LABC_Group08

Thijs Quast(thiqu264), Saewon Jun(saeju204)

2019 10 2

```
# generate data
library(astsa)
```

```
## Warning: package 'astsa' was built under R version 3.5.2
```

```
set.seed(1)
num = 50

w = rnorm( num+1, 0, 1)
v = rnorm( num ,0,1)

mu = cumsum( w ) # state : mu[0] , mu[1] ,... , mu[50]
y = mu[ -1] + v # obs: y[1] ,... , y[50]
# filter and smooth ( Ksmooth 0 does both )

ks = Ksmooth0( num , y , A =1, mu0=0, Sigma0=1, Phi =1, cQ =1, cR =1)
```

a.Write down the expression for the state space model that is being simulated.

$A_{t-1} = 1$ and $C_t = 1$.
Value of mu from our code is equivalent to z_t and y from our code is equivalent to x_t. So we can rewrite our state space model as following :

$$mu_t = mu_{t-1} + w_t$$

$$y_t = mu_t + v_t$$

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

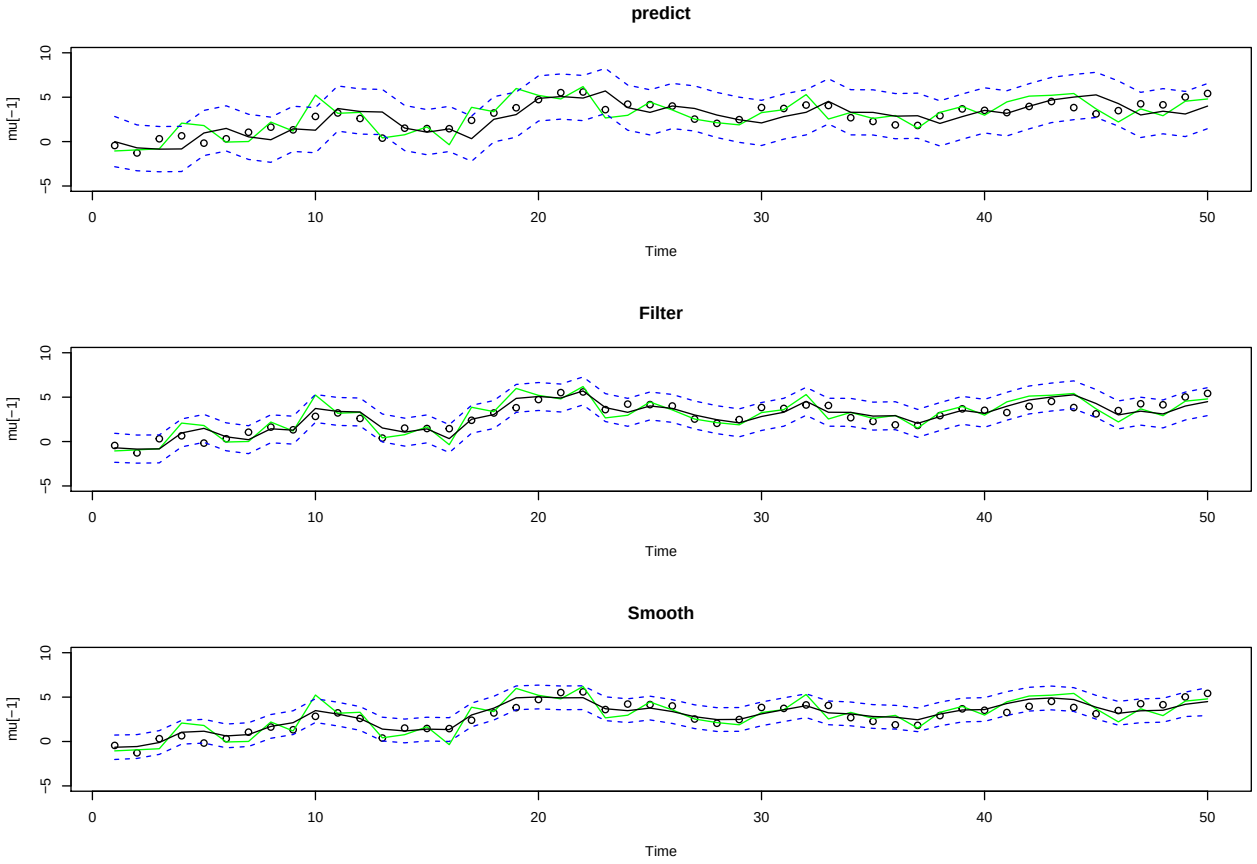
$$v_t \sim N(0, 1)$$

b.Run the script and compare the filtering results with a moving average smoother of order 5.

Run the script

```
plot_ks <- function(ks, num){
        Time = 1:num
        plot( Time , mu[ -1] , main ='predict', ylim =c(-5,10))
        lines( Time ,y ,col=" green ")
        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)

        plot( Time , mu[ -1] , main ='Filter', ylim =c( -5,10))
```

```
        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)

        plot( Time , mu[ -1] , main ='Smooth', ylim =c( -5,10))
        lines( Time ,y ,col=" green ")
        lines( ks$xs )
        lines( ks$xs +2* sqrt ( ks$Ps ) , lty =2, col=4)
        lines( ks$xs -2* sqrt ( ks$Ps ) , lty =2, col=4)
        mu[1]; ks$x0n ; sqrt ( ks$P0n ) # initial value info
}

par(mfrow=c(3,1))
plot_ks(ks=ks, num=50)
```

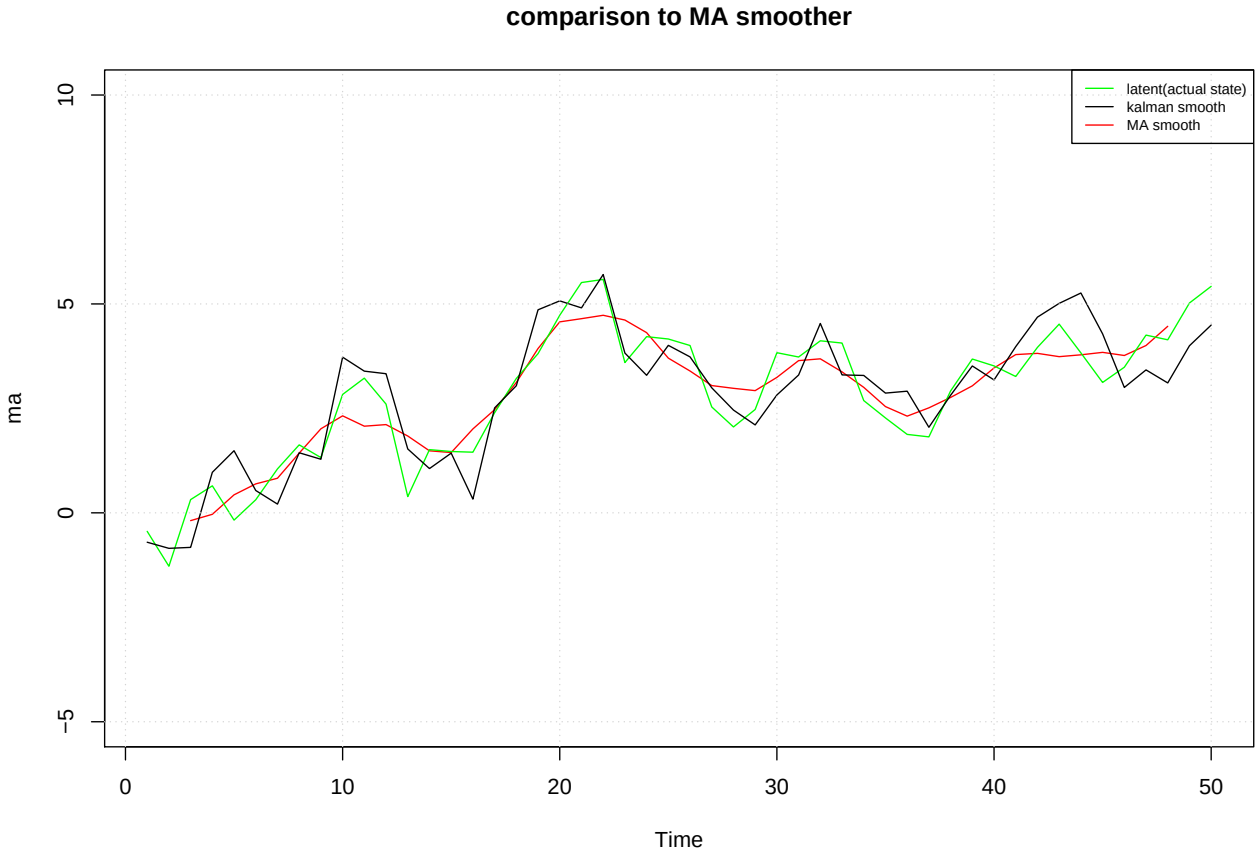



```
##           [,1]
## [1,] 0.7861514
```

Comparison to moving average smoother

```
#mu is latent
#y is obs
Time = 1:50
ma <- filter(mu[-1], filter=c(0.2,0.2,0.2,0.2,0.2), method="convolution")
```

```
plot( Time , ma , main ='comparison to MA smoother', ylim =c(-5,10), type="l", col="red")
lines( Time ,mu[-1] ,col=" green ") # latent -actual state
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)
grid()
legend("topright", legend=c("latent(actual state)", "kalman smooth", "MA smooth"), lty=c(1,1,1),
       col=c("green", "black", "red"), cex=0.7)
```

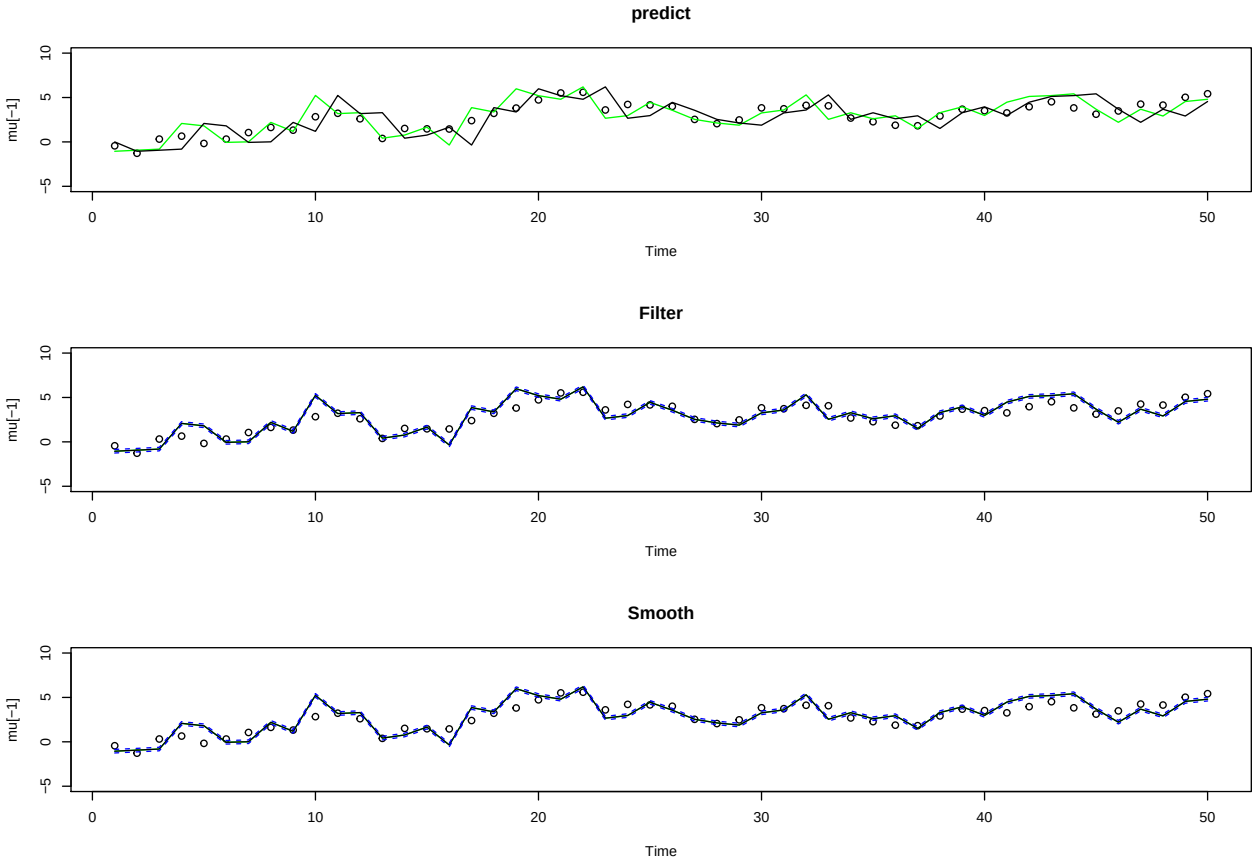


Kalman smooth (black line) gives you the result which is closer to actual state(green line).

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?

```
ks2 = Ksmooth0( num , y , A =1, mu0=0, Sigma0=1, Phi =1, cQ =10, cR =0.1)

par(mfrow=c(3,1))
plot_ks(ks=ks2, num=50)
```



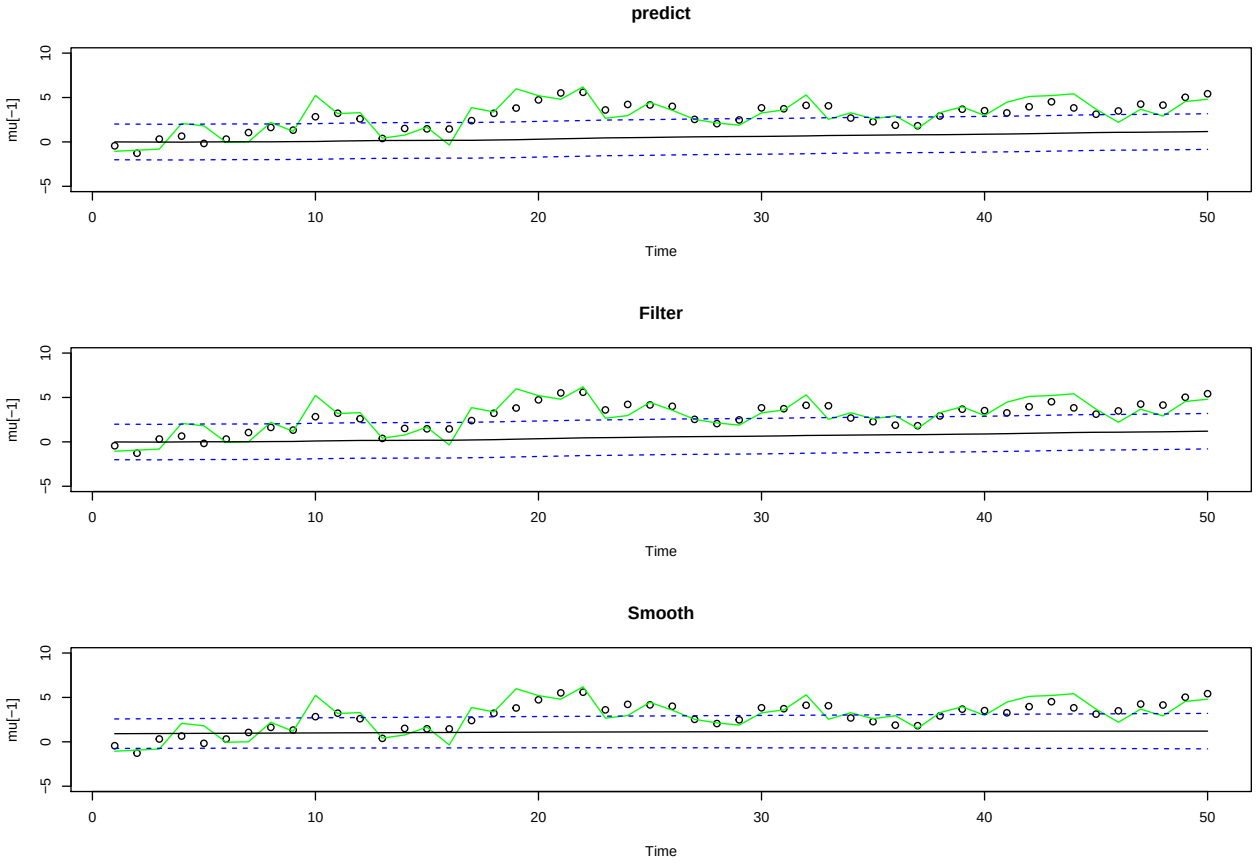

```
##             [,1]
## [1,] 0.9950377
```

Applying kalman filter we would like to figure out the actual state of Z. By controling the value of cR and cQ, you can decide whether you want to focus on observations or you want to focus on state evolution. If you have smaller value for cR compare to cQ, it means that it will focus more on observations.

Here we have smaller value for cR, which means it will focus more on observations. That's probably the reason why we got the smaller Pt and narrower confidence interval.

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?

```
ks3 = Ksmooth0( num , y , A =1, mu0=0, Sigma0=1, Phi =1, cQ =0.1, cR =10)

par(mfrow=c(3,1))
plot_ks(ks=ks3, num=50)
```




```
##          [,1]
## [1,] 0.82731
```

When you have bigger value for cR which means it will **less** focus on the observations, so you can see that the obtained values does not vary much.

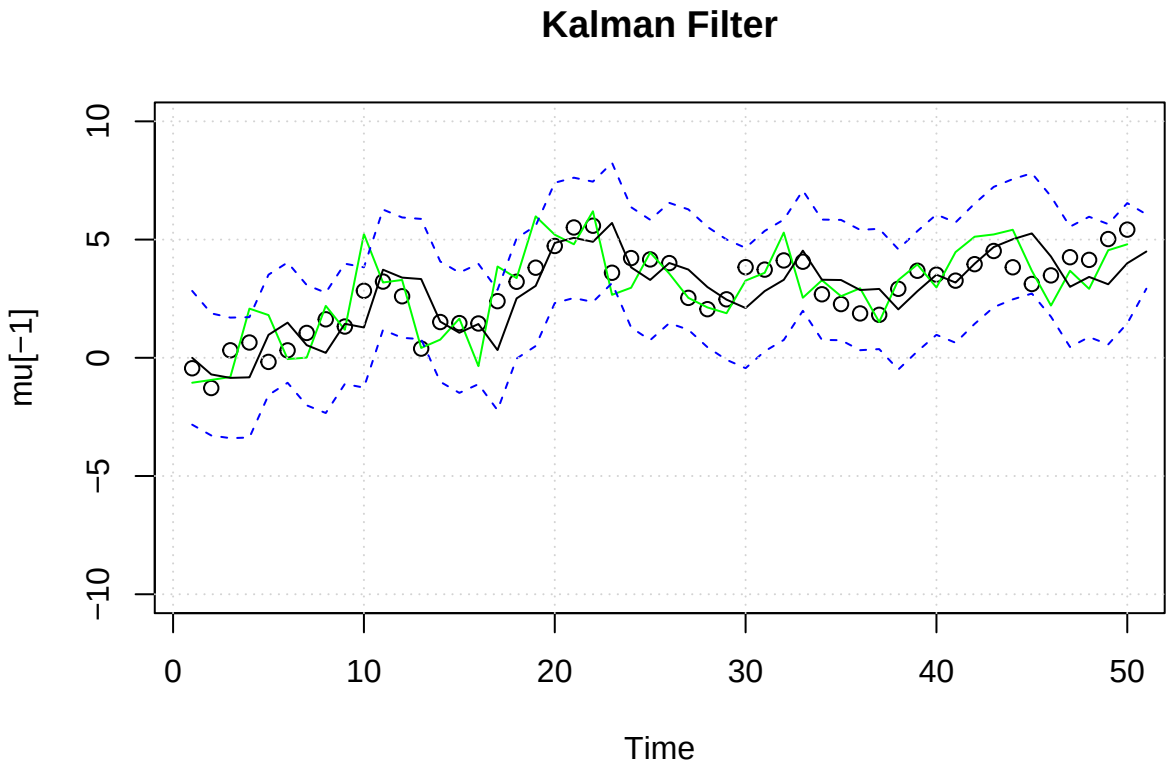
e.Implement your own Kalman filter and replace ksmooth() function with your script.

```
Kalman_filter <- function(At, Ct, Q, R, m0, P0, x, Time){

        #sine we have information of cQ and cR from previous step, we create
        #create covariance matrix from it.
        Qt <- t(Q)%*%Q
        Rt <- t(R)%*%R

        # initialization
        m <- c()
        m[1] <- m0

        Pt <- c()
        Pt[1] <- P0


        for (i in 1:Time){
```

```
                # prediction step
                m[i] <- At%*%m[i]
                Pt[i] <- At%*%Pt[i]%*%t(At) + Qt

                # observation update step
                Kt <- Pt[i] %*% t(Ct) %*% solve(Ct%*%Pt[i]%*%t(Ct) + Rt)
                m[i+1] <- m[i] + Kt %*% (x[i]-Ct%*%m[i])
                Pt[i+1] <- Pt[i] - Kt%*%Ct%*%Pt[i]


        }

        output <- list("Pt" = Pt, "m" = m)

        return(output)

}
```

```
set.seed(1)

num = 50

w = rnorm( num+1, 0, 1)
v = rnorm( num ,0,1)

mu = cumsum( w ) # state : mu[0] , mu[1] ,... , mu[50]
y = mu[-1] + v # obs: y[1] ,... , y[50]
# filter and smooth ( Ksmooth 0 does both )

my_ks <- Kalman_filter(At = 1, Ct = 1, Q = 1, R = 1, m0 = 0, P0 = 1, x = y, Time = 50)
```

```
plot( Time , mu[-1] , main ='Kalman Filter', ylim =c(-10,10))
lines(Time, y ,col=" green ")
lines(my_ks$m)
lines(my_ks$m + 2*sqrt(my_ks$Pt) , lty =2, col=4)
lines(my_ks$m - 2*sqrt(my_ks$Pt) , lty =2, col=4)
grid()
```



f.How do you interpret the Kalman gain?

By implementing the Kalman gain, the algorithm tries to get closer to the actual value by updating prediction by using its difference between actual value and obtained value.