Package ‘DHSr’

January 10, 2025

Type Package

Title Create Large Scale Repeated Regression Summary Statistics Dataset and Visualization Seamlessly

Version 0.1.0

Maintainer Arnab Samanta <arnob.shamanta62@gmail.com>

Description
Mapping, spatial analysis, and statistical modeling of microdata from sources such as the Demographic and Health Surveys <https://www.dhsprogram.com/> and Integrated Public Use Microdata Series <https://www.ipums.org/>. It can also be extended to other datasets. The package supports spatial correlation index construction and visualization, along with empirical Bayes approximation of regression coefficients in a multistage setup. The main functionality is repeated regression — for example, if we have to run regression for n groups, the group ID should be vertically composed into the variable for the parameter `location_var`. It can perform various kinds of regression, such as Generalized Regression Models, logit, probit, and more. Additionally, it can incorporate interaction effects. The key benefit of the package is its ability to store the regression results performed repeatedly on a dataset by the group ID, along with respective p-values and map those estimates.

License GPL-3

Encoding UTF-8

RoxygenNote 7.3.1

Imports dplyr, ggplot2, rlang, sf, spdep, viridis, nlme, MuMIn, tidyr, stats

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

Config/testthat/edition 3

VignetteBuilder knitr

NeedsCompilation no

Author Arnab Samanta [aut, cre] (<arnob.shamanta62@gmail.com>)

Repository CRAN

Date/Publication 2025-01-10 15:00:06 UTC

Contents

`cluster_map` *Create Cluster Map Based on Local Moran's I*

Description

This function creates a map of clusters based on Local Moran's I values. It identifies clusters using Queen contiguity and visualizes them on a map.

Usage

```
cluster_map(
  dataset,
  lisa_value,
  lisa_label,
  label,
  lisa_cutoff,
  location_var,
  location_name,
  level2 = NULL,
  id_start = 0,
  comparison = ">",
  min_area = 0,
  min_ = 5,
  title = "Clusters Based on Queen Contiguity",
  subtitle = "",
  footnote = "",
  legend_position = "bottom",
  color_scheme = "C"
)
```

Arguments

`dataset`	A spatial dataset of class ‘sf‘.
`lisa_value`	The name of the variable in the dataset containing Local Moran’s I values.
`lisa_label`	The name of the variable in the dataset containing the LISA label.
`label`	The specific label to filter clusters.
`lisa_cutoff`	A numeric value specifying the cutoff for LISA values.
`location_var`	The variable name indicating the primary location in the dataset.
`location_name`	The name of the variable for the location names.
`level2`	An optional second level of location hierarchy. Default is ‘NULL‘.
`id_start`	The starting value for cluster IDs. Default is ‘0‘.
`comparison`	The comparison operator for filtering (‘>‘, ‘<‘, ‘>=‘, etc.). Default is ‘>‘.
`min_area`	Minimum area required for a cluster to be considered valid. Default is ‘0‘.
`min_`	Minimum number of districts required for a cluster to be valid. Default is ‘5‘.
`title`	The title of the map. Default is ‘"Clusters Based on Queen Contiguity"‘.
`subtitle`	A subtitle for the map. Default is ‘""‘.
`footnote`	A footnote for the map. Default is ‘""‘.
`legend_position`	The position of the legend on the map. Default is ‘"bottom"‘.
`color_scheme`	The color scheme for the map. Default is ‘"C"‘.

Value

A list with the following components:

`dataset_with_clusters`	An ‘sf‘ object containing the dataset with assigned cluster IDs.
`summary_clusters`	A data frame summarizing cluster information, including regions, area, and the number of locations.
`plot`	A ‘ggplot‘ object for visualizing the clusters.

Examples

```
if (requireNamespace("spData", quietly = TRUE)) {
  library(sf)
  library(dplyr)
  library(spdep)
  library(spData)
  library(ggplot2)

  # Load US states data from spData
  us_states <- spData::us_states

  # Simplify for demonstration: Select a subset of columns
```

```
us_states_data <- us_states %>%
  select(GEOID, NAME) %>%
  mutate(mean_wealth = rnorm(nrow(us_states), 50, 10))  # Add mock data

# Define a shapefile path
shapefile_path <- tempfile(fileext = ".shp")
sf::st_write(us_states, shapefile_path, quiet = TRUE)

# Corrected listw function call using your package
us_states_listw <- DHSr::listw(
  shapefile_path = shapefile_path,
  data = us_states_data %>% sf::st_drop_geometry(),  # Drop geometry for compatibility
  loc_shape = "GEOID",
  loc_data = "GEOID",
  weight_function = function(d) exp(-d / 0.2)
)

# Apply Spdeplisa function
lisa_result <- DHSr::Spdeplisa(
  data = us_states_data,
  variable_name = "mean_wealth",
  listw = us_states_listw
)

# Add LISA labels
lisa_result <- lisa_result %>%
  mutate(lisa_label = case_when(
    lisa_I > 0 ~ "High-High",
    lisa_I < 0 ~ "Low-Low",
    TRUE ~ "Others"
  ))

# Apply cluster_map function
cluster_map_result <- DHSr::cluster_map(
  dataset = lisa_result,
  lisa_value = "lisa_I",
  lisa_label = "lisa_label",
  label = "High-High",
  lisa_cutoff = 0.5,
  location_var = "GEOID",
  location_name = "NAME",
  id_start = 1,
  comparison = ">",
  min_area = 0,
  min_ = 3,  # Reduced for smaller demonstration
  title = "Clusters Based on Queen Contiguity",
  subtitle = "High-High Clusters",
  footnote = "Generated using DHSr package",
  legend_position = "bottom",
  color_scheme = "C"
)

# View the resulting dataset with clusters
```

```
  head(cluster_map_result$dataset_with_clusters)

  # View the cluster summary
  print(cluster_map_result$summary_clusters)

  # Plot the clusters
  print(cluster_map_result$plot)
}
```

`listw` *Create Spatial Weights List*

Description

This function creates a spatial weights list using a shapefile and a dataset.

Usage

```
listw(
  shapefile_path,
  data,
  loc_shape,
  loc_data,
  weight_function = function(d) exp(-d/0.2)
)
```

Arguments

`shapefile_path`	A string specifying the file path to the shapefile.
`data`	A dataframe containing the variables to be analyzed.
`loc_shape`	A string specifying the column name in the shapefile used for merging.
`loc_data`	A string specifying the column name in the dataset that corresponds to the location variable.
`weight_function`	A function to calculate weights from distances. Defaults to ‘function(d) exp(-d / 0.2)‘.

Value

A spatial weights list object of class ‘listw‘.

Examples

```
if (requireNamespace("spData", quietly = TRUE)) {
    library(dplyr)
    library(sf)

    # Load US states data
    us_states <- spData::us_states

    # Simplify for demonstration: Select a subset of columns
    us_states_data <- us_states %>%
        select(GEOID, NAME) %>%
        mutate(mean_wealth = rnorm(nrow(us_states), 50, 10))  # Add mock data

    # Define a temporary shapefile path
    shapefile_path <- tempfile(fileext = ".shp")
    sf::st_write(us_states, shapefile_path, quiet = TRUE)

    # Use the listw function from the package
    us_states_listw <- DHSr::listw(
        shapefile_path = shapefile_path,
      data = us_states_data %>% sf::st_drop_geometry(),  # Drop geometry for compatibility
        loc_shape = "GEOID",
        loc_data = "GEOID",
        weight_function = function(d) exp(-d / 0.2)
    )

    # Verify the spatial weights list
    print(us_states_listw)
}
```

Repglmre2 *Loop through all locations and run GLMM for each*

Description

This function runs a mixed-effects generalized linear model (GLMM) for each location within a dataset.

Usage

```
Repglmre2(data, formula, location_var, random_effect_var, family)
```

Arguments

`data`	The dataset to be analyzed.
`formula`	The formula for the regression model.
`location_var`	The variable indicating different locations (e.g., ‘REGCODE‘).
`random_effect_var`	The variable to be used as a random effect (e.g., ‘hhid‘).
`family`	The family to be used for GLM (e.g., ‘binomial‘ for logistic regression, ‘poisson‘ for Poisson regression).

Value

A dataframe containing the results

Examples

```
set.seed(123)

# Create dummy data
  library(dplyr)
dummy_data <- data.frame(
  years_education = rnorm(100, 12, 3),    # Represents years of education
  gender_female = rbinom(100, 1, 0.5),    # 1 = Female, 0 = Male
  household_wealth = sample(1:5, 100, replace = TRUE),  # Wealth index from 1 to 5
  district_code = sample(1:10, 100, replace = TRUE)     # Represents district codes
) %>% arrange(district_code)

# Create HHid (Household ID), grouping every 3-4 records, and convert to character
dummy_data$HHid <- as.character(rep(1:20, each = 5, length.out = nrow(dummy_data)))

# Create a binary outcome variable for years of education
dummy_data$education_binary <- ifelse(dummy_data$years_education > 11, 1, 0)

# Define a logistic regression formula
formula <- education_binary ~ gender_female + household_wealth:gender_female

location_var <- "district_code"
random_effect_var <- "HHid"

# Run the logistic mixed-effects model across all locations (districts)
results <- DHSr::Repglmre2(data = dummy_data, formula = formula,
                      location_var = location_var, random_effect_var = random_effect_var,
                           family = binomial())

# Print the results
print(head(results))
```

`Replm2` *Run Regression Analysis for All Locations*

Description

This function runs a regression model for all unique locations within a dataset and combines the results.

Usage

```
Replm2(
  data,
  formula,
  location_var,
  response_distribution = "normal",
  family = NULL
)
```

Arguments

`data`	The dataset to be analyzed.
`formula`	The formula for the regression model.
`location_var`	The variable indicating different locations (e.g., ‘REGCODE‘).
`response_distribution`	The distribution of the response variable ("normal" for normal distribution, "other" for other distributions).
`family`	The family to be used for GLM if response_distribution is "other" (e.g., ‘binomial‘ for logistic regression).

Value

A dataframe containing the combined results for all locations.

Examples

```
set.seed(123)
library(dplyr)
dummy_data <- data.frame(
  years_education = rnorm(100, 12, 3),    # Represents years of education
  gender_female = rbinom(100, 1, 0.5),    # 1 = Female, 0 = Male
  household_wealth = sample(1:5, 100, replace = TRUE),  # Wealth index from 1 to 5
  district_code = sample(1:10, 100, replace = TRUE) # Represents district codes
) %>% arrange(district_code)

# Define a simple regression formula
formula <- years_education ~ gender_female + household_wealth + household_wealth:gender_female

# Run the regression across all locations (districts)
results1 <- Replm2(dummy_data, formula, "district_code", "normal")
print(results1)
```

`Replmre2` *Mixed-Effects Regression Analysis for All Locations*

Description

This function runs a mixed-effects regression model for all locations within a dataset.

Usage

```
Replmre2(data, formula, location_var, random_effect_var)
```

Arguments

`data`	The dataset to be analyzed.
`formula`	The formula for the regression model.
`location_var`	The variable indicating different locations (e.g., ‘REGCODE‘).
`random_effect_var`	The variable to be used as a random effect (e.g., ‘hhid‘).

Value

A dataframe containing the results

Examples

```
set.seed(123)
library(dplyr)
# Create dummy data
dummy_data <- data.frame(
  years_education = rnorm(100, 12, 3),    # Represents years of education
  gender_female = rbinom(100, 1, 0.5),    # 1 = Female, 0 = Male
  household_wealth = sample(1:5, 100, replace = TRUE),  # Wealth index from 1 to 5
  district_code = sample(1:10, 100, replace = TRUE)     # Represents district codes
) %>% arrange(district_code)

# Create HHid (Household ID), grouping every 3-4 records, and convert to character
dummy_data$HHid <- as.character(rep(1:20, each = 5, length.out = nrow(dummy_data)))

# Define a simple regression formula
formula <- years_education ~ gender_female + household_wealth:gender_female
location_var <- "district_code"
random_effect_var <- "HHid"

# Run mixed-effects regression for all districts
results <- DHSr::Replmre2(dummy_data, formula, location_var, random_effect_var)
print(head(results))
```

`single_glmre2` *Mixed-Effects Logistic Regression Analysis for a Specified Location*

Description

This function runs a mixed-effects logistic regression model for a specified location within a dataset.

Usage

```
single_glmre2(
  data,
  formula,
  location_var,
  random_effect_var,
  location_index,
  family = NULL
)
```

Arguments

`data`	The dataset to be analyzed.
`formula`	The formula for the regression model.
`location_var`	The variable indicating different locations (e.g., 'REGCODE').
`random_effect_var`	The variable to be used as a random effect (e.g., 'hhid').
`location_index`	The specific location index or number for which the model should be run.
`family`	The family to be used for GLM (e.g., 'binomial' for logistic regression).

Value

The results for sigle location to test

Examples

```
set.seed(123)
  library(dplyr)
# Create dummy data
dummy_data <- data.frame(
  years_education = rnorm(100, 12, 3),    # Represents years of education
  gender_female = rbinom(100, 1, 0.5),    # 1 = Female, 0 = Male
  household_wealth = sample(1:5, 100, replace = TRUE),  # Wealth index from 1 to 5
  district_code = sample(1:10, 100, replace = TRUE)     # Represents district codes
) %>% arrange(district_code)

# Create HHid (Household ID), grouping every 3-4 records, and convert to character
dummy_data$HHid <- as.character(rep(1:20, each = 5, length.out = nrow(dummy_data)))

# Create a binary outcome variable for years of education
dummy_data$education_binary <- ifelse(dummy_data$years_education > 11, 1, 0)

# Define a logistic regression formula
formula <- education_binary ~ gender_female + household_wealth:gender_female

# Set the location and random effect variables
location_var <- "district_code"
random_effect_var <- "HHid"

# Run the mixed-effects logistic regression for a specific location (e.g., district 1)
result_single_glmre <- single_glmre2(dummy_data, formula, location_var, random_effect_var,
 location_index = 1, family = binomial())

# View the result
print(result_single_glmre)
```

single_lm2 *Linear Regression Analysis for Specified Location*

Description

This function runs a linear regression model for a specified location within a dataset.

Usage

```
single_lm2(
  data,
  formula,
  location_var,
  response_distribution = "normal",
  family = NULL,
  location_index
)
```

Arguments

`data`	The dataset to be analyzed.
`formula`	The formula for the regression model.
`location_var`	The variable indicating different locations (e.g., ‘REGCODE‘).
`response_distribution`	The distribution of the response variable ("normal" for normal distribution, "other" for other distributions).
`family`	The family to be used for GLM if response_distribution is "other" (e.g., ‘binomial‘ for logistic regression).
`location_index`	The specific location index or number for which the model should be run.

Value

A dataframe containing the results for the specified location.

Examples

```
set.seed(123)
if (requireNamespace("dplyr", quietly = TRUE)) {
  library(dplyr)
# Create dummy data
dummy_data <- data.frame(
  years_education = rnorm(100, 12, 3),    # Represents years of education
  gender_female = rbinom(100, 1, 0.5),    # 1 = Female, 0 = Male
  household_wealth = sample(1:5, 100, replace = TRUE),  # Wealth index from 1 to 5
  district_code = sample(1:10, 100, replace = TRUE)     # Represents district codes
) %>% arrange(district_code)
# Define a simple regression formula
formula <- years_education ~ gender_female + household_wealth + household_wealth:gender_female

# Run the regression for a specific location (e.g., district 1)
result_single_lm <- single_lm2(dummy_data, formula, "district_code",
 response_distribution = "normal", location_index = 1)

# View the result
print(result_single_lm)
}
```

---

single_lmre2 *Mixed-Effects Regression Analysis for Specified Location*

Description

This function runs a mixed-effects regression model for a specified location within a dataset.

Usage

```
single_lmre2(data, formula, location_var, random_effect_var, location_index)
```

Arguments

data	The dataset to be analyzed.
formula	The formula for the regression model.
location_var	The variable indicating different locations (e.g., ‘REGCODE‘).
random_effect_var	The variable to be used as a random effect (e.g., ‘hhid‘).
location_index	The specific location index or number for which the model should be run.

Value

the results for the specified location to test

Examples

```
set.seed(123)
library(dplyr)
# Create dummy data
dummy_data <- data.frame(
  years_education = rnorm(100, 12, 3),    # Represents years of education
  gender_female = rbinom(100, 1, 0.5),    # 1 = Female, 0 = Male
  household_wealth = sample(1:5, 100, replace = TRUE),  # Wealth index from 1 to 5
  district_code = sample(1:10, 100, replace = TRUE)     # Represents district codes
) %>% arrange(district_code)

# Create HHid (Household ID), grouping every 3-4 records, and convert to character
```

```
dummy_data$HHid <- as.character(rep(1:20, each = 5, length.out = nrow(dummy_data)))

# Define a simple regression formula
formula <- years_education ~ gender_female + household_wealth:gender_female

# Set the location and random effect variables
location_var <- "district_code"
random_effect_var <- "HHid"

# Run the mixed-effects regression for a specific location (e.g., district 1)
result_single_lmre <- single_lmre2(dummy_data, formula, location_var,
random_effect_var, location_index = 1)

# View the result
print(result_single_lmre)
```

Spdeplisa — *Calculate Local Moran's I and Sign Combination Variables*

Description

This function calculates Local Moran's I for a specified variable in a dataset and creates sign combination variables based on the standardized variable and the local Moran's I values.

Usage

```
Spdeplisa(data, variable_name, listw)
```

Arguments

`data`	A dataframe containing the spatial data.
`variable_name`	A string representing the name of the variable to be analyzed.
`listw`	A listw object containing spatial weights for the dataset.

Value

A data frame containing the original data with additional columns:

`lisa_I`	Local Moran's I values for the specified variable.
`lisa_p`	P-values corresponding to the Local Moran's I values.
`z_i`	Standardized values of the input variable.
`sign_combination2`	Categories based on the sign of z_i and lisa_I (e.g., "positive-negative").
`sign_combination3`	Categories based on the sign of z_i and lisa_I (e.g., "High-High").

Examples

```
# Load necessary libraries
if (requireNamespace("spData", quietly = TRUE)) {
  library(spData)
  library(sf)
  library(dplyr)

  # Use US states data as a substitute for a shapefile
  us_states <- spData::us_states

  # Simplify for demonstration: Select a subset of columns
  us_states_data <- us_states %>%
    select(GEOID, NAME) %>%
    mutate(mean_wealth = rnorm(nrow(us_states), 50, 10))  # Add mock data

  # Define a temporary shapefile path
  shapefile_path <- tempfile(fileext = ".shp")
  sf::st_write(us_states, shapefile_path, quiet = TRUE)

  # Create spatial weights using the listw function from the package
  us_states_listw <- DHSr::listw(
    shapefile_path = shapefile_path,
    data = us_states_data %>% sf::st_drop_geometry(),  # Drop geometry for compatibility
    loc_shape = "GEOID",
    loc_data = "GEOID",
    weight_function = function(d) exp(-d / 0.2)
  )

  # Apply the Spdeplisa function
  lisa_result <- DHSr::Spdeplisa(
    data = us_states_data,
    variable_name = "mean_wealth",
    listw = us_states_listw
  )

  # View the result
  head(lisa_result)
}
```

`stein_beta` *Calculate Stein's Beta for Each Cluster*

Description

This function calculates Stein's Beta for each cluster within the dataset. It applies Stein's shrinkage estimator to the specified beta estimates within each cluster.

Usage

```
stein_beta(data, cluster_id, beta)
```

Arguments

`data`	A dataframe containing the data.
`cluster_id`	The name of the column representing the cluster IDs.
`beta`	The name of the column representing the beta estimates.

Value

A data frame containing the input data with additional columns:

`stein_beta`	The Stein-shrinkage adjusted beta values.
`lambda_d`	Shrinkage factors for each cluster.
`mu_beta_m`	Mean beta values for each cluster.
`sigma_hat_sq`	Estimated variance of the beta values within clusters.
`sum_of_squares`	Sum of squared deviations of beta values from their mean.

Examples

```
# Create dummy data
library(dplyr)
set.seed(123)
dummy_data <- data.frame(
  years_education = rnorm(100, 12, 3),    # Represents years of education
  gender_female = rbinom(100, 1, 0.5),    # 1 = Female, 0 = Male
  household_wealth = sample(1:5, 100, replace = TRUE),  # Wealth index from 1 to 5
  district_code = sample(1:10, 100, replace = TRUE)     # Represents district codes
) %>% arrange(district_code)

# Define a regression formula
formula <- years_education ~ gender_female + household_wealth + household_wealth:gender_female

# Run the regression for all districts
results1 <- DHSr::Replm2(dummy_data, formula, "district_code", "normal")

# Assign random clusters for demonstration
clusters <- data.frame(
  district_code = unique(dummy_data$district_code),
  cluster_id = sample(1:3, length(unique(dummy_data$district_code)), replace = TRUE)
)

# Merge clusters with regression results
cluster_beta <- merge(clusters, results1, by.x = "district_code", by.y = "location")

# Apply Stein Beta shrinkage
results_with_stein_beta <- DHSr::stein_beta(
  data = cluster_beta,
  cluster_id = "cluster_id",                # Column for cluster IDs
  beta = "estimate_gender_female"          # Column for beta estimates
)

# View results
```

```
print(head(results_with_stein_beta))
```

Index