estimation(x.var = predictors,
                              y.var = y_value,
                              dist.mat = distance_matrix,
                              adaptive = TRUE,
                              adptbwd.thresh = 0.5,
                              kernel = "bisquare",
                              alpha = 1,
                              progress = TRUE,
                              n=10,
                              nfolds = 5)

myst.est
```

`print.gwlfit` *Printing gwlfit objects*

Description

Printing gwlfit objects

Usage

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

Arguments

`x`	a `gwlfit` object
`...`	ellipsis for S3 method compatibility

Value

this function print key elements of a gwlfit object

Examples

```
predictors <- matrix(data = rnorm(2500), 50,50)
y_value <- sample(1:1000, 50)
coords <- data.frame("Lat" = rnorm(50), "Long" = rnorm(50))
distance_matrix <- compute_distance_matrix(coords)

my.gwl.fit <- gwl_fit(bw = 20,
                      x.var = predictors,
                      y.var = y_value,
                      kernel = "bisquare",
                      dist.mat = distance_matrix,
                      alpha = 1,
                      adaptive = TRUE,
                      progress = TRUE,
                      nfolds = 5)

my.gwl.fit
```

Index