Kalman Filtering with miscFuncs

Benjamin M. Taylor
Lancaster University, UK

Abstract

This vignette provides a program template for use with the `KFadvance` function.

Keywords: LaTeX tables, Kalman filter, web scraping, development tools.

1. Introduction

The purpose of this vignette is to provide a template R functions for implementing the Kalman Filter and for parameter estimation for Gaussian dynamic linear models. The functions should provide a FLEXIBLE basis on which to build R code for optimal linear filtering.

After loading **miscFuncs**, the templates below can be printed to the R console (and hence copied and pasted into an editor) using `KFtemplates`:

```
library(miscFuncs)
KFtemplates()
```

2. The Statistical Model

Let Θ_t be the state vector and Y_t be the observation vector at time t.

The function `KFadvance` works with COLUMN VECTORS.

The statistical model is:

$$\Theta_t = A_t\Theta_{t-1} + B_t + C_t\epsilon_t, \qquad \epsilon_t \sim \mathrm{N}(0, W_t) \tag{1}$$

$$Y_t = D_t\Theta_{t-1} + E_t + F_t\nu_t, \qquad \nu_t \sim \mathrm{N}(0, V_t) \tag{2}$$

Suppose Θ_t has dimension $n \times 1$ and Y_t has dimension $m \times 1$, then the matrices in the above have dimensions:

matrix	dimension
A_t	$n \times n$
B_t	$n \times 1$
C_t	$n \times n$
W_t	$n \times n$
D_t	$m \times n$
E_t	$m \times 1$
F_t	$m \times m$
V_t	$m \times m$

The matrix D_t acts like a design matrix in ordinary least squares regression.

The user must also specify priors (aka intial values) for Θ in the form of a prior mean $n \times 1$ matrix (called `Xpost` in the code below[1]) and a prior variance $n \times n$ matrix (called `Vpost` in the code below).

Typically some, or all of A_t, B_t, C_t, W_t, D_t, E_t, F_t, V_t will be parametrised. Part of the goal of filtering will typically involve estimating these parameters by maximum likelihood. To allow optimisation to work well, model parameters should be free to roam between real numbers. For example if a parameter represents a variance (ie should onyl take positive values) then the inside the `KFfit` function, use for example `sigma <- exp(param[1])` or if the parameter is on the $[0, 1]$ then use the inverse logistic function, `exp(param[2])/(1+exp(param[2]))`. The template for parameter estimation comes later.

3. Fitting the Model to Some Data

This section provides template code for Kalman filtering under the above model.

```
# see vignette for notation

KFfit <- function(  param,              # model parameters
                          data,               # a matrix containing the data ie Y
                          ,...,               # delete this line and include other arguments to be passed
                                      #     to the function here
                          optim=FALSE,        # set this to FALSE if not estimating parameters, otherwise,
                                              #     if parameters have already been estimated, then set to TRUE
                          history.means=FALSE,# whether to save a matrix of filtered E(Theta_t)
                          history.vars=FALSE, # whether to save a list of filtered V(Theta_t)
                          prior.mean=NULL,    # optional prior mean. column vector.    # Normally set
                          prior.var=NULL,     # optional prior mean. matrix.           # these inside the code
                          fit=FALSE,          # whether to return a matrix of fitted values
                          se.fit=FALSE,       # whether to return the standard error of the fitted values
                          se.predict=FALSE,   # whether to return the prediction standard error =
                                              #     se(fitted values) + observation variance V_t
                          noisy=TRUE          # whether to print a progress bar, useful.
                          na.rm=FALSE){       # whether to use NA handling. set to TRUE if any Y is NA

    start <- Sys.time()

    if(se.predict & !se.fit){
        stop("Must have se.fit=TRUE in order to compute se.predict") # leads to a computational saving
    }

    #
    # Here, I would suggest creating dummy names for your paramters eg
    # sigma.obs <- exp(model.param[1])
    #
    #

    T <- dim(data)[1]   # ASSUMES OBSERVATIONS ARE IN A (T x m) matrix, ie row t contains the data for time t.
                        # Note this is important for when KFadvance is called later
```

[1] this might seem a strange name, given that it is the prior, but in the template code below, this object is overwritten and eventually becomes the posterior mean.

```
if(is.null(prior.mean)){
    Xpost <- DEFINE PRIOR MEAN HERE
}
else{
    Xpost <- prior.mean
}

if(is.null(prior.var)){
    Vpost <- DEFINE PRIOR VARIANCE HERE
}
else{
    Vpost <- prior.var
}

if (history.means){
    Xrec <- Xpost
}
if(history.vars){
    Vrec <- list()
    Vrec[[1]] <- Vpost
}

# delete or complete the following rows as necessary, also appears in the loop that follows
A <- IF A IS FIXED OVER TIME THEN DEFINE IT HERE
B <- IF B IS FIXED OVER TIME THEN DEFINE IT HERE
C <- IF C IS FIXED OVER TIME THEN DEFINE IT HERE
W <- IF W IS FIXED OVER TIME THEN DEFINE IT HERE
D <- IF D IS FIXED OVER TIME THEN DEFINE IT HERE
E <- IF E IS FIXED OVER TIME THEN DEFINE IT HERE
F <- IF F IS FIXED OVER TIME THEN DEFINE IT HERE
V <- IF V IS FIXED OVER TIME THEN DEFINE IT HERE

loglik <- 0
fitmat <- c()
sefitmat <- c()
sepredictmat <- c()

if(noisy){
    pb <- txtProgressBar(min=1,max=T,style=3)
}

for(t in 1:T){

    # delete or complete the following rows as necessary
    A <- IF A IS TIME-VARYING THEN DEFINE IT HERE
    B <- IF B IS TIME-VARYING THEN DEFINE IT HERE
    C <- IF C IS TIME-VARYING THEN DEFINE IT HERE
    W <- IF W IS TIME-VARYING THEN DEFINE IT HERE
    D <- IF D IS TIME-VARYING THEN DEFINE IT HERE
    E <- IF E IS TIME-VARYING THEN DEFINE IT HERE
    F <- IF F IS TIME-VARYING THEN DEFINE IT HERE
    V <- IF V IS TIME-VARYING THEN DEFINE IT HERE

    # this bit calls KF advance
    new <- KFadvance(obs=data[t,],oldmean=Xpost,oldvar=Vpost,A=A,B=B,C=C,D=Dt,E=E,F=F,W=W,V=V,marglik=TRUE,log=TRUE,na.rm=na.rm)

    Xpost <- new$mean
            Vpost <- new$var

            if(t==1){ # used when this function is called iteratively one step at a time
                running.mean <- Xpost
                running.var <- Vpost
            }

            loglik <- loglik + new$mlik


            if (history.means){
        Xrec <- cbind(Xrec,Xpost)
    }
    if(history.vars){
        Vrec[[t+1]] <- Vpost # since first entry is the prior
    }

    if(fit){
        fitmat <- cbind(fitmat,D%*%Xpost + E)
    }
    if(se.fit){
        sefitmat <- cbind(sefitmat,sqrt(diag(D%*%Vpost%*%t(D))))
    }
    if(se.predict){
        sepredictmat <- cbind(sepredictmat,sqrt((sefitmat[,ncol(sefitmat)])^2+diag(F%*%V%*%t(F))))
    }

    if(noisy){
        setTxtProgressBar(pb,t)
    }
}

if(noisy){
```

```
        close(pb)
    }
    end <- Sys.time()

    if(noisy){
        cat("Time taken:",difftime(end,start,units="secs")," seconds.\n")
    }

    if(optim){
        return(-loglik) # just return the -log likelihood if in parameter estimation mode
    }
    else{
        retlist <- list(mean=Xpost,var=Vpost,mlik=loglik,data=data,running.mean=running.mean,running.var=running.var)
        if(history.means){
            retlist$history.means <- Xrec
        }
        if(history.vars){
            retlist$history.vars <- Vrec
        }
        if(fit){
            retlist$fit <- fitmat
        }
        if(se.fit){
            retlist$se.fit <- sefitmat
        }
        if(se.predict){
            retlist$se.predict <- sepredictmat
        }
        return(retlist)
    }
}
```

4. Parameter Estimation

This section provides template code for parameter estimation in Kalman filtering.

```
KFparest <- function(   data, # data ie Y
                        ...){ # delete and paste in OTHER ARGUMENTS TO BE PASSED TO KFfit
    start <- Sys.time()

    inits <- PUT INITIAL VALUES FOR PARAMETER VECTOR HERE

    # use optim to find optimal parameters
    oppars <- optim(inits,
                    KFfit,
                    data=data,
                    OTHER ARGUMENTS TO BE PASSED TO KFfit,  # delete and paste in OTHER ARGUMENTS
                                                            # TO BE PASSED TO KFfit
                    optim=TRUE,
                    control=list(trace=100))

    end <- Sys.time()
    cat("\n")
    cat("Time Taken",difftime(end,start,units="mins"),"\n")
    cat("\n")

    return(oppars)
}
```


Affiliation:

Benjamin M. Taylor
Division of Medicine,
Lancaster University,
Lancaster,
LA1 4YF,
UK
E-mail: b.taylor1@lancaster.ac.uk
URL: http://www.lancs.ac.uk/staff/taylorb1/