Package 'EIEntropy'

March 18, 2025

Title Ecological Inference Applying Entropy

Version 0.0.1.4

Depends R (>= 3.5.0)

Maintainer Silvia María Franco Anaya <sfrana@unileon.es>

Description

Implements two estimations related to the foundations of info metrics applied to ecological inference. These methodologies assess the lack of disaggregated data and provide an approach to obtaining disaggregated territorial-level data. For more details, see the following references: Fernández-Vázquez, E., Díaz-Dapena, A., Rubiera-Morollón, F. et al. (2020) ``Spatial Disaggregation of Social Indicators: An Info-Metrics Approach." <doi:10.1007/s11205-020-02455-z>. Díaz-Dapena, A., Fernández-Vázquez, E., Rubiera-Morollón, F., & Vinuela, A. (2021) ``Mapping poverty at the local level in Europe: A consistent spatial disaggregation of the AROPE indicator for France, Spain, Portugal and the United Kingdom." <doi:10.1111/rsp3.12379>.

License GPL-3

Encoding UTF-8

RoxygenNote 7.3.2

Imports dplyr, magrittr

Suggests devtools, knitr, rmarkdown, here

VignetteBuilder knitr, rmarkdown

Author Alberto Díaz-Dapena [aut, cph],
Esteban Fernández-Vázquez [aut, cph],
Silvia María Franco Anaya [aut, cre, cph]

NeedsCompilation no

Repository CRAN

Date/Publication 2025-03-17 23:30:02 UTC

Contents

`ei_gce` *Ecologic Inference applying entropy*

Description

The function ei_gce defines the Kullback-Leibler function which minimises the distance between the distribution of probabilities P and the distribution Q. The distribution Q is based on prior information that we have of our variable of interest previous to the analysis. The function will set the optimization parameters and, using the "nlminb" function, an optimal solution is obtained. The function defines the independent variables in the two databases needed, which we call dataA with "n_A" observations and dataB with "n_B" observations; and the function of the variable of interest y. Then the weights of each observation for the two databases used are defined, if there are not weights available it will be 1 by default. The errors are calculated pondering the support vector of dimension `var, 0, -var`. This support vector can be specified by the user. The default support vector is based on variance. We recommend a wider interval with v(1,0,-1) as the maximum. The restrictions are defined in order to guarantee consistency. The minimization of Kullback_Leibler distance is solved with "nlminb" function with maximum number of iterations 1000 and with tolerance defined by the user. If the user did not define tolerance it will be 1e-10 by default. For additional details about the methodology see Fernández-Vazquez, et al. (2020)

Usage

```
ei_gce(fn, dataA, dataB, q, weights = NULL, v, tol, iter)
```

Arguments

`fn`	Is the formula that represents the dependent variable in the optimization. In the context of this function, 'fn' is used to define the dependent variable to be optimized by the Kullback-Leibler divergence function. Note: If the dependent variable is categorical the sorting criterion for the columns, and therefore for J, is alphabetical order.
`dataA`	The data where the variable of interest y is available and also the independent variables. Note: The variables and weights used as independent variables must have the same name in 'dataA' and in 'dataB'
`dataB`	The data which contains information on the independent variables at a disaggregated level. Note: The variables and weights used as independent variables must have the same name in 'dataA' and in 'dataB'. The variables in both databases need to match up in content.
`q`	The prior distribution Q

weights	A character string specifying the column name to be used as weights in both 'dataA' and 'dataB' datasets. If the argument weights is provided and present in both datasets, the weights in each dataset will be normalized by the sum of the weights within that dataset. If weights is NULL or the specified column does not exist in both datasets, equal weights are applied across all observations.
v	The support vector
tol	The tolerance to be applied in the optimization function. If the tolerance is not specified, the default tolerance has been set in 1e-10
iter	The maximum number of iterations allowed for the optimization algorithm to run Increasing the number of iterations may improve the likelihood of finding an optimal solution, but can also increases computation time.If the maximum number of iterations is not specified, it will default to 1000

Details

To solve the optimization upper and lower bounds for p and w are settled, specifically, p and w must be above 0 and lower than 1. In addition, the initial values of p are settled as the defined prior and the errors (w) as 1/L.

Value

The function will provide you a dataframe called estimations with the next information:

- **weights** The weights used in the optimization process.
- **predictions** The prediction for each individual is calculated as the sum of the probability plus the error.
- **probabilities** Probabilities for each individual to each possibility j of the variable of interest y.
- **errors** Errors calculated to the j possibilities of y. The function provides information about the optimization process as:
- **divergencekl** The Kullback-Leibler divergence value resulting from the optimization.
- **iterations** Indicates the times the objective function and the gradient has been evaluated during the optimization process,if any.
- **message** Indicates the message if it has been generated in the process of optimization.
- **q** Indicates prior implemented in the optimization.
- **tol** Indicates the tolerance of the optimization process.
- **v** Indicates the support vector used in the function. The function provides a dataframe containing the information about lambda:
- **lambda** The estimated lambda values. It is provided an object with the restrictions checked which should be zero.
- **check restrictions** Being g1 the restriction related to the unit probability constraint, g2 to the error unit sum constraint, and g3 to the consistency restriction that implies that the difference between the cross moment in both datasets must be zero. The restriction g3 can be checked thoroughly with the objects by separate.
- **cross moments A** Cross moments in dataA.
- **cross moments B** Cross moments in dataB.

References

Fernandez-Vazquez, E., Diaz-Dapena, A., Rubiera-Morollon, F., Viñuela, A., (2020) Spatial Disaggregation of Social Indicators: An Info-Metrics Approach. Social Indicators Research, 152(2), 809–821. https://doi.org/10.1007/s11205-020-02455-z.

Examples

```
#In this example we use the data of this package
dataA <- financial()
dataB <- social()
# Setting up our function for the dependent variable.
fn               <- dataA$poor_liq ~ Dcollege+Totalincome+Dunemp
#In this case we know that the mean probability of being poor is 0.35.With this function
#we can add the information as information a priori. This information a priori correspond to the
#Q distribution and in this function is called q for the sake of simplicity:
q<- c(0.35,0.65)
v<- matrix(c(0.2,0,-0.2))
#Applying the function ei_gce to our databases. In this case dataA is the
#dataA where we have our variable of interest
#dataB is the data where we have the information for the disaggregation.
#w can be included if we have weights in both surveys
result  <- ei_gce(fn,dataA,dataB,q=q,weights="w",v=v)
```

`ei_gme` *Ecologic Inference applying entropy*

Description

The function ei_gme defines the Shannon entropy function which takes a vector of probabilities as input and returns the negative sum of p times the natural logarithm of p.The function will set the optimization parameters and using the "nlminb" function an optimal solution is obtained. The function defines the independent variables in the two databases needed, which we call dataA with "n_A" observations and dataB with "n_B" observations; and the function of the binary variable of interest y. Then the weights of each observation for the two databases used are defined, if there are no weights available it will be 1. The errors are calculated pondering the support vector of dimension `var, 0, -var`. This support vector can be specified by the user. The default support vector is based on variance.We recommend a wider interval with v(1,0,-1) as the maximum. The restrictions are defined to guarantee consistency. The optimization of the Shannon entropy function is solved with "nlminb" function with maximum number of iterations 1000 and with tolerance defined by the user.

Usage

```
ei_gme(fn, dataA, dataB, weights = NULL, tol, v, iter)
```

Arguments

`fn`	Is the formula that represents the dependent variable in the optimization. In the context of this function, 'fn' is used to define the dependent variable to be optimized by the entropy function. Note: If the dependent variable is categorical the sorting criterion for the columns, and therefore for J, is alphabetical order.
`dataA`	The data where the variable of interest y is available and also the independent variables. Note: The variables and weights used as independent variables must have the same name in 'dataA' and in 'dataB' The variables in both databases need to match up in content.
`dataB`	The data which contains information on the independent variables at a disaggregated level. Note: The variables and weights used as independent variables must be the same and must have the same name in 'dataA' and in 'dataB'
`weights`	A character string specifying the column name to be used as weights in both 'dataA' and 'dataB' datasets. If the argument `weights` is provided and present in both datasets, the weights in each dataset will be normalized by the sum of the weights within that dataset. If `weights` is NULL or the specified column does not exist in both datasets, equal weights are applied across all observations.
`tol`	The tolerance to be applied in the optimization function. If the tolerance is not specified, the default tolerance has been set in 1e-10
`v`	The support vector
`iter`	The maximum number of iterations allowed for the optimization algorithm to run Increasing the number of iterations may improve the likelihood of finding an optimal solution, but can also increases computation time.If the maximum number of iterations is not specified, it will default to 1000

Details

To solve the optimization upper and lower bounds for p and w are settled, specifically, p and w must be above 0 and lower than 1. In addition, the initial values of p are settled as a uniform distribution and the errors (w) as 1/L.

Value

The function will provide you a dataframe called table with the next information:

- **weights** The weights used in the optimization process.
- **predictions** The prediction for each individual is calculated as the sum of the probability plus the error.
- **probabilities** Probabilities for each individual to each possibility `j` of the variable of interest `y`.
- **errors** Errors calculated to the `j` possibilities of `y`. The function provides information about the optimization process as :
- **value_of_entropy** The value of entropy resulting from the optimization.
- **iterations** Indicates the times the objective function and the gradient has been evaluated during the optimization process

- **message** Indicates the message if it has been generated in the process of optimization
- **tol** Indicates the tolerance used in the optimization
- **v** Indicates the vector of support used in the function The function provides a dataframe containing the information about lambda:
- **lambda** The estimated lambda values. It is provided an object with the restrictions checked which should be approximately zero.
- **check restrictions** Being g1 the restriction related to the unit probability constraint, g2 to the error unit sum constraint, and g3 to the consistency restriction that implies that the difference between the cross moment in both datasets must be zero.

The restriction g3 can be checked thoroughly with the objects by separate.

- **cross moments A** Cross moments in `dataA`.
- **cross moments B** Cross moments in `dataB`.

References

Fernandez-Vazquez, E., Díaz-Dapena, A., Rubiera-Morollon, F., Viñuela, A., (2020) Spatial Disaggregation of Social Indicators: An Info-Metrics Approach. Social Indicators Research, 152(2), 809–821. https://doi.org/10.1007/s11205-020-02455-z.

Examples

```
#In this example we use the data of this package
dataA <- financial()
dataB <- social()
# Setting up our function for the dependent variable.
fn               <- dataA$poor_liq ~ Dcollege+Totalincome+Dunemp
#Applying the function ei_gme to our databases. In this case dataA
#is the data where we have our variable of interest dataB is the data
# where we have the information for the disaggregation.
#w can be included if we have weights in both surveys
#Tolerance in this example is fixed in 1e-10 and v will be (1,0,-1)
v=matrix(c(1, 0, -1), nrow = 1)
result  <- ei_gme(fn=fn,dataA=dataA,dataB=dataB,weights="w",v=v)
```

`financial` *Randomly Generated Data*

Description

This dataset contains 100 observations of 6 variables. The data was generated randomly for the purpose of exemplifying a database that could potentially be used with this function.

Usage

```
financial()
```

Format

A data frame with 100 observations of 6 variables called financial

Value

A data frame containing the loaded "financial" data from the .rds file.

- `Dcollege`: Dummy variable indicating college education.
- `Dunemp`: Dummy variable indicating unemployment.
- `Totalincome`: Total income of each observation.
- `poor_liq`: Dummy variable indicating liquid poverty.
- `w`: Weights for each observation.
- `n`: Identifier for observations.

Examples

```
data(financial)
head(financial)
```

---

plot.kl *Generate a Plot*

Description

This function generates a descriptive plot using the results obtained in ei_gce. It illustrates the mean and the confidence interval by disaggregated territorial unit.

Usage

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

Arguments

`x`	The output produced by ei_gce
`reg`	The data column containing the disaggregated territorial units
`...`	Additional arguments passed to the plotting function.

Value

This function provides a graph representing the weighted mean and confidence interval of each disaggregated territorial unit

plot.shannon *Generate a Plot*

Description

This function generates a descriptive plot using the result obtained in ei_gme. It illustrates the mean and the confidence interval by disaggregated territorial unit.

Usage

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

Arguments

x	The output produced by ei_gme
reg	The data column containing the disaggregated territorial units
...	Additional arguments passed to the plotting function.

Value

This function provides a graph representing the weighted mean and confidence interval of each disaggregated territorial unit

social *Randomly Generated Data*

Description

This dataset contains 200 observations of 6 variables. The data was generated randomly for the purpose of exemplifying a database that could potentially be used with this function.

Usage

```
social()
```

Format

A data frame with 200 observations of 6 variables called social

Value

A data frame containing the loaded "social" data from the .rds file.

- `Dcollege`: Dummy variable indicating college education.
- `Dunemp`: Dummy variable indicating unemployment.
- `Totalincome`: Total income of each observation.
- `reg`:Variable indicating the region of the observation.
- `w`: Weights for each observation.
- `n`: Identifier for observations.

#' @examples data(social) head(social)

`summary.kl` *Summary*

Description

This function provides a summary of the output obtained with the function ei_gce.

Usage

```
## S3 method for class 'kl'
summary(object, ...)
```

Arguments

`object`	The output obtained from ei_gce
`...`	Additional arguments passed to the summary function.

Value

This summary function returns the Kullback-Leibler divergence value and the last iteration in the optimization process. A dataframe with the means of the estimations for each characteristic j with the predictions the probabilities and the error estimated. A dataframe with the lambda estimated for each k.

- `Iterations`:Indicates the times the objective function and the gradient has been evaluated during the optimization process
- `divergencekl value`:The Kullback-Leibler divergence value resulting from the optimization.
- `mean_estimations`:The weighted mean of predictions, probabilities, and the errors for each category j of the variable y
- `lambda`:The estimated lambda values.

summary.shannon *Summary*

Description

This function provides a summary of the output obtained with the function ei_gme.

Usage

```
## S3 method for class 'shannon'
summary(object, ...)
```

Arguments

object	The output obtained from ei_gme
...	Additional arguments passed to the summary function.

Value

This summary function returns the entropy value and the last iteration in the optimization process. A dataframe with the means of the estimations for each characteristic j with the predictions the probabilities and the error estimated. A dataframe with the lambda estimated for each k.

- Iterations:Indicates the times the objective function and the gradient has been evaluated during the optimization process
- Entropy value:The value of entropy resulting from the optimization.
- mean_estimations: The weighted mean of predictions, p_dual, and the error for each category j of the variable y
- lambda:The estimated lambda values.

Index