Introduction to data

Joshua Registe

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.

```r
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

```r
names(cdc)
```

```
## [1] "genhlth"  "exerany"  "hlthplan" "smoke100" "height"   "weight"   "wtdesire"
## [8] "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).

```r
str(cdc)
```

```
## 'data.frame':    20000 obs. of  9 variables:
##  $ genhlth : Factor w/ 5 levels "excellent","very good",..: 3 3 3 3 2 2 2 2 3 3 ...
##  $ exerany : num  0 0 1 1 0 1 1 0 0 1 ...
##  $ hlthplan: num  1 1 1 1 1 1 1 1 1 1 ...
##  $ smoke100: num  0 1 1 0 0 0 0 0 1 0 ...
##  $ height  : num  70 64 60 66 61 64 71 67 65 70 ...
##  $ weight  : int  175 125 105 132 150 114 194 170 150 180 ...
##  $ wtdesire: int  175 115 105 124 130 114 185 160 130 170 ...
##  $ age     : int  77 33 49 42 55 55 31 45 27 44 ...
##  $ gender  : Factor w/ 2 levels "m","f": 1 2 2 2 2 2 1 1 2 1 ...
```

This data set has 20,000 cases and 9 variables as shown in the structure above. The datatype for each variable is the following: genhealth = nominal categorical (could be thought of as ordered from least to best health) exerany = nominal categorical hlthplan = nominal categorical smoke100 = nominal categorical height = numerical discreet weight = numerical discreet wtdesire = numerical discreet age = numerical discreet gender = nominal categorical

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

```r
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

```r
tail(cdc)
```

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

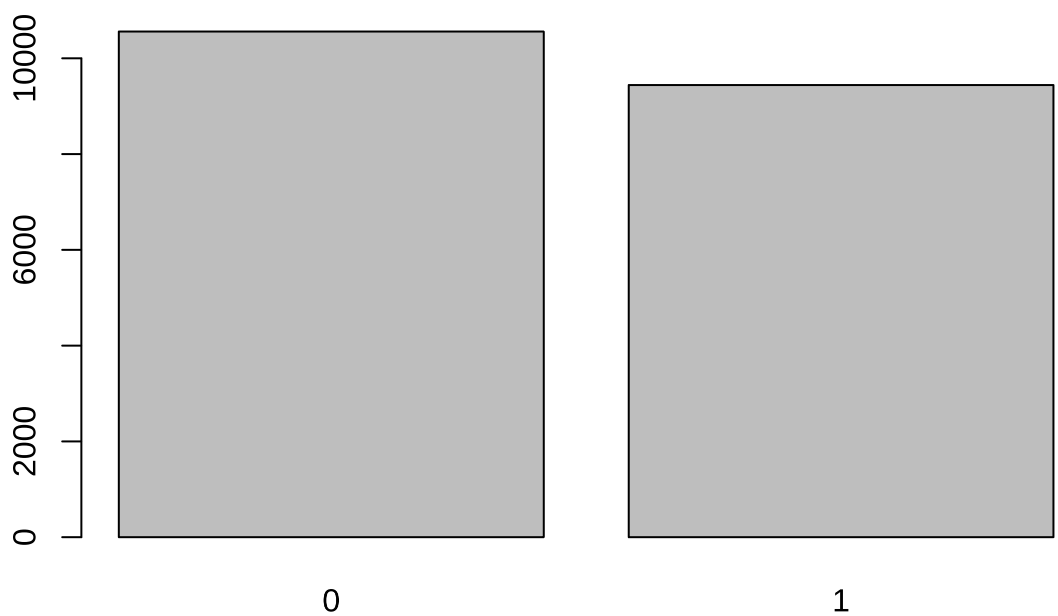
or instead look at the relative frequency distribution by typing

```
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.

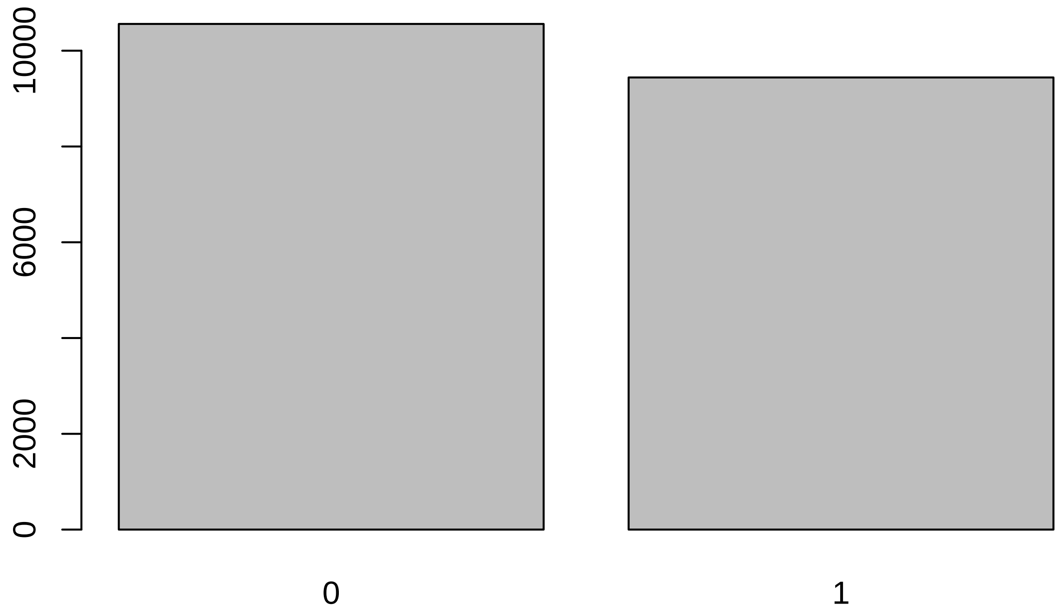
```
barplot(table(cdc$smoke100))
```



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?

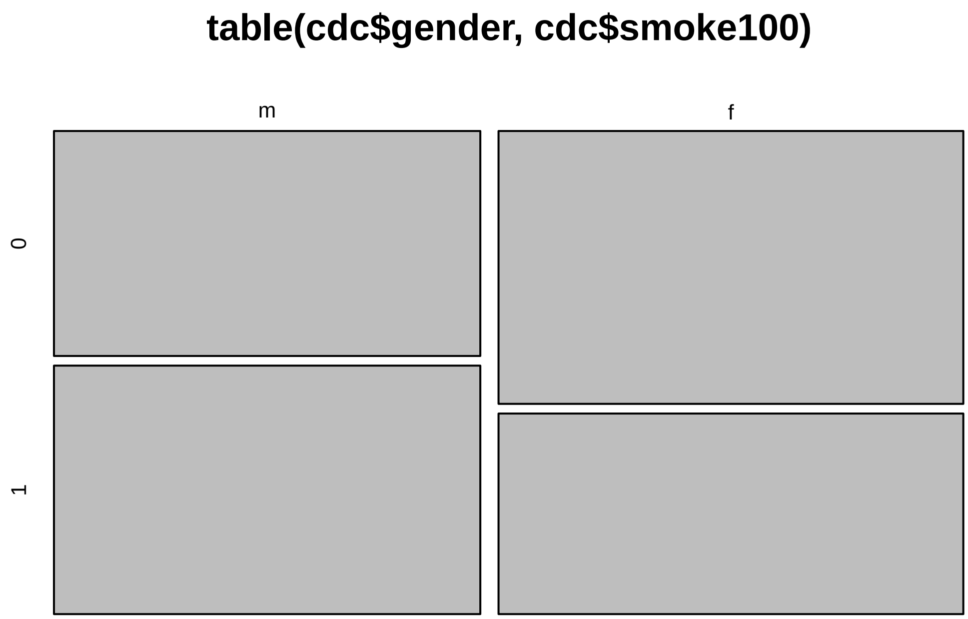
The `table` command can be used to tabulate any number of variables that you provide. For example, to examine which participants have smoked across each gender, we could use the following.

```
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?

the mosaic plot reveals that for this subset of data, a larger proportion of males smoked 100 cigarettes or more in their lifetime.

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"   "wtdesire"
## [8] "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

```r
head(cdc[,6])
```

```
## [1] 175 125 105 132 150 114
```

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.

```r
head(cdc$weight)
```

```
## [1] 175 125 105 132 150 114
```

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

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

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

Similarly, for just the first 10 respondents

```r
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

```r
head(cdc$gender == "m")
```

```
## [1]  TRUE FALSE FALSE FALSE FALSE FALSE
```

or

```r
head(cdc$age > 30)
```

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

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

```r
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

```r
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

```r
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

```r
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.

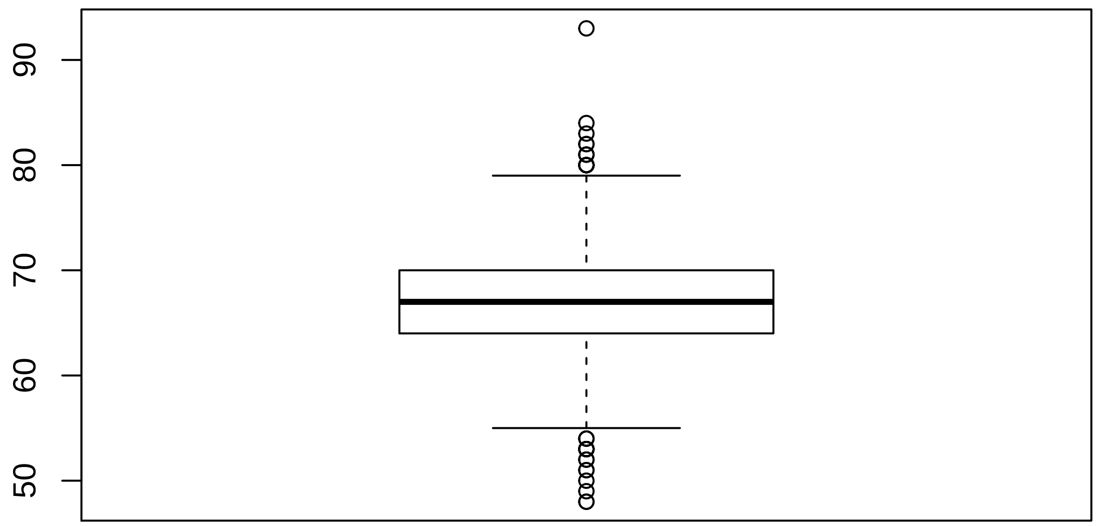
3. Create a new object called `under23_and_smoke` that contains all observations of respondents under the age of 23 that have smoked 100 cigarettes in their lifetime. Write the command you used to create the new object as the answer to this exercise.

   ```r
   under30_smoker<-subset(cdc, smoke100 ==1 & age<23)
   ```

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.

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



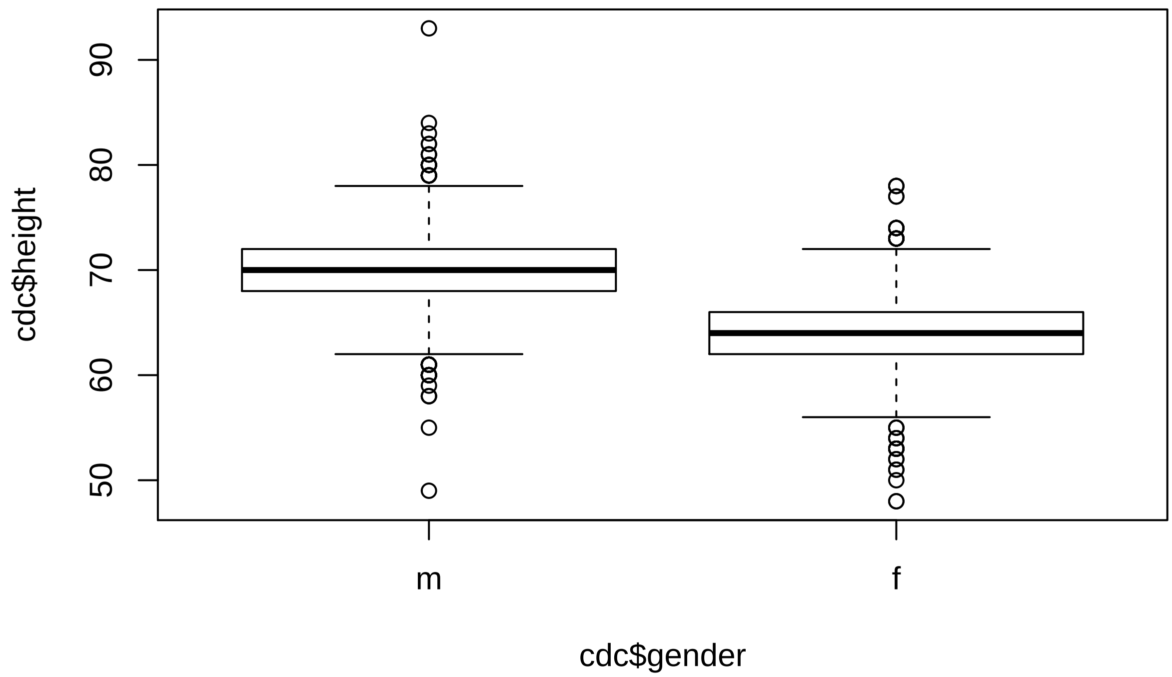
You can compare the locations of the components of the box by examining the summary statistics.

```
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.

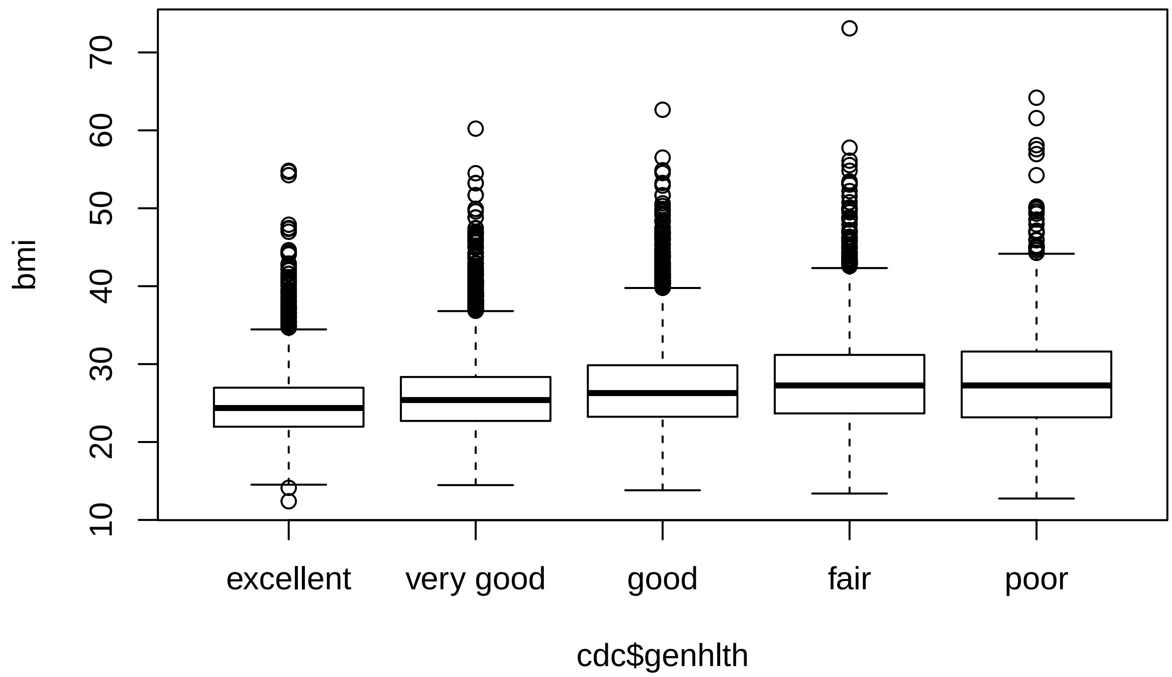
Next let's consider a new variable that doesn't show up directly in this data set: Body Mass Index (BMI) (http://en.wikipedia.org/wiki/Body_mass_index). BMI is a weight to height ratio and can be calculated as:

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

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)
```

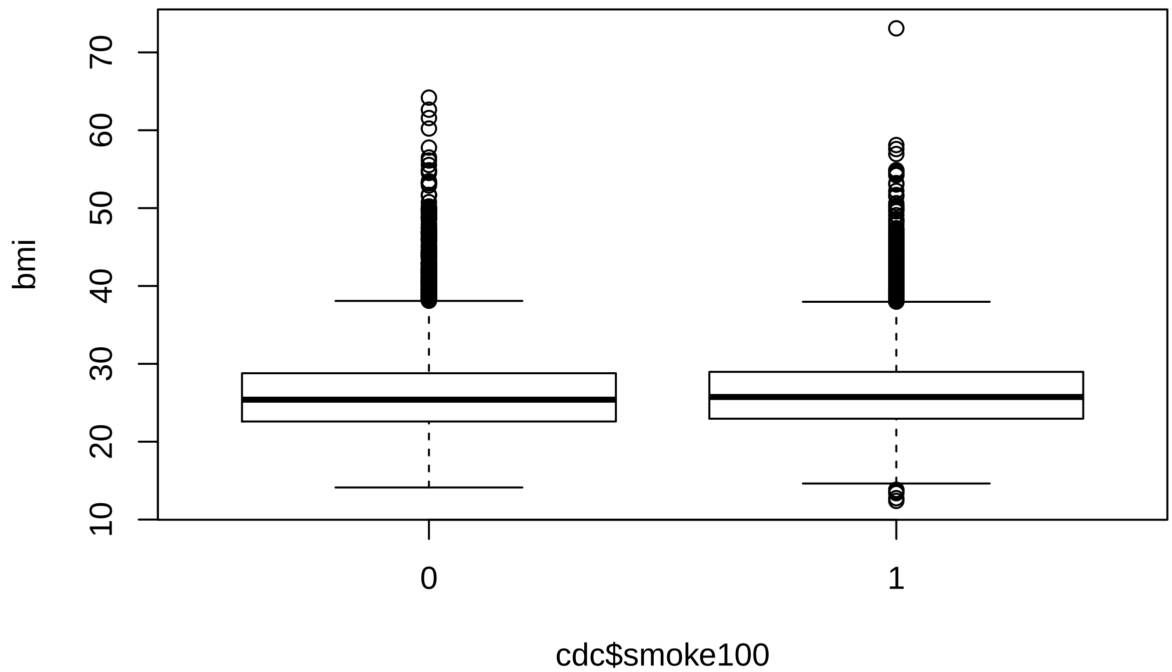


Notice that the first line above is just some arithmetic, but it's applied to all 20,000 numbers in the `cdc` data set. That is, for each of the 20,000 participants, we take their weight, divide by their height-squared and then multiply by 703. The result is 20,000 BMI values, one for each respondent. 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.

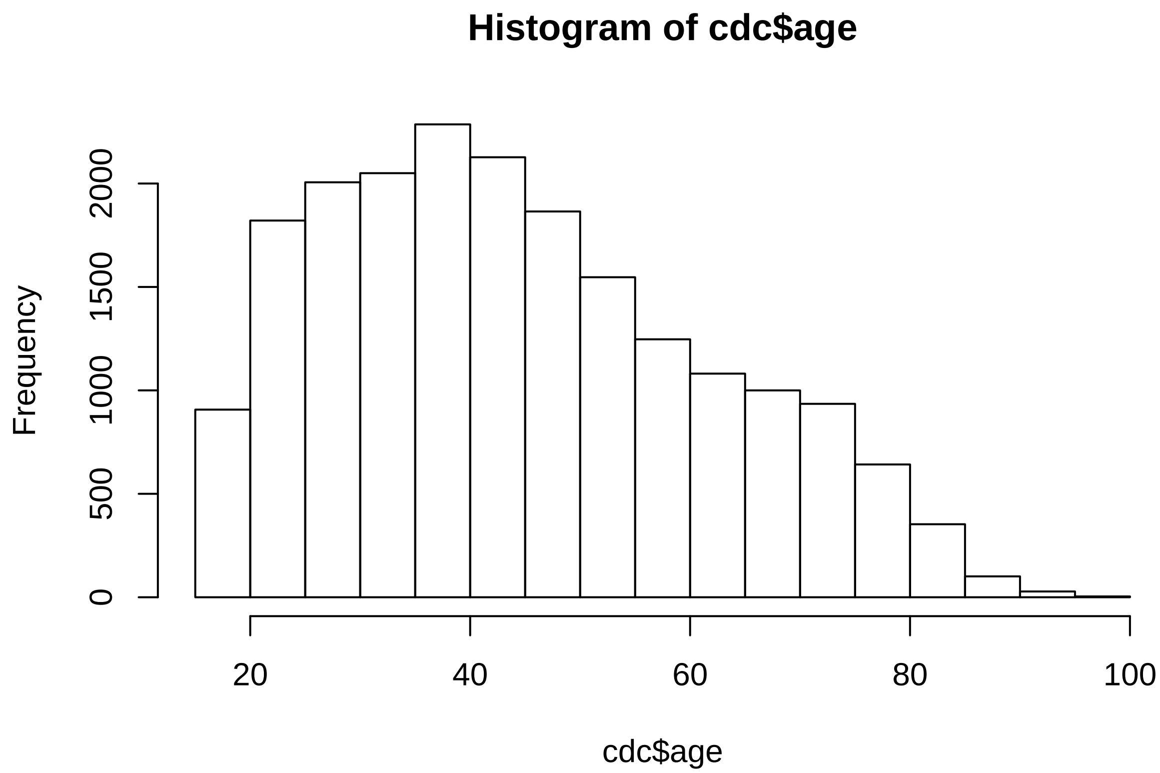
This box plot shows higher BMIs tended to suggest poorer health among people in this survey. Another thing to potentially compare BMI to is smokers. Some studies suggest smoking can lead to weight gain/loss, however based on the following plot does not show a relationship to BMI.

```
bmi <- (cdc$weight / cdc$height^2) * 703
boxplot(bmi ~ cdc$smoke100)
```



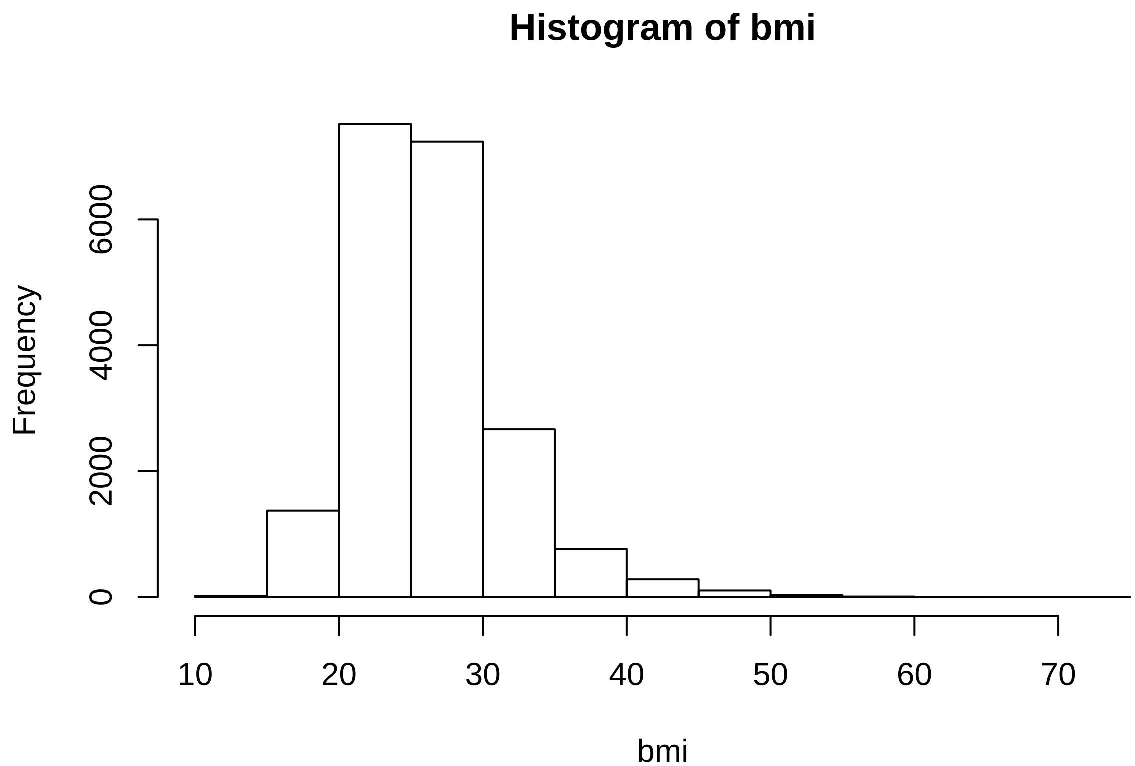
Finally, let's make some histograms. We can look at the histogram for the age of our respondents with the command

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

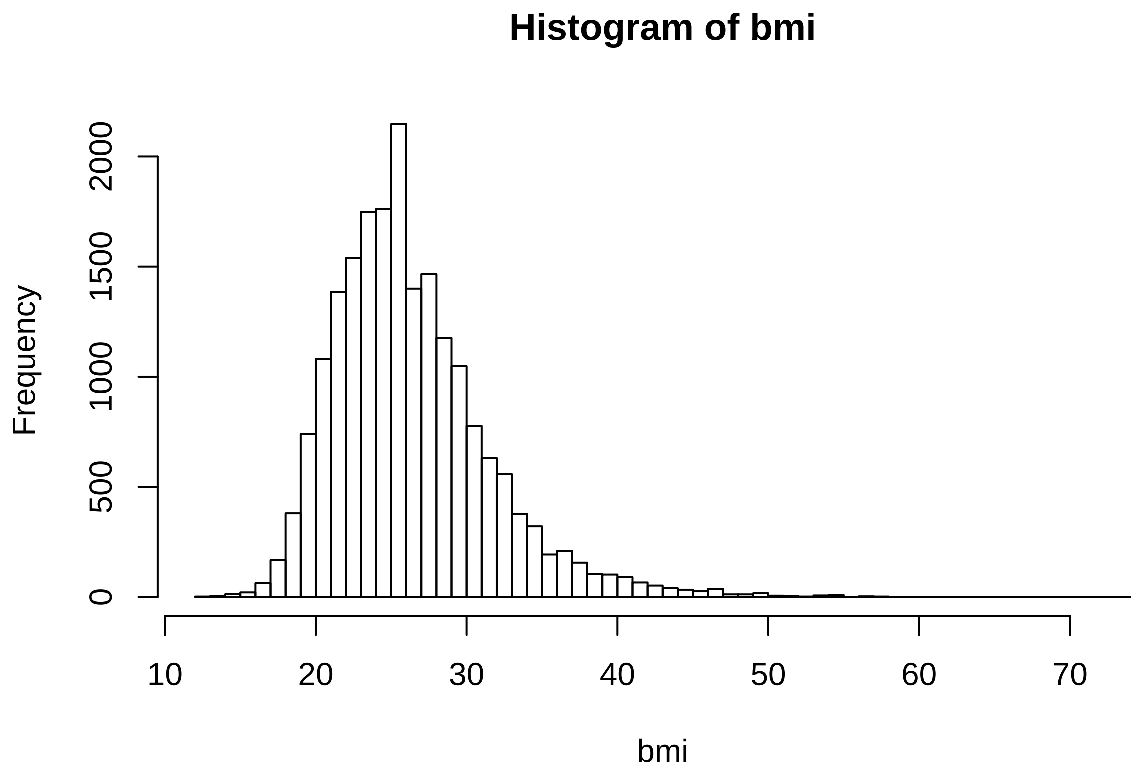


Histograms are generally a very good way to see the shape of a single distribution, but that shape can change depending on how the data is split between the different bins. You can control the number of bins by adding an argument to the command. In the next two lines, we first make a default histogram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

```
hist(bmi)
```




```r
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?

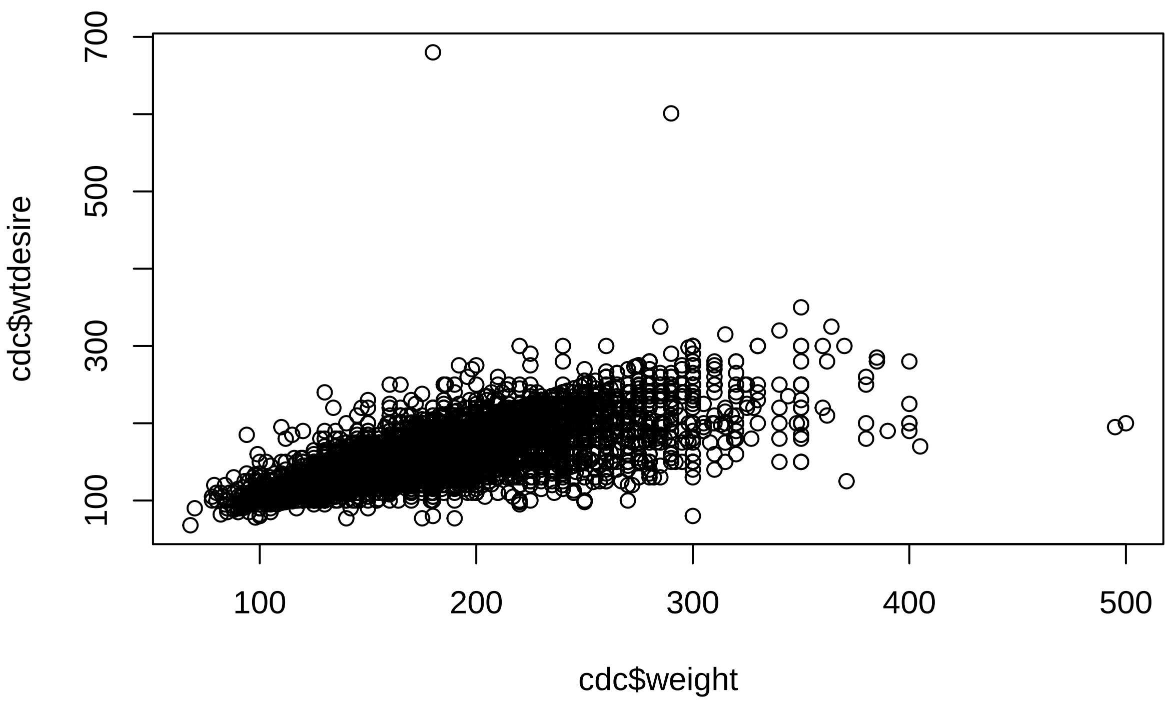
At this point, we've done a good first pass at analyzing the information in the BRFSS questionnaire. We've found an interesting association between smoking and gender, and we can say something about the relationship between people's assessment of their general health and their own BMI. We've also picked 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.

As shown in the following figure, there seems to be a positive relationship between the weight desired vs the actual weight. this is likely because people tend to desire weights near their existing weights especially for short term goals.

```r
plot(cdc$weight, cdc$wtdesire)
```



- 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?

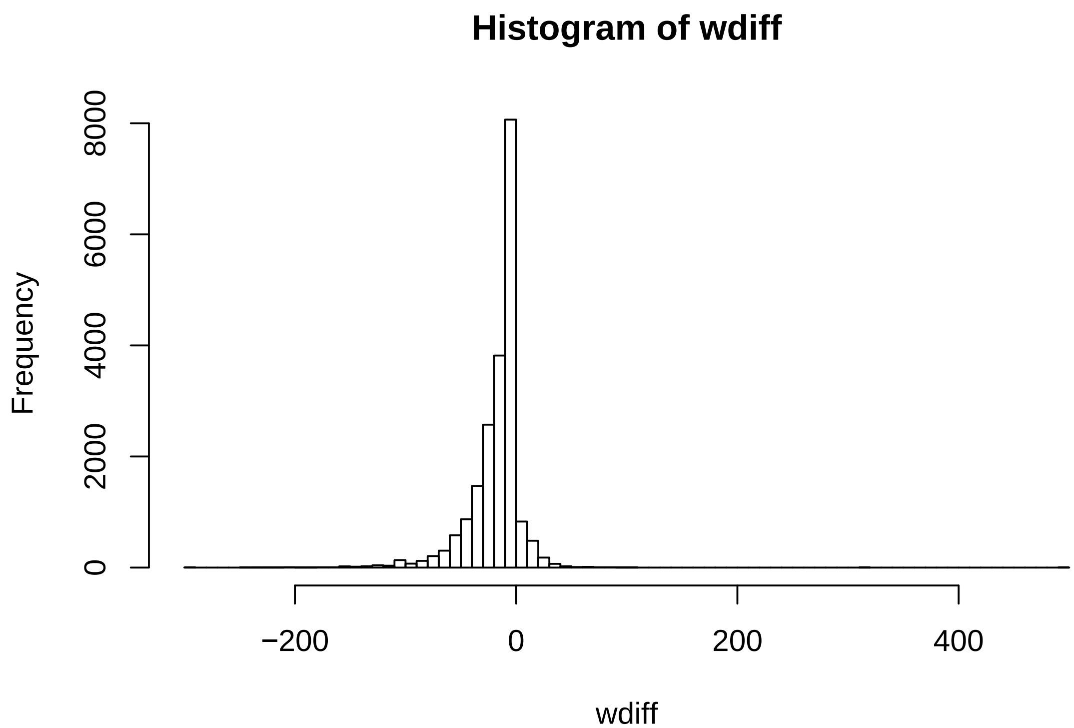
  wdiff is a continuous variable, if an observation wdiff is 0, that means that the person is already at their desired weight. If wdiff is positive, that means the person aspires to gain weight, and if it is negative, the person aspires to lose weight.

- 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?

```
summary(wdiff)
```

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

```
hist(wdiff, breaks = 100)
```



wdiff takes on a gaussian distribution and most people seem to want to lose weight. the median desired weight loss is 10 lbs and the mean is 14.59 lbs however the mean is likely skewed to outlier responses.

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

```
cdc$wdiff<-cdc$wtdesire-cdc$weight
#males
summary(subset(cdc,gender=="m"))[,10]
```

```
##
## "Min.   :-300.00  " "1st Qu.: -20.00  " "Median :  -5.00  " "Mean   : -10.71  "
##
## "3rd Qu.:   0.00  " "Max.   : 500.00  "
```

```
#females
summary(subset(cdc,gender=="f"))[,10]
```

```
##
## "Min.   :-300.00  " "1st Qu.: -27.00  " "Median : -10.00  " "Mean   : -18.15  "
##
## "3rd Qu.:   0.00  " "Max.   :  83.00  "
```

According to these summary statistics, men have a median desired weight loss of 5 lbs while woman have a median desired weight loss of -10. The mean desired weight loss for woman is also greater than for men.

- 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.

```
zscoreweight<-scale(cdc$weight)[,1]


length(zscoreweight[abs(zscoreweight<=1)])/length(zscoreweight)
```

```
## [1] 0.84675
```

Approximately 84.675% are within 1 standard deviation from the mean. This follows Gaussian distribution properties.