Time Series Analysis - lab03

Lennart Schilling (lensc874)

2019-10-10

Contents

Lab03

Assignment 1. Implementation of Kalman filter.

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.

State space model:
$$\begin{aligned} \mathbf{Z}_t &= A_{t-1}\mathbf{Z}_{t-1} + e_t \\ \mathbf{x}_t &= C_t \mathbf{z}_t + \nu_t \\ \nu_t &\sim N(0, R_t) \\ e_t &\sim N(0, Q_t) \end{aligned}$$

Given (adjusted) script

```r
library(astsa)

# Generating states and observations from true state space model.
  # Setting parameters.
  set.seed (1)
  n = 50
  Q_true = 1
  R_true = 1
  # Generating e_t (transition errors).
  e_t = rnorm (n = n + 1,
               mean = 0,
               sd = sqrt(Q_true)) # (e_0, e_1, ..., e_50)
  # Generating v_t (observation errors).
  v_t = rnorm (n = n,
               mean = 0,
               sd = sqrt(R_true)) # (v_1, v_2, ..., v_50)
  # Generating z_t (hidden states).
  z_t = cumsum (e_t) # state : z_0, z_1, ..., z_50
  # Generating x_t (observations).
  x_t = z_t[-1] + v_t # obs: x_1 , ..., x_50

# Implementing Kalman filter.
apply_kalman_filter = function(n, # T.
                               obs, # Given observations.
                               A, # Transition matrix A.
                               C, # Emission matrix C.
                               Q,
                               R,
                               mu0, # Mean of initial state.
                               sigma0) {
  # Filtering and smoothing (Ksmooth 0 does both) to predict hidden states.
  ks = Ksmooth0(num = n,
                y = obs, # observations
                A = C, # observation matrix C
                mu0 = mu0, # initial mean
                Sigma0 = sigma0, # initial covariance
                Phi = A, # intial transition matrix A
                cQ = Q, # Q
                cR = R) # R
  # Plotting results.
  par(mfrow = c(3, 1))
```

```r
  Time = 1:n
    # Prediction.
    plot (Time , z_t[-1], main = 'Predict ', ylim = c(-5, 10)) # true hidden states.
    lines (Time , x_t, col = " green ") # observations.
    lines (ks$xp) # state predictions.
    lines (ks$xp + 2 * sqrt (ks$Pp), lty = 2, col = 4) # Prediction band for predictions.
    lines (ks$xp - 2 * sqrt (ks$Pp), lty = 2, col = 4) # Prediction band for predictions.
    # Filtering.
    plot (Time , z_t[-1], main = 'Filter ', ylim = c(-5, 10))
    lines (Time , x_t, col = " green ")
    lines (ks$xf)
    lines (ks$xf + 2 * sqrt (ks$Pf), lty = 2, col = 4)
    lines (ks$xf - 2 * sqrt (ks$Pf), lty = 2, col = 4)
    # Smoothing.
    plot (Time , z_t[-1], main = 'Smooth ', ylim = c(-5, 10))
    lines (Time , x_t, col = " green ")
    lines (ks$xs)
    lines (ks$xs + 2 * sqrt (ks$Ps), lty = 2, col = 4)
    lines (ks$xs - 2 * sqrt (ks$Ps), lty = 2, col = 4)
  # Printing information about initialization.
  # True z_0.
  z_t[1]
  # Initial smoother mean, sd.
  ks$x0n
  sqrt (ks$P0n) # initial value info
}

# Applying Kalman filter.
apply_kalman_filter(n = n, obs = x_t, A = 1, C = 1, Q = 1, R = 1, mu0 = 0, sigma0 = 1)
```

Given Kalman filtering algorithm.

Inputs: A_t, C_t, Q_t, R_t, m_0, P_0 and $\mathbf{x}_{1:T}$.
initialization
$m_{1|0} \leftarrow m_0$, $P_{1|0} \leftarrow P_0$
for $t = 1$ to T **do**
    observation update step
    $K_t \leftarrow P_{t|t-1} C_t^{\mathrm{T}} (C_t P_{t|t-1} C_t^{\mathrm{T}} + R_t)^{-1}$
    $m_{t|t} \leftarrow m_{t|t-1} + K_t(\mathbf{x}_t - C_t m_{t|t-1})$
    $P_{t|t} \leftarrow (I - K_t C_t) P_{t|t-1}$
    prediction step
    $m_{t+1|t} \leftarrow A_t m_{t|t}$
    $P_{t+1|t} \leftarrow A_t P_{t|t} A_t^{\mathrm{T}} + Q_{t+1}$
end for
Outputs: $m_{t|t}$, $P_{t|t}$ for $t = 1 : T$

1a.

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

In the provided code, the following parameter settings are used for simulation:

- A = 1
- C = 1
- Q = 1
- R = 1

Therefore, we get the following expression of the state space model which is being simulated:

$\mathbf{Z}_t = \mathbf{Z}_{t-1} + e_t$

$\mathbf{x}_t = \mathbf{z}_t + \nu_t$

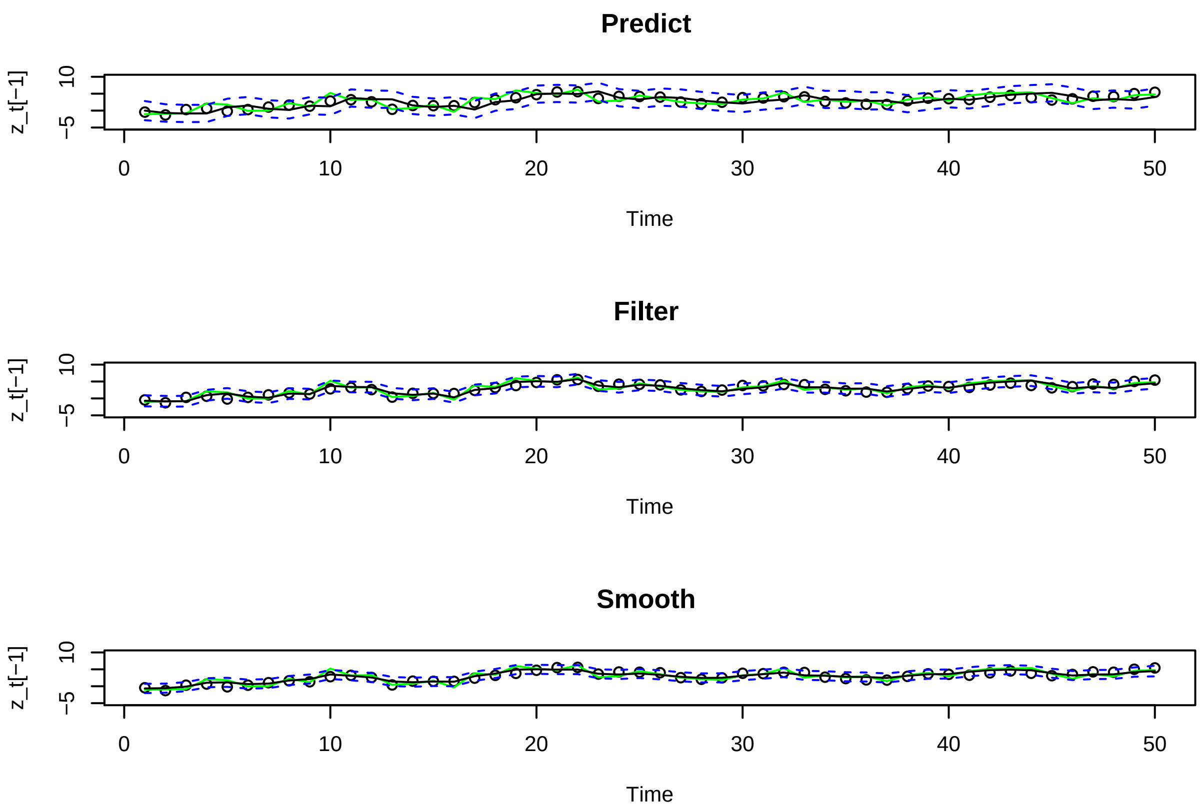
$\nu_t \sim N\left(0, 1\right)$

$e_t \sim N\left(0, 1\right)$

1b.

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

Results of running given script.




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

Results for MA(5)-smoother.

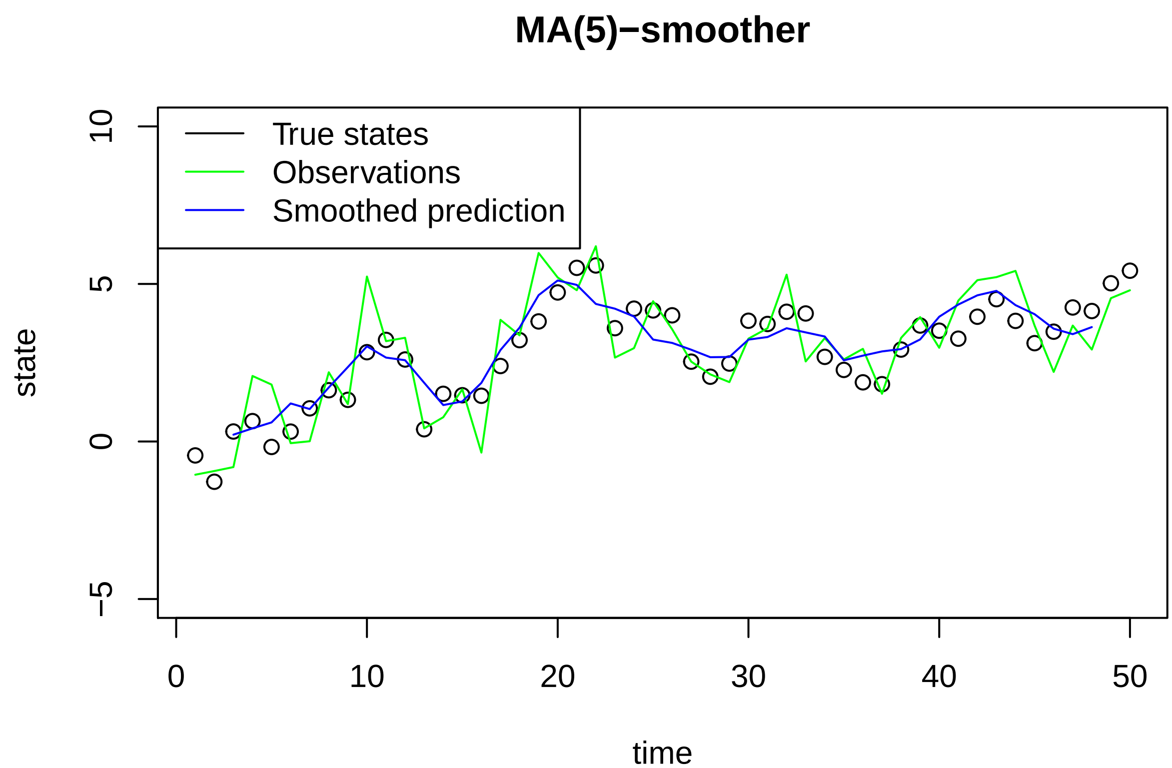
The moving average smoother averages the nearest order periods of each observation.

```r
library(forecast)
```

```

Attaching package: 'forecast'

The following object is masked from 'package:astsa':

    gas
```

```r
ma_5_smoother = ma(x = x_t,
                   order = 5)
plot (x = 1:length(x_t) , y = z_t[-1], col = "black",
      main = 'MA(5)-smoother', ylim = c(-5, 10),
      ylab = "state", xlab = "time") # true hidden states.
lines (x = 1:length(x_t), y = x_t, col = "green") # observations.
lines (x = 1:length(x_t), y = ma_5_smoother, col = "blue") # MA(5)-state-predictions.
```

```r
legend(x = -1, y=11,
       legend = c("True states", "Observations", "Smoothed prediction"),
       col = c("black", "green", "blue"),
       lty = 1)
```

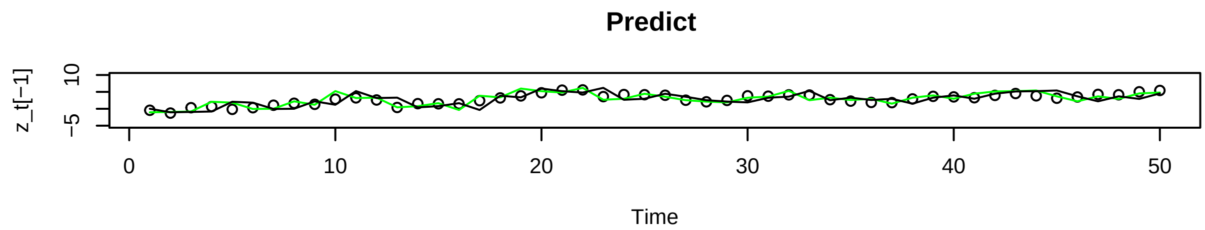


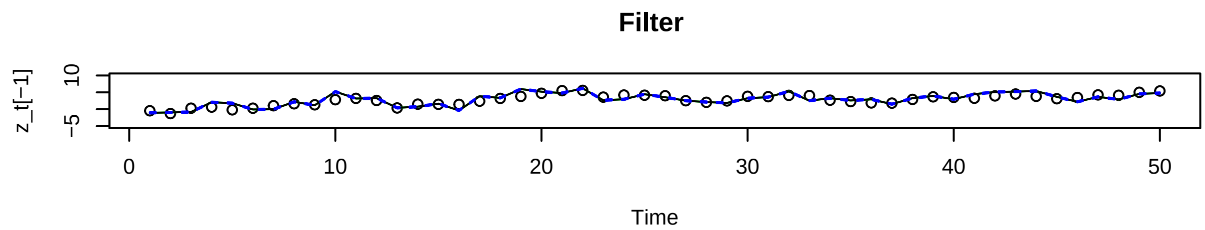
CONCLUSIONS:

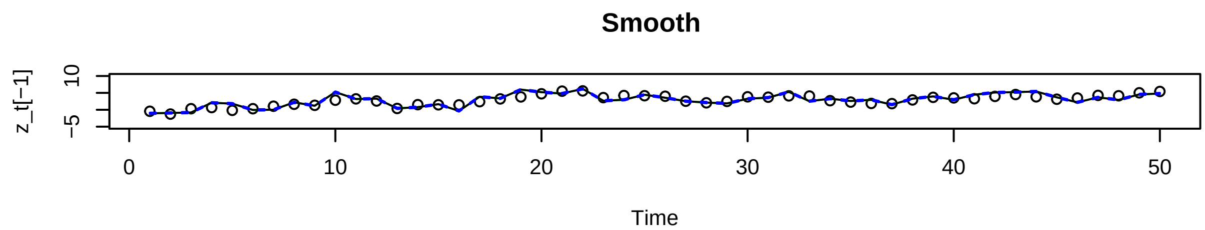
1c.

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?

```r
run_kalman_filter(n = n, states = z_t, obs = x_t,
                  A = 1, C = 1, Q = 10/1, R = 1/10, mu0 = 0, sigma0 = 1)
```







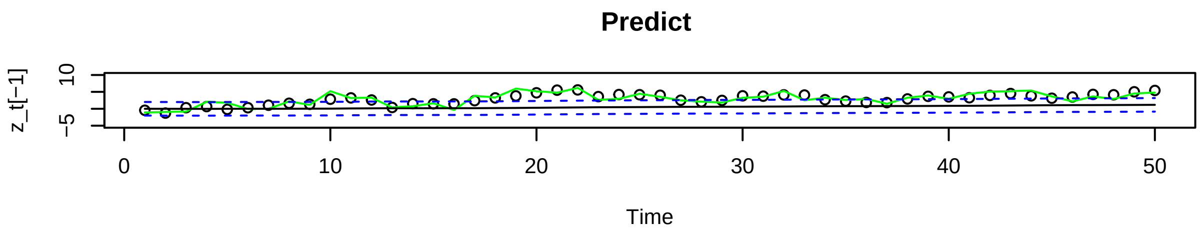

```
          [,1]
[1,] 0.9950377
```

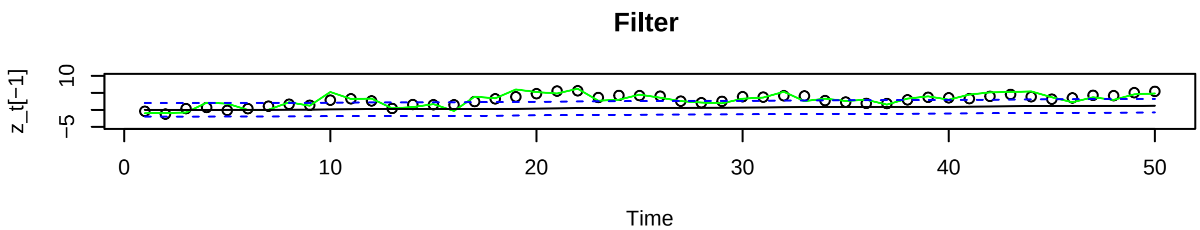
CONCLUSIONS

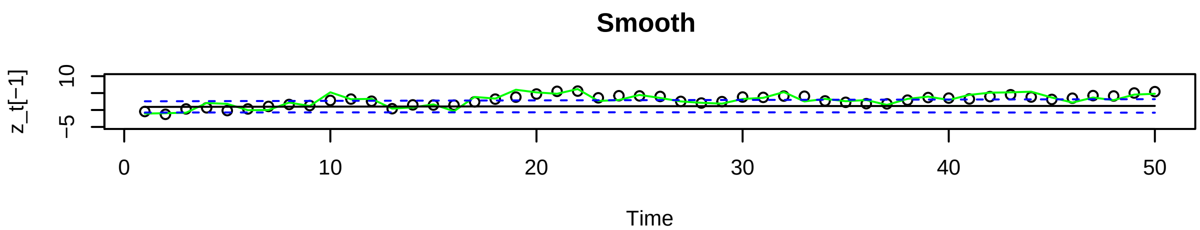
1d.

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?

```r
run_kalman_filter(n = n, states = z_t, obs = x_t,
                  A = 1, C = 1, Q = 1/10, R = 10/1, mu0 = 0, sigma0 = 1)
```







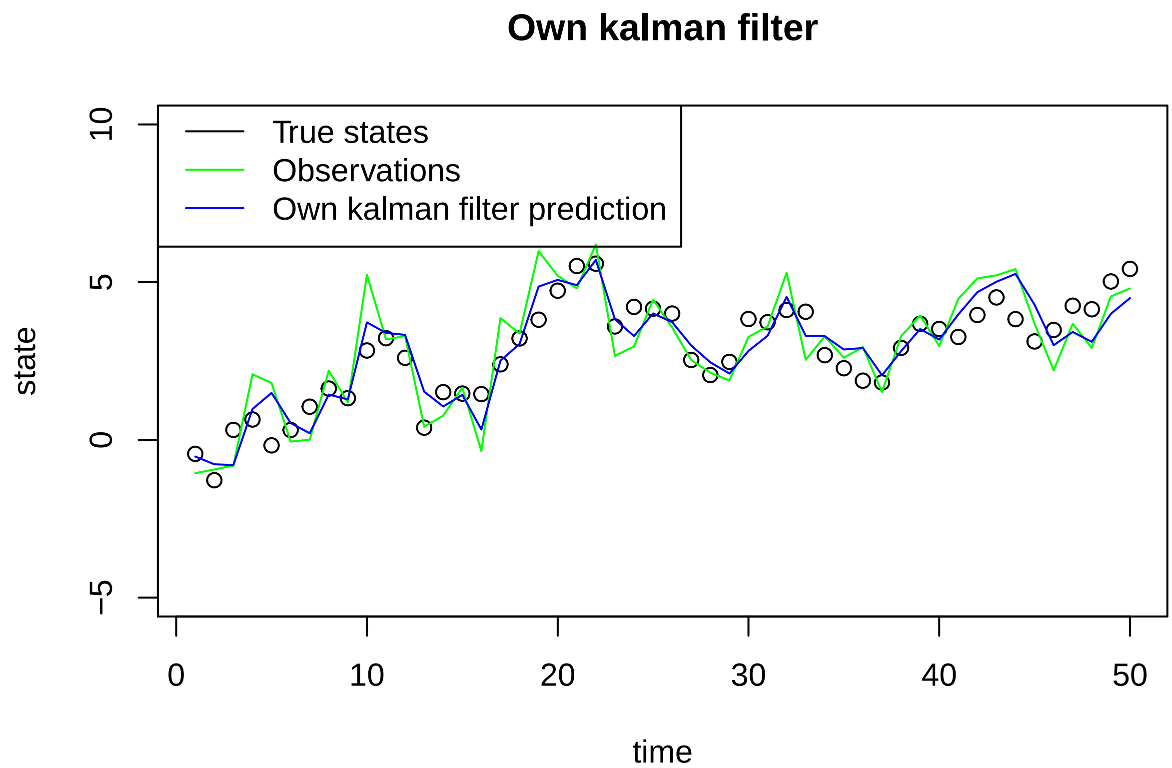

```
        [,1]
[1,] 0.82731
```

CONCLUSIONS

1e.

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

```
# Implementing own kalman filter.
own_kalman = function(n, # T.
                      obs, # Observations.
                      A, # Transition matrix A.
                      C, # Emission matrix C.
                      Q,
                      R,
                      m_0, # Mean of initial state.
                      P_0) {
  # Initialization.
  k_gain_t = c()
  m_t = m_0
  P_t = P_0

  for (t in 1:n) {

    # Observation update.
    k_gain_t[t] = P_t[t] * t(C) * solve(C * P_t[t] * t(C) + R)
    m_t[t] = m_t[t] + k_gain_t[t] * (obs[t] - C * m_t[t])
    P_t[t] = (1 - k_gain_t[t] * C) * P_t[t]

    # Prediction step.
    m_t[t+1] = A * m_t[t]
    P_t[t+1] = A * P_t[t] * t(A) + Q
  }

  # Return.
  return(list(m_t = m_t[1:n], P_t = P_t[1:n]))
}

# Running own kalman filter on same generate observations to predict states.
own_kalman_results = own_kalman(n = n, obs = x_t, A = 1, C = 1, Q = 1, R = 1, m_0 = 0, P_0 = 1)

# Plotting results.
# Prediction.
plot(x = 1:length(x_t), y = z_t[-1], main = 'Own kalman filter',
     ylab = "state", xlab = "time", ylim = c(-5, 10)) # true hidden states.
lines (x = 1:length(x_t), y = x_t, col = " green ") # observations.
lines (x = 1:length(x_t), y = own_kalman_results$m_t, col = "blue") # state predictions.
legend(x = -1, y=11,
       legend = c("True states", "Observations", "Own kalman filter prediction"),
       col = c("black", "green", "blue"),
       lty = 1)
```



1f.

How do you interpret the Kalman gain?