Timeseries-lab3

Prudhvi Peddmallu(prupe690),Zhixuan Duan(zhidu838)

01 October 2019

Library

```r
options(scipen=999)
library("tidyverse")
library("gridExtra")
library("knitr")
library("astsa")
library("matlib")
library("forecast")
```

1-Implementation of Kalman filter

Implementation of Kalman filter

Assignment 1

In table 1 a script for generation of data from simulation o fthe following state space model and implementation of the Kalman filter on the data is given.

$$\begin{aligned}
\mathbf{z}_t &= A_{t-1}\mathbf{z}_{t-1} + e_t, \\
\mathbf{x}_t &= C_t\mathbf{z}_t + \nu_t, \\
\nu_t &\sim N(0, R_t), \\
e_t &\sim N(0, Q_t).
\end{aligned}$$

a. Write down the expression for the state space model that is being simulated.
b. Run this scrip and compare the filtering results with a moving average smoother of order 5.
c. Also, compare the filtering outcome when R in the filter is 10 times smaller than its actual value while Q in the filter is 10 times larger than its actual value. How does the filtering outcome varies?
d. Now compare the filtering outcome when R in the filter is 10 times larger than its actual value while Q in the filter is 10 times smaller than its actual value. How does the filtering outcome varies?
e. Implement your own Kalman filter and replace ksmooth0 function with your script.
f. How do you interpret the Kalman gain?

In Table 2 the Kalman filtering algorithm is given for reference.

1a)-The expression for the state space model that is being simulated.

Assignment 1. Computations with simulated data

In table 1 a script for generation of data from simulation of the following state space model and implementation of the Kalman filter on the data is given.

$$Z_t = A_{t-1} Z_{t-1} + e_t$$
$$x_t = C_t z_t + \nu_t$$

$$\nu_t \sim N(0, R_t)$$
$$e_t \sim N(0, Q_t)$$

\fbox{\begin{minipage}{46.7em} ##a) Write down the expression for the state space model that is being simulated.

$$Z_t = 1 Z_{t-1} + e_t$$

$$x_t = 1 z_t + \nu_t$$

Where

$$v_t, e_t$$

$$\nu_t \sim N(0, 1)$$

$$e_t \sim N(0, 1)$$

But Zt is a running sum of et with the first value removed thus:

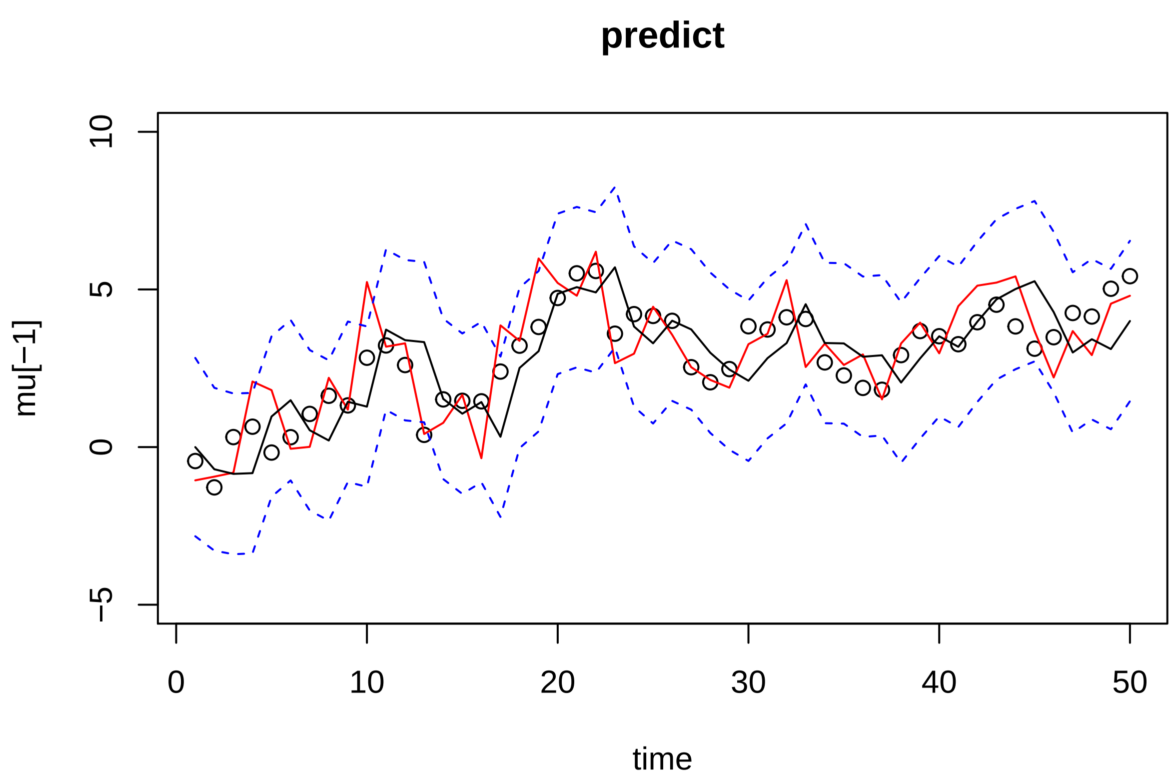
$$Z_t = \sum_{k=2}^{i} e_t[k], i = 2, 3, ...N = 50$$

Therefore:

$$X_t = 1 * \sum_{k=2}^{i} e_t[k] + \nu_t, i = 2, 3, ...N = 50$$

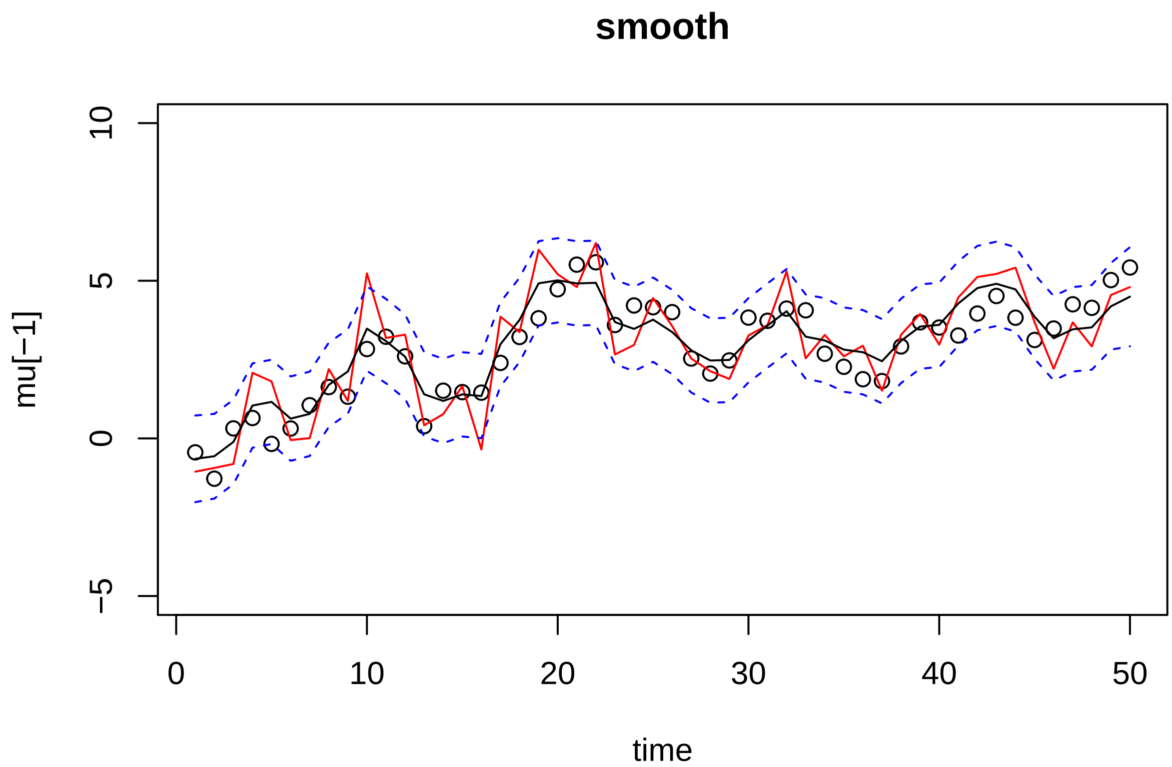
1b)-Run this scrip and compare the filtering results with a moving average smoother of order 5.

```r
#colours
cbbPalette <- c("#000000", "#E69F00", "#56B4E9", "#009E73",
"#F0E442", "#0072B2", "#D55E00", "#CC79A7")
set.seed(12345)
# create a dataset
set.seed(1)
num = 50
#random number generator
w = rnorm(num+1,0,1)
v = rnorm(num ,0,1)
mu = cumsum(w) # state space models mu[0],..., mu[50]
y = mu[-1] + v # observations  y[1],..., y[50]
# function Ksmooth0 does both filter and smooth
KS = Ksmooth0(num , y, A=1, mu0=0, Sigma0=1, Phi=1, cQ=1, cR=1)

time = 1:num
#predict-plot
plot(time , mu[-1], main="predict", ylim=c(-5,10))
lines(time ,y,col="red")
lines(KS$xp)
lines(KS$xp+2*sqrt(KS$Pp), lty=2, col=4)
lines(KS$xp -2*sqrt(KS$Pp), lty=2, col=4)
```

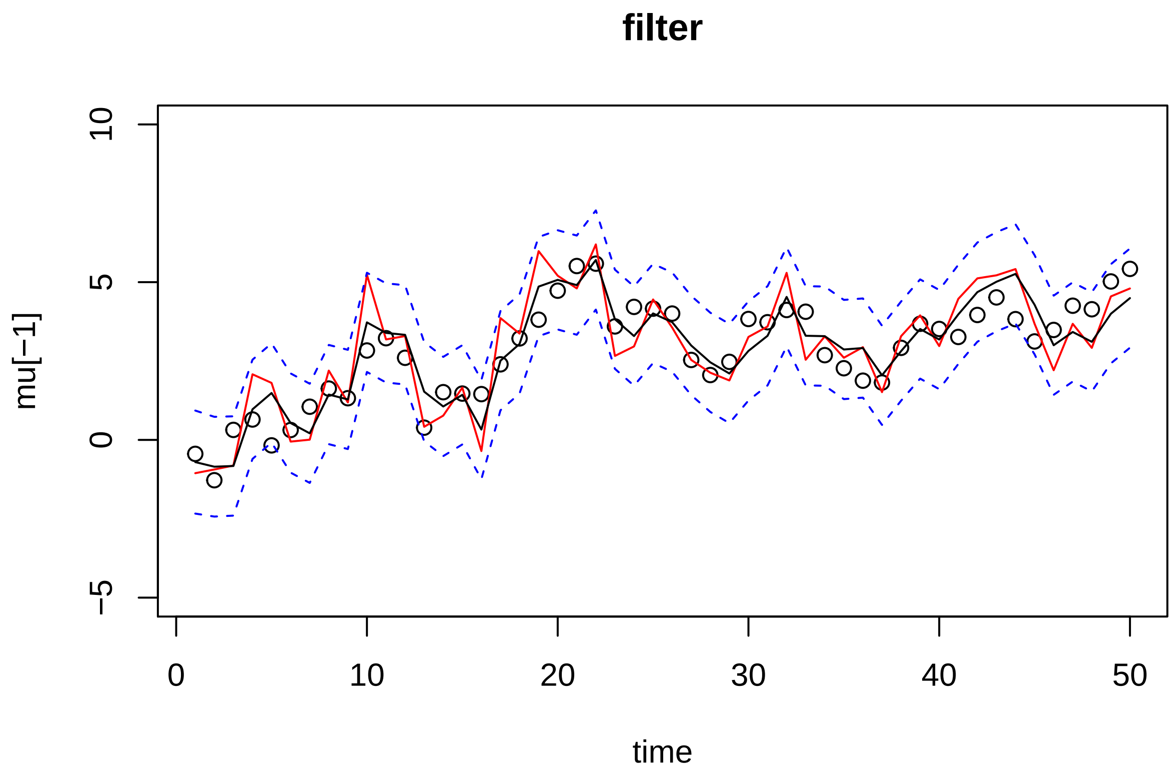



```r
#smooth-plot
plot(time , mu[-1], main="smooth", ylim=c(-5,10))
lines(time ,y,col="red")
```

```
lines(KS$xs)
lines(KS$xs+2*sqrt(KS$Ps), lty=2, col=4)
lines(KS$xs-2*sqrt(KS$Ps), lty=2, col=4)
```




```
#filter-plot
plot(time , mu[-1], main="filter", ylim=c(-5,10))
lines(time ,y,col="red")
lines(KS$xf)
lines(KS$xf+2*sqrt(KS$Pf), lty=2, col=4)
lines(KS$xf -2*sqrt(KS$Pf), lty=2, col=4)
```




```r
#mu values
mu[1]
```

```
## [1] -0.6264538
```

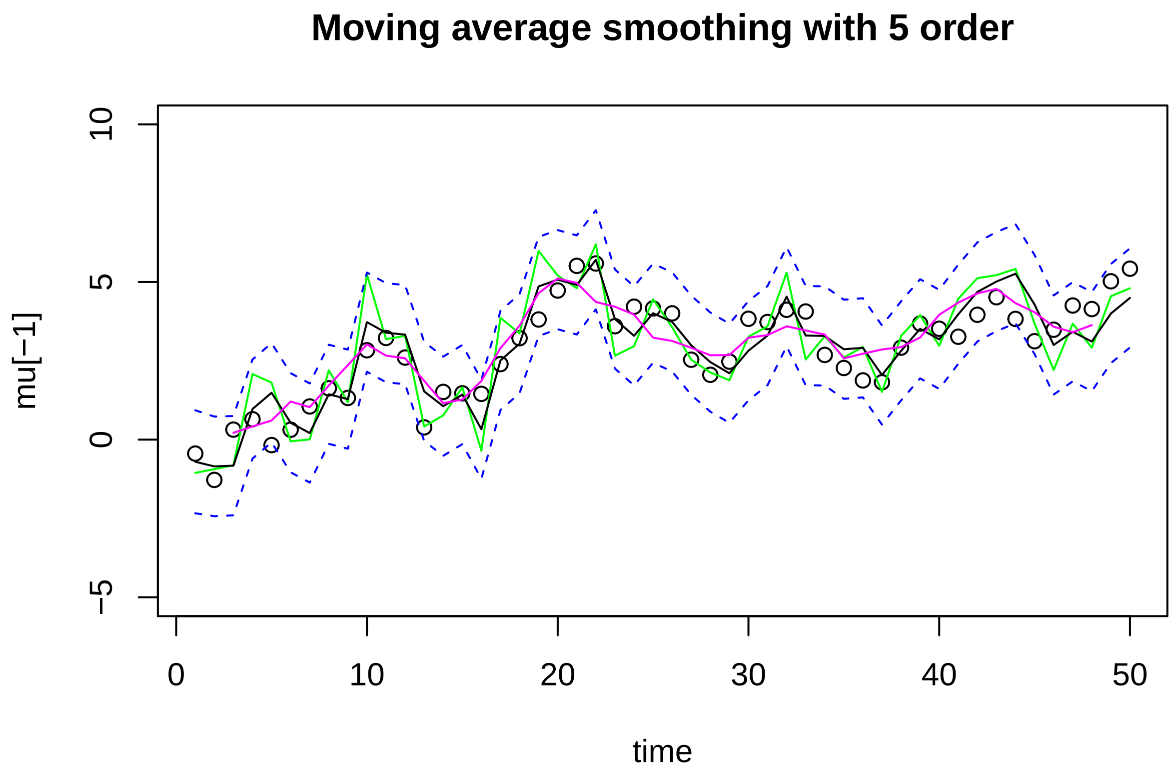
```r
KS$x0n
```

```
##            [,1]
## [1,] -0.3241541
```

```r
# initial value
sqrt(KS$P0n)
```

```
##           [,1]
## [1,] 0.7861514
```

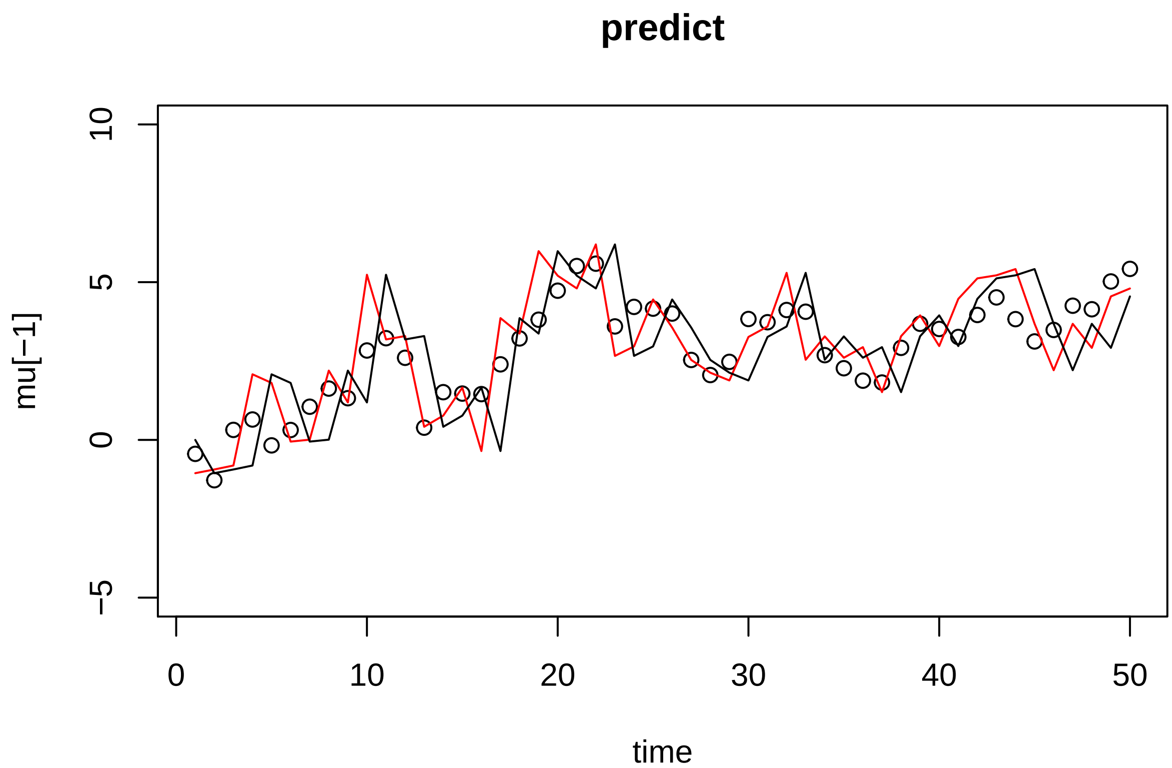
```r
# filtering results with a moving average smoother of 5 order
plot(time , mu[-1], ylim=c(-5,10), main="Moving average smoothing with 5 order")
lines(time ,y,col="green")
lines(KS$xf)
lines(KS$xf+2*sqrt(KS$Pf), lty=2, col=4)
lines(KS$xf -2*sqrt(KS$Pf), lty=2, col=4)
lines(ma(y, order=5), col=6)
```



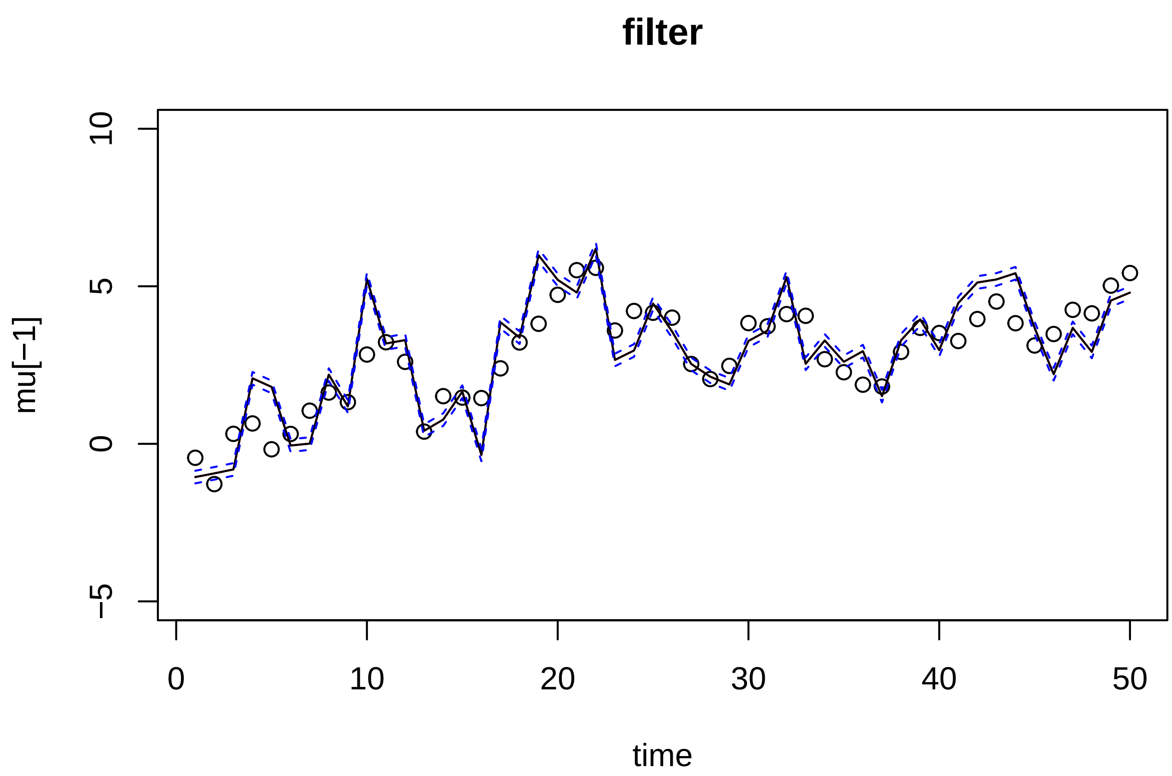
Analysis:-We find that the moving normal smoothing capacity with order 5 is the most noticeably terrible fit since its losing all the changeability that is caught by our kalman filter.

1c)-Also, compare the filtering outcome when R in the filter is 10 times smaller than its actual valuewhile Q in the filter is 10 times larger than its actual value. How does the fiering outcome varies?

```
# function Ksmooth0 does both filter and smooth
KS = Ksmooth0(num , y, A=1, mu0=0, Sigma0=1, Phi=1, cQ=10, cR=0.1)

time = 1:num
#predict-plot
plot(time , mu[-1], main="predict", ylim=c(-5,10))
lines(time ,y,col="red")
lines(KS$xp)
lines(KS$xp+2*sqrt(KS$Pp), lty=2, col=4)
lines(KS$xp -2*sqrt(KS$Pp), lty=2, col=4)
```



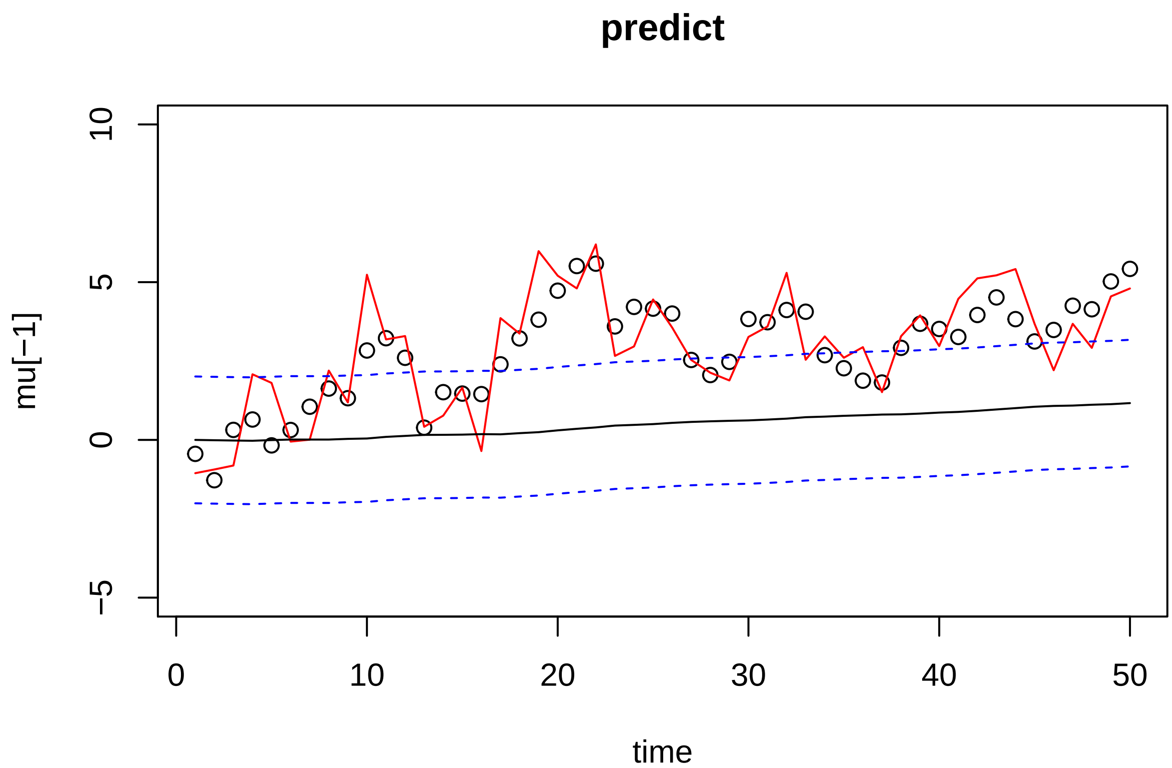

```r
#filter plot
plot(time , mu[-1], main="filter", ylim=c(-5,10))
lines(time ,y,col="red")
lines(KS$xf)
lines(KS$xf+2*sqrt(KS$Pf), lty=2, col=4)
lines(KS$xf -2*sqrt(KS$Pf), lty=2, col=4)
```



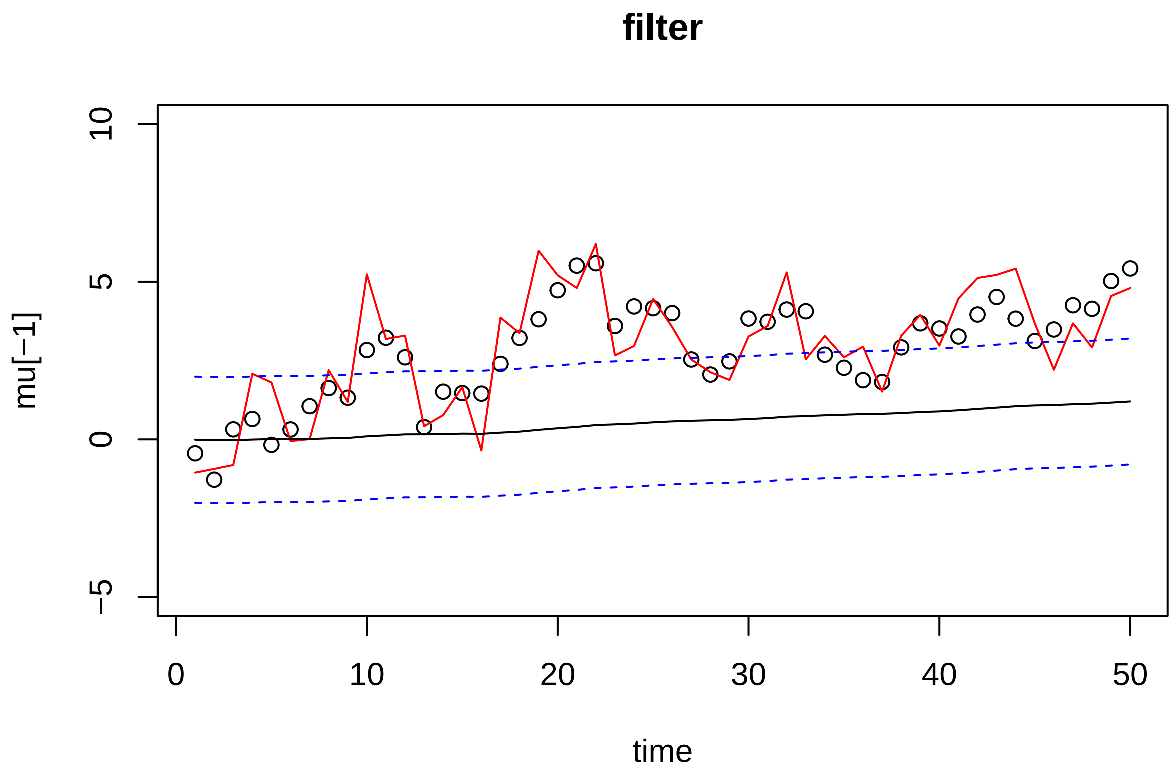
Here we find that the separating yield takes after the true worth considerably more than the previously run values.

1d)- Now compare the filtering outcome when R in the filter is 10 times larger than its actual value, while Qin the filter is 10 times smaller than its actual value. How does the filtering outcome varies?

```
# function Ksmooth0 does both filter and smooth
KS= Ksmooth0(num , y, A=1, mu0=0, Sigma0=1, Phi=1, cQ=0.1, cR=10)

time = 1:num
#predict plot
plot(time , mu[-1], main="predict", ylim=c(-5,10))
lines(time ,y,col="red")
lines(KS$xp)
lines(KS$xp+2*sqrt(KS$Pp), lty=2, col=4)
lines(KS$xp -2*sqrt(KS$Pp), lty=2, col=4)
```



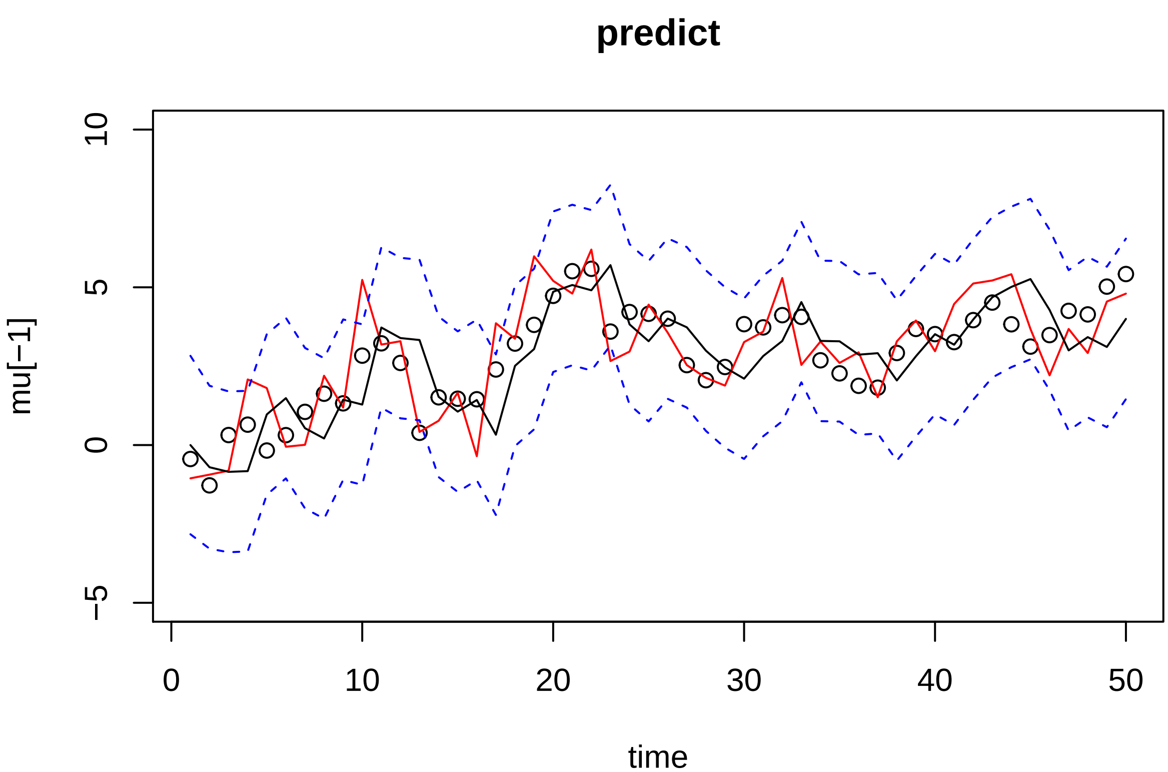

```
#filter-plot
plot(time , mu[-1], main="filter", ylim=c(-5,10))
lines(time ,y,col="red")
lines(KS$xf)
lines(KS$xf+2*sqrt(KS$Pf), lty=2, col=4)
lines(KS$xf -2*sqrt(KS$Pf), lty=2, col=4)
```



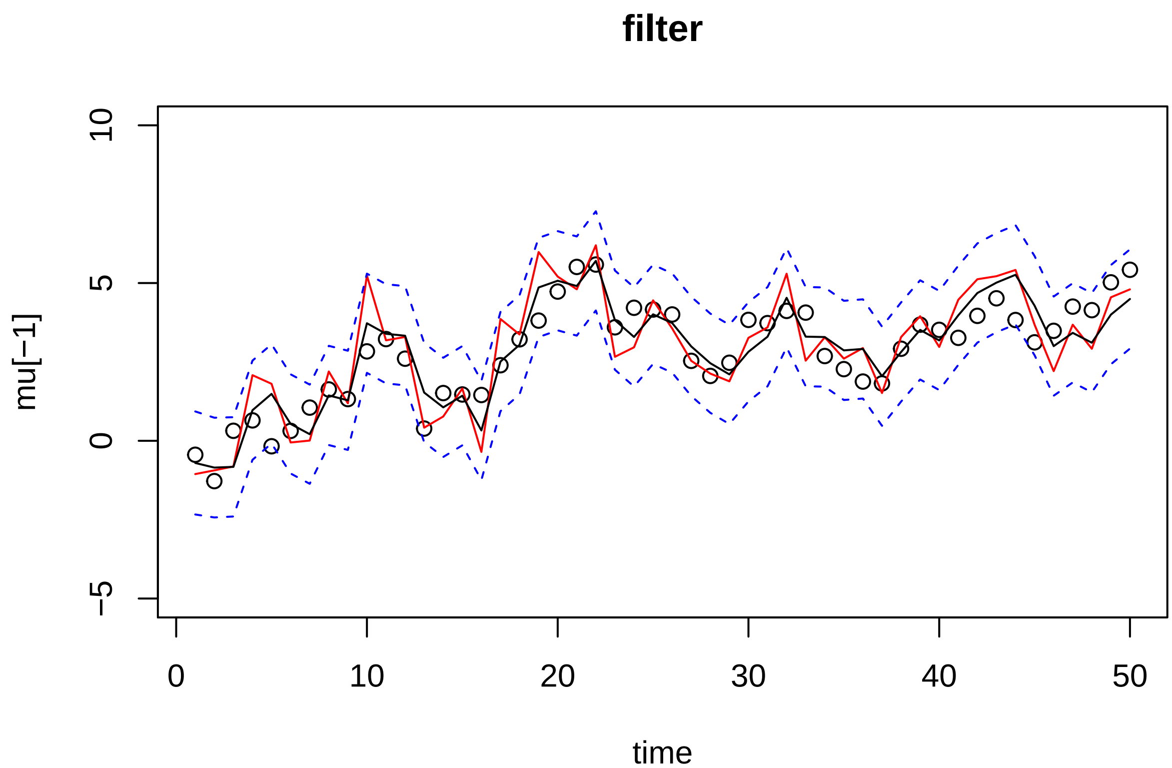
1e) Implement your own Kalman filter and replace ksmooth0 function with your script

```
kalman_filter <- function(num, y, A, mu0, Sigma0, Phi, cQ, cR)
{
    kf = astsa::Kfilter0(num, y, A, mu0, Sigma0, Phi, cQ, cR)
    pdim = nrow(as.matrix(Phi))
    xs = array(NA, dim = c(pdim, 1, num))
    Ps = array(NA, dim = c(pdim, pdim, num))
    J = array(NA, dim = c(pdim, pdim, num))
    xs[, , num] = kf$xf[, , num]
    Ps[, , num] = kf$Pf[, , num]
    for (k in num:2) {
        J[, , k - 1] = (kf$Pf[, , k - 1] %*% t(Phi)) %*% solve(kf$Pp[,
            , k])
        xs[, , k - 1] = kf$xf[, , k - 1] + J[, , k - 1] %*% (xs[,
            , k] - kf$xp[, , k])
        Ps[, , k - 1] = kf$Pf[, , k - 1] + J[, , k - 1] %*% (Ps[,
            , k] - kf$Pp[, , k]) %*% t(J[, , k - 1])
    }
    x00 = mu0
    P00 = Sigma0
    J0 = as.matrix((P00 %*% t(Phi)) %*% solve(kf$Pp[, , 1]),
        nrow = pdim, ncol = pdim)
```

```r
    x0n = as.matrix(x00 + J0 %*% (xs[, , 1] - kf$xp[, , 1]),
        nrow = pdim, ncol = 1)
    P0n = P00 + J0 %*% (Ps[, , 1] - kf$Pp[, , 1]) %*% t(J0)
    return(list(xs = xs, Ps = Ps, x0n = x0n, P0n = P0n, J0 = J0, J = J,
        xp = kf$xp, Pp = kf$Pp, xf = kf$xf, Pf = kf$Pf, like = kf$like,
        Kn = kf$K))
}

set.seed(1)
num = 50
w = rnorm(num+1,0,1)
v = rnorm(num ,0,1)
mu = cumsum(w)
y = mu[-1] + v
time = 1:num

KS_fun = kalman_filter(num=num , y=y, A=1, mu0=0, Sigma0=1, Phi=1, cQ=1, cR=1)

time = 1:num
#predict plot
plot(time , mu[-1], main="predict", ylim=c(-5,10))
lines(time ,y,col="red")
lines(KS_fun$xp)
lines(KS_fun$xp+2*sqrt(KS_fun$Pp), lty=2, col=4)
lines(KS_fun$xp -2*sqrt(KS_fun$Pp), lty=2, col=4)
```



```r
#filter-plot
plot(time , mu[-1], main="filter", ylim=c(-5,10))
lines(time ,y,col="red")
lines(KS_fun$xf)
lines(KS_fun$xf+2*sqrt(KS_fun$Pf), lty=2, col=4)
lines(KS_fun$xf -2*sqrt(KS_fun$Pf), lty=2, col=4)
```



1f)How do you interpret the Kalman gain?

Analysis: Kalman gain is given by $K = \frac{P_k H_k^T}{P_k H_k^T + R_k}$ where you will realize that the relative magnitudes of matrices R_k and P_k control a relation between the filter's use of predicted state estimate z_t and measurement x_t.

When R_k tends to zero then $x_t = x_{t-1} + K(y_t - H_k)$ suggests that when the magnitude of R is small, meaning that the measurements are accurate, the state estimate depends mostly on the measurements.

When the state is known accurately, then numerator is small compared to R, and the filter mostly ignores the measurements relying instead on the prediction derived from the previous state.