Lab 2 - Introduction to Data and Descriptive Statistics in R

NAME: Your Name

NETID: Your NetID

Lab Goals

1. To introduce data structures and data manipulation in R
2. To introduce code for descriptive statistics in R using the *cdc* data
3. To practice analyzing graphical data summaries

Data Structures in R

R has a lot of different data types. Here we will focus on vectors and data frames.

1. Vectors

One way to create vectors in R is by using the function `c()` which combines its arguments into a vector. Vectors in R do not need to be numeric. Run the following code chunk by setting 'eval=TRUE' and knitting your document to put these vectors into your workspace for this lab document.

```
a <- c(1, 2, 0, -5)  # numeric vector
b <- c("one", "two", "three")  # character vector
cc <- c(TRUE, FALSE, TRUE, FALSE, FALSE)  # logical vector
```

Note in the R code above, R ignores anything written after the # sign. Throughout this lab, you will occasionally see comments after lines of code written in this way. Also, `c` could have been used as the name of the logical vector, but since `c` is already a function it is best not to use it as a variable name too.

To find the size of the vector use the `length()` function, e.g.

```
length(cc)
```

```
## [1] 5
```

2. Data Frames

A data frame is the most common format for data in R. Let's form a small data frame from three vectors:

```
d <- c(1, 2, 3, 4) # can you recall a different way to do this from last lab?
e <- c("Ann", "Bob", "Will", NA)
f <- c(TRUE, FALSE, TRUE, FALSE)
mydata <- data.frame(d, e, f)
mydata
```

```
##   d    e     f
## 1 1  Ann  TRUE
## 2 2  Bob FALSE
```

```
## 3 3 Will  TRUE
## 4 4 <NA> FALSE
```

Note in the vector `e` above, `NA` indicates the 4th element of this vector is empty (or missing data).

```
colnames(mydata) <- c("ID", "Name", "Passed") # variable names
mydata
```

```
##   ID Name Passed
## 1  1  Ann   TRUE
## 2  2  Bob  FALSE
## 3  3 Will   TRUE
## 4  4 <NA>  FALSE
```

A column of a data frame can be referenced by using its name:

```
mydata$Passed
```

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

You can add additional rows or columns to a data frame (or matrix or vector) by using the `rbind()` and `cbind()` functions in R

```
Score <- c(24, 10, 11, 50)
newdata <- cbind(mydata, Score)
newdata
```

```
##   ID Name Passed Score
## 1  1  Ann   TRUE    24
## 2  2  Bob  FALSE    10
## 3  3 Will   TRUE    11
## 4  4 <NA>  FALSE    50
```

Descriptive Statistics in R with the cdc dataset

Let's work with some real data. The Behavioral Risk Factor Surveillance System (BRFSS) is an annual telephone survey of 350,000 people in the US for identifying risk factors in the adult population and reporting emerging health trends. For example, respondents are asked about their diet and weekly physical activity, their HIV/AIDS status, possible tobacco use, and 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 for this R Markdown document and also the workspace for the R console. Set `eval=TRUE` below to put these data into the workspace for the R Markdown document. To download these data into your workspace for the R Console, copy the code below and select "Run" from the menu above.

```
source("http://www.openintro.org/stat/data/cdc.R")
```

Note: When this data is read into R, by default it is given the name `cdc`.

Problem 1

Using the instructions in "Data Structures in R", above, put a code chunk here that gives the column names of the data frame `cdc`.

Problem 2

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 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
```

a) How many observations are there in this data set?

b) How many variables?

There are multiple ways that you can determine the number of observations and the number of variables.

```
nrow(cdc)
```

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

```
ncol(cdc)
```

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

```
dim(cdc)
```

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

c) For each of the following variables, identify its data type (categorical or quantitative).

genhlth:

exerany (1 if you exercise, 0 otherwise):

height:

age:

gender:

Problem 3

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, third 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
```

Add a new code chunk here to calculate the summary statistics for height.

Problem 4

For categorical data a frequency or relative frequency distribution provides a good summary of the sample data. 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

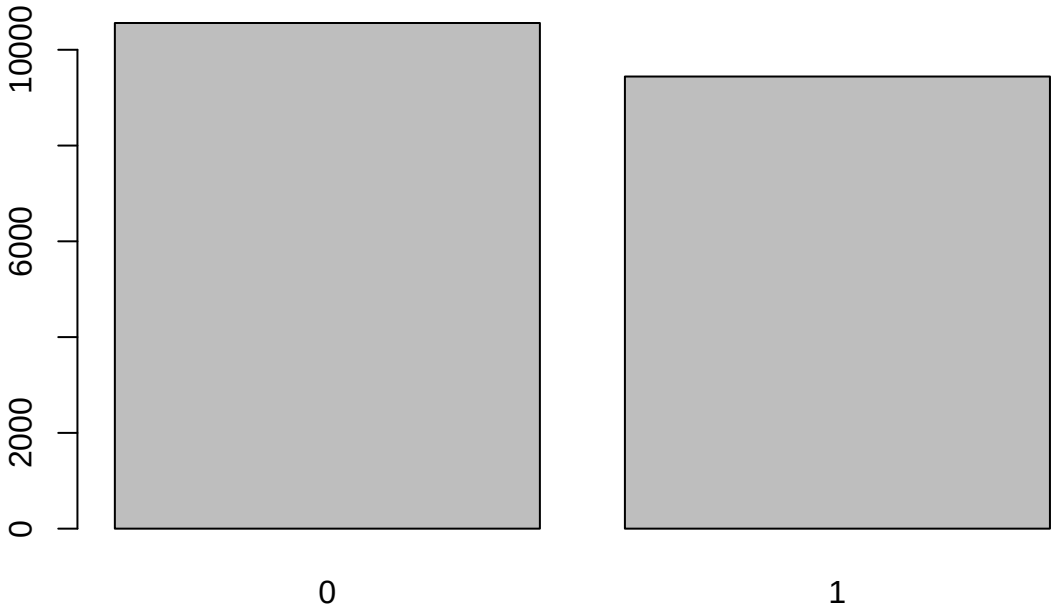
```
table(cdc$smoke100) / length(cdc$smoke100)
```

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

Notice how R automatically divides all entries in the table by `length(smoke100)` in the command above.

Next, we make a bar plot of the entries in the table by putting the table inside the barplot command.

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



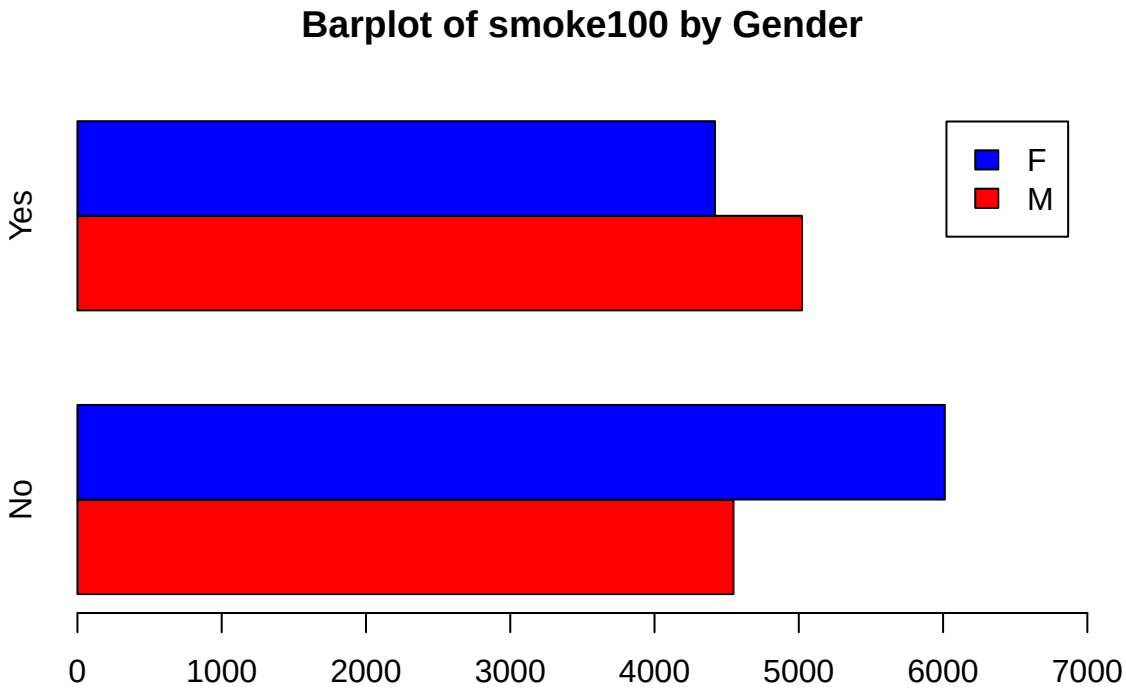
Plots created in R can be simple or very detailed. You can find all the options for the `barplot()` function by typing `?barplot` in the R console. Here we will create a new barplot where the counts are separated by gender. First, create a new table (named `smokebygen`) of `smoke100` by `gender` as follows.

```
smokebygen <- table(cdc$gender, cdc$smoke100)
smokebygen
```

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

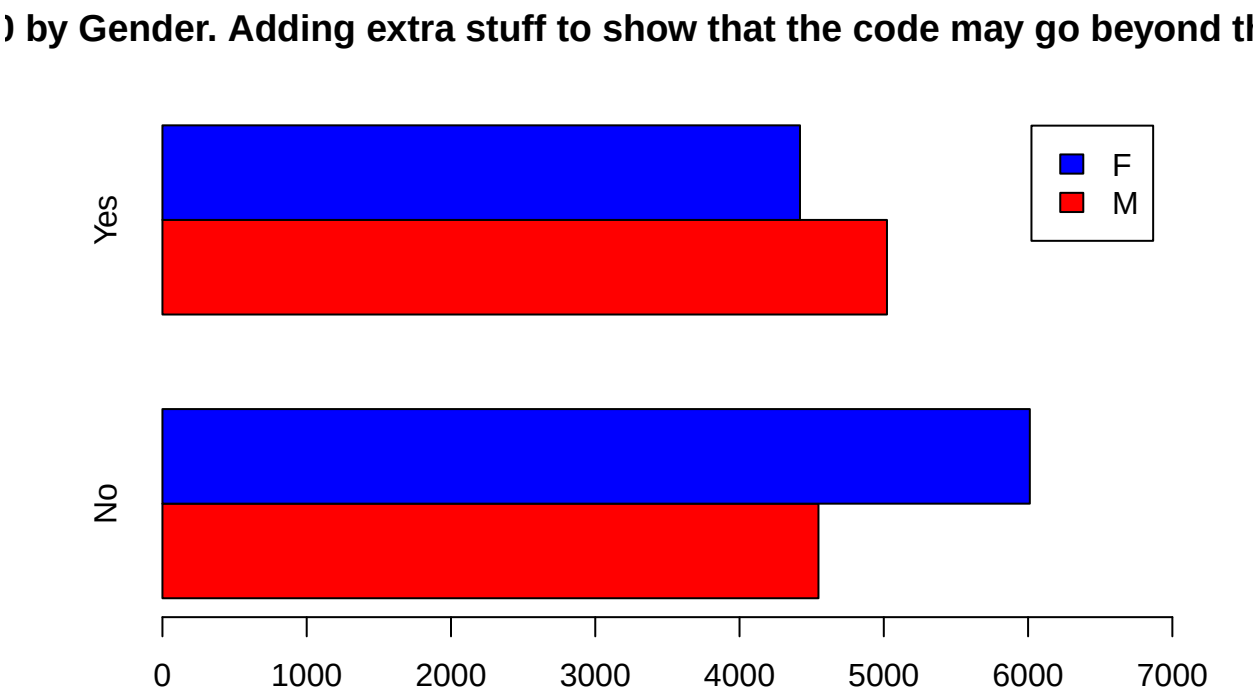
How many females in the study had not smoked 100 cigarettes in their lifetime?

```
barplot(smokebygen, xlim = c(0, 7000), names.arg = c('No', 'Yes'), legend.text = c('M', 'F'), beside = T
```



In the R console run the code `?barplot`. Reminder: This function will bring up the "Help" documentation for `barplot()` in the lower right panel.

```
barplot(smokebygen, xlim = c(0, 7000), names.arg = c('No', 'Yes'), legend.text = c('M', 'F'), beside = T
```



Pick three of the options used to create the barplot above and list them with a description of how each option changes the plot.

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. In fact, in the RStudio console, you can type `View(cdc)` to see the cdc data frame displayed as it would be in a spreadsheet.

Each row of the data frame 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 returns 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
```

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

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

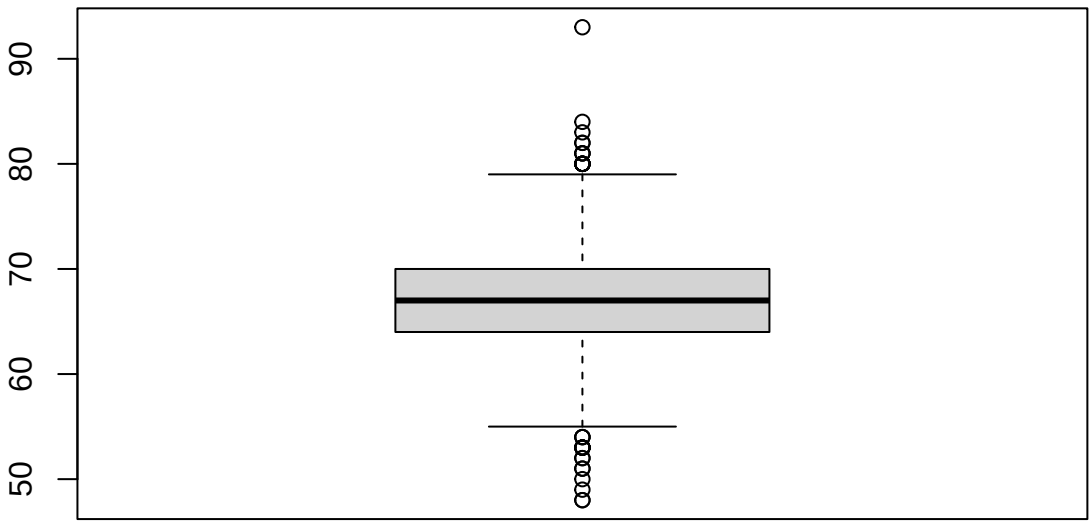
```
##       genhlth exerany hlthplan smoke100 height weight wtdesire age gender
## 1        good       0        1        0     70    175      175  77      m
## 2        good       0        1        1     64    125      115  33      f
## 3        good       1        1        1     60    105      105  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.

Quantitative data

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.

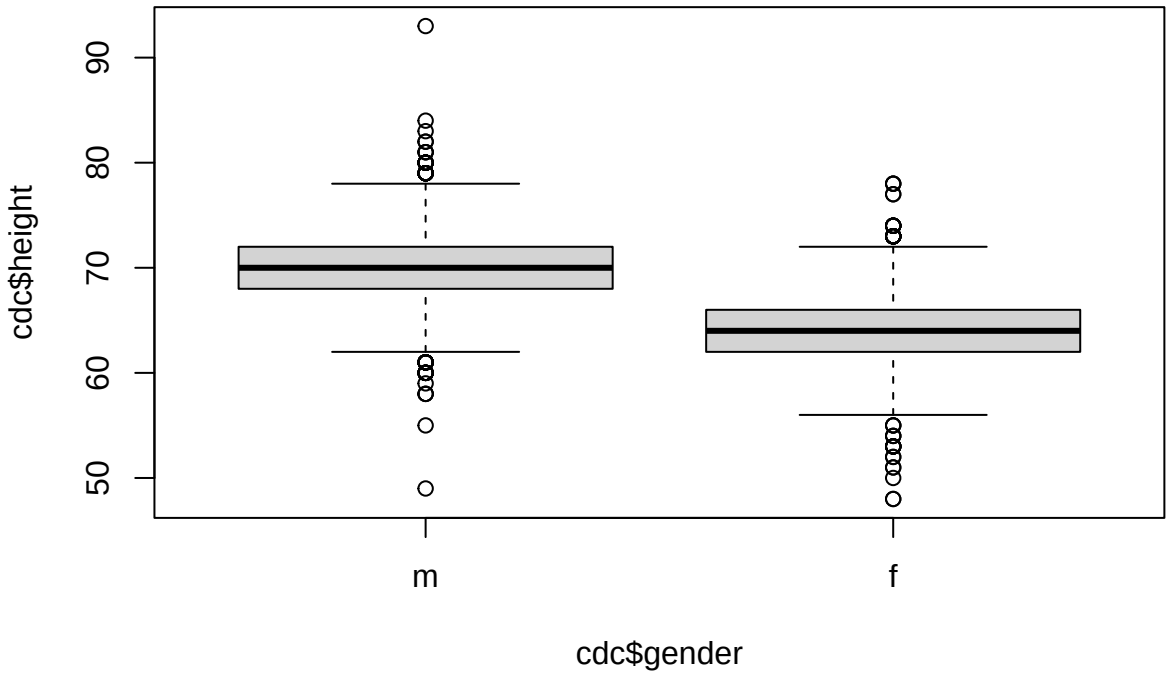
```
summary(cdc$height)
```

```
##    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 boxplots of heights where the groups are defined by gender.

R makes it easy to work with data in data frames. For example, suppose we are curious about the age distribution of females who weigh less than 140lbs. We can use the `subset` function to get a data frame that only includes rows corresponding to females weighing under 140lbs.

```
female_under140 <- subset(cdc, gender == "f" & weight < 140)
```

Notice that a double equals sign is used to get at the logical expression (i.e., true/false) and the & sign is used to indicate "and". In words, we ask for the gender variable to equal "f" *and* the weight variable to be less than 140.

To answer questions about this subgroup, we can simply use the data frame `female_under140` in place of `cdc`.

Problem 5

a) How many females in the data set weigh less than 140lbs?

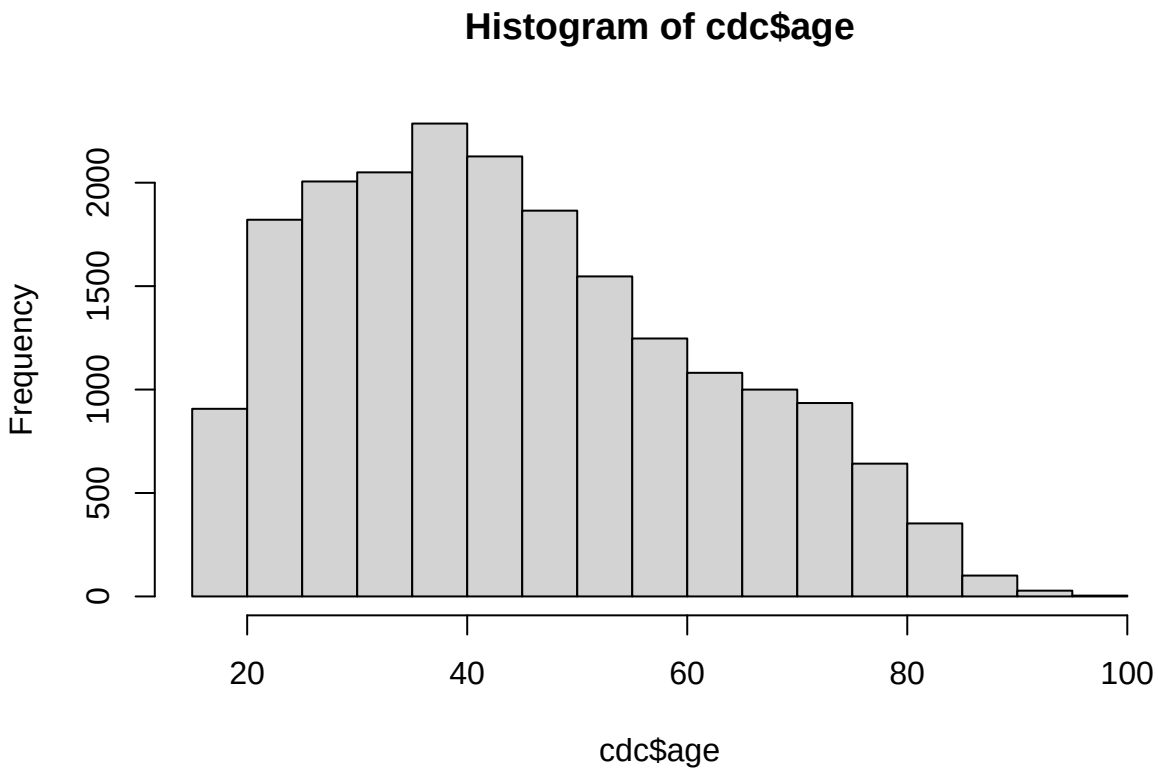
b) Show the summary statistics for age for this subgroup.

c) Compute the average height of men who exercise. Note: `Exerany` is 1 if an individual exercises at all and 0 otherwise.

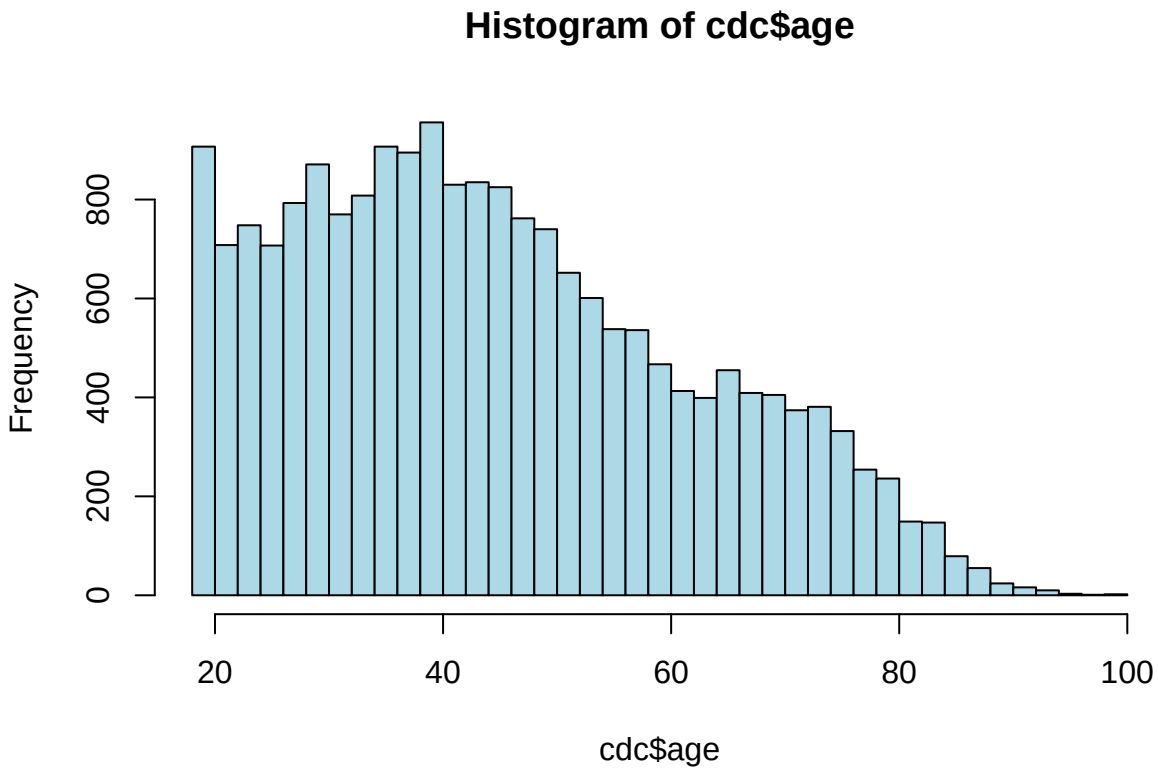
Problem 6

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 `age` and then one with 50 breaks.

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



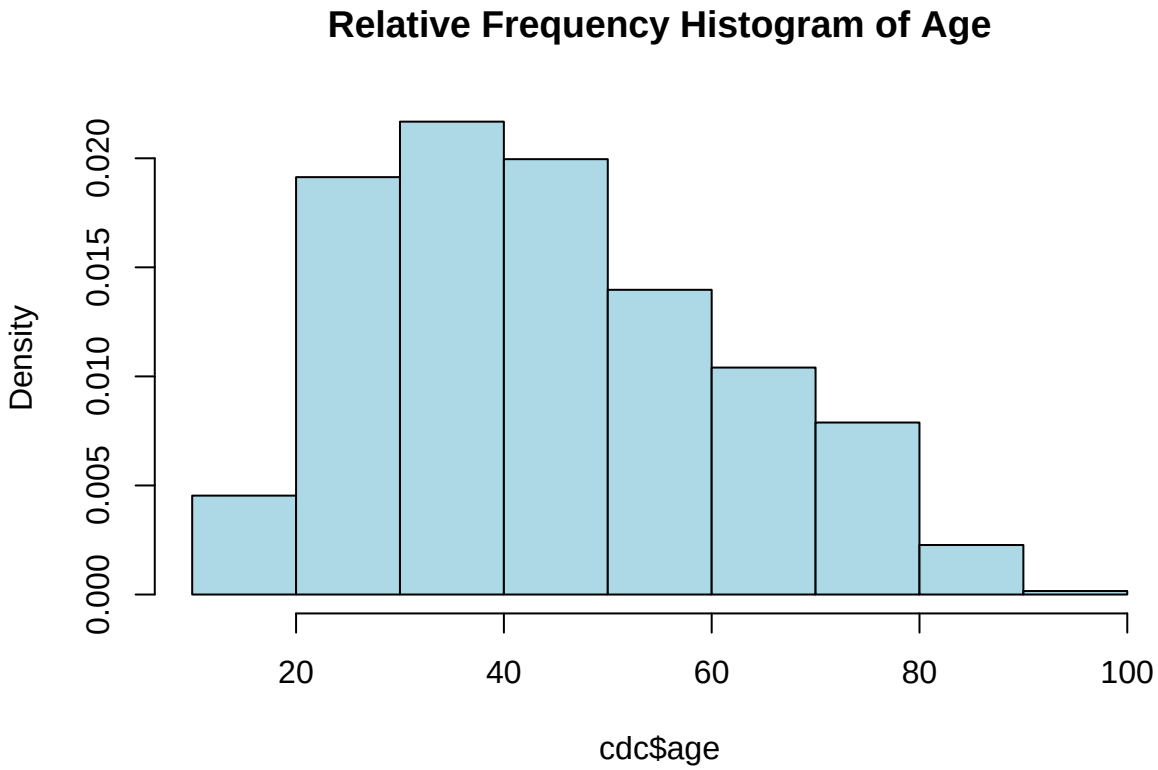

```r
hist(cdc$age, breaks = 50, col='lightblue')
```



How do these two histograms compare?

A relative frequency histogram is created by setting `freq=FALSE` in the `hist()` function. Using the code below, create a probability histogram for `age`.

```
hist(cdc$age, breaks = 10, col = 'lightblue', freq = FALSE, main = 'Relative Frequency Histogram of Age
```



Approximately what percent of those polled were between 60 and 70 years of age?