

Package ‘KFAS’

January 2, 2012

Version 0.6.1

Date 2011-02-23

Title Kalman filter and smoothers for exponential family state space models.

Author Jouni Helske <jouni.helske@jyu.fi>

Maintainer Jouni Helske <jouni.helske@jyu.fi>

Depends R (>= 2.8.0)

Description Package KFAS provides functions for Kalman filtering, state, disturbance and simulation smoothing, forecasting and simulation of state space models. All functions can use exact diffuse initialisation when distributions of some or all elements of initial state vector are unknown. Filtering, state smoothing and simulation functions use sequential processing algorithm, which is faster than standard approach, and it also allows singularity of prediction error variance matrix. KFAS also contains function for computing the likelihood of exponential family state space models and function for state smoothing of exponential family state space models.

License GPL (>= 2)

Repository CRAN

Date/Publication 2011-02-23 16:18:39

R topics documented:

distsmoother	2
effik	2
effik0	5
efsmoother	7
forecast	9
GlobalTemp	10
kf	11
ks	16
simsmoother	18

Index**21**

distsmoothen	<i>Disturbance smoothen</i>
--------------	-----------------------------

Description

Computes smoothed values of disturbance terms eps_t and eta_t , and their variances.

Usage

```
distsmoothen(out)
```

Arguments

`out` Output of function 'ks'. Optcal must be (TRUE,TRUE,TRUE,TRUE/FALSE).

Value

A list with the output elements from Kalman smoothen and following new elements:

epshat	$p \times n$ array of $E(\text{eps}_t y_{-1}, \dots, y_n)$.
epshatvar	$p \times p \times n$ array of $\text{var}(\text{eps}_t y_{-1}, \dots, y_n)$.
etahat	$r \times n$ array of $E(\text{eta}_t y_{-1}, \dots, y_n)$.
etahatvar	$r \times r \times n$ array of $\text{var}(\text{eta}_t y_{-1}, \dots, y_n)$.

References

Koopman, S.J. and Durbin J. (2001). Time Series Analysis by State Space Methods. Oxford: Oxford University Press

eflik	<i>Log-likelihood of exponential family state-space model.</i>
-------	--

Description

Function `eflik` computes log-likelihood function of univariate exponential family state-space model via simulation.

Usage

```
eflik(yt, Zt, Tt, Rt, Qt, a1, P1, P1inf, dist=c("Poisson",
"Binomial", "Negative binomial"), offset=1, nsim=1000, maxiter=50)
```

Arguments

<code>yt</code>	Matrix, array or vector of observations. Note that <code>yt</code> is univariate.
<code>Zt</code>	System matrix or array of observation equation.
<code>Tt</code>	System matrix or array of transition equation.
<code>Rt</code>	System matrix or array of transition equation.
<code>Qt</code>	Variance matrix or array of disturbance terms <code>eta_t</code> .
<code>a1</code>	Initial state vector.
<code>P1</code>	Variance matrix of <code>a1</code> . In diffuse case <code>P1star</code> , the non-diffuse part of <code>P1</code> .
<code>P1inf</code>	Diffuse part of <code>P1</code> .
<code>dist</code>	Distribution of <code>yt</code> .
<code>offset</code>	Vector of length <code>n</code> . See details.
<code>nsim</code>	Number of simulations.
<code>maxiter</code>	Maximum number of iterations used in linearisation.

Details

Function computes log-likelihood of exponential family state-space model. It is very recommended to use estimates gained from function `lik0` as initial values.

$$\log L_p(\psi) = \log L_g(\psi) + \log E*_g(w|y),$$

where $\log L_g(\psi)$ is log-likelihood of approximate gaussian distribution and $\log E*_g(w|y)$ is a Monte-Carlo approximation of $E_g[p(y|\theta)/g(y|\theta) | y]$. For details, see J. Durbin and S.J. Koopman (1997).

The general state space model for exponential family is given by

$$p(y_t|\theta_t) = \exp[\theta_t' * y_t - b_t(\theta_t) + f_t(y_t)] \text{ (observation equation)}$$

$$\alpha_{t+1} = T_t * \alpha_t + R_t * \eta_t \text{ (transition equation)}$$

where $\theta_t = Z_t * \alpha_t$ and $\eta_t \sim N(0, Q_t)$.

Approximating gaussian model is given by

$$y*_t = Z_t * \alpha_t + \epsilon_t \text{ (observation equation)}$$

$$\alpha_{t+1} = T_t * \alpha_t + R_t * \eta_t \text{ (transition equation)}$$

where $\epsilon_t \sim N(0, H*_t)$ and $\eta_t \sim N(0, Q_t)$.

If `yt` is Poisson distributed, parameter of Poisson distribution is `offset*lambda` and $\theta = \log(\lambda)$.

If `yt` is from binomial distribution, `offset` is a vector specifying number of trials at times 1,...,n, and $\theta = \log[\pi_t/(1-\pi_t)]$, where π_t is the probability of success at time `t`.

In case of negative binomial distribution, `offset` is vector of specified number of successes wanted at times 1,...,n, and $\theta = \log(1-\pi_t)$.

Note that this function works only for univariate observations.

Value

List with output from Kalman smoother, when model is approximated with gaussian distribution $g(y|\theta)$. Note that H_t is H^*_t . List also contains following items:

ytilde	y^* of approximating gaussian model.
theta	$Z_t * \alpha_t$ of approximating gaussian model.
likp	Value of $\log L_p$.
offset	
dist	Distribution of y_t .
lik	Value of log-likelihood function.

References

Durbin J. and Koopman, S.J. (1997). Monte Carlo Maximum Likelihood Estimation for Non-Gaussian State Space Models, *Biometrika*, Vol. 84, No. 3.
 Koopman, S.J. and Durbin J. (2001). *Time Series Analysis by State Space Methods*. Oxford: Oxford University Press

Examples

```
kk <- c(396, 399, 403, 434, 340, 519, 558, 566, 591, 566, 574, 646, 644,
646, 665, 693, 773, 834, 910, 1035, 1002, 1161, 1056, 1097, 1094, 1042,
1194, 1316, 1246, 1503, 1428, 1477, 1490, 1465, 1560, 1860, 2008, 2020, 2167)

v1k <- c(4623785, 4606307, 4612124, 4639657, 4666081, 4690574, 4711440,
4725664, 4738902, 4752528, 4764690, 4779535, 4799964, 4826933, 4855787,
4881803, 4902206, 4918154, 4932123, 4946481, 4964371, 4986431, 5013740,
5041992, 5066447, 5088333, 5107790, 5124573, 5139835, 5153498, 5165474,
5176209, 5188008, 5200598, 5213014, 5228172, 5246096, 5266268, 5288720)

n<-39
Zt<-array(c(1,0),c(1,2))
Tt<-diag(c(1,1))
Tt[1,2]<-1
Rt <- diag(2)
a1 <- array(0,dim=2)
P1 <- diag(0,2)
P1inf <-diag(1,2)

likfn<-function(pars,yt,Zt,Tt,Rt,a1,P1,P1inf,dist,offset=1)
{
Qt<-diag(exp(pars))
lik<-eflik0(yt,Zt,Tt,Rt,Qt,a1,P1,P1inf,dist,offset)$lik0
lik
}

opt <- optim(par=c(1,1), yt=array(kk,dim=c(1,39)), Zt=Zt, Tt=Tt, Rt=Rt,
```

```

a1=a1, P1=P1, P1inf=P1inf, offset=vlk, dist="Poisson", fn=likfn,
method="BFGS", control=list(fnscale=-1,trace=1,REPORT=1))

Qt<-diag(exp(opt$par))
out<-eflik0(yt=kk,Zt,Tt,Rt,Qt,a1,P1,P1inf,offset=vlk)
out$Qt*10000

likfn2<-function(pars,yt,Zt,Tt,Rt,a1,P1,P1inf,dist,offset=1)
{
Qt<-diag(exp(pars))
lik<-eflik(yt,Zt,Tt,Rt,Qt,a1,P1,P1inf,dist,offset,nsim=1000)$likp
lik
}

opt2 <- optim(par=opt$par, yt=array(kk,dim=c(1, 39)), Zt=Zt, Tt=Tt,
Rt=Rt, a1=a1, P1=P1, P1inf=P1inf, offset=vlk, dist="Poisson", fn=likfn2,
method="BFGS", control=list(fnscale=-1, trace=1, REPORT=1))

Qt<-diag(exp(opt2$par))
out<-eflik(yt=kk, Zt, Tt, Rt, Qt, a1, P1, P1inf, offset=vlk)
out$Qt*10000

```

eflik0	<i>Approximate log-likelihood and gaussian density of exponential family state-space model.</i>
--------	---

Description

Function `expflik` computes approximate log-likelihood and approximate gaussian density of univariate exponential family state-space model, based on Durbin and Koopman (1997, 2001).

Usage

```
eflik0(yt, Zt, Tt, Rt, Qt, a1, P1, P1inf, dist=c("Poisson",
"Binomial", "Negative binomial"), offset=1, maxiter=50)
```

Arguments

<code>yt</code>	Matrix, array or vector of observations. Note that <code>yt</code> is univariate.
<code>Zt</code>	System matrix or array of observation equation.
<code>Tt</code>	System matrix or array of transition equation.
<code>Rt</code>	System matrix or array of transition equation.
<code>Qt</code>	Variance matrix or array of disturbance terms <code>eta_t</code> .
<code>a1</code>	Initial state vector.
<code>P1</code>	Variance matrix of <code>a1</code> . In diffuse case <code>P1star</code> , the non-diffuse part of <code>P1</code> .
<code>P1inf</code>	Diffuse part of <code>P1</code> .

dist	Distribution of yt.
offset	Vector of length n. See details.
maxiter	Maximum number of iterations used in linearisation.

Details

Function approximates $p(\text{alphaly})$ with gaussian $g(\text{alphaly})$ which has same conditional mode ($\alpha_t, \alpha_{t+1|y}$) as $p(\text{alphaly})$, and computes approximate log-likelihood

$$\log L_0(\psi) = \log L_g(\psi) + \log E_g(w_T | y),$$

where $\log L_g(\psi)$ is log-likelihood of approximate gaussian distribution and $\log E_g(w_T | y)$ is a Taylor-approximation of $E_g[p(y|\theta)/g(y|\theta) | y]$. For details, see J. Durbin and S.J. Koopman (1997).

The general state space model for exponential family is given by

$$p(y_t | \theta_t) = \exp[\theta_t' y_t - b_t(\theta_t) + f_t(y_t)] \quad (\text{observation equation})$$

$$\alpha_{t+1} = T_t \alpha_t + R_t \eta_t \quad (\text{transition equation})$$

where $\theta_t = Z_t \alpha_t$ and $\eta_t \sim N(0, Q_t)$.

Approximating gaussian model is given by

$$y_t^* = Z_t \alpha_t + \epsilon_t \quad (\text{observation equation})$$

$$\alpha_{t+1} = T_t \alpha_t + R_t \eta_t \quad (\text{transition equation})$$

where $\epsilon_t \sim N(0, H_t^*)$ and $\eta_t \sim N(0, Q_t)$.

If y_t is Poisson distributed, parameter of Poisson distribution is $\text{offset} * \lambda$ and $\theta = \log(\lambda)$.

If y_t is from binomial distribution, offset is a vector specifying number of trials at times 1, ..., n, and $\theta = \log[\pi_t / (1 - \pi_t)]$, where π_t is the probability of success at time t.

In case of negative binomial distribution, offset is vector of specified number of successes wanted at times 1, ..., n, and $\theta = \log(1 - \pi_t)$.

Note that this function works only for univariate observations.

Value

List with output from Kalman smoother and distribution smoother, when model is approximated with gaussian distribution $g(y|\theta)$. Note that H_t is H_t^* . List also contains following items:

ytilde	y^* of approximating gaussian model.
theta	$Z_t \alpha_t$ of approximating gaussian model.
lik0	Value of $\log L_0$.
offset	
dist	Distribution of yt.

References

- Durbin J. and Koopman, S.J. (1997). Monte Carlo Maximum Likelihood Estimation for Non-Gaussian State Space Models, *Biometrika*, Vol. 84, No. 3.
- Koopman, S.J. and Durbin J. (2001). *Time Series Analysis by State Space Methods*. Oxford: Oxford University Press.

Examples

```

kk <- c(396, 399, 403, 434, 340, 519, 558, 566, 591, 566, 574, 646, 644,
646, 665, 693, 773, 834, 910,1035, 1002, 1161, 1056, 1097, 1094, 1042,
1194, 1316, 1246, 1503, 1428, 1477, 1490, 1465, 1560, 1860, 2008, 2020,
2167)

v1k <- c(4623785, 4606307, 4612124, 4639657, 4666081, 4690574, 4711440,
4725664, 4738902, 4752528, 4764690, 4779535, 4799964, 4826933, 4855787,
4881803, 4902206, 4918154, 4932123, 4946481, 4964371, 4986431, 5013740,
5041992, 5066447, 5088333, 5107790, 5124573, 5139835, 5153498, 5165474,
5176209, 5188008, 5200598, 5213014, 5228172, 5246096, 5266268, 5288720)

n<-39
Zt<-array(c(1,0),c(1,2))
Tt<-diag(c(1,1))
Tt[1,2]<-1
Rt <- diag(2)
a1 <- array(0,dim=2)
P1 <- diag(0,2)
P1inf <-diag(1,2)

likfn<-function(pars,yt,Zt,Tt,Rt,a1,P1,P1inf,dist,offset=1)
{
Qt<-diag(exp(pars))
lik<-eflik0(yt,Zt,Tt,Rt,Qt,a1,P1,P1inf,dist,offset)$lik0
lik
}

opt <- optim(par=c(1, 1), yt=array(kk,dim=c(1, 39)), Zt=Zt, Tt=Tt,
Rt=Rt, a1=a1, P1=P1, P1inf=P1inf, offset=v1k, dist="Poisson",
fn=likfn, method="BFGS", control=list(fnscale=-1, trace=1, REPORT=1))

Qt<-diag(exp(opt$par))
out<-eflik0(yt=kk, Zt, Tt, Rt, Qt, a1, P1, P1inf, offset=v1k)
out$Qt*10000

```

Description

Performs Kalman smoothing for exponential family models via simulation.

Usage

```
efsmoother(out,nsim)
```

Arguments

out Output from function 'expflik'.
 nsim Number of simulations used for smoothing.

Details

Function `efsmoother` performs Kalman smoothing of univariate exponential family state space model using the simulation algorithms by Koopman and Durbin (2001, 2002).

Value

A list with the output elements from `expflik` and following new elements:

nsim Number of simulations used for smoothing.
 ahat $m \times n$ array of $E(\text{alphat} \mid y_1, y_2, \dots, y_n)$.
 that $m \times n$ array of $Z_t \text{ * ahat}$.
 ahatvar $m \times m \times n$ array of $\text{Var}(\text{ahatly}_1, y_2, \dots, y_n)$.
 thatvar $1 \times 1 \times n$ array of $Z_t \text{ * Var}(\text{ahatly}_1, y_2, \dots, y_n) \text{ * } Z_t'$.

References

Koopman, S.J. and Durbin J. (2001). Time Series Analysis by State Space Methods. Oxford: Oxford University Press.

Durbin J. and Koopman, S.J. (2002). A simple and efficient simulation smoother for state space time series analysis, *Biometrika*, Volume 89, Issue 3.

Examples

```
library(KFAS)

kk <- c(396, 399, 403, 434, 340, 519, 558, 566, 591, 566, 574, 646, 644,
646, 665, 693, 773, 834, 910, 1035, 1002, 1161, 1056, 1097, 1094, 1042,
1194, 1316, 1246, 1503, 1428, 1477, 1490, 1465, 1560, 1860, 2008, 2020,
2167)

v1k <- c(4623785, 4606307, 4612124, 4639657, 4666081, 4690574, 4711440,
4725664, 4738902, 4752528, 4764690, 4779535, 4799964, 4826933, 4855787,
4881803, 4902206, 4918154, 4932123, 4946481, 4964371, 4986431, 5013740,
5041992, 5066447, 5088333, 5107790, 5124573, 5139835, 5153498, 5165474,
5176209, 5188008, 5200598, 5213014, 5228172, 5246096, 5266268, 5288720)

#Model

n <- 39
m <- 2
```

```

r <- 2
yt <- array(kk,dim=c(1,n))
offset <- array(vlk,dim=c(1,n))
Zt <- array(c(1,0),dim=c(1,m))
Tt <- array(c(1,0,1,1),dim=c(m,m))
Rt <- array(c(1,0,0,1),dim=c(m,r))
psi_hat <- c(-5.356458, -16.348376)
Qt <- array(c(exp(psi_hat[1]), 0, 0, exp(psi_hat[2])), dim=c(r, r))
a1 <- matrix(0,nrow=m,ncol=1)
P1 <- matrix(0,nrow=m,ncol=m)
P1inf<-diag(1,2)
dist <- "Poisson"
eflikout <- eflik0(yt, Zt, Tt, Rt, Qt, a1, P1, P1inf, dist, offset)
nsim <- 1000
smef <- efsmoothe(eflikout, nsim)

```

forecast

Forecast state space model

Description

Performs forecasting using output from function 'kf' (Kalman filter).

Usage

```

forecast(out, fc=1, Zt.fc=NULL, Tt.fc=NULL, Rt.fc=NULL,
Ht.fc=NULL, Qt.fc=NULL)

```

Arguments

out	Output from function 'kf'.
fc	Integer which states how many observations is forecasted.
Zt.fc	In case where matrix Z is not time-invariant, $p*m*fc$ array of matrix Z_t , $t=n+1, \dots, n+fc$.

Tt . fc	In case where matrix T is not time-invariant, $m \times m \times fc$ array of matrix T_t , $t=n+1, \dots, n+fc$.
Rt . fc	In case where matrix R is not time-invariant, $m \times r \times fc$ array of matrix R_t , $t=n+1, \dots, n+fc$.
Ht . fc	In case where matrix H is not time-invariant, $p \times p \times fc$ array of matrix H_t , $t=n+1, \dots, n+fc$.
Qt . fc	In case where matrix Q is not time-invariant, $r \times r \times fc$ array of matrix Q_t , $t=n+1, \dots, n+fc$.

Details

The state space model is given by

$$y_t = Z_t * \alpha_t + \text{eps}_t \text{ (observation equation)}$$

$$\alpha_{t+1} = T_t * \alpha_t + R_t * \eta_t \text{ (transition equation)}$$

where $\text{eps}_t \sim N(0, H_t)$ and $\eta_t \sim N(0, Q_t)$.

Dimensions of variables are:

'yt' $p \times n$

'Zt' $p \times m$ or $p \times m \times n$

'Tt' $m \times m$ or $m \times m \times n$

'Rt' $m \times r$ or $m \times r \times n$

'Ht' $p \times p$ or $p \times p \times n$

'Qt' $r \times r$ or $r \times r \times n$

Value

A list with the following elements:

yt . fc	$p \times fc$ array of forecasts of observations.
Ft . fc	$p \times p \times fc$ array of mean square error matrix.
at . fc	$m \times (fc+1)$ array of $E(\alpha_t y_1, y_2, \dots, y_n)$.
Pt . fc	$m \times m \times (fc+1)$ array of $\text{Var}(\alpha_t y_1, y_2, \dots, y_n)$.

GlobalTemp	<i>Two series of average global temperature deviations for years 1880-1987</i>
------------	--

Description

This data set contains two series of average global temperature deviations for years 1880-1987. These series are same as used in Shumway and Stoffer (2006), where they are known as HL and Folland series. For more details, see Shumway and Stoffer (2006, p. 327).

Usage

GlobalTemp

Format

An 2 times 108 array containing 216 observations.

Source

<http://lib.stat.cmu.edu/general/stoffer/tsa2/>

References

Shumway, Robert H. and Stoffer, David S. (2006). Time Series Analysis and Its Applications: With R examples.

 kf

Kalman Filter With Exact Diffuse Initialisation

Description

Performs Kalman filtering with exact diffuse initialisation using univariate approach (also known as sequential processing). Written in Fortran, uses subroutines from BLAS and LAPACK. See function 'ks' for smoothing.

Usage

```
kf(yt, Zt, Tt, Rt, Ht, Qt, a1, P1, P1inf=0, optcal=c(TRUE,
TRUE, TRUE, TRUE), tol=1e-7)
```

Arguments

yt	Matrix or array of observations.
Zt	System matrix or array of observation equation.
Tt	System matrix or array of transition equation.
Rt	System matrix or array of transition equation.
Ht	Variance matrix or array of disturbance terms ϵ_t of observation equation.
Qt	Variance matrix or array of disturbance terms η_t .
a1	Initial state vector.
P1	Variance matrix of a1. In diffuse case P1star, the non-diffuse part of P1.
P1inf	Diffuse part of P1. If non-zero, filtering starts with exact diffuse initialisation.
optcal	Vector of length 4. Calculate multivariate vt, Ft, Kt, Lt and their diffuse counterparts Finf, Fstar, Kinf, Kstar, Linf and Lstar. Default is c(TRUE,TRUE,TRUE,TRUE) which calculates all. Note that Kt cannot be calculated without Ft and Lt cannot be calculated without Kt, so even if optcal=c(TRUE,FALSE,TRUE,TRUE), Kt and Lt are not calculated.
tol	Tolerance parameter. Smallest covariance/variance value not counted for zero in diffuse phase. Default is 1e-7.

Details

Function `kf` performs Kalman filtering of gaussian multivariate state space model using the univariate approach from Koopman and Durbin (2000, 2001). Univariate approach is also known as sequential processing, see Anderson and Moore (1979). In case where the distributions of some or all of the elements of initial state vector are not known, `kf` uses exact diffuse initialisation using univariate approach by Koopman and Durbin (2000, 2003). Note that in univariate approach the prediction error variance matrices F_t , F_{inf} and F_{star} does not need to be non-singular, as there is no matrix inversions in univariate approach algorithm. This provides faster and more general filtering than normal multivariate Kalman filter algorithm.

Filter can deal partially or totally missing observation vectors. If $y_{t,i}$ is NA, it is interpreted as missing value, and the dimensions of v_{tuni} and F_{tuni} (or $F_{staruni}$ and F_{infuni}) are decreased and the corresponding elements of v_t are marked as NA.

The state space model is given by

$$y_t = Z_t * \alpha_t + \text{eps}_t \text{ (observation equation)}$$

$$\alpha_{t+1} = T_t * \alpha_t + R_t * \eta_t \text{ (transition equation)}$$

where $\text{eps}_t \sim N(0, H_t)$ and $\eta_t \sim N(0, Q_t)$. Note that error terms eps_t and η_t are assumed to be uncorrelated.

When P_{1inf} is non-zero, exact diffuse initialisation is used. Matrices P_t , K_t , F_t and L_t are decomposed to P_{inf} , P_{star} , K_{inf} , K_{star} , F_{inf} , F_{star} , L_{inf} and L_{star} . Diffuse phase is continued until P_{inf} becomes zero-matrix. See Koopman and Durbin (2000, 2001, 2003) for details for exact diffuse and non-diffuse filtering.

Notice that v_{tuni} , F_{tuni} , $F_{staruni}$, F_{infuni} , K_{tuni} , K_{infuni} and $K_{staruni}$ are usually not the same as those calculated in usual multivariate Kalman filter. Also the L_t , L_{inf} and L_{star} are not calculated explicitly. If usual v_t , F_t , F_{inf} , F_{star} , K_t , K_{inf} , K_{star} , L_t , L_{inf} and L_{star} are needed, use `optcal=c(TRUE, TRUE, TRUE, TRUE)`. When estimating parameters, it is suggested to use `optcal=c(FALSE, FALSE, FALSE, FALSE)` for maximum speed.

Dimensions of variables are:

'yt' $p*n$
 'Zt' $p*m$ or $p*m*n$
 'Tt' $m*m$ or $m*m*n$
 'Rt' $m*r$ or $m*r*n$
 'Ht' $p*p$ or $p*p*n$
 'Qt' $r*r$ or $r*r*n$
 'a1' $m*1$
 'P1' $m*m$
 'P1inf' $m*m$

where p is dimension of observation vector, m is dimension of state vector and n is number of observations.

Value

A list with the following elements:

`yt` $p*n$ matrix or array of observations.

ydimt	array of length n which has dimensions of observation vector at times $t=1, \dots, n$.
tv	array of length 4 where $tv[i]=0$ if i is time-invariant and otherwise $tv[i]=1$, i is dt, Tt, Rt, Qt.
Zt	system matrix or array of observation equation.
Tt	system matrix or array of transition equation.
Rt	system matrix or array of transition equation.
Ht	variance matrix or array of disturbance terms ϵ_t of observation equation.
Qt	variance matrix or array of disturbance terms η_t .
a1	initial state vector.
P1	variance matrix of a1. In diffuse case P1star, the non-diffuse part of P1 .
at	$m*(n+1)$ array of $E(\alpha_t y_1, y_2, \dots, y_{t-1})$.
Pt	$m*m*(n+1)$ array of $\text{Var}(\alpha_t y_1, y_2, \dots, y_{t-1})$.
vtuni	$p*1*n$ array of v_t of univariate approach.
Ftuni	F_t of univariate approach, $\text{Var}(vtuni)$.
Ktuni	$m*p*n$ array of Kalman gain of univariate approach.
Pinf, Pstar	$p*p*d+1$ arrays of diffuse phase decomposition of Pt.
Finfuni, Fstaruni	$p*p*d$ ($p*d$ arrays of diffuse phase decomposition of Ftuni.
Kinfuni, Kstaruni	$m*p*d$ arrays of diffuse phase decomposition of Ktuni.
d	the last index of diffuse phase, ie. the non-diffuse phase began from time $d+1$.
j	the index of last $y_{t,i}$ of diffuse phase.
p	the dimension of observation vector.
m	the dimension of state vector.
r	the dimension of variance matrix Qt.
n	the number of observations.
lik	Value of the log-likelihood function. If NaN, Ftuni _{i,t} was zero at some $t=d+1, \dots, n$ or Finfuni _{i,t} and Fstaruni _{i,t} was zero at some $t=1, \dots, d$.
optcal	
info	if info[1]=1, could not diagonalize Ht. If info[i]=1, $i=2,3,4$, Finf, Fstar or Ft was singular.
vt	$p*1*n$ array of $v_t = y_t - Z_t * a_t$.
Ft	$p*p*n$ array of $F_t = \text{Var}(v_t)$ of Kalman filter.
Kt	$m*p*n$ array of Kalman gain: $K_t = T_t * P_t * Z_t' * F_t^{-1}$.
Lt	the $m*m*n$ array, $L_t = T_t - K_t * Z_t$.
Finf, Fstar	$p*p*d$ arrays of diffuse phase decomposition of Ft.
Kinf, Kstar	$m*p*d$ arrays of diffuse phase decomposition of Kt.
Linf, Lstar	$m*m*d$ arrays of diffuse phase decomposition of Lt.
tol	Tolerance parameter.

References

Koopman, S.J. and Durbin J. (2000). Fast filtering and smoothing for non-stationary time series models, *Journal of American Statistical Assosiation*, 92, 1630-38.

Koopman, S.J. and Durbin J. (2001). *Time Series Analysis by State Space Methods*. Oxford: Oxford University Press.

Koopman, S.J. and Durbin J. (2003). Filtering and smoothing of state vector for diffuse state space models, *Journal of Time Series Analysis*, Vol. 24, No. 1.

Shumway, Robert H. and Stoffer, David S. (2006). *Time Series Analysis and Its Applications: With R examples*.

Examples

```
library(KFAS)

#Example of local level model
#Using the Nile observations
data(Nile)

yt<-t(data.matrix(Nile))
s2_eps<-15099
s2_eta<-1469.1

f.out<-kf(yt = yt, Zt = 1, Tt=1, Rt=1, Ht= s2_eps, Qt=s2_eta, a1 =
0, P1=1e7)

#a1 and P1 are not really estimated,
#should actually use exact diffuse initialisation:

fd.out<-kf(yt = yt, Zt = 1, Tt=1, Rt=1, Ht=s2_eps, Qt=s2_eta, a1 =
0, P1=0, P1inf=1)

#No stationary elements, P1=0, P1inf=1

#Plotting observations, non-diffuse and diffuse at, not plotting the a1=0:

ts.plot(Nile, ts(f.out$at[1,2:length(Nile)]), start=1872, end=1970),
ts(fd.out$at[1,2:length(Nile)]), start=1872, end=1970), col=c(1,2,3))

#Looks identical. Actually start of series differs little bit:

f.out$at[1,1:20]
fd.out$at[1,1:20]

#Example of multivariate local level model
```

```

#Two series of average global temperature deviations for years 1880-1987
#See Shumway and Stoffer (2006), p. 327 for details

data(GlobalTemp)
yt<-array(GlobalTemp,c(2,108))

#Estimating the variance parameters

likfn<-function(par, yt, a1, P1, P1inf) #Function to optim
{
L<-matrix(c(par[1],par[2],0,par[3]),ncol=2)
H<-L%*%t(L)
q11<-exp(par[4])
lik<-kf(yt = yt, Zt = matrix(1,nrow=2), Tt=1, Rt=1, Ht=H, Qt=q11, a1 =
a1, P1=P1, P1inf=P1inf, optcal=c(FALSE,FALSE,FALSE,FALSE))
lik$lik
}

#Diffuse initialisation
#Notice that diffuse initialisation without univariate approach does not
#work here because Finf is non-singular and non-zero

est<-optim(par=c(.1,0,.1,.1), likfn, method="BFGS",
control=list(fnscale=-1), hessian=TRUE, yt=yt, a1=0, P1=0, P1inf=1)

pars<-est$par
L<-matrix(c(pars[1],pars[2],0,pars[3]),ncol=2)
H<-L%*%t(L)
q11<-exp(pars[4])

kfd<-kf(yt = yt, Zt = matrix(1,nrow=2), Tt=1, Rt=1,
Ht=H, Qt=q11, a1 = 0, P1=0, P1inf=1)

#Same as non-diffuse, initial values from Shumway and Stoffer (2006):

est<-optim(par=c(.1,0,.1,.1), likfn, method="BFGS",
control=list(fnscale=-1), hessian=TRUE, yt=yt, a1=-0.35, P1=0.01, P1inf=0)

pars<-est$par
L<-matrix(c(pars[1],pars[2],0,pars[3]),ncol=2)
H<-L%*%t(L)
q11<-exp(pars[4])

kfd<-kf(yt = yt, Zt = matrix(1,nrow=2), Tt=1, Rt=1, Ht=H, Qt=q11,
a1 = -0.35, P1=0.01, P1inf=0)

kfd$Qt
kfd$Qt
kfd$Ht
kfd$Ht
#Estimated Qt and Ht differs

```

```

ts.plot(ts(yt[1,],start=1880),ts(yt[2,],start=1880),ts(kfd$at[1,],start=1880),ts(kfnd$at[1,],start=1880),col=c
#differs at start

#Example of stationary ARMA(1,1)

n<-1000
ar1<-0.8
ma1<-0.3
sigma<-0.5
yt<-arima.sim(model=list(ar=ar1,ma=ma1), n=n, sd=sigma)

dt <- matrix(0, nrow = 2)
Zt<-matrix(c(1,0),ncol=2)
Tt<-matrix(c(ar1,0,1,0),ncol=2)
Rt<-matrix(c(1,ma1),ncol=1)
a1<-matrix(c(0,0),ncol=1)
P1<-matrix(0,ncol=2,nrow=2)
P1[1,1]<-1/(1-ar1^2)*(1+ma1^2+2*ar1*ma1)
P1[1,2]<-ma1
P1[2,1]<-ma1
P1[2,2]<-ma1^2
f.out<-kf(yt = array(yt,dim=c(1,n)), Zt = Zt, Tt=Tt, Rt=Rt, Ht= 0,
Qt=sigma^2, a1 = a1, P1=P1)

```

ks

Kalman Smoother With Exact Diffuse Initialisation

Description

Performs Kalman smoothing with exact diffuse phase using univariate approach. Programmed with Fortran, uses subroutines from BLAS and LAPACK. See function 'kf' for smoothing.

Usage

```
ks(out)
```

Arguments

out Output from function 'kf'

Details

Function ks performs Kalman smoothing of gaussian (multivariate) state space model using the univariate approach by Koopman and Durbin (2000, 2001, 2003). In case where the distributions of some or all elements of initial state vector are unknown, ks uses exact diffuse phase using univariate approach by Koopman and Durbin (2000).

Value

A list with the output elements from Kalman filter and following new elements:

ahat $m \times n$ array of $E(\text{alphat} \mid y_1, y_2, \dots, y_n)$.
 Vt $m \times m \times n$ array of $\text{Var}(\text{alphat}_1, y_2, \dots, y_n)$.
 rt $m \times n + 1$ array of weighted sums of innovations $v_j, j=t+1, \dots, n$. Notice that in literature t in rt goes from $0, \dots, n$. Here $t=1, \dots, n+1$. Same applies to all r and N variables.
 rt0, rt1 $m \times d + 1$ arrays of diffuse phase decomposition of rt .
 Nt $m \times m \times n + 1$ array of $\text{Var}(rt)$.
 Nt0, Nt1, Nt2 $m \times m \times d + 1$ arrays of diffuse phase decomposition of Nt .

References

Koopman, S.J. and Durbin J. (2000). Fast filtering and smoothing for non-stationary time series models, *Journal of American Statistical Association*, 92, 1630-38.

Koopman, S.J. and Durbin J. (2001). *Time Series Analysis by State Space Methods*. Oxford: Oxford University Press.

Koopman, S.J. and Durbin J. (2003). Filtering and smoothing of state vector for diffuse state space models, *Journal of Time Series Analysis*, Vol. 24, No. 1.

Shumway, Robert H. and Stoffer, David S. (2006). *Time Series Analysis and Its Applications: With R examples*.

Examples

```
library(KFAS)

#Example of multivariate local level model
#Two series of average global temperature deviations for years 1880-1987
#See Shumway and Stoffer (2006), p. 327 for details

data(GlobalTemp)
yt<-array(GlobalTemp,c(2,108))

#Estimating the variance parameters

likfn<-function(par, yt, a1, P1, P1inf) #Function to optim
{
  L<-matrix(c(par[1],par[2],0,par[3]),ncol=2)
  H<-L%*%t(L)
  q11<-exp(par[4])
  lik<-kf(yt = yt, Zt = matrix(1,nrow=2), Tt=1, Rt=1, Ht=H, Qt=q11, a1 =
  a1, P1=P1, P1inf=P1inf, optcal=c(FALSE,FALSE,FALSE,FALSE))
}
```

```

lik$lik
}

est<-optim(par=c(.1,0,.1,.1), likfn, method="BFGS",
control=list(fnscale=-1), hessian=TRUE, yt=yt, a1=0, P1=0, P1inf=1)
#Diffuse initialisation

pars<-est$par
L<-matrix(c(pars[1],pars[2],0,pars[3]),ncol=2)
H<-L%*%t(L)
q11<-exp(pars[4])

kfd<-kf(yt = yt, Zt = matrix(1,nrow=2), Tt=1, Rt=1,
Ht=H, Qt=q11, a1 = 0, P1=0, P1inf=1)

ksd<-ks(kfd)

#Same as non-diffuse, initial values from Shumway and Stoffer (2006):

est<-optim(par=c(.1,0,.1,.1), likfn, method="BFGS",
control=list(fnscale=-1), hessian=TRUE, yt=yt, a1=-0.35, P1=0.01, P1inf=0 )

pars<-est$par
L<-matrix(c(pars[1],pars[2],0,pars[3]),ncol=2)
H<-L%*%t(L)
q11<-exp(pars[4])

kfnd<-kf(yt = yt, Zt = matrix(1,nrow=2), Tt=1, Rt=1, Ht=H, Qt=q11, a1 =
-0.35, P1=0.01, P1inf=0)

ksnd<-ks(kfnd)

kfd$Qt
kfnd$Qt
kfd$Ht
kfnd$Ht
#Estimated Qt and Ht differs

#smoothed values:
ts.plot(ts(yt[1,],start=1880),ts(yt[2,],start=1880),ts(ksd$ahat[1,],start=1880),ts(ksnd$ahat[1,],start=1880),c

```

Description

Generates random samples of state vector α from conditional density $p(\alpha | y)$.

Usage

```
sims smoother(yt, Zt, Tt, Rt, Ht, Qt, a1, P1, P1inf=0, nsim=1, tol=1e-7)
```

Arguments

yt	Matrix or array of observations.
Zt	System matrix or array of observation equation.
Tt	System matrix or array of transition equation.
Rt	System matrix or array of transition equation.
Ht	Variance matrix or array of disturbance terms eps_t of observation equation.
Qt	Variance matrix or array of disturbance terms eta_t .
a1	Initial state vector.
P1	Variance matrix of a1. In diffuse case P1star, the non-diffuse part of P1.
P1inf	Diffuse part of P1.
nsim	Number of samples. Default is 1.
tol	Tolerance parameter. Smallest covariance/variance value not counted for zero in diffuse phase. Default is 1e-7.

Details

Function `sims smoother` generates random samples of state vector $\alpha = (\alpha_1, \dots, \alpha_n)$ from conditional density $p(\alpha | y)$.

The state space model is given by

$$y_t = Z_t * \alpha_t + \text{eps}_t \text{ (observation equation)}$$

$$\alpha_{t+1} = T_t * \alpha_t + R_t * \text{eta}_t \text{ (transition equation)}$$

where $\text{eps}_t \sim N(0, H_t)$ and $\text{eta}_t \sim N(0, Q_t)$

Simulation smoother algorithm is from article by J. Durbin and S.J. Koopman (2002).

Value

$m * n * \text{nsim}$ array of simulated state vectors α .

References

Durbin J. and Koopman, S.J. (2002). A simple and efficient simulation smoother for state space time series analysis, *Biometrika*, Volume 89, Issue 3

Examples

```

library(KFAS)

#Example of multivariate local level model
#Two temperature datas from David S. Stoffer's webpage

y1<-scan("http://www.stat.pitt.edu/stoffer/tsa2/data/HL.dat")
y2<-scan("http://www.stat.pitt.edu/stoffer/tsa2/data/folland.dat")
yt<-rbind(y1,y2)

#Estimating the variance parameters

likfn<-function(par, yt, a1, P1, P1inf) #Function to optim
{
  L<-matrix(c(par[1],par[2],0,par[3]),ncol=2)
  H<-L%*%t(L)
  q11<-exp(par[4])
  lik<-kf(yt = yt, Zt = matrix(1,nrow=2), Tt=1, Rt=1,
  Ht=H, Qt=q11, a1 = a1, P1=P1, P1inf=P1inf, optcal=c(FALSE,FALSE,FALSE,FALSE))
  lik$lik
}

est<-optim(par=c(.1,0,.1,.1), likfn, method="BFGS", control=list(fnscale=-1),
hessian=TRUE, yt=yt, a1=0, P1=0, P1inf=1) #Diffuse initialisation

pars<-est$par
L<-matrix(c(pars[1],pars[2],0,pars[3]),ncol=2)
H<-L%*%t(L)
q11<-exp(pars[4])

kfd<-kf(yt = yt, Zt = matrix(1,nrow=2), Tt=1, Rt=1,
Ht=H, Qt=q11, a1 = 0, P1=0, P1inf=1)

ksd<-ks(kfd)

alpha<-simsmoother(yt = yt, Zt = matrix(1,nrow=2), Tt=1, Rt=1,
Ht=H, Qt=q11, a1 = 0, P1=0, P1inf=1, nsim=1000)

ahat<-NULL
for(i in 1:108) ahat[i]<-mean(alpha[1,i,])

ts.plot(ts(ahat),ts(ksd$ahat[1,]),col=c(1,2))

```

Index

*Topic **datasets**

GlobalTemp, [10](#)

distsmoother, [2](#)

eflik, [2](#)

eflik0, [5](#)

efsmoother, [7](#)

filter (kf), [11](#)

forecast, [9](#)

GlobalTemp, [10](#)

Kalman (kf), [11](#)

kf, [11](#)

ks, [16](#)

simsmoother, [18](#)