Time Series (732A62) Lab1 Group 7

Anubhav Dikshit(anudi287) and Maximilian Pfundstein(maxpf364)

03 October, 2019

Contents

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.

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

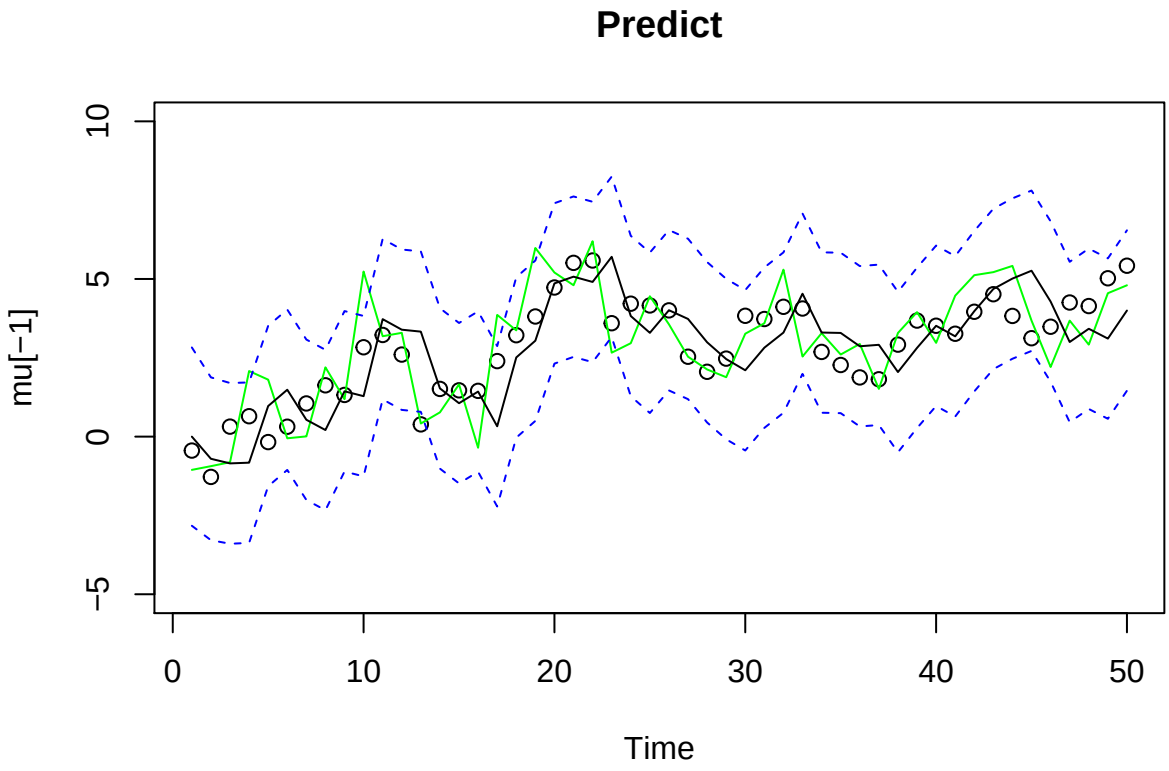
$$\nu_t \sim N(0, R_t)$$
$$e_t \sim N(0, Q_t)$$

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

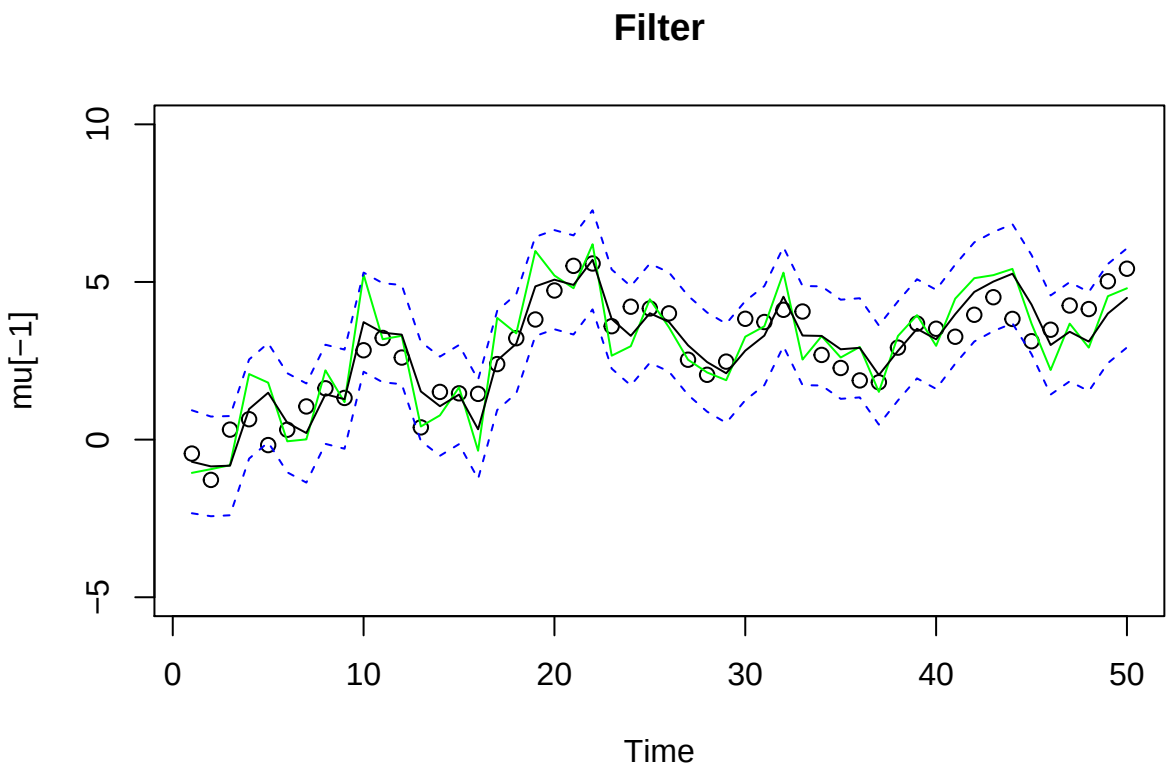
$$z_k = z_{k-1} + N(0, Q_t)$$
$$x_k = z_k + N(0, R_t)$$

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

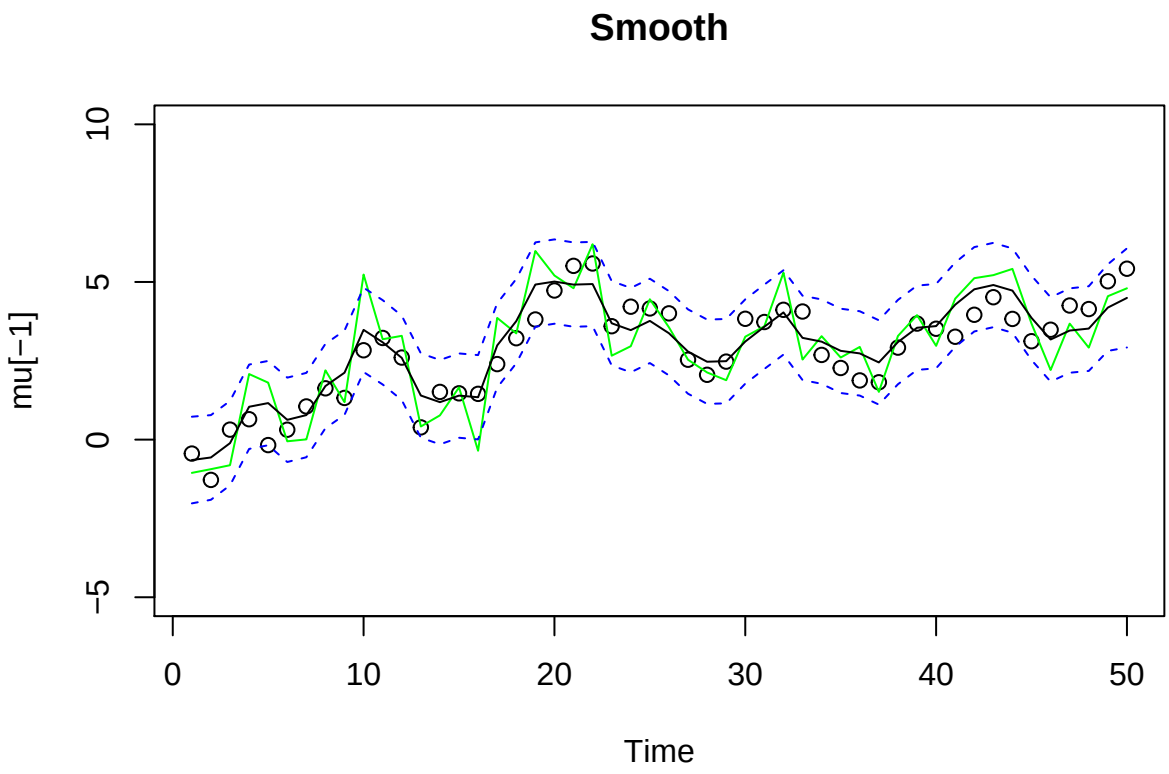
```
# generate  dataset
set.seed(1)
num = 50
w = rnorm(num+1,0,1)
v = rnorm(num ,0,1)
mu = cumsum(w) # state: mu[0], mu[1],..., mu[50]
y = mu[-1] + v # obs: y[1],..., y[50]
# filter  and  smooth (Ksmooth0 does  both)
ks = Ksmooth0(num , y, A=1, mu0=0, Sigma0=1, Phi=1, cQ=1, cR=1)
# start  figurepar(mfrow=c(3,1))
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]
```

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

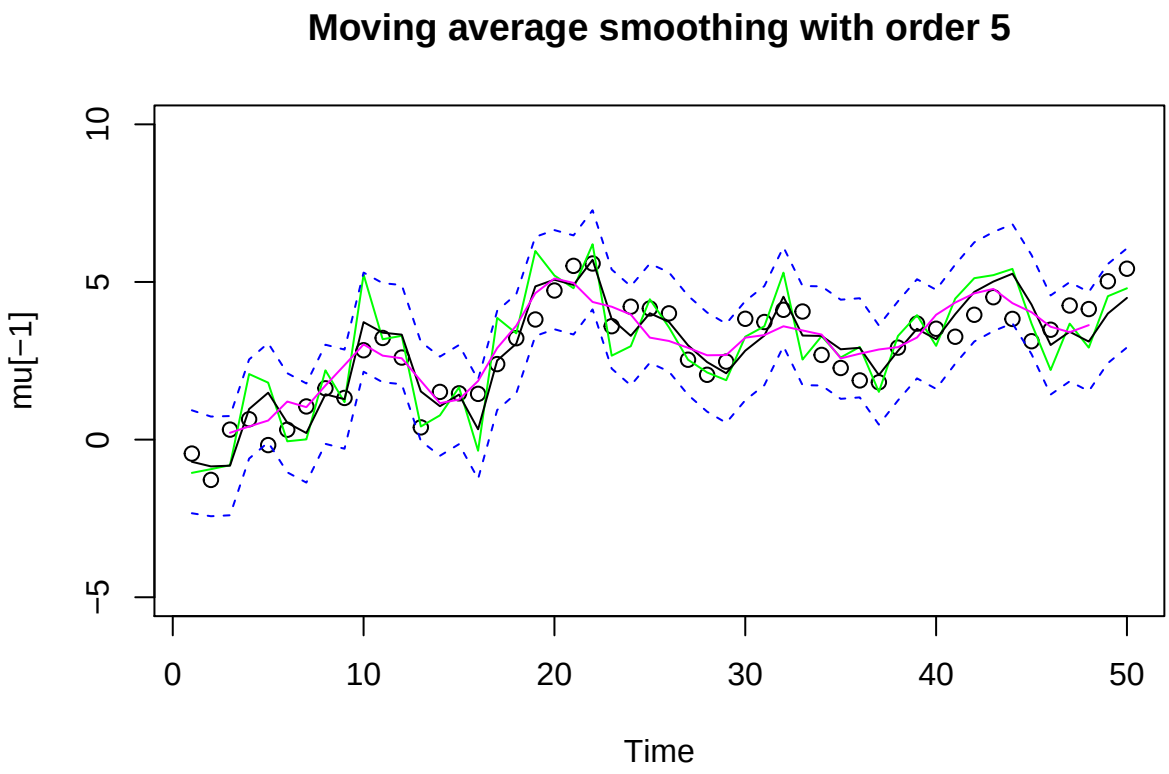
```
ks$x0n
```

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

```
sqrt(ks$P0n) # initial  value  info
```

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

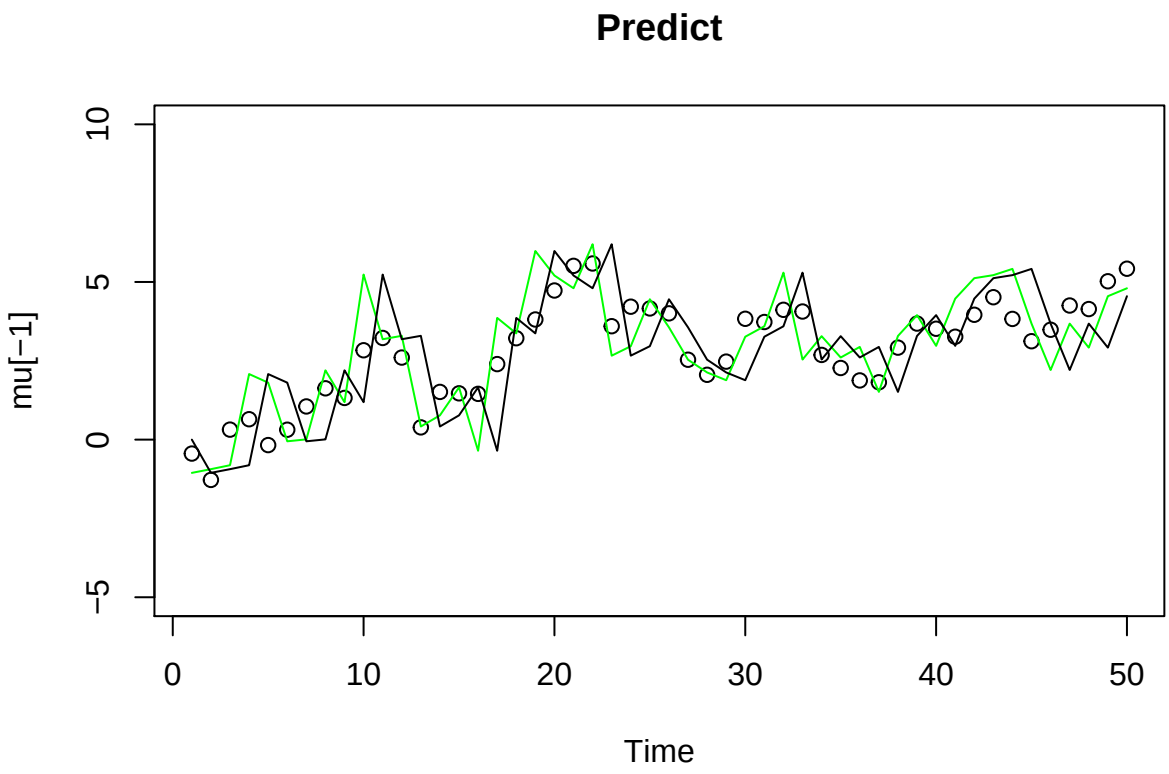
```
# filering with moving average 5
plot(Time , mu[-1], ylim=c(-5,10), main="Moving average smoothing with order 5")
lines(Time ,y,col="green")
lines(ks$xf)
lines(ks$xf+2*sqrt(ks$Pf), lty=2, col=4)
lines(ks$xf -2*sqrt(ks$Pf), lty=2, col=4)
lines(ma(y, order=5, ), col=6)
```



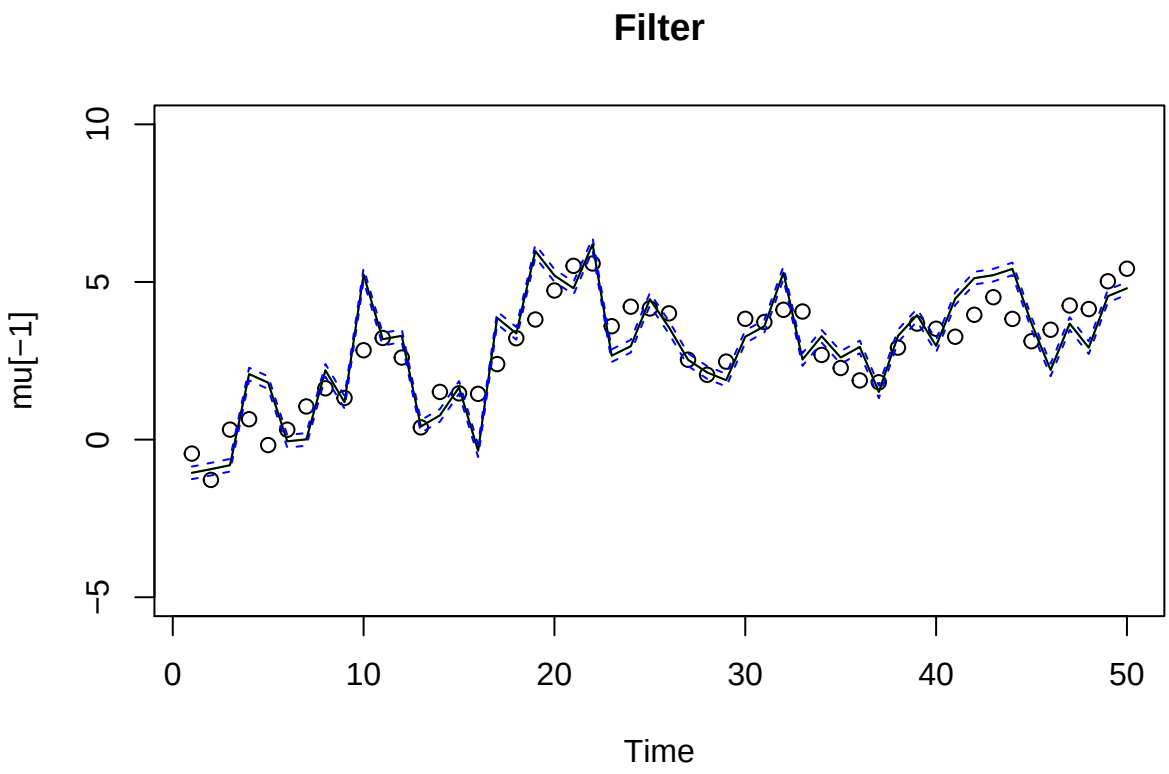
Analysis: We find that the moving average smoothing function with order 5 is the worst fit since its losing all the variability that is captured by our kalman filter.

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?

```
# filter  and  smooth (Ksmooth0 does  both)
ks = Ksmooth0(num , y, A=1, mu0=0, Sigma0=1, Phi=1, cQ=10, cR=0.1)
# start  figurepar(mfrow=c(3,1))
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)
```



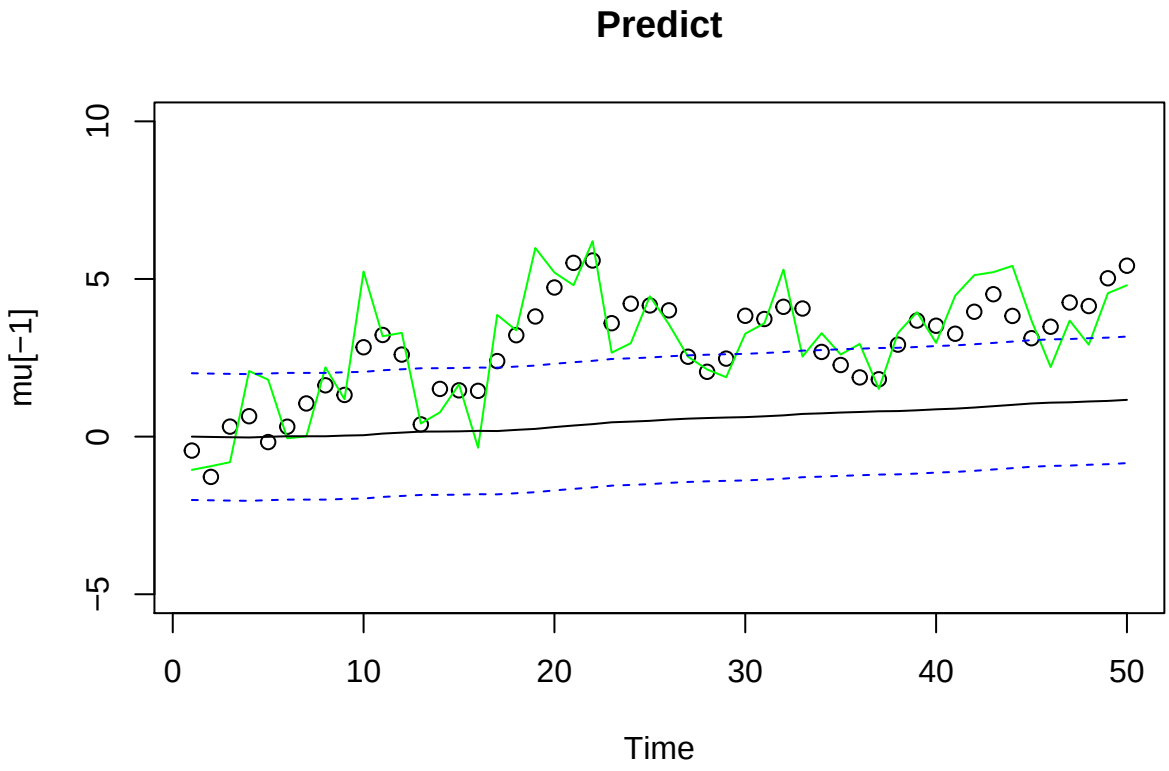

```
plot(Time , mu[-1], main="Filter", ylim=c(-5,10))
lines(Time ,y,col="green")
lines(ks$xf)
lines(ks$xf+2*sqrt(ks$Pf), lty=2, col=4)
lines(ks$xf -2*sqrt(ks$Pf), lty=2, col=4)
```



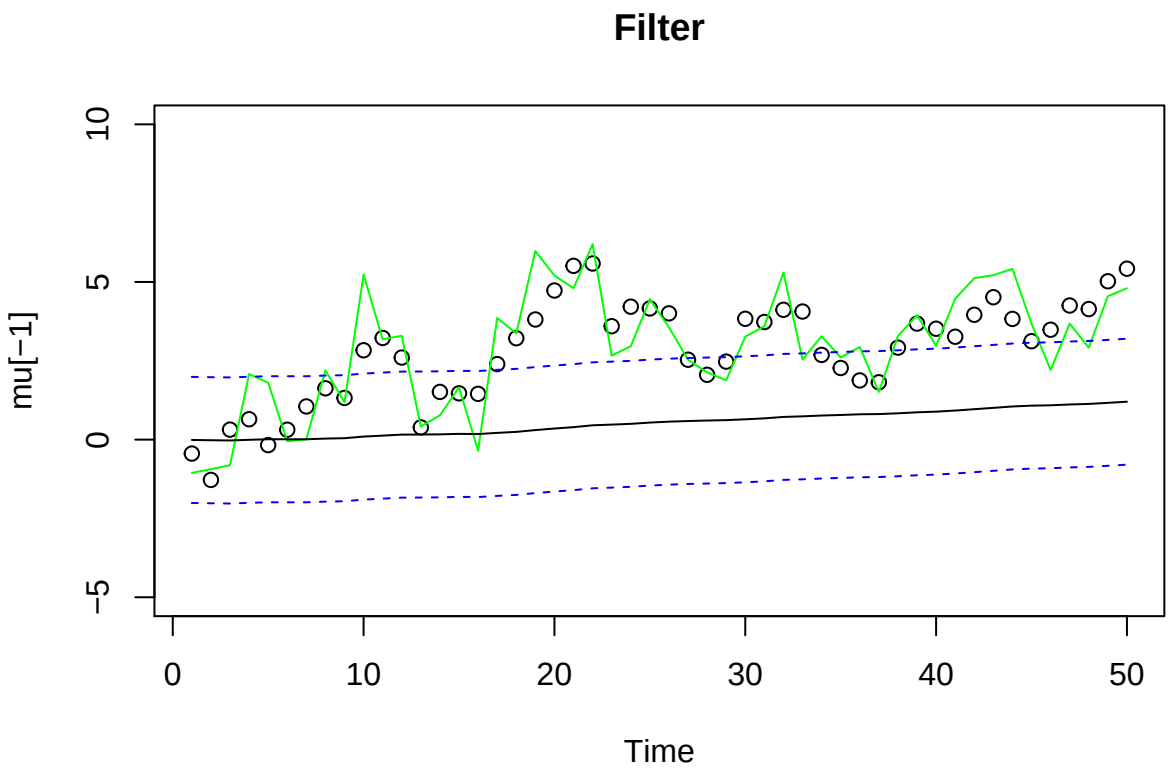
Analysis: We find that the filtering output resembles the true value much more than the previousily run values.

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?

```
# filter  and  smooth (Ksmooth0 does  both)
ks = Ksmooth0(num , y, A=1, mu0=0, Sigma0=1, Phi=1, cQ=0.1, cR=10)
# start  figurepar(mfrow=c(3,1))
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)
```

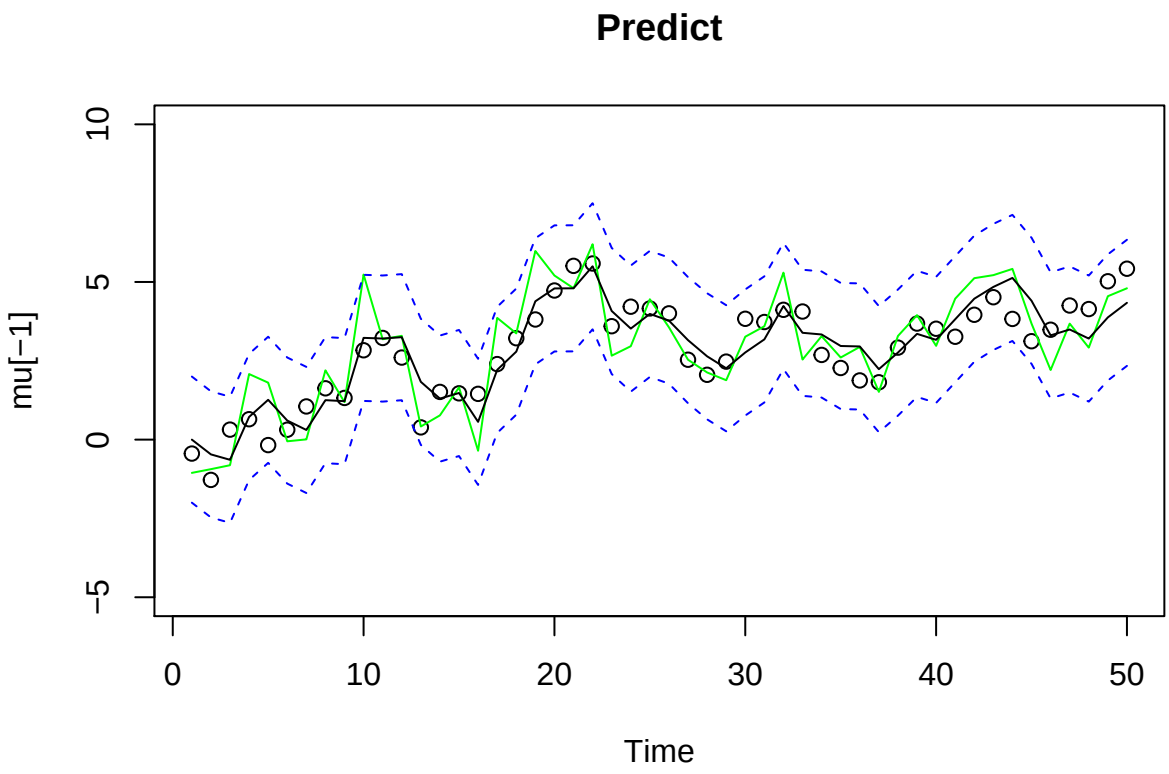


Analysis: We find that the filtering output is far off the true value and hardly moving in terms of the predicted values, this is because kalman gain assumes that the uncertainity in measurement is high and it largely sticks with the mean of the series.

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

```
#Times: Number of time steps
#A: How the mean of z_t will be affected by z_(t-1) Used in transition model
#C: Scale the mean (z_t) in the emission model
#Q: Covariate matrix in the emission model
#R: Covariate matrix in the transition model
#y: All observations
#mu0: first mean
#sigma0: first varience
my_kalman <- function(Times, y, A, mu0, Sigma0, C, Q, R) {
  mu <- rep(1,Times)
  sigma <- rep(1,Times)
  my_unweighted <- rep(1,Times)
  sigma_unweighted <- rep(1,Times)
  kalman_gain <- rep(1,Times)

  # Our best guess is that my_1 is mu0 else it could be the first observation
  mu[1] <- mu0

  # We don't know what sigma_1 is, so we choose 1 else it would be provided
  sigma[1] <- Sigma0
```

```
  for (t in 2:Times) {
  # Calculate the unweighted prediction of the mean
  my_unweighted[t] <- A[t]%*%mu[t - 1]

  # Calculate the unweighted prediction of the covariate matrix
  sigma_unweighted[t] <- A[t]%*%sigma[t - 1]%*%t(A[t]) + Q[t]

  # Kalman gain Used to weight between our unweighted prediction and the obs
  kalman_gain[t] <- sigma_unweighted[t]%*%t(C[t]) %*% solve(C[t]%*%sigma_unweighted[t]%*%t(C[t] + R[t]))

  # Calculate the weighted mean, thus our prediction of the hidden state
  mu[t] <- my_unweighted[t] + kalman_gain[t]%*%(y[t] - C[t]%*%my_unweighted[t])

    # Calculate the weighted covariance matrix, thus our prediction of the predition error
    sigma[t] <- (1 - kalman_gain[t]%*%C[t])%*%sigma_unweighted[t]
  }

  return (list(mu = mu, sigma = sigma))
}


# generate  dataset
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]



my_ks = my_kalman(Times=num , y=y, A=rep(1,num), mu0=0, Sigma0=1,
                  C=rep(1,num), Q=rep(1,num), R=rep(1,num))



plot(Time , mu[-1], main="Predict", ylim=c(-5,10))
lines(Time ,y,col="green")
lines(my_ks$mu)
lines(my_ks$mu+2*sqrt(my_ks$sigma), lty=2, col=4)
lines(my_ks$mu -2*sqrt(my_ks$sigma), lty=2, col=4)
```



f) 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.

Appendix

```
knitr::opts_chunk$set(echo = TRUE)
options(scipen=999)

library("tidyverse") #ggplot and dplyr
library("gridExtra") # combine plots
library("knitr") # for pdf
library("astsa") #dataset oil and gas is present here
library("matlib") # for inv and I
library("forecast")
```

```r
# The palette with black:
cbbPalette <- c("#000000", "#E69F00", "#56B4E9", "#009E73",
                "#F0E442", "#0072B2", "#D55E00", "#CC79A7")
set.seed(12345)

# generate  dataset
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 (Ksmooth0 does  both)
ks = Ksmooth0(num , y, A=1, mu0=0, Sigma0=1, Phi=1, cQ=1, cR=1)
# start  figurepar(mfrow=c(3,1))
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

# filering with moving average 5
plot(Time , mu[-1], ylim=c(-5,10), main="Moving average smoothing with order 5")
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)


# filter  and  smooth (Ksmooth0 does  both)
ks = Ksmooth0(num , y, A=1, mu0=0, Sigma0=1, Phi=1, cQ=10, cR=0.1)
# start  figurepar(mfrow=c(3,1))
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)
```

```r
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$Pf), lty=2, col=4)
# filter  and  smooth (Ksmooth0 does  both)
ks = Ksmooth0(num , y, A=1, mu0=0, Sigma0=1, Phi=1, cQ=0.1, cR=10)
# start  figurepar(mfrow=c(3,1))
Time = 1:num
plot(Time , mu[-1], main="Predict", ylim=c(-5,10))
lines(Time ,y,col="green")
lines(ks$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)
#Times: Number of time steps
#A: How the mean of z_t will be affected by z_(t-1) Used in transition model
#C: Scale the mean (z_t) in the emission model
#Q: Covariate matrix in the emission model
#R: Covariate matrix in the transition model
#y: All observations
#mu0: first mean
#sigma0: first varience
my_kalman <- function(Times, y, A, mu0, Sigma0, C, Q, R) {
  mu <- rep(1,Times)
  sigma <- rep(1,Times)
  my_unweighted <- rep(1,Times)
  sigma_unweighted <- rep(1,Times)
  kalman_gain <- rep(1,Times)

  # Our best guess is that my_1 is mu0 else it could be the first observation
  mu[1] <- mu0

  # We don't know what sigma_1 is, so we choose 1 else it would be provided
  sigma[1] <- Sigma0

  for (t in 2:Times) {
  # Calculate the unweighted prediction of the mean
  my_unweighted[t] <- A[t]%*%mu[t - 1]

  # Calculate the unweighted prediction of the covariate matrix
  sigma_unweighted[t] <- A[t]%*%sigma[t - 1]%*%t(A[t]) + Q[t]

  # Kalman gain Used to weight between our unweighted prediction and the obs
  kalman_gain[t] <- sigma_unweighted[t]%*%t(C[t]) %*% solve(C[t]%*%sigma_unweighted[t]%*%t(C[t] + R[t]))

  # Calculate the weighted mean, thus our prediction of the hidden state
  mu[t] <- my_unweighted[t] + kalman_gain[t]%*%(y[t] - C[t]%*%my_unweighted[t])
```

```
    # Calculate the weighted covariance matrix, thus our prediction of the predition error
    sigma[t] <- (1 - kalman_gain[t]%*%C[t])%*%sigma_unweighted[t]
  }

  return (list(mu = mu, sigma = sigma))
}


# generate  dataset
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]



my_ks = my_kalman(Times=num , y=y, A=rep(1,num), mu0=0, Sigma0=1,
                  C=rep(1,num), Q=rep(1,num), R=rep(1,num))



plot(Time , mu[-1], main="Predict", ylim=c(-5,10))
lines(Time ,y,col="green")
lines(my_ks$mu)
lines(my_ks$mu+2*sqrt(my_ks$sigma), lty=2, col=4)
lines(my_ks$mu -2*sqrt(my_ks$sigma), lty=2, col=4)
```