Time Series Analysis - lab03

Lennart Schilling (lensc874), Sridhar Adhikarla (sriad858)

2019-10-12

Contents

```r
library(astsa)
library(forecast)
library(ggplot2)
```

Assignment 1

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

The state space model being simulated is -

$$
\begin{aligned}
z_t &= (A_{t-1} * z_{t-1}) + e_t \quad - Transition\ equation \\
x_t &= (C_t * z_t) + v_t \quad - Observation\ equation \\
v_t &= Normal(0, R_t) \\
e_t &= Normal(0, Q_t)
\end{aligned}
$$

The parameter values for this state space model are -

$$
\begin{aligned}
A &= 1 \\
C &= 1 \\
R &= 1 \\
Q &= 1
\end{aligned}
$$

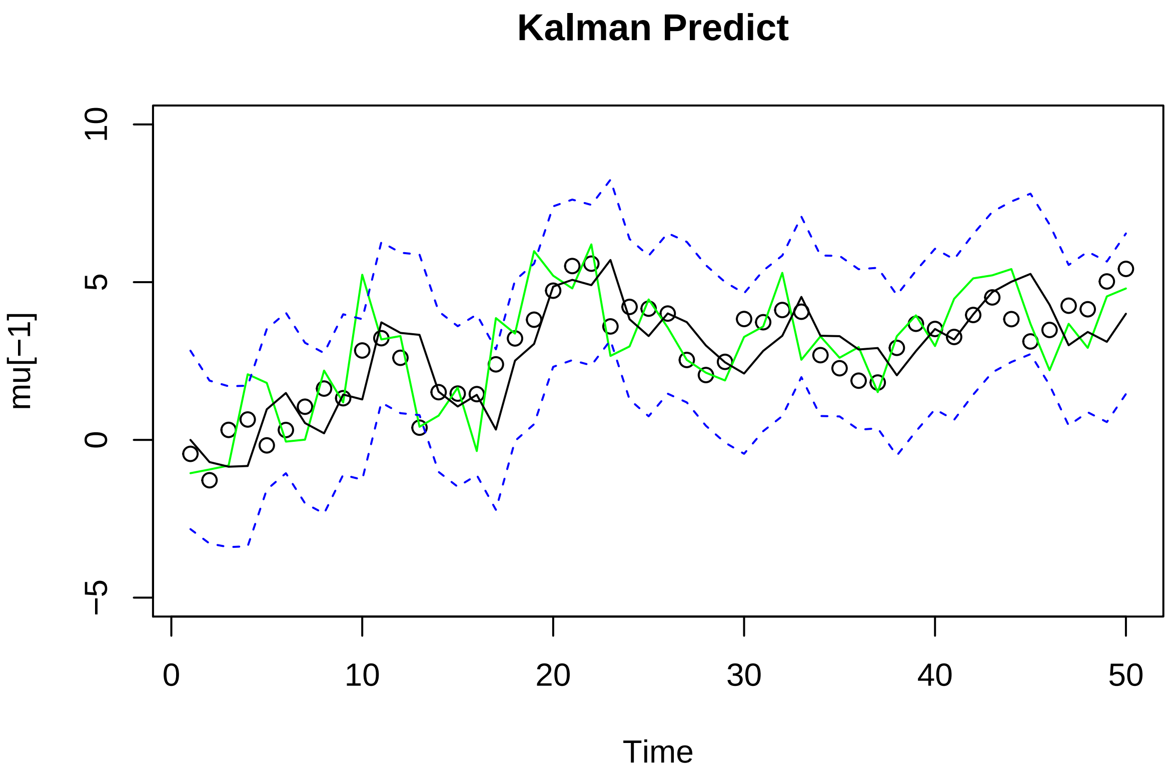
The state space model with these parameter values is -

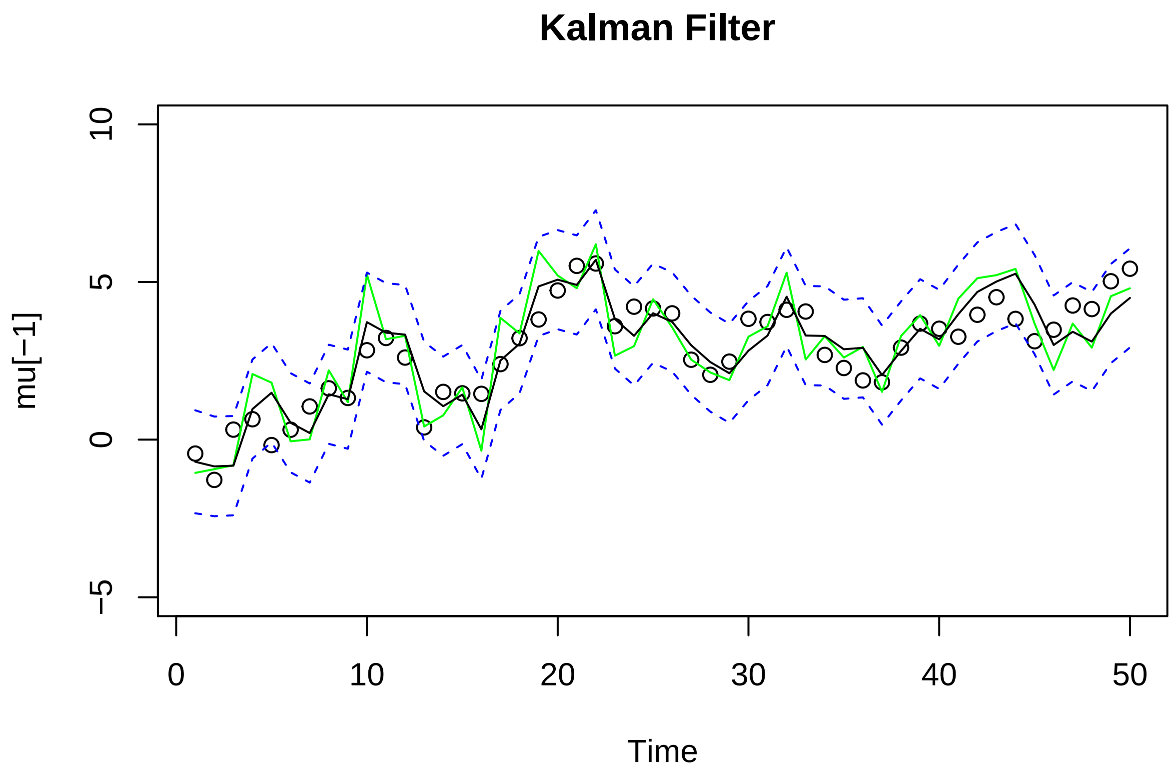
$$
\begin{aligned}
z_t &= z_{t-1} + e_t \quad - Transition\ equation \\
x_t &= z_t + v_t \quad - Observation\ equation \\
v_t &= Normal(0, 1) \\
e_t &= Normal(0, 1)
\end{aligned}
$$

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

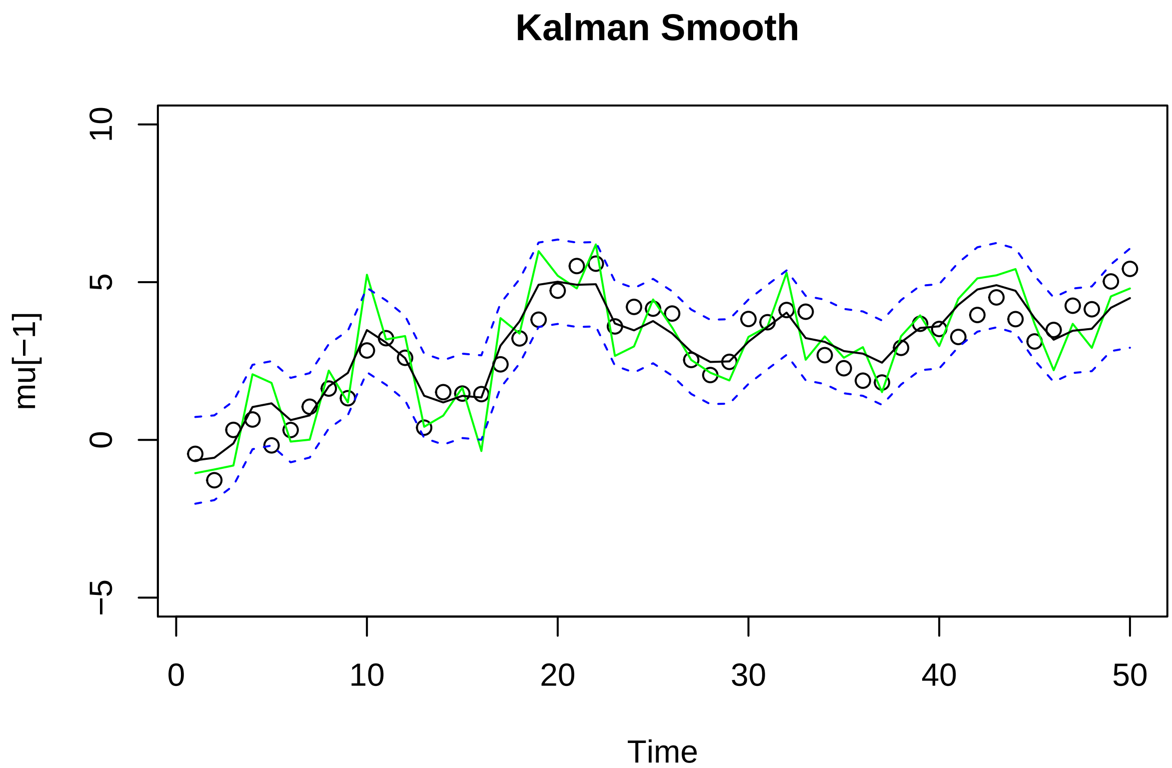
Results from running the kalman filter

```
#utiliy functions

#used to generate data simulated from the SSM
gen_Data = function(num=50, Q=1, R=1){
  set.seed(1)
  w = rnorm(num+1,0,R)
  v = rnorm(num ,0,Q)
  mu = cumsum(w) # state : mu[0], mu[1] ,... , mu[50]
  y = mu[-1] + v # obs: y[1] ,... , y[50]
  return(list(Y = y, MU = mu))
}

#fits a kalman filter on
fit_Kalman = function(y, Q=1, R=1){
  # filter and smooth ( Ksmooth 0 does both )
  ks = Ksmooth0(length(y) , y, A=1, mu0=0, Sigma0=1, Phi=1, cQ=Q, cR=R)
  return(ks)
}

gen_Plot = function(ks, mu, y){
  num = length(y)
  Time = 1:num
  plot (Time , mu[-1], main ='Kalman 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$Pp), lty =2, col=4)
  plot (Time , mu[-1], main ='Kalman Filter ', ylim =c(-5,10))
  lines (Time ,y,col=" green ")
  lines (ks$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 ='Kalman 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)
}

dat = gen_Data()
ks = fit_Kalman(dat$Y)
gen_Plot(ks, dat$MU, dat$Y)
```


Kalman Predict
10
5
0
−5
mu[−1]
0
10
20
30
40
50
Time



Kalman Filter
mu[−1]
Time
10
5
0
−5
0
10
20
30
40
50



Results on running a moving average smoother on the data

```
ma5_smooth = ma(dat$Y, order = 5)
num = length(dat$Y)

ggplot() +
  geom_line(aes(x=1:num, y=dat$MU[-1], col="True State")) +
  geom_line(aes(x=1:num, y=dat$Y, col="Observations")) +
  geom_line(aes(x=1:num, y=ma5_smooth, col="Smoothed Prediction")) +
  ggtitle("MA(5) smoother") + xlab("Time") + ylab("Value") +
  theme(plot.title = element_text(hjust = 0.5))
```

As we can see from the plots the Kalman Filter fit to the data was better than the one with MA(5) smoother. If followed the true data closely where as the MA(5) model, the predictions are not as close.

```r
t1 = 4:49
print("MA(5) smoother SSE")
```

```
## [1] "MA(5) smoother SSE"
```

```r
sum((dat$MU[t1] - ma5_smooth[t1-1])^2)
```

```
## [1] 18.75434
```

```r
print("Kalman smoother SSE")
```

```
## [1] "Kalman smoother SSE"
```

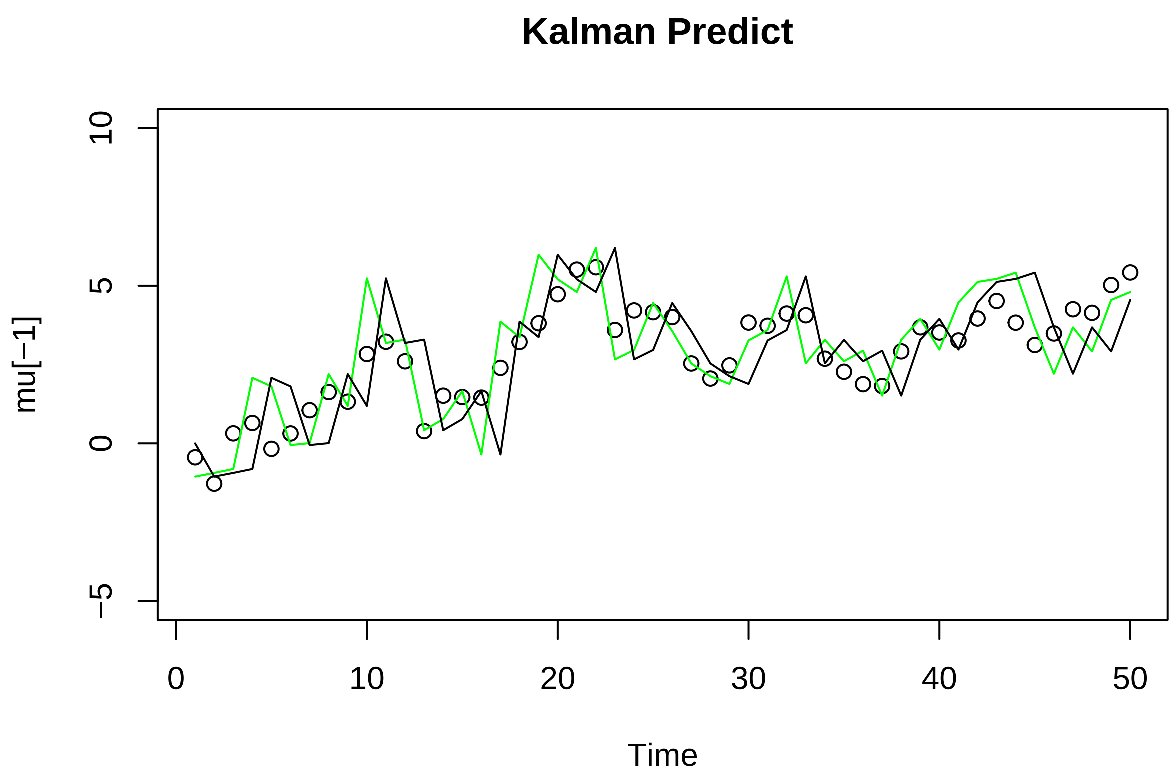
```r
sum((dat$MU[t1] - ks$xs[t1-1])^2)
```

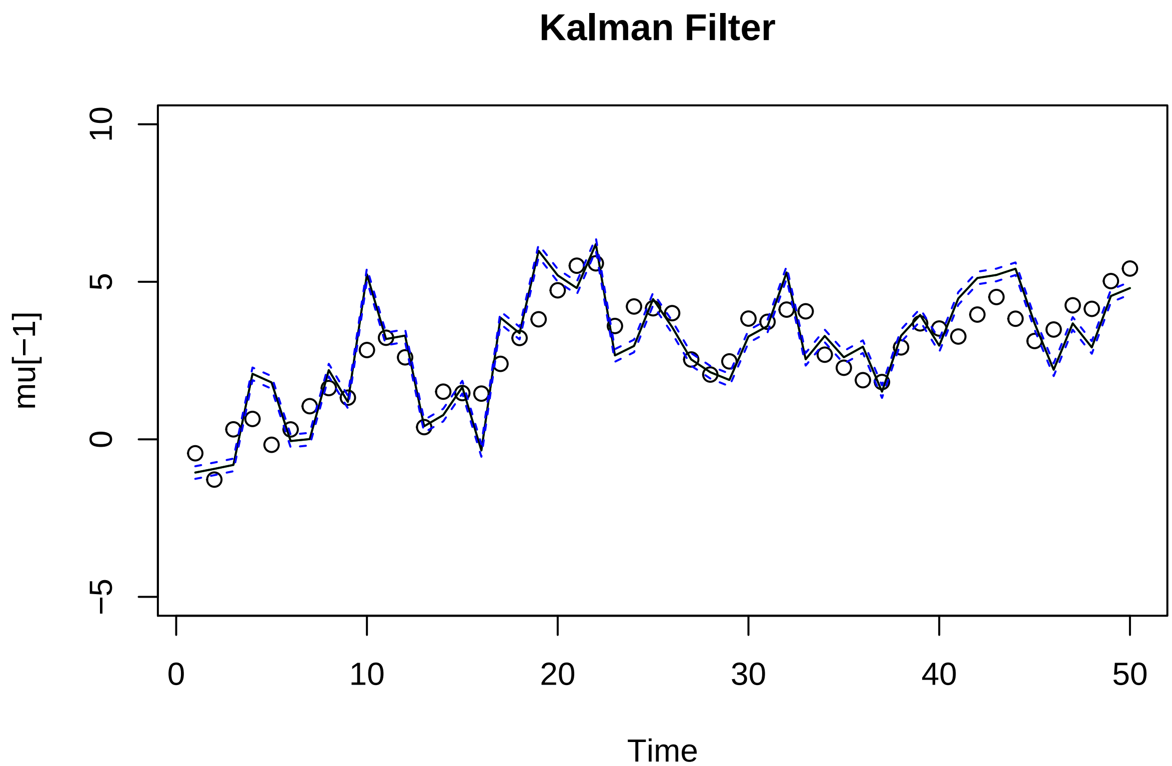
```
## [1] 15.81689
```

The SSE values for the two smoothers show clearly that the kalman smoother is doing a better job, though the difference is not much.

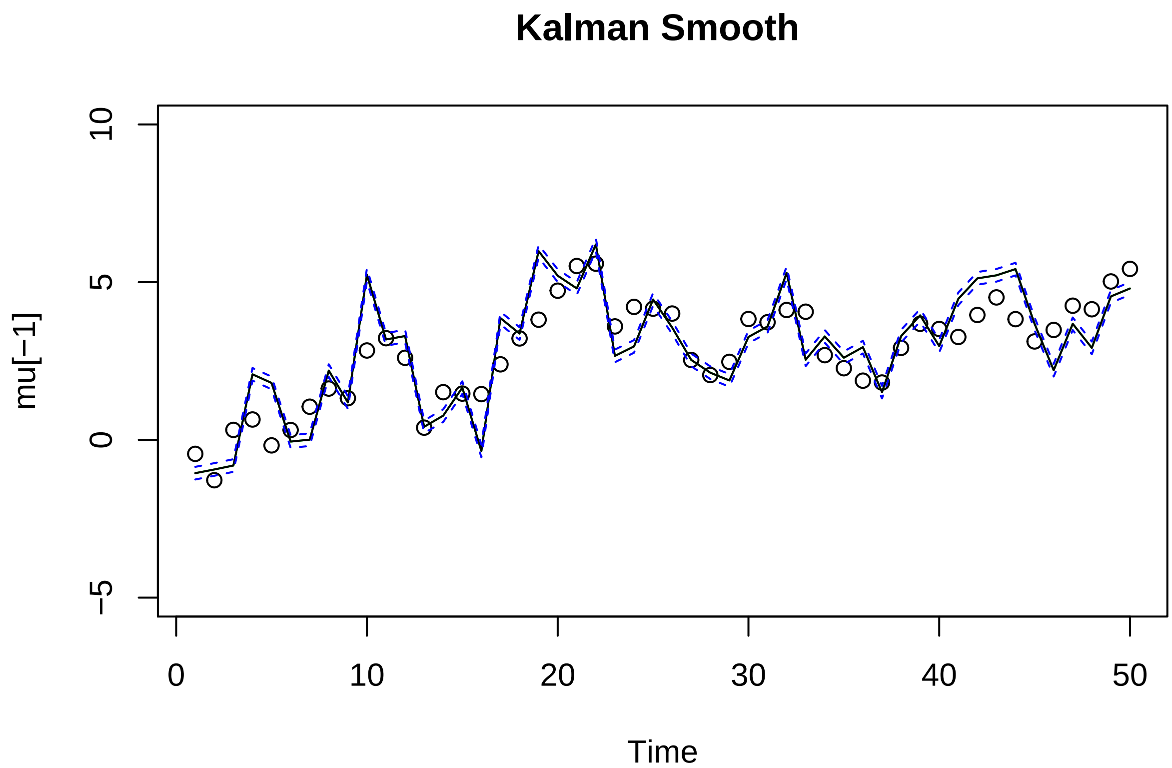
3 Also, compare the filtering outcome when R in the filter is 10 times smaller than its actual value and while Q in the filter is 10 times larger than its actual value. How does the filtering outcome varies?

```
ks3 = fit_Kalman(dat$Y, Q = 10, R = 1/10)
gen_Plot(ks3, dat$MU, dat$Y)
```





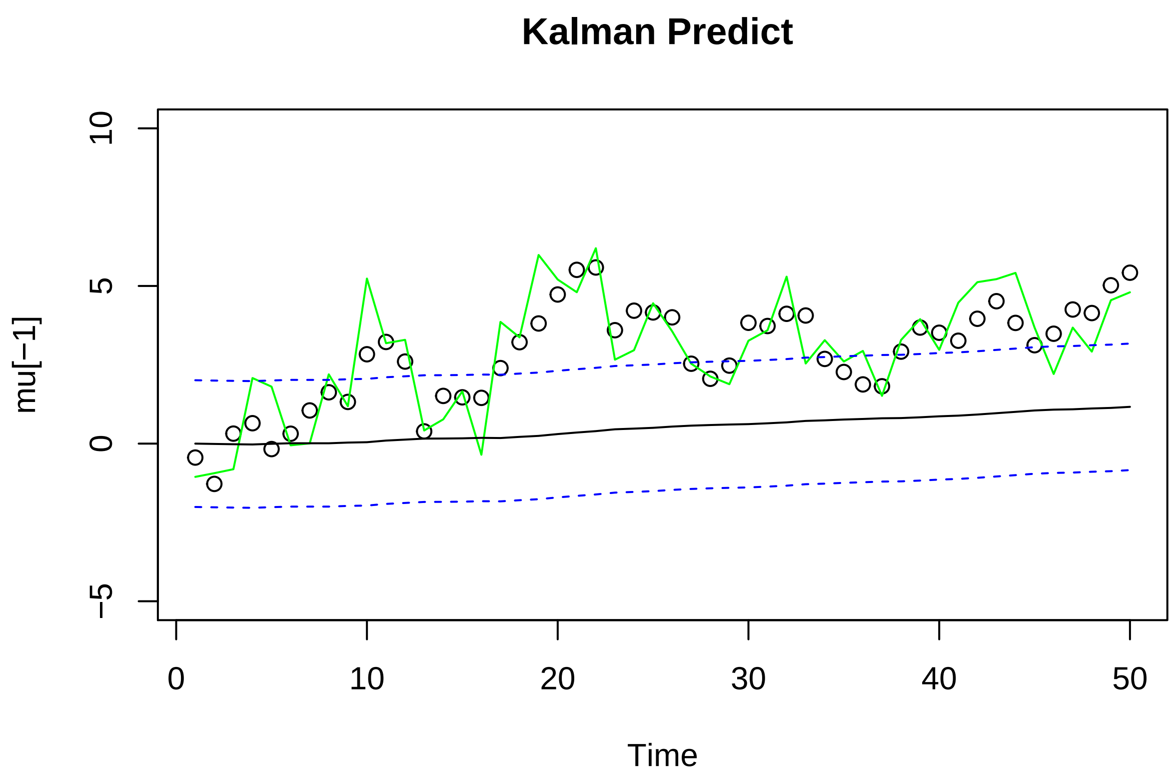
Kalman Filter
10
5
0
−5
mu[−1]
0
10
20
30
40
50
Time

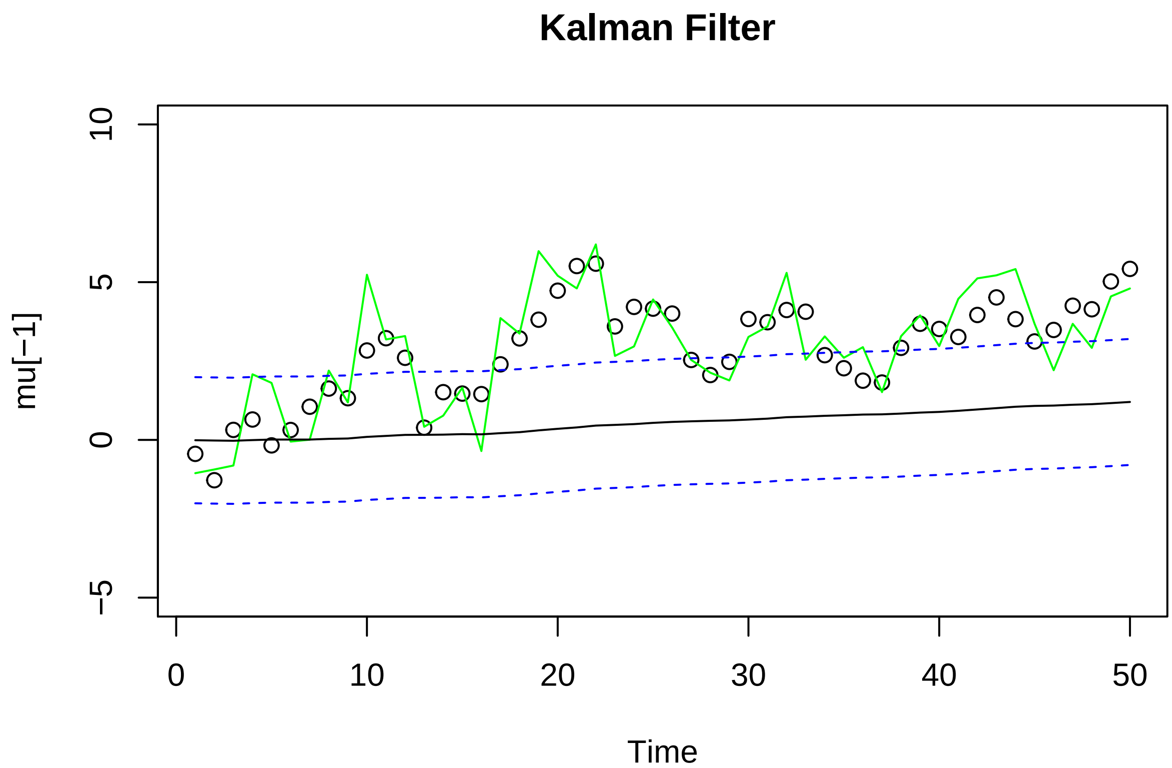


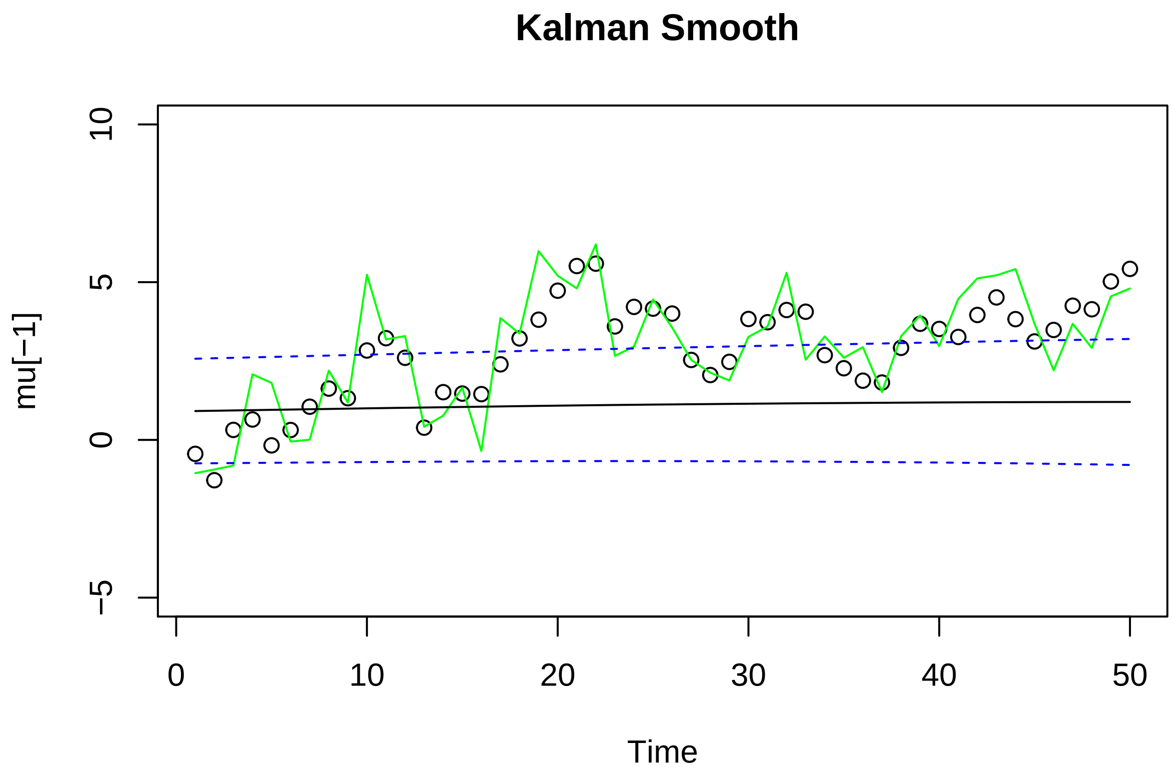
The parameter R is for the varience of the error in measurements. Making it 10 times smaller means that the measurements are very accurate and the estimates should be taken care of. This is why the confidence bands for the prediction are so close to the prediction. For the kalman smooth and filter fit the line overlaps the observation completely. Since it assumes that the measurements are accurate it just uses that to make the get the next point. Also having a large value for Q means that the prediction relies more on the observations than the previous estimate. So, it follows the observations closely.

4 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?

```
ks4 = fit_Kalman(dat$Y, Q = 1/10, R = 10)
gen_Plot(ks4, dat$MU, dat$Y)
```



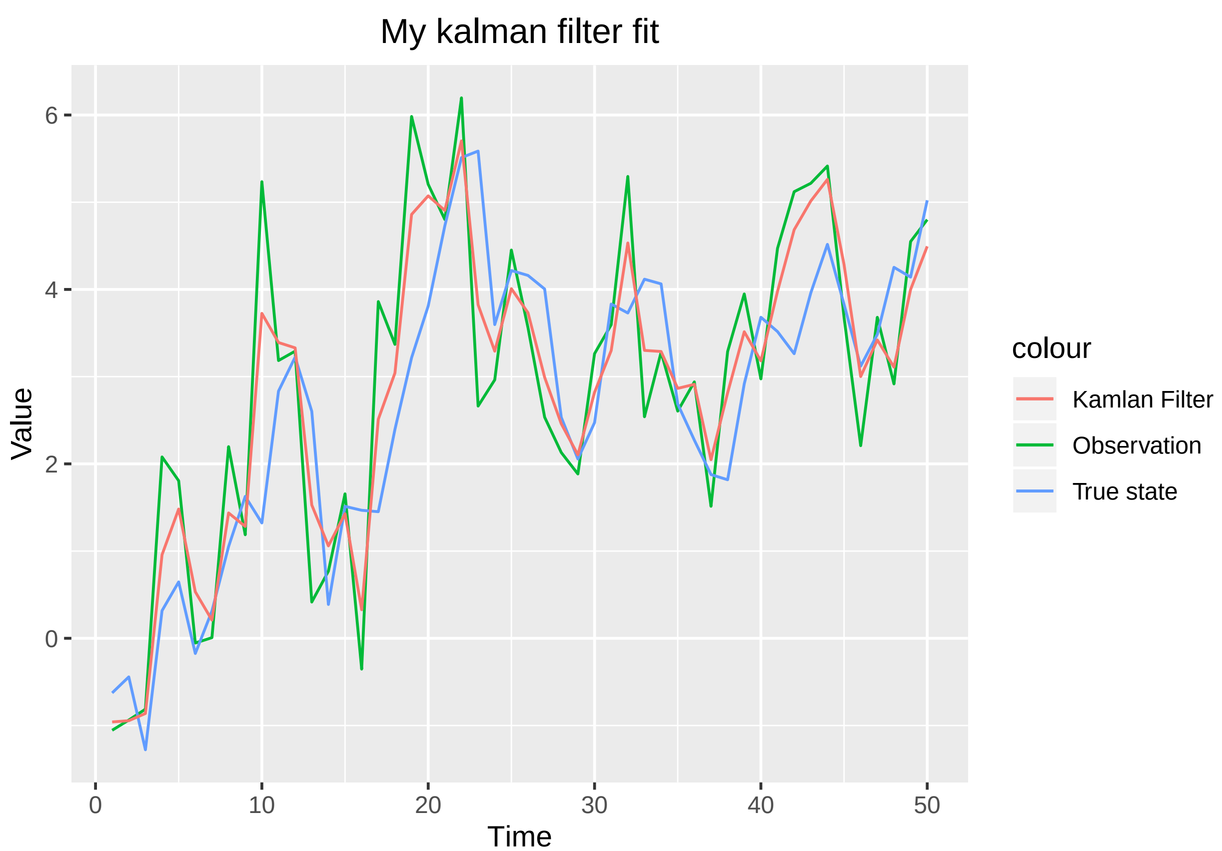

Kalman Filter
10
5
0
−5
mu[−1]
0
10
20
30
40
50
Time



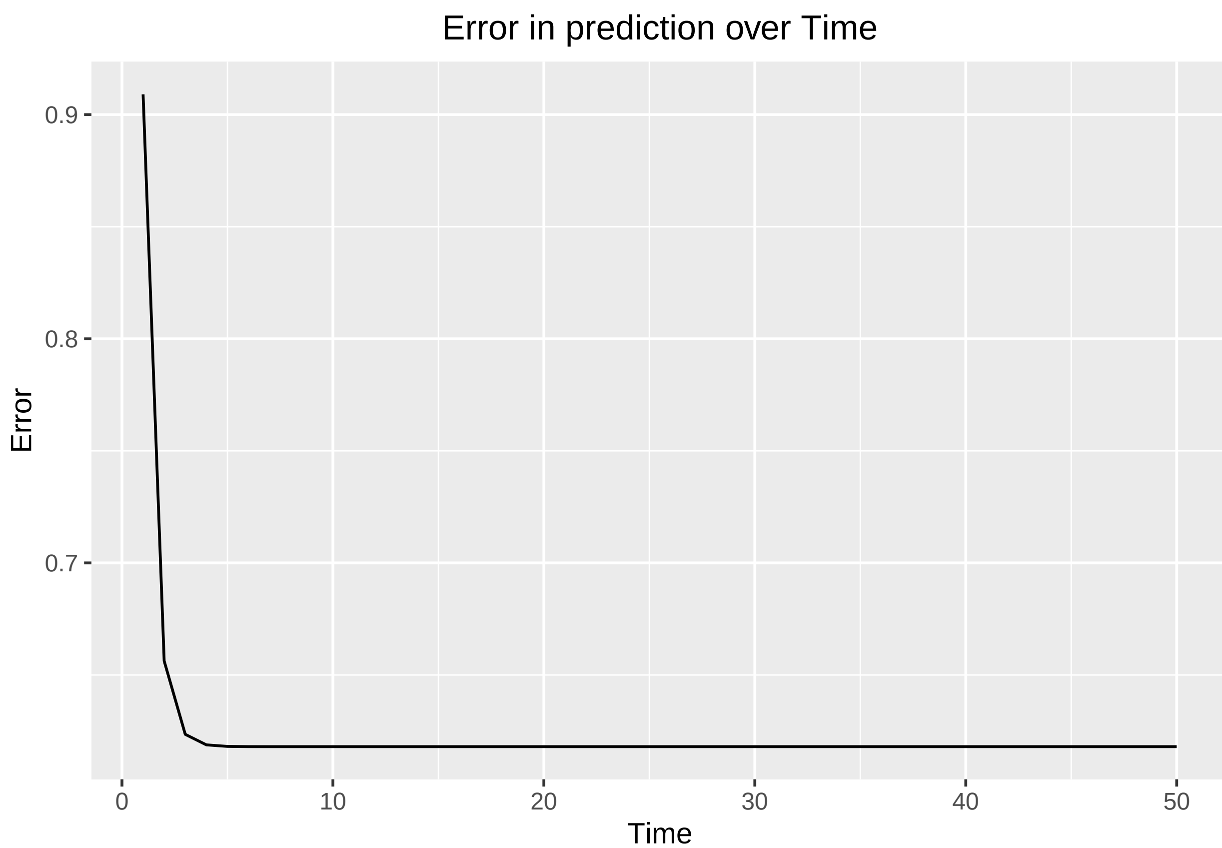
Having a large value for R means that the observations are not reliable and it just goes along with the prediction it had in the previous step. This is the reason we get almost straight line as a fit to the data. It does not deviate much from its initial prediction and since it assumes that the predictions are accurate the confidence interval bands are much further away from the prediction. Having a small value for Q supports this behaivous as it does not rely much on the observations not and instead relies on the previous estimate.

5 Implement your own Kalman filter and replace ksmooth0 function with your script.

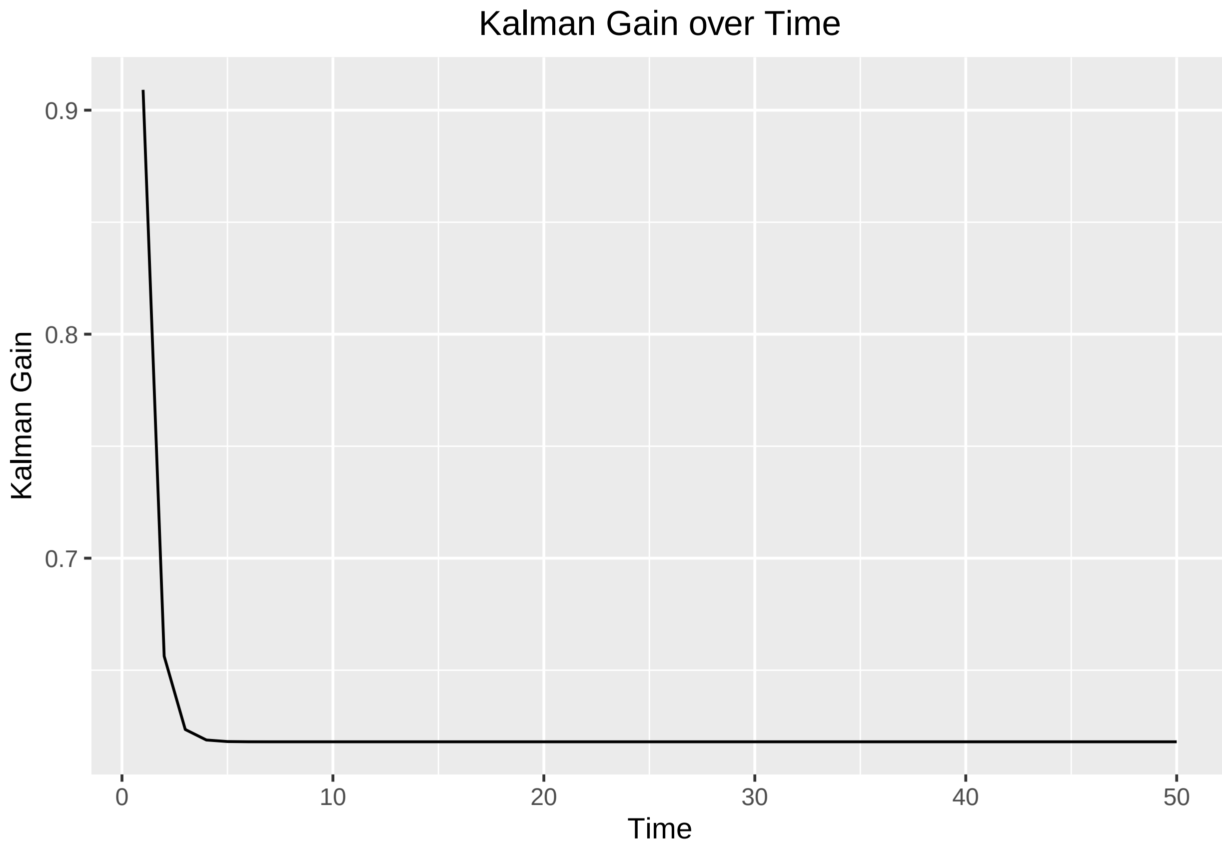
```
my_kalman = function( y, mea_0, err_p0, A=1, C=1, Q=1, R=1){
  n = length(y); kal_gain = c(); mea_t = mea_0; err_pt = err_p0

  for (t in 1:n) {
    kal_gain[t] = err_pt[t]/(err_pt[t] + R)
    mea_t[t] = mea_t[t] + kal_gain[t] * (y[t] - mea_t[t]*C)
    err_pt[t] = (1 - kal_gain[t]*C) * err_pt[t]
    mea_t[t+1] = A*mea_t[t]
    err_pt[t+1] = A*err_pt[t] + Q
  }
  return(list(klf = mea_t[1:n], err_pred = err_pt[1:n], kg = kal_gain[1:n]))
}


my_kal = my_kalman(dat$Y, 0, 10)

ggplot()+
  geom_line(aes(x=1:50, y=dat$Y, col="Observation")) +
  geom_line(aes(x=1:50, y=dat$MU[1:50], col="True state")) +
  geom_line(aes(x=1:50, y=my_kal$klf, col="Kamlan Filter")) +
  ggtitle("My kalman filter fit") + xlab("Time") + ylab("Value") +
  theme(plot.title = element_text(hjust = 0.5))
```



```r
ggplot() +
  geom_line(aes(x=1:50, my_kal$err_pred)) +
  ggtitle("Error in prediction over Time") + xlab("Time") + ylab("Error") +
  theme(plot.title = element_text(hjust = 0.5))
```




```r
ggplot() +
  geom_line(aes(x=1:50, my_kal$kg)) +
  ggtitle("Kalman Gain over Time") + xlab("Time") + ylab("Kalman Gain") +
  theme(plot.title = element_text(hjust = 0.5))
```



As we can see from the plot the kalman gain starts off with a large value close to 1 and it gets rid of the error in prediction in very few steps as the kalman gain gets to a stable value of approximately 0.5 in about 5 steps. This shows the fast convergence of the kalman filter.

6 How do you interpret the Kalman gain?

Kalman gain can take up a value between 0 and 1. It controls how much should our next prediction be affected by the observation. A large value of kalman gain implies that the measurements are accurate and the estimates are unstable. This leads to a large deviation in the prediction from the previous step. A small value of kalman gain implies that the measurements are inacurate and the estimates are stable with small errors. So it uses up most of the previous estimate to make the next prediction without much contribution from the observation. The Kalman gain is the relative weight given to the measurements and current state estimate, and can be "tuned" to achieve a particular performance. With a high gain, the filter places more weight on the most recent measurements, and thus follows them more responsively.