HW3_Q2

2023-02-05

```
library(ggplot2)
library(gridExtra)
library(tidyverse)
library(xts)
library(dlm)
```

The data set linked below contains a time series generated from the following univariate dynamic linear model:

$$\begin{aligned}
y_t &= f_t\theta_t + v_t \\
\theta_t &= g_t\theta_{t-1} + w_t \\
\\
F_t &= 1.2 \\
G_t &= 0.8 \\
\theta_0 &\sim N(0, \sigma_\theta^2 = 25) \\
v_t &\sim N(0, \sigma_v^2 = 9) \\
w_t &\sim N(0, \sigma_w^2 = 4)
\end{aligned}$$

Question 2a.

a. Apply a Kalman filter to this data to make one-step ahead predictions of θ_t given $y_{1:(t-1)}$. Create a times-series plot containing the observations y_t and one-step ahead predictions of θ_t. Include a 95% confidence band around your θ_t predictions.

```r
fn_2a = "DLM_Data.csv"

### Read in CSV
df = read.csv(fn_2a)

### Initialize a DLM model object with parameters at the given values above
dlm_mod = dlm(FF = 1.2,
               GG = 0.8,
               V = 9,  ### Do not sqrt() these values as inputs are sig^2 NOT sig
               W = 4,  ### Do not sqrt() these values as inputs are sig^2 NOT sig
               m0 = 0,
               C0 = 25)  ### Do not sqrt() these values as inputs are sig^2 NOT sig

### Filter the data using the DLM above
df_filtered = dlmFilter(y = df$yt, mod = dlm_mod)

### Smooth data based off of filtered data above. For use in sub-question e.
df_smoothed = dlmSmooth(df_filtered)

### Forecast data based off of filtered data above. For use in sub-question f.
df_forecasted = dlmForecast(df_filtered, nAhead = 10)
```

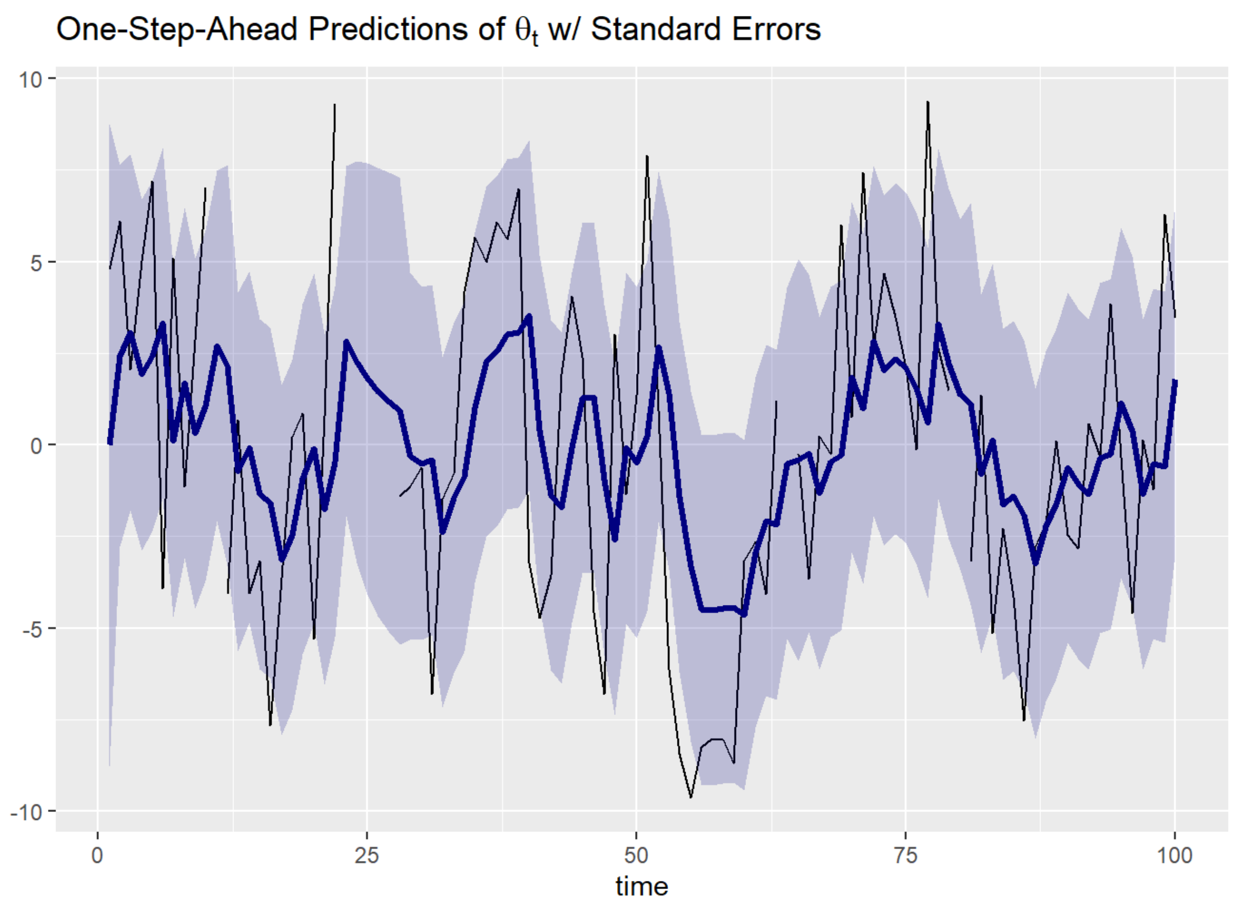
```r
### Define all variables in advance to make it easier to reference in questions below!
fix_Ft = 1.2
fix_Gt = 0.8
fix_Vt = 9
fix_Wt = 4
fix_m0 = 0
fix_C0 = 25

dlm_at = df_filtered$a
dlm_ft = df_filtered$f
dlm_mt = dropFirst(df_filtered$m)
dlm_Rt = unlist(dlmSvd2var(df_filtered$U.R, df_filtered$D.R))
dlm_Ct = dropFirst(unlist(dlmSvd2var(df_filtered$U.C, df_filtered$D.C)))
dlm_st = dropFirst(df_smoothed$s)
dlm_sst = dropFirst(unlist(dlmSvd2var(df_smoothed$U.S, df_smoothed$D.S)))
dlm_forc = df_forecasted$f
dlm_forcvar = df_forecasted$Q
```

```
### One-step-ahead predictions of theta give y_1:(t-1) and standard errors
df$prediction = dlm_at
df$predSE = sqrt(dlm_Rt)

### Create the base plot of observations
gg_base = ggplot(df, aes(y = yt, x = time)) + geom_line(color = "black")

### Layer the one-step-ahead predictions over the observations
gg_base +
  geom_line(data = df, aes(y = prediction, x = time), color = "navy", linewidth = 1.2) +
  geom_ribbon(data = df, aes(x = time, ymin = prediction - 1.96 * predSE, ymax = prediction + 1.96 *
predSE), fill = "navy", alpha = 0.2) +
    labs(title = expression( paste("One-Step-Ahead Predictions of ", theta[t], " w/ Standard Error
s"))) +
    ylab("")
```



Report the numerical values found for a_{40} and R_{40}.

- a_{40} = 3.529
- R_{40} = 5.951

Question 2b.

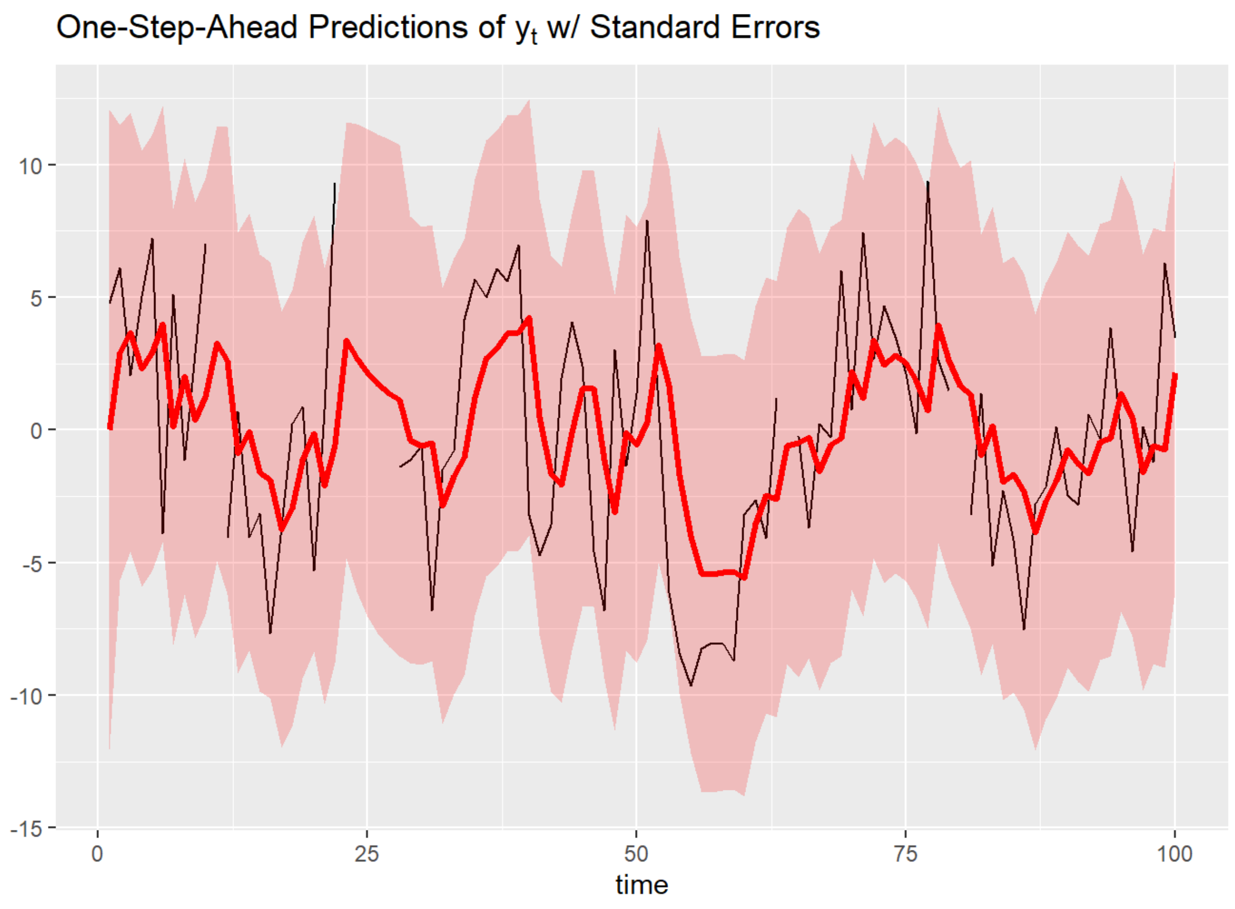
b. Apply a Kalman filter to this data to make one-step-ahead predictions of y_t given $y_{1:(t-1)}$. Create a time-series plot showing the observed values of y_t and one-step ahead predictions of y_t. Include a 95% confidence band around your y_t predictions.

```
### Predictive dist. y_t given y_1:(t-1)
### For now, use the dlm generated parameters.
m_ft = fix_Ft * dlm_at ### This matches df_filtered$f
m_Qt = fix_Ft * dlm_Rt * fix_Ft + fix_Vt ### Since Ft and Vt do not vary with time, constants can be
used. Transpose of a scalar is the same scalar.

### Check manual vs. dlm
# m_ft - dlm_ft ### If all elements == 0 then operations above should be correct
```

```
### One-step-ahead predictions of yt and standard errors
df$pred_yt = dlm_ft
df$pySE = sqrt(m_Qt) ### Use values calculated from above

gg_base +
  # geom_line(data = df, aes(y = prediction, x = time), color = "navy", linewidth = 1.2) +
  geom_line(data = df, aes(y = pred_yt, x = time), color = "red", linewidth = 1.2) +
  geom_ribbon(data = df, aes(x = time, ymin = pred_yt - 1.96 * pySE, ymax = pred_yt + 1.96 * pySE), f
ill = "red", alpha = 0.2) +
    labs(title = expression( paste("One-Step-Ahead Predictions of ", y[t], " w/ Standard Errors"))) +
    ylab("")
```



Report the numerical values of f_{40} and Q_{40}. (Hint: R's DLM package does not provide these values directly, so you will need to calculate them.)

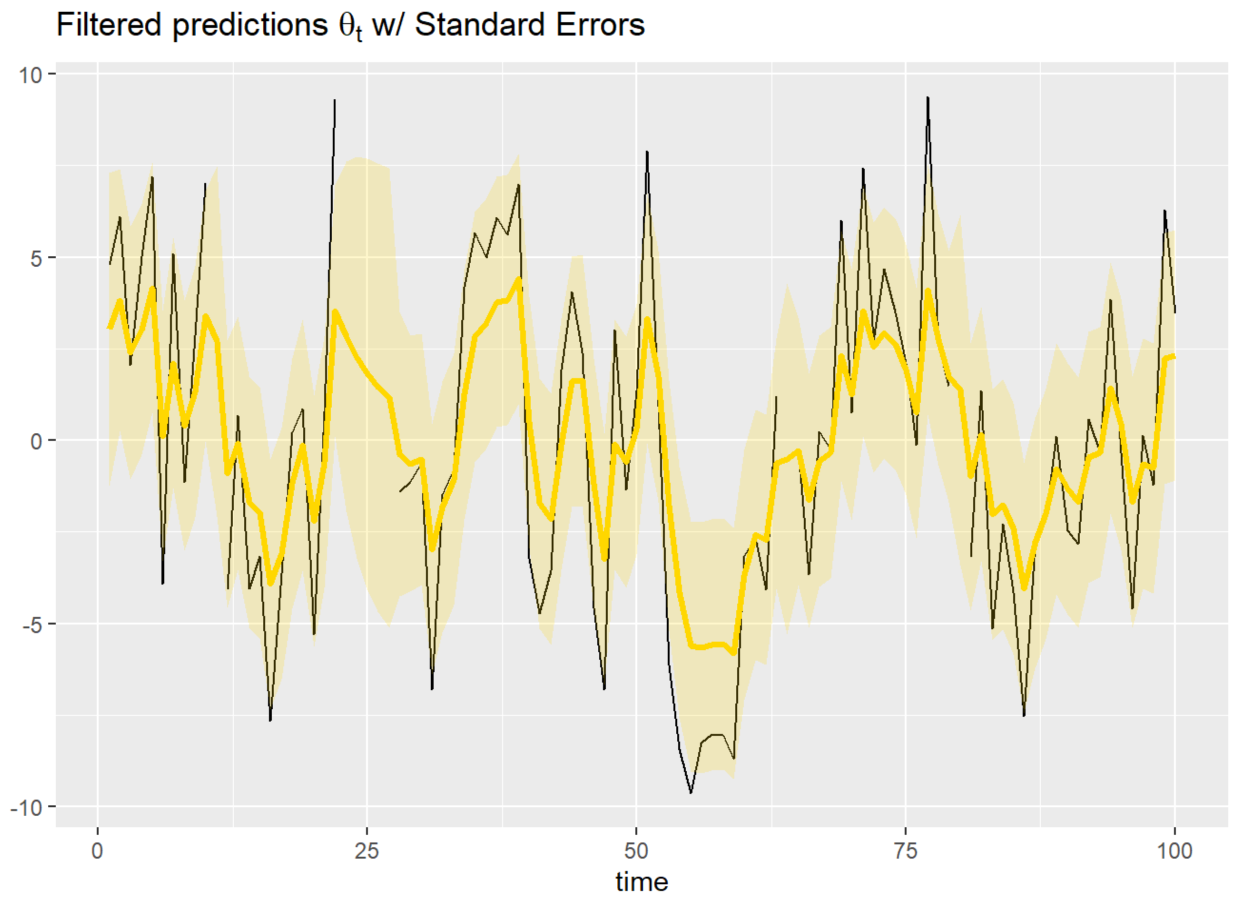
- f_{40} = 4.235
- Q_{40} = 17.569

Question 2c.

c. Apply a Kalman filter to this data to find the filtering distribution of the values of θ_t given $y_{1:(t)}$. Create a time-series plot showing the observed values of y_t and filtered predictions of θ_t. Include a 95% confidence band around your θ_t predictions.

```
### Filtering distribution of theta_t given y_1:t
df$filtered = dlm_mt
df$filtSE = sqrt(dlm_Ct)

gg_base +
  # geom_line(data = df, aes(y = prediction, x = time), color = "navy", linewidth = 1.2) +
  # geom_line(data = df, aes(y = pred_yt, x = time), color = "red", linewidth = 1.2) +
  geom_line(data = df, aes(y = filtered, x = time), color = "gold", linewidth = 1.2) +
  geom_ribbon(data = df, aes(x = time, ymin = filtered - 1.96 * filtSE, ymax = filtered + 1.96 * filt
SE), fill = "gold", alpha = 0.2) +
  labs(title = expression( paste("Filtered predictions ", theta[t], " w/ Standard Errors"))) +
  ylab("")
```



Report the numerical values of m_{40} and C_{40}.

- m_{40} = 0.501
- C_{40} = 3.048

Question 2d.

d. The filtering distribution of $\theta_{22} \mid y_{1:22}$ is $N(m_{22} = 3.539, C_{22} = 3.048)$ (your answer should match this).

- m_{22} = 3.539
- C_{22} = 3.048

Analytically (i.e., not using code) show that the predictive distribution of $\theta_{28} \mid y_{1:27}$ is $N(a_{28} = .928, R_{28} = 10.557)$.

$$a_t = G_t m_{t-1}$$
$$R_t = G_t C_{t-1} G_t' + W_t$$

$$a_t = 0.8m_{27} = 0.8 * 1.15972 = 0.9277$$
$$R_t = 0.8C_{27}0.8^\intercal + W_{27} = 0.8 * 10.24539 * 0.8 + 4 = 10.557047$$

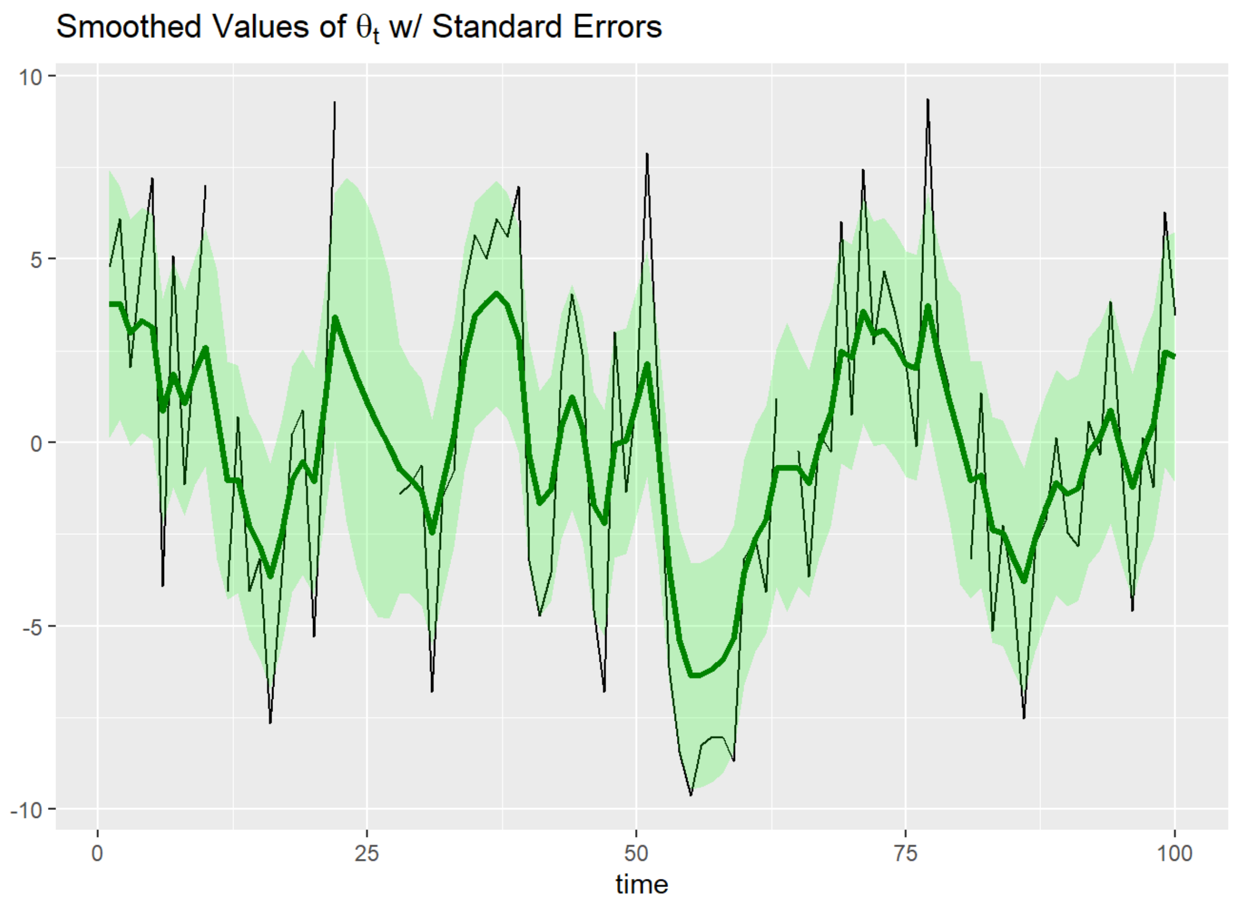
$$a_t = 0.928$$
$$R_t = 10.557$$

Question 2e.

e. Apply a Kalman smoother to this data to create the smoothing distribution for θ_t given $y_{1:T}$. Create a time-series plot showing the observed values of y_t and smoothed estimates of θ_t. Include a 95% confidence band around your θ_t predictions.

```
df$smoothed = dlm_st
df$sSE = sqrt(dlm_sst)

gg_base +
    geom_line(data = df, aes(y = smoothed, x = time), color = "darkgreen", linewidth = 1.2) +
    geom_ribbon(data = df, aes(x = time, ymin = smoothed - 1.96 * sSE, ymax = smoothed + 1.96 * sSE),
                fill = "green",
                alpha = 0.2) +
      labs(title = expression(paste("Smoothed Values of ", theta[t], " w/ Standard Errors"))) +
      ylab("")
```



Additionally, report your values of θ_t for the values of t such that y_t is missing.

```
missing_indices = which(is.na(df_filtered$y)) ### Grab missing time indices
st_for_missing_yt = dlm_st[missing_indices] ### Grab values of theta at missing time indices

data.frame(t_values_of_missing_yt=missing_indices, theta_t = st_for_missing_yt )
```

```
##   t_values_of_missing_yt    theta_t
## 1                     11  0.7579377
## 2                     23  2.5279539
## 3                     24  1.7674521
## 4                     25  1.0953229
## 5                     26  0.4779599
## 6                     27 -0.1155051
## 7                     64 -0.6779439
## 8                     80  0.0865769
```

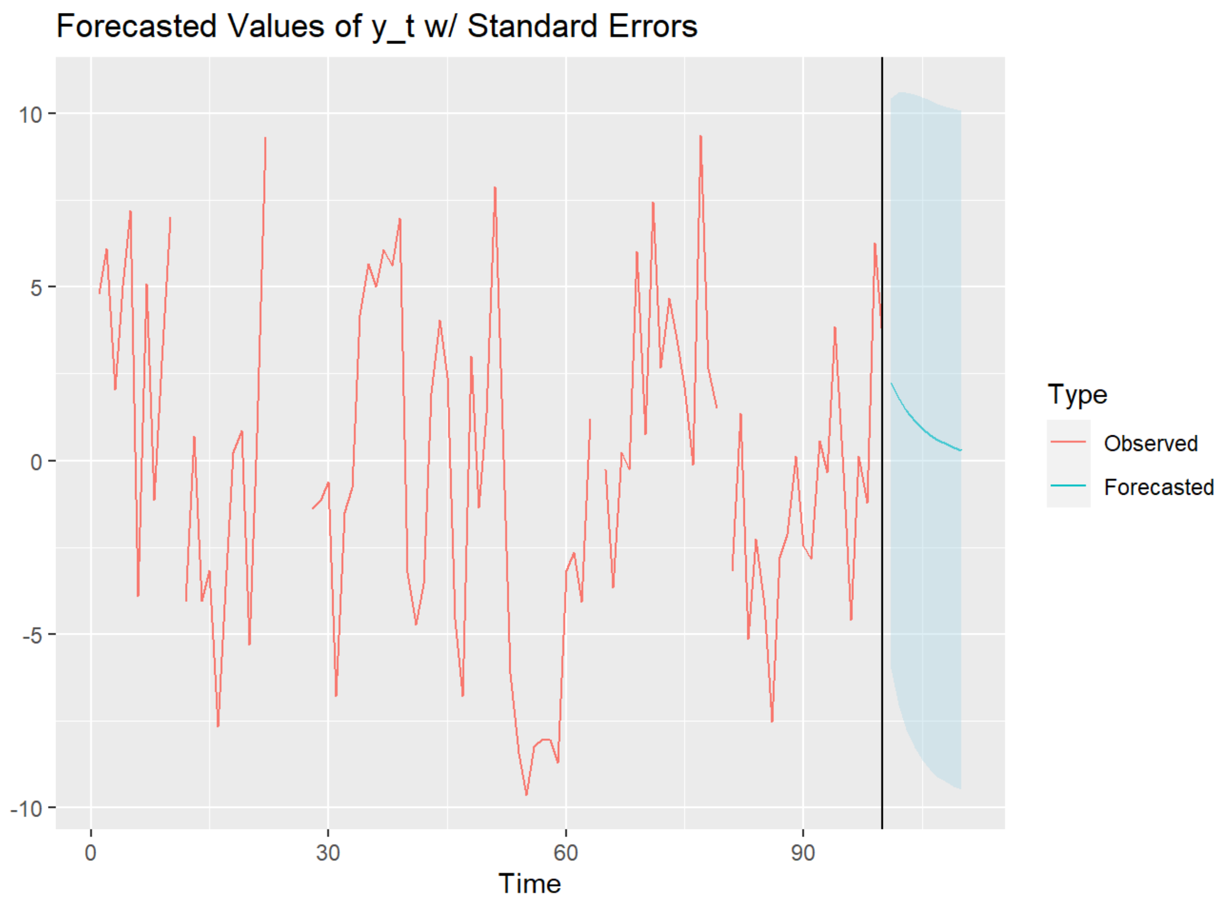
Question 2f.

f. Create a plot showing forecasted values (using the DLM forecasting methods discussed in lecture) of $y_{101:110}$ (including confidence bands), along with the original plot of $y_{1:100}$.

```
df_forecasted = data.frame(time=1:110, forecasted=c(rep(NA,100),dlm_forc), forecasted_SE=c(rep(NA,10
0),sqrt(unlist(dlm_forcvar))))

### Combine the data into a single dataframe with bind rows. Combine for plotting
combined_forecasted_df = bind_rows(
  data.frame(Time = df$time, Type = factor(rep("Observed", length(df$time)), levels = c("Observed",
"Forecasted")), x = as.numeric(df$yt)),
  data.frame(Time = df_forecasted$time, Type = factor(rep("Forecasted", length(df_forecasted$time)),
levels = c("Observed","Forecasted")), x = as.numeric(df_forecasted$forecasted)),
  )

ggplot(combined_forecasted_df, aes(x = Time)) +
    geom_line(aes(y = x, col = Type)) +
    # Per HW prompt, use 1.96 stdevs ON EACH SIDE of the mean for 95% CI bands
    geom_ribbon(data = df_forecasted,
                aes(x = time,
                    ymin = forecasted - 1.96*forecasted_SE,
                    ymax = forecasted + 1.96*forecasted_SE),
                 fill = "lightblue",
                 alpha = 0.4) +
    geom_vline(xintercept = 100) +
    labs(title = "Forecasted Values of y_t w/ Standard Errors") +
    ylab("")
```



Report the numerical values of Q_{101} and Q_{110} and provide a non-technical explanation for why the predictive variance of y_{101} is less than that y_{110}?

- Q_{101} = 17.569

- Q_{110} = 24.866

The predictive variance of y_{101} is less than that y_{110} because, similar to what we've seen in previous homework with time series, the further "into the future" we forecast the less confident we can be. As we've seen, "the obvious correlation introduced by the sampling of adjacent points in time can severely restrict the applicability of the many conventional statistical methods traditionally dependent on the assumption that these adjacent observations are independent and identically distributed."[1] The dependence on adjacent observations is the primary cause. Trying to predict the weather tomorrow will be more accurate than 10 days from now because there's more time for things to change between now and the longer time period.

Citations

[1]Shumway, R.H. and Stoffer, D.S. (2017). Time Series Analysis and Its Applications with R Examples (4th Ed.), Springer.