Week 5: Relative standing and random variables

Stat 201: Statistics I

Measures of Relative Standing and Boxplots

Z-scores

A z-score for a given value is value minus mean divided by standard deviation. We can always execute this calculation "by hand."

```
# What is the z-score for 30 mpg?
x <- 30

z <- (x - mean(mtcars$mpg)) / sd(mtcars$mpg)   # Notice the parentheses
z
```

```
## [1] 1.644178
```

There is an R function to do this more directly, but it is not much easier to use and may be harder to remember. I'll give it here for reference.

```
z.2 <- scale(x, mean(mtcars$mpg), sd(mtcars$mpg))
z.2[1,1]
```

```
## [1] 1.644178
```

The find the reverse, to find a value in the units of the data set corresponding to a given z-score, again the simplest method is to just do the calculation "by hand".

```
(z * sd(mtcars$mpg)) + mean(mtcars$mpg)
```

```
## [1] 30
```

Percentiles

To find values which correspond to a particular percentile (or multiple percentiles), we use the `quantile()` function.

```
# Find the 30th percentile of MPG
quantile(mtcars$mpg, .3)
```

```
##   30%
## 15.98
```

The reverse, finding the percentile of a value, is trickier. We need to employ a slightly advanced technique of calling an R function which will give us a function which we then can use to get our percentile.

```
# What is the percentile of 24.4 MPG
pc <- ecdf(mtcars$mpg)(24.4)

# To display as a percentile, as the book defines it,
#   multiply by 100 and round up
ceiling(pc*100)
```

```
## [1] 82
```

Quartiles and 5 number summary

Since quartiles are just particular percentiles, we already know how to find them. We just need to give the `quantile` function a vector of percentiles to find.

```
# Find Q1, Q2 and Q3
quantile(mtcars$mpg, c(.25, .50, .75))
```

```
##    25%    50%    75%
## 15.425 19.200 22.800
```

We can use the same method to produce a 5 number summary.

```
quantile(mtcars$mpg, c(0, .25, .50, .75, 1))
```

```
##     0%    25%    50%    75%   100%
## 10.400 15.425 19.200 22.800 33.900
```

Alternatively, we can use the `summary()` function. This is another general purpose function which will attempt to provide meaningful information on anything that is passed to it. Results are not always as useful as one would hope, but if we give it a vector of quantitative data, it will return the 5 number summary plus the mean.

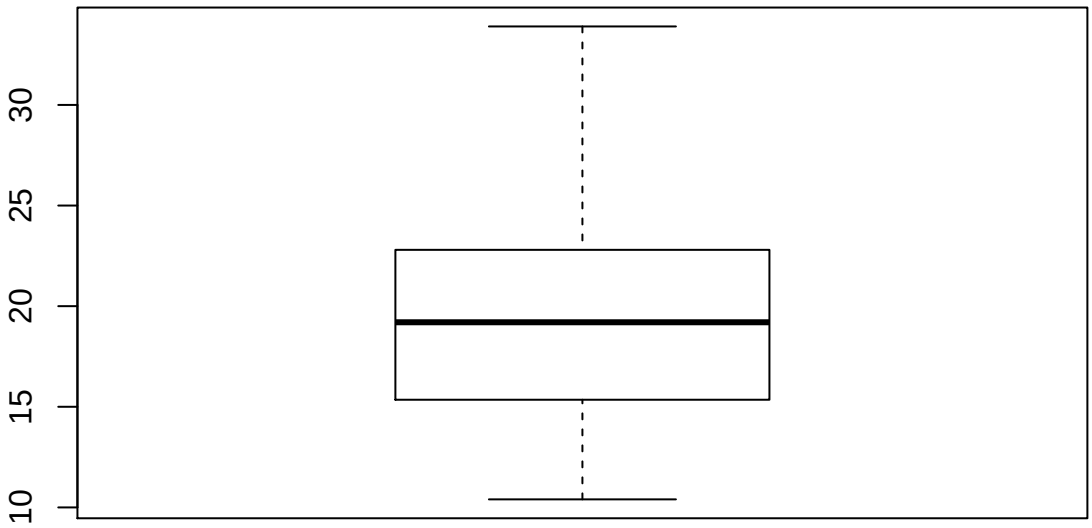
```
summary(mtcars$mpg)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   10.40   15.43   19.20   20.09   22.80   33.90
```

Boxplots

Boxplots are produced with the `boxplot()` function. By default, vertical plots are produced.

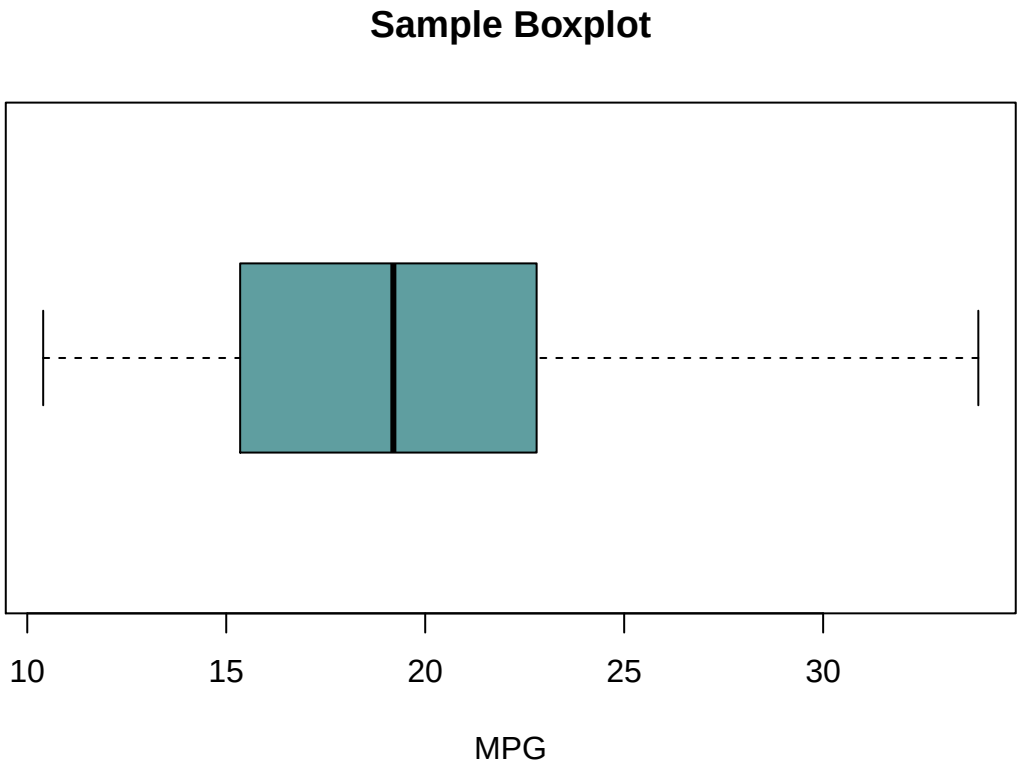
```
boxplot(mtcars$mpg)
```



However, we can draw a horizontal boxplot and make it a little prettier by adding some parameters to the `boxplot()` function.

```
boxplot(mtcars$mpg,
        horizontal = TRUE,    # Draw a horizontal plot
        col = "cadetblue",
```

```
        main = "Sample Boxplot",
        xlab = "MPG")
```



Probability Distributions

To work with probability distributions, we need to create a table (or in R terms a data frame) that contains the possible values and their associated probabilities.

```
prob.dist <- data.frame(x=0:5, P.x.=c(0.03, 0.13, 0.25, 0.34, 0.16, 0.09))
prob.dist
```

```
##   x P.x.
## 1 0 0.03
## 2 1 0.13
## 3 2 0.25
## 4 3 0.34
## 5 4 0.16
## 6 5 0.09
```

The first step is to determine if this is really a probability distribution, if the probabilities add to 1.

```
sum(prob.dist$P.x.)
```

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

Now that we've verified that we are indeed working with a true distribution, we can calculate mean and standard deviation. There are no R functions expressly for this purpose. However, as we noted that the mean of a distribution is merely the weighted mean of the values with the probabilities as weights, R does have a function for that.

```r
pd.mean <- weighted.mean(prob.dist$x, prob.dist$P.x.)
pd.mean
```

```
## [1] 2.74
```

Standard deviation will take a little more effort. Variance is the weighted mean of the difference from the mean squared, again with probabilities as weights. And standard deviation is the squared root of variance.

```r
# Find variance
pd.var <- weighted.mean((prob.dist$x - pd.mean)^2, prob.dist$P.x.)
pd.var
```

```
## [1] 1.4924
```

```r
# SD is square root of variance
pd.sd <- sqrt(pd.var)
pd.sd
```

```
## [1] 1.221638
```