Time Series Analysis - Lab 03 (Group 7)

Anubhav Dikshit (anudi287) and Maximilian Pfundstein (maxpf364)

2019-10-03

Contents

1 Assignment 1

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.

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

1.1 a) State Space Expression

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

Answer: The state space model is given as the following, as $A = 1$ and $\Phi = 1$, which are scalar.

$$\mathbf{z}_t = \mathbf{z}_{t-1} + \mathcal{N}(0, Q_t)$$
$$\mathbf{x}_t = \mathbf{z}_t + \mathcal{N}(0, R_t)$$

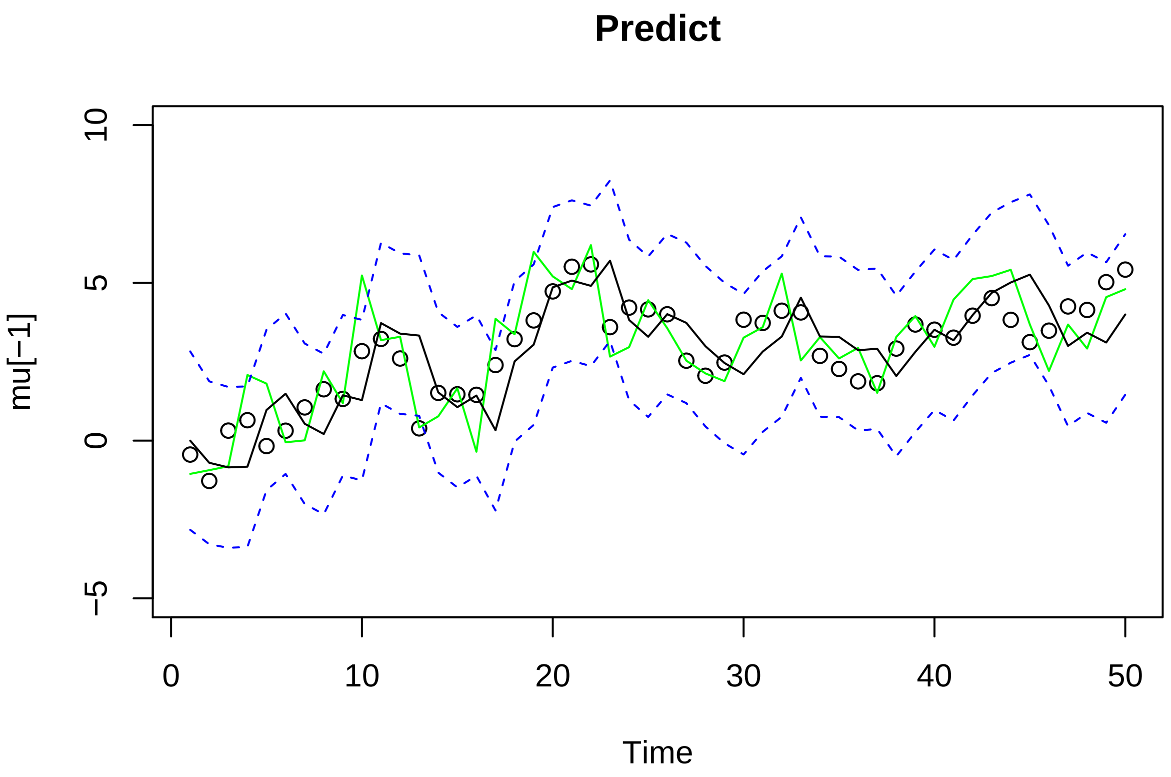
1.2 b) Comparison of Filtering vs Moving Average

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

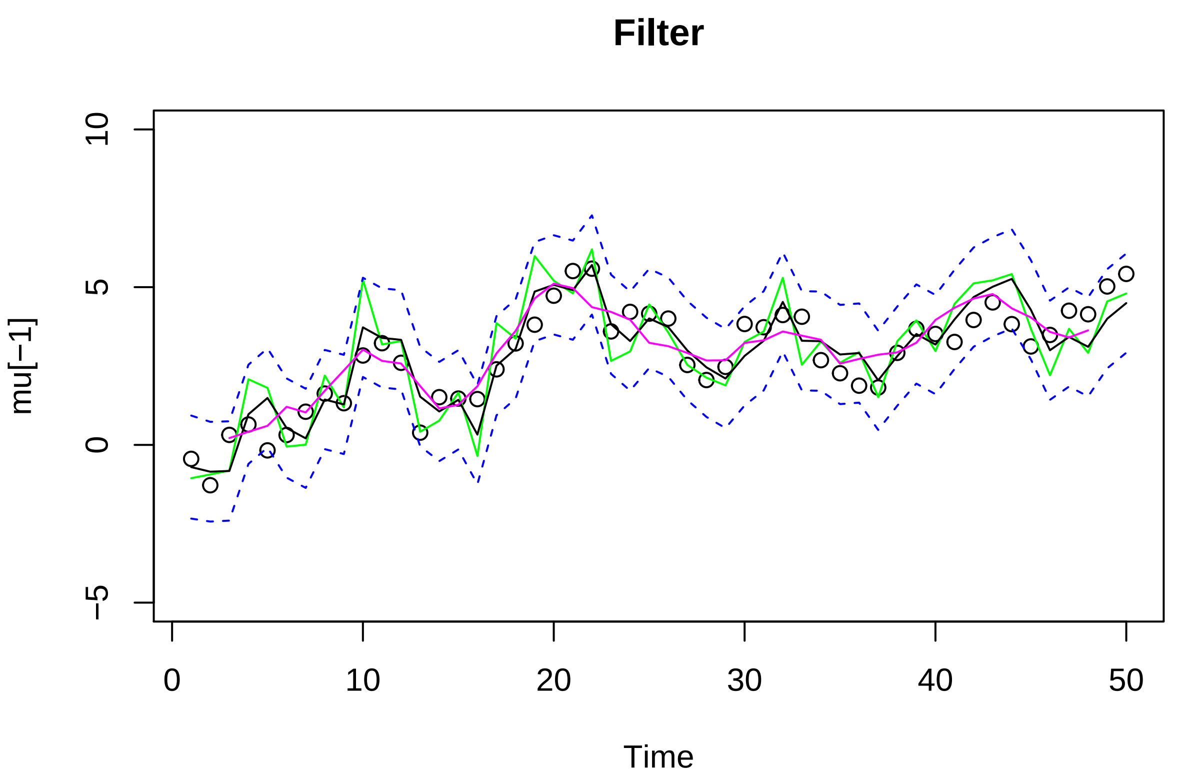
Answer: We includes an additional line in the Filter plot, which shows the moving average smoother of order 5. The dots represent the real states, the **green** line the observed values, the **black** line the smoothed values will their prediction interval in **blue** and the **purple** line represents the moving average.

We see that the moving average is less bumpy and tends to represent the mean value. The filtering is edgier and follows the the real oberstaions more closely; at least it also covers the spikes quite well while the movering average fails doing so (which does not necessarily make it less accurate). As we know that the changes are basically just random noise which we cannot model, we think that the moving average would provide the better model here.

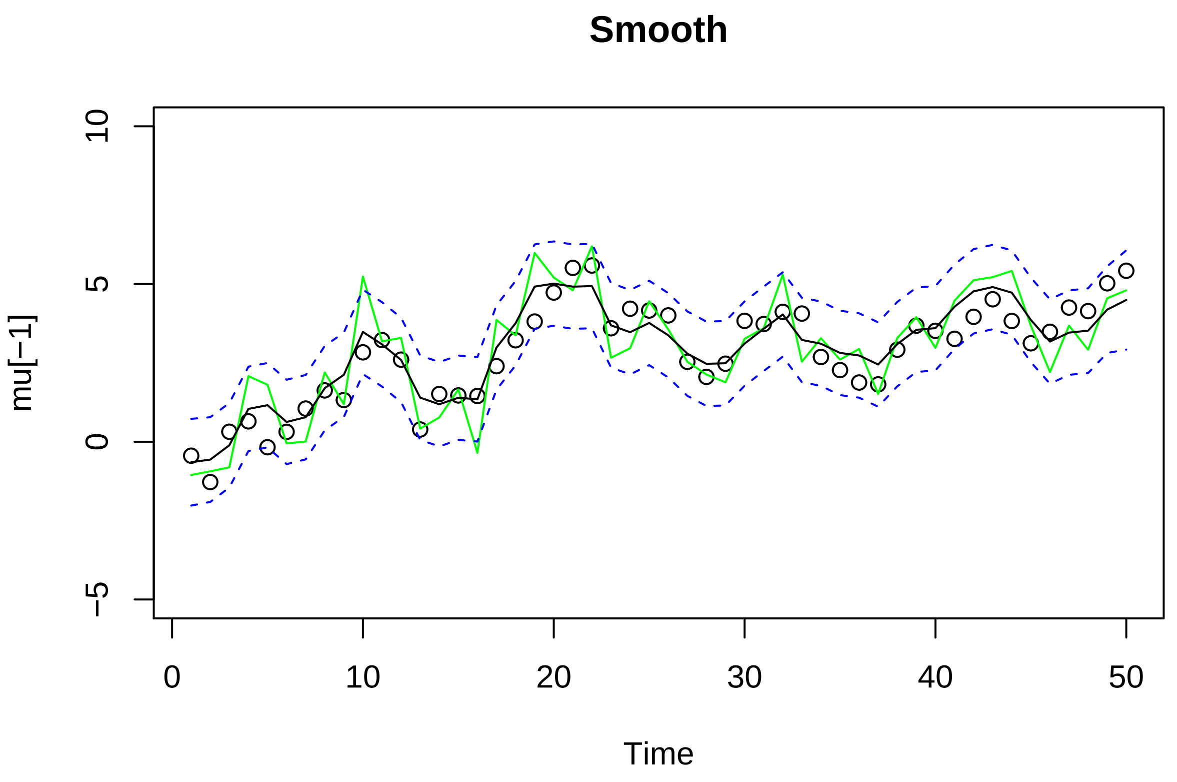
```r
# generate dataset
set.seed(1)
num = 50
w = rnorm(num+1, 0, 1)
v = rnorm(num, 0, 1)
mu = cumsum(w) # state: mu[0], mu[1],..., mu[50]
y = mu[-1] + v # obs: y[1],..., y[50]

# 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)
```



```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)
lines(forecast::ma(y, order=5), col=6)
```



```
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]
```

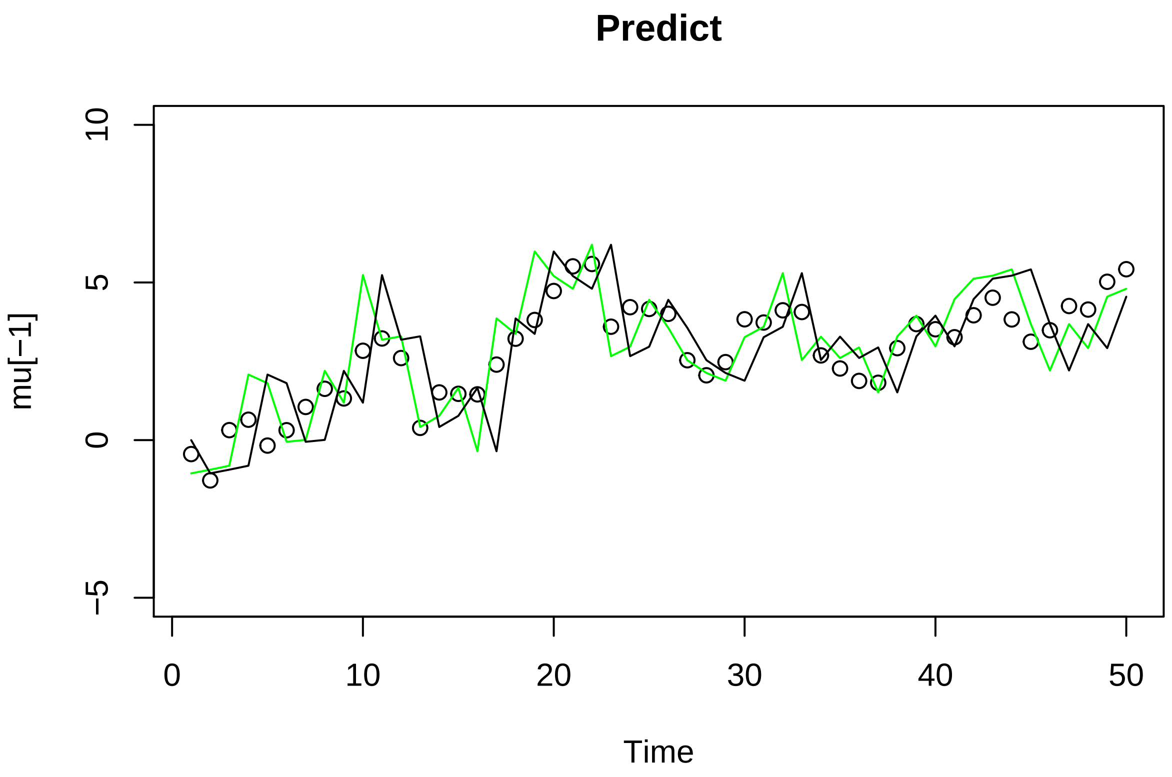
```
## [1] -0.6264538
```

1.3 c) Small R

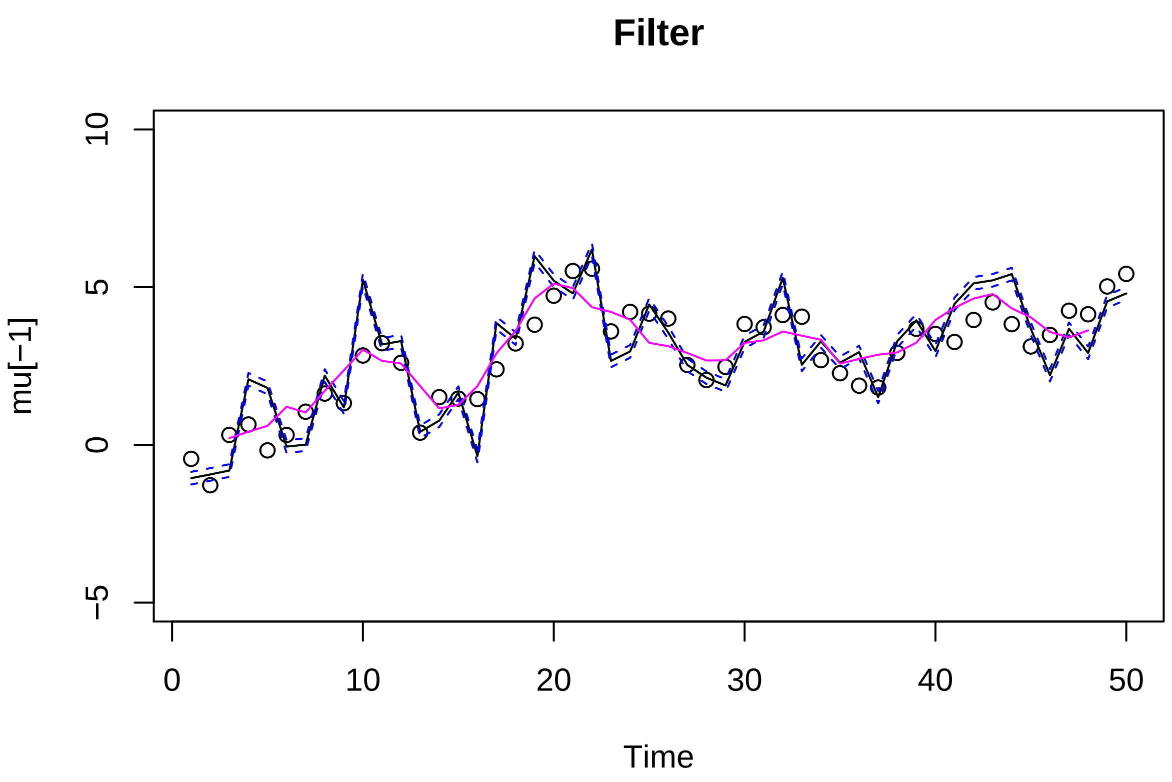
Task: Also, compare the filtering outcome when R in the filter is 10 times smaller than its actual value while Q in the filter is 10 times larger than its actual value. How does the filtering outcome varies?

Answer: Changig the values of Q and R as described gives way more confidence in the observed values compared to the model. That is exactly what we observe in the plots, we basically take the observations as our prediction.

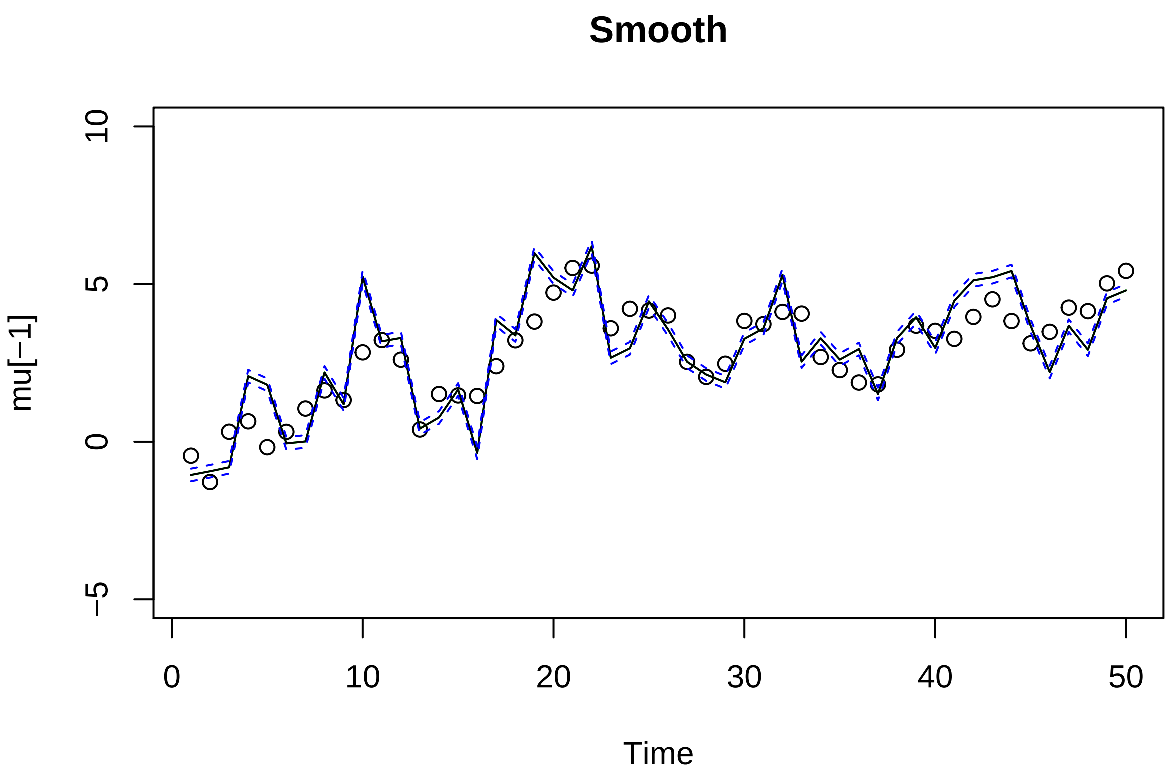
```
# generate dataset
set.seed(1)
num = 50
w = rnorm(num+1, 0, 1)
v = rnorm(num, 0, 1)
mu = cumsum(w) # state: mu[0], mu[1],..., mu[50]
y = mu[-1] + v # obs: y[1],..., y[50]

# filter  and  smooth (Ksmooth0 does both)
ks = Ksmooth0(num, y, A=1, mu0=0, Sigma0=1, Phi=1, cQ=1*10, cR=1/10)
# 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)
```



```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)
lines(forecast::ma(y, order=5), col=6)
```



```
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]
```

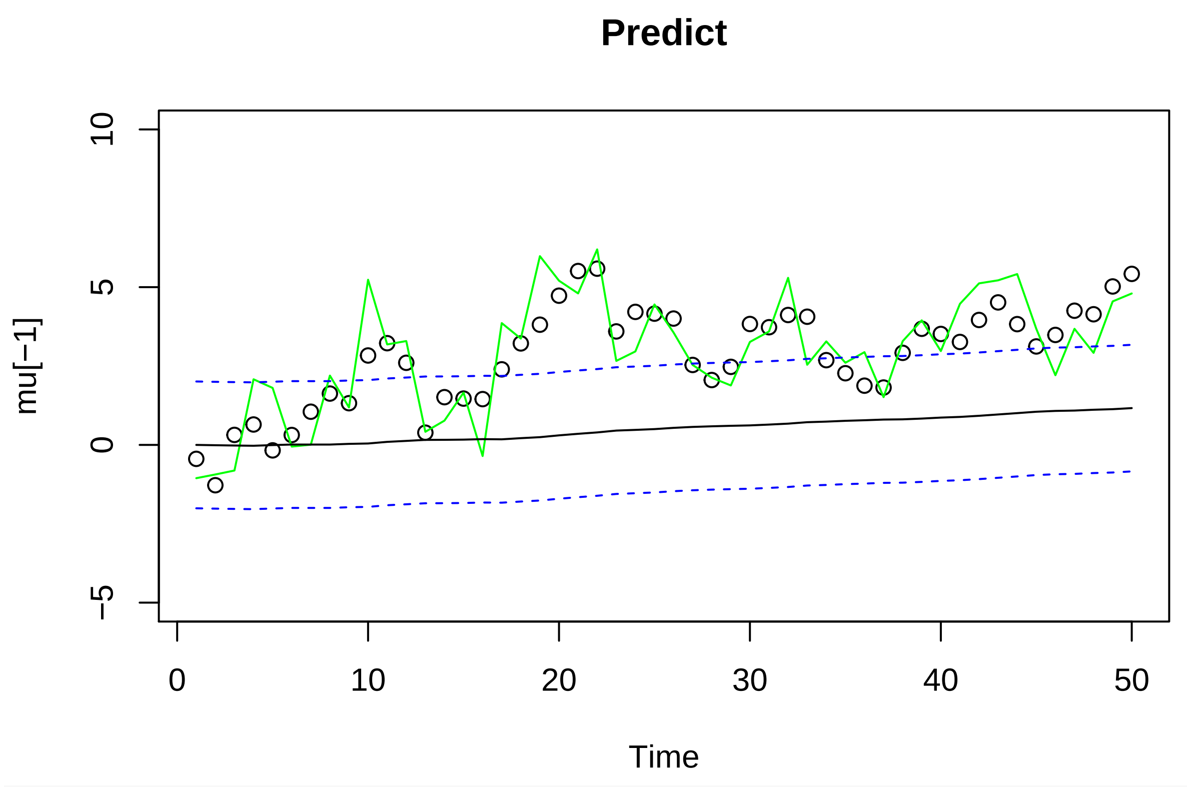
```
## [1] -0.6264538
```

1.4 d) Large R

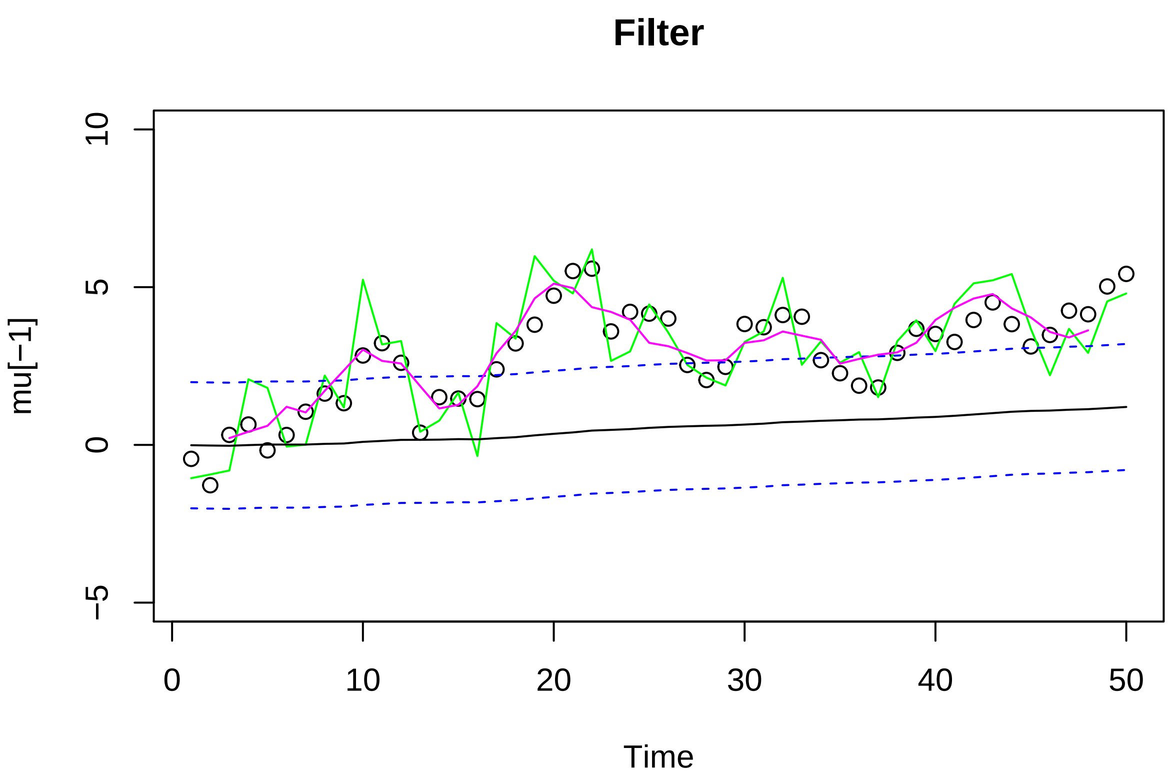
Task: Now compare the filtering outcome when R in the filter is 10 times larger than its actual value while Q in the filter is 10 times smaller than its actual value. How does the filtering outcome varies?

Answer: Changig the values of Q and R opposite to the previous exercise gives way more confidence in the model compared to the observations. That is exactly what we observe in the plots, we basically take the model as our prediction, which is just slightly adjustet over time by our observations, but not enough to actually level out the large error our *assumption* of R and Q caused. The model, as described in exercise **a)** basically assumes a non-changig state, if the error is removed.

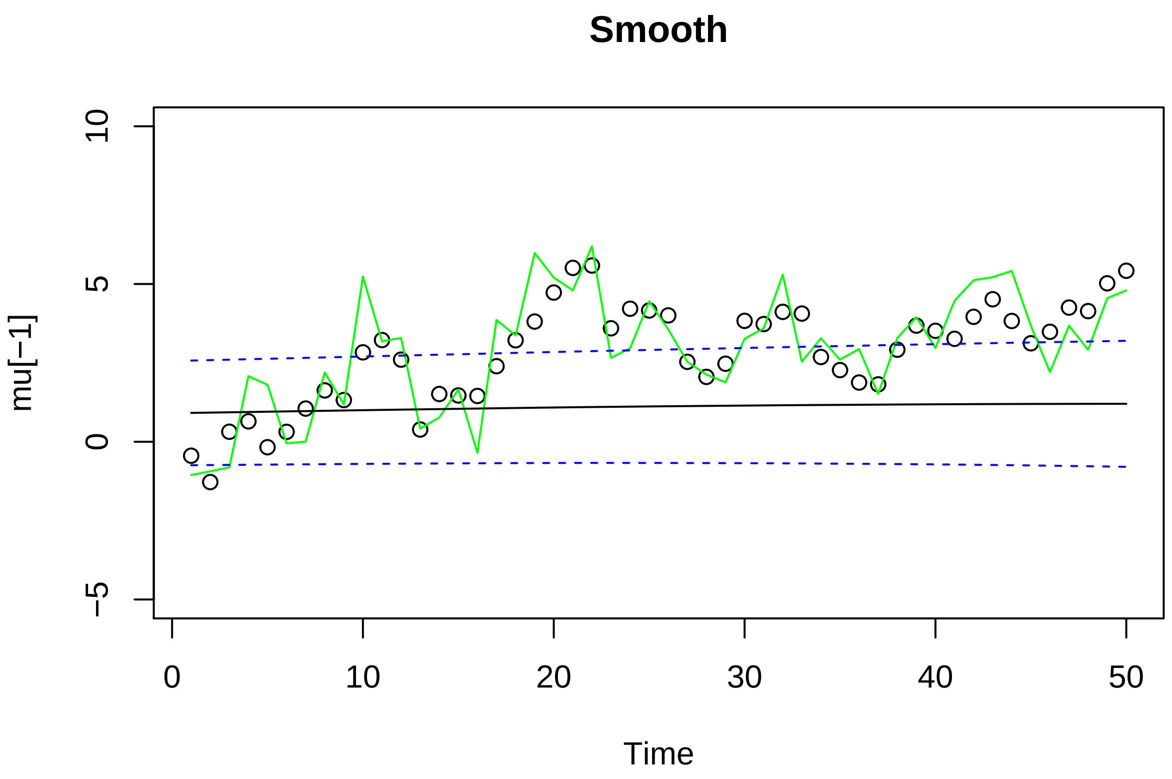
```
# generate dataset
set.seed(1)
num = 50
w = rnorm(num+1, 0, 1)
v = rnorm(num, 0, 1)
mu = cumsum(w) # state: mu[0], mu[1],..., mu[50]
y = mu[-1] + v # obs: y[1],..., y[50]

# filter  and  smooth (Ksmooth0 does both)
ks = Ksmooth0(num, y, A=1, mu0=0, Sigma0=1, Phi=1, cQ=1/10, cR=1*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)
lines(forecast::ma(y, order=5), col=6)
```



```r
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
```

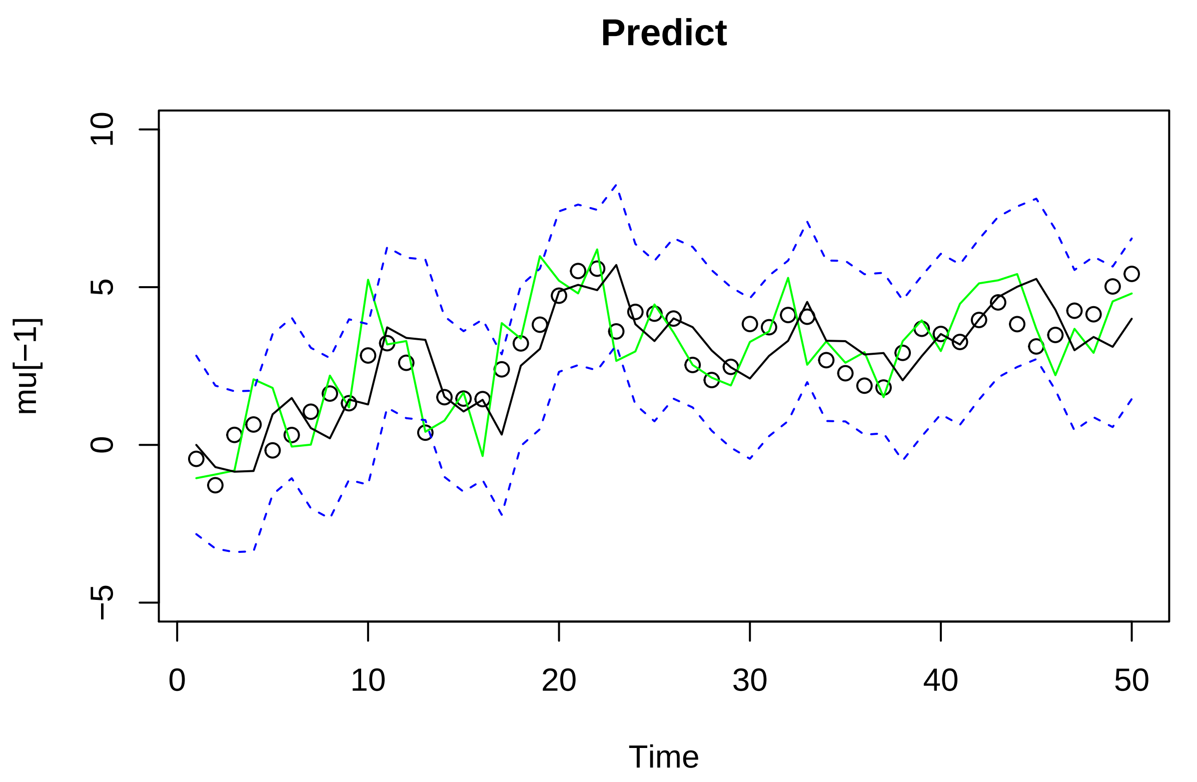
1.5 e) Kalman Filter Implementation

Task: Implement your own Kalman filter and replace `ksmooth0()` function with your script.

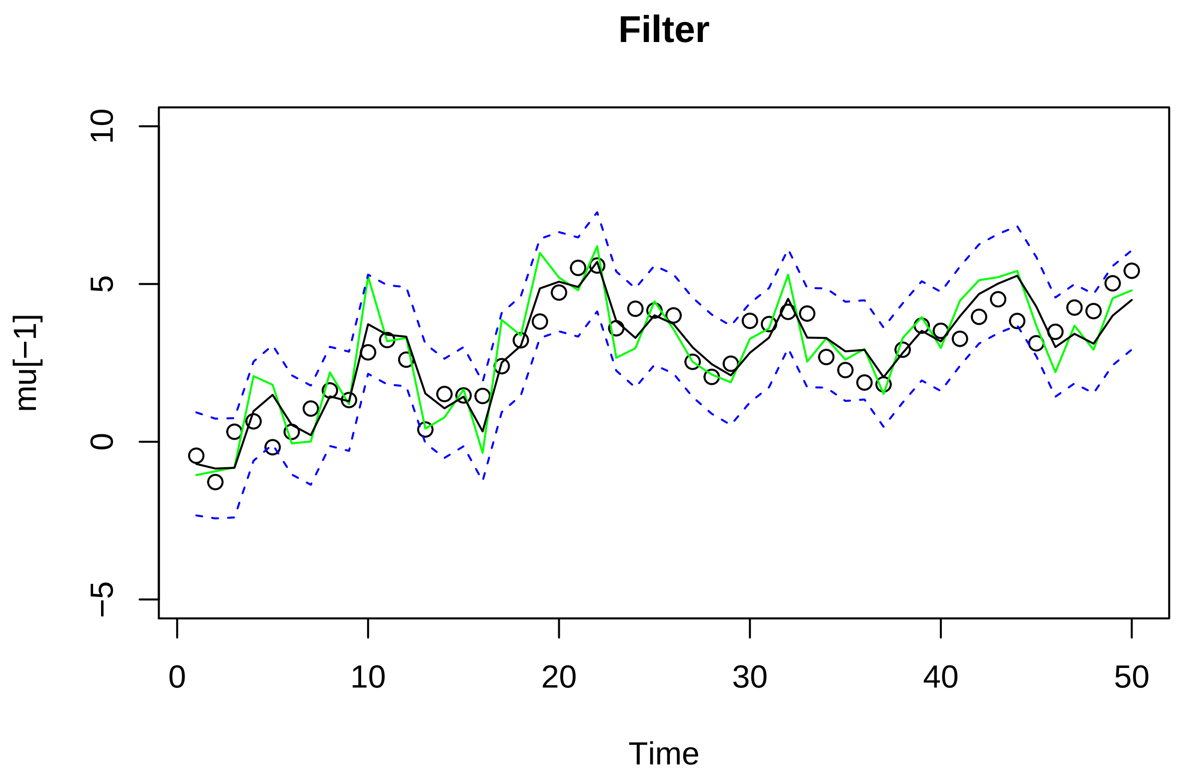
1.6 f) Kalman Gain Interpretation

Task: How do you interpret the Kalman gain?

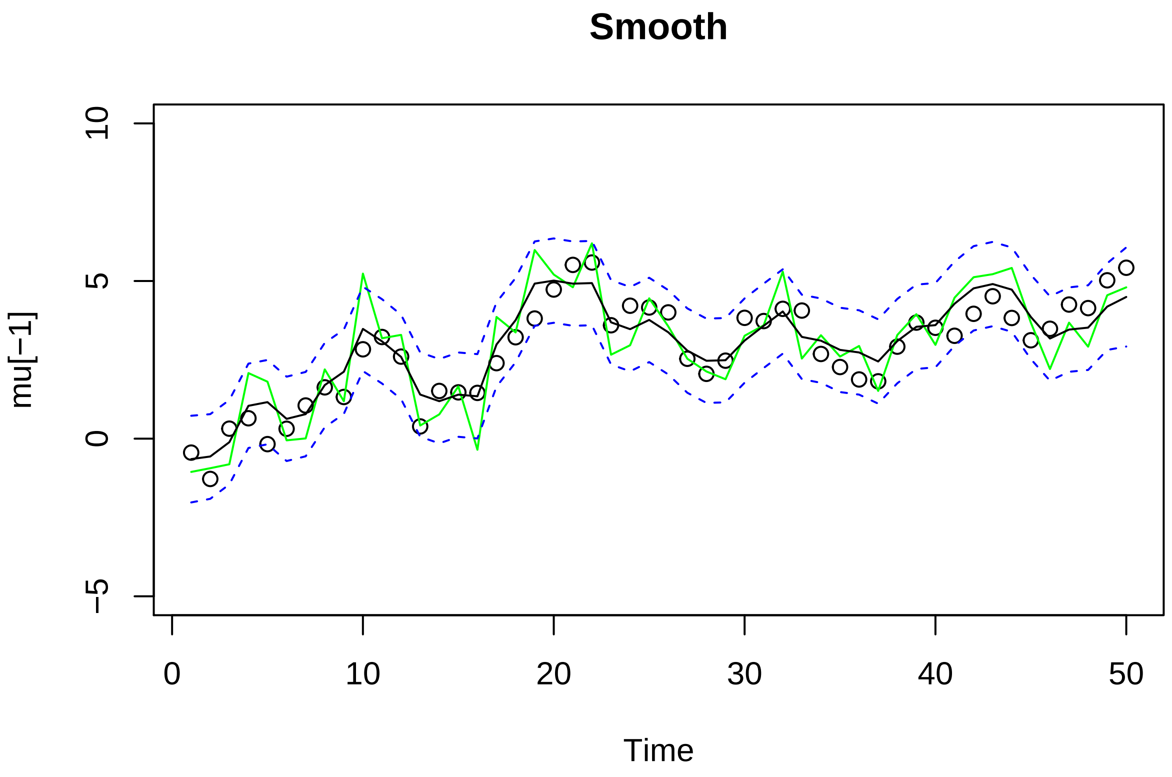
```
# generate  dataset
set.seed(1)
num = 50
w = rnorm(num+1, 0, 1)
v = rnorm(num, 0, 1)
mu = cumsum(w) # state: mu[0], mu[1],..., mu[50]
y = mu[-1] + v # obs: y[1],..., y[50]

# 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)
```



```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)
```



```
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
```

2 Source Code

```
library(astsa)
library(forecast)
library(kableExtra)
knitr::opts_chunk$set(echo = TRUE)

# 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))
```

```r
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)
lines(forecast::ma(y, order=5), col=6)

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]


# 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*10, cR=1/10)
# 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)
lines(forecast::ma(y, order=5), col=6)

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]


# 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=1/10, cR=1*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)
lines(forecast::ma(y, order=5), col=6)

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]


# 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]
```