Week 4: Summary statistics in R

Stat 201: Statistics I

Frequency Distributions

Factors

Frequency tables can be built for either categorical or quantitative data. Looking at the structure of `mtcars`, we can see that all of the variables (columns) are numeric. However, looking at the data set documentation, there are some which should probably be treated as categorical, such as `cyl` (number of cylinders), `am` (transmission type) and `gear` (number of forward gears). While sometimes we can leave such variables as they are, there are many R functions which work with categorical variables that expect the variables to have a specific data type. The data type for a categorical variable is a factor. Luckily, it is easy to convert numeric data to a factor.

```
cyl.fac <- as.factor(mtcars$cyl)

str(cyl.fac)
```

```
##  Factor w/ 3 levels "4","6","8": 2 2 1 2 3 2 3 1 1 2 ...
```

Now `str` tells us we have a factor with 3 levels (possible values). The data for this variable is now stored as 1, 2 or 3, corresponding to the first level, the second level, etc. If we display the data, however, the level labels will be printed.

```
cyl.fac
```

```
##  [1] 6 6 4 6 8 6 8 4 4 6 6 8 8 8 8 8 8 4 4 4 4 8 8 8 8 4 4 4 8 6 8 4
## Levels: 4 6 8
```

Frequency tables

To build a frequency table from a factor is simple. We use the `table()` function.

```
table(cyl.fac)
```

```
## cyl.fac
##  4  6  8
## 11  7 14
```

Frequency tables of quantitative variables are a little more complicated. If we try just using the table function on a quantitative variable, we will get the frequency of each unique value in the data set.

```
table(mtcars$hp)
```

```
##
##  52  62  65  66  91  93  95  97 105 109 110 113 123 150 175 180 205 215
##   1   1   1   2   1   1   1   1   1   1   3   1   2   2   3   3   1   1
## 230 245 264 335
##   1   2   1   1
```

In order to get a frequency table as we expect, with the variable values separated into classes or ranges of values, we are going to need to first create a factor representing those classes using the `cut()` function and then build a table from that factor.

```
# Divide hp into 5 classes
hp.cls <- cut(mtcars$hp, 5)

hp.tab <- table(hp.cls)
hp.tab
```

```
## hp.cls
## (51.7,109]  (109,165]  (165,222]  (222,278]  (278,335]
##         10          9          8          4          1
```

If we want the table to look more like what is presented in the book, we can put the results in a data frame.

```
hp.ft <- data.frame(hp.tab)
hp.ft
```

```
##       hp.cls Freq
## 1 (51.7,109]   10
## 2  (109,165]    9
## 3  (165,222]    8
## 4  (222,278]    4
## 5  (278,335]    1
```

Notice when we display the data frame, we no long have row numbers along the left side. In this case they have been replaced by row names, which were generated by the `table()` function passed to the data frame. To see the row names of any data object, say `x`, call `rownames(x)`.

We can add relative frequencies and cumulative frequencies by doing a little math.

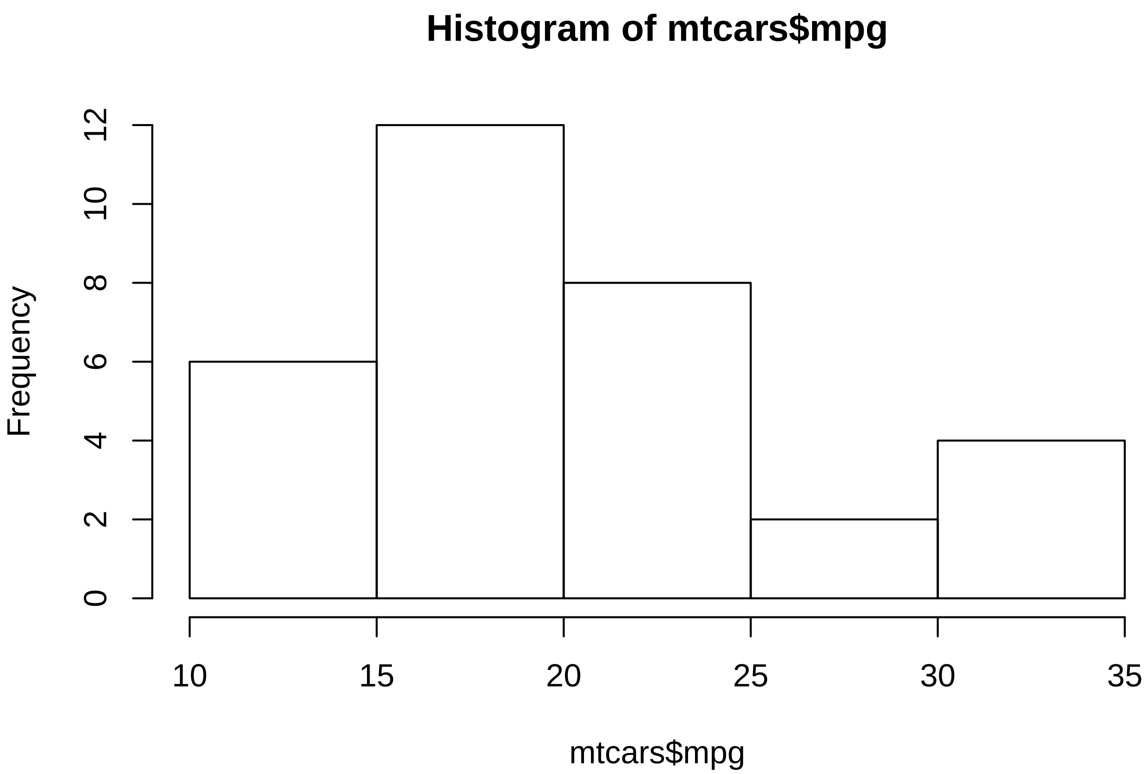
```
# To add a column to a data frame, simply assign a vector to a
#   named column as though it already existed.

# Relative frequency is class count / total count
hp.ft$rel.freq <- hp.ft$Freq/sum(hp.ft$Freq)

# Function cumsum returns a vector of cumulative counts
hp.ft$cum.freq <- cumsum(hp.ft$Freq)

hp.ft
```

```
##       hp.cls Freq rel.freq cum.freq
## 1 (51.7,109]   10  0.31250       10
## 2  (109,165]    9  0.28125       19
## 3  (165,222]    8  0.25000       27
## 4  (222,278]    4  0.12500       31
## 5  (278,335]    1  0.03125       32
```

Histograms

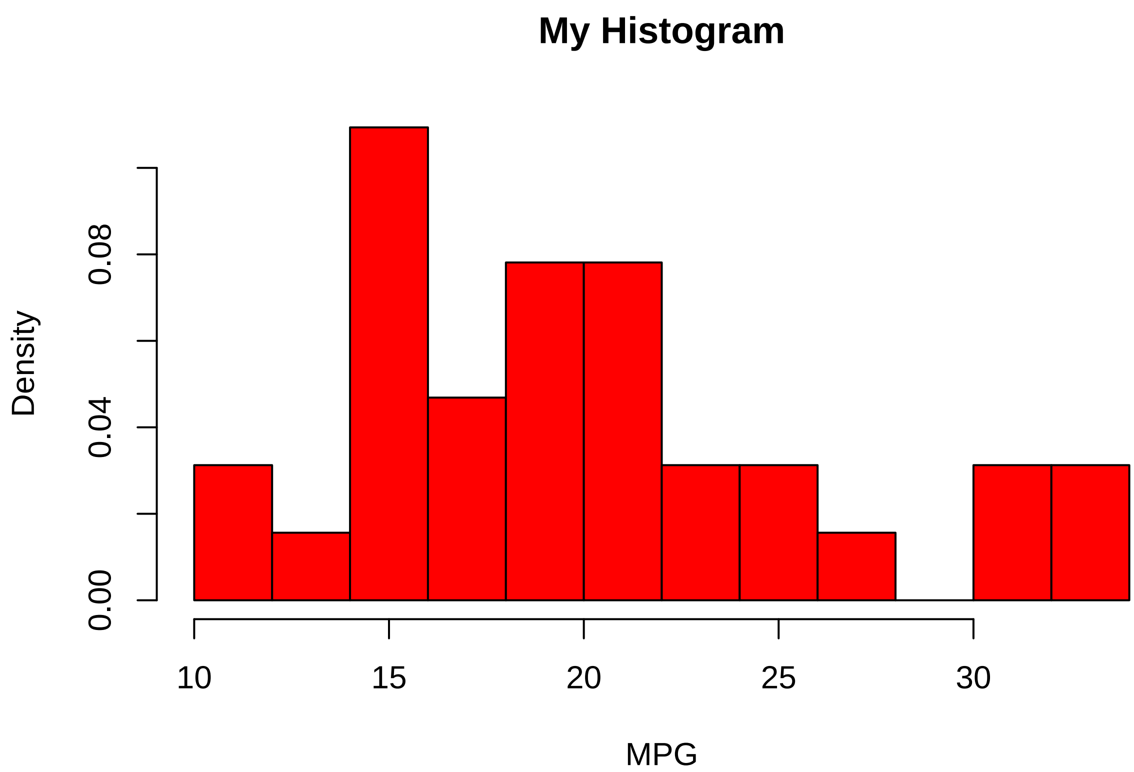
The function to create a histogram is `hist()`. It expects a numeric vector.

```
hist(mtcars$mpg)
```



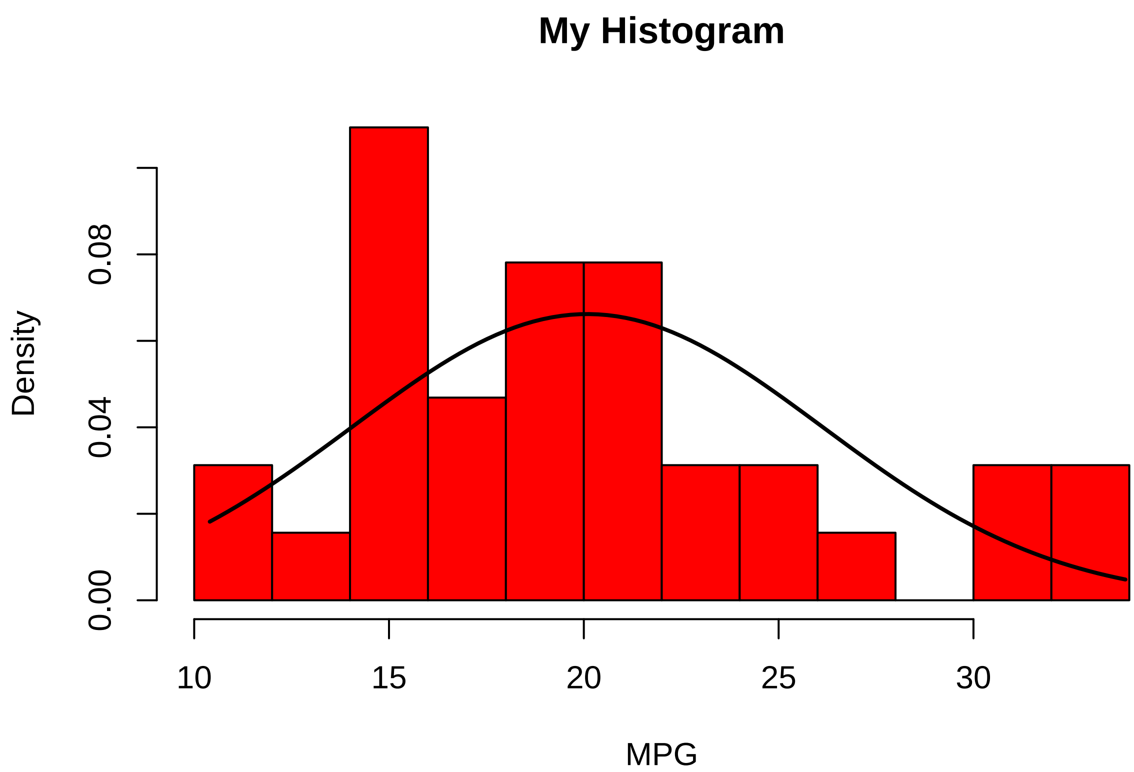
If we just want to get a look at the data distribution, that's all we really need. However, if we want to produce graphs which will be seen by others, we can clean it up a bit.

```
hist(mtcars$mpg,
     breaks = 10,           # Number of classes, R treats this as a suggestion
     probability = TRUE,    # Display relative frequencies on y-axis
     main = "My Histogram", # Main title
     xlab = "MPG",          # X-axis title
     col = "red"            # Bar color
     )
```



To compare the shape to a normal distribution, we can draw a normal curve over it.

```
# These functions will be discussed in later chapters
x.values <- seq(min(mtcars$mpg), max(mtcars$mpg), len=100)
norm.values <- dnorm(x.values, mean(mtcars$mpg), sd(mtcars$mpg))

hist(mtcars$mpg, breaks = 10, probability = TRUE, main = "My Histogram",
     xlab = "MPG", col = "red")

lines(x.values, norm.values, lwd=2)
```



Summary statistics

Measures of Center

Mean

We have already seen the `mean()` function.

```
mean(mtcars$mpg)
```

```
## [1] 20.09062
```

Median

The function for medians is simply `median()`

```
median(mtcars$mpg)
```

```
## [1] 19.2
```

Mode

There is not a built-in R function for mode. For many data sets it might be easiest to figure the mode by just examining the data or producing a frequency table. As above, we can create frequency tables with the table function.

```r
# Find the mode of categorical variable
table(mtcars$cyl)
```

```
##
##  4  6  8
## 11  7 14
```

We can easily see that the mode for cylanders is 8. When looking for the mode of a quantitative variable, we don't want to "cut" the values into classes. We can make a table with the variable directly.

```r
table(mtcars$hp)
```

```
##
##  52  62  65  66  91  93  95  97 105 109 110 113 123 150 175 180 205 215
##   1   1   1   2   1   1   1   1   1   1   3   1   2   2   3   3   1   1
## 230 245 264 335
##   1   2   1   1
```

The resulting table from a numeric variable is often harder to read because of the many values. We can help ourselves by sorting the table before displaying it.

```r
# The table returned by the table() function is passed directly
#   into the sort function, which returns a sorted table.
sort(table(mtcars$hp))
```

```
##
##  52  62  65  91  93  95  97 105 109 113 205 215 230 264 335  66 123 150
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   2   2   2
## 245 110 175 180
##   2   3   3   3
```

Now, it is clear that there are three modes, 110, 175, and 180, each with a frequency of 3.

Midrange

R also lacks a function for midrange, but it is easily calculated. Remember, midrange is halfway between the minimum and maximum values, which we calculate as the mean of those values.

```r
mean(c(min(mtcars$mpg), max(mtcars$mpg)))
```

```
## [1] 22.15
```

Measures of Variation

Range

The `range()` function does not provide the range as we expect it here. It returns instead the minimum and maximum values of the vector provided. Range, as a measure of variation, is the difference of those values.

```r
diff(range(mtcars$mpg))
```

```
## [1] 23.5
```

Variance and standard deviation

R does have functions for these important and widely used statistics. They are `var()` and `sd()`.

```
var(mtcars$mpg)
```

```
## [1] 36.3241
```

and…

```
sd(mtcars$mpg)
```

```
## [1] 6.026948
```