Package: klovan (via r-universe)

October 15, 2024

Title Geostatistics Methods and Klovan Data

Version 0.1.0

Description A comprehensive set of geostatistical, visual, and analytical methods, in conjunction with the expanded version of the acclaimed J.E. Klovan's mining dataset, are included in 'klovan'. This makes the package an excellent learning resource for Principal Component Analysis (PCA), Factor Analysis (FA), kriging, and other geostatistical techniques. Originally published in the 1976 book 'Geological Factor Analysis', the included mining dataset was assembled by Professor J. E. Klovan of the University of Calgary. Being one of the first applications of FA in the geosciences, this dataset has significant historical importance. As a well-regarded and published dataset, it is an excellent resource for demonstrating the capabilities of PCA, FA, kriging, and other geostatistical techniques in geosciences. For those interested in these methods, the 'klovan' datasets provide a valuable and illustrative resource. Note that some methods require the 'RGeostats' package. Please refer to the README or Additional_repositories for installation instructions. This material is based upon research in the Materials Data Science for Stockpile Stewardship Center of Excellence (MDS3-COE), and supported by the Department of Energy's National Nuclear Security Administration under Award Number DE-NA0004104.

License MIT + file LICENSE

Depends R (>= 2.10)

Imports MASS, dplyr, fields, GGally, ggforce, ggplot2, ggrepel, gstat, metR, pracma, sp, tibble, magrittr, tidyselect

Suggests testthat (>= 3.0.0), sf, spelling, knitr, rmarkdown, RGeostats

Additional_repositories http://rgeostats.free.fr

Config/testthat/edition 3

Encoding UTF-8

LazyData true

RoxygenNote 7.3.0

Language en-US

VignetteBuilder knitr

NeedsCompilation no

Author Jonathan E Gordon [aut] (<https://orcid.org/0009-0007-5958-7386>), Eric K Helfer [aut] (<https://orcid.org/0000-0002-8958-7690>), Hope E Omodolor [aut] (<https://orcid.org/0009-0005-7842-406X>), Jeffery M Yarus [aut] (<https://orcid.org/0000-0002-9331-9568>), Roger H French [aut, cre, cph] (<https://orcid.org/0000-0002-6162-0532>)

Maintainer Roger H French <rxf131@case.edu>

Date/Publication 2024-02-01 10:20:02 UTC

Repository https://frenchrh.r-universe.dev

RemoteUrl https://github.com/cran/klovan

RemoteRef HEAD

RemoteSha c4b866421ecf9b5e9d61d3db278d3565e4f0f27b

Contents

`calc_eigenvalues` *Calculate Eigenvalues*

Description

This function calculates the eigenvalues of a given covariance matrix or a klovan dataset. In case of a klovan dataset, it is first converted into a covariance matrix. For further details on klovan datasets, refer to the README.

Usage

```
calc_eigenvalues(data)
```

Arguments

`data`	A covariance matrix made with `covar_mtrx()` function, or a A dataset of class data.frame. The data should contain 'C_X' and 'C_Y' columns representing the x and y coordinates of the data points and excludes any rank, ID, or column not for analysis.

Value

A data frame with two columns: "Evalues_COV" and "pc.names1". "Evalues_COV" represents the eigenvalues for each principal component listed in "pc.names1".

Examples

```
data("Klovan_Row80")
calc_eigenvalues(covar_mtrx(Klovan_Row80)) # view eigenvalues
```

calc_eigenvectors *Calculate Eigenvectors*

Description

This function calculates the Eigenvectors of a given covariance matrix or a klovan dataset. In case of a klovan dataset, it is first converted into a covariance matrix. For further details on klovan datasets, refer to the README.

Usage

```
calc_eigenvectors(data)
```

Arguments

data	A covariance matrix made with `covar_mtrx()` function, or a A dataset of class data.frame. The data should contain 'C_X' and 'C_Y' columns representing the x and y coordinates of the data points and excludes any rank, ID, or column not for analysis.

Value

A data frame with two columns: "Evalues_COV" and "pc.names1". "Evalues_COV" represents the eigenvectors for each principal component listed in "pc.names1".

Examples

```
data("Klovan_Row80")
calc_eigenvectors(covar_mtrx(Klovan_Row80)) # view eigenvectors
```

cardinal_sine_model *Cardinal Sine Model*

Description

Calculate the cardinal sine model based on the given parameters.

Usage

```
cardinal_sine_model(h, sill)
```

Arguments

h	Distance.
sill	Sill value.

Value

Computed value based on the cardinal sine model.

`cauchy_model` *Cauchy Model*

Description

Calculate the Cauchy model based on the given parameters.

Usage

```
cauchy_model(h, nugget, sill, range, a)
```

Arguments

`h`	Distance.
`nugget`	Nugget effect.
`sill`	Sill value.
`range`	Range value.
`a`	Additional parameter.

Value

Computed value based on the Cauchy model.

`correlation_plot` *Correlation Plot*

Description

The correlation plot is a summary showing the relationship among variables.The plot below is a 10 x 10 table where each variable is plotted against every other variable.In the top half of the table, the correlation coefficients are displayed. In the bottom half, the scatter plots are shown along with a regression line. Down the diagonal axis, the variable histograms are show.

Usage

```
correlation_plot(data)
```

Arguments

`data`	A dataset of class data.frame. The data should contain 'C_X' and 'C_Y' columns representing the x and y coordinates of the data points and excludes any rank, ID, or column not for analysis, see README for details

Value

a correlation plot displaying correlation coefficients, the scatter plots with a regression line and, the variable histograms in a 10 x 10 table.

Examples

```
data("Klovan_Row80")
correlation_plot(Klovan_Row80)
```

`cor_mtrx` *correlation matrix*

Description

correlation matrix

Usage

```
cor_mtrx(data)
```

Arguments

`data`	A dataset of class data.frame. The data should contain 'C_X' and 'C_Y' columns representing the x and y coordinates of the data points and excludes any rank, ID, or column not for analysis.

Value

a correlation matrix as matrix object

Examples

```
data("Klovan_Row80")
cor_mtrx(range_transform(Klovan_Row80)) # view correlation matrix
corMtrx <- cor_mtrx(Klovan_Row80) # save correlation matrix as object
```

cosinus_model *Cosinus Model*

Description

Calculate the cosinus model based on the given parameters.

Usage

```
cosinus_model(h, sill, a)
```

Arguments

h	Distance.
sill	Sill value.
a	Additional parameter.

Value

Computed value based on the cosinus model.

covar_mtrx *Create Co-variance Matrix*

Description

This function creates a non-normalized co-variance matrix from the given klovan dataset. For further details on klovan datasets, refer to the README.

Usage

```
covar_mtrx(data)
```

Arguments

data	A dataset of class data.frame. The data should contain 'C_X' and 'C_Y' columns representing the x and y coordinates of the data points and excludes any rank, ID, or column not for analysis.

Value

A non-normalized co-variance matrix of the klovan data.

Examples

```
data("Klovan_Row80")
covar_mtrx(Klovan_Row80) # view co-variance matrix
```

eigen_contribution *Calculate Eigen Contribution*

Description

This function calculates the sum of all the eigenvalues from a provided covariance matrix or klovan dataset. Each eigenvalue is divided by the sum of the eigenvalues to determine its proportional contribution. This yields the percent contribution of each eigenvalue and provides an understanding of the proportion of total variance explained by each eigenvalue.

Usage

```
eigen_contribution(Cov_Mtrx)
```

Arguments

Cov_Mtrx	A covariance matrix used to compute the eigenvalues or A dataset of class data.frame. The data should contain 'C_X' and 'C_Y' columns representing the x and y coordinates of the data points and excludes any rank, ID, or column not for analysis, see README for details

Value

A data frame with columns: "EigenValues", "CumSum", "CumSumPct", "pc.names". Where: - "EigenValues": The eigenvalues - "CumSum": The cumulative sum of the eigenvalues - "CumSumPct": The proportional contribution of each eigenvalue - "pc.names": The principal component names

Examples

```
data("Klovan_Row80")
your_cov_Mtrx <- covar_mtrx(Klovan_Row80) # example covariance matrix
eigen_contribution(Klovan_Row80) # view the data frame
eigen_contribution(your_cov_Mtrx) # view the data frame
eigen_contribution(covar_mtrx(Klovan_Row80)) # view the data frame
```

exponential_model *Exponential Model*

Description

Calculate the exponential model based on the given parameters.

Usage

```
exponential_model(h, nugget, sill, range)
```

Arguments

h	Distance.
nugget	Nugget effect.
sill	Sill value.
range	Range value.

Value

Computed value based on the exponential model.

---

factor_analysis *Perform Factor Analysis*

Description

This function performs a Factor Analysis on a provided dataset using the "Varimax" orthogonal rotation method. It also calculates the factor scores for each factor.

Usage

```
factor_analysis(data, num_fac = 3)
```

Arguments

data	A dataset of class data.frame. The data should contain 'C_X' and 'C_Y' columns representing the x and y coordinates of the data points and excludes any rank, ID, or column not for analysis, see README for details
num_fac	A numeric value indicating the number of factors to analyze. It's recommended to use 3, which is also the default value.

Value

A data frame containing the calculated factors.

Examples

```
data("Klovan_Row80")
factor_analysis(Klovan_Row80)
factor_analysis(Klovan_Row80, 3)
```

`factor_cor_plot` *factor correlation plot*

Description

factor correlation plot

Usage

```
factor_cor_plot(
  data,
  FAC_1,
  FAC_2,
  num_fac = 3,
  text_col = "red",
  line_col = "lightblue"
)
```

Arguments

`data`	A dataset of class data.frame. The data should contain 'C_X' and 'C_Y' columns representing the x and y coordinates of the data points and excludes any rank, ID, or column not for analysis, see README for details. It will be that will be converted into a plottable dataframe, see README for details or a plottable data frame created from the `factor_analysis()` function
`FAC_1`	the first factor to be compared. A string that can be chosen from FA1:FA3 or FA1:FAnum_fac e.g. "FA1"
`FAC_2`	the first factor to be compared. A string that can be chosen from FA1:FA3 or FA1:FAnum_fac e.g. "FA2"
`num_fac`	a numeric value for how many factors to analyze. Recommended to use 3 and default to 3.
`text_col`	an R color, the color of the text lables, defaults to "red"
`line_col`	an R color, the color of the lines, defaults to "lightblue"

Value

a ggplot object of the correlation plot

Examples

```
data("Klovan_Row80")
fa_plot1 <- factor_cor_plot(Klovan_Row80, "FAC1", "FAC2", 2)
fa_plot1

factor_cor_plot(Klovan_Row80, "FAC1", "FAC3")
```

```
fa_plot2 <-factor_cor_plot(factor_analysis(Klovan_Row80), "FAC1", "FAC3", 4)
fa_plot2
```

`factor_score_plot` *Map the Factor Scores*

Description

This function creates a faceted plot of each rotated factor score, which could be interpreted as the elements of a "phantom" variable. This function aids in defining the relationship between the phantom variables and the known ore body by producing a contoured map for each variable.

Usage

```
factor_score_plot(
  Interp_Data,
  overlay,
  data,
  FA_colors = c(FA1 = "black", FA2 = "blue", FA3 = "darkred", FA4 = "green", FA5 =
    "purple", FA6 = "orange", FA7 = "yellow", FA8 = "pink", FA9 = "cyan", FA10 =
    "magenta")
)
```

Arguments

`Interp_Data`	A plottable data frame produced by the `inv_dis_wt()` or `kriging()` functions.
`overlay`	A Boolean input. If TRUE, the plot will overlay isolines; if FALSE, it will not.
`data`	a klovan dataset (transformed, untransformed, outlier, etc), see README for details.
`FA_colors`	A named vector of colors for different factors. Defaults to a set color palette.

Value

A ggplot object representing the Factor Scores plot.

Examples

```
data("Klovan_Row80")
factor_plot1 <- factor_score_plot(inv_dis_wt(Klovan_Row80), TRUE, data = Klovan_Row80)
factor_plot1

your_interp_data_IDW <- inv_dis_wt(Klovan_Row80, 3)
factor_score_plot(your_interp_data_IDW, FALSE, data = Klovan_Row80)
```

gamma_model *Gamma Model*

Description

Calculate the gamma model based on the given parameters.

Usage

```
gamma_model(h, nugget, sill, range, a)
```

Arguments

h	Distance.
nugget	Nugget effect.
sill	Sill value.
range	Range value.
a	Additional parameter.

Value

Computed value based on the gamma model.

gaussian_model *Gaussian Model*

Description

Calculate the Gaussian model based on the given parameters.

Usage

```
gaussian_model(h, nugget, sill, range)
```

Arguments

h	Distance.
nugget	Nugget effect.
sill	Sill value.
range	Range value.

Value

Computed value based on the Gaussian model.

inv_dis_wt *Inverse Distance Weighting*

Description

This function applies the Inverse Distance Weighting interpolation algorithm

Usage

```
inv_dis_wt(data, num_fac = 3)
```

Arguments

data	A dataset of class data.frame. The data should contain 'C_X' and 'C_Y' columns representing the x and y coordinates of the data points and excludes any rank, ID, or column not for analysis, see README for details
num_fac	a numeric value for how many factors to analyze. Recommended to use 3 and default to 3.

Value

a data frame with interpolated data for the whole graph. Data frame has collumns: "C_X" "C_Y" "value" "FA": C_X, C_Y are coordinates and "value" is the value for the "FA" the relevant factor.

Examples

```
data("Klovan_Row80")
inv_dis_wt(Klovan_Row80, 4)
inv_dis_wt(Klovan_Row80, 3)
```

Klovan_2D_all_outlier *Klovan mining dataset*

Description

The klovan outlier dataset provides exploration data for geostatistics and analytical methods with an erroneous data point. The region has undergone extensive geological investigation, revealing the position of a known ore body. The erroneous allows for investigation into how outliers effect the packages analytical methods.

Usage

```
Klovan_2D_all_outlier
```

Format

A data frame with observations of various geological variables @item rank: Rank of the observation @item C_X: X-coordinate of the observation @item C_Y: Y-coordinate of the observation @item P_Elong: Elongation of the geological feature @item P_Fe: Iron content in the geological feature @item P_Fold: Folding factor of the geological feature @item P_Fract: Fracturing level of the geological feature @item P_Mg: Magnesium content in the geological feature @item P_Na: Sodium content in the geological feature @item P_Space: Spatial arrangement factor of the geological feature @item P_Sulfide: Sulfide content in the geological feature @item P_Veins: Veining factor of the geological feature @item P_XLSize: Size of the geological feature

---

Klovan_Row80 *Klovan mining dataset*

Description

The klovan dataset provides exploration data for geostatistics and analytical methods. The region has undergone extensive geological investigation, revealing the position of a known ore body.

Usage

```
Klovan_Row80
```

Format

A data frame with observations of various geological variables @item rank: Rank of the observation @item C_X: X-coordinate of the observation @item C_Y: Y-coordinate of the observation @item P_Elong: Elongation of the geological feature @item P_Fe: Iron content in the geological feature @item P_Fold: Folding factor of the geological feature @item P_Fract: Fracturing level of the geological feature @item P_Mg: Magnesium content in the geological feature @item P_Na: Sodium content in the geological feature @item P_Space: Spatial arrangement factor of the geological feature @item P_Sulfide: Sulfide content in the geological feature @item P_Veins: Veining factor of the geological feature @item P_XLSize: Size of the geological feature

---

kriging *Perform Kriging Interpolation*

Description

This function performs kriging interpolation on spatial data using ridge regression to calculate the kriging weights. It uses either regular inverse or generalized inverse with ridge regression based on the availability of regular inverse for the given covariance matrix.

Usage

```
kriging(
  data,
  factor,
  grid_cell_size = NA,
  nugget,
  sill,
  range_val,
  a,
  model_name
)
```

Arguments

data	A dataset of class data.frame. The data should contain 'C_X' and 'C_Y' columns representing the x and y coordinates of the data points and excludes any rank, ID, or column not for analysis, see README for details
factor	The target factor (FAC) to be interpolated using kriging.
grid_cell_size	The desired cell size for the grid. Default is NA, which will calculate the cell size based on the average distance between data points.
nugget	The nugget effect parameter for the variogram model.
sill	The sill parameter for the variogram model.
range_val	The range parameter for the variogram model.
a	Additional parameter (depends on the variogram model) use NA if not needed.
model_name	The name of the model to use for variogram fitting and kriging. Options include "Sph1", "Exp1", "Gau1", "Mat1", "Pow1", "Quad1", "Card1", "Gam1", "Cau1", "Sta1", "Ord1", "Tri1", and "Cos1". use functionprint_model_names() for more information

Value

A data frame containing the interpolated values for the target factor (FAC).

Examples

```
data(Klovan_Row80)
# Perform kriging interpolation for FAC1
kriging_results <- kriging(Klovan_Row80, factor = 1, grid_cell_size = NA,
nugget=.0001, sill=2.5, range_val=1000, a=NA, model_name="Sph1")
```

`kriging.auto` *Automatic Kriging Interpolation with Factor Analysis Preprocessing*

Description

This function performs automatic kriging interpolation with factor analysis preprocessing on input data. The optimization may not work as intended use higher num_init_test and num_fin_test values or run the function multiples times to ensure an accurate result.

Usage

```
kriging.auto(
  data,
  num_fac = 3,
  grid_cell_size = NA,
  num_init_test = 8,
  num_fin_test = 200,
  nugget_bounds = c(0, 0.2),
  sill_bounds = c(0, 20000),
  range_bounds = c(0, 25000)
)
```

Arguments

`data`	A dataset of class data.frame. The data should contain 'C_X' and 'C_Y' columns representing the x and y coordinates of the data points and excludes any rank, ID, or column not for analysis.
`num_fac`	A numeric value indicating the number of factors to analyze. Default is 3.
`grid_cell_size`	The desired cell size for the grid. Default is NA, which will calculate the cell size based on the average distance between data points.
`num_init_test`	The number of random starts for initial model optimization. Default is 8
`num_fin_test`	The number of random starts for final model optimization. Default is 200
`nugget_bounds`	A numeric vector specifying the lower and upper bounds for the nugget parameter during optimization. Default is c(0, .2).
`sill_bounds`	A numeric vector specifying the lower and upper bounds for the sill parameter during optimization. Default is c(0, 20000).
`range_bounds`	A numeric vector specifying the lower and upper bounds for the range parameter during optimization. Default is c(0, 25000).

Value

A data frame with interpolated data for the whole grid. Data frame has columns: "C_X", "C_Y", "value", "FA". "C_X" and "C_Y" are the coordinates, "value" is the interpolated value, and "FA" indicates the relevant factor the value corresponds to.

Examples

```
data("Klovan_Row80")
kriging.auto(Klovan_Row80)
```

`manage_outliers` *Outlier Management*

Description

This function appends a new column to the input data, marking potential outliers. Once identified, these outliers can either be removed or imputed.

Usage

```
manage_outliers(data, property)
```

Arguments

`data`	A dataset of class data.frame.
`property`	A string representing the property on which the range transformation is based.

Value

The input dataset, supplemented with a new Boolean column. TRUE signifies a high likelihood of an outlier, while FALSE signifies a low likelihood.

Examples

```
data("Klovan_2D_all_outlier")
manage_outliers(Klovan_2D_all_outlier, "P_Mg")
```

`matern_model` *Matern Model*

Description

Calculate the Matern model based on the given parameters.

Usage

```
matern_model(h, nugget, sill, range, a)
```

Arguments

h	Distance.
nugget	Nugget effect.
sill	Sill value.
range	Range value.
a	Shape parameter.

Value

Computed value based on the Matern model.

---

order_1_gc_model *Order-1 G.C. (General Covariance) Model*

Description

Calculate the order-1 G.C. model based on the given parameters.

Usage

```
order_1_gc_model(h, nugget, sill, range)
```

Arguments

h	Distance.
nugget	Nugget effect.
sill	Sill value.
range	Range value.

Value

Computed value based on the order-1 G.C. model.

pc_cor_plot *Principal Component Correlation Plot*

Description

This function generates a correlation plot, also known as a "circle" plot, which compares the loadings from one principal component (PC) against another. It visualizes the similarity among original variables and their correlation with each PC, revealing potential clusters. The function also adds annotations for understanding positive and negative values in different quadrants.

Usage

```
pc_cor_plot(data, PC_1, PC_2, text_col = "red")
```

Arguments

data	A dataset of class data.frame. The data should contain 'C_X' and 'C_Y' columns representing the x and y coordinates of the data points and excludes any rank, ID, or column not for analysis, see README for details
PC_1	A string specifying the first PC for comparison, can be chosen from "PC1" to "PC10". For example, "PC1".
PC_2	A string specifying the second PC for comparison, can be chosen from "PC1" to "PC10". For example, "PC2".
text_col	An R color for the text labels. Defaults to "red".

Value

A ggplot object representing the correlation plot.

Examples

```
data("Klovan_Row80")
pc_cor_plot(Klovan_Row80, "PC1", "PC2")
```

power_model *Power Model*

Description

Calculate the power model based on the given parameters.

Usage

```
power_model(h, nugget, sill, a)
```

Arguments

h	Distance.
nugget	Nugget effect.
sill	Sill value.
a	Power parameter.

Value

Computed value based on the power model.

---

print_model_names *Print Model Names*

Description

This function prints the names of the predefined model functions.

Usage

```
print_model_names()
```

Value

NULL (This function is used for printing the model names only.)

Examples

```
print_model_names()
```

---

quadratic_exponential_model *Quadratic Exponential Model*

Description

Calculate the quadratic exponential model based on the given parameters.

Usage

```
quadratic_exponential_model(h, nugget, sill, range, a)
```

Arguments

h	Distance.
nugget	Nugget effect.
sill	Sill value.
range	Range value.
a	Additional parameter.

Value

Computed value based on the quadratic exponential model.

---

range_transform *range transform*

Description

Normalize the data using a 'Range' transform . In the returned data table, note that in each column of the normalized Data Table, the variables will range from 0 to 1.

Usage

```
range_transform(data)
```

Arguments

data	A dataset of class data.frame. The data should contain 'C_X' and 'C_Y' columns representing the x and y coordinates of the data points and excludes any rank, ID, or column not for analysis, see README for details

Value

a range transformed version of a klovan dataset.

Examples

```
data("Klovan_Row80")
T_Klovan <- range_transform(Klovan_Row80)
```

Rgeo_database *Database Construction*

Description

Constructs a database from a provided dataset using specified factors. For more details on the dataset format, see the package README.

Usage

```
Rgeo_database(data, num_fac = 3, property)
```

Arguments

data	A dataset of class data.frame. The data should contain 'C_X' and 'C_Y' columns representing the x and y coordinates of the data points and excludes any rank, ID, or column not for analysis, see README for details
num_fac	A numeric value indicating the number of factors to analyze. Default is 3.
property	A string indicating which factor to build variogram from e.g. "RC1" or "RC2"

Details

The Rgeo_database function constructs a db-class object from the provided dataset using the number of factors specified by num_fac and made for use with property.

Value

A db-class object containing the factors selected with num_fac and made for use with property.

Examples

```
if(requireNamespace("RGeostats", quietly = TRUE)){
    library(RGeostats)
    data("Klovan_Row80", package = "klovan")
    Rgeo_database(Klovan_Row80, 3, "RC3")
}
```

Rgeo_kriging *Kriging Interpolation for Databases*

Description

Performs kriging interpolation on a provided database using 'RGeostats' methods. The data is interpolated over a grid covering the entire area of interest.

Usage

```
Rgeo_kriging(db, model, dx = NA, dy = NA)
```

Arguments

db	A db-class object. Should be created using the `Rgeo_database()` function.
model	An S4 plottable Rgeostats omnidirectional variogram model. Should be created using the `Rgeo_vario_model()` function.
dx	Optional. The grid cell size in the x-direction. If not provided, it is calculated as the average of the ranges in x and y directions divided by 50.
dy	Optional. The grid cell size in the y-direction. If not provided, it is calculated as the average of the ranges in x and y directions divided by 50.

Details

The `Rgeo_kriging()` function performs kriging interpolation based on the provided database (db) and variogram model (model). The grid cell sizes dx and dy can be optionally specified or will be automatically determined based on the data. Results can be visualized with the `Rgeo_kriging_plot()` function and summary statistics can be printed by simply calling the returned kriged object.

Value

A S4 plottable Rgeostats kriged database. Can be plotted using the `Rgeo_kriging_plot` function. Summary statistics for the kriging process can be printed by simply calling the returned dbgrid3 object.

Examples

```
if(requireNamespace("RGeostats", quietly = TRUE)){
    library(RGeostats)
    data("Klovan_Row80", package = "klovan")
    db <- Rgeo_database(Klovan_Row80, 3, "RC3")
    model <- Rgeo_vario_model(db, 3, "RC3", lag = 500, model = 13)
    krig <- Rgeo_kriging(db, model)
    krig # prints summary statistics for kriging
}
```

Rgeo_kriging_plot *Plotting Kriged Database*

Description

This function creates a plot of the kriged database. The function is designed specifically for visualizing geostatistical data.

Usage

```
Rgeo_kriging_plot(krig_db, db, property)
```

Arguments

krig_db	A kriged database object, resulting from the Rgeo_kriging function.
db	The original database object that was used to generate the kriged database.
property	A character string representing the property (or column name) from the database that you want to visualize e.g. "RC1" or "RC2".

Details

The function takes a kriged database and the original database, then generates a comparative plot for a specific property. This helps in understanding the effect of kriging on the selected property.

Value

A plot comparing the specified property in the original and kriged databases.

Examples

```
if(requireNamespace("RGeostats", quietly = TRUE)){
    library(RGeostats)
    data("Klovan_Row80", package = "klovan")
    db <- Rgeo_database(Klovan_Row80, 3, "RC3")
    model <- Rgeo_vario_model(db, 3, "RC3", lag = 500, model = 13)
    krig_db <- Rgeo_kriging(db, model)
    Rgeo_kriging_plot(krig_db, db, "RC3")
}
```

Rgeo_vario_construct_plot *Experimental Ominidirectional Variogram Plot using Regostats*

Description

This function constructs an Experimental Ominidirectional Variogram using 'Rgeostats'.

Usage

```
Rgeo_vario_construct_plot(db, num_fac, property, lag, nlag = 10)
```

Arguments

db	The db-class containing the data information used to calculate the experimental variogram. The variograms are calculated for the set of "z*" variables present in the db.
num_fac	A numeric value indicating how many factors to analyze. Default is 3.
property	A string indicating which factor to build variogram from e.g. "RC1" or "RC2"
lag	Array containing the distance lags for each calculation direction. If the lag is not defined, set as NA. A default lag is calculated so that the maximum distance is equal to half of the field diagonal
nlag	Array containing the number of lags for each calculation direction If nlag not defined, set as NA. If the number of lags is not defined, it defaults to 10.

Value

a plottable Rgeostats Experimental Ominidirectional Variogram model

Examples

```
if(requireNamespace("RGeostats")){
    library(RGeostats)
    data("Klovan_Row80", package = "klovan")
    db <- Rgeo_database(Klovan_Row80, 3, "RC3")
    Rgeo_vario_construct_plot(db, 3, "RC3", lag = 500)
}
```

Rgeo_vario_model *Ominidirectional Variogram Model using Rgeostats*

Description

This function calculates an omnidirectional variogram model for a given dataset using 'RGeostats' package. The variogram model can be useful for understanding the spatial dependence structure of the data. The function also allows the user to choose the number of factors to analyze, the property to build the variogram from, and the type of model to use for the variogram.

Usage

```
Rgeo_vario_model(db, num_fac, property, lag, nlag = 10, model)
```

Arguments

db	A db-class object. This is the dataset used to calculate the experimental variogram. The variogram is calculated for the set of "z*" variables present in the db.
num_fac	A numeric value indicating how many factors to analyze. This helps to limit the scope of the analysis to a specific number of factors. Default is 3.
property	A string indicating which factor (or property) to build the variogram from. For example, it can be "RC1" or "RC2".
lag	A numeric value or an array containing the distance lags for each calculation direction. If the lag is not defined, set as NA. A default lag is calculated so that the maximum distance is equal to half of the field diagonal.
nlag	A numeric value or an array containing the number of lags for each calculation direction. If nlag is not defined, set it as NA. If the number of lags is not defined, it defaults to 10.
model	A numeric value indicating what type of model to use in the variogram. This parameter corresponds to the model types provided by the RGeostats package. Run the line 'melem.name()' in RGeostats to see the number corresponding to each model.

Value

An object of class 'model'. This is a plottable Rgeostats omnidirectional variogram model. It can be used for further geostatistical analysis or for visualizing the spatial structure of the data.

Examples

```
if(requireNamespace("RGeostats")){
    library(RGeostats)
    data("Klovan_Row80", package = "klovan")
    db <- Rgeo_database(Klovan_Row80, 3, "RC3")
    model <- Rgeo_vario_model(db, 3, "RC3", lag = 500, model = 13)
}
```

scree_plot *scree plot*

Description

scree plot

Usage

```
scree_plot(
  EigenPlot,
  bar_fill = "lightblue",
  outline = "darkblue",
  eigen_line = "red",
  cum_eigen_line = "blue"
)
```

Arguments

EigenPlot	A dataset of class data.frame. The data should contain 'C_X' and 'C_Y' columns representing the x and y coordinates of the data points and excludes any rank, ID, or column not for analysis, see README for details or, a covariance matrix that will be converted into a plottable data frame or, a plottable data frame created by the `eigen_contribution()` function
bar_fill	an R color, The fill color for the bars, defaults to "lightblue"
outline	an R color, the outline color of the bars, defaults to "darkblue"
eigen_line	an R color, the color of the eigenvalues line, defaults to "red"
cum_eigen_line	an R color, the color of the cummulative eigenvalues line, defaults to "blue"

Value

a ggplot object of the scree plot

Examples

```
data("Klovan_Row80")
scree_plot(eigen_contribution(covar_mtrx(Klovan_Row80)))


Scree1 <- scree_plot(Klovan_Row80)
Scree1

your_eigen_data1 <- eigen_contribution(Klovan_Row80)
scree_plot(your_eigen_data1)
```

spherical_model *Spherical Model*

Description

Calculate the spherical model based on the given parameters.

Usage

```
spherical_model(h, nugget, sill, range)
```

Arguments

h	Distance.
nugget	Nugget effect.
sill	Sill value.
range	Range value.

Value

Computed value based on the spherical model.

stable_model *Stable Model*

Description

Calculate the stable model based on the given parameters.

Usage

```
stable_model(h, nugget, sill, range, a)
```

Arguments

h	Distance.
nugget	Nugget effect.
sill	Sill value.
range	Range value.
a	Additional parameter.

Value

Computed value based on the stable model.

triangle_model *Triangle Model*

Description

Calculate the triangle model based on the given parameters.

Usage

```
triangle_model(h, nugget, sill, range)
```

Arguments

h	Distance.
nugget	Nugget effect.
sill	Sill value.
range	Range value.

Value

Computed value based on the triangle model.

vario_plot *Create Variogram Plot*

Description

This function calculates the empirical variogram for a given target factor (FAC) and plots it along with the fitted variogram based on the specified variogram model.

Usage

```
vario_plot(
  data,
  factor,
  nlags = NA,
  lags = NA,
  nugget,
  sill,
  range_val,
  a,
  model_name
)
```

Arguments

data	A dataset of class data.frame. The data should contain 'C_X' and 'C_Y' columns representing the x and y coordinates of the data points and excludes any rank, ID, or column not for analysis, see README for details
factor	The target factor (FAC) to be used for variogram calculation and fitting.
nlags	The number of lag bins for variogram calculation. Default is NA, which will use Sturges' formula to determine the number of lags.
lags	The lag width for variogram calculation. Default is NA, which will calculate the lag width based on the range of distances.
nugget	The nugget effect parameter for the variogram model.
sill	The sill parameter for the variogram model.
range_val	The range parameter for the variogram model.
a	Additional parameter (depends on the variogram model) use NA if not needed.
model_name	The name of the model to use for variogram fitting. Available options include "Sph1", "Exp1", "Gau1", "Mat1", "Pow1", "Quad1", "Card1", "Gam1", "Cau1", "Sta1", "Ord1", "Tri1", and "Cos1". Use function `print_model_names()` for more information.

Value

A plot displaying the empirical variogram and the fitted variogram based on the specified model.

Examples

```
data(Klovan_Row80)
# Plot variogram for FAC1
vario_plot(Klovan_Row80, factor = 1, nlags = 10, nugget = 0.01, sill = 2.5,
range_val = 1000, a = NA, model_name = "Sph1")
```

Index