Package 'stpca'

April 28, 2016

Type Package

Title Tools To Investigate Reducing Monitoring Networks

Version 0.1

Date 2016-03-23

Author Kelly Gallacher [aut, cre],
Claire Miller [aut],
Marian Scott [aut]

Maintainer Kelly Gallacher <kelly_gallacher@hotmail.com>

Description This package consists of functions that can be used to investigate re-designing monitoring networks, with particular emphasis on river networks. For spatiotemporal data this package will allow users to identify areas of the river network where observatations have remained stable over time, and also identify groups of monitoring sites with similar behaviour over time. Weights can be incorporated to adjust for spatial and temporal correlation. For spatial data, this package will allow users to investigate how reducing the size of the monitoring network affects predictions at unobserved locations, and also how the choice of sampling scheme affects predictions. Many of the functions in this package are wrappers for functions from other packages and these are referenced throughout the help files.

License CC0

LazyData TRUE

RoxygenNote 5.0.1

Suggests knitr, maptools, rmarkdown, testthat

Depends R (>= 3.2.0)

Imports
expm, geoR, graphics, grDevices, GWmodel, matrixcalc, Matrix, missMDA, sampling, sp, SSN

VignetteBuilder knitr

R topics documented:

`completeData` *Wrapper function for* `missMDA` *package to impute missing values*

Description

A wrapper function to impute missing values using the `missMDA` package (Josse and Husson (2012)). The function estimates the number of principal components used to impute missing values and carries out multiple imputation. Diagnostic plots can also be produced if required.

Usage

```
completeData(x, pca.mode = NULL, ncp.min = 0, ncp.max = 5,
  scale = FALSE, mean.plot = TRUE, response.var = "response (units)",
  estim.plot = TRUE, nbsim = 100, pNA = 0.05, nboot = 100,
  mi.plot = FALSE)
```

Arguments

`x`	data.frame. Missing values should be indicated by NA. Columns MUST be monitoring sites while rows MUST be timepoints.
`pca.mode`	character string. MUST be specified. `"Smode"` is PCA performed on data where columns are monitoring sites. `"T-mode"` is for PCA performed on data where columns are time points. See Details for further information.
`ncp.min`	integer. Minimum number of principal components used to estimate missing values. Default is 0. See Details for more information.
`ncp.max`	integer. Maximum number of principal components used to estimate missing values. Default is 5.
`scale`	logical. TRUE will divide values in x by column standard deviation and should be used if PCA is to be performed on correlation matrix. Default is FALSE and is equivalent to PCA performed on covariance matrix.
`mean.plot`	logical. If TRUE will produce a plot of the mean time series of incomplete data and mean time series of data completed using imputation. Default is TRUE.
`response.var`	character string. The text for the y-axis label in `mean.plot`. Default is "response (units)". Only used if `mean.plot = TRUE`.
`estim.plot`	logical. If TRUE (default) then a plot is produced showing the cross validation score for `ncp.min` to `ncp.max` principal components. See Details for further information.
`nbsim`	integer. Number of simulations for K-fold cross validation.
`pNA`	numeric. Must be in the range (0, 1). Specifies the proportion of values to leave out during K-fold cross validation. See Details for further information.

`nboot`	integer. Number of simulated data sets to create during multiple imputation. See Details for further information.
`mi.plot`	logical. If TRUE will produce a diagnostic plot to assess variability from imputed data. Default is FALSE. See Details for further information.

Details

This function uses "K-fold" cross validation to estimate the number of principal component to use to impute missing values. Alternative cross validation methods are available in the `estim_ncpPCA` function in the `missMDA` package. In order to assess uncertainty due to missingness, this function also implements multiple imputation, based on the `MIPCA` function in the `missMDA` package. At present, `method.mi` from `MIPCA` is set to `"Boot"`. Further information can be found by consulting the help documentation in the `missMDA` package.

`pca.mode` The `pca.mode` must be specified otherwise an error message will be produced. This can be specified as either `"Smode"` or `"Tmode"`. For S-mode, PCA is performed on a data matrix where the columns are monitoring sites and the rows are regularly spaced time points. For T-mode, PCA will be performed on a data matrix where the columns are regularly spaced time points and the rows are monitoring sites. See the Common_Patterns vignette or Richman (1986) for more information.

`ncp.min` If this equals 0 then missing values have been imputed using column means. This suggests that variables (columns) in the data are not correlated.

`estim.plot` This plots the number of principal components used to impute missing values against the K-fold cross validation score. The number of principal components used to produce the completed data is the one corresponding to the minimum cross validation score.

`pNA` This is the proportion of values to remove from the observed data that will be used for "Kfold" cross validation. Further details about this can be found in help file for `estim_ncpPCA` function in the `missMDA` package.

`nboot` is the number of simulated data sets to create during multiple imputation, to assess variability of results due to missingness. Further information can be found in the `MIPCA` help file in the `missMDA` package.

`mi.plot` This produces the `"dim"` plot from the `choice` argument in `plot.MIPCA` function in the `missMDA` package. The plot can only be produced if the number of principal components (`ncp`) used to calculate missing values is 2 or more. If `TRUE` then the plot will be produced for the first 2 principal components otherwise if `ncp` < 2 a warning message is produced. The function will run a bit quicker if this is set to `FALSE` as the plot takes a few minutes to be produced.

Value

`ncp` is the number of components to use to calculate missing values, estimated using K-fold cross validation.

`criterion` are the cross validation values for ncp.min to ncp.max.

`res.MIPCA` contains `res.MI = nboot` simulated data sets used to assess variability from missing values and to produce `mi.plot`, `res.imputePCA` is the completed data set with no missing values and calculated using `ncp` principal components, `call` is the call to the function.

The function will produce plots showing `criterion` against `ncp`, `mi.plot` if TRUE and `ncp` > 1, `mean.plot` if TRUE.

References

Josse, J. and F. Husson (2012). Handling missing values in exploratory multivariate data analysis methods. Journal de la Societe Francaise de Statistique, 153(2), 79-99.

Richman, M. B. (1986), Rotation of principal components. J. Climatol., 6: 293-335. doi:10.1002/joc.3370060305

Examples

```
library(stpca)

## load a demo dataset containing missing values
data(demoYmiss)

## Imputation in T-mode
Tmode.impute <- completeData(demoYmiss, pca.mode = "Tmode", ncp.min = 0,
                             ncp.max = 1, scale = FALSE, mean.plot = TRUE,
                             response.var ="response (units)",
                             estim.plot = FALSE, nbsim = 10, pNA = 0.05,
                             nboot = 10, mi.plot = TRUE)
## Imputation in S-mode
Smode.impute <- completeData(x = demoYmiss, pca.mode = "Smode", ncp.min = 0,
                             ncp.max = 4, scale = FALSE, mean.plot = TRUE,
                             response.var ="response  (units)",
                             estim.plot = TRUE, nbsim = 10, pNA = 0.05,
                             nboot = 10, mi.plot = TRUE)
```

createWeightS — *Create an asymmetric matrix of spatial weights from an SSN object.*

Description

The `createWeightS` function extracts the stream distance matrix for observed data from a `SpatialStreamNetwork-clas` object and constructs an asymmetric weight matrix based on the specified additive function column.

Usage

```
createWeightS(ssndata, afvcol)
```

Arguments

`ssndata`	filepath for an object of `SpatialStreamNetwork-class`. Include the .ssn folder in the path.
`afvcol`	character string. This should be the column name in the SSN object containing additive function values.

Details

Peterson and ver Hoef (2010) Appendix A shows how weights can be calculated for the tail-up model, where weights include information about the flow connected structure in the river network, network distance and relative influence of upstream sites on downstream sites. Weights are based on the variable used to calculate the additive function in the SSN object. `createWeightS` follows the steps described in Appendix A but stops before the matrix is forced to symmetry.

Value

Produces an asymmetric $p \times p$ matrix (p = number of monitoring sites) where columns = upstream sites, rows = downstream sites.

References

Peterson, E. E. and J. M. ver Hoef (2010). A mixed-model moving average approach to geostatistical modeling in stream networks. Ecology 91, 644-651.

Examples

```
library(stpca)

## get filepath for SSN object
ssndata <- system.file("demoSSN/demoNet.ssn", package = "stpca")

## create matrix of spatial weights
weightS <- createWeightS(ssndata = ssndata, afvcol = "addfunccol")
```

`createWeightT` *Create a matrix of temporal weights.*

Description

The `createWeightT` function creates a symmetric matrix of temporal weights based on temporal autocorrelation. At present only AR(1) structure has been implemented.

Usage

```
createWeightT(n, rho, corr.form = "AR1")
```

Arguments

`n`	integer. This should specify the number of time points in the data.
`rho`	scalar. Rho must be a value between 0 and 1 reflecting the strength of temporal autocorrelation.
`corr.form`	character string. At present only "AR1" is implemented for autoregressive temporal correlation of lag 1.

Details

Description of AR1 correlation.

Value

An $n \times n$ matrix of temporal weights.

Examples

```
weightT <- createWeightT(n = 5, rho = 0.5)
```

`demoNet` *Simulated observations of a single water quality parameter on a river network.*

Description

Simulated values of a single water quality parameter recorded at 30 monitoring sites on a simulated river network.

Usage

```
demoNet
```

Format

An object of `SpatialStreamNetwork-class` containing simulated observations at a single time point of a single water quality parameter. The SSN object contains two data.frames. "Obs" contains data and other variables related to 30 observed locations and "preds" contains the same information but for 30 unobserved prediction locations. Full details of the structure of an SSN object can be found by looking at `SpatialStreamNetwork-class`. The "Obs" and "preds" data.frames for this simulated network contain the following variables:

locID unique identifier for each monitoring site.

upDist the upstream distance (unitless) from outlet to the monitoring site.

pid unique identifier for each row of the "Obs" and "preds" data.frames.

netID identifier for the river network.

rid identifier for the stream segment.

ratio location of monitoring site on a stream segment as a proportion of the segment length from the downstream end of the segment.

shreve Shreve's number for the stream segment.

addfunccol additive function value, based on Shreve's number.

NEAR_X x-coordinate of monitoring site location (equivalent to Easting in real data).

NEAR_Y y-coordinate of monitoring site location (equivalent to Northing in real data).

Sim_Values simulated observed values of a single water quality parameter.

strata a randomly allocated category used for stratified sampling.

weight a value in the range (0,1) used for weighted sampling.

Source

See vignette for Creating Demo Data

`demoY` *Spatiotemporal observations of a single water quality parameter.*

Description

A complete demo dataset of a single water quality parameter recorded at regularly spaced time intervals and several monitoring sites, with no missing values.

Usage

```
demoY
```

Format

A data.frame with 30 variables (columns, monitoring sites) and 25 observations (rows, time points). These time points are assumed to be regularly spaced so could represent weekly/monthly/annual intervals.

x1 - x30 25 regularly spaced temporal observations at sites x1 - x30

Source

See vignette for Creating Demo Data

`demoYmiss` *Spatiotemporal observations of a single water quality parameter.*

Description

A demo dataset of a single water quality parameter recorded at regularly spaced time intervals and several monitoring sites, with 15% of missing values.

Usage

```
demoYmiss
```

Format

A data.frame with 30 variables (columns, monitoring sites) and 25 observations (rows, time points). These time points are assumed to be regularly spaced so could represent weekly/monthly/annual intervals.

x1 - x30 25 regularly spaced temporal observations at sites x1 - x30

Source

See vignette for Creating Demo Data

`plot_sampNet` *Plot the results from* `sampNet`

Description

This function will plot the results from creating several sampled networks using `sampNet`. This will allow the user to make comparisons between sampling schemes and proportion of monitoring sites retained in sampled networks.

Usage

```
plot_sampNet(samp.results, cov.params.plots = TRUE, pred.plots = TRUE,
  log.TUrange = FALSE, log.Eucrange = FALSE)
```

Arguments

`samp.results`	character. Contains the name of the object in which the results from `sampNet` are stored.
`cov.params.plots`	logical. If `TRUE`, then plots of the covariance parameters estimated for `nsim` sampled networks will be produced. See Details for further information.
`pred.plots`	logical. If `TRUE`, then plots of predicted values and standard errors estimated for `nsim` sampled networks will be produced. See Details for further information.
`log.TUrange`	logical. If `TRUE`, then the y-axis and values plotted in the Tail-up range parameter plot will be transformed using a natural logarithmic transformation. See Details.
`log.Eucrange`	logical. If `TRUE`, then the y-axis and values plotted in the Euclidean range parameter plot will be transformed using a natural logarithmic transformation. See Details.

Details

This function can be used to produce plots of several parameters/values estimated for each `prop.sites` and `samp.scheme` combination.

If `cov.params.plots = TRUE` and `class(x)` is `SpatialStreamNetwork-class`, then five plots will be produced showing the covariance function parameters estimated for `nsim` sampled networks: Tail-up range, Euclidean range, Tail-up partial sill, Euclidean partial sill, and nugget. If `class(x)` is `"data.frame"` then thre plots will be produced: range, partial sill, and nugget.

If `pred.plots = TRUE`, then three plots will be produced: root mean square predicted error (RMSPE) for observed locations, prediction error for prediction locations, and average kriging standard error (AKSE) ratio.

RMSPE is calculated for observed locations as follows:

$$\sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\hat{Y}(s_i) - Y(s_i)\right)^2}$$

where N is the number of monitoring sites, $\hat{Y}(s_i)$ is the value at site i predicted from a model fitted to $\mathbf{s}_{-i}$, data from all monitoring sites except site i, and $Y(s_i)$ is the observed value at site i. This is also known as Leave One Out Cross Validation (LOOCV). This is a measure of how well the model fits the data. Low prediction error is desireable.

Prediction error is calculated for prediction locations as follows:

$$\sqrt{\frac{1}{N_{pred}}\sum_{i=1}^{N_{pred}}\left(\hat{Y}_{full}(u_i)-\hat{Y}_{sampled}(u_i)\right)^2}$$

where N_{pred} is the number of prediction locations, $\hat{Y}_{full}(u_i)$ is the value predicted at prediction (unobserved) location i where the prediction is made using a model fitted to the full monitoring network, and $\hat{Y}_{sampled}(u_i)$ is the value predicted at prediction (unobserved) location i where the prediction is made using a model fitted to a sampled (reduced) network. This is a measure of how well the model based on fewer monitoring sites predicts at unobserved locations compared to the model based on the full monitoring network. As with RMSPE, low values are desired.

AKSE ratio can be calculated as the ratio of AKSE from a model based on a sampled (reduced) network to AKSE from a model based on the full monitoring netowrk: $\text{AKSE}_{sampled}/\text{AKSE}_{full}$ where

$$\text{AKSE}=\sqrt{\frac{1}{N_{pred}}\sum_{i=1}^{N_{pred}}\sigma^2(u_i)}$$

and N_{pred} is the number of prediction locations, and $\sigma^2(u_i)$ is the kriging variance at prediction location i.

If `log.TUrange = TRUE` and `class(x)` is `SpatialStreamNetwork-class`, then values for the Tail-up range parameter will have a natural log transformation applied. This is useful if the `nsim` estimated values vary by orders of magnitude and it is left to the user to decide which display (transformed values or untransformed values) is most useful for their own application. The transformation is also available for the Euclidean range parameter for `class(x)` is `SpatialStreamNetwork-class` or `"data.frame"`.

Value

Two sets of plots will be produced depending on argument values specified.

Examples

```
data(sampDemo)
plot_sampNet(sampDemo,
             cov.params.plots = TRUE,
             pred.plots = TRUE)
```

`plot_stpca` *Visualize (weighted) PCA results.*

Description

Visualize the results from applying PCA, adjusted for spatial and/or temporal weights if appropriate. This will plot the results from the output of `stpca`.

Usage

```
plot_stpca(x, plots = c("map", "biplot", "glyph", "ts", "meanPM"),
  pc.map = 1, probs = c(0, 0.25, 0.75, 1), pc.biplot = c(1, 2),
  biplot.factor = 100, pc.glyph = c(1, 2, 3), river, r1 = 10, coords,
  pc.ts = c(1, 2, 3), pc.meanPM = c(1, 2, 3), meanPM.factor = 0.05)
```

Arguments

`x`	character string. This should specify the name of the object in which the results of `stpca` are stored.
`plots`	character string. Specifies the type of plot to be produced. Can include any combination of c("map", "biplot", "glyph", "ts", "meanPM"). See Details for further information.
`pc.map`	vector (of length one). Specifies which principal component to plot on a map. Default is PC1.
`probs`	vector. A vector of probabilities used to calculate color breaks when `plots = "map"`. Values should be in the range [0,1].
`pc.biplot`	vector. Specifies the principal components to plot when `plots = "biplot"`. Default is PC1 on x-axis and PC2 on y-axis.
`biplot.factor`	scalar. A scaling value used to produce the biplot. Higher values make arrows shorter.
`pc.glyph`	vector. Specifies the principal components to plot when `plots = "glyph"`. It is recommended that the glyph plot is used only to display 3 or more principal components.
`river`	character string. Name of object containing river network shapefile. Used for plotting results on a map of the river network where `plots = c("map", "glyph")`.
`r1`	scalar. Used to control relative lengths of spokes on glyphs in glyph plot. Larger values decrease the size of the glyphs. Default is 10.
`coords`	matrix of 2 columns. The columns should contain the x- and y-axis coordinates for the monitoring site locations respectively and MUST be in the same order as monitoring sites are specified in rows (`pca.mode = "Tmode"`) or columns (`pca.mode = "Smode"`) in the dataframe specified in `stpca`.
`pc.ts`	vector. Specifies which principal components should have principal components/scores plotted as time series. Should only be used when `pca.mode = "Smode"` in `stpca`. Default is c(1,2,3).
`pc.meanPM`	vector. Specifies which principal components should be plotted. Should only be used when `pca.mode = "Smode"` in `stpca`. Default is c(1,2,3).
`meanPM.factor`	scalar. Used to control the contrast between the mean time series over all sites and +/- a multiple of the principal components/scores. See Details.

Details

The type of plot to use will depend on how many principal components are of interest. If only 1 PC is of interest then `plots = "map"` should be used, or `plots = "biplot"` if a pair of PC's should be plotted. For 3 or more PC's then `plots = "glyph"` is recommended.

`"map"` will display loadings (S-mode PCA) or scores (T-mode PCA) for a single PC. Color breaks are specified using the `probs` argument.

"biplot" displays the standard biplot, commonly used to investigate the results of PCA and displays both scores and loadings on the same plot, for any 2 PC's specified using pc.biplot. Arrows represent loadings and points represent principal component scores.

"glyph" is based on glyph.plot in the GWmodel package, and displays a map of the river network (if river is specified) and glyphs showing the loadings (S-mode PCA) or scores (T-mode PCA) for the principal components specified using pc.glyph. The lengths of the spokes are relative to each other and the spoke at the 12 o'clock position represents the first PC specified in pc.glyph, and other PC's are represented moving round clockwise. Blue indicates positive values and red indicates negative values. A message will appear when glyph plots are produced prompting the user to check that the matrix of coordinates has monitoring sites in the same order as in the rows (pca.mode = "Tmode") or columns (pca.mode = "Smode") of the dataframe used for stpca. Incorrect ordering will result in glyphs being displayed at the wrong locations. If a glyph plot is produced for fewer than three PC's then the plot will be produced but a message suggesting other plots might be more appropriate will also appear. Glyph plots and code are based on Isabella et. al. (2015).

"ts" displays a time series plot of the principal components when pca.mode = "Smode" in stpca. This plot is not produced for pca.mode = "Tmode" since a time series of loadings has no meaningful interpretation. Rembember that since loadings can be positive or negative, the temporal pattern described by each PC might the negative of what is displayed on the plot.

"meanPM" displays the mean time series for all sites, +/- a multiple (meanPM.factor) of the principal components specified using pc.meanPM. Red symbols are the mean time series + (meanPM.factor*scores), blue symbols are the mean time series - (meanPM.factor*scores). This plot shows the variation around the mean captured by each principal component. The pattern might indicate that a principal component describes a shift in the mean, where some monitoring sites have generally higher/lower values than the mean. The principal component might describe a dampening of the mean signal, where the seasonal pattern oscillates at a lower frequency than the mean pattern. Another possibility is that the prinicpal compnent captures a time lag at some sites compared to the mean. This list is not exhaustive but has been included to provide some suggestions for interpretation.

Value

Produces the specified plot(s) for the principal components specified and for each weighting scheme used in stpca.

References

Biplot code is based on:

http://steviep42.bitbucket.org/YOUTUBE.DIR/BB_phys_stats_ex1.R and https://www.youtube.com/watch?v=I5GxNzKLIoU.

Glyph plot based on:

Isabella Gollini, Binbin Lu, Martin Charlton, Christopher Brunsdon, Paul Harris (2015). GWmodel: An R Package for Exploring Spatial Heterogeneity Using Geographically Weighted Models. Journal of Statistical Software, 63(17), 1-50. http://www.jstatsoft.org/v63/i17/.

Examples

```
library(stpca)
library(maptools) #for reading in shapefile
library(SSN) #to import SSN object

## get data.frame with no missing values
data(demoY)
```

```
## unweighted T-mode PCA
Tmode.pca.uw <- stpca(x=demoY, pca.mode="Tmode", pca.wt=c("unweighted"))

## spatial, temporal, and spatiotemporal weighted S-mode PCA

# create spatial weights
ssndata <- system.file("demoSSN/demoNet.ssn", package="stpca")
weightS <- createWeightS(ssndata=ssndata, afvcol="addfunccol")

# create temporal weights
weightT <- createWeightT(n=nrow(demoY), rho=0.5)

# check that the order of monitoring sites in x is the same as the
# columns in spatial.wt.  They are the same if the following results
# in 1.
mean(colnames(demoY) == colnames(weightS))

# weighted S-mode PCA
Smode.pca.all <- stpca(x = demoY, pca.mode = "Smode",
                       spatial.wt = weightS, temporal.wt = weightT,
                       pca.wt = c("unweighted", "spatial", "temporal",
                                  "spatiotemporal"))

## get the river network shapefile, in this case the edges information in
## demoNet
demoNet.path <- system.file("demoSSN/demoNet.ssn", package="stpca")
river.path <- paste(demoNet.path, "edges.shp", sep="/")
river <- readShapeLines(river.path)
demoNet <- importSSN(demoNet.path)
data1 <- getSSNdata.frame(demoNet, "Obs")
coords <- c("NEAR_X", "NEAR_Y")
coords <- data1[,coords]

## plot the results

# unweighted T-mode PCA
plot_stpca(x = Tmode.pca.uw, plots = "map", coords = coords,
           river = river, pc.map = 1, probs = seq(0, 1, by=0.5))
plot_stpca(x = Tmode.pca.uw, plots = "biplot")
plot_stpca(x = Tmode.pca.uw, plots = "glyph", pc.glyph=1:2,
           river = river, coords = coords, r1=5)


# spatial, temporal, and spatiotemporal weighted S-mode PCA
plot_stpca(x = Smode.pca.all, plots = "map", coords = coords,
           river = river)
plot_stpca(x = Smode.pca.all, plots = "biplot")
plot_stpca(x = Smode.pca.all, plots = "glyph", river = river,
           coords = coords)
plot_stpca(x = Smode.pca.all, plots = "ts", pc.ts = 1:3)
plot_stpca(x = Smode.pca.all, plots = "meanPM")
```

sampDemo *Sampled networks from* demoNet

Description

Results from applying sampNet to demoNet to illustrate plot_sampNet.

Usage

```
sampDemo
```

Format

sampNet was applied to the data from demoNet using the following options:

First, set the seed as shown below, and get the data:

```
set.seed(618999)
obs.data <- getSSNdata.frame(demoNet, "Obs")
preds.coords <- getSSNdata.frame(demoNet, "preds")
```

Next, run `sampDemo <- sampNet()` with the following arguments:

- `x = obs.data`
- `nsim = 5`
- `prop.sites = c(0.8, 0.7)`
- `samp.scheme = c("random", "stratified", "weighted")`
- `data.col = "Sim_Values"`
- `coords.col = c(9, 10)`
- `preds.coords = preds.coords[,c("NEAR_X", "NEAR_Y")]`
- `siteID = "locID"`
- `strat.col = "strata"`
- `weight.col = "weight"`

The `sampDemo` object contains results for random, weighted and stratified sampling schemes where 70% and 80% of monitoring sites were retained. Five sampled networks were created for each `prop.sites` and `samp.scheme` combination.

Source

See vignette for Creating Demo Data

`sampNet` *Create sampled networks.*

Description

Investigate the effect of reducing the size of the monitoring network on covariance function parameters and predictions based on reduced networks. This function allows you to sample networks using random, stratified, and weighted sampling schemes. This function relies heavily on the SSN, geoR (http://www.leg.ufpr.br/geoR), and sampling (https://cran.r-project.org/web/packages/sampling/sampling.pdf) packages.

Usage

```
sampNet(x, nsim, prop.sites, samp.scheme = c("random"), sub.filepath, predpts,
  response.col, addfunccol, siteID, strat.col, weight.col, coords.col, data.col,
  preds.coords, print.sim = TRUE)
```

Arguments

`x`	an object of class `"data.frame"` or `SpatialStreamNetwork-class`
`nsim`	integer. Specifies the number of sampled networks to be created.
`prop.sites`	numeric. Specifies a vector containing the proportions of monitoring sites to retain. Must only take values between 0 and 1.
`samp.scheme`	character vector. Specifies the type of sampling scheme to use. Can include "random", "stratified", or "weighted". See Details for more information.
`sub.filepath`	character string. Specifies the filepath to temporarily store sampled networks. Only used if `x` is a `SpatialStreamNetwork-class` object. See Details for further information.
`predpts`	character string. Specifies the name of the dataframe in the SSN object containing prediction locations. Only used if `x` is a `SpatialStreamNetwork-class` object.
`response.col`	character string. Specifies the name of the column in the dataframe of observed values in the SSN object that contains the response variable. Only used if `x` is a `SpatialStreamNetwork-class` object
`addfunccol`	character string. Contains the name of the column in the dataframe of observed values in the SSN objcet containing the additive function values. Only used if `x` is a `SpatialStreamNetwork-class` object.
`siteID`	character string. Specifies the name of the column in the data.frame or SSN object that contains monitoring site ID's.
`strat.col`	character string. Specifies the name of the column in the data.frame or SSN object that contains the stratum information. Only used when stratified sampling scheme is implemented. See Details for further information.
`weight.col`	character string. Specifies the name of the column in the data.frame or SSN object that contains the values used for weighted sampling. Only used when weighted sampling scheme is implemented. See Details for further information.
`coords.col`	a vector with the column numbers corresponding to the spatial coordinates. Only used if `x` is of class `"data.frame"`.
`data.col`	a scalar with column number corresponding to the column in which the observed data values are stored. Only used if `x` is of class `"data.frame"`
`preds.coords`	an N x 2 matrix or data-frame with the 2-D coordinates of the N prediction location. Only used if `x` is of class `"data.frame"`
`print.sim`	logical. If TRUE (default) then a message will be printed after each sampled network is created. See Details for further information.

Details

This function creates `nsim` sampled networks and fits a model with a spatial covariance function to each sampled network. Various model outputs and predicted values based on the spatial covariance function are stored.

If data are from a SpatialStreamNetwork-class object the spatial covariance model with Epanechnikov Tail-up, Gaussian Euclidean, and nugget components is fitted to each sampled network using glmssn from the SSN package. If the data are from a data.frame then an exponential spatial covariance model is fitted using likfit in the geoR package. For both types of data, the spatial covariance model is fitted using REML.

If the data are from a data.frame then initialization values are required to fit the spatial covariance model. The variog and variofit functions from the geoR package are used to estimate the initialization values using all default options, and specifying cov.model = "exponential" in variofit.

The spatial covariance model is fitted assuming no spatial trend and so for the glmssn model, this is equivalent of formula = response.col ~ 1. For the likfit model, this is equivalent to specifying trend = "cte".

The number of monitoring sites in a sampled network is equal to the sum of sites to be included in each stratum if "stratified" is included in samp.scheme. Otherwise, the number of monitoring sites in a sampled network is ceiling(nrow(data) * prop.sites) where data is the name of the dataframe. The number of monitoring sites for each value in prop.sites is calculated using some rounding and so there can be differences between the total number of monitoring sites in "stratified" and "random" for the same prop.sites value. Because of this, the number of sites to be included in a sampled network is set to be the total included in the stratified samples so that the same number of sites are included for all sampling schemes. This allows comparisons to be made between sampling schemes.

samp.scheme is a character vector and can take any combination of "random" for random sampling, "stratified" for stratified sampling, or "weighted" for weighted sampling. In "random" sampling, each monitoring site has the same probability of being included in the sampled network. For "stratified" sampling, proportional stratification is used so that the sampled network has the same proportion of sites within each strata as the full network. strat.col is the column in the data object containing the variable used for stratification. This column should be a factor variable. For "weighted" sampling, a column should be included in the data object containing weights in the range [0,1] reflecting the probability of a monitoring site being included in the sampled network. If weights are not in the range [0,1] then the function will stop running and a warning message is produced saying that weight.col does not contain appropriate values. It is recommended that if any of the weights are 0 then a small constant (0.001 say) is added, unless the user really does not want this monitoring site included in the sampled network. Likewise, if any of the weights are 1 then a small constant should be subtracted, unless the user wants this site to be included in every sampled network.

sub.filepath is a location to temporarily store sampled networks. For each nsim, a new SpatialStreamNetwork-clas object is created containing the monitoring sites included in the sampled network. A new SSN object is created and stored in sub.filepath for each nsim and removed once the SSN model has been fitted and various outputs calculated and stored.

print.sim can be switched off if you do not require a message to be printed after each sampled network is created. The message will show you which comination of ‘samp.scheme‘ 100*‘prop.sites‘ ‘nsim‘ has just been evaluated. The other part of the message that is printed is a result of importing the sampled SSN object using importSSN within the sampNet function (this part cannot be suppressed).

Value

Returns a list with results for each combination of samp.scheme and prop.sites. The name of each list entry contains the sampling scheme and percentage of sites retained in the sampled networks. For example "random.50" means that samp.scheme = "random" and prop.sites = 0.5. Each of these will contain the following:

samp.site is a matrix with nsim columns, each of which contains the ID's of monitoring sites included in each sampled network.

cov.params is a matrix with nsim rows and 5 columns if x is an SSN object: Tail-up partial sill, Tail-up range, Euclidean partial sill, Euclidean partial range, nugget. If x is of class "data.frame" then this will have 3 columns: range, partial sill, and nugget.

crossval.pred is a matrix with nsim columns and the number of rows is the number of monitoring sites in each sampled network. Values are predicted values from leave one out cross validation where the spatial covariance model is fitted to all of the monitoring sites except one, and the response at the omitted site is estimated from the fitted model. See CrossValidationSSN if x is an SSN object or xvalid if x is of class "data.frame" for further details.

crossval.se is a matrix with nsim columns and number of rows is equal to the number of monitoring sites in a sampled network. Values are the standard errors of crossval.pred.

cross.stats a matrix containing root mean square prediction error (RMSPE). RMSPE is calculated as:

$$\sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\hat{Y}(s_i) - Y(s_i)\right)^2}$$

where N is the number of monitoring sites, $\hat{Y}(s_i)$ is the value at site i predicted from a model fitted to s$_{-i}$, data from all monitoring sites except site i, and $Y(s_i)$ is the observed value at site i. This is also known as Leave One Out Cross Validation (LOOCV). This is a measure of how well the model fits the data. Low prediction error is desireable.

preds.value is a matrix with nsim columns and number of rows equal to the number of prediction locations. Values are predicted response at unobserved locations based on the spatial covariance function estimated for sampled network and were obtained using predict.glmssn if x is an SSN object, or xvalid if x is of class "data.frame".

preds.se is a matrix with nsim columns and number of rows equal to the number of prediction locations. Values are standard error of predicted values for preds.value. See predict.glmssn if x is an SSN object, or xvalid if x is of class "data.frame" for further details. (Note that if x is of class "data.frame" then the value here is the square root of krige.var from krige.conv).

Other results, all of which are used in plot_sampNet include:

results.names a character string containing all combinations of samp.scheme and prop.sites.

prop.sites is the vector of values entered for the prop.sites argument.

samp.scheme is the vector of values entered for the samp.scheme argument.

all contains various outputs related to fitting a spatial covariance model to the full monitoring network. Outputs are model details (all.model), covariance function parameter estimates (all.covparams), cross validation statistics (all.cross.stats), and predicted values with corresponding standard errors (all.preds).

columns is a vector containing response.col and the column containing standard errors of predicted values.

call is a list containing the argument values specified in the function.

References

geoR:

Paulo J. Ribeiro Jr and Peter J. Diggle (2015). geoR: Analysis of Geostatistical Data. Rpackage version 1.7-5.1. https://CRAN.R-project.org/package=geoR

`sampling`:

Yves Tille and Alina Matei (2015). sampling: Survey Sampling. R package version 2.7.https://CRAN.R-project.org/package=sampling

`SSN`:

Ver Hoef, J. M. and Peterson, E. E. (2010) A moving average approach for spatial statistical models of stream networks (with discussion). Journal of the American Statistical Association 105:6-18. http://dx.doi.org/10.1198/jasa.2009.ap08248

Jay M. Ver Hoef, Erin E. Peterson, David Clifford, Rohan Shah (2014). SSN: An R Package for Spatial Statistical Modeling on Stream Networks. Journal of Statistical Software, 56(3), 1-43. http://www.jstatsoft.org/v56/i03/.

Examples

```
library(stpca)
library(SSN)

## get the data
x <- importSSN(system.file("demoSSN/demoNet.ssn", package = "stpca"),
               predpts = "preds")

## x is an SSN object
test.samp <- sampNet(x = x, nsim = 3, prop.sites = c(0.5),
                     samp.scheme = c("random"),
                     sub.filepath = paste(tempdir(),"/subset1.ssn", sep = ""),
                     predpts = "preds", response.col = "Sim_Values",
                     addfunccol = "addfunccol", siteID = "locID")


## x is of class "data.frame"
## in order to demonstrate this we need to first extract
## the data from the SSN object so that the data are stored
## in a data.frame.  sampNet() is then applied to data in a
## data.frame rather than an SSN object as in the previous
## example.

# get the data.frame containing observed values
data(demoNet)
x <- getSSNdata.frame(demoNet, "Obs")

# get the data.frame containing prediction locations
preds.coords <- getSSNdata.frame(demoNet, "preds")[,c("NEAR_X", "NEAR_Y")]

test.samp <- sampNet(x, nsim = 5, prop.sites = c(0.8, 0.7),
                     samp.scheme = c("random", "stratified", "weighted"),
                     data.col = "Sim_Values", coords.col = c(9, 10),
                     preds.coords = preds.coords, siteID = "locID",
                     strat.col = "strata", weight.col = "weight")
```

`stpca` *Perform PCA to identify common spatiotemporal patterns, adjusting for spatial and/or temporal autocorrelation where appropriate.*

Description

PCA is performed on spatiotemporal data in T-mode or S-mode using singular value decomposition and can be adjusted for spatial and/or temporal autocorrelation if appropriate. If spatial or temporal weights are used then the returned loadings and principal components are backtransformed using transformation described in paper (being written).

Usage

```
stpca(x, center = TRUE, scale = FALSE, pca.mode = NULL,
  pca.wt = c("unweighted"), spatial.wt, temporal.wt, scree = TRUE, k = 10)
```

Arguments

`x`	data.frame. This should be a complete data set with no missing values. Columns MUST be monitoring sites while rows MUST be time points. See Details for further information.
`center`	logical. TRUE (default) will subtract column means from each column of the `x`.
`scale`	logical. TRUE will divide values in `x` by the column standard deviation. Default is FALSE. See Details for further information.
`pca.mode`	character string. MUST be specified. Options are "Smode" or "T-mode". See Details for further information.
`pca.wt`	character vector. "unweighted" (default) performs standard PCA, "spatial" adjusts for spatial autocorrelation, "temporal" adjusts for temporal autocorrelation, "spatiotemporal" adjusts for both spatial and temporal autocorrelation. See Details for further information.
`spatial.wt`	character string. Specifies the name of the matrix containing spatial weights. Only used if `pca.wt` contains "spatial" or "spatiotemporal". Can be created by the user or using `createWeightS`.
`temporal.wt`	character string. Specifies the name of the matrix containing temporal weights. Only used if `pca.wt` contains "temporal" or "spatiotemporal". Can be created by the user or using `createWeightT`.
`scree`	logical. TRUE (default) means a scree plot will be produced for the first k principal components.
`k`	integer. The number of principal components to use when producing scree plot. Only used if `scree = TRUE`. Default is 10.

Details

`x` must always have columns = monitoring sites and rows = time points, regardless of the analysis the user intends to perform. If `pca.mode = "Tmode"` is specified, the function will automatically transpose the data.

`x` If column names in `spatial.wt` are not the same or in a different order from `rownames(x)` then an error message is produced.

`scale` If TRUE then this is equivalent to performing PCA on the correlation matrix. If FALSE then this is equivalent to performing PCA on the covariance matrix.

`pca.mode` PCA can be applied to spatiotemporal data in either S-mode or T-mode. In S-mode, the columns of `x` are monitoring sites while the rows are time points. In T-mode, the columns of `x` are time points while the rows are monitoring sites.

`spatial.wt` The column names of the spatial weights matrix must be of the same form and in the same order as `rownames(x)`. An error is produced if these do no match. The user should order `x` and set `rownames(x)` to correspond to the ordering in `spatial.wt`.

Value

The function returns a list containing the following: Lists called codeunweighted, `spatial`, `temporal`, `spatiotemporal` are produced depending on what was included in `pca.wt`. Each of these contains `scores` = principal components, `loads` = loadings, `var` = variance of each principal component, `var.prop` = proportion of total variance explained by each principal component, `cumvar` = cumulative proportion of total variance explained.

`pca.mode` contains the option used for `pca.mode`.

`pca.wt` contains the option used for `pca.wt`.

`call` contains the function call details.

`row.names` contains the row names of the data.frame on which PCA was performed. This will be row names of `x` if `pca.mode = "Smode"` or column names of `x` if `pca.mode = "Tmode"`.

`col.names` contains the column names of the data.frame on which PCA was performed. This will be column names of `x` if `pca.mode = "Smode"` or row names of `x` if `pca.mode = "Tmode"`.

`var.value` contains `var` from each of `unweighted`, `spatial`, `temporal` and `spatiotemporal` combined into a single matrix.

`mean.timeseries` contains the average time series accross all sites. This is stored for use when `plots="meanPM"` in `plot_stpca` and is calculated as `apply(x, 1, mean)`.

A scree plot of the first `k` principal components is produced if `scree = TRUE`, with the variance of each component on the y-axis. The text labels on the plot show the percentage variance explained by each principal component.

Examples

```
library(stpca)

## get data.frame with no missing values
data(demoY)

## unweighted T-mode PCA
Tmode.pca.uw <- stpca(x=demoY, pca.mode="Tmode", pca.wt=c("unweighted"))

## spatial, temporal, and spatiotemporal weighted S-mode PCA

# create spatial weights
ssndata <- system.file("demoSSN/demoNet.ssn", package="stpca")
weightS <- createWeightS(ssndata=ssndata, afvcol="addfunccol")

# create temporal weights
weightT <- createWeightT(n=nrow(demoY), rho=0.5)

# check that the order of monitoring sites in x is the same as the
# columns in spatial.wt.  They are the same if the following results
# in 1.
mean(colnames(demoY) == colnames(weightS))

# weighted S-mode PCA
Smode.pca.all <- stpca(x = demoY, pca.mode = "Smode",
                       spatial.wt = weightS,
                       temporal.wt = weightT,
                       pca.wt = c("unweighted", "spatial",
                                  "temporal", "spatiotemporal"))
```

Index