Introduction to data

Some define Statistics as the field that focuses on turning information into knowledge. The first step in that process is to summarize and describe the raw information - the data. In this lab, you will gain insight into public health by generating simple graphical and numerical summaries of a data set collected by the Centers for Disease Control and Prevention (CDC). As this is a large data set, along the way you'll also learn the indispensable skills of data processing and subsetting.

Getting started

The Behavioral Risk Factor Surveillance System (BRFSS) is an annual telephone survey of 350,000 people in the United States. As its name implies, the BRFSS is designed to identify risk factors in the adult population and report emerging health trends. For example, respondents are asked about their diet and weekly physical activity, their HIV/AIDS status, possible tobacco use, and even their level of healthcare coverage. The BRFSS Web site (http://www.cdc.gov/brfss) contains a complete description of the survey, including the research questions that motivate the study and many interesting results derived from the data.

We will focus on a random sample of 20,000 people from the BRFSS survey conducted in 2000. While there are over 200 variables in this data set, we will work with a small subset.

We begin by loading the data set of 20,000 observations into the R workspace. After launching RStudio, enter the following command.

```
source("more/cdc.R")
```

The data set `cdc` that shows up in your workspace is a *data matrix*, with each row representing a *case* and each column representing a *variable*. R calls this data format a *data frame*, which is a term that will be used throughout the labs.

To view the names of the variables, type the command

```
names(cdc)
```

```
## [1] "genhlth"  "exerany"  "hlthplan" "smoke100" "height"   "weight"
## [7] "wtdesire" "age"      "gender"
```

This returns the names `genhlth`, `exerany`, `hlthplan`, `smoke100`, `height`, `weight`, `wtdesire`, `age`, and `gender`. Each one of these variables corresponds to a question that was asked in the survey. For example, for `genhlth`, respondents were asked to evaluate their general health, responding either excellent, very good, good, fair or poor. The `exerany` variable indicates whether the respondent exercised in the past month (1) or did not (0). Likewise, `hlthplan` indicates whether the respondent had some form of health coverage (1) or did not (0). The `smoke100` variable indicates whether the respondent had smoked at least 100 cigarettes in her lifetime. The other variables record the respondent's `height` in inches, `weight` in pounds as well as their desired weight, `wtdesire`, `age` in years, and `gender`.

1. How many cases are there in this data set? How many variables? For each variable, identify its data type (e.g. categorical, discrete).

   There are 20,000 cases and 9 variables in this dataset.

   -genhlth: categorical, ordinal
   -exerany: categorical, nominal
   -hlthplan: categorical, nominal

-smoke100: categorical, nominal
-height: numerical, discrete
-weight: numerical, discrete
-wtdesire: numerical, discrete
-age: numerical, discrete
-gender: categorical, nominal

We can have a look at the first few entries (rows) of our data with the command

```
head(cdc)
```

```
##     genhlth exerany hlthplan smoke100 height weight wtdesire age gender
## 1      good       0        1        0     70    175      175  77      m
## 2      good       0        1        1     64    125      115  33      f
## 3      good       1        1        1     60    105      105  49      f
## 4      good       1        1        0     66    132      124  42      f
## 5 very good       0        1        0     61    150      130  55      f
## 6 very good       1        1        0     64    114      114  55      f
```

and similarly we can look at the last few by typing

```
tail(cdc)
```

```
##         genhlth exerany hlthplan smoke100 height weight wtdesire age
## 19995      good       0        1        1     69    224      224  73
## 19996      good       1        1        0     66    215      140  23
## 19997 excellent       0        1        0     73    200      185  35
## 19998      poor       0        1        0     65    216      150  57
## 19999      good       1        1        0     67    165      165  81
## 20000      good       1        1        1     69    170      165  83
##       gender
## 19995      m
## 19996      f
## 19997      m
## 19998      f
## 19999      f
## 20000      m
```

You could also look at *all* of the data frame at once by typing its name into the console, but that might be unwise here. We know `cdc` has 20,000 rows, so viewing the entire data set would mean flooding your screen. It's better to take small peeks at the data with `head`, `tail` or the subsetting techniques that you'll learn in a moment.

Summaries and tables

The BRFSS questionnaire is a massive trove of information. A good first step in any analysis is to distill all of that information into a few summary statistics and graphics. As a simple example, the function `summary` returns a numerical summary: minimum, first quartile, median, mean, second quartile, and maximum. For `weight` this is

```
summary(cdc$weight)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    68.0   140.0   165.0   169.7   190.0   500.0
```

R also functions like a very fancy calculator. If you wanted to compute the interquartile range for the respondents' weight, you would look at the output from the summary command above and then enter

```
190 - 140
```

```
## [1] 50
```

R also has built-in functions to compute summary statistics one by one. For instance, to calculate the mean, median, and variance of `weight`, type

```
mean(cdc$weight)
```

```
## [1] 169.683
```

```
var(cdc$weight)
```

```
## [1] 1606.484
```

```
median(cdc$weight)
```

```
## [1] 165
```

While it makes sense to describe a quantitative variable like `weight` in terms of these statistics, what about categorical data? We would instead consider the sample frequency or relative frequency distribution. The function `table` does this for you by counting the number of times each kind of response was given. For example, to see the number of people who have smoked 100 cigarettes in their lifetime, type

```
table(cdc$smoke100)
```

```
##
##     0     1
## 10559  9441
```

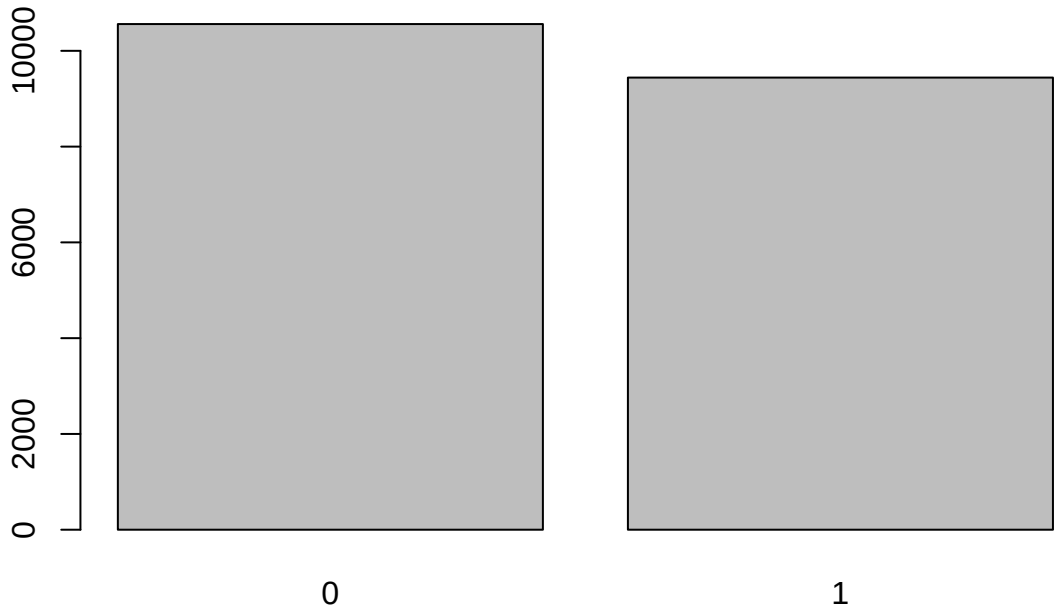
or instead look at the relative frequency distribution by typing

```
table(cdc$smoke100)/20000
```

```
##
##       0       1
## 0.52795 0.47205
```

Notice how R automatically divides all entries in the table by 20,000 in the command above. This is similar to something we observed in the Introduction to R; when we multiplied or divided a vector with a number, R applied that action across entries in the vectors. As we see above, this also works for tables. Next, we make a bar plot of the entries in the table by putting the table inside the `barplot` command.

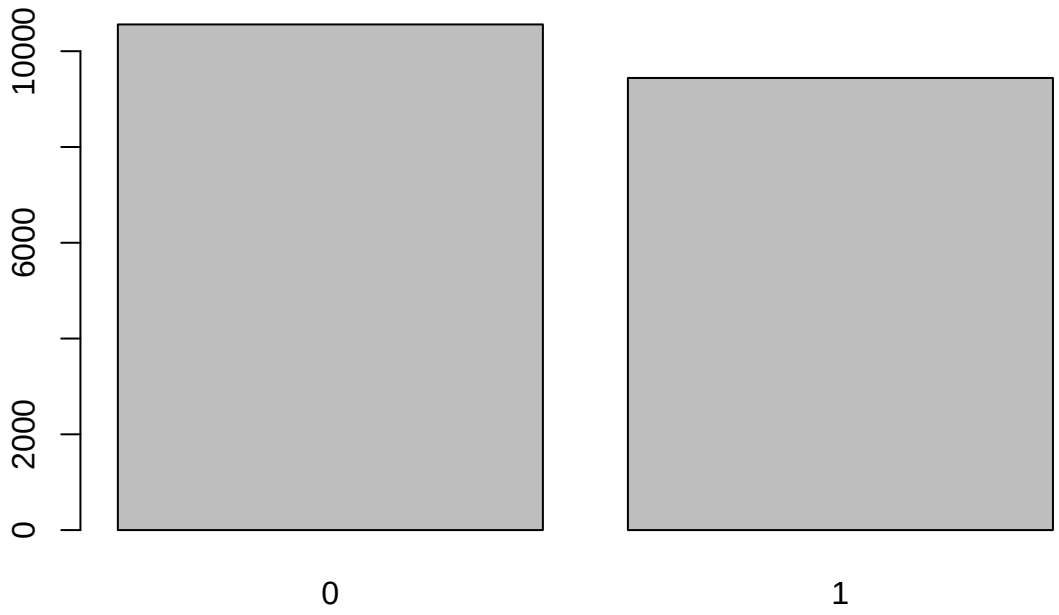
```
barplot(table(cdc$smoke100))
```



Notice what we've done here! We've computed the table of `cdc$smoke100` and then immediately applied the graphical function, `barplot`. This is an important idea: R commands can be nested. You could also break this into two steps by typing the following:

```
smoke <- table(cdc$smoke100)

barplot(smoke)
```



Here, we've made a new object, a table, called `smoke` (the contents of which we can see by typing `smoke` into the console) and then used it in as the input for `barplot`. The special symbol `<-` performs an *assignment*, taking the output of one line of code and saving it into an object in your workspace. This is another important idea that we'll return to later.

2. Create a numerical summary for `height` and `age`, and compute the interquartile range for each. Compute the relative frequency distribution for `gender` and `exerany`. How many males are in the sample? What proportion of the sample reports being in excellent health?

numerical summary for height

```
summary(cdc$height)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   48.00   64.00   67.00   67.18   70.00   93.00
```

numerical summary for age

```
summary(cdc$age)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   18.00   31.00   43.00   45.07   57.00   99.00
```

interquartile range for height

```
70 - 64
```

```
## [1] 6
```

interquartile range for age

```
57 - 31
```

```
## [1] 26
```

relative frequency distribution for gender

```
table(cdc$gender)
```

```
##
##     m     f
##  9569 10431
```

relative frequency distribution for exerany

```
table(cdc$exerany)
```

```
##
##     0     1
##  5086 14914
```

There are 9,569 males in the sample.

To find how many in the sample report being in excellent health, we use the table function again to find the relative frequency distribution for genhlth

```
table(cdc$genhlth)
```

```
##
## excellent very good      good      fair      poor
##      4657      6972      5675      2019       677
```

It appears that 4,657 individuals report being in excellent health from the sample.

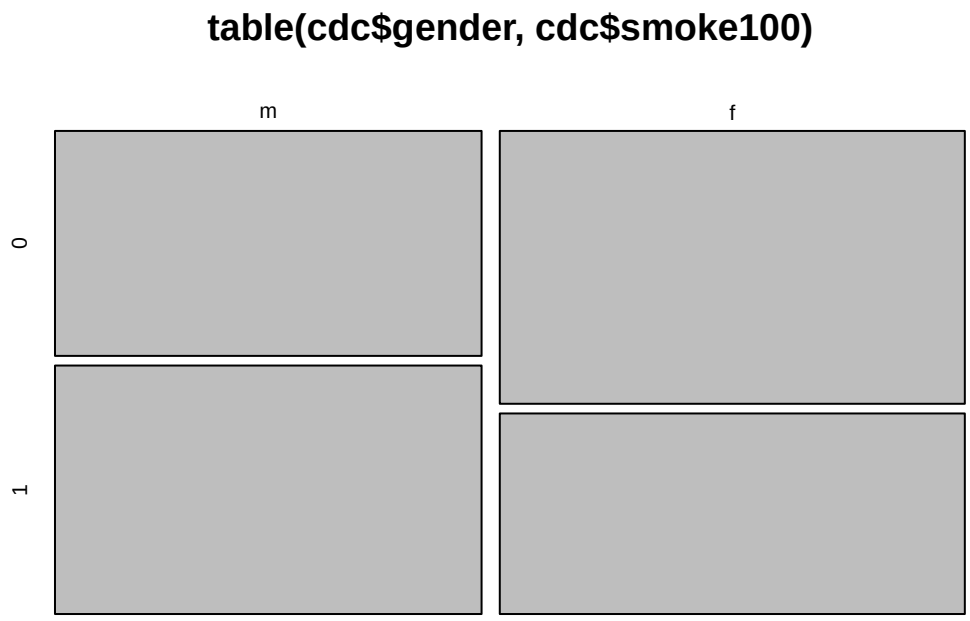
The `table` command can be used to tabulate any number of variables that you provide. For example, to examine which participants have smoked across each gender, we could use the following.

```
table(cdc$gender,cdc$smoke100)
```

```
##
##        0    1
##   m 4547 5022
##   f 6012 4419
```

Here, we see column labels of 0 and 1. Recall that 1 indicates a respondent has smoked at least 100 cigarettes. The rows refer to gender. To create a mosaic plot of this table, we would enter the following command.

```
mosaicplot(table(cdc$gender,cdc$smoke100))
```



We could have accomplished this in two steps by saving the table in one line and applying `mosaicplot` in the next (see the table/barplot example above).

3. What does the mosaic plot reveal about smoking habits and gender?

After looking at the mosaic plot, as well as calculating proportions of male smokers to male non-smokers and female smokers to female non-smokers, we can see that there is a higher proportion of male smokers than female smokers in the sample.

```
total_males <- 4547 + 5022
total_females <- 6012 + 4419

prop_male_smokers <- 5022 / total_males
prop_female_smokers <- 4419 / total_females

prop_male_smokers
```

```
## [1] 0.5248197
```

```
prop_female_smokers
```

```
## [1] 0.4236411
```

**

Interlude: How R thinks about data

We mentioned that R stores data in data frames, which you might think of as a type of spreadsheet. Each row is a different observation (a different respondent) and each column is a different variable (the first is `genhlth`, the second `exerany` and so on). We can see the size of the data frame next to the object name in the workspace or we can type

```
dim(cdc)
```

```
## [1] 20000     9
```

which will return the number of rows and columns. Now, if we want to access a subset of the full data frame, we can use row-and-column notation. For example, to see the sixth variable of the 567th respondent, use the format

```
cdc[567,6]
```

```
## [1] 160
```

which means we want the element of our data set that is in the 567th row (meaning the 567th person or observation) and the 6th column (in this case, weight). We know that `weight` is the 6th variable because it is the 6th entry in the list of variable names

```
names(cdc)
```

```
## [1] "genhlth"  "exerany"  "hlthplan" "smoke100" "height"   "weight"
## [7] "wtdesire" "age"      "gender"
```

To see the weights for the first 10 respondents we can type

```
cdc[1:10,6]
```

```
##  [1] 175 125 105 132 150 114 194 170 150 180
```

In this expression, we have asked just for rows in the range 1 through 10. R uses the `:` to create a range of values, so 1:10 expands to 1, 2, 3, 4, 5, 6, 7, 8, 9, 10. You can see this by entering

```
1:10
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

Finally, if we want all of the data for the first 10 respondents, type

```
cdc[1:10,]
```

```
##      genhlth exerany hlthplan smoke100 height weight wtdesire age gender
## 1       good       0        1        0     70    175      175  77      m
## 2       good       0        1        1     64    125      115  33      f
## 3       good       1        1        1     60    105      105  49      f
## 4       good       1        1        0     66    132      124  42      f
```

```
## 5  very good       0        1        0     61    150      130  55      f
## 6  very good       1        1        0     64    114      114  55      f
## 7  very good       1        1        0     71    194      185  31      m
## 8  very good       0        1        0     67    170      160  45      m
## 9       good       0        1        1     65    150      130  27      f
## 10      good       1        1        0     70    180      170  44      m
```

By leaving out an index or a range (we didn't type anything between the comma and the square bracket), we get all the columns. When starting out in R, this is a bit counterintuitive. As a rule, we omit the column number to see all columns in a data frame. Similarly, if we leave out an index or range for the rows, we would access all the observations, not just the 567th, or rows 1 through 10. Try the following to see the weights for all 20,000 respondents fly by on your screen

```
cdc[,6]
```

Recall that column 6 represents respondents' weight, so the command above reported all of the weights in the data set. An alternative method to access the weight data is by referring to the name. Previously, we typed `names(cdc)` to see all the variables contained in the cdc data set. We can use any of the variable names to select items in our data set.

```
cdc$weight
```

The dollar-sign tells R to look in data frame `cdc` for the column called `weight`. Since that's a single vector, we can subset it with just a single index inside square brackets. We see the weight for the 567th respondent by typing

```
cdc$weight[567]
```

```
## [1] 160
```

Similarly, for just the first 10 respondents

```
cdc$weight[1:10]
```

```
##  [1] 175 125 105 132 150 114 194 170 150 180
```

The command above returns the same result as the `cdc[1:10,6]` command. Both row-and-column notation and dollar-sign notation are widely used, which one you choose to use depends on your personal preference.

A little more on subsetting

It's often useful to extract all individuals (cases) in a data set that have specific characteristics. We accomplish this through *conditioning* commands. First, consider expressions like

```
cdc$gender == "m"
```

or

```
cdc$age > 30
```

These commands produce a series of `TRUE` and `FALSE` values. There is one value for each respondent, where `TRUE` indicates that the person was male (via the first command) or older than 30 (second command).

Suppose we want to extract just the data for the men in the sample, or just for those over 30. We can use the R function `subset` to do that for us. For example, the command

```
mdata <- subset(cdc, cdc$gender == "m")
```

will create a new data set called `mdata` that contains only the men from the `cdc` data set. In addition to finding it in your workspace alongside its dimensions, you can take a peek at the first several rows as usual

```
head(mdata)
```

```
##      genhlth exerany hlthplan smoke100 height weight wtdesire age gender
## 1       good       0        1        0     70    175      175  77      m
## 7  very good       1        1        0     71    194      185  31      m
## 8  very good       0        1        0     67    170      160  45      m
## 10      good       1        1        0     70    180      170  44      m
## 11 excellent       1        1        1     69    186      175  46      m
## 12      fair       1        1        1     69    168      148  62      m
```

This new data set contains all the same variables but just under half the rows. It is also possible to tell R to keep only specific variables, which is a topic we'll discuss in a future lab. For now, the important thing is that we can carve up the data based on values of one or more variables.

As an aside, you can use several of these conditions together with `&` and `|`. The `&` is read "and" so that

```
m_and_over30 <- subset(cdc, gender == "m" & age > 30)
```

will give you the data for men over the age of 30. The `|` character is read "or" so that

```
m_or_over30 <- subset(cdc, gender == "m" | age > 30)
```

will take people who are men or over the age of 30 (why that's an interesting group is hard to say, but right now the mechanics of this are the important thing). In principle, you may use as many "and" and "or" clauses as you like when forming a subset.

3. Create a new object called `under23_and_smoke` that contains all observations of respondents under the age of 23 that have smoked 100 cigarettes in their lifetime. Write the command you used to create the new object as the answer to this exercise.

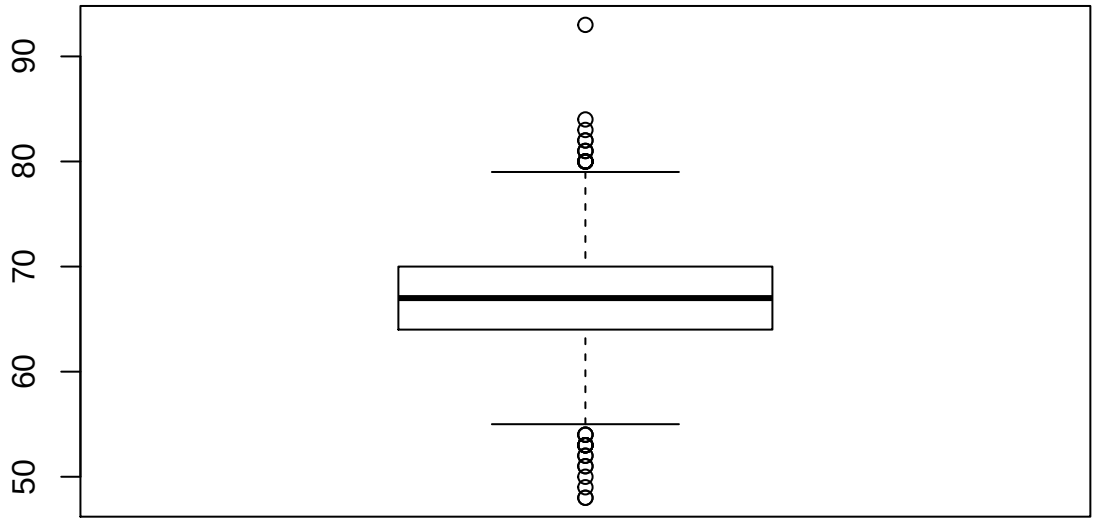
```
under23_and_smoke <- subset(cdc, age < 23 & smoke100 == 1)
head(under23_and_smoke)
```

```
##       genhlth exerany hlthplan smoke100 height weight wtdesire age gender
## 13  excellent       1        0        1     66    185      220  21      m
## 37  very good       1        0        1     70    160      140  18      f
## 96  excellent       1        1        1     74    175      200  22      m
## 180      good       1        1        1     64    190      140  20      f
## 182 very good       1        1        1     62     92       92  21      f
## 240 very good       1        0        1     64    125      115  22      f
```

Quantitative data

With our subsetting tools in hand, we'll now return to the task of the day: making basic summaries of the BRFSS questionnaire. We've already looked at categorical data such as `smoke` and `gender` so now let's turn our attention to quantitative data. Two common ways to visualize quantitative data are with box plots and histograms. We can construct a box plot for a single variable with the following command.

```
boxplot(cdc$height)
```



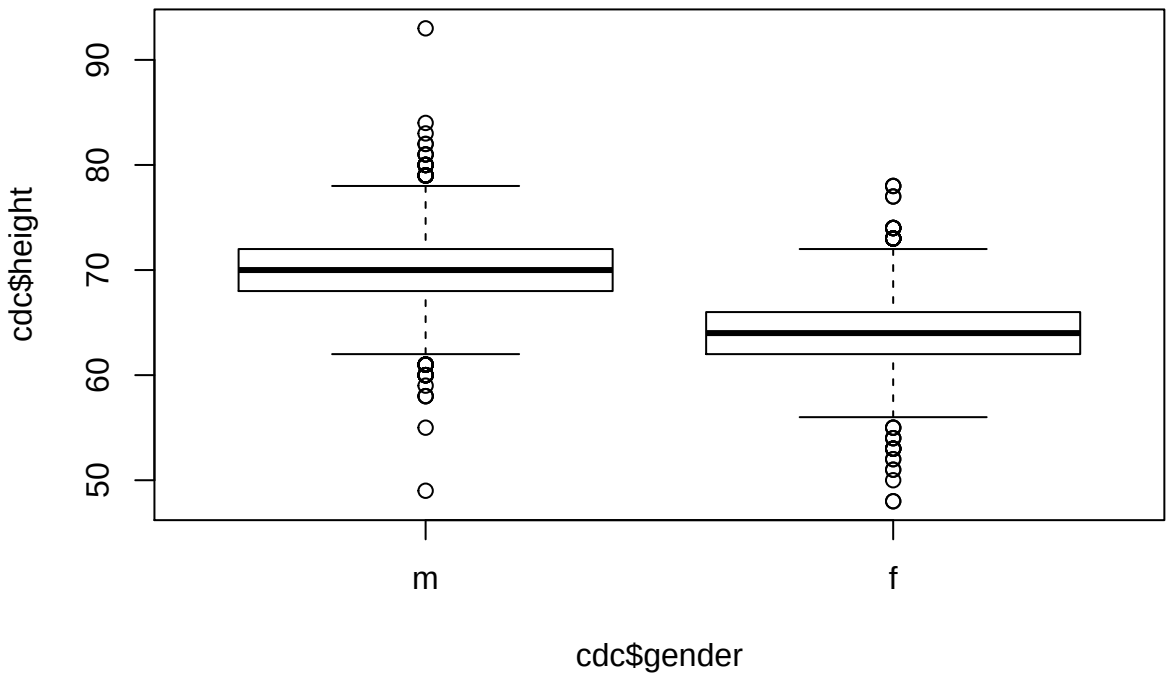
You can compare the locations of the components of the box by examining the summary statistics.

```
summary(cdc$height)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   48.00   64.00   67.00   67.18   70.00   93.00
```

Confirm that the median and upper and lower quartiles reported in the numerical summary match those in the graph. The purpose of a boxplot is to provide a thumbnail sketch of a variable for the purpose of comparing across several categories. So we can, for example, compare the heights of men and women with

```
boxplot(cdc$height ~ cdc$gender)
```



The notation here is new. The ~ character can be read *versus* or *as a function of*. So we're asking R to give us a box plots of heights where the groups are defined by gender.

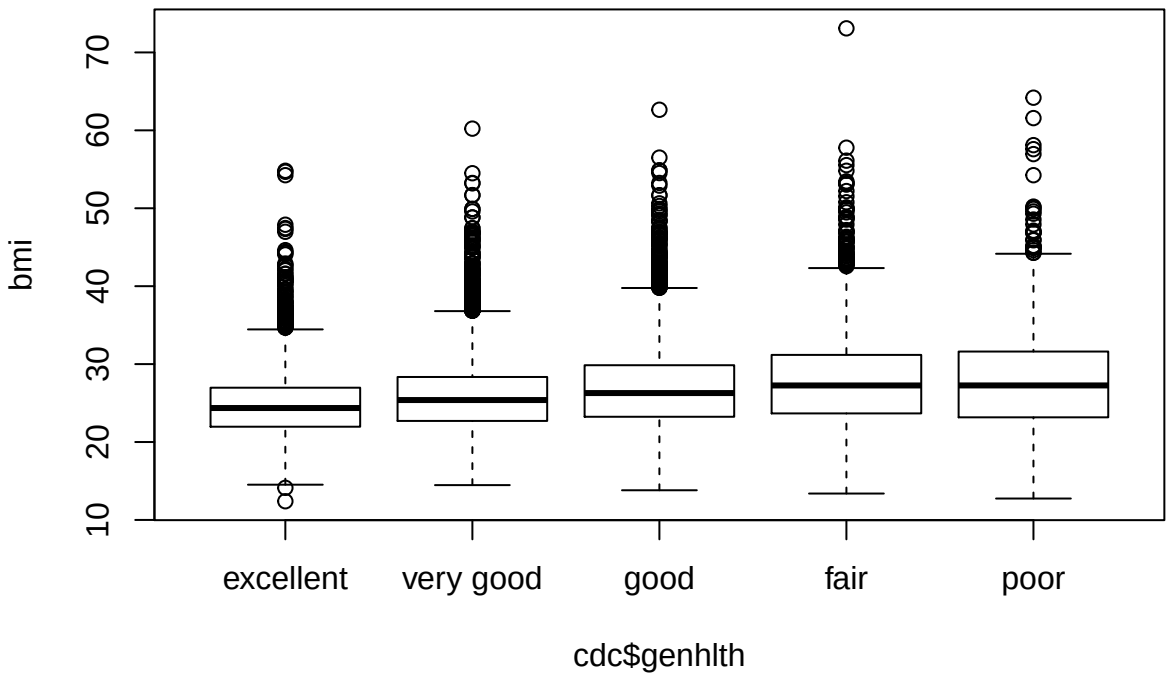
Next let's consider a new variable that doesn't show up directly in this data set: Body Mass Index (BMI) (http://en.wikipedia.org/wiki/Body_mass_index). BMI is a weight to height ratio and can be calculated as:

$$BMI = \frac{weight\ (lb)}{height\ (in)^2} * 703$$

703 is the approximate conversion factor to change units from metric (meters and kilograms) to imperial (inches and pounds).

The following two lines first make a new object called `bmi` and then creates box plots of these values, defining groups by the variable `cdc$genhlth`.

```
bmi <- (cdc$weight / cdc$height^2) * 703
boxplot(bmi ~ cdc$genhlth)
```



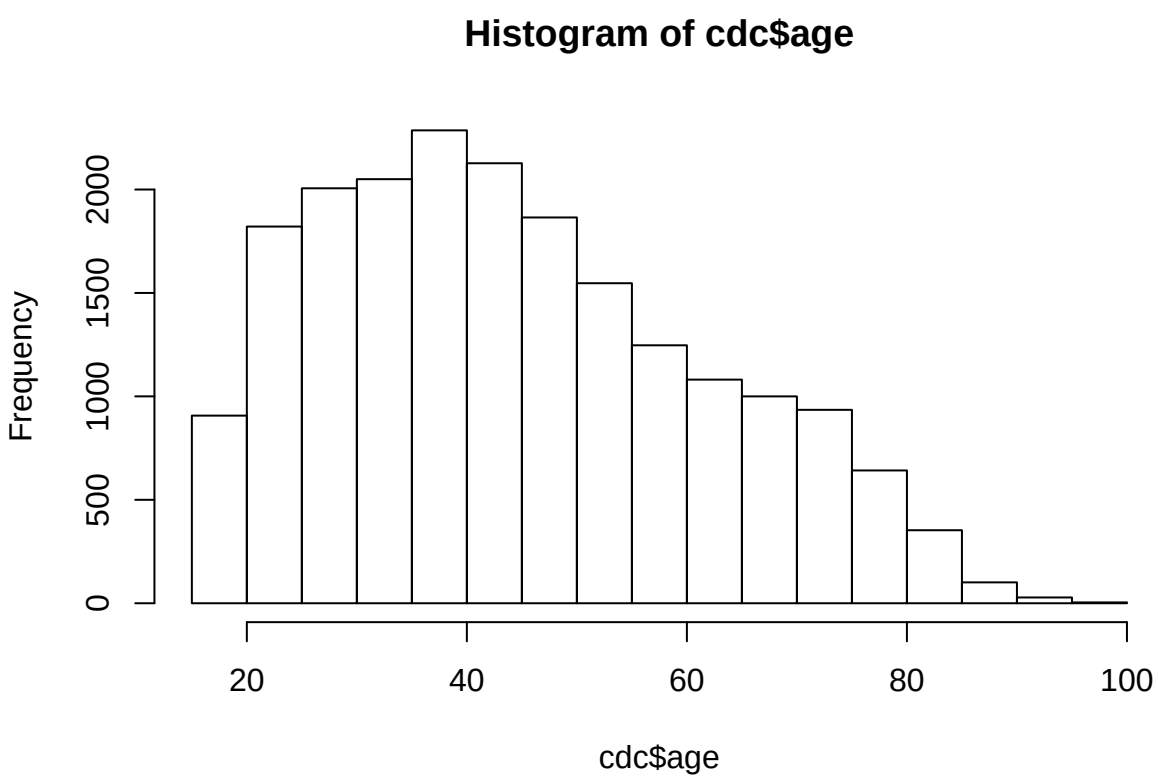
Notice that the first line above is just some arithmetic, but it's applied to all 20,000 numbers in the `cdc` data set. That is, for each of the 20,000 participants, we take their weight, divide by their height-squared and then multiply by 703. The result is 20,000 BMI values, one for each respondent. This is one reason why we like R: it lets us perform computations like this using very simple expressions.

4. What does this box plot show? Pick another categorical variable from the data set and see how it relates to BMI. List the variable you chose, why you might think it would have a relationship to BMI, and indicate what the figure seems to suggest.

   The box plot shows that median BMI trends lower for those that responded as having better health. There also appears to be smaller interquartile ranges for those in the sample that responded with better health, than those that responded as having poorer health.

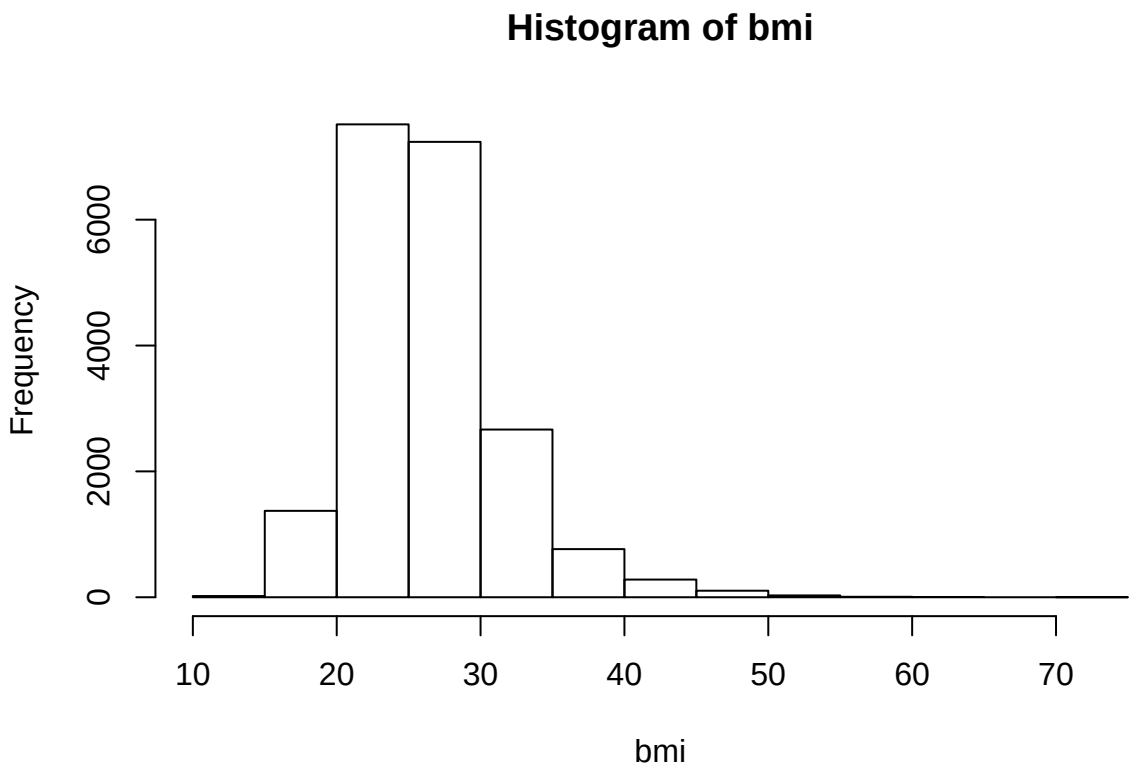
Finally, let's make some histograms. We can look at the histogram for the age of our respondents with the command

```
hist(cdc$age)
```

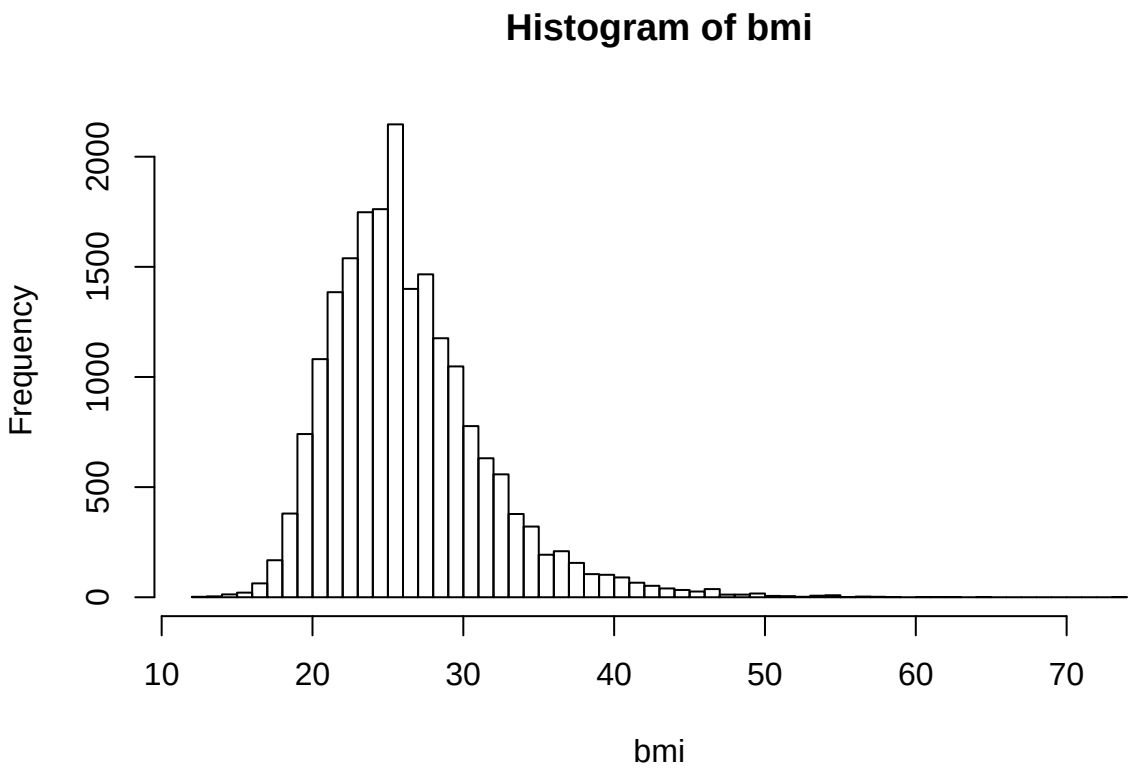


Histograms are generally a very good way to see the shape of a single distribution, but that shape can change depending on how the data is split between the different bins. You can control the number of bins by adding an argument to the command. In the next two lines, we first make a default histogram of `bmi` and then one with 50 breaks.

```
hist(bmi)
```




```
hist(bmi, breaks = 50)
```



Note that you can flip between plots that you've created by clicking the forward and backward arrows in the lower right region of RStudio, just above the plots. How do these two histograms compare?

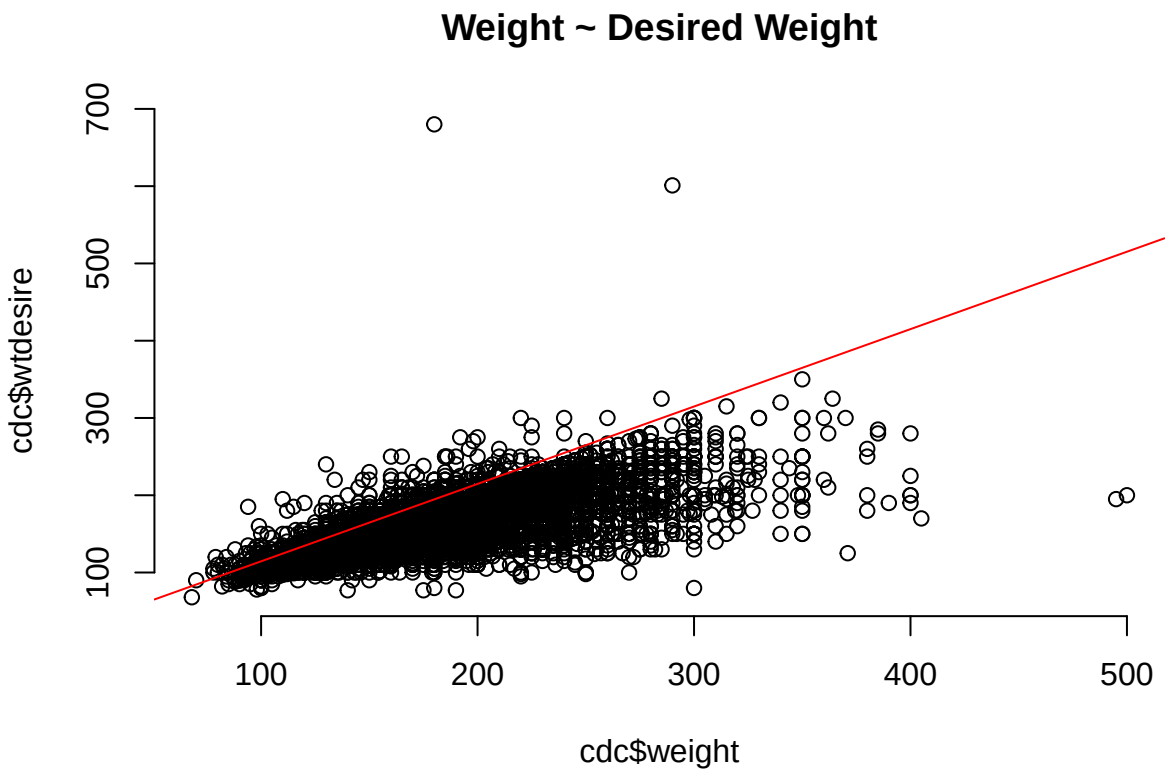
The y-axis scaling is very different between the two plots, which makes sense given that the breaks have increased from the first plot to the second plot. With only around 10 breaks in the first plot, there are more BMI values from the dataset lumped into these fewer breaks, dissimilar to the second plot with 50 breaks, which naturally lumped less BMI values into each break. Having more breaks in the second plot provides more information to the viewer along the x- and y-axis, allowing the viewer to get more specific frequency counts for each specific BMI value.

At this point, we've done a good first pass at analyzing the information in the BRFSS questionnaire. We've found an interesting association between smoking and gender, and we can say something about the relationship between people's assessment of their general health and their own BMI. We've also picked up essential computing tools – summary statistics, subsetting, and plots – that will serve us well throughout this course.

On Your Own

- Make a scatterplot of weight versus desired weight. Describe the relationship between these two variables.

```
plot(cdc$weight, cdc$wtdesire, main = 'Weight ~ Desired Weight', frame = FALSE)
abline(lm(cdc$weight ~ cdc$wtdesire, data = cdc), col = 'red')
```



After plotting current weight versus desired weight, we see a positive correlation between a repondents current weight and their desired weight. In other words, as respondents current weight increases, their desired weight also increases. Also, with the addition of a regression line, we can see that many plots fall below the regression line relative to the number of plots that fall above the regression line. This insinuates that respondents typically report desired weights that are less than their current weight. Overall, individuals in this sample seem to want to weigh less than their current weight!

- Let's consider a new variable: the difference between desired weight (`wtdesire`) and current weight (`weight`). Create this new variable by subtracting the two columns in the data frame and assigning them to a new object called `wdiff`.

```
wdiff <- cdc$wtdesire - cdc$weight
```

- What type of data is `wdiff`? If an observation `wdiff` is 0, what does this mean about the person's weight and desired weight. What if `wdiff` is positive or negative?

```
typeof(wdiff)
```

```
## [1] "integer"
```

wdiff is an object that holds integers that are the difference between each respondent's current weight and their desired weight. If an observation wdiff is 0, then a person's current weight is the same as their desired weight. They do not have a desire to be a different weight than

their current weight. If wdiff is positive then a person has a desired weight that is heavier than their current weight (they would like to gain weight). If wdiff is negative, then a person has a desired weight that is lighter than their current weight (they would like to lose weight).

- Describe the distribution of `wdiff` in terms of its center, shape, and spread, including any plots you use. What does this tell us about how people feel about their current weight?

  After looking at some measures of center, as well as plotting the wdiff values in a frequency table, I've found that the median desired weight of people in the sample is 10 pounds less than their current weight. Additionally, the average desired weight is close to 15 pounds less than their current weight. Because there is roughly equal trailing off in both directions when plotting this as a histogram (Histogram #3, below), we can say the shape is symmetric. Additionally, after adjusting the boxplot below to compensate for outliers (Boxplot #1, below), you can see that half of the data (people's responses) fall within an interquartile range of a desire to lose anywhere between zero and 21 pounds from their current weight. The standard deviation also reflects this, at 24.04. Finally, I did a few calculations to see what percent of the wdiff values were considered outliers given our mean desired weight to be roughly 15 pounds less than people's current weight - I found that roughly 6% of responses for desired weight were outliers. Given the dataset has 20,000 observations, this did create some challenges when plotting values. I tried to compensate for this by taking the log of wdiff values as well as putting limits on both axes, when needed.

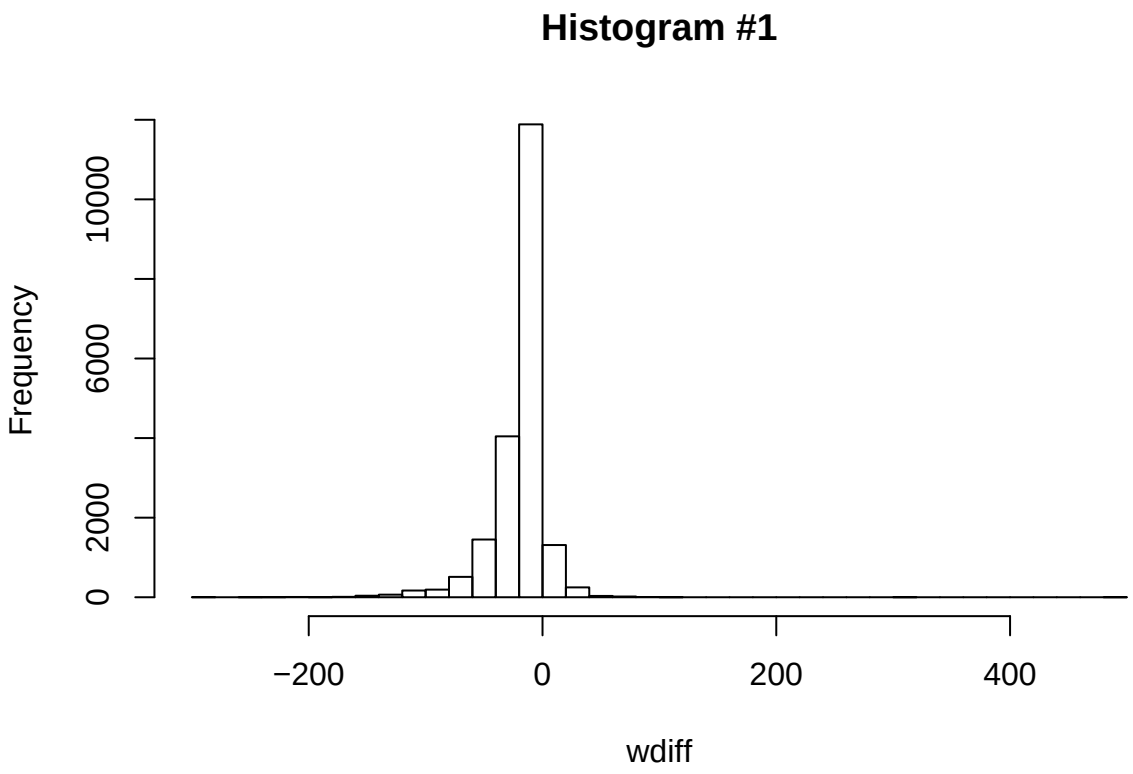
```
summary(wdiff)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -300.00  -21.00  -10.00  -14.59    0.00  500.00
```

```
library(psych)
describe(wdiff)
```

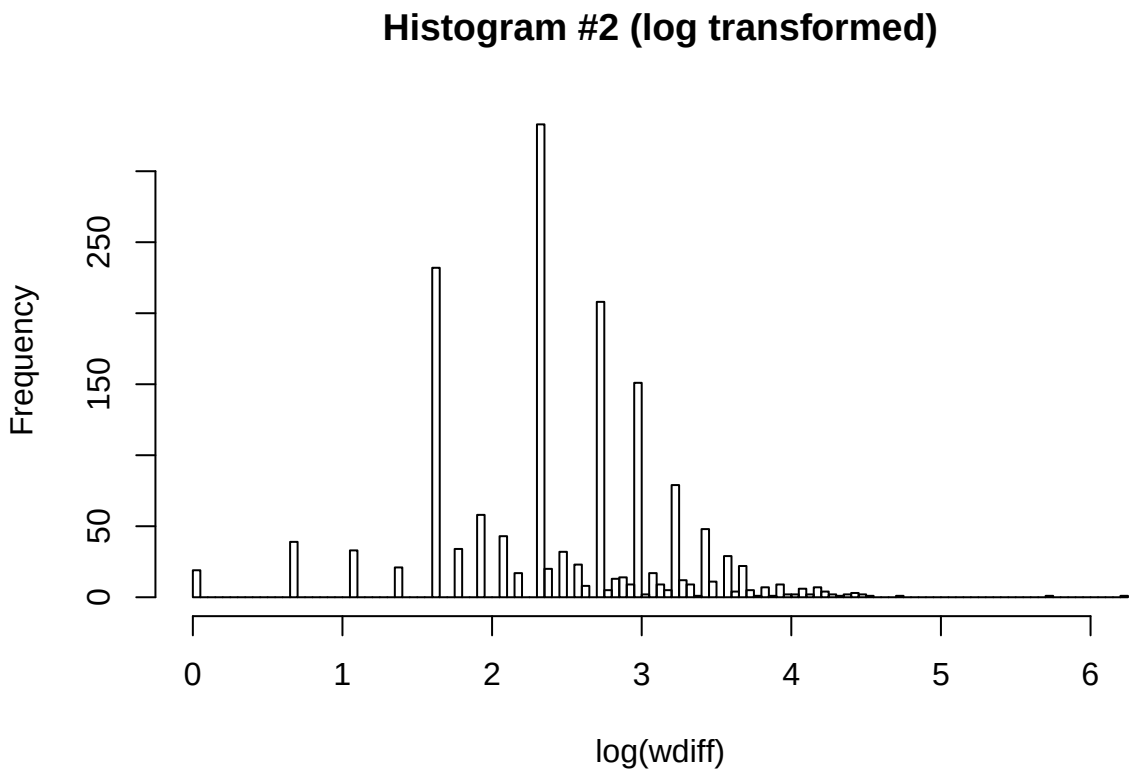
```
##    vars     n   mean    sd median trimmed   mad  min max range  skew
## X1    1 20000 -14.59 24.05    -10  -11.41 14.83 -300 500   800 -1.45
##    kurtosis   se
## X1    21.61 0.17
```

```
hist(wdiff, breaks = 50, main = 'Histogram #1')
```

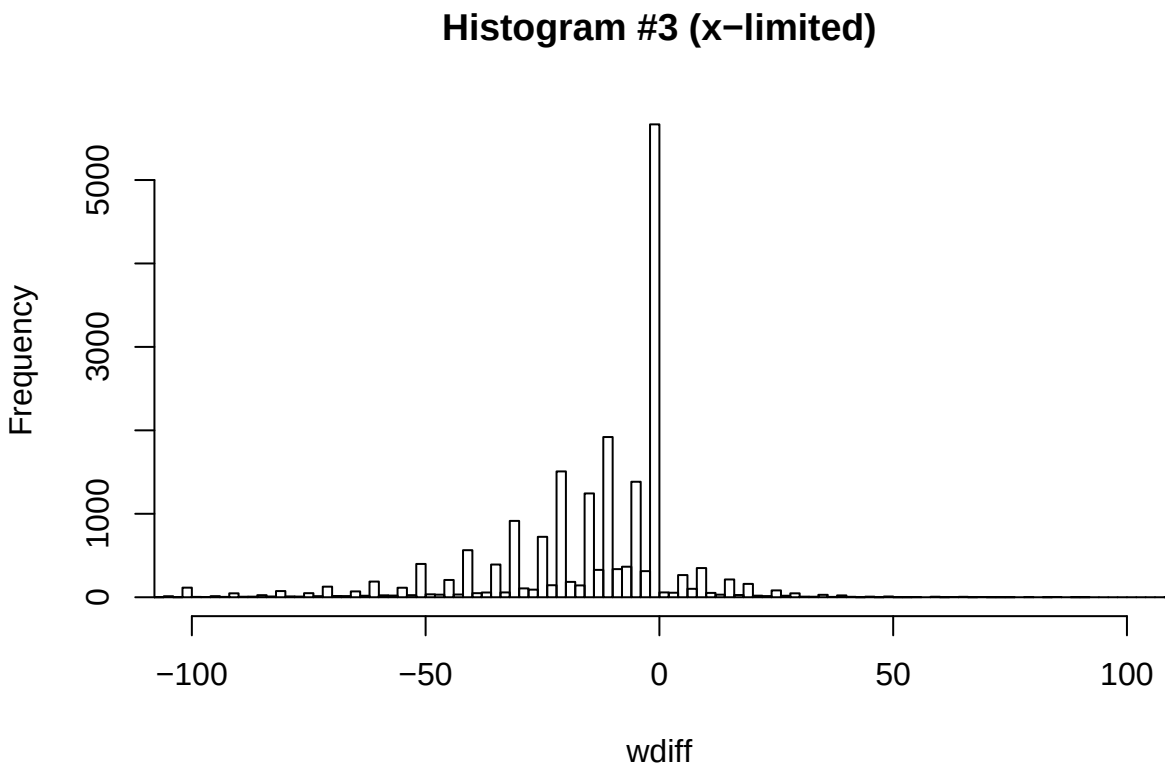



```
hist(log(wdiff), breaks = 100, main = 'Histogram #2 (log transformed)')
```

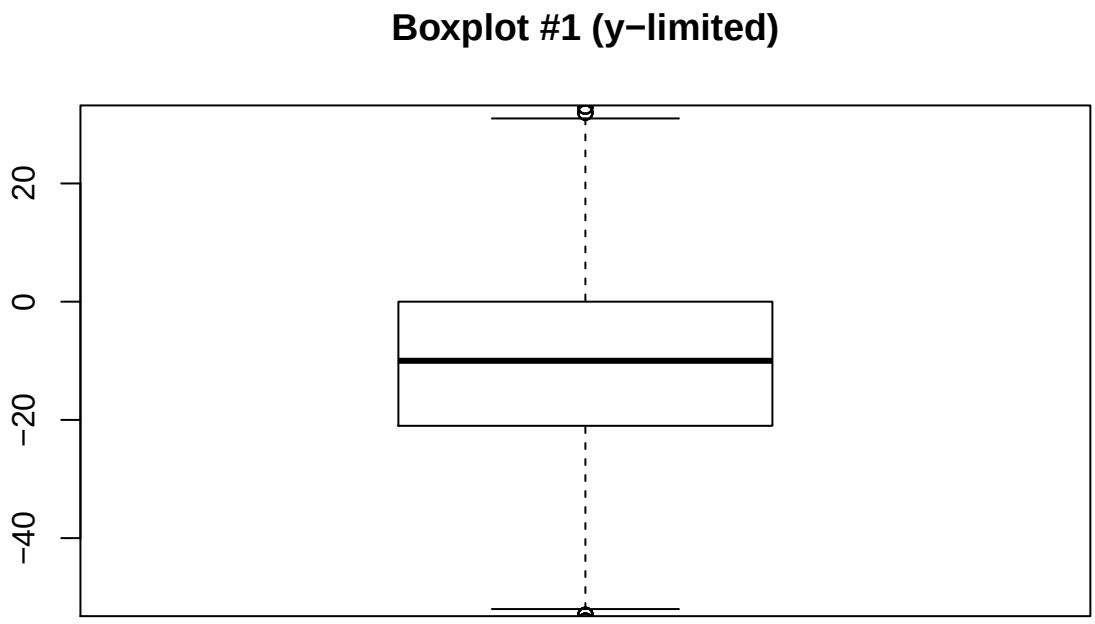
```
## Warning in log(wdiff): NaNs produced
```



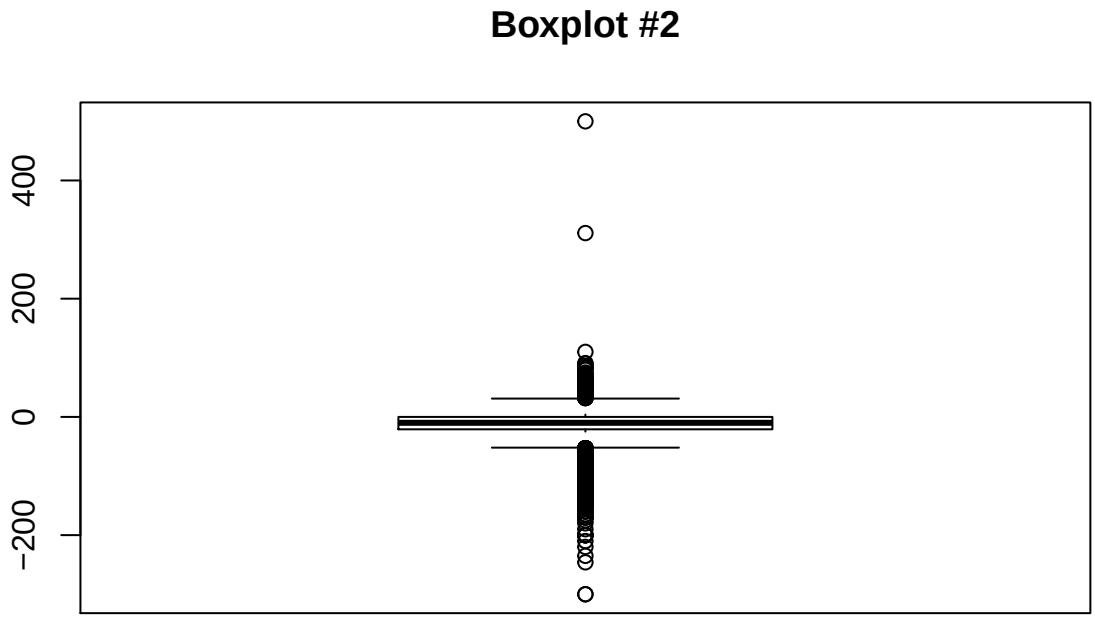

```
hist(wdiff, breaks = 300, xlim = c(-100, 100), main = 'Histogram #3 (x-limited)')
```




```
# to view the IQR better
boxplot(wdiff, ylim = c(-50, 30), main = 'Boxplot #1 (y-limited)')
```




```
# to show outliers
boxplot(wdiff, main = 'Boxplot #2')
```




```
# calculating percent of outliers relative to full dataset

# calculating the "minimum" of the boxplot (Q1 - 1.5*IQR)
minimum <- (-21) - (1.5 * 21)
minimum
```

```
## [1] -52.5
```

```
# calculating the "maximum" of the boxplot (Q3 + 1.5*IQR)
maximum <- (0) + (1.5 * 21)
maximum
```

```
## [1] 31.5
```

```
wdiff_outliers <- length(wdiff[wdiff > 31.5 | wdiff < -52.5])
wdiff_outliers / length(wdiff)
```

```
## [1] 0.0647
```

- Using numerical summaries and a side-by-side box plot, determine if men tend to view their weight differently than women.

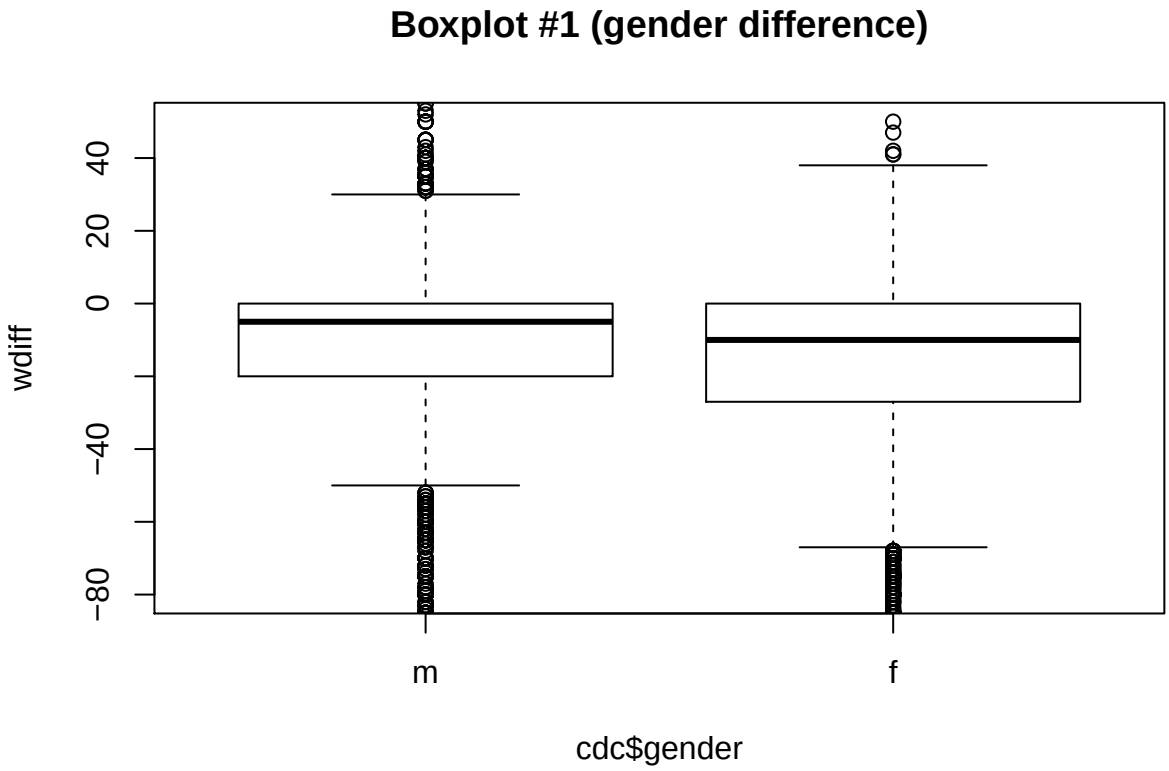
  From the numerical summaries below, splitting wdiff by gender, you can see that the median desired weight for males is 5 pounds less than their current weight. For females,

the median desired weight is ten pounds less than their current weight. This is also supported by the means, as well as visually in the side-by-side box plot below ('Boxplot #1 (gender difference)') that males tend to have a desire to lose less weight than females in this sample.

```
tapply(wdiff, cdc$gender, summary)
```

```
## $m
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -300.00  -20.00   -5.00  -10.71    0.00  500.00
##
## $f
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -300.00  -27.00  -10.00  -18.15    0.00   83.00
```

```
boxplot(wdiff ~ cdc$gender, ylim = c(-80, 50), main = 'Boxplot #1 (gender difference)')
```



- Now it's time to get creative. Find the mean and standard deviation of `weight` and determine what proportion of the weights are within one standard deviation of the mean.

First, finding the mean and standard deviation. From the describe function below, we can see that the mean weight is 169.68 and the standard deviation is 40.08.

```r
describe(cdc$weight)
```

```
##    vars     n   mean    sd median trimmed   mad min max range skew
## X1    1 20000 169.68 40.08    165  166.58 37.06  68 500   432 0.96
##    kurtosis   se
## X1     1.99 0.28
```

Then, calculating the low bound of one standard deviation from the mean, as well as calculating the high bound of one standard deviation from the mean.

```r
mean_weight <- 169.68
weight_low_sd1 <- 169.68 - 40.08
weight_high_sd1 <- 169.68 + 40.08

weight_low_sd1
```

```
## [1] 129.6
```

The low bound of one standard deviation from the mean weight is 129.6. The high bound of one standard deviation from the mean weight is 209.76.

```r
weight_high_sd1
```

```
## [1] 209.76
```

Now, we need to subset the data to only include weights that fall within this range of values (209.76 to 129.60). I subsetted the data below:

```r
weights_one_sd <- subset(cdc, weight >= 129.60 & weight < 209.76)
weights_within_one_sd <- dim(weights_one_sd)[1]
```

It looks like, after subsetting the data to only include weights within the bounds of 129.6 and 209.76, that there are 14,152 values that match this criteria. Finally, we can take the proportion:

```r
prop_weight_one_sd <- 14152 / 20000
prop_weight_one_sd
```

```
## [1] 0.7076
```

We can conclude that 70.76% of the weights in the sample fall within one standard deviation from the mean.