Package ‘starma’

February 11, 2016

Type Package

Title Modelling Space Time AutoRegressive Moving Average (STARMA) Processes

Version 1.3

Date 2016-02-11

Author Felix Cheysson

Maintainer Felix Cheysson <felix@cheysson.fr>

Description Statistical functions to identify, estimate and diagnose a Space-Time AutoRegressive Moving Average (STARMA) model.

License GPL-2

Imports Rcpp (>= 0.11.1), ggplot2, scales, graphics, stats

LinkingTo Rcpp, RcppArmadillo

Depends

Suggests spdep

NeedsCompilation yes

Repository CRAN

Date/Publication 2016-02-11 11:30:58

R topics documented:

`starma-package` *Space Time AutoRegressive Moving Average (STARMA) modelling for space-time series*

Description

This package aims to provide all the tools needed to identify, estimate and diagnose STARMA models for space-time series. It follows the three-stage iterative model building procedure developed by (Box and Jenkins, 1970) and extended to space-time modelling by (Pfeifer and Deutsch, 1980). Designed with large datasets in mind, the package has been optimized by integrating C++ code via Rcpp and RcppArmadillo (Eddelbuettel and Sanderson, 2014). Furthermore, the parameter estimation, which is usually computationally very expensive when using common optimization routines, uses a Kalman filter (see Cipra and Motykova, 1987), making it extremely efficient when dealing with large datasets.

Details

Package:	starma
Type:	Package
Version:	1.2
Date:	2015-11-12
License:	GPL-2

The three stages of the iterative model building procedure are as follow, after centering the space-time series with `stcenter`:

- Identification: Using `stacf` and `stpacf`, the user should try to identify which parameters should be estimated.

- Estimation: Use `starma` to estimate the parameters.

- Diagnostic: Use `stacf`, `stpacf` and `stcor.test` to check whether the residuals of the models are similar to white noise.

Refer to (Box and Jenkins, 1970) for details over the three-stage iterative model building procedure.

Author(s)

Felix Cheysson

Maintainer: Felix Cheysson <felix@cheysson.fr>

References

- Box, G. E. P., & Jenkins, G. M. (1970). Time Series Analysis: Forecasting and Control. Holden Day.

- Pfeifer, P., & Deutsch, S. (1980). A Three-Stage Iterative Procedure for Space-Time Modeling. Technometrics, 22(1), 35-47. doi:10.1080/00401706.1980.10486099

- Pfeifer, P., & Deutsch, S. (1981). Variance of the Sample Space-Time Autocorrelation Function. Journal of the Royal Statistical Society. Series B (Methodological), 43(1): 28-33.

- Cipra, T., & Motykova, I. (1987). Study on Kalman filter in time series analysis. Commentationes Mathematicae Universitatis Carolinae, 28(3).

- Dirk Eddelbuettel, Conrad Sanderson (2014). RcppArmadillo: Accelerating R with high-performance C++ linear algebra. Computational Statistics and Data Analysis, Volume 71, March 2014, pages 1054-1063. URL http://dx.doi.org/10.1016/j.csda.2013.02.005

Examples

```
# Load spdep library to easily create weight matrices
library(spdep)

# Create a 5x5 regular grid which will be our lattice
sites <- matrix(0, 25, 2)
for (i in 1:5) {
for (j in 1:5)
sites[(i-1)*5 + j, ] <- c(i, j) - .5
}
plot(sites)

# Create a uniform first order neighbourhood
knb <- dnearneigh(sites, 0, 1)
plot(knb, sites)

# Lag the neighbourhood to create other order matrices
knb <- nblag(knb, 4)
klist <- list(order0=diag(25),
           order1=nb2mat(knb[[1]]),
           order2=nb2mat(knb[[2]]),
           order3=nb2mat(knb[[3]]),
           order4=nb2mat(knb[[4]]))

# Simulate a STARMA(2;1) process
eps <- matrix(rnorm(200*25), 200, 25)
star <- eps
for (t in 3:200) {
star[t,] <- (.4*klist[[1]] + .25*klist[[2]]) %*% star[t-1,] +
(.25*klist[[1]]                ) %*% star[t-2,] +
(            - .3*klist[[2]]) %*% eps[t-1,] +
eps[t, ]
}

star <- star[101:200,] # Remove first observations
star <- stcenter(star) # Center and scale the dataset

# Identify the process
stacf(star, klist)
stpacf(star, klist)

# Estimate the process
ar <- matrix(c(1, 1, 1, 0), 2, 2)
```

```
ma <- matrix(c(0, 1), 1, 2)
model <- starma(star, klist, ar, ma)
model
summary(model)

# Diagnose the process
stcor.test(model$residuals, klist, fitdf=4)
stacf(model$residuals, klist)
stpacf(model$residuals, klist)
```

nb_mat *Neighbourhood weight matrices for France's 94 departments*

Description

This data file provides three neighbourhoods for the 94 metropolitan French departments:

- `dlist`: distance-based neighbourhoods; two departments are considered neighbours if their centroids are within range of 100km.

- `klist`: four closest neighbours; each department is connected to its four closest neighbours, the distance being calculated between centroids.

- `blist`: common border neighbours; two departments are considered neighbours if they share a border.

These neighbourhoods are designed to be used within the `starma-package`. First element is the identity matrix (0-th order neighbours). Second element is the common border contingency matrix of the department (1-st order neighbours). Elements three to five are the weight matrices lagged from the previous one (2-nd to 4-th order neighbours).

They have been computed used the package spdep and its functions `readShapePoly`, `poly2nb` and `nblag`.

Usage

```
nb_mat
```

Format

Three lists of 5 weight matrices, of dimension 94x94

stacf *Space-time autocorrelation functions*

Description

The functions defined below are the main tools to the identification and the diagnostic part of the three-stage iterative model procedure building. `stacf` and `stpacf` respectively compute the autocorrelation and partial autocorrelation functions of a space-time series.

Usage

```
stacf(data, wlist, tlag.max=NULL, plot=TRUE, use.ggplot=TRUE)
stpacf(data, wlist, tlag.max=NULL, plot=TRUE, use.ggplot=TRUE)
```

Arguments

`data`	a matrix or data frame containing the space-time series: row-wise should be the temporal observations, with each column corresponding to a site.
`wlist`	a list of the weight matrices for each k-th order neighbours, first one being the identity.
`tlag.max`	the maximum time lag for the space-time autocorrelation functions. If `tlag.max = NULL`, it will use a large enough number of time lags.
`plot`	whether to plot the autocorrelation functions or not.
`use.ggplot`	if `plot = TRUE`, whether to use `ggplot2` or not to display the autocorrelation functions. Not using `ggplot2` is depreciated.

Details

`stacf` and `stpacf` respectively compute the space-time autocorrelation and partial autocorrelation functions of the serie `data` between `s`-th and `0`-th order neighbors at time lag `t`, for `s` ranging from `0` to `length(wlist)` and `t` ranging from `1` to `tlag.max`.

The autocorrelation function is computed as follows:

$$\hat{\rho}_l(s) = frac\hat{\gamma}_{l0}(s)(\hat{\gamma}_{ll}(0)\hat{\gamma}_{00}(s))^{1/2}$$

The partial autocorrelation functions are computed solving iteratively the Yule Walker equations for increasing time lags and space lags.

Note that the identification might be biased if the partial autocorrelation functions are not computed with enough space lags, since Yule Walker equations are sensible to the maximum space lag given.

Value

An object of class `matrix` containing the estimated acf. Row-wise are the different time lags, column-wise the different space lags.

Author(s)

Felix Cheysson

References

Pfeifer, P., & Deutsch, S. (1980). A Three-Stage Iterative Procedure for Space-Time Modeling. Technometrics, 22(1), 35-47. doi:10.1080/00401706.1980.10486099

Examples

```
data(nb_mat) # Get neighbourhood matrices

# Simulate a STARMA model
eps <- matrix(rnorm(94*200), 200, 94)
sim <- eps
for (t in 3:200) {
sim[t,] <- (.4*blist[[1]] + .25*blist[[2]]) %*% sim[t-1,] +
(.25*blist[[1]]                ) %*% sim[t-2,] +
(            - .3*blist[[2]]) %*% eps[t-1,] +
eps[t, ]
}

sim <- sim[101:200,]
sim <- stcenter(sim) # Center and scale the dataset

# Plot stacf and stpacf
stacf(sim, blist)
stpacf(sim, blist)
```

---

starma *Space-time series estimation procedure*

Description

starma fits a STARMA model to a space-time series. It is the central function for the estimation part of the three-stage iterative model building procedure.

Usage

```
starma(data, wlist, ar, ma, iterate=1)

## S3 method for class 'starma'
print(x, ...)
```

Arguments

`data`	a matrix or data frame containing the space-time series: row-wise should be the temporal observations, with each column corresponding to a site.
`wlist`	a list of the weight matrices for each k-th order neighbours, first one being the identity.
`ar`	either an integer specifying the maximum time lag of the AR part, or a matrix filled with 0 or 1 indicating whether 'row'-th tlag, 'col'-th slag AR parameter should be estimated.
`ma`	either an integer specifying the maximum time lag of the MA part, or a matrix filled with 0 or 1 indicating whether 'row'-th tlag, 'col'-th slag MA parameter should be estimated.
`iterate`	an integer specifying how many times the Kalman filter should be re-run on itself (see Details).
`x`	a `starma` class object.
`...`	unused

Details

The definition here used for STARMA models is the following:

$$z_t = \sum_{k=1}^{p} \sum_{l=0}^{\lambda_k} \phi_{kl} W^{(l)} z_{t-k} + \sum_{k=1}^{q} \sum_{l=0}^{m_k} \theta_{kl} W^{(l)} \epsilon_{t-k} + \epsilon_t$$

`starma` uses a Kalman filter algorithm (Cipra and Motykova, 1987): the parameters are set as the state vector of the state space system, making the iterations of the algorithm estimate directly the parameters. Thus, no optimization routine is required, making the algorithm extremely efficient time wise and computationally wise. Furthermore, the code has been written in C++ using Rcpp and RcppArmadillo (Eddelbuettel and Sanderson, 2014).

Note that, as the residuals must be iteratively estimated when running the Kalman filter, a single run might lead to poor results when estimating an MA parameter. Re-running the Kalman filter at least once, using the previously estimated parameters to add prior knowledge on the residuals leads to better estimates. For STAR model (when no MA parameter needs be estimated), the function ignores the `iterate` argument.

One of the strength of this estimation function is that the user can to estimate as few parameters as needed, even at high time and/or space lags, since the possibility to input a 1/0 matrix as AR and MA orders is given.

Value

A list of class `starma` containing the following elements:

`phi`	The estimated AR parameters
`phi_sd`	The corresponding standard errors
`theta`	The estimated MA parameters
`theta_sd`	The corresponding standard errors

`sigma2`	The white noise variance matrix estimated by the Kalman filter. Note that, to achieve parcimony, only the mean of the diagonal elements should be kept (since the noise is supposed to be Gaussian anyway)
`residuals`	The estimated residuals of the model
`loglik`	The conditional log likelihood of the model
`bic`	The corresponding BIC
`call`	The function call
`df`	Degrees of freedom of the model: (nb of obs) - (nb of parameters)

Author(s)

Felix Cheysson

References

- Cipra, T., & Motykova, I. (1987). Study on Kalman filter in time series analysis. Commentationes Mathematicae Universitatis Carolinae, 28(3).

- Dirk Eddelbuettel, Conrad Sanderson (2014). RcppArmadillo: Accelerating R with high-performance C++ linear algebra. Computational Statistics and Data Analysis, Volume 71, March 2014, pages 1054-1063. URL http://dx.doi.org/10.1016/j.csda.2013.02.005

Examples

```
data(nb_mat) # Get neighbourhood matrices

# Simulate a STARMA model
eps <- matrix(rnorm(94*200), 200, 94)
sim <- eps
for (t in 3:200) {
sim[t,] <- (.4*diag(94) + .25*blist[[2]]) %*% sim[t-1,] +
(.25*diag(94)                ) %*% sim[t-2,] +
(            - .3*blist[[2]]) %*% eps[t-1,] +
eps[t, ]
}

sim <- sim[101:200,]
sim <- stcenter(sim) # Center and scale the dataset

# Autocorrelation functions
stacf(sim, blist)
stpacf(sim, blist)

# Select parameters to estimate
ar <- matrix(0, 2, 2)
ar[ ,1] <- 1 # phi10 and phi20
ar[1,2] <- 1 # phi11
ma <- matrix(c(0,1), 1, 2) # theta11

# Run the Kalman filter algorithm
model <- starma(sim, blist, ar, ma)
```

```
summary(model)
```

`stcenter` *Space-time centering and scaling function*

Description

`stcenter` centers and scales the space-time series `data` such that its mean is 0 and its standard error 1.

Usage

```
stcenter(data, center=TRUE, scale=TRUE)
```

Arguments

`data`	a matrix or data frame containing the space-time series: row-wise should be the temporal observations, with each column corresponding to a site.
`center`	a logical value indicating whether the series should be centered or not (subtracting the mean).
`scale`	a logical value indicating whether the series should be scaled or not (dividing by the empiric stand deviation).

Details

To be able to apply the three-stage iterative model building procedure method for STARMA models, data must be centered beforehand (since `starma` doesn't estimate an intercept coefficient).

The only difference with the R function `scale` is that it doesn't center and scale column by column, but globally, since all the observations come from a single process in the case of space time series.

Value

An object of the same class as `data`, that is either a `matrix` or a `data.frame`.

Author(s)

Felix Cheysson

Examples

```
data <- matrix(rnorm(9400, mean=5, sd=2), 100, 94)
data <- stcenter(data) # Center and scale the dataset

# Check for mean
sum(data) / (nrow(data) * ncol(data))

# Check for sd
sqrt( sum(data^2) / (nrow(data) * ncol(data) - 1) )
```

`stcor.test` *Space-time series non correlation test*

Description

`stcor.test` computes an extension of the Box-Pierce test statistic to accept or reject the non correlation of the distinct observations of a given space-time series. It is one of the main functions for the diagnostic part of the three-stage iterative model building procedure.

Usage

```
stcor.test(data, wlist, tlag=NULL, slag=NULL, fitdf=0)

## S3 method for class 'stcor.test'
print(x, ...)
```

Arguments

`data`	a matrix or data frame containing the space-time series: row-wise should be the temporal observations, with each column corresponding to a site.
`wlist`	a list of the weight matrices for each k-th order neighbours, first one being the identity.
`tlag`	the maximum time lag for the space-time autocorrelation functions. If `tlag = NULL`, it will use a large enough number of time lags.
`slag`	the maximum space lag for the space-time autocorrelation functions. If `slag = NULL`, it will use as many space lags as possible (as many as `length(wlist)`).
`fitdf`	number of degrees of freedom to be subtracted if `data` is a series of residuals.
`x`	a `starma` class object.
`...`	unused

Details

Since (Pfeifer and Deutsch, 1981) gives:

$$Var(\hat{\rho}_l(s)) \approx \frac{1}{N(T-s)}$$

We can extend Box-Pierce test statistic to space-time series:

$$N\sum(T-s)\hat{\rho}_l(s)^2 \sim \chi^2(slag \times tlag)$$

`stcor.test` can be applied to a space-time series to test the null hypothesis of non correlation.

It is useful to check if the residuals of a STARMA models are multivariate white noise. In this case, `fitdf` should be set equal to the number of parameters in the model.

Please note that this is an empirical extension and it has not yet been the subject of a paper. The specifications of the weight matrices has not been studied either and could lead to inconsistencies.

Value

A `data.frame` containing the following elements:

`X_squared`	The value of the chi squared statistic
`df`	The degrees of freedom of the statistic (taking `fitdf` into account)
`p.value`	The p-value of the test

Author(s)

Felix Cheysson

References

- Pfeifer, P., & Deutsch, S. (1980). A Three-Stage Iterative Procedure for Space-Time Modeling. Technometrics, 22(1): 35-47. doi:10.1080/00401706.1980.10486099

- Pfeifer, P., & Deutsch, S. (1981). Variance of the Sample Space-Time Autocorrelation Function. Journal of the Royal Statistical Society. Series B (Methodological), 43(1): 28-33.

Examples

```
data(nb_mat)

eps <- matrix(rnorm(94*200), 200, 94)
sim <- eps
for (t in 3:200) {
sim[t,] <- (.4*blist[[1]] + .25*blist[[2]]) %*% sim[t-1,] +
(.25*blist[[1]]                ) %*% sim[t-2,] +
(            - .3*blist[[2]]) %*% eps[t-1,] +
eps[t, ]
}

sim <- sim[101:200,]
sim <- stcenter(sim) # Center and scale the dataset

# Test for multivariate normality
stcor.test(sim, blist) # Data is correlated
stcor.test(eps, blist) # Data should not be correlated (unless you're 5% unlucky)
```

`stcov` *Space-time covariance function*

Description

`stcov` computes the space-time covariance of the serie `data` between `slag1`-th and `slag2`-th order neighbours at time lag `tlag`.

Usage

```
stcov(data, wlist, slag1, slag2, tlag)
```

Arguments

`data`	a matrix or data frame containing the space-time series: row-wise should be the temporal observations, with each column corresponding to a site.
`wlist`	a list of the weight matrices for each k-th order neighbours, first one being the identity.
`slag1, slag2`	the space lags for the space-time covariance.
`tlag`	the time lag for the space-time covariance.

Details

`stcov` is mainly used as an internal function for the computation of `stacf` and `stpacf`. `slag1` and `slag2` must be lower than `length(wlist)`.

It is computed as follows:

$$\hat{\gamma}_{lk}(s) = \frac{1}{N(T-s)} Tr \left(\sum_{t=s+1}^{T} W^{(k)\prime} W^{(l)} z_t z_{t-k}^{\prime} \right)$$

Value

A numeric.

Author(s)

Felix Cheysson

References

Pfeifer, P., & Deutsch, S. (1980). A Three-Stage Iterative Procedure for Space-Time Modeling. Technometrics, 22(1), 35-47. doi:10.1080/00401706.1980.10486099

Examples

```
data(nb_mat) # Get neighbourhood matrices

data <- matrix(rnorm(9400), 100, 94)

# Compute covariance between 2-nd and 1-st order neighbours, at time lag 5
stcov(data, blist, 2, 1, 5)
```

`stplot` *Plot for space-time series autocorrelation functions*

Description

`stplot` renders a nice 2d plot for autocorrelation functions.

Usage

```
stplot(acf, ci, call, ggplot=T)
```

Arguments

`acf`	a matrix containing the autocorrelation functions of a given space-time series: row-wise should be the temporal observations, with each column corresponding to a space lag.
`ci`	confidence intervals for the autocorrelation functions.
`call`	the name of the plot.
`ggplot`	a boolean indicating whether to use ggplot2 functions (they are recommended).

Details

This function plots the calculated autocorrelation functions of a space-time series.

In practice, the user should not use this function, as it is being called automatically when using `stacf` or `stpacf`. The confidence intervals for the autocorrelation functions are approximated by

$$Var\left(\hat{\rho}_l(k)\right) \approx \frac{1}{N(T-k)}$$

where N is the number of sites, and T the number of temporal observations.

Value

NULL

Author(s)

Felix Cheysson

References

- Pfeifer, P., & Deutsch, S. (1981). Variance of the Sample Space-Time Autocorrelation Function. Journal of the Royal Statistical Society. Series B (Methodological), 43(1): 28-33.

Examples

```
data(nb_mat) # Get neighbourhood matrices

# Simulate a STARMA model
eps <- matrix(rnorm(94*200), 200, 94)
sim <- eps
for (t in 3:200) {
sim[t,] <- (.4*diag(94) + .25*blist[[2]]) %*% sim[t-1,] +
(.25*diag(94)                ) %*% sim[t-2,] +
(            - .3*blist[[2]]) %*% eps[t-1,] +
eps[t, ]
}

sim <- sim[101:200,]
sim <- stcenter(sim) # Center and scale the dataset

# Autocorrelation functions
sim.stacf <- stacf(sim, blist, plot=FALSE)

# Plot the autocorrelation function
stplot(sim.stacf, 2 / sqrt(nrow(sim) * ncol(sim)), "stacf(sim, blist)")
```

`summary.starma` *Summary method for space-time series fitted models*

Description

summary method for class "starma".

Usage

```
## S3 method for class 'starma'
summary(object, ...)
## S3 method for class 'summary.starma'
print(x, ...)
```

Arguments

object	a starma class object.
x	a summary.starma class object.
...	unused

Details

print.summary.starma formats the coefficients, standard errors, etc. and additionally gives 'significance stars'.

Value

An object of class `summary.starma` containing the following elements:

`call`	An object of mode "`call`": a symbolic description of the fitted model
`coefficients`	A data frame containing the estimates, standard errors, etc. of the coefficients of the fitted model

Author(s)

Felix Cheysson

Examples

```
data(nb_mat) # Get neighbourhood matrices

# Simulate a STARMA model
eps <- matrix(rnorm(94*200), 200, 94)
sim <- eps
for (t in 3:200) {
sim[t,] <- (.4*diag(94) + .25*blist[[2]]) %*% sim[t-1,] +
(.25*diag(94)               ) %*% sim[t-2,] +
(           - .3*blist[[2]]) %*% eps[t-1,] +
eps[t, ]
}

sim <- sim[101:200,]
sim <- stcenter(sim) # Center and scale the dataset

# Select parameters to estimate
ar <- matrix(0, 2, 2)
ar[ ,1] <- 1 # phi10 and phi20
ar[1,2] <- 1 # phi11
ma <- matrix(c(0,1), 1, 2) # theta11

# Run the Kalman filter algorithm
model <- starma(sim, blist, ar, ma)

# Get summary
summary(model)
```

Index