Introductory Time Series with R
Chapter 1 Exercises

Jacob Carey

December 16, 2013

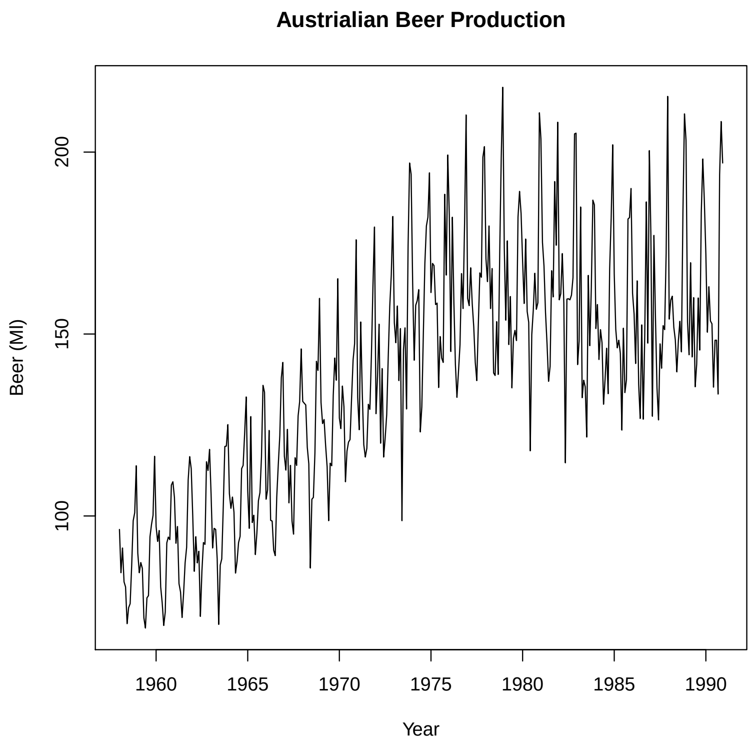
Note that all data (unless otherwise noted) were obtained from
`http://elena.aut.ac.nz/~pcowpert/ts`

1. Carry out the following exploratory time series analysis in R using either the chocolate or the beer production data from §1.4.3.

   (a) Produce a time plot of the data. Plot the aggregated annual series and a boxplot that summarizes the observed values for each season, and comment on the plots.

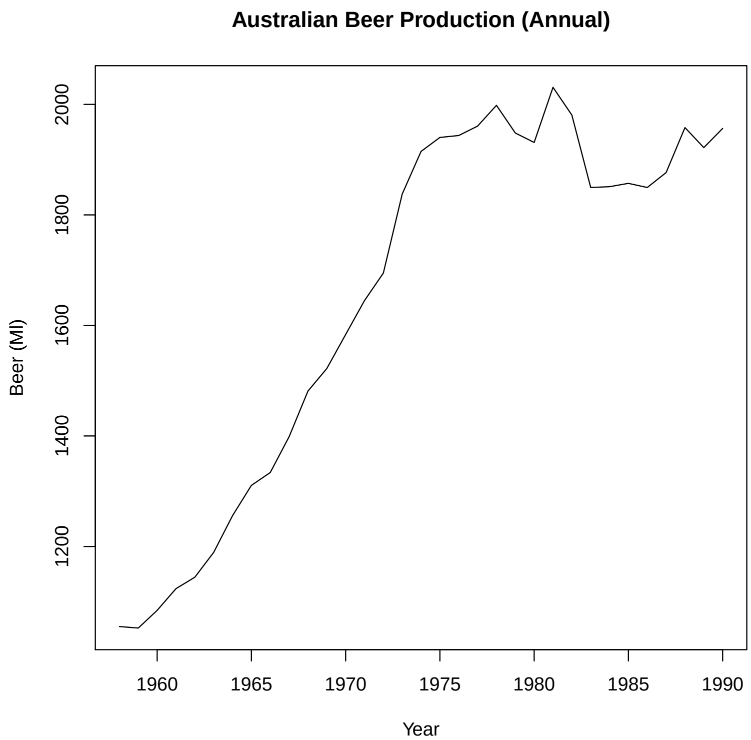
We choose to use the beer data and have imported it as a time series, `Beer.ts`.

```
plot(Beer.ts, xlab="Year", ylab="Beer (Ml)",
     main="Austrialian Beer Production")
```



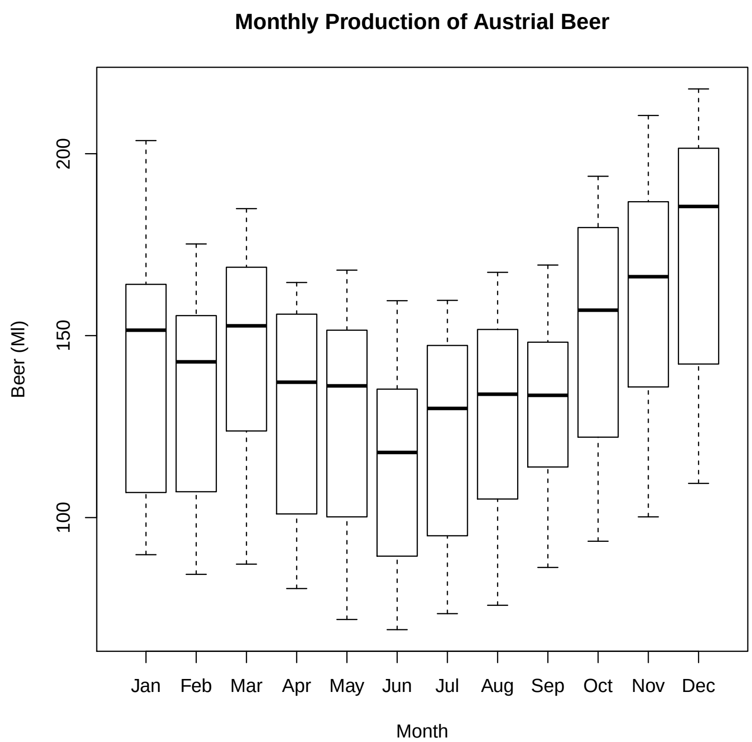
We aggregate the Beer data to create the second plot

```
Beer.year <- aggregate(Beer.ts, FUN=sum)
plot(Beer.year, xlab="Year", ylab="Beer (Ml)",
     main="Australian Beer Production (Annual)")
```



Finally, we use the `cycle` function to create a boxplot of monthly beer production.

```
boxplot(Beer.ts~cycle(Beer.ts), xlab="Month",
        ylab="Beer (Ml)",
        main="Monthly Production of Austrial Beer",
        xaxt="n")
axis(1, 1:12, month.abb[1:12])
```

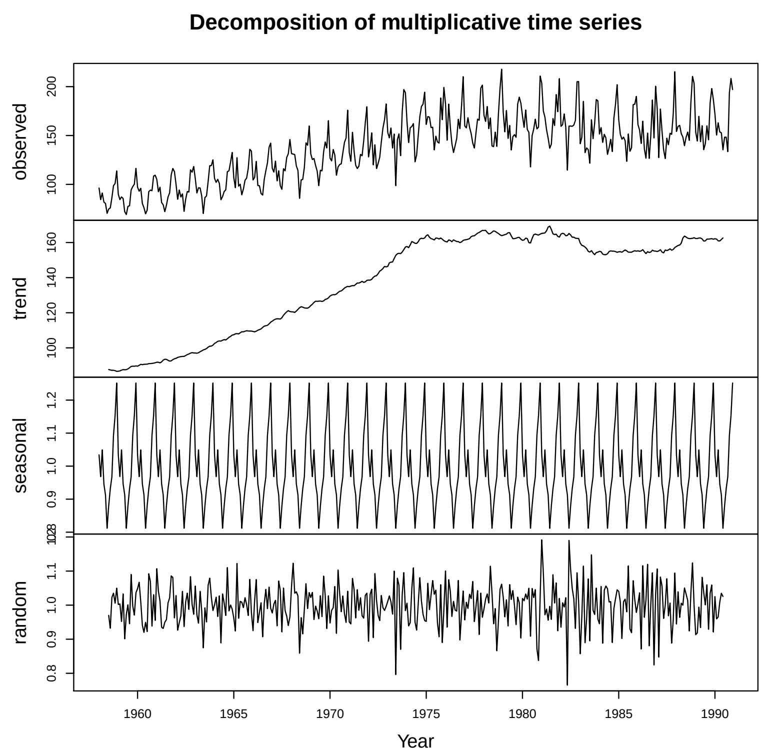


(b) Decompose the series into the components trend, seasonal effect, and residuals, and plot the decomposed series. Produce a plot of the trend with a superimposed seasonal effect.
First we decompose the series, choosing multiplicate decomposition, by using the `decompose` function.

```
Beer.decompose <- decompose(Beer.ts, "multi")
Trend <- Beer.decompose$trend
Seasonal <- Beer.decompose$seasonal
```

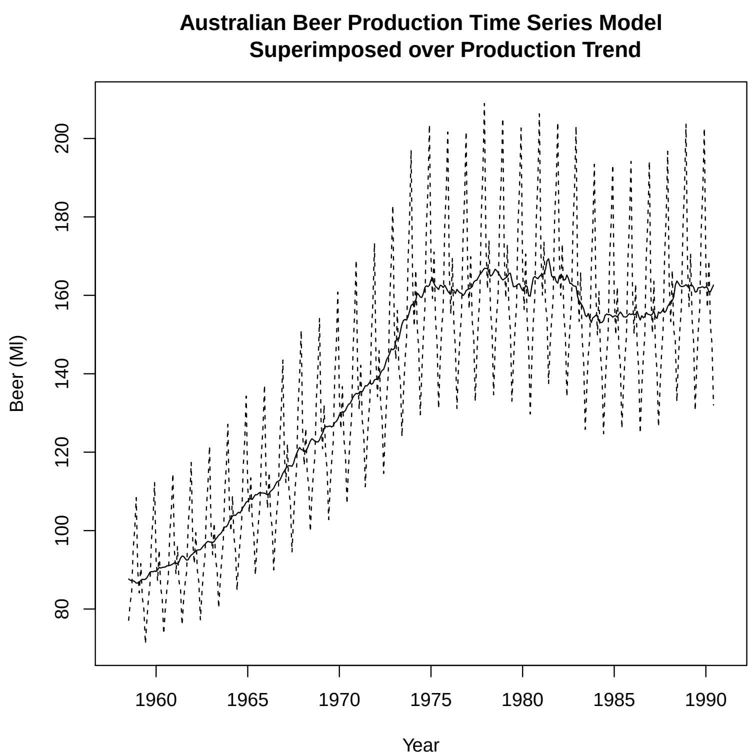
Next, we plot the decomposition.

```
plot(Beer.decompose, xlab = "Year")
```



Finally, we create a plot of the time series model superimposed over the trend. We use the `ts.plot` function, which is designed for plotting time series, and can superimpose plots by `cbind`-ing the time-series objects.

```
ts.plot(cbind(Trend, Trend*Seasonal), lty=1:2,
        xlab="Year", ylab="Beer (Ml)",
        main="Australian Beer Production Time Series Model
        Superimposed over Production Trend")
```



2. Many economic time series are based on indices. A price index is the ratio of the cost of a basket of goods now to its cost in some base year. In the Laspeyre formulation, the basked is based on typical purchases in the base year. You are asked to calculate an index of of motoring cost from the following data. The clutch represents all mechanical parts, and the quantity allows for this.

	quantity.00	unit.price.00	quantity.04	unit.price.04
car	0.33	18000.00	0.50	20000.00
petrol	2000.00	0.80	1500.00	1.60
servicing	40.00	40.00	20.00	60.00
tyre	3.00	80.00	2.00	120.00
clutch	2.00	200.00	1.00	360.00

The *Laspeye Price Index* at time t relative to base year 0 is

$$\frac{\sum q_{i0}p_{it}}{\sum q_{i0}p_{i0}}$$

Calculate the LI_t for 2004 relative to 2000.
We have created a dataset from the listed table and call the data frame `Auto.df`. We use the following code to calculate the LI_t

```
LI_t <- sum(Auto.df$quantity.00*Auto.df$unit.price.04)/
  sum(Auto.df$quantity.00*Auto.df$unit.price.00)
LI_t

## [1] 1.358
```

3. The *Paasche Price Index* at time t relative to base year 0 is

$$\frac{\sum q_{it}p_{it}}{\sum q_{it}p_{i0}}$$

   (a) Use the data above to calculate the PI_t for 2004 relative to 2000.
   Using the created `Auto.df`, we calculate the PI_t

```
PI_t <- sum(Auto.df$quantity.04*Auto.df$unit.price.04)/
  sum(Auto.df$quantity.04*Auto.df$unit.price.00)
PI_t

## [1] 1.25
```

   (b) Explain why the PI_t is usually lower than the LI_t.

   People tend to buy fewer of things as the prices increase, and since the quantity used in this calculation comes from step t instead of step 0, the quantity for items that have increased in price is typically lower

   (c) Calculate the *Irving-Fisher Price Index* as the geometric mean of the LI_t and PI_t. (The geometric mean of a sample of n items is the nth root of their product.)
   Using the calculated PI_t and LI_t, we calculate the *Irving-Fisher Price Index*

```
sqrt(LI_t * PI_t)

## [1] 1.303
```