Homework 6

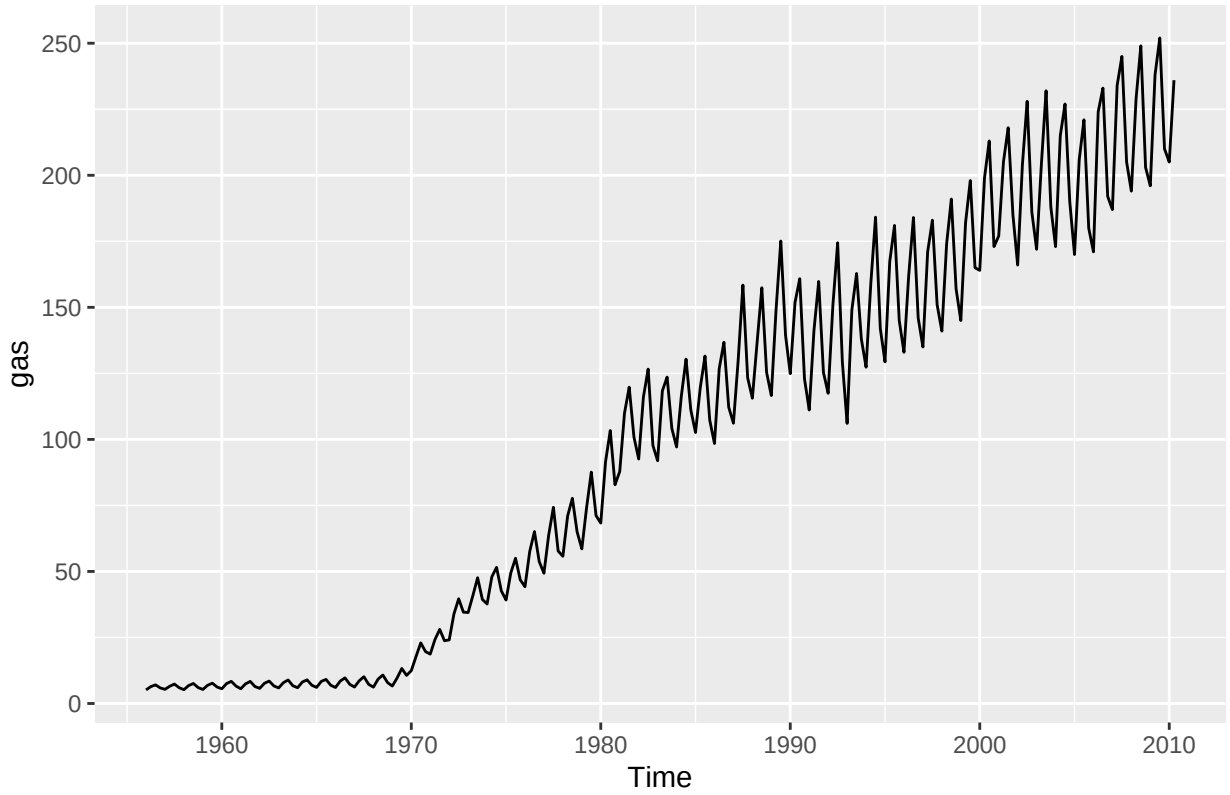
Joshua Ingram

10/14/2020

Problem 1

1.

```
autoplot(gas)
```

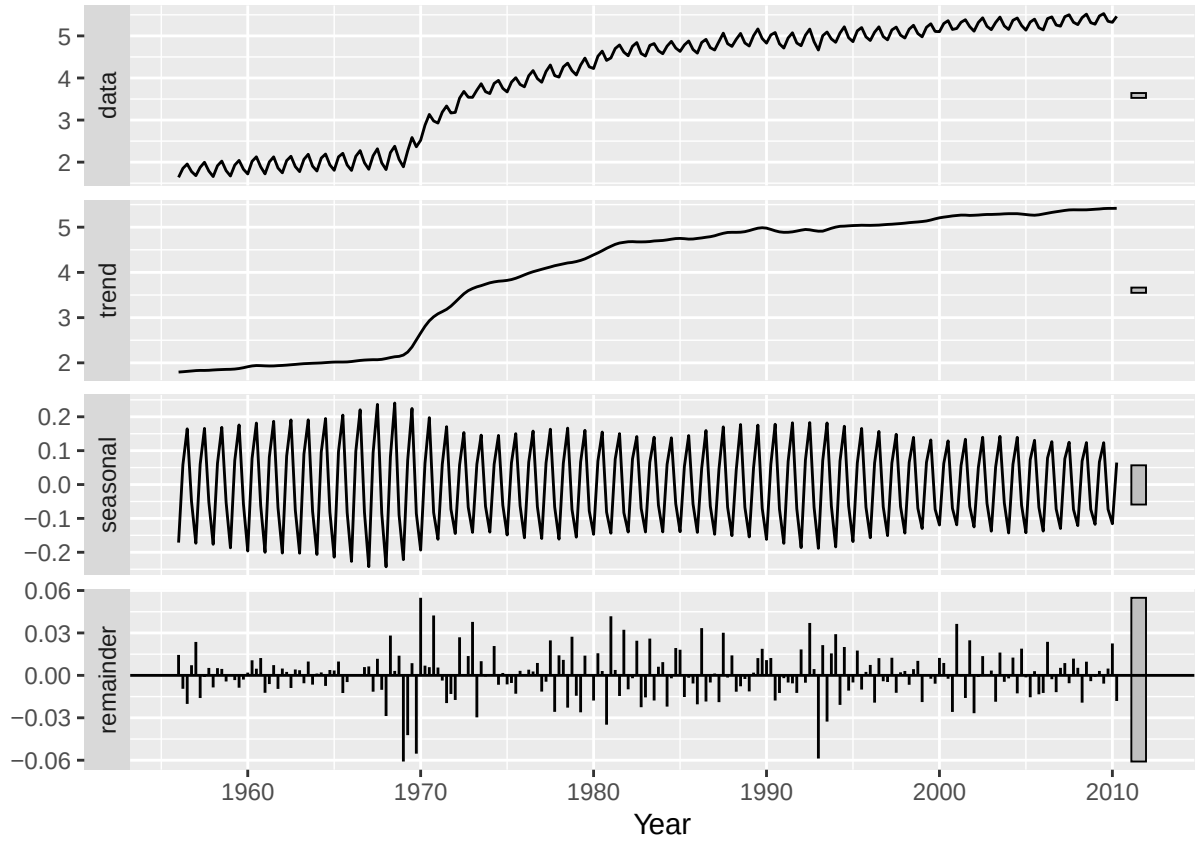


We observe a drastic increase in the uptrend of the production of gas just before 1970. The data is quarterly and there appears to be seasonality. As the production of gas increases over time, the variance of the production increases, resulting in an increase in the magnitude of the "swings" between quarterly production.

2.

There is not constant variance, so in order to use STL additive decomposition we need to log the gas variable.

```
stl_fit <- stl(log(gas), s.window = 7, t.window = 7)
autoplot(stl_fit, xlab = "Year")
```



In terms of log(gas production) - From 1956 to about 1969, there is a subtle but steady growth in the production of gas. After 1969, there is a sudden change in the rate of increase in the production of gas, as the uptrend is much stronger. There does not appear to be any cyclical nature to the data, with the only structural change to note occuring close to 1970. There is obvious seasonality in the data, with production in Q1 being lowest, followed by an increase in Q2 and Q3, then a drop in Q4. The seasonal component does appear to have different variance depending on the production level, even in log scale.

3.

The seasonal component in STL additive decomposition is calculated via "loess" or "local polynomial regression". This loess is fitted to all the sequence of Q1 values, then Q2, etc to get the fits for each quarter. After this, the seasonally adjusted time series is calculated from subtracting the seasonal component from the time series data. Next, the trend-cycle component is calculated using loess on the seasonally adjusted data.

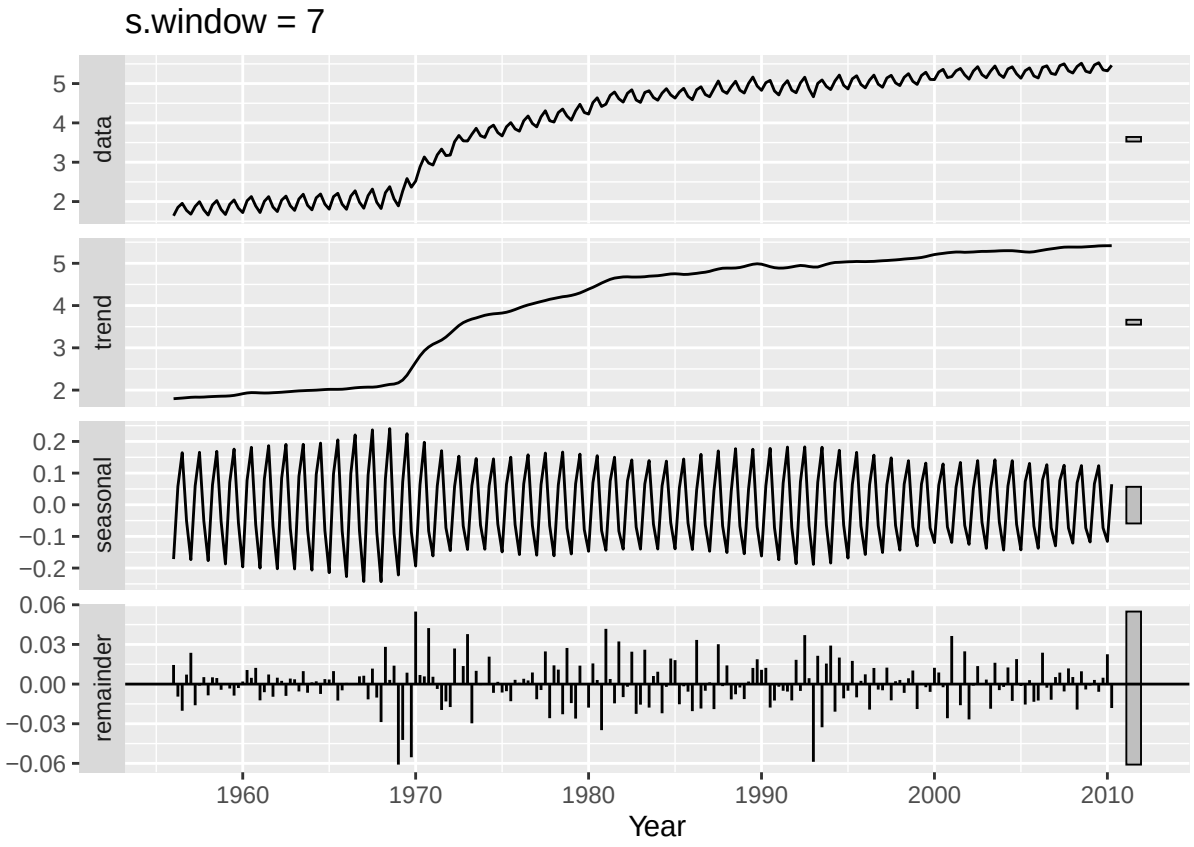
Finally, the updated seasonal component is calculated by subtracting the time series from the estimated seasonal component and the updated trend-cycle component is calculated by adding back the times series data. These steps are repeated several times to calculate the seasonal component (and trend-cycle and remainder).

4.

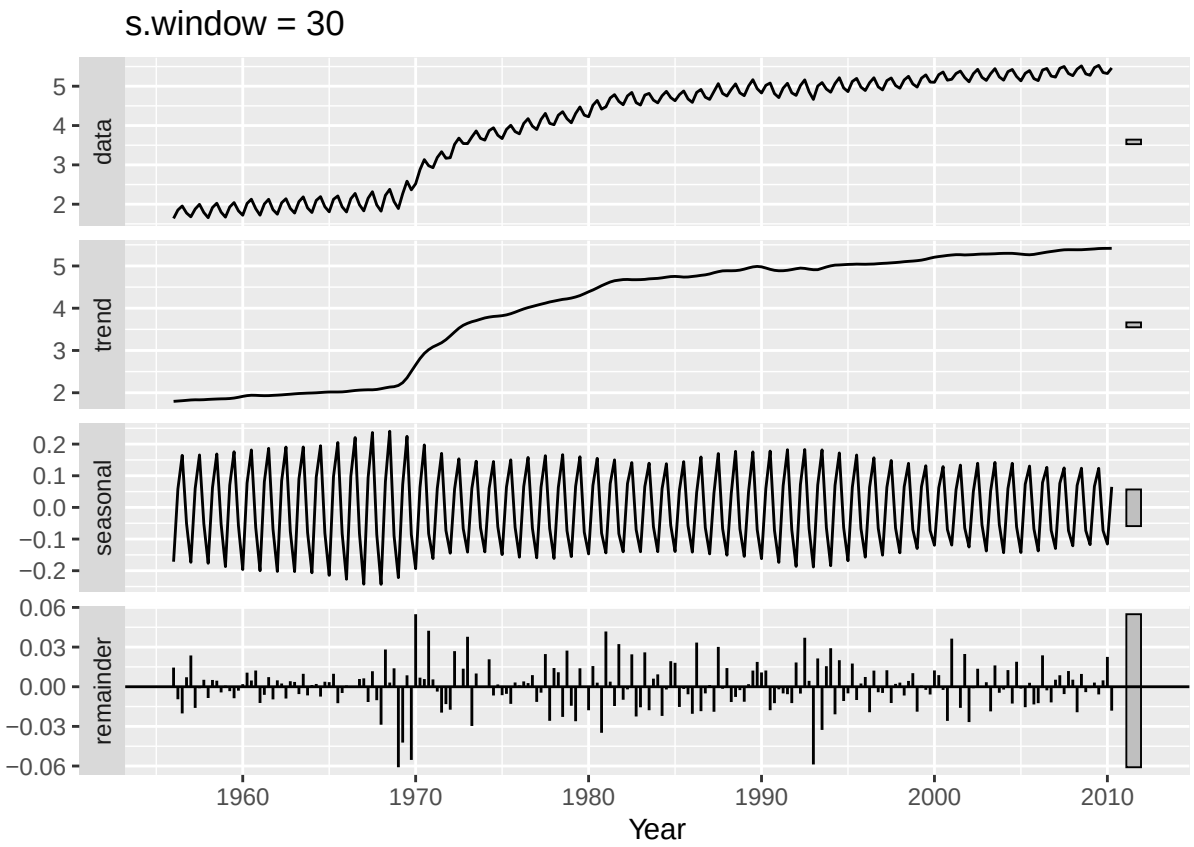
a.

Increasing the s.window value from 7 to 30 will decrease the variation in the swings in the seasonal component across time (aka increase the smoothness in seasonal component - Q1s will look more similar throughout all year with s.window = 30).

```
stl_fit_s7 <- stl(log(gas), s.window = 7, t.window = 7)
autoplot(stl_fit, xlab = "Year")  + ggtitle("s.window = 7")
```



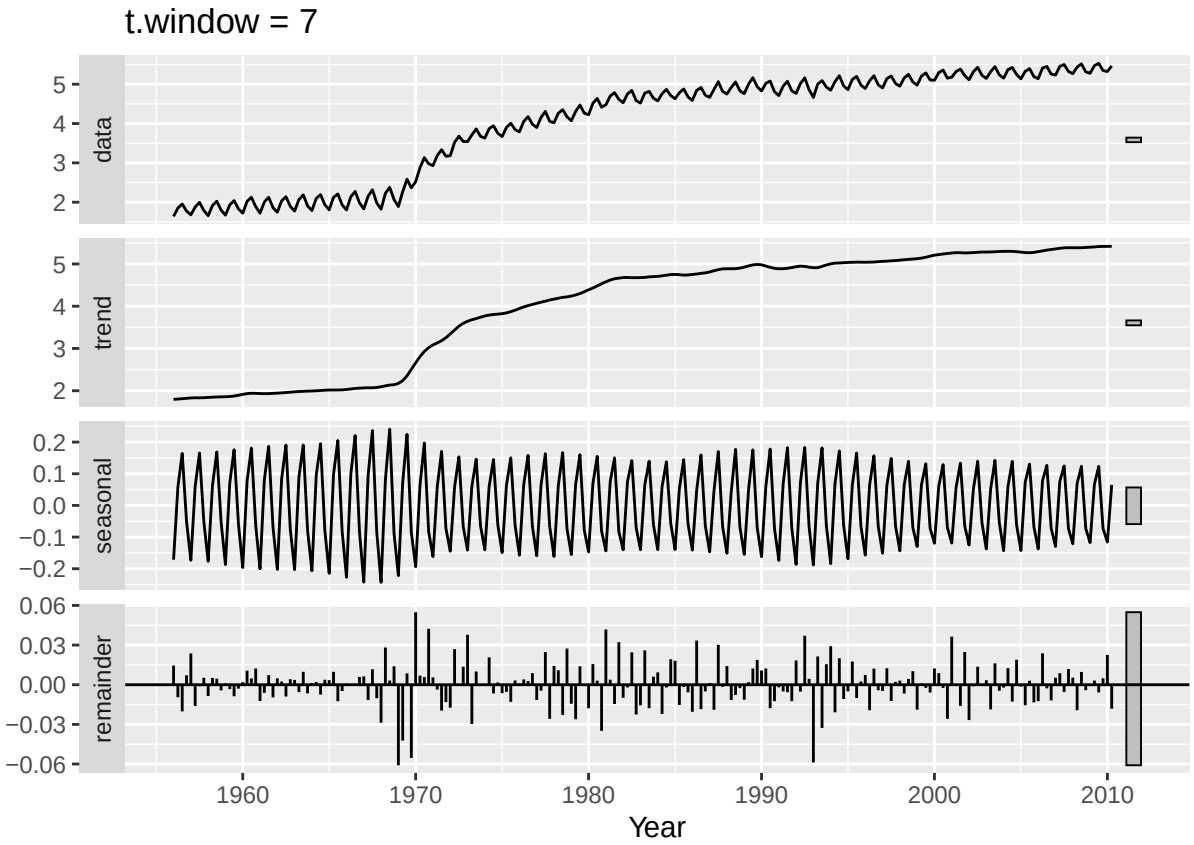

```
stl_fit_s30 <- stl(log(gas), s.window = 30, t.window = 7)
autoplot(stl_fit, xlab = "Year") + ggtitle("s.window = 30")
```



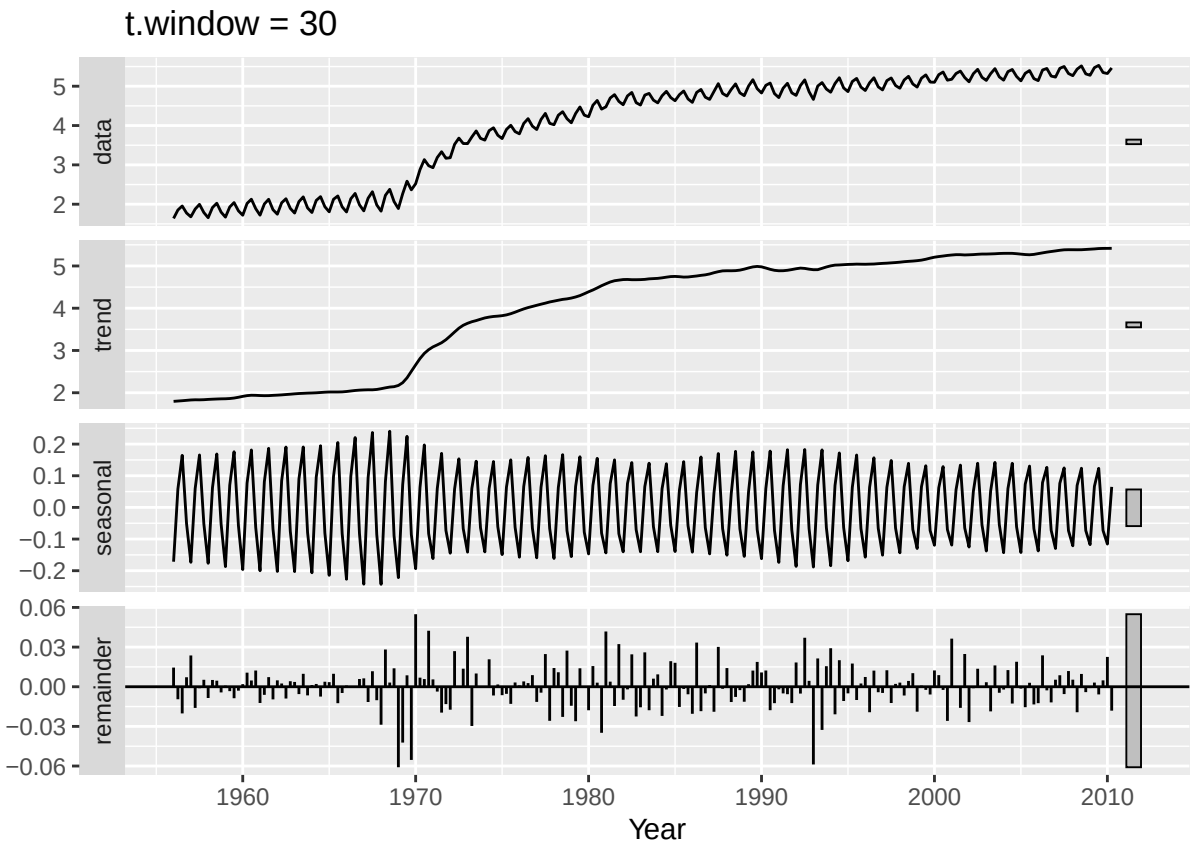
b.

Increasing the t.window value from 7 to 30 will increase the smoothness of the trend-cycle component.

```
stl_fit_t7 <- stl(log(gas), s.window = 7, t.window = 7)
autoplot(stl_fit, xlab = "Year")  + ggtitle("t.window = 7")
```

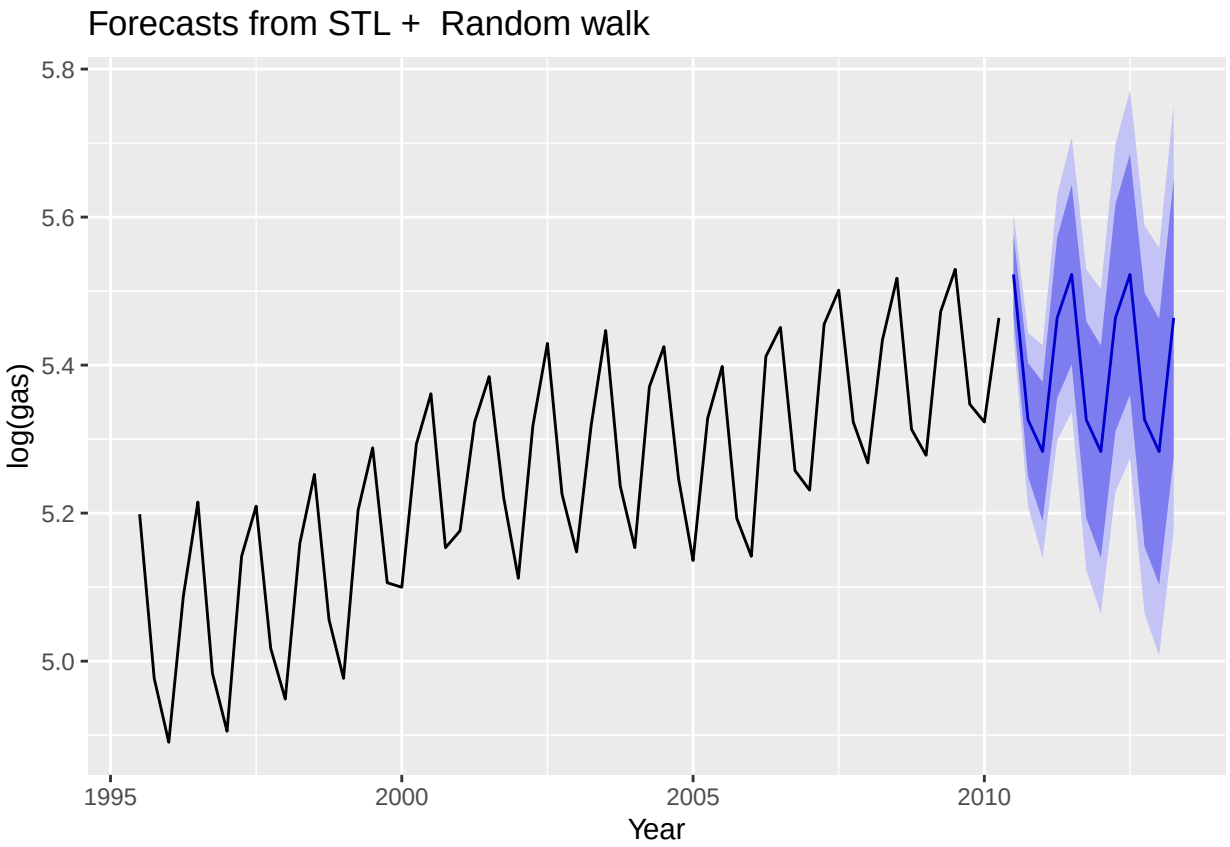



```
stl_fit_t30 <- stl(log(gas), s.window = 7, t.window = 30)
autoplot(stl_fit, xlab = "Year") + ggtitle("t.window = 30")
```



5.

```
stl_fit %>% forecast(method = "naive", h = 12) %>% autoplot(include = 60, xlab = "Year", title = "3-yea
```

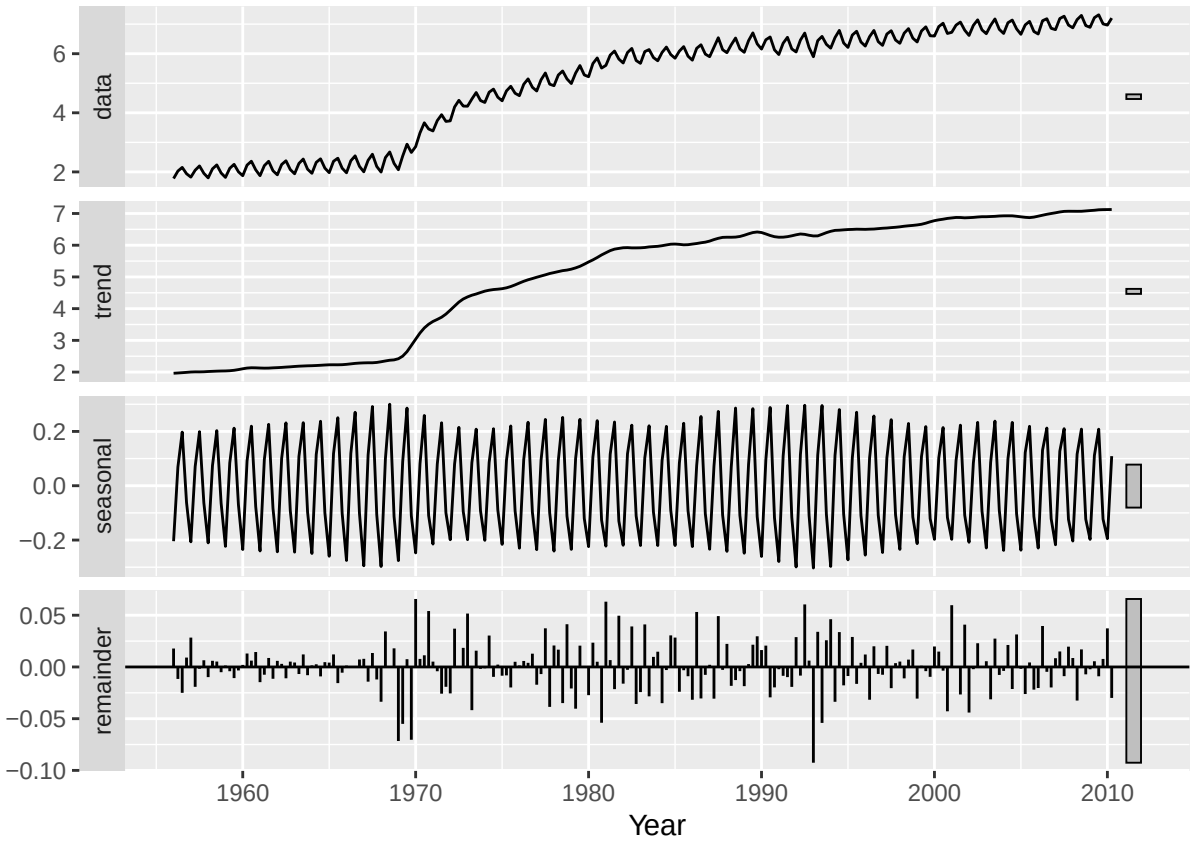


As we forecase further into the future, the prediction intervals get wider. These forecasts are on the log scale, not original scale.

Problem 2

1.

```
stl_fit_sqrt <- stl(BoxCox(gas,BoxCox.lambda(gas)), s.window = 7, t.window = 7)
lambda_optimal <- BoxCox.lambda(gas)
autoplot(stl_fit_sqrt, xlab = "Year")
```



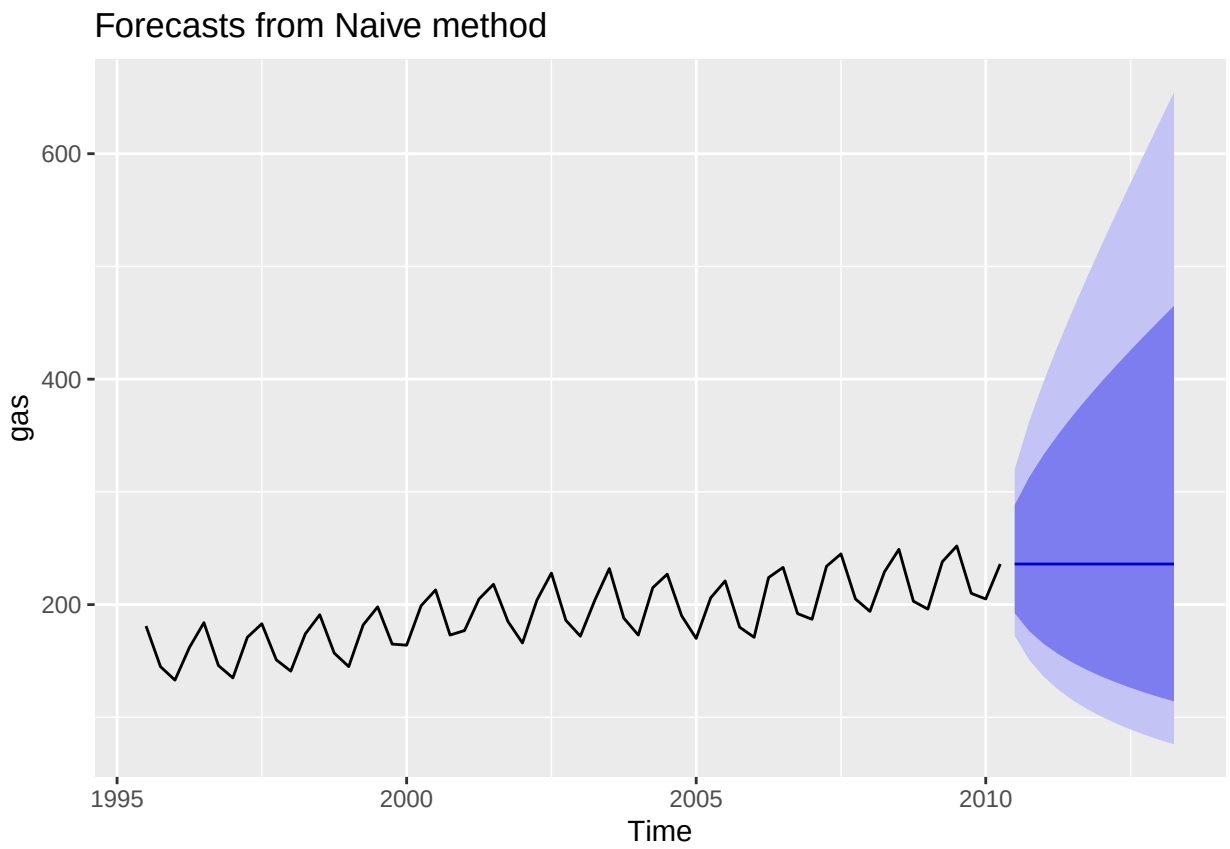
We can use the Box-Cox transformation. The BoxCox.lambda function determined that the optimal lambda that minimizes the change in variance is 0.09695486. Our seasonal component is much better than with the log transformation.

$$w_t(y_t) = \frac{y_t^{0.09695486} - 1}{0.09695486}$$

2.

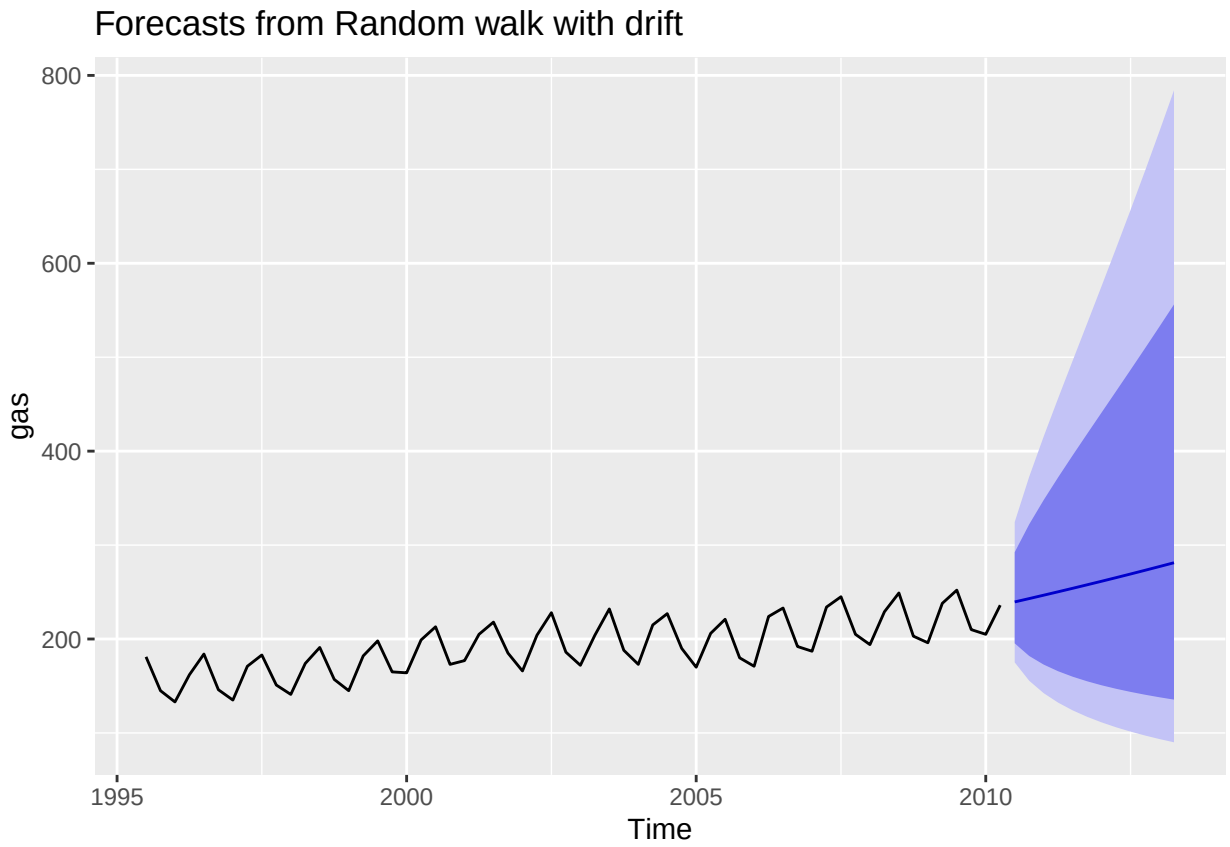
a.

```
lambda_optimal <- BoxCox.lambda(gas)
fit <- naive(gas, h = 12, lambda = lambda_optimal)
autoplot(fit, include = 60)
```



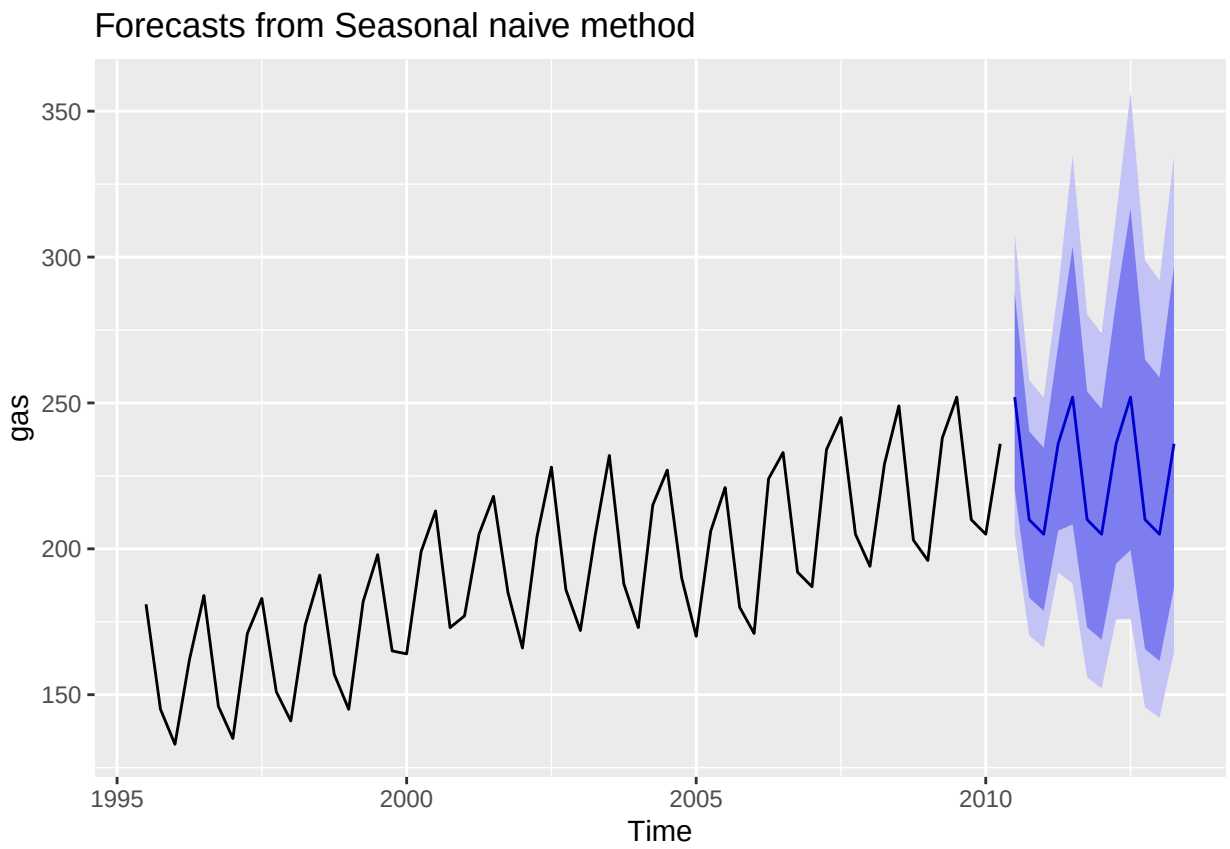
b.

```
lambda_optimal <- BoxCox.lambda(gas)
fit <- rwf(gas, h = 12, drift = TRUE, lambda = lambda_optimal)
autoplot(fit, include = 60)
```



c.

```
lambda_optimal <- BoxCox.lambda(gas)
fit <- snaive(gas, h = 12, lambda = lambda_optimal)
autoplot(fit, include = 60)
```



The naive method is the least resonable in appearance, as it forecasts a constant value with wide prediction intervals. The seasonal naive method is better, as it takes into account the seasonal component, but does not include any trend component. The drift method seems to be the best, as it forecast a trend in the data, but it does lose out on giving seasonal component predictions.

Problem 3

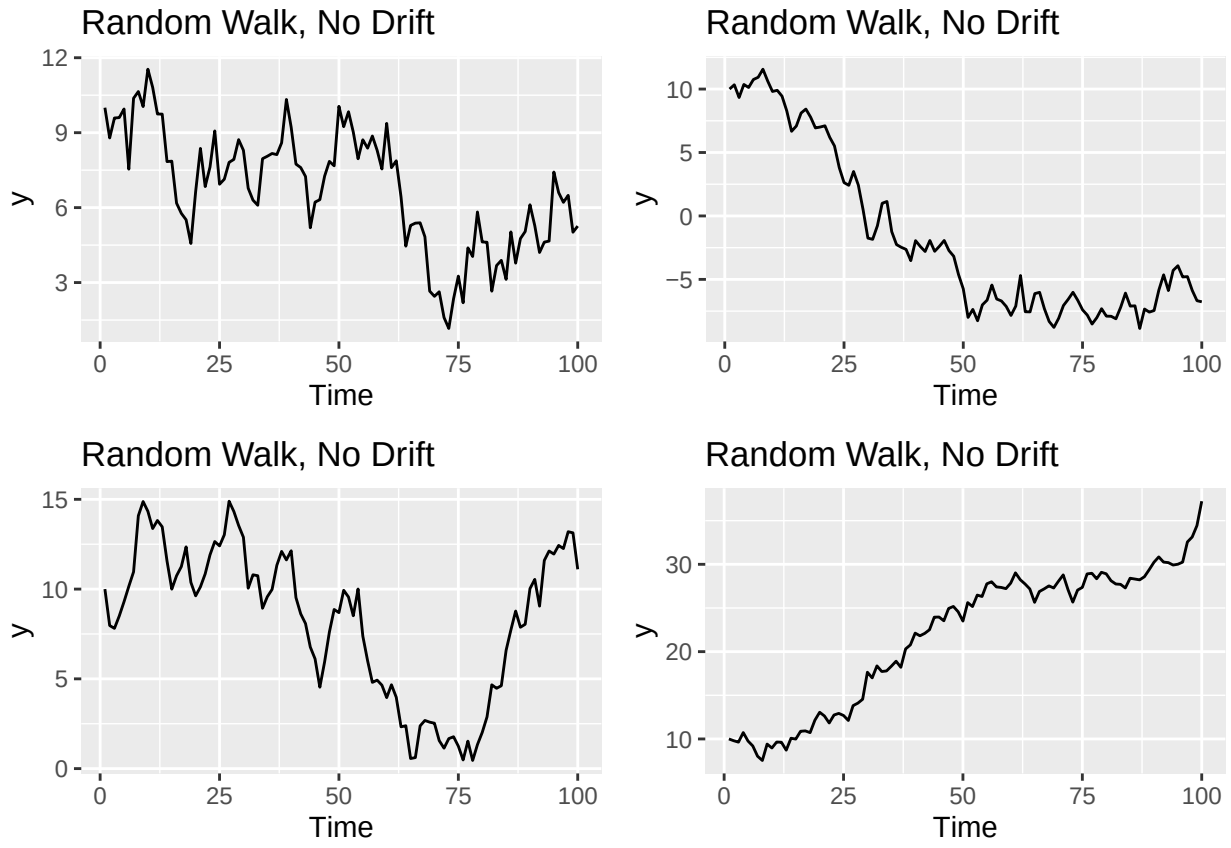
1.

a.

```
df_nodrift <- data.frame(1, 10, 10, 10, 10)
colnames(df_nodrift) <- c("Time", "y1", "y2", "y3", "y4")
for (i in seq(2, 100)){
  df_nodrift[i,] <- c(i,
                      df_nodrift[i-1, 2] + rnorm(1,0,1),
                      df_nodrift[i-1, 3] + rnorm(1,0,1),
                      df_nodrift[i-1, 4] + rnorm(1,0,1),
                      df_nodrift[i-1, 5] + rnorm(1,0,1))
}

plot1 <- ggplot(data = df_nodrift, aes(x = Time, y = y1)) + geom_line() + labs(title = "Random Walk, No
plot2 <- ggplot(data = df_nodrift, aes(x = Time, y = y2)) + geom_line() + labs(title = "Random Walk, No
```

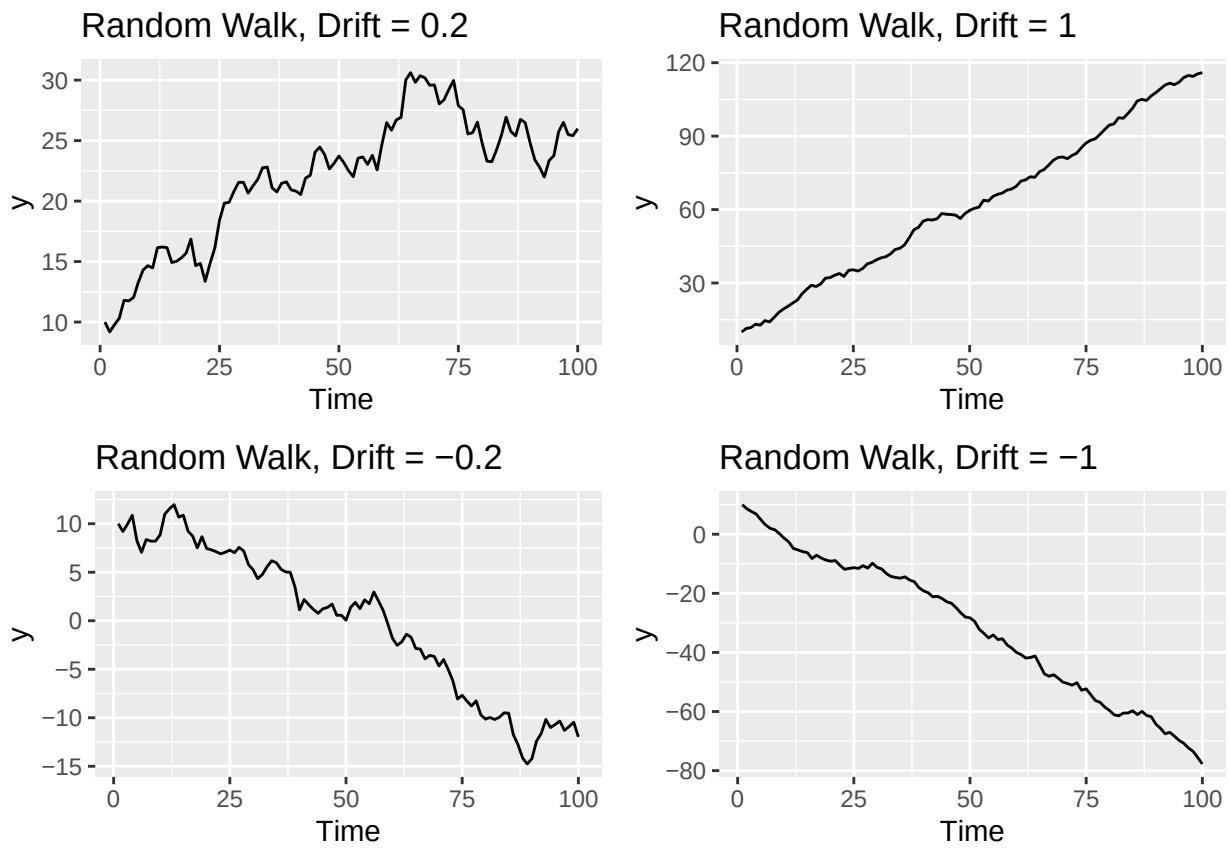
```
plot3 <- ggplot(data = df_nodrift, aes(x = Time, y = y3)) + geom_line() + labs(title = "Random Walk, No
plot4 <- ggplot(data = df_nodrift, aes(x = Time, y = y4)) + geom_line() + labs(title = "Random Walk, No
grid.arrange(plot1, plot2, plot3, plot4, ncol = 2)
```



b.

```
df_nodrift <- data.frame(1, 10, 10, 10, 10)
colnames(df_nodrift) <- c("Time", "y1", "y2", "y3", "y4")
for (i in seq(2, 100)){
  df_nodrift[i,] <- c(i,
                      df_nodrift[i-1, 2] + 0.2 + rnorm(1,0,1),
                      df_nodrift[i-1, 3] + 1 + rnorm(1,0,1),
                      df_nodrift[i-1, 4] - 0.2 + rnorm(1,0,1),
                      df_nodrift[i-1, 5] - 1 + rnorm(1,0,1))
}

plot1 <- ggplot(data = df_nodrift, aes(x = Time, y = y1)) + geom_line() + labs(title = "Random Walk, Dri
plot2 <- ggplot(data = df_nodrift, aes(x = Time, y = y2)) + geom_line() + labs(title = "Random Walk, Dri
plot3 <- ggplot(data = df_nodrift, aes(x = Time, y = y3)) + geom_line() + labs(title = "Random Walk, Dri
plot4 <- ggplot(data = df_nodrift, aes(x = Time, y = y4)) + geom_line() + labs(title = "Random Walk, Dri
grid.arrange(plot1, plot2, plot3, plot4, ncol = 2)
```



2.

$$

$$

3.

Show that

$$\hat{y}_{T+h|T} = Y_T + h \times \frac{\sum_{t=2}^{T} y_t - y_{t-1}}{T-1} = Y_T + h \times \frac{y_T - y_1}{T-1}$$

Given

$$\hat{y}_{T+h|T} = Y_T + h \times \frac{\sum_{t=2}^{T} y_t - y_{t-1}}{T-1}$$

Let's look at just the numerator of the fraction

$$\sum_{t=2}^{T} y_t - y_{t-1}$$

$$= (y_2 - y_1) + (y_3 - y_2) + ... + (y_{T-1} - y_{T-2}) + (y_T - y_{T-1})$$

Using the associative property of addition, we can rearrange our values

$$= -y_1 + (y_2 - y_2) + (y_3 - y_3) + ... + (y_{T-1} - y_{T-1}) + y_T = y_T - y_1$$

Thus

$$\sum_{t=2}^{T} y_t - y_{t-1} = y_T - y_1$$

Going back to the original formula, we can replace the original numerator with the above values since they are equivalent. Resulting in

$$Y_T + h \ \times \frac{\sum_{t=2}^{T} y_t - y_{t-1}}{T-1} = Y_T + h \ \times \frac{y_T - y_1}{T-1}$$

Thus

$$\hat{y}_{T+h|T} = Y_T + h \ \times \frac{y_T - y_1}{T-1}$$