HW 1

Rahul Sangole

April 14, 2018

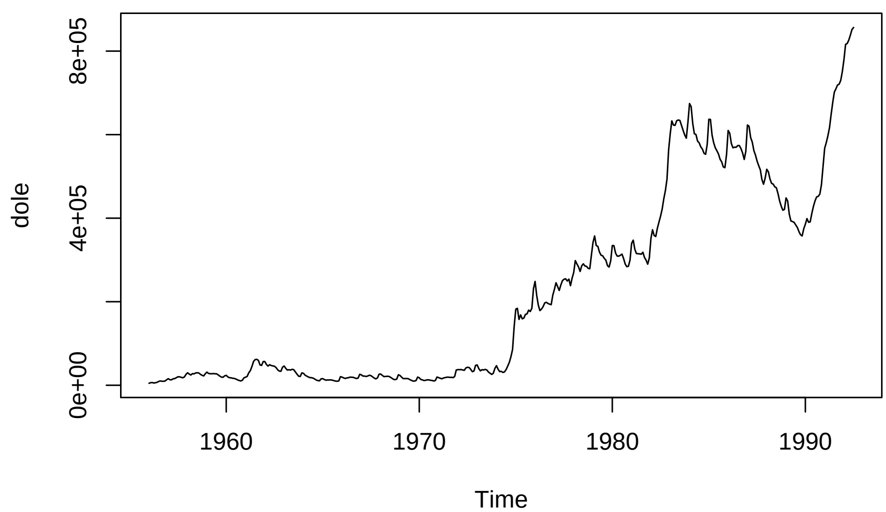
Contents

Secion 2.8

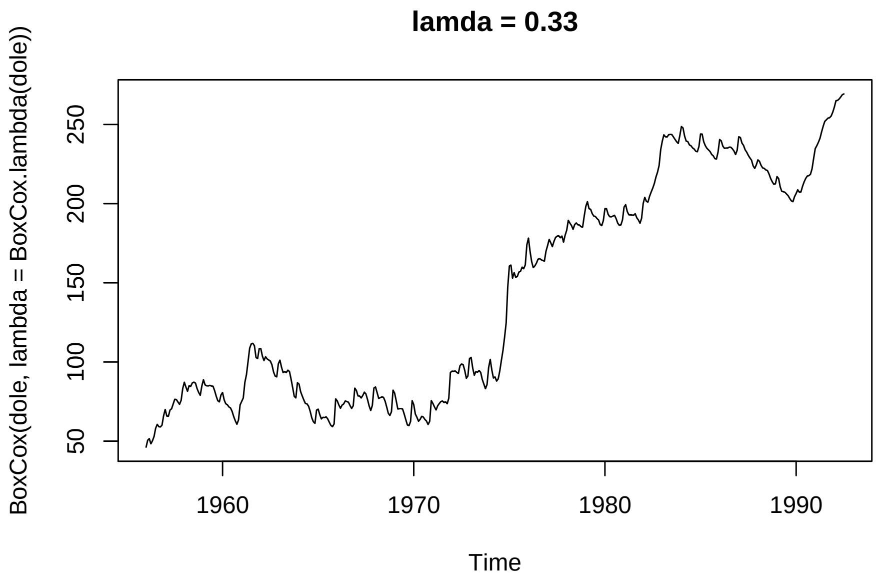
For each of the following series (from the fma package), make a graph of the data. If transforming seems appropriate, do so and describe the effect.

Monthly total of people on unemployed benefits in Australia (January 1956–July 1992).

```
plot(dole)
```

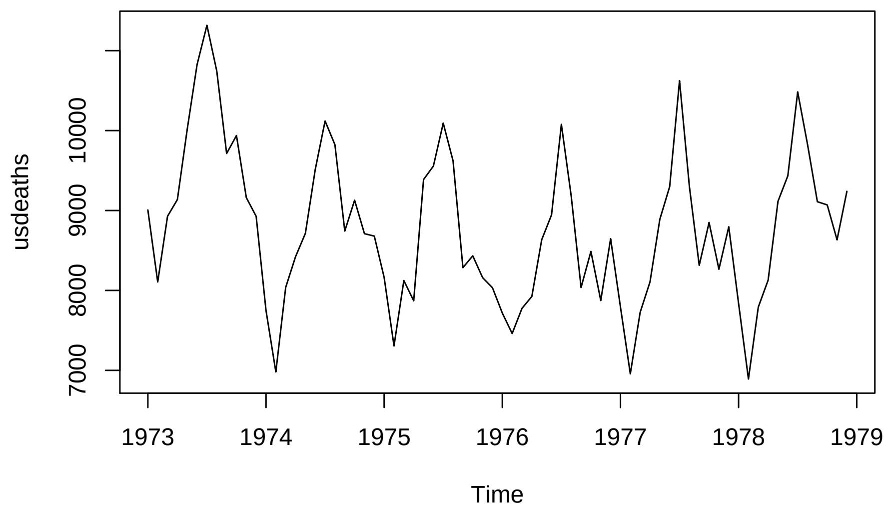



```
plot(BoxCox(dole,lambda = BoxCox.lambda(dole)),
     main=paste0('lamda = ',round(BoxCox.lambda(dole),2)))
```

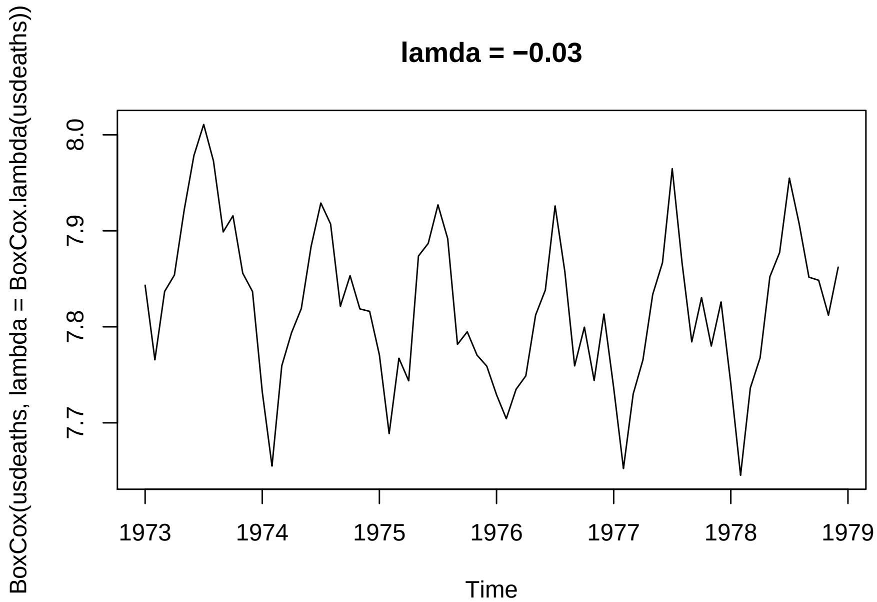


Monthly total of accidental deaths in the United States (January 1973–December 1978).

```r
plot(usdeaths)
```

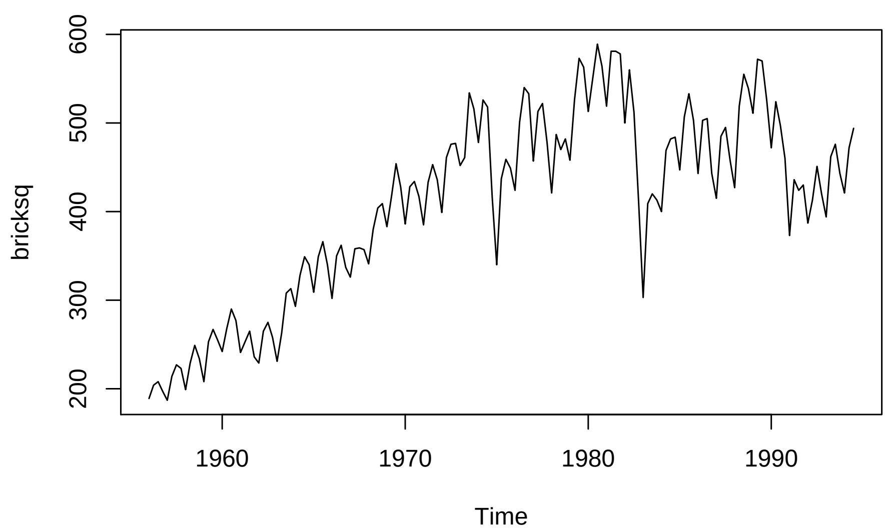



```r
plot(BoxCox(usdeaths,lambda = BoxCox.lambda(usdeaths)),
     main=paste0('lamda = ',round(BoxCox.lambda(usdeaths),2)))
```

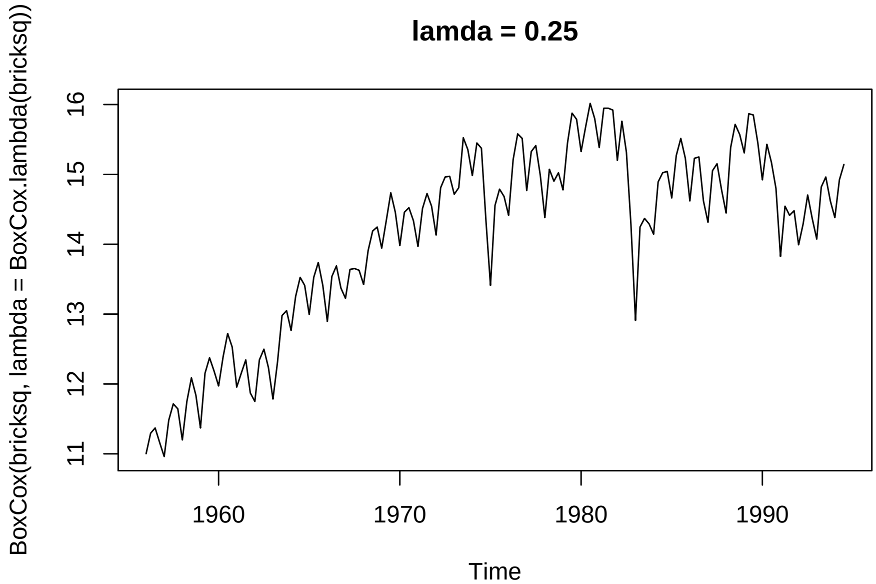


Quarterly production of bricks (in millions of units) at Portland, Australia (March 1956–September 1994).

```
plot(bricksq)
```



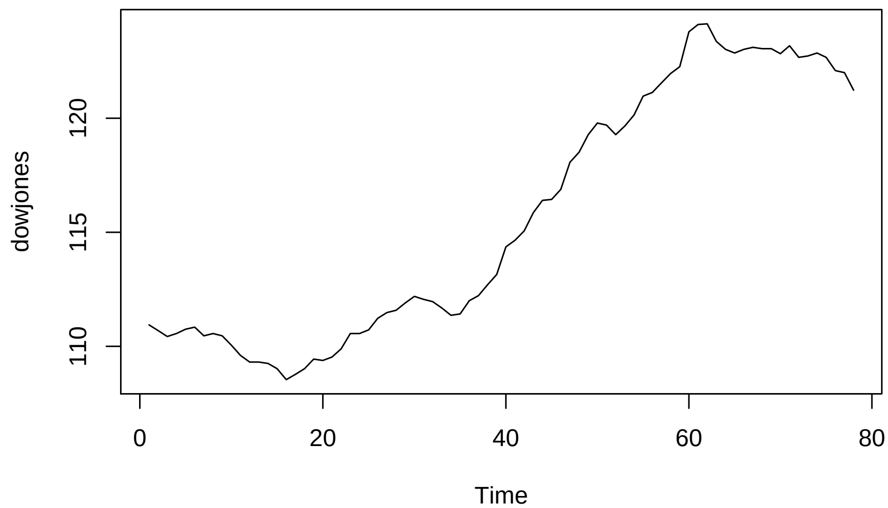

```
plot(BoxCox(bricksq,lambda = BoxCox.lambda(bricksq)),
     main=paste0('lamda = ',round(BoxCox.lambda(bricksq),2)))
```



Use the Dow Jones index (data set dowjones) to do the following:

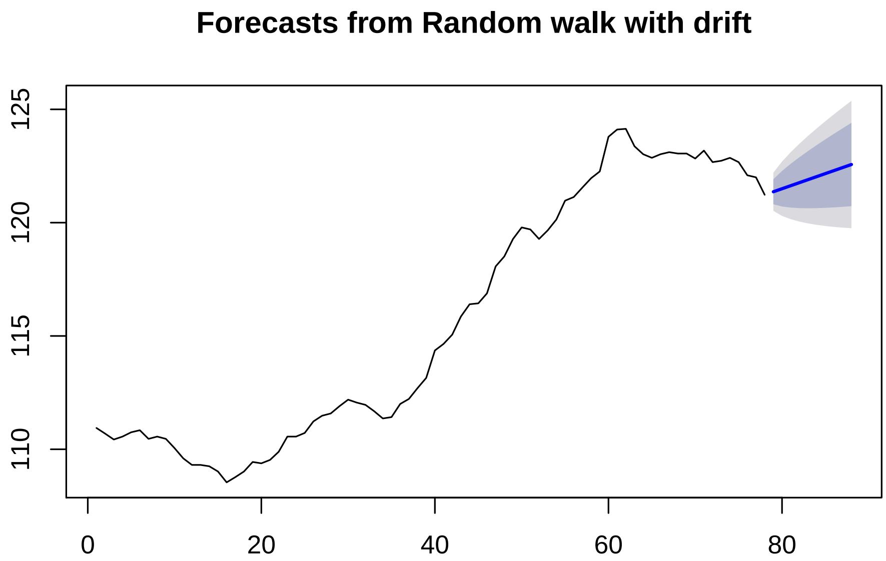
Produce a time plot of the series.

```
plot(dowjones)
```



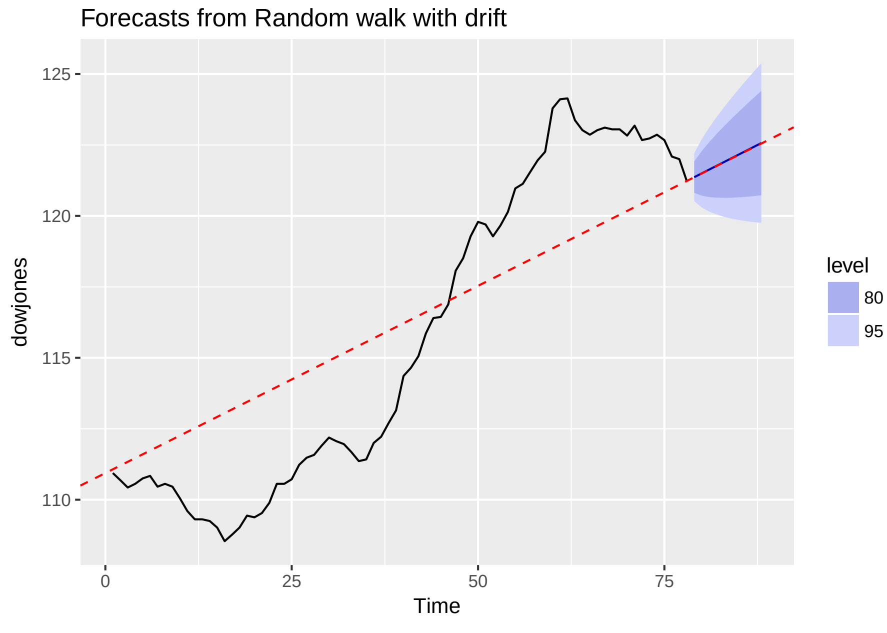
Produce forecasts using the drift method and plot them.

```r
plot(rwf(dowjones,h=10, drift = T))
```



Show that the graphed forecasts are identical to extending the line drawn between the first and last observations.

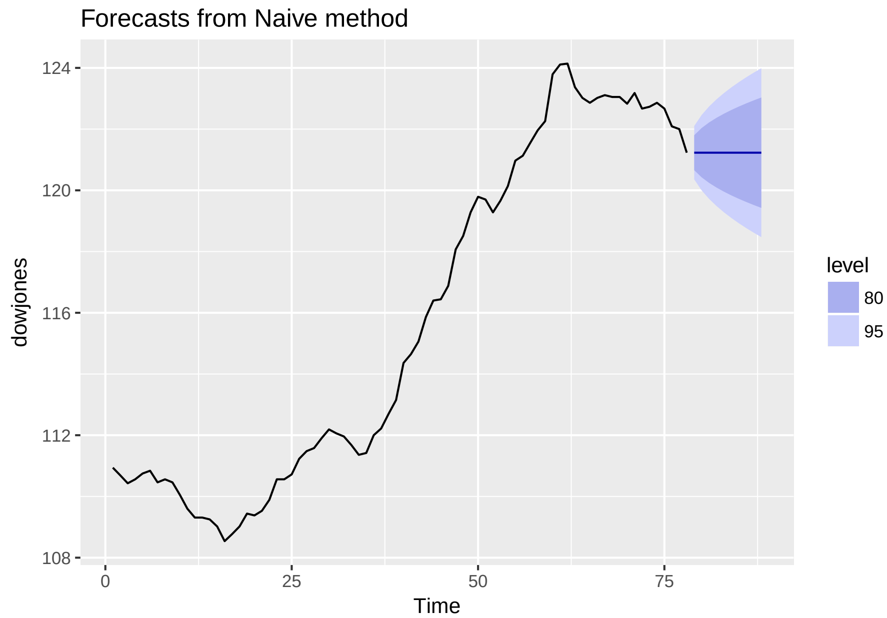
```r
ggplot2::autoplot(rwf(dowjones,h=10, drift = T))+
    geom_abline(intercept = dowjones[1], slope = (dowjones[78] - dowjones[1])/78,
                color = 'red', lty=2)
```



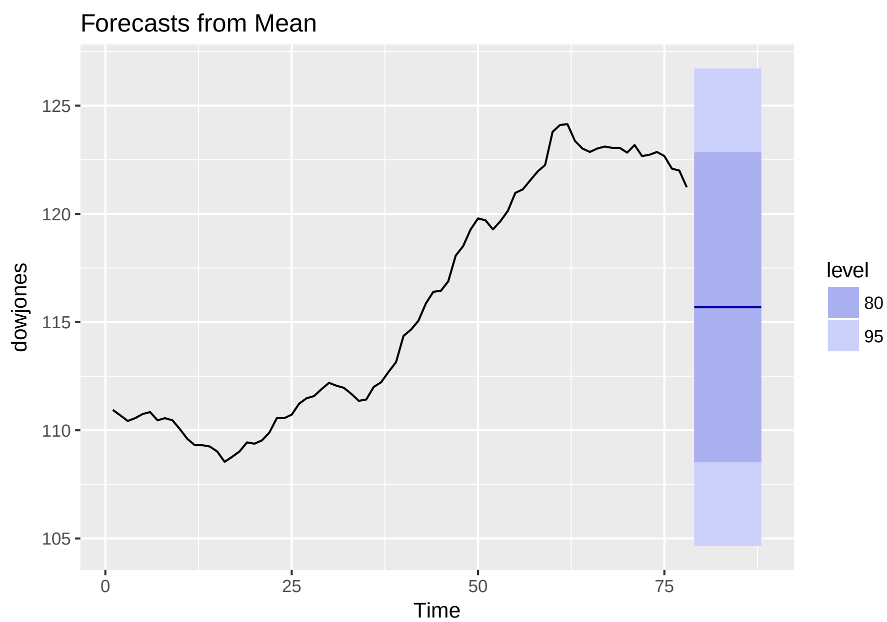
Try some of the other benchmark functions to forecast the same data set. Which do you think is best? Why?

Perhaps the naive is a bit better than the drift model since atleast it doesn't predict the signal is going to increase! But, all of these are quite bad.

```
ggplot2::autoplot(naive(dowjones, h=10))
```



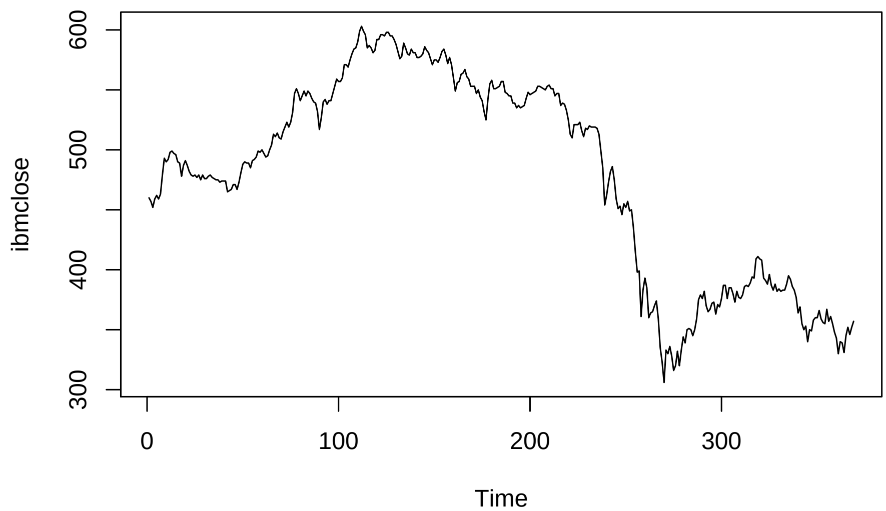

```
ggplot2::autoplot(meanf(dowjones, h=10))
```



Consider the daily closing IBM stock prices (data set ibmclose).

Produce some plots of the data in order to become familiar with it.

```
plot(ibmclose)
```



Split the data into a training set of 300 observations and a test set of 69 observations.

```r
train <- window(ibmclose, end = 300)
test <- window(ibmclose, start = 301)
```

Try various benchmark methods to forecast the training set and compare the results on the test set. Which method did best?

```r
nv <- naive(train)
snv <- snaive(train)
mf <- meanf(train)
drft <- rwf(train,drift = T)
forecast(nv, test) %>% accuracy
```

```
##                      ME     RMSE     MAE         MPE     MAPE MASE
## Training set -0.2809365 7.302815 5.09699 -0.08262872 1.115844    1
##                   ACF1
## Training set 0.1351052
```

```r
forecast(snv, test) %>% accuracy
```

```
##                      ME     RMSE     MAE         MPE     MAPE MASE
## Training set -0.2809365 7.302815 5.09699 -0.08262872 1.115844    1
##                   ACF1
## Training set 0.1351052
```

```r
forecast(mf, test) %>% accuracy
```

```
##                        ME     RMSE      MAE       MPE     MAPE     MASE
## Training set 1.660438e-14 73.61532 58.72231 -2.642058 13.03019 11.52098
##                   ACF1
## Training set 0.9895779
```

```r
forecast(drft, test) %>% accuracy
```

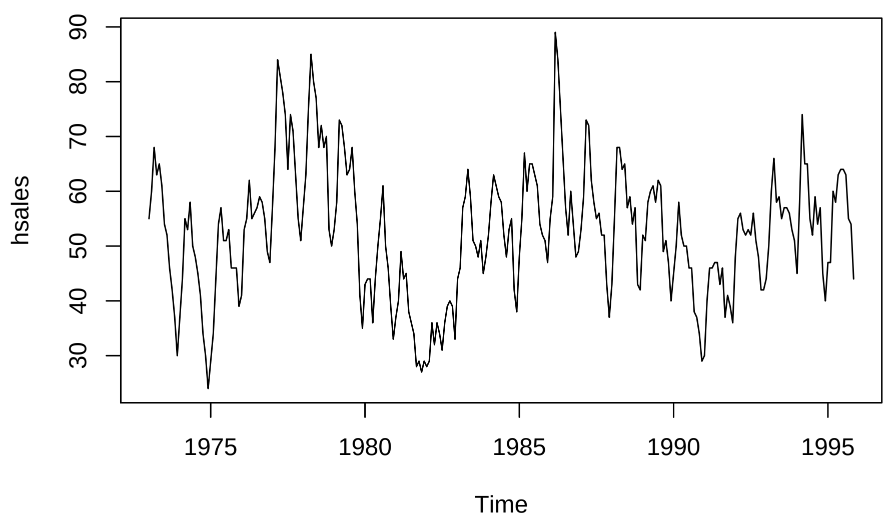
```
##                          ME     RMSE      MAE         MPE    MAPE     MASE
## Training set -3.916293e-14 7.297409 5.127996 -0.02530123 1.12165 1.006083
##                   ACF1
## Training set 0.1351052
```

Looks like (using RMSE, MPE) the drift method seems (marginally) the better one.

Consider the sales of new one-family houses in the USA, Jan 1973 – Nov 1995 (data set hsales).

Produce some plots of the data in order to become familiar with it.

```r
plot(hsales)
```



Split the hsales data set into a training set and a test set, where the test set is the last two years of data.

```r
train <- window(hsales, end = 1992)
test <- window(hsales, start = 1993)
```

Try various benchmark methods to forecast the training set and compare the results on the test set. Which method did best?

```r
nv <- naive(train)
snv <- snaive(train)
mf <- meanf(train)
drft <- rwf(train,drift = T)
forecast(nv, test) %>% accuracy
```

```
##                        ME    RMSE      MAE        MPE    MAPE      MASE
## Training set -0.03070175 6.45735 5.153509 -0.8533137 10.2482 0.5983475
```

```
##                     ACF1
## Training set 0.1730619
```

```r
forecast(snv, test) %>% accuracy
```

```
##                      ME     RMSE      MAE       MPE     MAPE MASE    ACF1
## Training set -0.5391705 10.80387 8.612903 -3.596793 18.05441    1 0.83344
```

```r
forecast(mf, test) %>% accuracy
```

```
##                        ME     RMSE      MAE       MPE     MAPE     MASE
## Training set 9.901644e-16 12.60327 10.00847 -6.596578 21.44585 1.162032
##                     ACF1
## Training set 0.8689799
```

```r
forecast(drft, test) %>% accuracy
```

```
##                         ME     RMSE      MAE        MPE    MAPE      MASE
## Training set -3.812934e-13 6.457277 5.152701 -0.7901168 10.2439 0.5982537
##                     ACF1
## Training set 0.1730619
```

Using MASE, we can see that the naive or drift forecasts do fairly better than the others in this case.

Section 4.10

1. Electricity consumption was recorded for a small town on 12 randomly chosen days.

Here's what the electricity consumption data looks like:

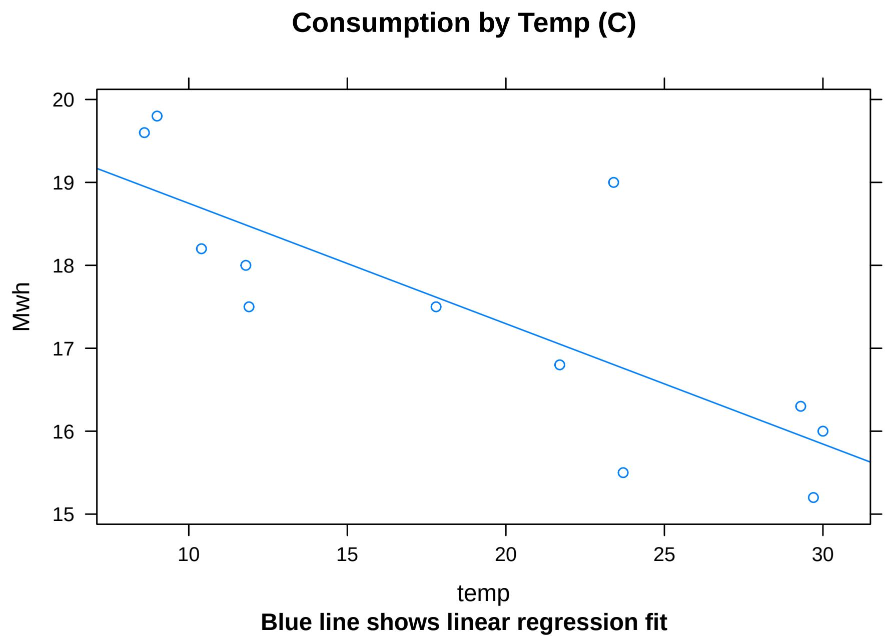
```r
head(econsumption)
```

Mwh	temp
16.3	29.3
16.8	21.7
15.5	23.7
18.2	10.4
15.2	29.7
17.5	11.9

a. Plot the data and find the regression model for Mwh with temperature as an explanatory variable. Why is there a negative relationship?

Here's a plot of consumption by temperature.

```r
lattice::xyplot(Mwh~temp, econsumption, type=c('p','r'),
                main = 'Consumption by Temp (C)',
                sub='Blue line shows linear regression fit')
```



Building a linear model:

```
fit <- lm(Mwh~temp, econsumption)
summary(fit)
```

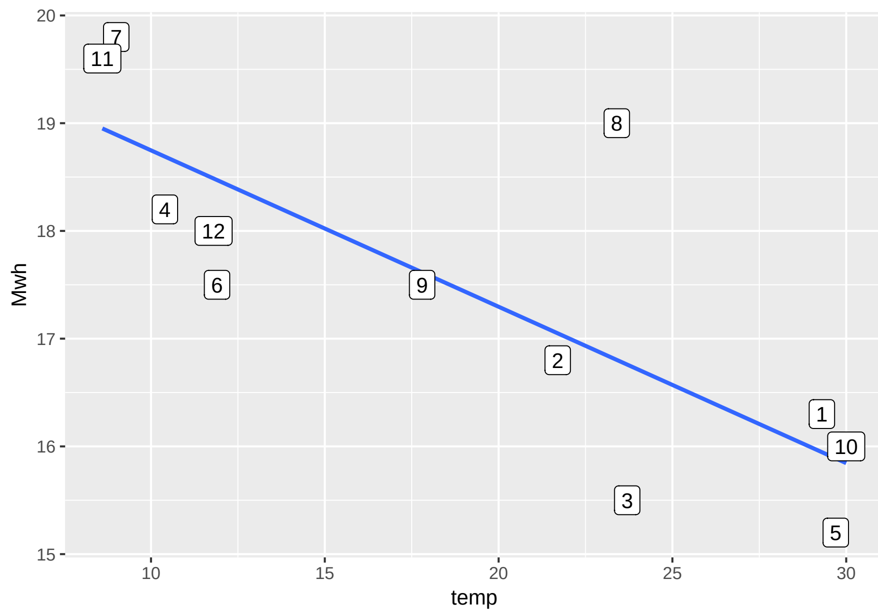
```
##
## Call:
## lm(formula = Mwh ~ temp, data = econsumption)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.2593 -0.5395 -0.1827  0.4274  2.1972
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 20.19952    0.73040   27.66 8.86e-11 ***
## temp        -0.14516    0.03549   -4.09  0.00218 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9888 on 10 degrees of freedom
## Multiple R-squared:  0.6258, Adjusted R-squared:  0.5884
## F-statistic: 16.73 on 1 and 10 DF,  p-value: 0.00218
```

Both the coefficients are statistically significant at the 0.01 level, with a negative slope for temperature. The temperature range in question is between 10C and 30C, which would indicate that the increase in electric consumption as temperature reduces is due to usage of electric heaters in households. Thus the slope is negative.

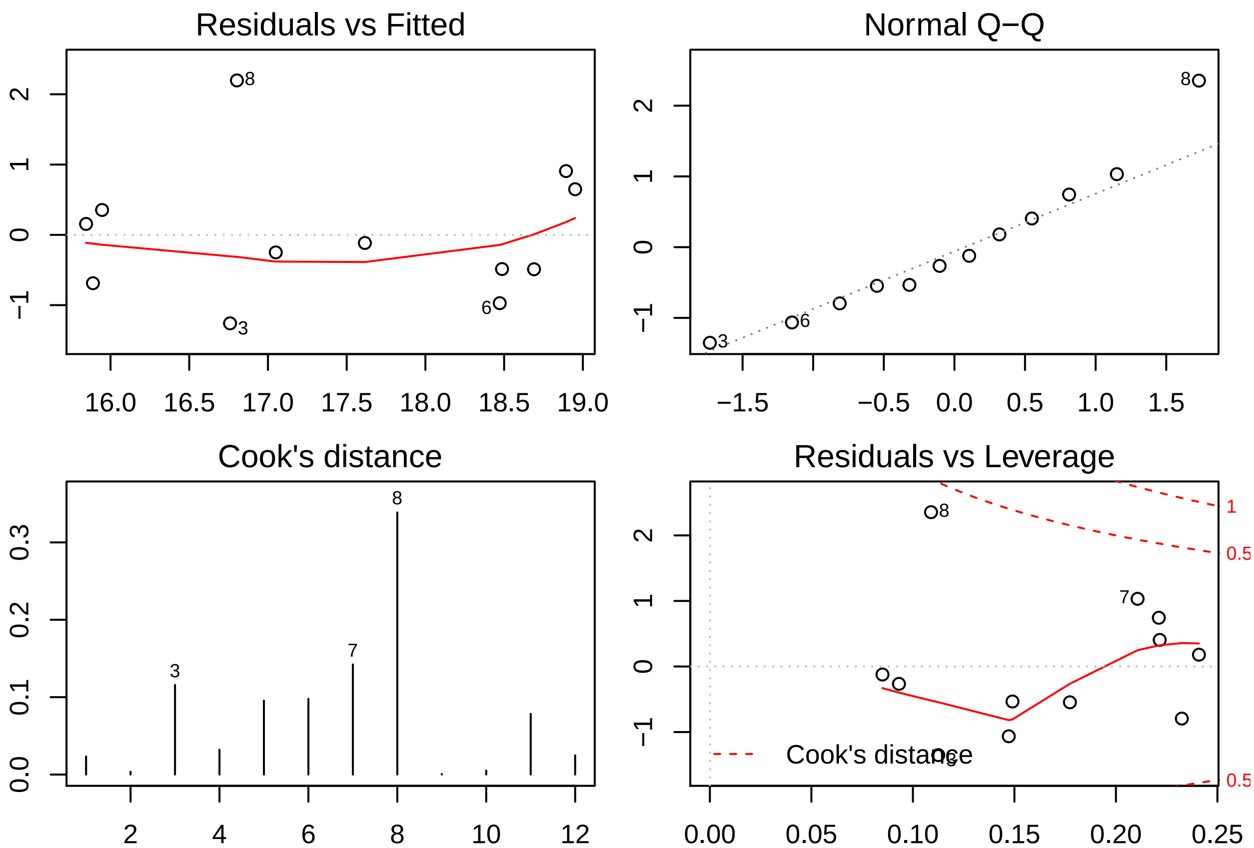
b. Produce a residual plot. Is the model adequate? Are there any outliers or influential observations?

The model seems OK. There are two main outliers - 3 and 8, which show up with higher residuals, high Cook's distance. Pt 7 also has high Cook's distance. Though, if we look at the leverage statistics, we see that neither 3 nor 8 have high leverage, i.e. they don't really pull the regression line one way or the other. Pt 7, on the other hand, is at the left end of the regression line and has a higher leverage.

```
econsumption$id <- 1:12
ggplot(econsumption, aes(x=temp,y=Mwh))+geom_smooth(se = F,method = 'lm') +
    geom_label(aes(label=id))
```



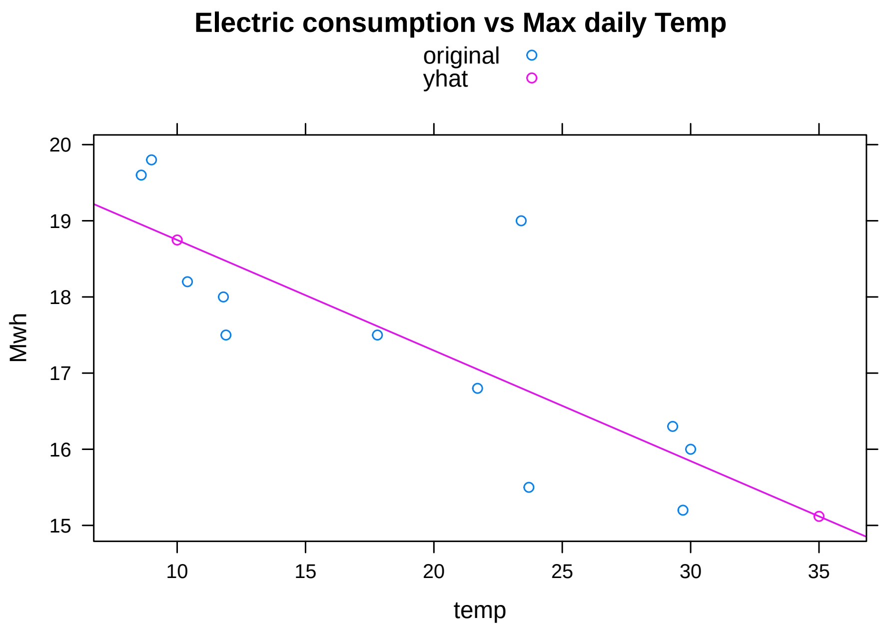

```
old.par <- par(); par(mfrow=c(2,2), mar=c(2,2,2,1))
plot(fit, which = c(1,2,4,5)) ;par(old.par)
```



c. Use the model to predict the electricity consumption that you would expect for a day with maximum temperature 10C and a day with maximum temperature 35C. Do you believe these predictions?

- The prediction at 10C does make sense. It falls within the cluster of points in that region and seems like a useful estimate.
- The prediction at 35C extrapolates the data and is more suspect. It's possible that as it get's warmer, the air conditioning load would increase and this power consumption too.

```
preds <- data.frame(temp=c(10,35))
preds$Mwh <- predict(fit,newdata = preds)
preds$type <- 'yhat'
econsumption$type <- 'original'
econsumption <- bind_rows(econsumption, preds)
lattice::xyplot(Mwh~temp,groups=type,econsumption,auto.key=T,type=c('p','r'),
                main='Electric consumption vs Max daily Temp')
```



d. Give prediction intervals for your forecasts.

These are the prediction intervals for a 95% CI.

```
predict(fit,newdata = preds, interval = 'predict', level = 0.95)
```

```
##        fit      lwr      upr
## 1 18.74795 16.34824 21.14766
## 2 15.11902 12.49768 17.74035
```

2. The following table gives the winning times (in seconds) for the men's 400 meters final in each Olympic Games from 1896 to 2012...

a. Update the data set `olympic` to include the winning times from the last few Olympics.

```
# Source: https://www.olympic.org/athletics/400m-men
olympic <- olympic %>% bind_rows(tribble(~Year, ~time, 2000, 43.84,2004, 44.00,2008,
                                         43.75,2012, 43.94,2016, 43.03))
tail(olympic,8)
```

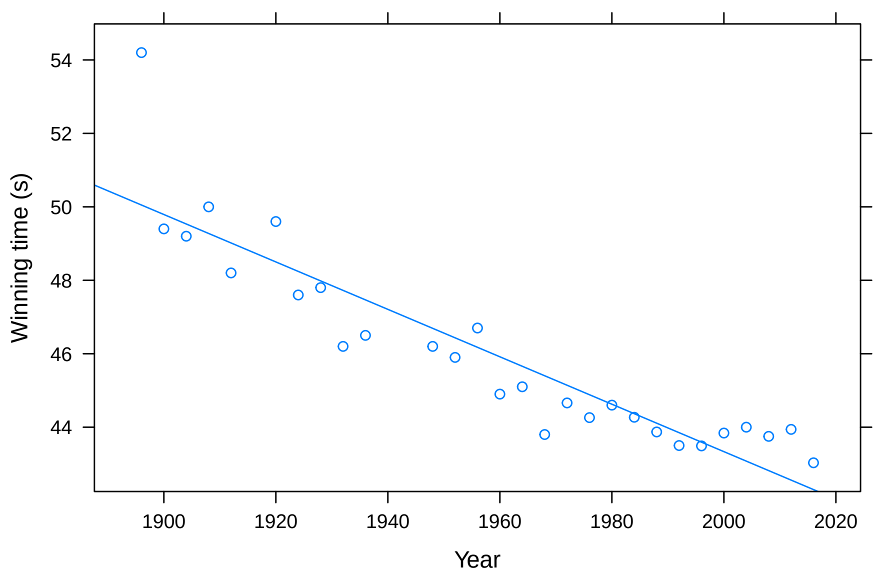
	Year	time
21	1988	43.87
22	1992	43.50
23	1996	43.49
24	2000	43.84
25	2004	44.00
26	2008	43.75
27	2012	43.94

	Year	time
28	2016	43.03

b. Plot the winning time against the year. Describe the main features of the scatter-plot.

- The winning time is consistantly decreasing
- The plot seems to be getting asymptotic in nature
- The variation across time seems to be continously reducing

```
lattice::xyplot(time~Year,olympic,type=c('p','r'),ylab='Winning time (s)')
```



c. Fit a regression line to the data. Obviously the winning times have been decreasing, but at what *average* rate per year?

The winning times reduce by 0.064 seconds on average year over year.

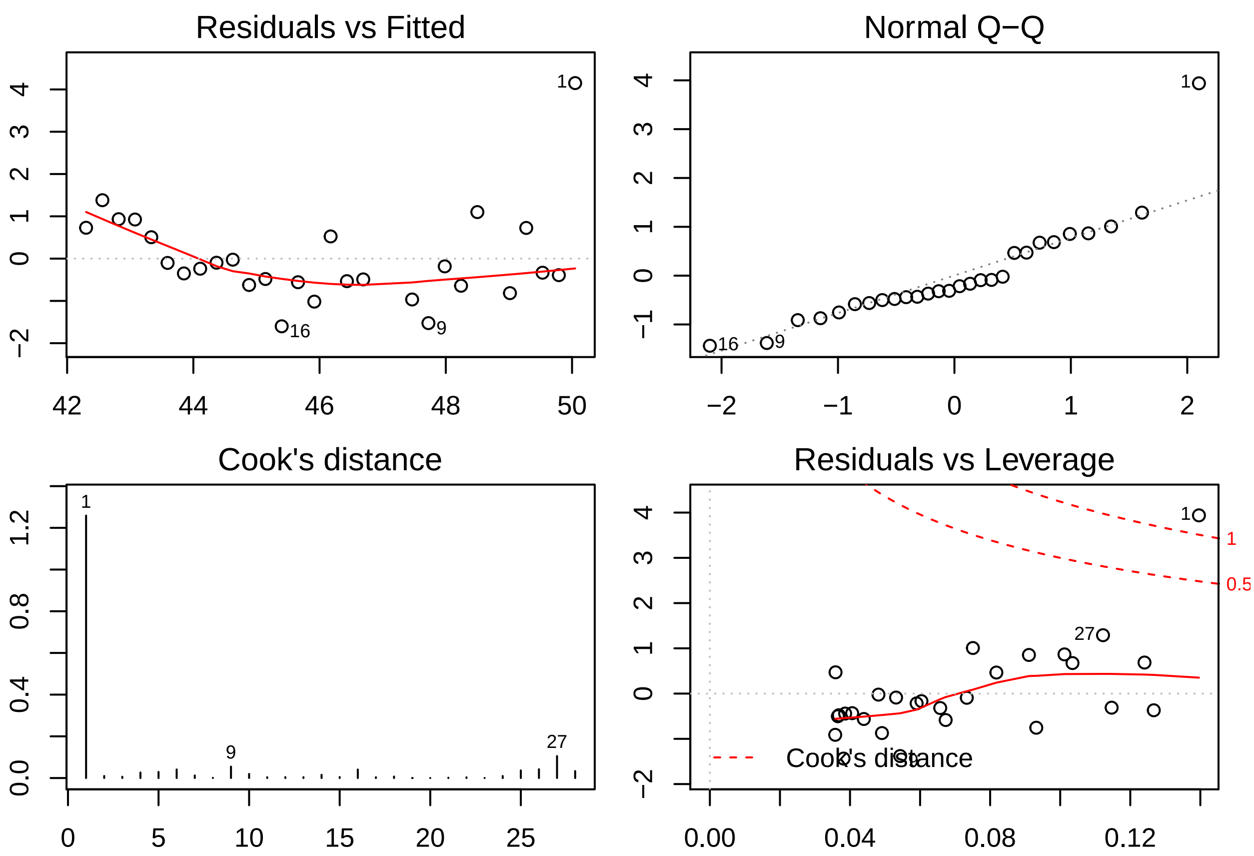
```
summary(lm(time~Year, olympic))
```

```
##
## Call:
## lm(formula = time ~ Year, data = olympic)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.6002 -0.5747 -0.2858  0.5751  4.1505
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 172.481477  11.487522   15.02 2.52e-14 ***
## Year         -0.064574   0.005865  -11.01 2.75e-11 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.136 on 26 degrees of freedom
## Multiple R-squared:  0.8234, Adjusted R-squared:  0.8166
## F-statistic: 121.2 on 1 and 26 DF,  p-value: 2.752e-11
```

d. Plot the residuals against the year. What does this indicate about the suitability of the fitted line?

```
old.par <- par(); par(mfrow=c(2,2), mar=c(2,2,2,1))
plot(lm(time~Year, olympic), which = c(1,2,4,5)); par(old.par)
```



A linear fit isn't the right one for this data, as we can already see from the scatter plots. The residuals confirm this:

- There is a pattern in the residuals^yhat plot - Higher residuals at the beginning, then lower, then higher again. We can also confirm this from the Durbin-Watson statistic, which is ~1.2 with a p-val of 0.008 indicating we must reject the null hypothesis Ho: There is no autocorrleation in the residuals.

```
lmtest::dwtest(lm(time~Year, olympic))
```

```
##
##  Durbin-Watson test
```

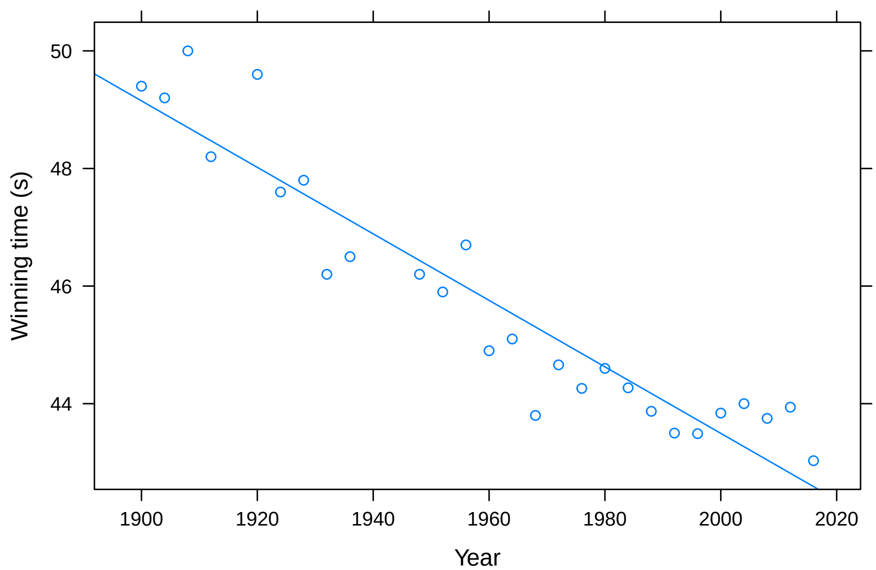
```
##
## data:  lm(time ~ Year, olympic)
## DW = 1.2244, p-value = 0.008555
## alternative hypothesis: true autocorrelation is greater than 0
```

- QQplot and Cook's distance plots show pt-1 as a massive outlier
- The leverage plot also confirms pt-1 to show high leverage

Predict the winning time for the men's 400 meters final in the 2000, 2004, 2008 and 2012 Olympics. Give a prediction interval for each of your forecasts. What assumptions have you made in these calculations?

I would first eliminate the first point because it's a clear outlier.

```
olympic <- tail(olympic,-1)
lattice::xyplot(time~Year,olympic,type=c('p','r'),ylab='Winning time (s)')
```



Fitting a new regrssion line, and predicting the last 4 years now. The printout shows the prediction interval at a 95% confidence.

```
fit <- lm(time~Year, olympic)
olympic <- olympic %>% bind_cols(tbl_df(predict(fit, interval = 'predict', level = 0.95)))
```

```
## Warning in predict.lm(fit, interval = "predict", level = 0.95): predictions on current data
```

```
tail(olympic)
```

	Year	time	fit	lwr	upr
22	1996	43.49	43.72141	42.15134	45.29147
23	2000	43.84	43.49522	41.91866	45.07178
24	2004	44.00	43.26904	41.68532	44.85275
25	2008	43.75	43.04285	41.45132	44.63438
26	2012	43.94	42.81667	41.21667	44.41666

	Year	time	fit	lwr	upr
27	2016	43.03	42.59048	40.98138	44.19958

These calculations assume:

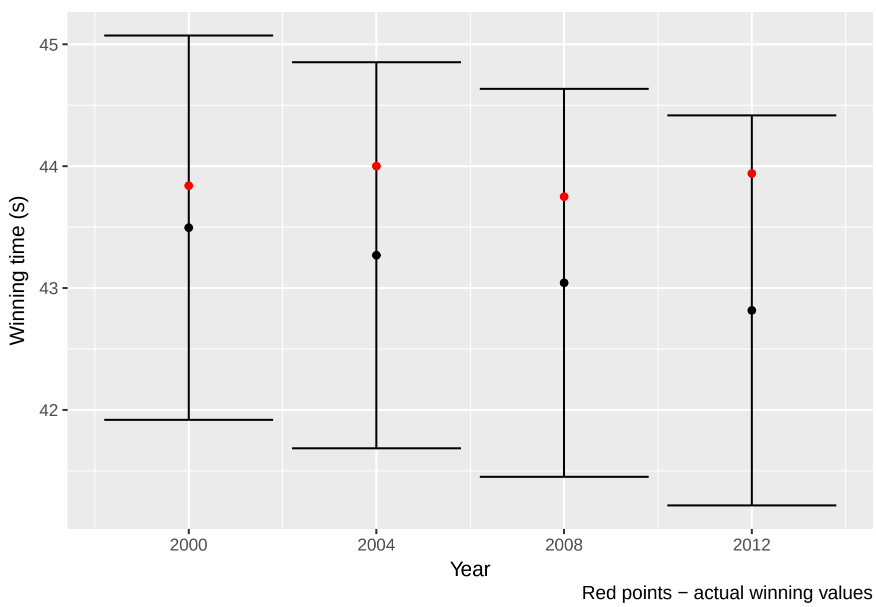
- Each observation is IID
- There is no effect of seasonality or external factors considered (location, temperature, humidity etc)

e. Find out the actual winning times for these Olympics (see www.databaseolympics.com). How good were your forecasts and prediction intervals?

The linear trend cannot account for the uptick in winning times between 2000 and 2012. So the actuals do lie between the 95% PI yet we can see that the actuals start approaching the ends of the error bars at 2012.

```
olympic %>% filter(Year %in% c(2000,2004,2008,2012)) %>% mutate(e = time - fit) %>%
    ggplot(aes(x=Year))+
    geom_errorbar(aes(ymin=lwr,ymax=upr))+
    geom_point(aes(y=fit))+
    geom_point(aes(y=time),col='red')+
    scale_x_continuous(breaks = c(2000,2004,2008,2012))+
    labs(y='Winning time (s)', caption = 'Red points - actual winning values')
```

```
## Warning: package 'bindrcpp' was built under R version 3.4.4
```



3. An elasticity coefficient is the ratio of the percentage change in the forecast variable (y) to the percentage change in the predictor variable (x)... Express y as a function of x and show that the coefficient beta1 is the elasticity coefficient.

Section 6.7

1. Show that a 3x5MA is equivalent to a 7-term weighted moving average with weights of 0.067, 0.133, 0.200, 0.200, 0.200, 0.133, and 0.067.

- See next page *

2. `plastics` dataset question

The dataset:

```
plastics
```

```
##    Jan  Feb  Mar  Apr  May  Jun  Jul  Aug  Sep  Oct  Nov  Dec
## 1  742  697  776  898 1030 1107 1165 1216 1208 1131  971  783
## 2  741  700  774  932 1099 1223 1290 1349 1341 1296 1066  901
## 3  896  793  885 1055 1204 1326 1303 1436 1473 1453 1170 1023
## 4  951  861  938 1109 1274 1422 1486 1555 1604 1600 1403 1209
## 5 1030 1032 1126 1285 1468 1637 1611 1608 1528 1420 1119 1013
```

a. Plot the time series of sales of product A. Can you identify seasonal fluctuations and/or a trend?

```
ggplot2::autoplot(plastics, main ='Monthly sale (in thou) of Product A')+
    ggplot2::geom_smooth(se = F)
```

Datum / Date:

HW #1 ⟨2.8⟩ ·1 ·2 ·3 ·4

⟨4.10⟩ ·1 ·2 ·3

⟨6.7⟩ ×1 ·2 ·3

3x5 MA:

$$\hat{T}_{t^*}\Big|_{5MA} = \frac{1}{5}\sum_{k=-2}^{k=2} y_{t-k} = \left(\frac{y_{t-2}}{5} + \frac{y_{t-1}}{5} + \frac{y_t}{5} + \frac{y_{t+1}}{5} + \frac{y_{t+2}}{5}\right)$$

$$\hat{T}_t\Big|_{3\times5MA} = \frac{1}{3}\hat{y}_{t^*-1} + \frac{1}{3}\hat{y}_{t^*} + \frac{1}{3}\hat{y}_{t^*+1}$$

$$= \frac{1}{3}\left[\frac{1}{5}y_{t-3} + \frac{1}{5}y_{t-2} + \frac{1}{5}y_{t-1} + \frac{1}{5}y_t + \frac{1}{5}y_{t+1}\right]$$
$$+ \frac{1}{3}\left[\frac{1}{5}y_{t-2} + \frac{1}{5}y_{t-1} + \frac{1}{5}y_t + \frac{1}{5}y_{t+1} + \frac{1}{5}y_{t+2}\right]$$
$$+ \frac{1}{3}\left[\frac{1}{5}y_{t-1} + \frac{1}{5}y_t + \frac{1}{5}y_{t+1} + \frac{1}{5}y_{t+2} + \frac{1}{5}y_{t+3}\right]$$

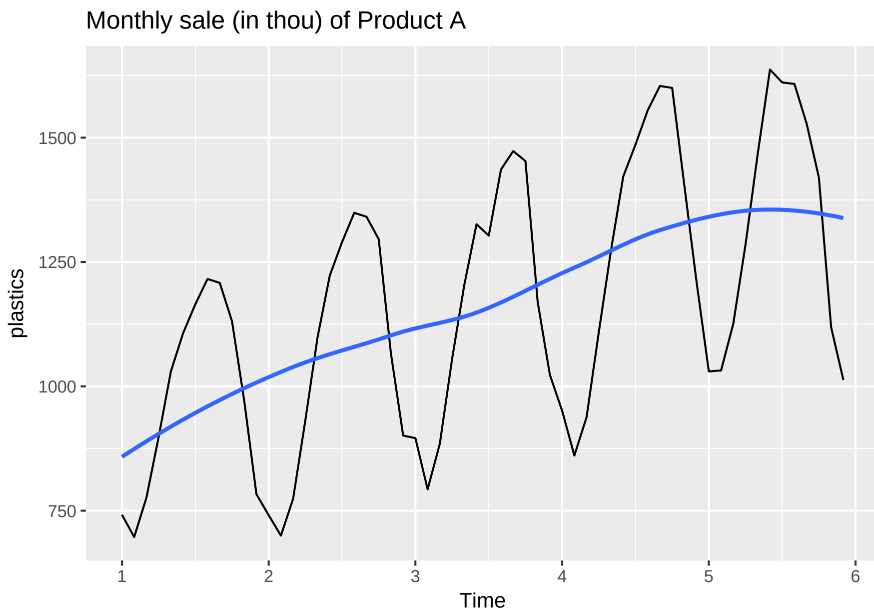
$$\hat{T}_t\Big|_{3\times5MA} = \frac{1}{3}\cdot\frac{1}{5}y_{t-3} + 2\cdot\frac{1}{3}\cdot\frac{1}{5}\cdot y_{t-2} + 3\cdot\frac{1}{3}\cdot\frac{1}{5}\cdot(y_{t-1} + y_t + y_{t+1}) + 2\cdot\frac{1}{3}\cdot\frac{1}{5}\cdot y_{t-2} + \frac{1}{3}\cdot\frac{1}{5}\cdot y_{t+3}$$

$\Rightarrow$ Wts = [0.067, 0.133, 0.2, 0.2, 0.2, 0.133, 0.067]

SAME AS 7-TERM WTED M.A.

34

Figure 1:



- Seasonal fluctuations exist, with peaks ~Aug-Sep and reduced sales at the year end
- A monotonic near-linear trend also exists year over year

b. Use a classical multiplicative decomposition to calculate the trend-cycle and seasonal indices.

```r
multi_decomp <- decompose(plastics, type = 'multiplicative')
```

The trend and seasonal indices can be accessed via:

```r
multi_decomp$trend
```

```
##         Jan       Feb       Mar       Apr       May       Jun       Jul
## 1        NA        NA        NA        NA        NA        NA  976.9583
## 2 1000.4583 1011.2083 1022.2917 1034.7083 1045.5417 1054.4167 1065.7917
## 3 1117.3750 1121.5417 1130.6667 1142.7083 1153.5833 1163.0000 1170.3750
## 4 1208.7083 1221.2917 1231.7083 1243.2917 1259.1250 1276.5833 1287.6250
## 5 1374.7917 1382.2083 1381.2500 1370.5833 1351.2500 1331.2500        NA
##         Aug       Sep       Oct       Nov       Dec
## 1  977.0417  977.0833  978.4167  982.7083  990.4167
## 2 1076.1250 1084.6250 1094.3750 1103.8750 1112.5417
## 3 1175.5000 1180.5417 1185.0000 1190.1667 1197.0833
## 4 1298.0417 1313.0000 1328.1667 1343.5833 1360.6250
## 5        NA        NA        NA        NA        NA
```

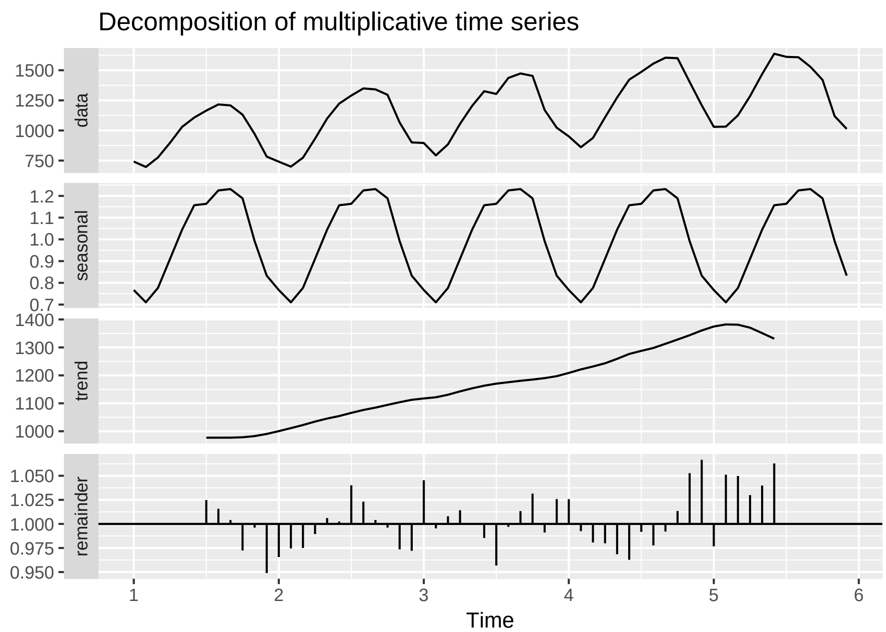
```r
multi_decomp$seasonal
```

```
##         Jan       Feb       Mar       Apr       May       Jun       Jul
## 1 0.7670466 0.7103357 0.7765294 0.9103112 1.0447386 1.1570026 1.1636317
## 2 0.7670466 0.7103357 0.7765294 0.9103112 1.0447386 1.1570026 1.1636317
## 3 0.7670466 0.7103357 0.7765294 0.9103112 1.0447386 1.1570026 1.1636317
## 4 0.7670466 0.7103357 0.7765294 0.9103112 1.0447386 1.1570026 1.1636317
```

```
## 5 0.7670466 0.7103357 0.7765294 0.9103112 1.0447386 1.1570026 1.1636317
##         Aug       Sep       Oct       Nov       Dec
## 1 1.2252952 1.2313635 1.1887444 0.9919176 0.8330834
## 2 1.2252952 1.2313635 1.1887444 0.9919176 0.8330834
## 3 1.2252952 1.2313635 1.1887444 0.9919176 0.8330834
## 4 1.2252952 1.2313635 1.1887444 0.9919176 0.8330834
## 5 1.2252952 1.2313635 1.1887444 0.9919176 0.8330834
```

c. Do the results support the graphical interpretation from part (a)?

```r
ggplot2::autoplot(multi_decomp)
```

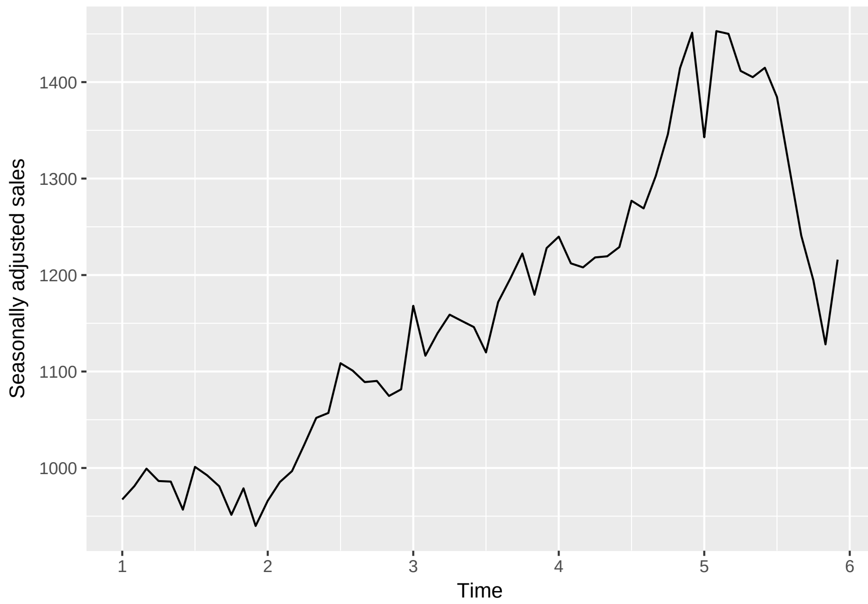


Yes

- We can see a clear increasing trend
- We can see that the seasonality has extracted a lot of the variation from the original series.
- Once the in a linear trend is removed, a majority of the variation is explained by the seasonal component - most of the remainder is ~1. At places where it's not equal to one, the magnitudes are fairly small: [-0.025,0.05] about 1 compared to that of seasonal [-0.3, 0.2] about 1.

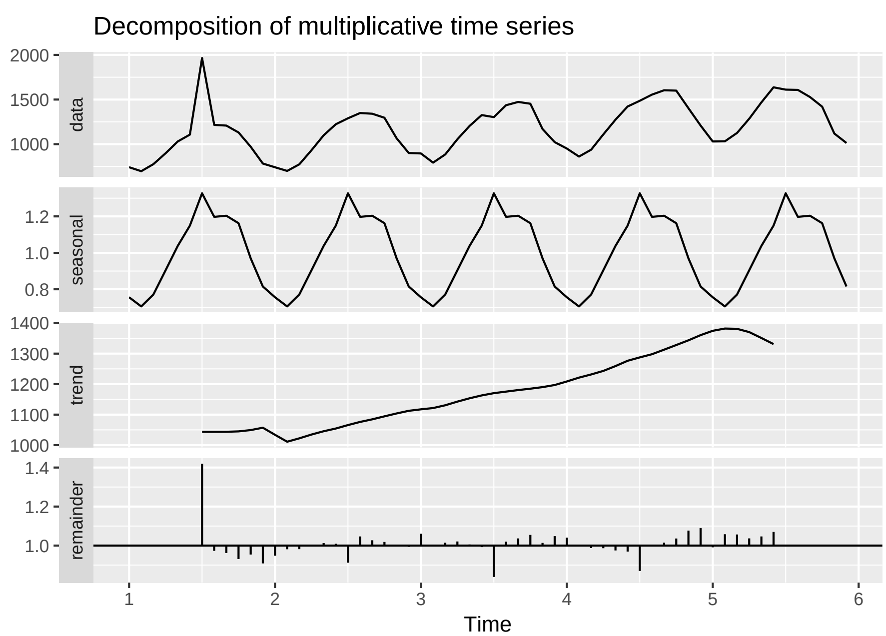
d. Compute and plot the seasonally adjusted data.

```r
ggplot2::autoplot(seasadj(multi_decomp), ylab='Seasonally adjusted sales')
```



e. Change one observation to be an outlier (e.g., add 500 to one observation), and recompute the seasonally adjusted data. What is the effect of the outlier?

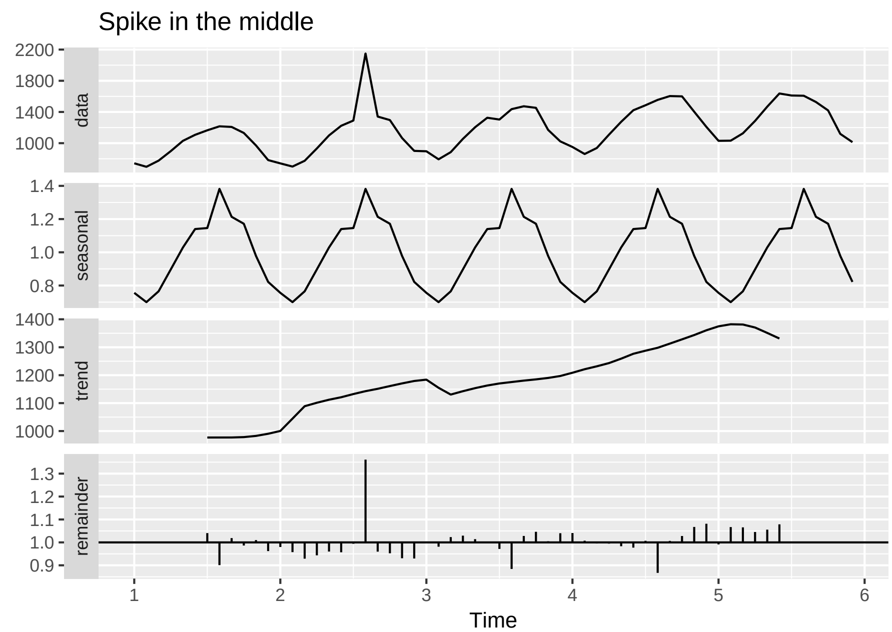
```
plastics.withoutlier <- plastics
plastics.withoutlier[7] <- plastics.withoutlier[7] + 800
ggplot2::autoplot(decompose(plastics.withoutlier, type = 'multiplicative'))
```



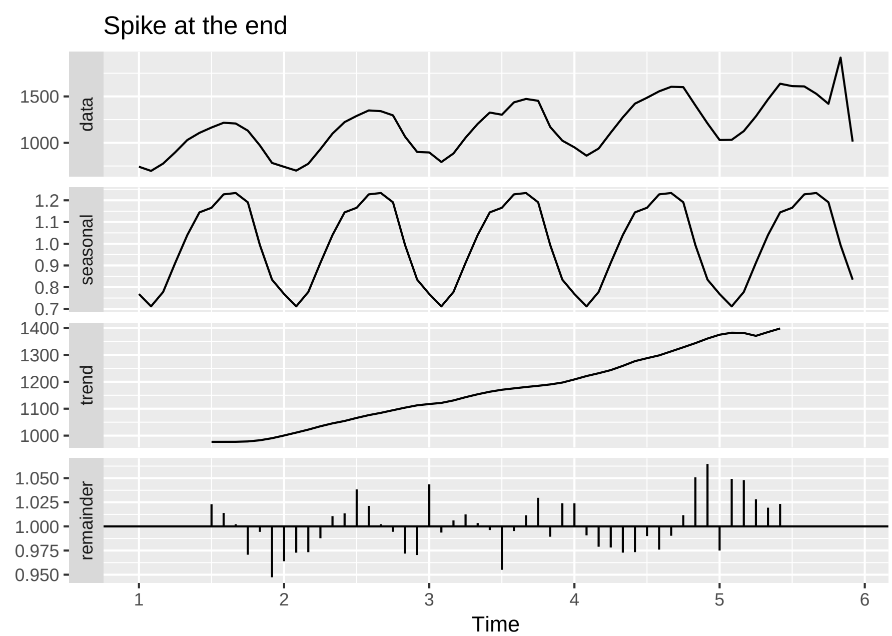
- Classic decomposition is not robust to outliers.
- It affects the seasonal contributions throughout the time series, since `decompose()` cannot adjust seasonal contributions over time.
- As a result, we can see high residuals at (T+7) throughout the time series.

f. Does it make any difference if the outlier is near the end rather than in the middle of the time series?

```
plastics.withoutlier1 <- plastics
plastics.withoutlier1[20] <- plastics.withoutlier[20] + 800
plastics.withoutlier2 <- plastics
plastics.withoutlier2[59] <- plastics.withoutlier[59] + 800
ggplot2::autoplot(decompose(plastics.withoutlier1, type = 'multiplicative'))+
    labs(title='Spike in the middle')
```




```
ggplot2::autoplot(decompose(plastics.withoutlier2, type = 'multiplicative'))+
    labs(title='Spike at the end')
```



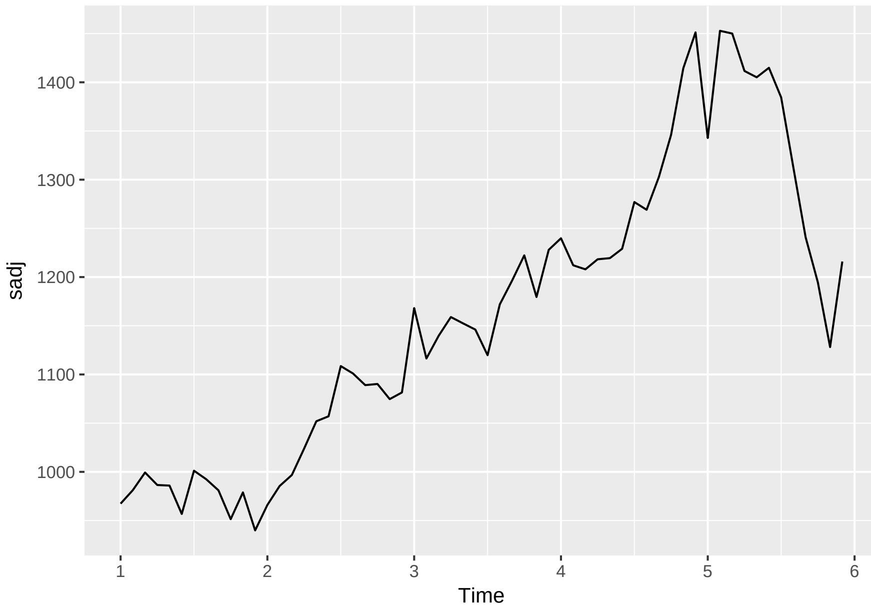
Yes.

The reason is that when the spike is at the end, if that portion of the signal is `NA` in the trend extraction using MA, it is not considered during calculation of the seasonal contributions.

g. Use a random walk with drift to produce forecasts of the seasonally adjusted data.

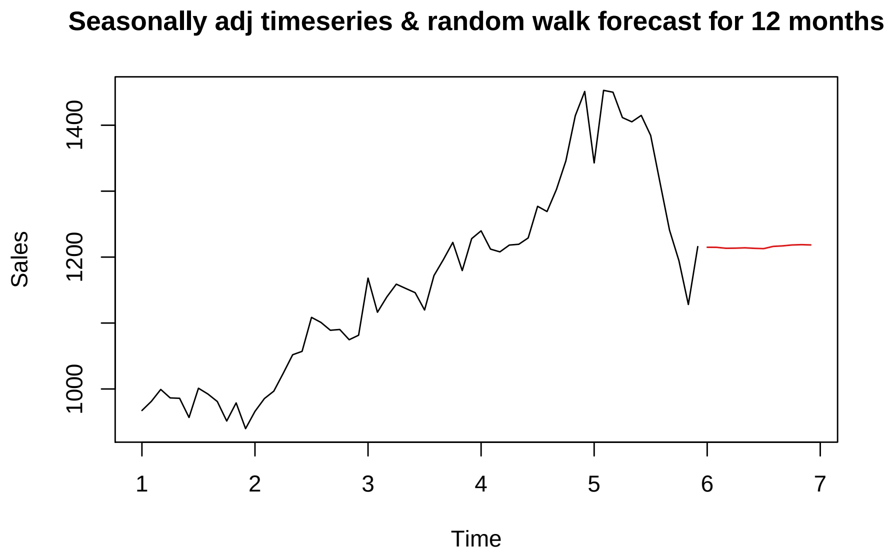
To extract the seasonally adjusted data:

```
sadj <- seasadj(multi_decomp)
ggplot2::autoplot(sadj)
```



Adding the random walk here, shown in red.

```
h <-  12
mu <- tail(sadj, 1)
rwalk <- as.numeric(mu) + cumsum(rnorm(h,))
rwalk <- ts(rwalk, frequency = 12, start = c(6,1))
together <- ts.union(sadj, rwalk)
plot(together, main = 'Seasonally adj timeseries & random walk forecast for 12 months',
     plot.type = 's',col=c('black','red'), ylab = 'Sales')
```



h. Reseasonalize the results to give forecasts on the original scale.

Multiplying the seasonal component to the multiplicative series, we get results back in the original scale.

```
rwalk_reseasoned <- rwalk * as.numeric(multi_decomp$seasonal)[1:12]
together <- ts.union(sadj, rwalk_reseasoned)
plot(together, main = 'Random walk forecast for 12 months',plot.type = 's',
     col=c('black','red'), ylab = 'Sales')
```