Chapter 1: Exploratory Data Analysis

2023-03-04

Some notes about R Markdown

The following code is added to the YAML header for setting up automated section numbering:

```
output:
  html_document:
    number_sections: TRUE
```

The following code chunk should be included at the top of all R Markdown document:

```
{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```

There are options in writing code chunks and the following is a brief description of some useful options.

- `include =`: `inclue = FALSE` prevents code and results from appearing in the finished file. R Markdown still runs the code in the chunk, and the results can be used by other chunks.
- `echo =`: `echo = TRUE` shows the code, where `echo = FALSE` prevents code, but not the results from appearing in the finished file. This is a useful way to embed figures (see the boxplot below).
- `eval =`: `eval = TRUE` means to evaluate (run) the code, where `eval = FALSE` will just show the code but not evalutate it.

Estimates of Location

The dataset `state.csv` is used to demonstrate the calculation of the various estimates of location.

```
# Import the dataset
state <- read.csv('Data/state.csv')
```

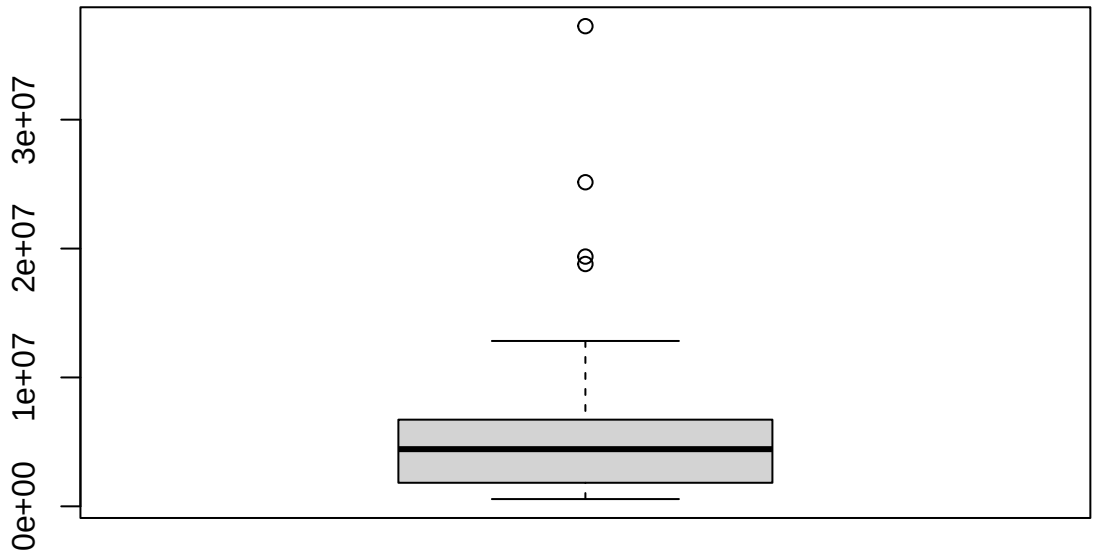
```
# First few observations
head(state)
```

```
##        State Population Murder.Rate Abbreviation
## 1    Alabama    4779736         5.7           AL
## 2     Alaska     710231         5.6           AK
## 3    Arizona    6392017         4.7           AZ
## 4   Arkansas    2915918         5.6           AR
## 5 California   37253956         4.4           CA
## 6   Colorado    5029196         2.8           CO
```

```r
# Summary statistics
summary(state)
```

```
##     State              Population        Murder.Rate     Abbreviation
##  Length:50          Min.   :  563626   Min.   : 0.900   Length:50
##  Class :character   1st Qu.: 1833004   1st Qu.: 2.425   Class :character
##  Mode  :character   Median : 4436370   Median : 4.000   Mode  :character
##                     Mean   : 6162876   Mean   : 4.066
##                     3rd Qu.: 6680312   3rd Qu.: 5.550
##                     Max.   :37253956   Max.   :10.300
```

The boxplot of the variable `Population` is displayed below:



Mean (average)

The sum of all values divided by the number of values.

```r
# Mean
mean(state[['Population']])
```

```
## [1] 6162876
```

Trimmed mean (truncated mean)

The average of all values after dropping a fixed number of extreme values. It is calculated by dropping a fixed number of sorted values at each end and then taking an average of the remaining values.

A trimmed mean eliminates the influence of extreme values.

```
# Trimmed mean
# 'trim=0.1' drops 10% from each end
mean(state[['Population']], trim=0.1)
```

```
## [1] 4783697
```

A further example for trimmed mean:

```
# Data with all the middle values equal to 4
skewdata <- c(1,4,4,4,4,4,4,4,4,1000)
```

```
# The mean is influenced by the outlier 1000
mean(skewdata)
```

```
## [1] 103.3
```

```
# The trimmed mean is calculated by dropping the first and last observation
mean(skewdata, trim=0.1)
```

```
## [1] 4
```

Median (50th percentile)

The value such that one-half of the data lies above and below.

Compared to the mean, which uses all observations, the median depends only on the values in the center of the sorted data. While this might seem to be a disadvantage, since the mean is much more sensitive to the data, there are many instances in which the median is a better metric for location.

```
# Median
median(state[['Population']])
```

```
## [1] 4436370
```

Weighted mean (weighted average)

The sum of all values times a weight divided by the sum of the weights.

There are two main motivations for using a weighted mean:

- Some values are intrinsically more variable than others, and highly variable observations are given a lower weight.
- The data collected does not equally represent the different group that we are interested in mearsuring.

If we want to compute the average murder rate for the country, we need to use a weighted mean or median to account for different populations in the states.

```
# Weighted mean
weighted.mean(state[['Murder.Rate']], w=state[['Population']])
```

```
## [1] 4.445834
```

Weighted median

The value such that one-half of the sum of the weights lies above and below the sorted data.

Since base *R* doesn't have a function for weighted median, we need to install a package such as `matrixStats`:

```
# Weighted median
library('matrixStats')
weightedMedian(state[['Murder.Rate']], w=state[['Population']])
```

```
## [1] 4.4
```

Percentile (quantile)

The value such that P percent of the data lies below.

Outliers

The median is referred to as a robust estimate of location since it is not influenced by outliers (extreme cases) that could skew the results. An outlier is any value that is very distant from the other values in a data set. Being an outlier in itself does not make a data value invalid or erroneous. When outliers are the result of bad data, the mean will result in a poor estimate of location, while the median will still be valid.

The median is not the only robust estimate of location. In fact, a trimmed mean is widely used to avoid the influence of outliers. The trimmed mean can be thought of as a compromise between the median and the mean: it is robust to extreme values in the data, but uses more data to calculate the estimate for location.

Robust (resistant)

Not sensitive to extreme values.

Estimates of variability

Deviations (errors, residuals)

The difference between the observed values and the estimate of location.

Variance (mean-squared-error)

The sum of squared deviations from the mean divided by $n - 1$ where n is the number of data values.

```r
# Variance
var(state[['Population']])
```

```
## [1] 4.689833e+13
```

The `var()` function in base R calculates the sample variance by the following formula:

$$\text{Variance} = s^2 = \frac{\sum_{i=1}^{n}(x_i - \overline{x})^2}{n-1}$$

Degree of Freedom, and n or $n-1$

We have $n\check{}1$ in the denominator in the variance formula, instead of n, leading into the concept of *degrees of freedom*. It is based on the premise that you want to make estimates about a population, based on a sample.

If you use the intuitive denominator of n in the variance formula, you will underestimate the true value of the variance and the standard deviation in the population. This is referred to as a *biased estimate*. However, if you divide by $n\check{}1$ instead of n, the variance becomes an *unbiased estimate*.

To fully explain why using n leads to a *biased estimate* involves the notion of *degrees of freedom*, which takes into account the number of constraints in computing an estimate. In this case, there are $n\check{}1$ degrees of freedom since there is one constraint: the standard deviation depends on calculating the sample mean.

Standard deviation

The square root of the variance.

```r
# Standard deviation
sd(state[['Population']])
```

```
## [1] 6848235
```

Similarly, the `sd()` function in base R calculates the sample standard deviation.

$$\text{Standard deviation} = s = \sqrt{\text{Variance}}$$

We can write function to calculate the population mean, variance and standard deviation. The following is an example:

```r
# Functions return population mean, variance and standard deviation
mvsdp <- function(x)
{
  # x is the vector of values
  n   <- length(x)       # recover size
  mux <- sum(x)/n        # mean
  ex2 <- sum(x^2)/n      # E(X^2)
  s2x <- ex2 - mux^2     # alternative variance
  sdx <- sqrt(s2x)       # SD
  return(c(mux,s2x,sdx)) # return the triple
}
```

```
# Calculate the population triple for the population
mvsdp(state[['Population']])
```

```
## [1] 6.162876e+06 4.596036e+13 6.779407e+06
```

We can get the sample estimates by minor modification of the code:

```
# Functions return sample mean, variance and standard deviation
mvsds <- function(x)
{
  # x is the vector of values
  n    <- length(x)        # recover size
  mux  <- sum(x)/n         # mean
  ex2  <- sum(x^2)/n       # E(X^2)
  s2x  <- ex2 - mux^2      # alternative variance
  s2xs <- s2x*(n/(n-1))    # sample variance
  sdxs<-  sqrt(s2xs)       # SD
  return(c(mux,s2xs,sdxs)) # return the triple
}
```

```
# Calculate the sample triple for the population
mvsds(state[['Population']])
```

```
## [1] 6.162876e+06 4.689833e+13 6.848235e+06
```

These results match the result by the built-in functions in base *R*.

Mean absolute deviation (l1-norm, Manhattan norm)

The mean of the absolute values of the deviations from the mean.

The mean absolute deviation is computed with the formula:

$$\text{Mean absolute deviation} = \frac{\sum_{i=1}^{n} |x_i - \overline{x}|}{n}$$

where $\overline{x}$ is the sample mean.

Median absolute deviation from the median (MAD)

The median of the absolute values of the deviations from the median.

Neither the *variance*, the *standard deviation*, nor the *mean absolute deviation* is robust to outliers and extreme values. The *variance* and *standard deviation* are especially sensitive to outliers since they are based on the squared deviations. A robust estimate of variability is the *median absolute deviation from the median* or **MAD**:

$$\text{Median absolute deviation} = \text{Median}(|x_1 - m|, |x_2 - m|, ..., |x_N - m|)$$

where m is the median. Like the median, the MAD is not influenced by extreme values.

```
# Mean absolute deviation from the median (MAD)
mad(state[['Population']])
```

```
## [1] 3849870
```

Order statistics (ranks)

Metrics based on the data values sorted from smallest to biggest.

Range

The difference between the largest and the smallest value in a data set.

The minimum and maximum values themselves are useful to know and are helpful in identifying outliers, but the range is extremely sensitive to outliers and not very useful as a general measure of dispersion in the data.

Percentile (quantile)

To avoid the sensitivity to outliers, we can look at the range of the data after dropping values from each end. Formally, these types of estimates are based on differences between *percentiles*.

The value such that P percent of the values take on this value or less and $(100 - P)$ percent take on this value or more.

Interquantile range (IQR)

A common measurement of variability is the difference between the 25th percentile and the 75th percentile, called the *interquartile range* (or IQR).

```
# Interquartile range (IQR)
IQR(state[['Population']])
```

```
## [1] 4847308
```

Exploring the data distribution

Percentiles

Percentiles are valuable for summarizing the entire distribution. It is common to report the quartiles (25th, 50th, and 75th percentiles) and the deciles (the 10th, 20th, ..., 90th percentiles). Percentiles are especially valuable for summarizing the tails (the outer range) of the distribution. Popular culture has coined the term one-percenters to refer to the people in the top 99th percentile of wealth.

The following *R* code produce the percentiles of the murder rate by state.

```
# Percentiles of the murder rate by state
quantile(state[['Murder.Rate']], p=c(.05, .25, .5, .75, .95))
```

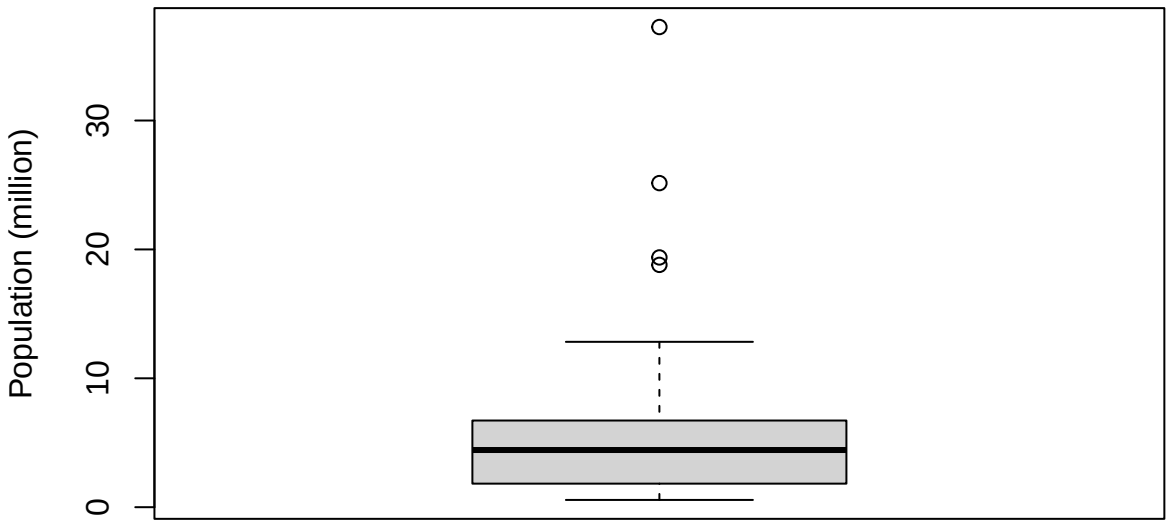
```
##    5%   25%   50%   75%   95%
## 1.600 2.425 4.000 5.550 6.510
```

The median is 4 murders per 100,000 people, although there is quite a bit of variability: the 5th percentile is only 1.6 and the 95th percentile is 6.51.

Boxplot (box and whiskers plot)

Boxplots are based on percentiles and give a quick way to visualize the distribution of data.

```
# Boxplot of the population by state
boxplot(state[['Population']]/1000000, ylab='Population (million)')
```



The top and bottom of the box are the 75th and 25th percentiles, respectively. The median is shown by the horizontal line in the box. The dashed lines, referred to as *whiskers*, extend from the top and bottom of the box to indicate the range for the bulk of the data. By default, the *R* function extends the *whiskers* to the furthest point beyond the box, except that it will not go beyond 1.5 times the IQR. Any data outside of the *whiskers* is plotted as single points or circles (often considered outliers).

Frequency table

A tally of the count of numeric data values that fall into a set of intervals (bins).

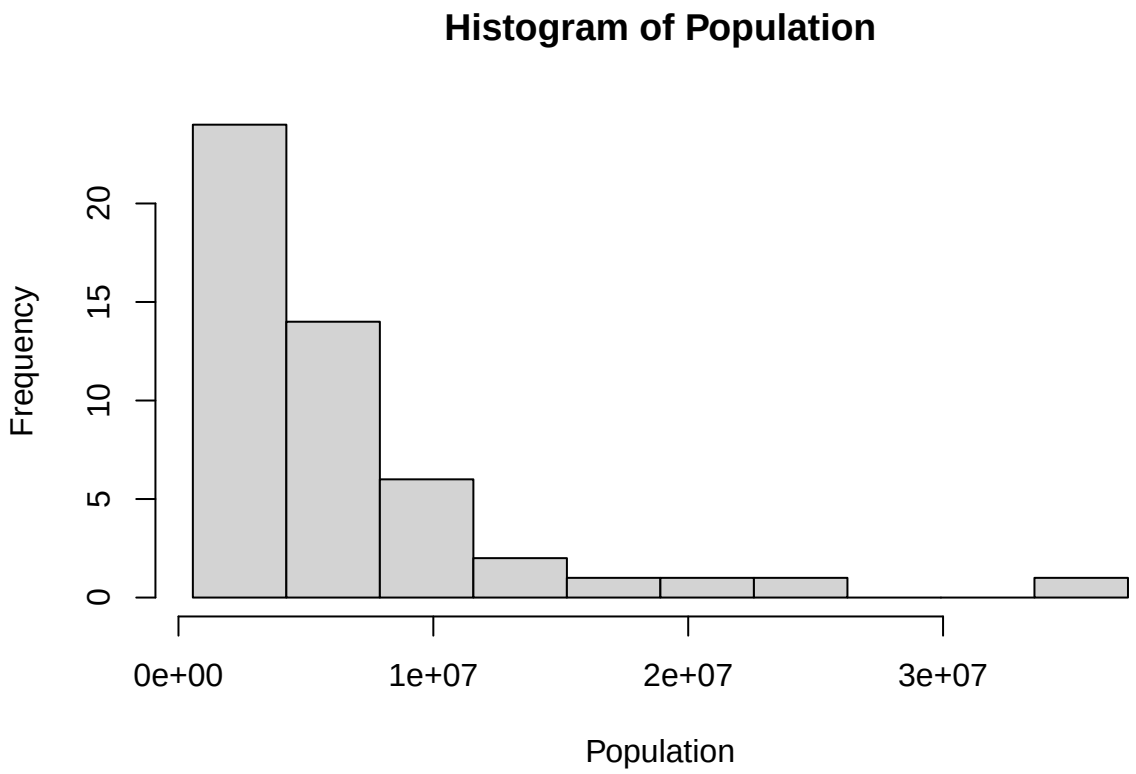
```
# Frequency table of the population by state
breaks <- seq(from=min(state[['Population']]),
              to=max(state[['Population']]), length=11)
pop_freq <- cut(state[['Population']], breaks=breaks,
                right=TRUE, include.lowest=TRUE)
table(pop_freq)
```

```
## pop_freq
## [5.64e+05,4.23e+06]  (4.23e+06,7.9e+06]  (7.9e+06,1.16e+07] (1.16e+07,1.52e+07]
##                  24                   14                   6                   2
## (1.52e+07,1.89e+07] (1.89e+07,2.26e+07] (2.26e+07,2.62e+07] (2.62e+07,2.99e+07]
##                   1                    1                   1                   0
## (2.99e+07,3.36e+07] (3.36e+07,3.73e+07]
##                   0                   1
```

Histogram

A plot of the frequency table with the bins on the x-axis and the count (or proportion) on the y-axis. While visually similar, bar charts should not be confused with histograms.

```
# Histogram
hist(state[['Population']], breaks=breaks, xlab='Population', main='Histogram of Population')
```



Density plot

A smoothed version of the histogram, often based on a *kernel density estimate*. A density plot shows the distribution of data values as a continuous line.

In *R*, you can compute a density estimate using the density function:

```
# Density estimate superposed on a histogram
hist(state[['Murder.Rate']], freq=FALSE, main='Density of state muder rates',
     ylab='Density', xlab='Murder Rate (per 100,000)')
lines(density(state[['Murder.Rate']]))
```