Package ‘kalmanfilter’

March 8, 2024

Type Package
Title Kalman Filter
Version 2.1.1
Date 2024-03-01
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. There is also a function to handle time varying parameters.


License GPL (>= 2)
Imports Rcpp (>= 1.0.9)
LinkingTo Rcpp, RcppArmadillo
RoxygenNote 7.2.3
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 2024-03-08 05:00:06 UTC

R topics documented:

contains ......................................................... 2
gen_inv ........................................................... 2
contains

Check if list contains a name

Description
Check if list contains a name

Usage
contains(s, L)

Arguments
s a string name
L a list object

Value
boolean

gen_inv

Generalized matrix inverse

Description
Generalized matrix inverse

Usage
gen_inv(m)

Arguments
m matrix

Value
matrix inverse of m
kalman_filter

Description
Kalman Filter

Usage
kalman_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 (or array of length yt), initial guess for the unobserved components P0 - N_b x N_b matrix (or array of length yt), initial guess for the covariance matrix of the unobserved components Dm - N_b x 1 matrix (or array of length yt), constant matrix for the state equation Am - N_y x 1 matrix (or array of length yt), constant matrix for the observation equation Fm - N_b X p matrix (or array of length yt), state transition matrix Hm - N_y x N_b matrix (or array of length yt), observation matrix Qm - N_b x N_b matrix (or array of length yt), state error covariance matrix Rm - N_y x N_y matrix (or array of length yt), state error covariance matrix betaO - N_y x N_o matrix (or array of length yt), coefficient matrix for the observation exogenous data betaS - N_b x N_s matrix (or array of length yt), coefficient matrix for the state exogenous data

- **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

Value
list of cubes and matrices output by the Kalman filter

Examples

```r
## Not run:
# Stock and Watson Markov switching dynamic common factor
library(kalmanfilter)
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, 1, 0, 0, 0, 0, 0, 0, 0, 0),
                 c(0, 0, 0, 0, 0, 0, -0.2965, -0.0057, 0, 0, 0, 0),
                 c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -0.3959, -0.1903, 0, 0),
                 c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
                 c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
                 c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
                 c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
                 c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
                 c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
                 c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 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, 0.0001, 0, 0, 0.0001, 0, 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, 0, 0, 0, 0, 0, 0, 0),
                 c(0.0051, -0.0033, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0),
                 c(0.00011, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
                 c(0.00051, -0.0033, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0),
                 c(0.00012, -0.0005, 0.0001, 0.0002, 0, 0, 0, 0, 0, 1, 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(0, nrow = nrow(ssm["Am"]), ncol = nrow(ssm["Am"]))

ssm["Rm"] = array(ssm["Rm"], dim = c(dim(ssm["Rm"]), 2))

ssm["B0"] = matrix(c(rep(-4.60278, 4), 0, 0, 0, 0, 0, 0, 0),
                   nrow = nrow(ssm["B0"]))

ssm["B0"] = array(ssm["B0"], dim = c(dim(ssm["B0"]), 2))

ssm["B0"][, 2] = rep(0.82146, 4)

ssm["P0"] = rbind(c(2.1775, 1.5672, 0.9002, 0.4483, 0, 0, 0, 0, 0, 0, 0, 0),
                   c(0.9002, 1.5672, 2.1775, 1.5672, 0.9002, 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, 0, 0, 0, 0, 0, 0),
                   c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
                   c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
                   c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
                   c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
                   c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
                   c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
                   c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
                   c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
                   c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0))

ssm["P0"] = array(ssm["P0"], dim = c(dim(ssm["P0"]), 2))

#Log, difference and standardize the data

data[, c(vars) := lapply(.SD, log), .SDcols = c(vars)]

data[, c(vars) := lapply(.SD, function(x){
  x - shift(x, type = "lag", n = 1)
}), .SDcols = c(vars)]

data[, c(vars) := lapply(.SD, scale), .SDcols = c(vars)]

#Convert the data to an NxT matrix

yt = t(data[, c(vars), with = FALSE])
kalman_filter_cpp

kf = kalman_filter(ssm, yt, smooth = TRUE)

## End(Not run)

kalman_filter_cpp  Kalman Filter

Description
Kalman Filter

Usage
kalman_filter_cpp(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
- **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

Value
list of matrices and cubes output by the Kalman filter

Examples
# Nelson-Siegel dynamic factor yield curve
library(kalmanfilter)
library(data.table)
data(treasuries)
tau = unique(treasuries$maturity)

# Set up the state space model
ssm = list()
ssm[["Fm"] ] = rbind(c(0.9720, -0.0209, -0.0061)
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
### sw_dcf

*Stock and Watson Dynamic Common Factor Data Set*

**Description**

Stock and Watson Dynamic Common Factor Data Set

**Usage**

```r
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**


---

### treasuries

*Treasuries*

**Description**

Treasuries

**Usage**

```r
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
  sw_dcf, 7
  treasuries, 7
contains, 2

gen_inv, 2

kalman_filter, 3
kalman_filter_cpp, 5

Rginv, 6

sw_dcf, 7

treasuries, 7