**STATE-SPACE MODEL AND KALMAN FILTERS**

A Kalman filter is an optimal estimation algorithm used to estimate states of a system from indirect and uncertain measurements.

Package:dlm()

The function dlm is used to create Dynamic Linear Model objects. as.dlm and is.dlm coerce an object to a Dynamic Linear Model object and test whether an object is a Dynamic Linear Model.

There several functions in dlm package:

dlmModPoly :Create an n-th order polynomial DLM

Description: The function creates an nth order polynomial DLM.

Usage :dlmModPoly(order = 2, dV = 1, dW = c(rep(0, order - 1), 1), m0 = rep(0, order), C0 = 1e+07 \* diag(nrow = order))

Arguments: order :order of the polynomial model. The default corresponds to a stochastic linear trend. dV variance of the observation noise. dW diagonal elements of the variance matrix of the system noise. m0 m0, the expected value of the pre-sample state vector. C0 C0, the variance matrix of the pre-sample state vector.

dlmFilter: DLM filtering

Description: The functions applies Kalman filter to compute filtered values of the state vectors, together with their variance/covariance matrices. By default the function returns an object of class "dlmFiltered".

Usage dlmFilter(y, mod, debug = FALSE, simplify = FALSE)

Arguments y the data. y can be a vector, a matrix, a univariate or multivariate time series.

Mod: an object of class dlm, or a list with components m0, C0, FF, V, GG, W, and optionally JFF, JV, JGG, JW, and X, defining the model and the parameters of the prior distribution.

Debug: if FALSE, faster C code will be used, otherwise all the computations will be performed in R. simplify :should the data be included in the output

dlmSmooth: DLM smoothing

Description: The function apply Kalman smoother to compute smoothed values of the state vectors, together with their variance/covariance matrices.

dlmForecast: Prediction and simulation of future observations

Description The function evaluates the expected value and variance of future observations and system states. It can also generate a sample from the distribution of future observations and system states.

Usage :dlmForecast(mod, nAhead = 1, method = c("plain", "svd"), sampleNew = FALSE)

Arguments :mod an object of class "dlm", or a list with components m0, C0, FF, V, GG, and W, defining the model and the parameters of the prior distribution.

mod :can also be an object of class "dlmFiltered", such as the output from dlmFilter. nAhead number of steps ahead for which a forecast is requested.

Method: method="svd" uses singular value decomposition for the calculations. Currently, only method="plain" is implemented.

dlmMLE: Parameter estimation by maximum likelihood

Description :The function returns the MLE of unknown parameters in the specification of a state space model.

Usage: dlmMLE(y, parm, build, method = "L-BFGS-B", ..., debug = FALSE)

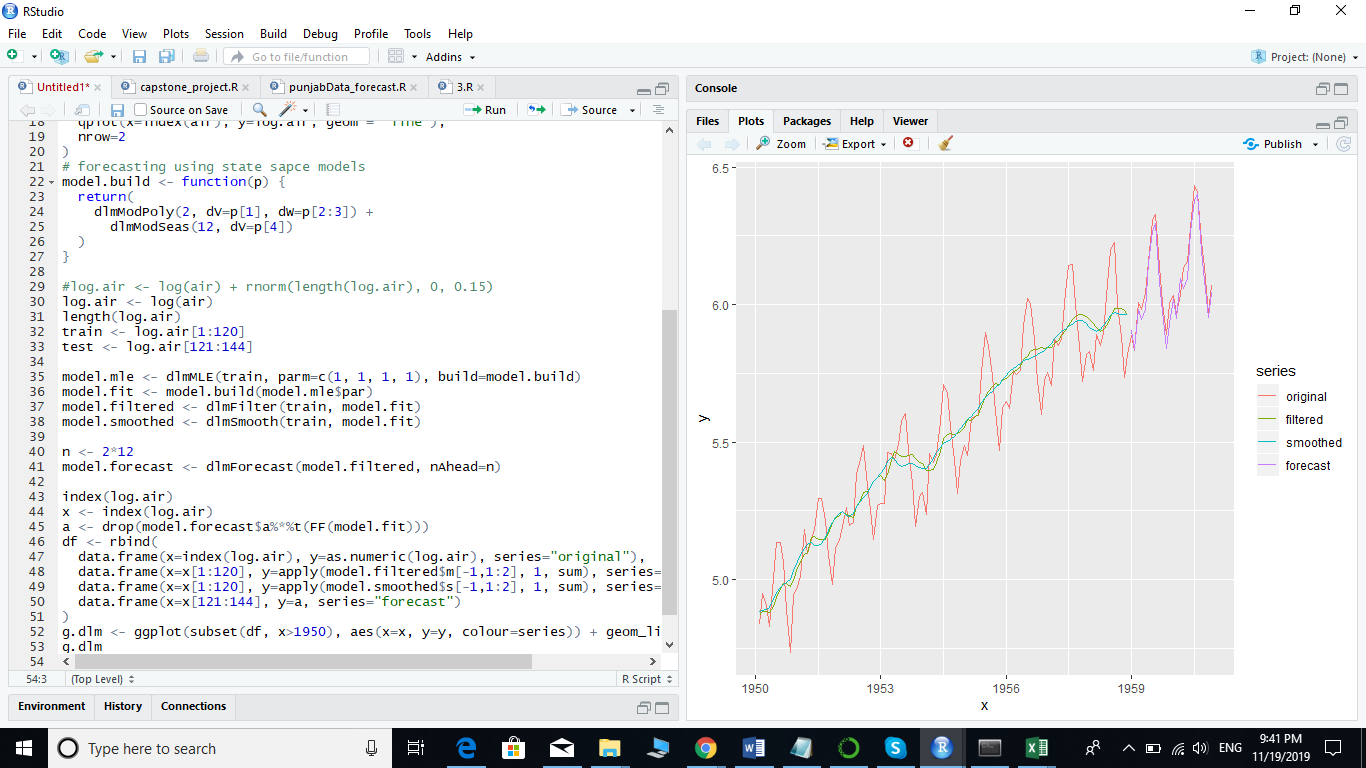
Arguments: y a vector, matrix, or time series of data.

parm :vector of initial values - for the optimization routine - of the unknown parameters.

Build: a function that takes a vector of the same length as parm and returns an object of class dlm, or a list that may be interpreted as such.

method :passed to optim. ... additional arguments passed to optim and build.

debug :if debug=TRUE, the likelihood calculations are done entirely in R, otherwise C functions are use.



Above screenshot is the code and the forecasts using inbuilt data set i.e. AirPassenger. Which I have run.

Code :

data("AirPassengers")

head(AirPassengers)

dim(AirPassengers)

library(dplyr)

library(ggplot2)

air <- AirPassengers

log.air <- log(air)

grid.arrange(

qplot(x=index(air), y=air, geom = "line"),

qplot(x=index(air), y=log.air, geom = "line"),

nrow=2

)

# forecasting using state sapce models

model.build <- function(p) {

return(

dlmModPoly(2, dV=p[1], dW=p[2:3]) +

dlmModSeas(12, dV=p[4])

)

}

#log.air <- log(air) + rnorm(length(log.air), 0, 0.15)

log.air <- log(air)

length(log.air)

train <- log.air[1:120]

test <- log.air[121:144]

#Here we need to calculate parm:

model.mle <- dlmMLE(train, parm=c(1, 1, 1, 1), build=model.build)

model.fit <- model.build(model.mle$par)

model.filtered <- dlmFilter(train, model.fit)

model.smoothed <- dlmSmooth(train, model.fit)

n <- 2\*12

model.forecast <- dlmForecast(model.filtered, nAhead=n)

index(log.air)

x <- index(log.air)

a <- drop(model.forecast$a%\*%t(FF(model.fit)))

df <- rbind(

data.frame(x=index(log.air), y=as.numeric(log.air), series="original"),

data.frame(x=x[1:120], y=apply(model.filtered$m[-1,1:2], 1, sum), series="filtered"),

data.frame(x=x[1:120], y=apply(model.smoothed$s[-1,1:2], 1, sum), series="smoothed"),

data.frame(x=x[121:144], y=a, series="forecast")

)

g.dlm <- ggplot(subset(df, x>1950), aes(x=x, y=y, colour=series)) + geom\_line()

g.dlm

Below are the few links that I have referenced for the state-space model and Kalman filters.

<http://gradientdescending.com/state-space-models-for-time-series-analysis-and-the-dlm-package/>

<http://lalas.github.io/quantitativeThoughts/r/2014/09/01/dlmTutorial.html>

<https://in.mathworks.com/videos/understanding-kalman-filters-part-1-why-use-kalman-filters--1485813028675.html>

One package used for state space modelling:

R package KFAS for state space modelling:

<https://cran.r-project.org/web/packages/KFAS/vignettes/KFAS.pdf>