TSA_lab3

Zijie Feng

2019/10/2

Assignment 1

a)

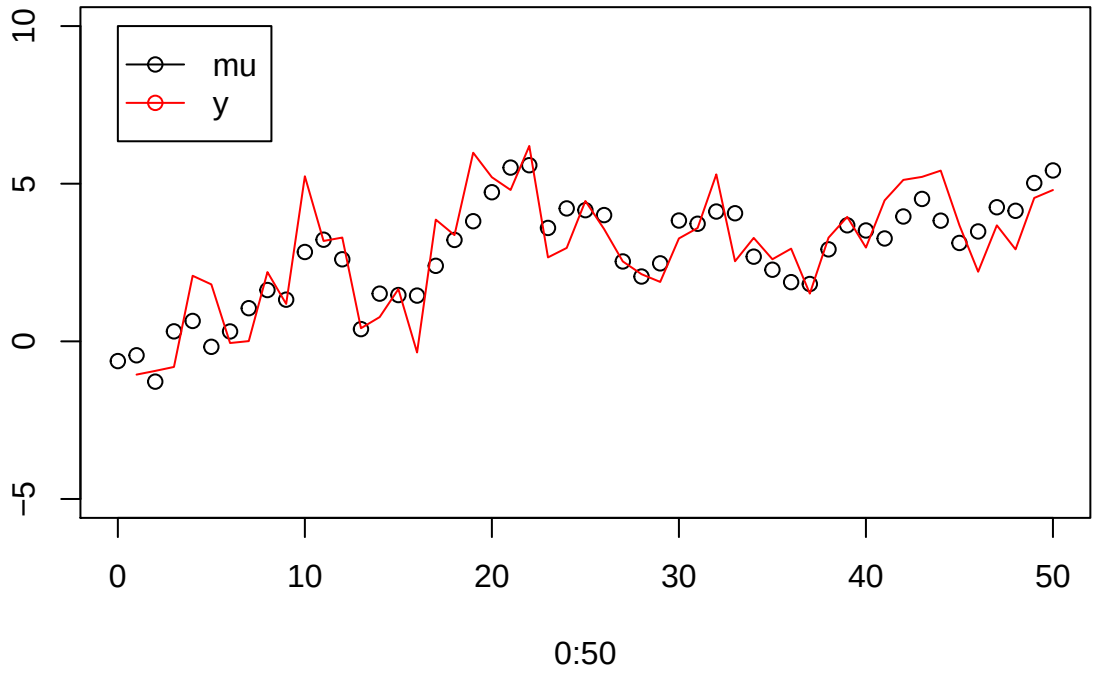
Based on the code above, we get the relation for hidden states μ's is

$$\mu_t = \mu_{t-1} + e_t$$

and the relation between μ_t and y is

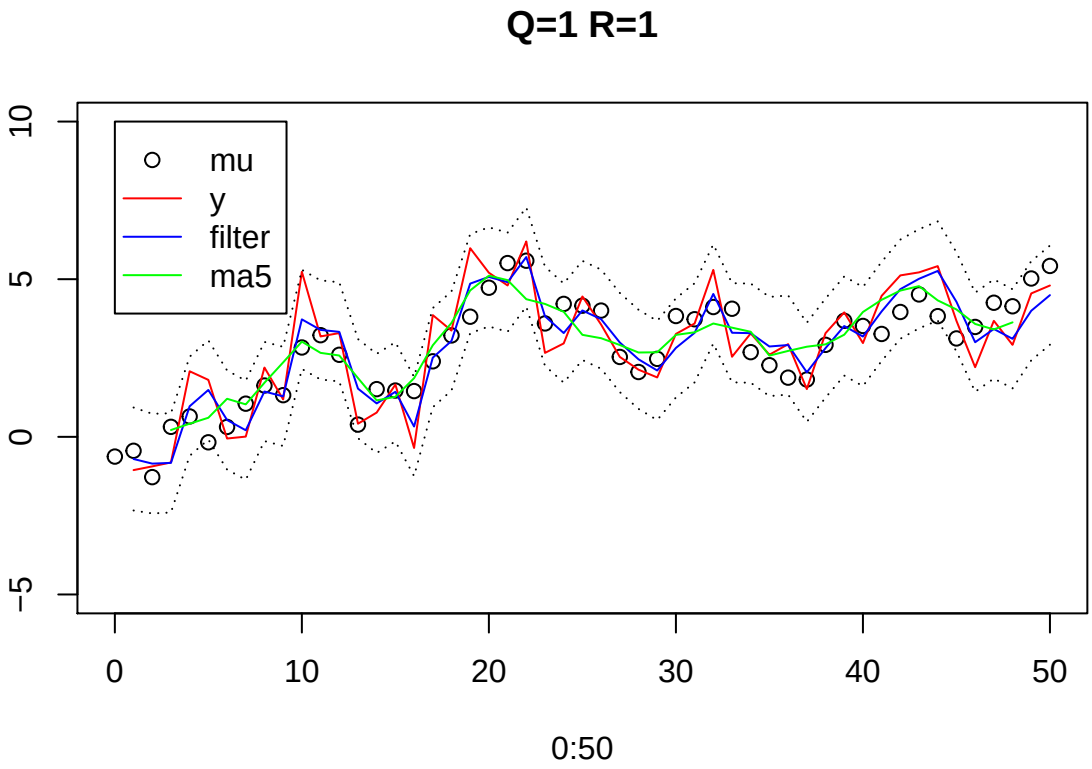
$$y_t = \mu_t + v_t$$

where $\mu_1 \sim N(0, 1)$. It also means the matrices A and C of SSM are just 1×1 matrices, and the variances of $e_t \sim N(0, 1)$ and $v_t \sim N(0, 1)$ are just constants.



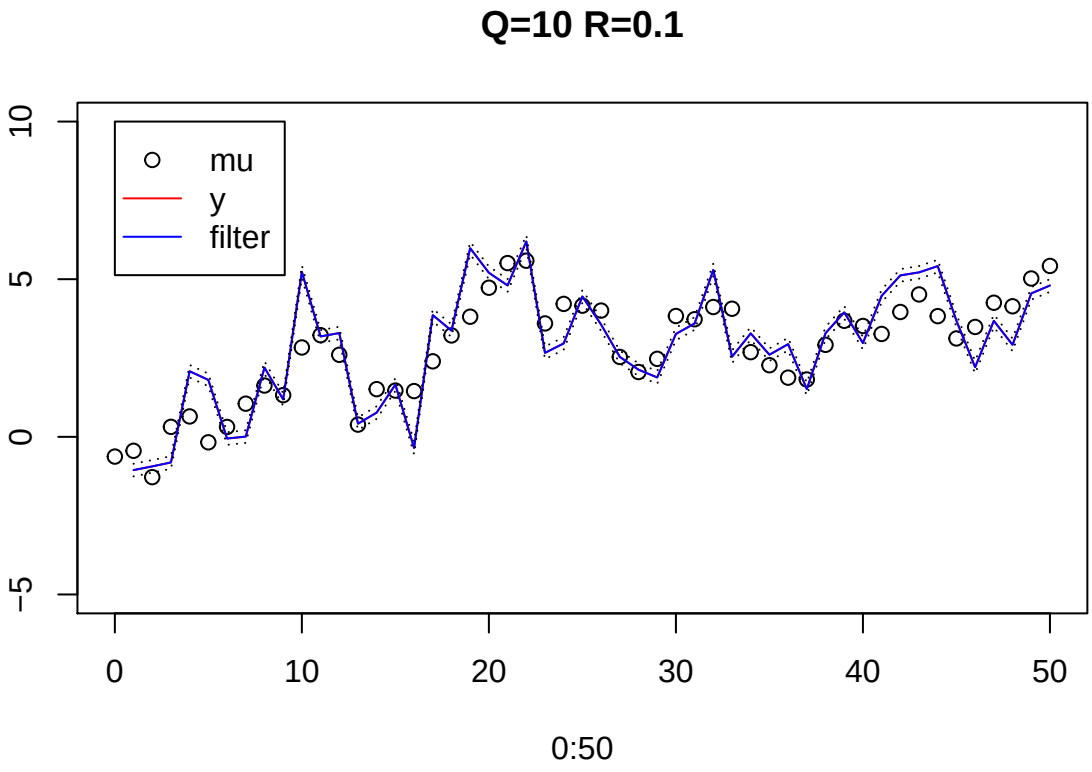
Points represent the latent/hidden states and the red line represents the corresponding observations.

b)



The blue solid line is created by `KsmoothO` (Kalman Filter) and the green line is created by a moving average smoother via observations y with order 5. The green line is stabler than the filtered line, but it fits worse visually. In addition, two dashed lines are the 95% confident interval for the filtered line and all the states are covered in such interval.

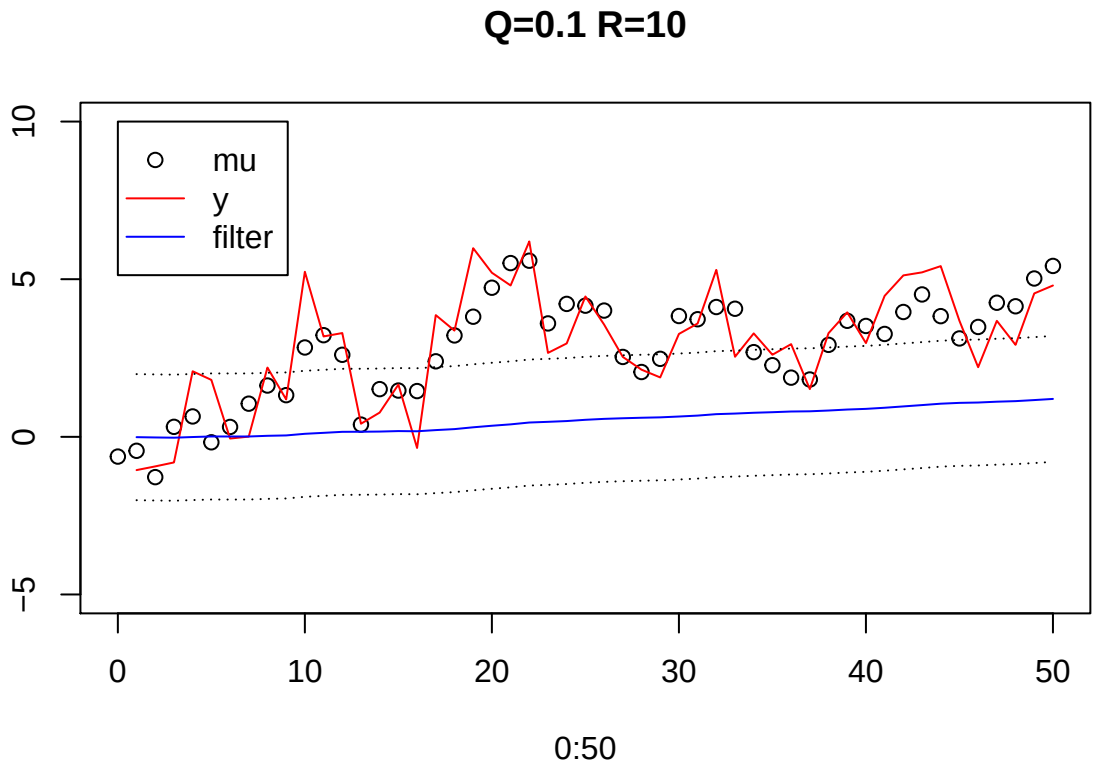
c)



When R is 10 times smaller and Q is 10 times larger in the filtering, the new result (blue line) seems to overlay the observations (red line). The reason might be that Q means the noise between the nearby states, as it affects the change of states evidently. Thus, all the changes among states are acceptable, so do the pattern of observations.

To be more specific, variable R means the noise between state and observed point at each time. Therefore, the confident interval will narrow when R is smaller.

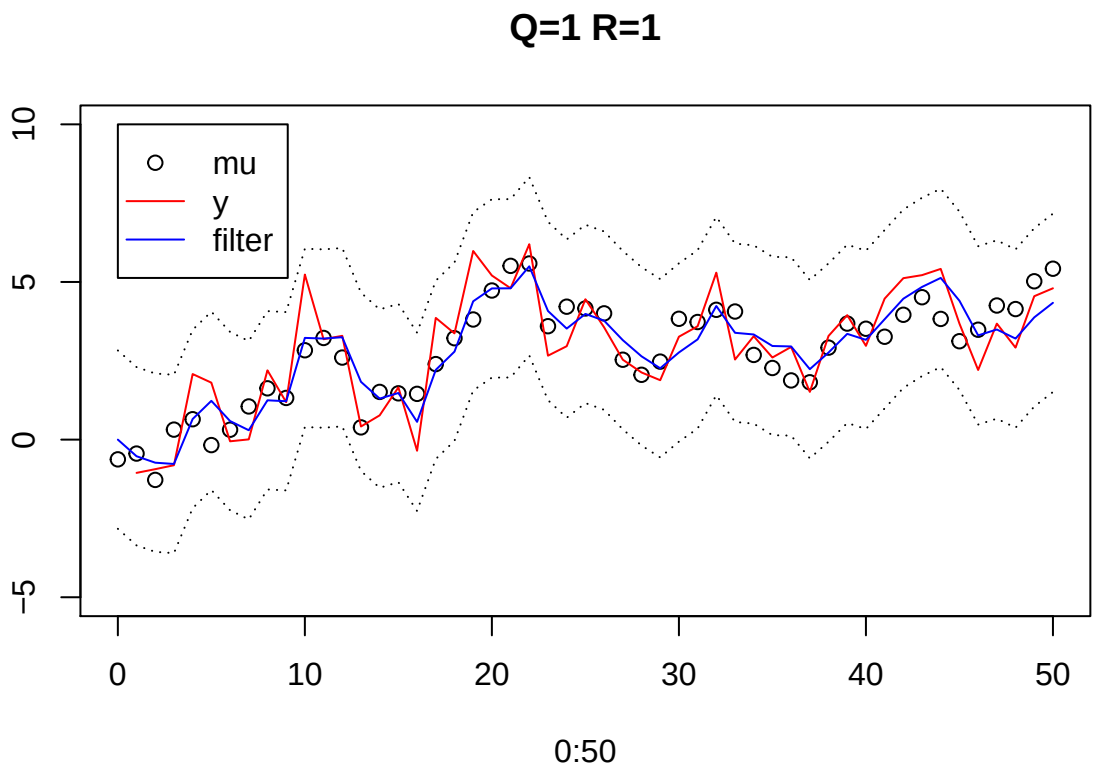
d)



The filtered path is too stable here. It tends to have no difference between states and observations when Q is smaller, since the change is too tiny. On the other hand, large R provides wide confident interval for such filter.

e)

```r
# lecture 6 pg.25
# phi: C
# y: observations
kalman_filter <- function(num, y, A, mu0, Sigma0, Phi, cQ, cR){
  K <- Sigma0%*%Phi%*%solve(Phi%*%Sigma0%*%t(Phi) + cR)
  mu_new <- matrix(ncol=1+num)
  mu_new[1] <- mu0
  Sigma_new <- Sigma0
  for (i in 1:num) {
    mu_n <-mu_new[i]+K%*%(y[i]-Phi%*%mu_new[i])        # observation update step
    Sigma_n <- Sigma_new-K%*%Phi%*%Sigma_new
    mu_new[i+1] <- A%*%mu_n                            # prediction step
    Sigma_new <- A%*%Sigma_n%*%t(A)+cQ
  }
  return(list(xf=mu_new, Pf=c(Sigma_new) ))
}
res <- kalman_filter(50, y, A=1, mu0=0, Sigma0=1, Phi=1, cQ=1, cR=1)
```



f)

How do you interpret the Kalman gain K?

The Kalman gain K is the optimal solution for posterior-state estimation via least squares method, since we define the linear relation between observation y and state mu. Such solution can provide us the minimal mean-squared error (MMSE) of linear regression model, and affect the update of estimated states.

Appendix

```
knitr::opts_chunk$set(echo = FALSE)
library(forecast)
library(astsa)
rm(list=ls())
# generate data
set.seed(1)
num = 50
w = rnorm(num+1,0,1)
v = rnorm(num ,0,1)
mu = cumsum (w) # hidden state : mu[0], mu[1] ,... , mu[50]
y = mu[-1] + v # obs: y[1] ,... , y[50]
plot(0:50, mu, type="p", ylim=c(-5,10), ylab="")
lines(1:50, y, type="l", col="red")
legend(0,10,legend=c("mu","y"),
       col=c("black","red"), pch=1, lty=1)
# filter and smooth ( Ksmooth 0 does both )
ks = Ksmooth0(num, y, A=1, mu0=0, Sigma0=1, Phi=1, cQ=1, cR=1)

plot(0:50, mu,ylim =c(-5,10), ylab="", main="Q=1 R=1")
lines(1:50, y, type="l", col="red")
lines(ks$xf, col="blue")
lines(1:50, ma(y, 5), col="green")
# lines(1:50,as.numeric(filter(y, rep(1/5,5),sides=2 )), col="Yellow")
lines(ks$xf+2* sqrt (ks$Pf), lty =3, col=1)
lines(ks$xf -2* sqrt (ks$Pf), lty =3, col=1)
legend(0,10,legend=c("mu","y","filter","ma5"),
       col=c("black","red","blue","green"), pch=c(1,NA,NA,NA), lty=c(NA,1,1,1))
ks = Ksmooth0(num , y, A=1, mu0=0, Sigma0=1, Phi=1, cQ=10, cR=0.1)

plot(0:50, mu,ylim =c(-5,10), ylab="", main="Q=10 R=0.1")
lines(1:50, y, type="l", col="red")
lines(ks$xf, col="blue")
lines(ks$xf+2* sqrt (ks$Pf), lty =3, col=1)
lines(ks$xf -2* sqrt (ks$Pf), lty =3, col=1)
legend(0,10,legend=c("mu","y","filter"),
       col=c("black","red","blue"), pch=c(1,NA,NA), lty=c(NA,1,1))
ks = Ksmooth0(num , y, A=1, mu0=0, Sigma0=1, Phi=1, cQ=0.1, cR=10)

plot(0:50, mu,ylim =c(-5,10), ylab="", main="Q=0.1 R=10")
lines(1:50, y, type="l", col="red")
lines(ks$xf, col="blue")
lines(ks$xf+2* sqrt (ks$Pf), lty =3, col=1)
lines(ks$xf -2* sqrt (ks$Pf), lty =3, col=1)
legend(0,10,legend=c("mu","y","filter"),
       col=c("black","red","blue"), pch=c(1,NA,NA), lty=c(NA,1,1))
# lecture 6 pg.25
# phi: C
# y: observations
kalman_filter <- function(num, y, A, mu0, Sigma0, Phi, cQ, cR){
  K <- Sigma0%*%Phi%*%solve(Phi%*%Sigma0%*%t(Phi) + cR)
  mu_new <- matrix(ncol=1+num)
```

```
  mu_new[1] <- mu0
  Sigma_new <- Sigma0
  for (i in 1:num) {
    mu_n <-mu_new[i]+K%*%(y[i]-Phi%*%mu_new[i])        # observation update step
    Sigma_n <- Sigma_new-K%*%Phi%*%Sigma_new
    mu_new[i+1] <- A%*%mu_n                            # prediction step
    Sigma_new <- A%*%Sigma_n%*%t(A)+cQ
  }
  return(list(xf=mu_new, Pf=c(Sigma_new) ))
}
res <- kalman_filter(50, y, A=1, mu0=0, Sigma0=1, Phi=1, cQ=1, cR=1)

plot(0:50, mu,ylim =c(-5,10), ylab="", main="Q=1 R=1")
lines(1:50, y, type="l", col="red")
lines(0:50, res$xf, col="blue")
lines(0:50, res$xf+2*sqrt(res$Pf), lty =3, col=1)
lines(0:50, res$xf-2*sqrt(res$Pf), lty =3, col=1)
legend(0,10,legend=c("mu","y","filter"),
       col=c("black","red","blue"), pch=c(1,NA,NA), lty=c(NA,1,1))
```