Package ‘TSCS’

January 20, 2025

Version 0.1.1

Title Time Series Cointegrated System

Maintainer Tianjian Yang <yangtj5@mail2.sysu.edu.cn>

Description A set of functions to implement Time Series Cointegrated System (TSCS) spatial interpolation and relevant data visualization.

Depends R (>= 3.4.2)

Imports stats, ggplot2 (>= 2.2.1), tseries (>= 0.10-42), rgl (>= 0.98.1), grDevices

License GPL (>= 2.0)

RoxygenNote 6.0.1

Encoding UTF-8

LazyData true

Suggests knitr, rmarkdown, R.rsp

VignetteBuilder knitr, R.rsp

NeedsCompilation no

Author Tianjian Yang [aut, cre]

Repository CRAN

Date/Publication 2017-10-02 11:19:48 UTC

Contents

`appraisal_index` *Compute Appraisal Index of Interpolation/Prediction Result*

Description

Two appraisal indexes used for evaluating the result of interpolation/prediction - RMSE and standard deviation of error.

Usage

```
appraisal_index(est, true)
```

Arguments

`est`	a numeric vector; estimations.
`true`	a numeric vector; true values.

Details

- The first appraisal index is RMSE, abbr. of root-mean-square error. It is used for measuring the differences between estimated values by a method and the values actually observed. Smaller RMSE means more accurate interpolation/prediction.
- The second appraisal index is standard deviation of error, which is used for measuring how far the errors are spread out from their mean, namely, stability of errors. Smaller value means greater stability of errors, suggesting that errors would not fluctuate heavily due to difference of data.

Value

A list of 2 is returned, including:

`RMSE` numeric; RMSE.

`std` numeric; standard deviation of error.

See Also

`plot_compare`

Examples

```
## Not run:

## TSCS spatial interpolation procedure:

basis <- tscsRegression(data = data, h = 1, v = 1, alpha = 0.01); # regression
basis$percentage # see the percentage of cointegrated relationships
est <- tscsEstimate(matrix = basis$coef_matrix, newdata = newdata, h = 1, v = 1); # estimation
str(est)

## comparison of estimates and true values:

plot_compare(est = est$estimate[,3], true = true) # graphic comparison
index <- appraisal_index(est = est$estimate[,3], true = true); # RMSE & std
index

## data visualization:

plot_dif(data = data[,1:2], h = 1, v = 1) # differentiate boundary and interior spatial locations
plot_NA(newdata = newdata) # show spatial locations with missing value, for a cross-section data
plot_map(newdata = newdata) # plot the 2D spatial map, for a cross-section data

## End(Not run)
```

`plot3D_dif` *Plot Interior Spatial Locations and System Boundary - 3D Map*

Description

`plot3D_dif` differentiates boundary and interior spatial locations in a spatial domain (a collection of spatial locations with their coordinates). Since TSCS method is only capable of interpolation but not extrapolation, it is necessary to highlight the difference between interior spatial locations and system boundary.

Usage

```
plot3D_dif(coords, h1, h2, v, xlab = NULL, ylab = NULL, zlab = NULL,
  title = NULL, cex = 3)
```

Arguments

`coords`	data frame; should only contain the three variables: X coordinate, Y coordinate and Z coordinate. Each row uniquely denotes a spatial location. (coordinates must be numeric)
`h1`	numeric; side length of the unit cubic grid in X coordinate direction (horizontal).
`h2`	numeric; side length of the unit cubic grid in Y coordinate direction (horizontal).
`v`	numeric; side length of the unit cubic grid in Z coordinate direction (vertical).

`xlab`	a label for the x axis, defaults to the name of X coordinate.
`ylab`	a label for the y axis, defaults to the name of Y coordinate.
`zlab`	a label for the z axis, defaults to the name of Z coordinate.
`title`	a main title for the plot.
`cex`	numeric; size of point to be plotted for each spatial location. (default: 3)

Details

- The resulting plot is interactive, where the red points are interior spatial locations while the black points denote system boundary.
- `plot3D_dif` is exclusive to 3D rectangular grid system. Similarly, if you want to fathom how this package handles 2D rectangular grid system, please refer to `plot_dif`.

See Also

`plot_dif`, `plot3D_NA`, `plot3D_map`

Examples

```
## Not run:

## TSCS spatial interpolation procedure:

basis <- tscsRegression3D(data = data, h1 = 3.75, h2 = 2.5, v = 5, alpha = 0.01);
basis$percentage
est <- tscsEstimate3D(matrix = basis$coef_matrix, newdata = newdata, h1 = 3.75, h2 = 2.5, v = 5);
str(est)

## comparison of estimates and true values:

plot_compare(est = est$estimate[,4], true = true)
index <- appraisal_index(est = est$estimate[,4], true = true);
index

## data visualization:

plot3D_dif(data = data[,1:3], h1 = 3.75, h2 = 2.5, v = 5)
plot3D_NA(newdata = newdata)
plot3D_map(newdata = newdata)

## End(Not run)
```

plot3D_map *Visualize Spatial(Cross-Section) Data of a Given Time Point - 3D Map*

Description

`plot_map` draws a three-dimensional spatial map. It is plotted based on the cross-section data of a given time point, which is also often extracted from spatio-temporal data.

Usage

```
plot3D_map(newdata, xlab = NULL, ylab = NULL, zlab = NULL, title = NULL,
  cex = 9, colorNA = "white")
```

Arguments

`newdata`	data frame; should only contain the four variables in order: X coordinate, Y coordinate, Z coordinate and observation. This is the cross-section data or pure spatial data of a particular time point you have selected, with missing observations that you want to predict. (coordinates must be numeric)
`xlab`	a label for the x axis, defaults to the name of X coordinate.
`ylab`	a label for the y axis, defaults to the name of Y coordinate.
`zlab`	a label for the z axis, defaults to the name of Z coordinate.
`title`	a main title for the plot.
`cex`	numeric; size of plotting point for each spatial locations. (default: 9)
`colorNA`	colour for missing values/observations. (default: "white")

Details

- The resulting plot is interactive.
- `plot3D_map` is exclusive to 3D rectangular grid system. Similarly, if you want to fathom how this package handles 2D rectangular grid system, please refer to `plot_map`.

See Also

`plot_map`, `plot3D_NA`, `plot3D_dif`

Examples

```
## Not run:

## TSCS spatial interpolation procedure:

basis <- tscsRegression3D(data = data, h1 = 3.75, h2 = 2.5, v = 5, alpha = 0.01);
basis$percentage
est <- tscsEstimate3D(matrix = basis$coef_matrix, newdata = newdata, h1 = 3.75, h2 = 2.5, v = 5);
str(est)

## comparison of estimates and true values:

plot_compare(est = est$estimate[,4], true = true)
index <- appraisal_index(est = est$estimate[,4], true = true);
index

## data visualization:

plot3D_dif(data = data[,1:3], h1 = 3.75, h2 = 2.5, v = 5)
plot3D_NA(newdata = newdata)
```

```
plot3D_map(newdata = newdata)

## End(Not run)
```

---

plot3D_NA *Visualize the Spatial Distribution of Missing Observations - 3D Map*

Description

plot3D_NA shows spatial locations with or without missing observation. It is plotted based on the cross-section data of a given time point, which is also often extracted from spatio-temporal data.

Usage

```
plot3D_NA(newdata, xlab = NULL, ylab = NULL, zlab = NULL, title = NULL,
  cex = 3, color = "orange", colorNA = "blue")
```

Arguments

newdata	data frame; should only contain the four variables in order: X coordinate, Y coordinate, Z coordinate and observation. This is the cross-section data or pure spatial data of a particular time point you have selected, with missing observations that you want to predict. (coordinates must be numeric)
xlab	a label for the x axis, defaults to the name of X coordinate.
ylab	a label for the y axis, defaults to the name of Y coordinate.
zlab	a label for the z axis, defaults to the name of Z coordinate.
title	a main title for the plot.
cex	numeric; size of plotting point for each spatial location. (default: 3)
color	colour to be used to fill the spatial locations. (default: "orange")
colorNA	colour for denoting missing values/observations. (default: "blue")

Details

- The resulting plot is interactive.
- plot3D_NA is exclusive to 3D rectangular grid system. Similarly, if you want to fathom how this package handles 2D rectangular grid system, please refer to plot_NA.

See Also

plot_NA, plot3D_map, plot3D_dif

Examples

```
## Not run:

## TSCS spatial interpolation procedure:

basis <- tscsRegression3D(data = data, h1 = 3.75, h2 = 2.5, v = 5, alpha = 0.01);
basis$percentage
est <- tscsEstimate3D(matrix = basis$coef_matrix, newdata = newdata, h1 = 3.75, h2 = 2.5, v = 5);
str(est)

## comparison of estimates and true values:

plot_compare(est = est$estimate[,4], true = true)
index <- appraisal_index(est = est$estimate[,4], true = true);
index

## data visualization:

plot3D_dif(data = data[,1:3], h1 = 3.75, h2 = 2.5, v = 5)
plot3D_NA(newdata = newdata)
plot3D_map(newdata = newdata)

## End(Not run)
```

`plot_compare` *Graphic Comparison Between Estimates and True Values*

Description

Provided that you have the true values of missing observations, you can compare them with the results of interpolation. `plot_compare` visualizes the comparison between estimates and true values. (NB: this plotting function can also be used in other similar situations involving comparison between estimates and true values.)

Usage

```
plot_compare(est, true, cex = 1, width = 1, P = 6/7, AI = TRUE)
```

Arguments

`est`	a numeric vector; estimations.
`true`	a numeric vector; true values.
`cex`	numeric; size of point to be plotted. (default: 1)
`width`	numeric; width of fitted straight line. (default: 1)
`P`	numeric, between 0 and 1; position for superimposing values of appraisal indexes. (default: 6/7)
`AI`	logical; `TRUE` for presenting appraisal indexes while `FALSE` for not. (default: TRUE)

Details

Attentions:

- The values in `est` and `true` vectors should be arranged in the same order, in correspondence with the sequence of observations.
- If the maximum value of either `est` or `true` is greater than 1000, or the minimum is smaller than -1000, please make appropriate transformation that limits your data to bound [-1000,1000].

In the plot:

- The big red point is the origin.
- The red line stands for straight line `y = x`.
- The blue line stands for fitted straight line.

See Also

`appraisal_index`

Examples

```
## Not run:

## TSCS spatial interpolation procedure:

basis <- tscsRegression(data = data, h = 1, v = 1, alpha = 0.01) # regression
basis$percentage # see the percentage of cointegrated relationships
est <- tscsEstimate(matrix = basis$coef_matrix, newdata = newdata, h = 1, v = 1) # estimation
str(est)

## comparison of estimates and true values:

plot_compare(est = est$estimate[,3], true = true) # graphic comparison
index <- appraisal_index(est = est$estimate[,3], true = true); # RMSE & std
index

## data visualization:

plot_dif(data = data[,1:2], h = 1, v = 1) # differentiate boundary and interior spatial locations
plot_NA(newdata = newdata) # show spatial locations with missing value, for a cross-section data
plot_map(newdata = newdata) # plot the 2D spatial map, for a cross-section data

## End(Not run)
```

plot_dif — *Plot Interior Spatial Locations and System Boundary - 2D Map*

Description

`plot_dif` differentiates boundary and interior spatial locations in a spatial domain (a collection of spatial locations with their coordinates). Since TSCS method is only capable of interpolation but not extrapolation, it is necessary to highlight the difference between interior spatial locations and system boundary.

Usage

```
plot_dif(coords, h, v, xlab = NULL, ylab = NULL, title = NULL, cex = 1)
```

Arguments

`coords`	data frame; should only contain the two variables: X coordinate and Y coordinate. Each row uniquely denotes a spatial location. (coordinates must be numeric)
`h`	numeric; side length of the unit grid in X coordinate direction.
`v`	numeric; side length of the unit grid in Y coordinate direction.
`xlab`	a label for the x axis, defaults to the name of X coordinate.
`ylab`	a label for the y axis, defaults to the name of Y coordinate.
`title`	a main title for the plot.
`cex`	numeric; size of plotting point for each spatial location. (default: 1)

Details

`plot_dif` is exclusive to 2D rectangular grid system. Similarly, if you want to fathom how this package handles 3D rectangular grid system, please refer to `plot3D_dif`.

See Also

`plot3D_dif`, `plot_NA`, `plot_map`

Examples

```
## Not run:

## TSCS spatial interpolation procedure:

basis <- tscsRegression(data = data, h = 1, v = 1, alpha = 0.01); # regression
basis$percentage # see the percentage of cointegrated relationships
est <- tscsEstimate(matrix = basis$coef_matrix, newdata = newdata, h = 1, v = 1); # estimation
str(est)
```

```
## comparison of estimates and true values:

plot_compare(est = est$estimate[,3], true = true) # graphic comparison
index <- appraisal_index(est = est$estimate[,3], true = true); # RMSE & std
index

## data visualization:

plot_dif(data = data[,1:2], h = 1, v = 1) # differentiate boundary and interior spatial locations
plot_NA(newdata = newdata) # show spatial locations with missing value, for a cross-section data
plot_map(newdata = newdata) # plot the 2D spatial map, for a cross-section data

## End(Not run)
```

---

plot_map *Visualize Spatial(Cross-Section) Data of a Given Time Point - 2D Map*

Description

plot_map draws a two-dimensional spatial map. It is plotted based on the cross-section data of a given time point, which is also often extracted from spatio-temporal data.

Usage

```
plot_map(newdata, xlab = NULL, ylab = NULL, title = NULL, cex = 2,
  shape = 15, low = "blue", mid = "yellow", high = "red",
  na.value = "white", midpoint = NULL)
```

Arguments

newdata	data frame; should only contain the three variables in order: X coordinate, Y coordinate and observation. This is the cross-section data or pure spatial data of a particular time point you have selected, with missing observations that you want to predict. (coordinates must be numeric)
xlab	a label for the x axis, defaults to the name of X coordinate.
ylab	a label for the y axis, defaults to the name of Y coordinate.
title	a main title for the plot.
cex	numeric; size of plotting point for each spatial locations. (default: 2)
shape	either an integer specifying a symbol or a single character to be used as the default in plotting points. (default: 15)
low, high	colours for low and high ends of the gradient. (default: "blue","red")
mid	colour for midpoint of the gradient. (default: "yellow")
na.value	colour for missing values/observations. (default: "white")
midpoint	numeric; the midpoint of the gradient scale, defaults to the midpoint value of index presented.

Details

`plot_map` is exclusive to 2D rectangular grid system. Similarly, if you want to fathom how this package handles 3D rectangular grid system, please refer to `plot3D_map`.

See Also

`plot3D_map`, `plot_NA`, `plot_dif`

Examples

```
## Not run:

## TSCS spatial interpolation procedure:

basis <- tscsRegression(data = data, h = 1, v = 1, alpha = 0.01); # regression
basis$percentage # see the percentage of cointegrated relationships
est <- tscsEstimate(matrix = basis$coef_matrix, newdata = newdata, h = 1, v = 1); # estimation
str(est)

## comparison of estimates and true values:

plot_compare(est = est$estimate[,3], true = true) # graphic comparison
index <- appraisal_index(est = est$estimate[,3], true = true); # RMSE & std
index

## data visualization:

plot_dif(data = data[,1:2], h = 1, v = 1) # differentiate boundary and interior spatial locations
plot_NA(newdata = newdata) # show spatial locations with missing value, for a cross-section data
plot_map(newdata = newdata) # plot the 2D spatial map, for a cross-section data

## End(Not run)
```

`plot_NA` *Visualize the Spatial Distribution of Missing Observations - 2D Map*

Description

`plot_NA` shows spatial locations with or without missing observation. It is plotted based on the cross-section data of a given time point, which is also often extracted from spatio-temporal data.

Usage

```
plot_NA(newdata, xlab = NULL, ylab = NULL, title = NULL, cex = 1)
```

Arguments

newdata	data frame; should only contain the three variables in order: X coordinate, Y coordinate and observation. This is the cross-section data or pure spatial data of a particular time point you have selected, with missing observations that you want to predict. (coordinates must be numeric)
xlab	a label for the x axis, defaults to the name of X coordinate.
ylab	a label for the y axis, defaults to the name of Y coordinate.
title	a main title for the plot.
cex	numeric; size of plotting point for each spatial location. (default: 1)

Details

plot_NA is exclusive to 2D rectangular grid system. Similarly, if you want to fathom how this package handles 3D rectangular grid system, please refer to plot3D_NA.

See Also

plot3D_NA, plot_map, plot_dif

Examples

```
## Not run:

## TSCS spatial interpolation procedure:

basis <- tscsRegression(data = data, h = 1, v = 1, alpha = 0.01); # regression
basis$percentage # see the percentage of cointegrated relationships
est <- tscsEstimate(matrix = basis$coef_matrix, newdata = newdata, h = 1, v = 1); # estimation
str(est)

## comparison of estimates and true values:

plot_compare(est = est$estimate[,3], true = true) # graphic comparison
index <- appraisal_index(est = est$estimate[,3], true = true); # RMSE & std
index

## data visualization:

plot_dif(data = data[,1:2], h = 1, v = 1) # differentiate boundary and interior spatial locations
plot_NA(newdata = newdata) # show spatial locations with missing value, for a cross-section data
plot_map(newdata = newdata) # plot the 2D spatial map, for a cross-section data

## End(Not run)
```

`TSCS` *A Package for TSCS Spatial Interpolation Method*

Description

This package provides functions to implement TSCS spatial interpolation and relevant data visualization. For TSCS method, the current version is only able to make use of spatio-temporal data whose spatial domain is a 2D or 3D rectangular grid system.

Details

1. TSCS (abbr. of Time Series Cointegrated System) method is a spatial interpolation method based on analysis of historical spatio-temporal data. It can be regarded as a desirable alternative to spatio-temporal interpolation in some cases where we merely intend to interpolate a series of cross-section data at each observed time point for a given spatial domain.
2. The basic assumption of TSCS method is that, for any spatial location within the spatial domain of spatio-temporal data, its time series and the time series of its adjacent spatial locations are cointegrated (long-term equilibrium relationships).
3. As to TSCS method, package of the current version is only able to make use of spatio-temporal data whose spatial domain is a 2D or 3D rectangular grid system.

Package Functions

- `tscsRegression, tscsRegression3D` : obtains regression coefficient matrix, the first step of TSCS for 2D and 3D rectangular grid system respectively.
- `tscsEstimate, tscsEstimate3D` : estimates the missing observations within a cross-section data (pure spatial data) of a particular time point you have selected, the second step of TSCS for 2D and 3D rectangular grid system respectively.
- `plot_dif, plot3D_dif` : differentiates boundary and interior spatial locations in a spatial domain.
- `plot_NA, plot3D_NA` : shows spatial locations with or without missing observation in a spatial domain.
- `plot_map, plot3D_map` : draws the spatial map for a cross-section data.
- `plot_compare` : visualizes the comparison between estimates and true values (if you have).
- `appraisal_index` : computes the two appraisal indexes used for evaluating the result of interpolation/prediction - RMSE and standard deviation of error. (if you have the true values)

Author(s)

Tianjian Yang <yangtj5@mail2.sysu.edu.cn>

`tscsEstimate` *The Second Step of TSCS for 2D Rectangular Grid System - Estimation*

Description

`tscsEstimate` estimates the missing observations within the cross-section data (pure spatial data) of a particular time point you have selected, namely, the interpolation process.

Usage

```
tscsEstimate(matrix, newdata, h, v)
```

Arguments

`matrix`	data frame; the first return value `coef_matrix` of function `tscsRegression` in the first step of TSCS.
`newdata`	data frame; should only contain the three variables in order: X coordinate, Y coordinate and observation. This is the cross-section data or pure spatial data of a particular time point you have selected, with missing observations that you want to predict. (coordinates must be numeric)
`h`	numeric; side length of the unit grid in X coordinate direction.
`v`	numeric; side length of the unit grid in Y coordinate direction.

Details

- The first step of TSCS spatial interpolation should be carried out by function `tscsRegression`, which is the prerequisite of `tscsEstimate`.
- For 3D rectangular grid system, the procedure of TSCS stays the same. Please see `tscsRegression3D` and `tscsEstimate3D`.
- Attentions: Since TSCS is only capable of interpolation but not extrapolation, please make sure that the missing observations in a given spatial domain are all located at interior spatial locations. Otherwise, extrapolation would occur with an error following.

Value

A list of 3 is returned, including:

`estimate` data frame; estimate of missing observations which contains the 3 variables in order: X coordinate, Y coordinate and estimation.

`complete` data frame; an updated version of the cross-section data (pure spatial data) `newdata`, with all of its missing observations interpolated.

`NA_id` an integer vector; reveals the instance ID, in data frame `newdata`, of spatial locations with missing observation.

See Also

`tscsRegression`, `tscsEstimate3D`, `plot_NA`, `plot_map`

Examples

```
## Not run:

## TSCS spatial interpolation procedure:

basis <- tscsRegression(data = data, h = 1, v = 1, alpha = 0.01); # regression
basis$percentage # see the percentage of cointegrated relationships
est <- tscsEstimate(matrix = basis$coef_matrix, newdata = newdata, h = 1, v = 1); # estimation
str(est)

## comparison of estimates and true values:

plot_compare(est = est$estimate[,3], true = true) # graphic comparison
index <- appraisal_index(est = est$estimate[,3], true = true); # RMSE & std
index

## data visualization:

plot_dif(data = data[,1:2], h = 1, v = 1) # differentiate boundary and interior spatial locations
plot_NA(newdata = newdata) # show spatial locations with missing value, for a cross-section data
plot_map(newdata = newdata) # plot the 2D spatial map, for a cross-section data

## End(Not run)
```

`tscsEstimate3D` *The Second Step of TSCS for 3D Rectangular Grid System - Estimation*

Description

`tscsEstimate` estimates the missing observations within the cross-section data (pure spatial data) of a particular time point you have selected, namely, the interpolation process.

Usage

```
tscsEstimate3D(matrix, newdata, h1, h2, v)
```

Arguments

`matrix`	data frame; the first return value `coef_matrix` of function `tscsRegression3D` in the first step of TSCS.
`newdata`	data frame; should only contain the four variables in order: X coordinate, Y coordinate, Z coordinate and observation. This is the cross-section data or pure spatial data of a particular time point you have selected, with missing observations that you want to predict. (coordinates must be numeric)
`h1`	numeric; side length of the unit cubic grid in X coordinate direction (horizontal).
`h2`	numeric; side length of the unit cubic grid in Y coordinate direction (horizontal).
`v`	numeric; side length of the unit cubic grid in Z coordinate direction (vertical).

Details

- The first step of TSCS spatial interpolation should be carried out by function `tscsRegression3D`, which is the prerequisite of `tscsEstimate3D`.
- For 2D rectangular grid system, the procedure of TSCS stays the same. Please see `tscsRegression` and `tscsEstimate`.
- Attentions: Since TSCS is only capable of interpolation but not extrapolation, please make sure that the missing observations in a given spatial domain are all located at interior spatial locations. Otherwise, extrapolation would occur with an error following.

Value

A list of 3 is returned, including:

`estimate` data frame; estimate of missing observations which contains the 4 variables in order: X coordinate, Y coordinate, Z coordinate and estimation.

`complete` data frame; an updated version of the cross-section data (pure spatial data) `newdata`, with all of its missing observations interpolated.

`NA_id` an integer vector; reveals the instance ID, in data frame `newdata`, of spatial locations with missing observation.

See Also

`tscsRegression3D`, `tscsEstimate`, `plot3D_NA`, `plot3D_map`

Examples

```
## Not run:

## TSCS spatial interpolation procedure:

basis <- tscsRegression3D(data = data, h1 = 3.75, h2 = 2.5, v = 5, alpha = 0.01);
basis$percentage
est <- tscsEstimate3D(matrix = basis$coef_matrix, newdata = newdata, h1 = 3.75, h2 = 2.5, v = 5);
str(est)

## comparison of estimates and true values:

plot_compare(est = est$estimate[,4], true = true)
index <- appraisal_index(est = est$estimate[,4], true = true);
index

## data visualization:

plot3D_dif(data = data[,1:3], h1 = 3.75, h2 = 2.5, v = 5)
plot3D_NA(newdata = newdata)
plot3D_map(newdata = newdata)

## End(Not run)
```

`tscsRegression` *The First Step of TSCS for 2D Rectangular Grid System - Regression*

Description

To implement TSCS spatial interpolation for a spatial domain that is a 2D rectangular grid system, the first step is obtaining regression coefficient matrix, which can be done by function `tscsRegression`. It is the prerequisite of TSCS interpolation process because the 'matrix' derived from historical spatio-temporal data is the initial value of the second step - estimating missing observations.

Usage

```
tscsRegression(data, h, v, alpha = 0.05)
```

Arguments

`data`	data frame; should contain these variables in order: X coordinate, Y coordinate and observations as time goes on. That is to say, each row should include X and Y coordinate first, and then a time series. This is the historical spatio-temporal data that you intend to analyze as the basis for interpolation later on in `tscsEstimate`.
`h`	numeric; side length of the unit grid in X coordinate direction.
`v`	numeric; side length of the unit grid in Y coordinate direction.
`alpha`	numeric; specify the significance level for ADF test, to test if the time series of a group of spatial locations are cointegrated. (default: 0.05)

Details

- The second step of TSCS spatial interpolation should be carried out by function `tscsEstimate`, where you have to input the cross-section data or pure spatial data of a particular time point you have selected, with missing observations that you want to predict.
- For 3D rectangular grid system, the procedure of TSCS stays the same. Please see `tscsRegression3D` and `tscsEstimate3D`.
- Attentions: (1) Since TSCS is only capable of interpolation but not extrapolation, it is necessary to highlight the difference between interior spatial locations and system boundary. Function `plot_dif` can help. (2) NA value in historical spatio-temporal data `data` is not allowed. Please handle them beforehand (such as filling these NA values through spatio-temporal kriging).

Value

A list of 2 is returned, including:

`coef_matrix` data frame; regression coefficient matrix to be used as input parameter of function `tscsEstimate` in the second step of TSCS interpolation.

`percentage` numeric; percentage of cointegrated relationships, a measurement of the degree it satisfies the assumption of cointegrated system. It is highly affected by parameter `alpha`, the significance level you have set. Explicitly, smaller `alpha` results in smaller `percentage`.

See Also

tscsEstimate, tscsRegression3D, plot_dif

Examples

```
## Not run:

## TSCS spatial interpolation procedure:

basis <- tscsRegression(data = data, h = 1, v = 1, alpha = 0.01); # regression
basis$percentage # see the percentage of cointegrated relationships
est <- tscsEstimate(matrix = basis$coef_matrix, newdata = newdata, h = 1, v = 1); # estimation
str(est)

## comparison of estimates and true values:

plot_compare(est = est$estimate[,3], true = true) # graphic comparison
index <- appraisal_index(est = est$estimate[,3], true = true); # RMSE & std
index

## data visualization:

plot_dif(data = data[,1:2], h = 1, v = 1) # differentiate boundary and interior spatial locations
plot_NA(newdata = newdata) # show spatial locations with missing value, for a cross-section data
plot_map(newdata = newdata) # plot the 2D spatial map, for a cross-section data

## End(Not run)
```

---

tscsRegression3D — *The First Step of TSCS for 3D Rectangular Grid System - Regression*

Description

To implement TSCS spatial interpolation for a spatial domain that is a 3D rectangular grid system, the first step is obtaining regression coefficient matrix, which can be done by function `tscsRegression3D`. It is the prerequisite of TSCS interpolation process because the 'matrix' derived from historical spatio-temporal data is the initial value of the second step - estimating missing observations.

Usage

```
tscsRegression3D(data, h1, h2, v, alpha = 0.05)
```

Arguments

`data`	data frame; should contain these variables in order: X coordinate, Y coordinate, Z coordinate and observations as time goes on. That is to say, each row should include X, Y and Z coordinate first, and then a time series. This is the historical spatio-temporal data that you intend to analyze as the basis for interpolation later on in `tscsEstimate3D`.

`h1`	numeric; side length of the unit cubic grid in X coordinate direction (horizontal).
`h2`	numeric; side length of the unit cubic grid in Y coordinate direction (horizontal).
`v`	numeric; side length of the unit cubic grid in Z coordinate direction (vertical).
`alpha`	numeric; specify the significance level for ADF test, to test if the time series of a group of spatial locations are cointegrated. (default: 0.05)

Details

- The second step of TSCS spatial interpolation should be carried out by function `tscsEstimate3D`, where you have to input the cross-section data or pure spatial data of a particular time point you have selected, with missing observations that you want to predict.
- For 2D rectangular grid system, the procedure of TSCS stays the same. Please see `tscsRegression` and `tscsEstimate`.
- Attentions: (1) Since TSCS is only capable of interpolation but not extrapolation, it is necessary to highlight the difference between interior spatial locations and system boundary. Function `plot3D_dif` can help. (2) NA value in historical spatio-temporal data `data` is not allowed. Please handle them beforehand (such as filling these NA values through spatio-temporal kriging).

Value

A list of 2 is returned, including:

`coef_matrix` data frame; regression coefficient matrix to be used as input parameter of function `tscsEstimate` in the second step of TSCS interpolation.

`percentage` numeric; percentage of cointegrated relationships, a measurement of the degree it satisfies the assumption of cointegrated system. It is highly affected by parameter `alpha`, the significance level you have set. Explicitly, smaller `alpha` results in smaller `percentage`.

See Also

`tscsEstimate3D`, `tscsRegression`, `plot3D_dif`

Examples

```
## Not run:

## TSCS spatial interpolation procedure:

basis <- tscsRegression3D(data = data, h1 = 3.75, h2 = 2.5, v = 5, alpha = 0.01);
basis$percentage
est <- tscsEstimate3D(matrix = basis$coef_matrix, newdata = newdata, h1 = 3.75, h2 = 2.5, v = 5);
str(est)

## comparison of estimates and true values:

plot_compare(est = est$estimate[,4], true = true)
index <- appraisal_index(est = est$estimate[,4], true = true);
index
```

```
## data visualization:

plot3D_dif(data = data[,1:3], h1 = 3.75, h2 = 2.5, v = 5)
plot3D_NA(newdata = newdata)
plot3D_map(newdata = newdata)

## End(Not run)
```

Index