Lecture 5: Graphics in Base R

STAT 450, Fall 2021

Preliminaries

Numerical variables take on numerical values that are usually measurements or counts. It makes sense to take the sum or mean of a numerical variable. Some examples: height, weight, temperature

Categorical variables take on values that fall into distinct categories. Some examples: gender, education level

The types of graphics that we consider depend on whether the variable is numerical or categorical.

In this lecture we will discuss graphics in base R, which is the original system for creating graphics in R. You do not need to load any packages to use the plotting functions in base R. Next week we will move on to a more modern R graphics package called `ggplot2`.

CDC Data Set

Here we consider survey data from the Centers for Disease Control and Prevention (CDC).[1] The data set contains information on a random sample of 1000 people from the survey. Run the following command to read the data set into R:

```
cdc <- read.csv("https://ericwfox.github.io/data/cdc1000.csv")
```

The function `read.csv()` reads the data set into R from the specified web address. The data set is in a CSV format (comma delimited values). Type `help(read.csv)` into the console to learn more about this function.

[1] Source: http://www.cdc.gov/brfss

To preview the data set and check the dimension type the following commands:

```
head(cdc)
```

```
##     genhlth exerany hlthplan smoke100 height weight wtdesire age gender
## 1 very good       1        1        0     64    145      135  43      f
## 2 very good       0        1        0     64    174      174  20      f
## 3      fair       0        1        1     72    255      220  33      m
## 4 very good       1        1        1     69    190      175  57      m
## 5 very good       1        1        0     70    150      150  47      m
## 6      good       0        1        0     70    175      160  44      m
```

```
dim(cdc)
```

```
## [1] 1000    9
```

We can see clearly now that the data frame contains 1000 rows and 9 columns (variables). Each of the variables corresponds to a question that was asked in the survey. Descriptions of the variables are provided below:

- `genhlth`: a categorical variable indicating general health, with categories excellent, very good, good, fair, and poor
- `exerany`: a categorical variable, 1 if the respondent exercised in the past month and 0 otherwise
- `hlthplan`: a categorical variable, 1 if the respondent has some form of health coverage and 0 otherwise
- `smoke100`: a categorical variable, 1 if the respondent has smoked at least 100 cigarettes in their entire life and 0 otherwise
- `height`: a numerical variable, respondent's height in inches
- `weight`: a numerical variable, respondent's weight in pounds
- `wtdesire`: a numerical variable, respondent's desired weight in pounds
- `age`: a numerical vector, respondent's age in years
- `gender`: a categorical variable, respondent's gender

Tables and Bar Plots

For categorical variables, the function `table()` counts the number of observations falling in each category. For example, to see the number of people who have smoked at least 100 cigarettes in their lifetime, type

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

```
##
##   0   1
## 537 463
```

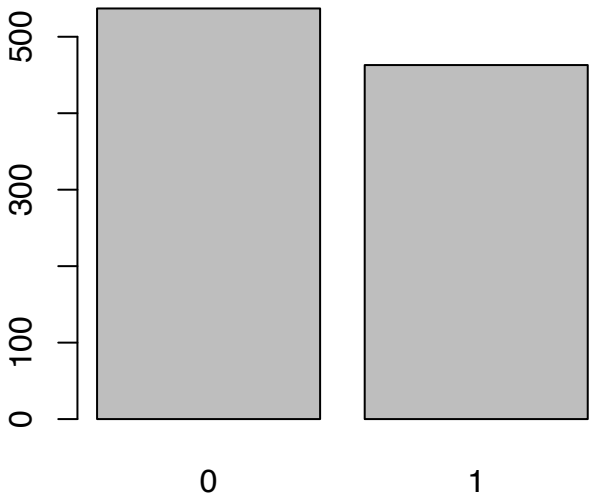
or instead to look at the proportions, type

```
table(cdc$smoke100)/1000
```

```
##
##     0     1
## 0.537 0.463
```

Next, to make a bar plot of the entries in the table, put the table inside the `barplot()` command.

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

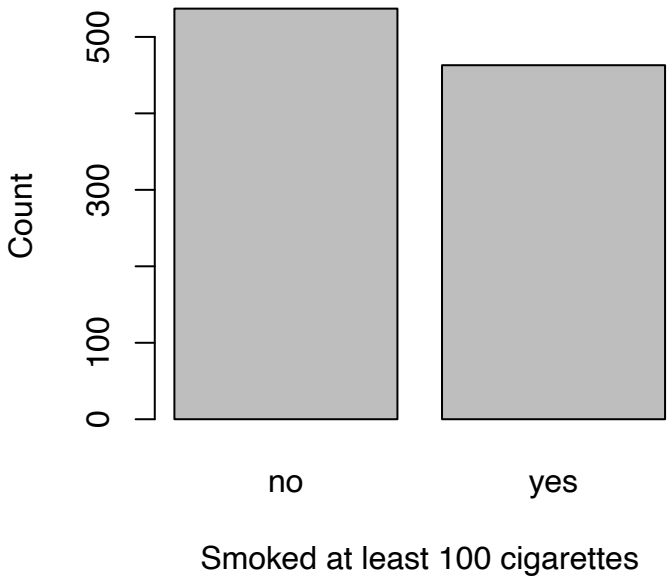


Notice what we have done here: We computed the table of `cdc$smoke100` and then applied the graphical function, `barplot()`. This is an important concept: R commands can be nested. You could also break this into two steps by typing the following:

```
smoke_tb <- table(cdc$smoke100)
barplot(smoke_tb)
```

The bar plot can be improved by adding more descriptive labels.

```
barplot(table(cdc$smoke100), xlab = "Smoked at least 100 cigarettes",
        ylab = "Count", names.arg = c("no", "yes"))
```



The `table()` command can also be used to make a contingency table with two categorical variables. For example:

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

```
##
##       0   1
##   f 301 211
##   m 236 252
```

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 include the row and column totals use `addmargins()`.

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

```
##
##          0    1  Sum
##   f    301  211  512
##   m    236  252  488
##   Sum  537  463 1000
```

Exercise: According to the contingency table above, what proportion of males have smoked at least 100 cigarettes?

Factors

Consider a table of `genhlth`, which is a categorical variable for the general health of a respondent.

```
class(cdc$genhlth)
```

```
## [1] "character"
```

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

```
##
## excellent      fair      good      poor very good
##       225        97       285        39       354
```

By default, R orders the categories alphabetically, which is not very useful here. We can change the ordering by using `factors`, which are a type of object in R used represent categorical data.

```
health_levels <- c("poor", "fair", "good", "very good", "excellent")
genhlth_fct <- factor(cdc$genhlth, levels = health_levels)
table(genhlth_fct)
```

```
## genhlth_fct
##      poor      fair      good very good excellent
##        39        97       285       354       225
```

```
class(genhlth_fct)
```

```
## [1] "factor"
```

As demonstrated above, the `factor()` function creates a factor, and the `levels` argument specifies the ordering of the categories.

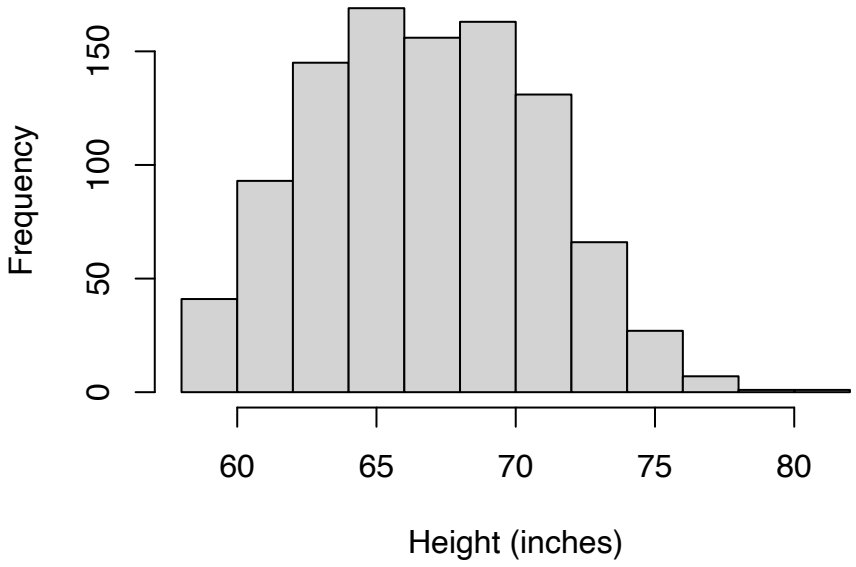
Exercise: Create a bar plot of `genhlth` with categories ordered as poor, fair, good, very good, excellent.

Histogram

Histograms are a useful way to visualize the distribution of a single numerical variable. To construct a histogram, the range of the data is divided into bins of equal width. Then the number of observations falling in each bin are counted. The counts are plotted as rectangles over each bin.

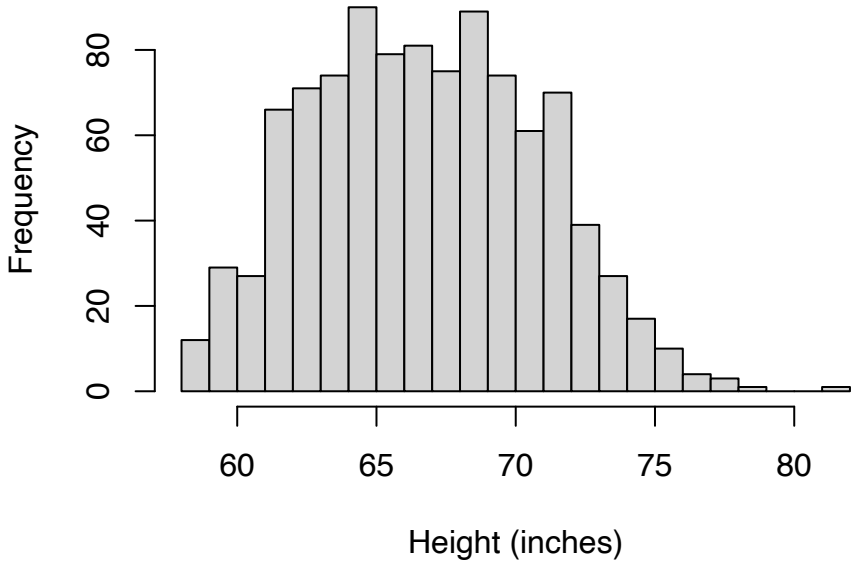
The `hist()` function creates a histogram in R. For example:

```
hist(cdc$height, xlab = "Height (inches)", main = "")
```



The argument `breaks` can be used to control the number of bins. Although, the default number of bins is usually adequate.

```
hist(cdc$height, breaks = 30, xlab = "Height (inches)", main = "")
```



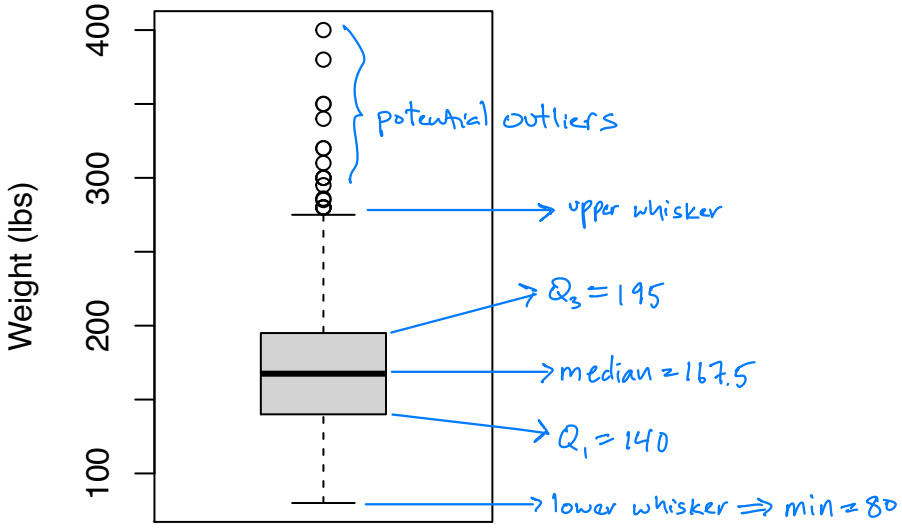
Box Plot

A box plot is another useful was to display the distribution of a numerical variable, and identify outliers.

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

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    80.0   140.0   167.5   171.1   195.0   400.0
```

```
boxplot(cdc$weight, ylab = "Weight (lbs)")
```



The "box" displays the first quartile (Q_1), median, and third quartile (Q_3), respectively. Recall that the first quartile is the value such that 25% of the data falls below, the median is the value such that 50% of the data falls below, and the third quartile is the value such that 75% of the data falls below.

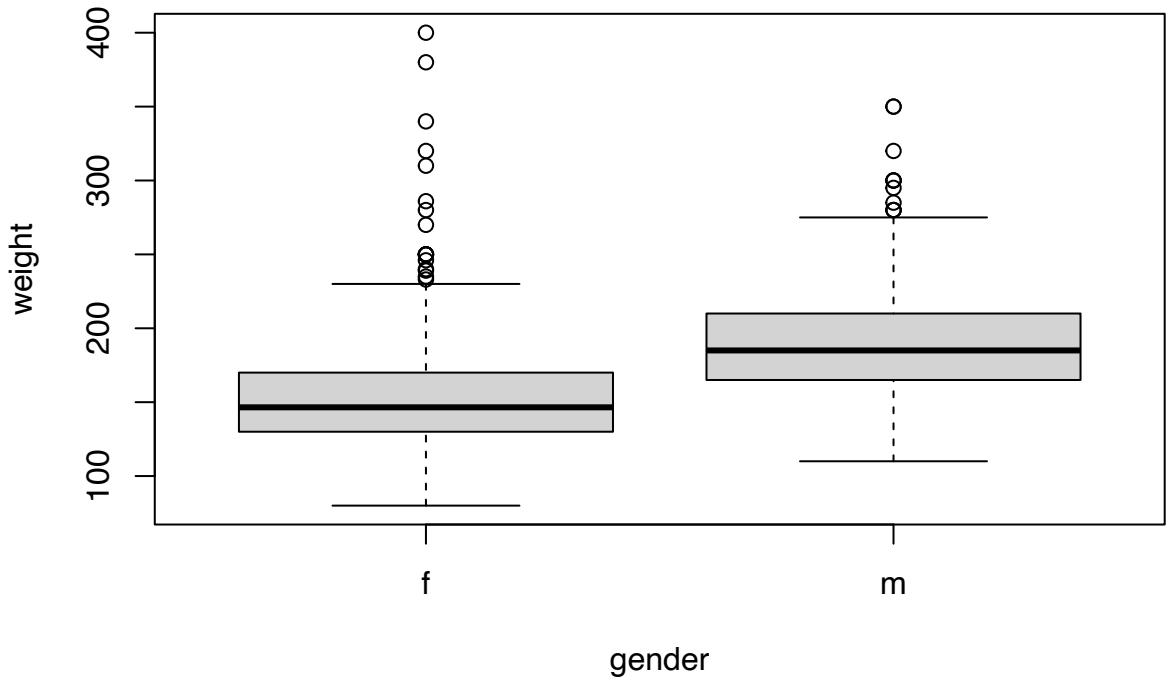
Any points that are outside the whiskers are considered outliers. Technically, the rule is that points that are greater than $Q_3 + 1.5 * IQR$ or less than $Q_1 - 1.5 * 1QR$ are classified as outliers, where $IQR = Q_3 - Q_1$ is the interquartile range.

Exercise: Make a histogram and box plot for `wtdesire`. Describe the shape of the distribution.

Side-by-Side Box Plot

The plot below shows box plots of weight for the males and females separately.

```
boxplot(weight ~ gender, data = cdc)
```

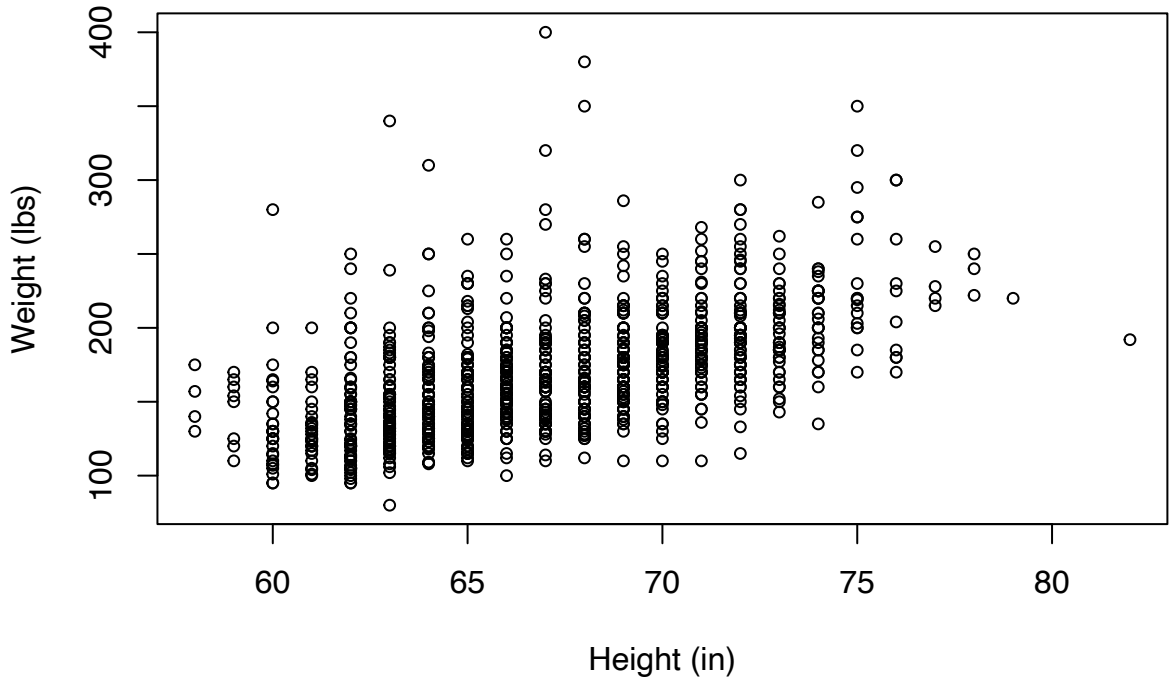


This function uses the formula notation `y ~ x`, where `y` is a numerical variable, and `x` is categorical.

Scatter Plot

Last, scatter plots are used to show the relationship between two numerical variables. We have already introduced scatter plots in a previous lecture (lecture 3). To make a scatter plot use the `plot()` function.

```
plot(cdc$height, cdc$weight,
     xlab = "Height (in)", ylab = "Weight (lbs)", cex=0.75)
```



The argument `cex` controls the points size (magnification relative to a default of 1).

Alternatively, the following code produces the same plot:

```
plot(weight ~ height, data = cdc,
     xlab = "Height (in)", ylab = "Weight (lbs)", cex=0.75)
```

This code is similar to the side-by-side box plot. First, `weight ~ height` specifies that `weight` should be plotted on the y-axis, and `height` on the x-axis. Second, `data = cdc` specifies the data frame containing the variables (so we don't need to use the extraction operator `$` when calling the variables in the function).