BHao\_HW1

library(fma)  
# data(package='fma')

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

Given the exponential nature of population growth in absolute terms, a log transformation makes sense. To remove the impact of population changes, we could look at the data on a per-capita basis as well.

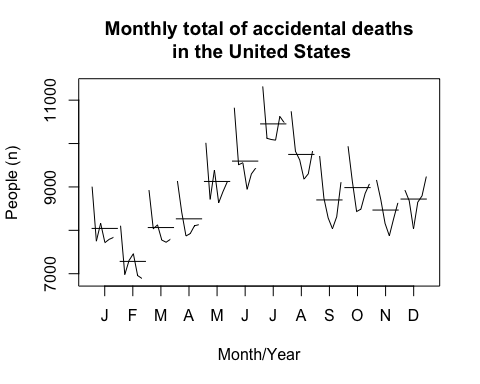
plot(log(dole), ylab = 'People (log n)', xlab = 'Year',   
 main = 'Monthly total of people on unemployed\nbenefits in Australia')



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

Given the 1) monthly seasonality of this data and 2) the general declining trend over the years, the monthplot chart seemed like an appropriate choice. To remove the impact of population changes, we could look at the data on a per-capita basis as well.

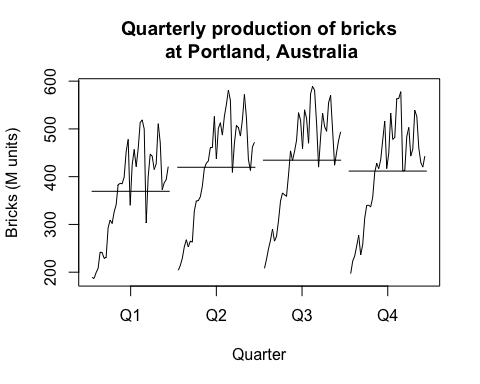
monthplot(usdeaths, ylab = 'People (n)', xlab = 'Month/Year',   
 main = 'Monthly total of accidental deaths\n in the United States')



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

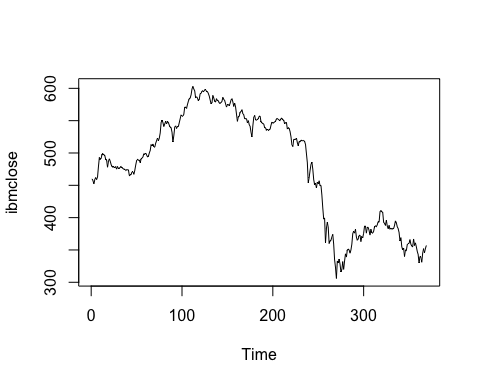
Again, there appears to be some quarterly seasonality in the data as well as a general increase over time (with a few years of sharp decline); as such, a monthplot (or quarterly in this case) seems to capture that information well.

monthplot(bricksq, ylab = 'Bricks (M units)', xlab = 'Quarter',   
 main = 'Quarterly production of bricks\n at Portland, Australia')



## 2.8.3.a. 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)



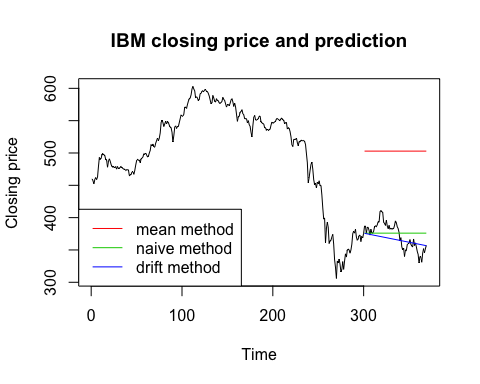
## 2.8.3.b. Split the data into a training set of 300 observations and a test set of 69 observations.

ibm\_train = window(ibmclose, start=1 , end=300)  
ibm\_test = window(ibmclose, start=301)

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

Of the mean, naive and seasonal naive methods, the seasonal naive method produced the most accurate result.

ibm\_meanfit = meanf( ibm\_train, h = 69)  
ibm\_naivefit = rwf( ibm\_train, h = 69)  
ibm\_driftfit = rwf( ibm\_train, h = 69, drift = TRUE)  
  
plot(ibmclose, ylab = 'Closing price', xlab = 'Time',   
 main = 'IBM closing price and prediction')  
lines(ibm\_meanfit$mean , col = 2)  
lines(ibm\_naivefit$mean, col = 3)  
lines(ibm\_driftfit$mean, col = 4)  
legend('bottomleft', lty = 1, col = c(2, 3, 4),  
 legend = c('mean method', 'naive method', 'drift method'))



accuracy(ibm\_meanfit , ibm\_test)

## ME RMSE MAE MPE MAPE  
## Training set 1.660438e-14 73.61532 58.72231 -2.642058 13.03019  
## Test set -1.306180e+02 132.12557 130.61797 -35.478819 35.47882  
## MASE ACF1 Theil's U  
## Training set 11.52098 0.9895779 NA  
## Test set 25.62649 0.9314689 19.05515

accuracy(ibm\_naivefit, ibm\_test)

## ME RMSE MAE MPE MAPE MASE  
## Training set -0.2809365 7.302815 5.09699 -0.08262872 1.115844 1.000000  
## Test set -3.7246377 20.248099 17.02899 -1.29391743 4.668186 3.340989  
## ACF1 Theil's U  
## Training set 0.1351052 NA  
## Test set 0.9314689 2.973486

accuracy(ibm\_driftfit, ibm\_test)

## ME RMSE MAE MPE MAPE  
## Training set 2.870480e-14 7.297409 5.127996 -0.02530123 1.121650  
## Test set 6.108138e+00 17.066963 13.974747 1.41920066 3.707888  
## MASE ACF1 Theil's U  
## Training set 1.006083 0.1351052 NA  
## Test set 2.741765 0.9045875 2.361092