Package ‘kalmanfilter’

September 2, 2022

Type Package

Title Kalman Filter

Version 1.0.0

Date 2022-09-01

Description 'Rcpp' implementation of the multivariate Kalman filter for state space models that can handle missing values and exogenous data in the observation and state equations. Kim, Chang-Jin and Charles R. Nelson (1999) ``State-Space Models with Regime Switching: Classical and Gibbs-Sampling Approaches with Applications'' <http://econ.korea.ac.kr/ cjkim/doi:10.7551/mitpress/6444.001.0001><http://econ.korea.ac.kr/~{}cjkim/>.

License GPL (>= 2)

Imports Rcpp (>= 1.0.9)

LinkingTo Rcpp, RcppArmadillo

RoxygenNote 7.2.0

Suggests data.table (>= 1.14.2), maxLik (>= 1.5-2), ggplot2 (>= 3.3.6), gridExtra (>= 2.3), knitr, rmarkdown, testthat

VignetteBuilder knitr

Encoding UTF-8

NeedsCompilation yes

Author Alex Hubbard [aut, cre]

Maintainer Alex Hubbard <hubbard.alex@gmail.com>

Depends R (>= 3.5.0)

Repository CRAN

Date/Publication 2022-09-02 08:10:09 UTC

R topics documented:

filter 2
gen_inv 2
kalmanfilter 3

kalman_filter 3
kalman_lik 4
likelihood 5
Rginv 6
treasuries 6

Index **7**

`filter` *Kalman Filter*

Description

Kalman Filter

Usage

```
filter(sp, yt, Xo, Xs, smooth = FALSE)
```

Arguments

`sp`	list describing the state space model
`yt`	matrix of data
`Xo`	matrix of exogenous observation data
`Xs`	matrix of exogenous state data
`smooth`	boolean indication whether to run the backwards smoother

`gen_inv` *Generalized matrix inverse*

Description

Generalized matrix inverse

Usage

```
gen_inv(m)
```

Arguments

`m`	matrix

`kalmanfilter` *Kalman Filter*

Description

kalmanfilter Rcpp implementation of the multivariate Kalman filter for state space models that can handle missing values and exogenous data in the observation and state equations. See the package vignette using `browseVignettes("kalmanfilter")` to view it in your browser.

Author(s)

Alex Hubbard

`kalman_filter` *Kalman Filter*

Description

Kalman Filter

Usage

```
kalman_filter(ssm, yt, Xo = NULL, Xs = NULL, smooth = FALSE)
```

Arguments

`ssm`	list describing the state space model
`yt`	matrix of data
`Xo`	matrix of exogenous observation data
`Xs`	matrix of exogenous state data
`smooth`	boolean indication whether to run the backwards smoother

Value

list

Examples

```
#Nelson-Siegel Dynamic Factor Yield Curve Model
library(kalmanfilter)
library(data.table)
data(treasuries)
tau = unique(treasuries$maturity)

#Set up the state space model
```

```
ssm = list()
ssm[["Fm"]] = rbind(c(0.97, -0.03, -0.01),
                    c(0.08, 0.79, -0.15),
                    c(-0.15, 0.04, 0.88))
ssm[["Dm"]] = matrix(c(0.14, -0.1, 0.25), nrow = nrow(ssm[["Fm"]]), ncol = 1)
ssm[["Qm"]] = rbind(c(0.13, 0.12, -0.05),
                    c(0.12, 0.25, -0.07),
                    c(-0.05, -0.07, 1.02))
ssm[["Hm"]] = cbind(rep(1, 11),
                    -(1 - exp(-tau*0.04))/(tau*0.04),
                    (1 - exp(-tau*0.04))/(tau*0.04) - exp(-tau*0.04))
ssm[["Am"]] = matrix(0, nrow = length(tau), ncol = 1)
ssm[["Rm"]] = diag(c(0.01, 0.00, 0.012, 0.01, 0.01, 0.00,
                     0.01, 0.01, 0.01, 0.03, 0.04))
ssm[["betaO"]] = matrix(0, nrow = length(tau), ncol = 1)
ssm[["betaS"]] = matrix(0, nrow = nrow(ssm[["Fm"]]), ncol = 1)
ssm[["B0"]] = matrix(c(5.72, -0.80, 1.51), nrow = nrow(ssm[["Fm"]]), ncol = 1)
ssm[["P0"]] = diag(rep(0.01, nrow(ssm[["Fm"]])))

#Convert to an NxT matrix
yt = dcast(treasuries, "date ~ maturity", value.var = "value")
yt = t(yt[, 2:ncol(yt)])
kalman_filter(ssm, yt)
```

`kalman_lik` *Kalman Likelihood*

Description

Kalman Likelihood

Usage

```
kalman_lik(ssm, yt, Xo = NULL, Xs = NULL, w = NULL)
```

Arguments

ssm	list describing the state space model
yt	matrix of data
Xo	matrix of exogenous observation data
Xs	matrix of exogenous state data
w	matrix of weights

Value

numeric

Examples

```
#Nelson-Siegel Dynamic Factor Yield Curve Model
library(kalmanfilter)
library(data.table)
data(treasuries)
tau = unique(treasuries$maturity)

#Set up the state space model
ssm = list()
ssm[["Fm"]] = rbind(c(0.97, -0.03, -0.01),
                    c(0.08, 0.79, -0.15),
                    c(-0.15, 0.04, 0.88))
ssm[["Dm"]] = matrix(c(0.14, -0.1, 0.25), nrow = nrow(ssm[["Fm"]]), ncol = 1)
ssm[["Qm"]] = rbind(c(0.13, 0.12, -0.05),
                    c(0.12, 0.25, -0.07),
                    c(-0.05, -0.07, 1.02))
ssm[["Hm"]] = cbind(rep(1, 11),
                    -(1 - exp(-tau*0.04))/(tau*0.04),
                    (1 - exp(-tau*0.04))/(tau*0.04) - exp(-tau*0.04))
ssm[["Am"]] = matrix(0, nrow = length(tau), ncol = 1)
ssm[["Rm"]] = diag(c(0.01, 0.00, 0.012, 0.01, 0.01, 0.00,
                     0.01, 0.01, 0.01, 0.03, 0.04))
ssm[["betaO"]] = matrix(0, nrow = length(tau), ncol = 1)
ssm[["betaS"]] = matrix(0, nrow = nrow(ssm[["Fm"]]), ncol = 1)
ssm[["B0"]] = matrix(c(5.72, -0.80, 1.51), nrow = nrow(ssm[["Fm"]]), ncol = 1)
ssm[["P0"]] = diag(rep(0.01, nrow(ssm[["Fm"]])))

#Convert to an NxT matrix
yt = dcast(treasuries, "date ~ maturity", value.var = "value")
yt = t(yt[, 2:ncol(yt)])
kalman_lik(ssm, yt)
```

`likelihood` *Kalman Likelihood*

Description

Kalman Likelihood

Usage

```
likelihood(sp, yt, Xo, Xs, w)
```

Arguments

sp	list describing the state space model
yt	matrix of data
Xo	matrix of exogenous observation data

`Xs`	matrix of exogenous state data
`w`	matrix of weights

`Rginv` *R's implementation of the Moore-Penrose pseudo matrix inverse*

Description

R's implementation of the Moore-Penrose pseudo matrix inverse

Usage

```
Rginv(m)
```

Arguments

`m`	matrix

`treasuries` *Treasuries*

Description

Treasuries

Usage

```
data(treasuries)
```

Format

data.table with columns DATE, VARIABLE, VALUE, and MATURITY The data is quarterly frequency with variables DGS1MO, DGS3MO, DGS6MO, DGS1, DGS2, DGS3, DGS5, DGS7, DGS10, DGS20, and DGS30

Source

FRED

Index

∗ **datasets**
    treasuries, 6

filter, 2

gen_inv, 2

kalman_filter, 3
kalman_lik, 4
kalmanfilter, 3

likelihood, 5

Rginv, 6

treasuries, 6