Package ‘kimfilter’

October 13, 2022

Type Package

Title Kim Filter

Version 1.0.0

Date 2022-09-01

Description ‘Rcpp’ implementation of the multivariate Kim filter, which combines the Kalman and Hamilton filters for state probability inference. The filter is designed for state space models and 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>

License GPL (>= 2)

Imports Rcpp (>= 1.0.9)

LinkingTo Rcpp, RcppArmadillo

RoxygenNote 7.2.1

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 2022-09-14 06:00:05 UTC
R topics documented:

contains ......................................................... 2
gen_inv ......................................................... 2
kimfilter ....................................................... 3
kim_filter ...................................................... 3
Rginv ............................................................ 5
self_rbind ....................................................... 5
ss_prob .......................................................... 6
sw_dcf .......................................................... 7

Index

contains

Description
Check if list contains a name

Usage
contains(s, L)

Arguments
s a string name
L a list object

Value
boolean

gen_inv

Description
Generalized matrix inverse

Usage
gen_inv(m)

Arguments
m matrix
**Value**

matrix inverse of m

---

**Description**

`kimfilter` Rcpp implementation of the multivariate Kim filter, which combines the Kalman and Hamilton filters for state probability inference. The filter is designed for state space models and can handle missing values and exogenous data in the observation and state equations. `browseVignettes("kimfilter")` to view it in your browser.

**Author(s)**

Alex Hubbard

---

**kim_filter**  
**Kim Filter**

**Description**

Kim Filter

**Usage**

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

**Arguments**

- `ssm` : list describing the state space model, must include names `B0 - N_b x 1` matrix, initial guess for the unobserved components `P0 - N_b x N_b` matrix, initial guess for the covariance matrix of the unobserved components `Dm - N_b x 1` matrix, constant matrix for the state equation `Am - N_y x 1` matrix, constant matrix for the observation equation `Fm - N_b x p` matrix, state transition matrix `Hm - N_y x N_b` matrix, observation matrix `Qm - N_b x N_b` matrix, state error covariance matrix `Rm - N_y x N_y` matrix, state error covariance matrix `betaO - N_y x N_o` matrix, coefficient matrix for the observation exogenous data `betaS - N_b x N_s` matrix, coefficient matrix for the state exogenous data `Pm - n_state x n_state` matrix, state transition probability matrix
- `yt` : `N x T` matrix of data
- `Xo` : `N_o x T` matrix of exogenous observation data
- `Xs` : `N_s x T` matrix of exogenous state
- `weight` : column matrix of weights, `T x 1`
- `smooth` : boolean indication whether to run the backwards smoother
kim_filter

Value

list of cubes and matrices output by the Kim filter

Examples

# Stock and Watson Markov switching dynamic common factor
library(kimfilter)
library(data.table)
data(sw_dcf)
data = sw_dcf[, colnames(sw_dcf) != "dcoinc", with = FALSE]
vars = colnames(data)[colnames(data) != "date"]

# Set up the state space model
ssm = list()
ssm["Fm"] = rbind(c(0.8760, -0.2171, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
                   c(0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
                   c(0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0),
                   c(0, 0, 0, 0, 0.0364, -0.0008, 0, 0, 0, 0, 0, 0),
                   c(0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0),
                   c(0, 0, 0, 0, 0, -0.2965, -0.0657, 0, 0, 0, 0),
                   c(0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0),
                   c(0, 0, 0, 0, 0, 0, -0.3959, -0.1903, 0, 0),
                   c(0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0),
                   c(0, 0, 0, 0, 0, 0, 0, 0, 0, -0.2436, 0.1281),
                   c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0))
ssm["Fm"] = array(ssm["Fm"], dim = c(dim(ssm["Fm"]), 2))
ssm["Dm"] = matrix(c(-1.5700, rep(0, 11)), nrow = nrow(ssm["Fm"]), ncol = 1)
ssm["Dm"] = array(ssm["Dm"], dim = c(dim(ssm["Dm"]), 2))
ssm["Dm"][, 2] = 0.2802
ssm["Qm"] = diag(c(1, 0, 0, 0, 0.0001, 0, 0.0001, 0, 0.0001, 0, 0.0001, 0))
ssm["Qm"] = array(ssm["Qm"], dim = c(dim(ssm["Qm"])), 2)
ssm["Hm"] = rbind(c(0.0058, -0.0033, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0),
                   c(0.0011, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0),
                   c(0.0051, -0.0033, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0),
                   c(0.0012, -0.0005, 0.0001, 0.0001, 0.0002, 0, 0, 0, 0, 0, 0, 0),
                   c(0.0012, -0.0005, 0.0001, 0.0001, 0.0002, 0, 0, 0, 0, 0, 0, 0))
ssm["Hm"] = array(ssm["Hm"], dim = c(dim(ssm["Hm"])), 2)
ssm["Am"] = matrix(0, nrow = nrow(ssm["Hm"])), ncol = 1)
ssm["Am"] = array(ssm["Am"], dim = c(dim(ssm["Am"]), 2))
ssm["Rm"] = matrix(c(0, 0, 0, 0, -4.00278, 0, 0, 0, 0, 0, 0, 0),
                   c(0, 0, 0, 0, 1.00273, 0, 0, 0, 0, 0, 0, 0),
                   c(0, 0, 0, 0, 0, 1.00273, 0, 0, 0, 0, 0, 0),
                   c(0, 0, 0, 0, 0, 0, 1.00273, 0, 0, 0, 0, 0),
                   c(0, 0, 0, 0, 0, 0, 0, 1.00273, 0, 0, 0, 0),
                   c(0, 0, 0, 0, 0, 0, 0, 0, 1.00273, 0, 0, 0),
                   c(0, 0, 0, 0, 0, 0, 0, 0, 0, 1.00273, 0, 0),
                   c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.00273, 0),
                   c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.00273),
                   c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0))
ssm["B0"] = array(c(rep(-4.00278, 4), 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
                   c(2.1775, 1.5672, 0.9002, 0.4483, 0, 0, 0, 0, 0, 0, 0, 0),
                   c(1.5672, 1.5672, 1.5672, 0.9002, 0, 0, 0, 0, 0, 0, 0, 0),
                   c(0.9002, 1.5672, 2.1775, 0.9002, 0, 0, 0, 0, 0, 0, 0, 0),
                   c(0.4483, 0.9002, 1.5672, 2.1775, 0, 0, 0, 0, 0, 0, 0, 0),
                   c(0, 0, 0, 0, 0, 0.0001, 0, 0, 0, 0, 0, 0),
                   c(0, 0, 0, 0, 0, 0.0001, 0, 0, 0, 0, 0, 0),
                   c(0, 0, 0, 0, 0, -0.0001, 0, 0, 0, 0, 0, 0),
                   c(0, 0, 0, 0, 0, 0, 0.0001, 0, 0, 0, 0, 0),
                   c(0, 0, 0, 0, 0, 0, 0, 0.0001, 0, 0, 0, 0),
                   c(0, 0, 0, 0, 0, 0, 0, 0, 0.0001, 0, 0, 0),
                   c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0.0001, 0, 0),
                   c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.0001, 0, 0),
                   c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.0001))
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

Value

matrix inverse of m

self_rbind  Matrix self rowbind

Description

Matrix self rowbind

Usage

self_rbind(mat, times)
**Arguments**

mat  
matrix

times  
integer

**Value**

matrix

---

**ss_prob**  
*Finds the steady state probabilities from a transition matrix mat =* 
\[
\begin{pmatrix}
  p_{11} & p_{21} & \cdots & p_{m1} \\
  p_{12} & p_{22} & \cdots & p_{m2} \\
  \vdots & \vdots & \ddots & \vdots \\
  p_{1m} & p_{2m} & \cdots & p_{mm}
\end{pmatrix}
\]  
where the columns sum to 1

**Description**

Finds the steady state probabilities from a transition matrix mat = |p_{11} p_{21} ... p_{m1}| |p_{12} p_{22} ... p_{m2}| |... ...| |p_{1m} p_{2m} ... p_{mm}| where the columns sum to 1

**Usage**

`ss_prob(mat)`

**Arguments**

mat  
square SxS matrix of probabilities with column sums of 1. S represents the number of states

**Value**

matrix of dimensions Sx1 with steady state probabilities

**Examples**

```r
library(kimfilter)
Pm = rbind(c(0.8406, 0.0304), c(0.1594, 0.9696))
ss_prob(Pm)
```
Description

Stock and Watson Markov Switching Dynamic Common Factor Data Set

Usage

data(sw_dcf)

Format

data.table with columns DATE, VARIABLE, VALUE, and MATURITY The data is monthly frequency with variables ip (industrial production), gmyxpg (total personal income less transfer payments in 1987 dollars), mtq (total manufacturing and trade sales in 1987 dollars), lpnag (employees on non-agricultural payrolls), and dcoinc (the coincident economic indicator)

Source

Index

* datasets
  sw_dcf, 7
contains, 2

gen_inv, 2

kim_filter, 3
kimfilter, 3

Rginv, 5

self_rbind, 5
ss_prob, 6
sw_dcf, 7