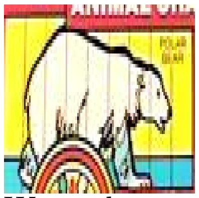

Crackers

We explore a data set on crackers using the tools of exploratory data analysis.

1 The crackers data set

We look at a data set containing various nutritional information gleaned from the side of cracker boxes at a supermarket in Oberlin Ohio [1]. To access the data we download it from a website by issuing the command:

```
> crackers = read.csv("http://www.math.csi.cuny.edu/st/R/crackers.csv")
```

The variable `crackers` contains the data. The data set contains measurements on 11 variables with 92 different types. This can be seen with the command

```
> dim(crackers)
```

```
[1] 92 11
```

How might we view this data? We can see all the values at once by typing the name of the data set, `crackers`. However, the values will quickly scroll by. The `edit()` function provides a better alternative:

```
> edit(crackers)
```

A basic spread-sheet like window should open showing the data set. *You will be unable to proceed until this window is closed.*

Question 1: Scroll through the `Company` variable and note any companies you do not recognize.

Question 2: Which company has the most cracker products on the shelves? How many different products do they have?

The numeric variables are all per-serving measurements. To see what variables are available, the variable names can be read off from the spread sheet, or printed out using the `names()` function:

```
> names(crackers)
```

```
 [1] "Company"            "Product"            "Crackers"
 [4] "Grams"              "Calories"           "Fat.Calories"
 [7] "Fat.Grams"          "Saturated.Fat.Grams" "Sodium"
[10] "Carbohydrates"      "Fiber"
```

[1] This data set has been contributed to the *Journal of Statistical Education* by Carolyn Cuff of Westminster College. Some of the variable names were shortened from the original. The data is census data for all crackers sold at this grocery chain. Can you tell what the chain is?

What we would like to be able to do is access the data in the variables. In order to do this easily, we `attach` the data set:

```
> attach(crackers)
```

Now we can refer to the variables by name, such as `Product` (case is important).

2 Categorical data

Categorical data may be summarized with tables, barplots, or even pie charts. The functions to produce these are `table()`, `barplot()` and `pie()`.

To see how these are made do the following exercises.

Question 3: `Company` is a categorical variable. You can make a table of values with the `table()` function:

```
> table(Company)
```

Do so. What are the top three companies?

Question 4: To make a barplot, you use the `barplot()` function on the tabulated data. To see why compare the outputs of these two commands:

```
> barplot(Company)
> barplot(table(Company))
```

(The extra argument `las=2`, as in `barplot(table(Company),las=2)` will produce a slightly better graph.) Explain the differences.

Question 5: A pie chart is made with the function `pie()`, as in

```
> pie(table(Crackers))
```

Make a pie chart of the data. A common gripe about pie charts is one can't compare areas easily. From your graphic can you tell if "Adrienne" sells more different products than "Old London?"

3 Numeric variables

Most of the variables in the data set are numeric. Numeric summaries are available through the functions `mean()`, `sd()`, `median()`, `IQR()`, and `summary()`.

Question 6: Apply the five functions above to the `Calories` data set. What values do you get?

Question 7: The grams per serving variable, `Grams`, has missing data. When this is the case, the extra argument `na.rm=TRUE` is often needed. Verify that `mean(Grams)` is not what is wanted, but

```
> mean(Grams, na.rm=TRUE)
```

provides the expected answer.

There are a number of graphical means to summarize a data set. For instance stem and leaf diagrams, dotplots, histograms, densities, and boxplots. All of these devices allow us to quickly visualize the following: the minimum value of the data, the maximum value, the range, where the bulk of the data sits, if there are any values far from the bulk, and the general shape of the data.

The `R` functions used are `stem()`, `stripchart()`, `hist()`, `density()`, and `boxplot()`

3.1 Stem and Leaf diagrams

For instance, a stem-and-leaf diagram of the calories per serving is made as follows:

```
> stem(Calories)

  The decimal point is 1 digit(s) to the right of the |

   4 | 00
   6 | 000000000000000000000000000000000
   8 | 000000000000
  10 | 00
  12 | 0000000000000
  14 | 000000000000000000000000000
  16 | 000
```

From the diagram we can see that the smallest number is 40, the largest 160, the range is 120 and that there are two distinct "peaks" around 60 and 140.

Question 8: Issue the command

```
> stem(Grams)
```

To make a simple stem and leaf diagram of the grams per serving. What is the range of values? Describe the shape of the data set.

Question 9: The choice of stem is automatically determined by `stem()`. However, you can override it by using the `scale=` argument. For instance, try the command

```
> stem(Grams, scale=1/2)
```

Does this make a better stem and leaf diagram than before? Explain why

3.2 Dotplots

A dotplot can be produced with the `stripchart()` function. [2]

The simplest use of `stripchart()` will not stack points when there are ties, you must ask for this behavior. For example, to make the dotplot (Figure 1) of the data in `Grams` (grams per serving), we issue the following command: [3]

```
> stripchart(Grams, method = "stack")
```

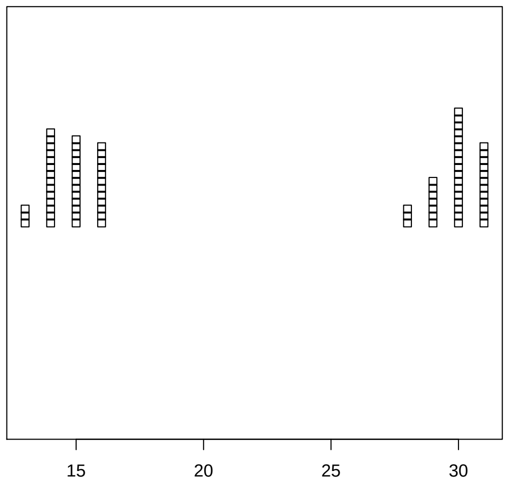

Figure 1: Dotplot of grams per serving

Question 10: Make a stripchart of the data in the variable `Calories`.

1. What is the range of the data?
2. What is the shape of the data?
3. Estimate the mean of the data using the idea of balancing.
4. Are there any crackers that are "average" by this measure?
5. Check your estimate, by computing the mean.

[2] Alternatively, you can use the `DOTplot()` function that may be installed using `source("http://www.math.csi.cuny.edu/st/R/DOTplot.R")`

[3] If you use `DOTplot()` the graphic is produced by `DOTplot(Grams)`.

3.3 Histograms

The dotplots made with this much data are pretty busy, a histogram may better show the key features of the data set.

Histograms are made using the function `hist()`, as in its use to produce a histogram of the amount of sodium:

```
> hist(Sodium)
```

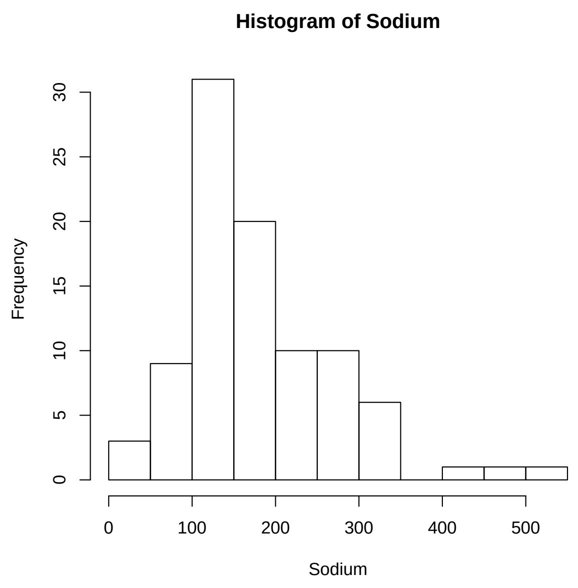

Figure 2: Histogram of sodium amount in crackers.

Question 11: Based on the histogram in Figure 2 do the following:

1. Estimate the range of the data
2. Estimate the mean of the data set
3. Estimate the median of the data set
4. Describe the shape of the data

Check your numeric answers using the appropriate function.

Question 12: Produce a histogram of the number of crackers per serving. Based on the histogram do the following

1. Estimate the range of the data
2. Estimate the mean of the data set

3. Estimate the median of the data set

4. Describe the shape of the data

Check your numeric answers using the appropriate function.

Density estimates

A density estimate is similar to a histogram, in that it visually summarizes the shape of a distribution. However, they have advantages, in that they are more directly related to the population density and many may be used simultaneously.

Density estimates are generated using the `density()` command. This command simply produces the numbers, to visualize the density it may be added to a histogram or plotted by itself.

To plot a density by itself we use the `plot()` function in combination with `density()`. For instance, to plot the density of crackers per serving (Figure 3):

```
> plot(density(Crackers))
```

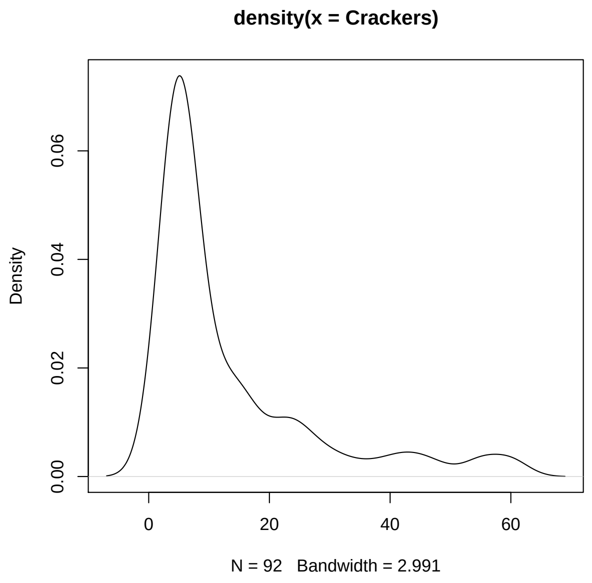

Figure 3: Density plot of crackers per serving

To add a density estimate to a histogram, requires one to first make the histogram using a probability scale (so the total area is 1). This requires the extra argument `probability=TRUE`, which we abbreviate to `prob=T`. The density plot is then *added* [4] to the existing plot using `lines()`.

For instance, a histogram with density estimate of the crackers per serving is generated (Figure 4), as with:

[4]If you close the graphics window before adding the line, you will get an error message.

```
> hist(Crackers, prob = T)
> lines(density(Crackers))
```

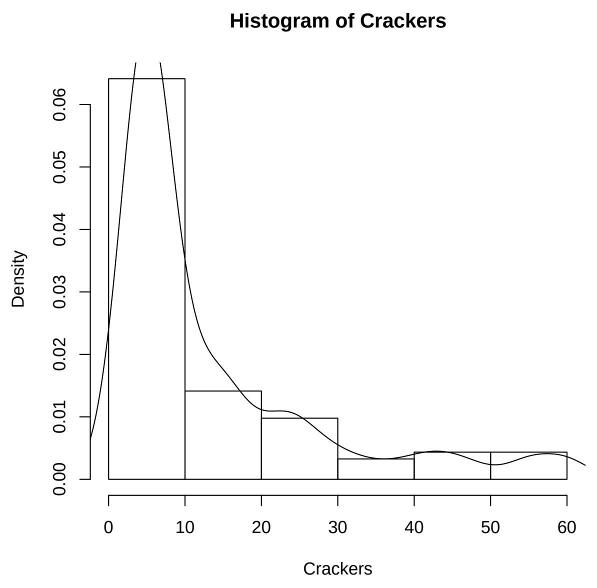

Figure 4: Histogram and density estimate of the number of crackers per serving

Question 13: Based on Figure 3 answer the following:

1. Is the data set symmetric or skewed?
2. Are there long tails? If so, which?
3. Estimate the mean and median. Which is greater and why.
4. Check your estimates using the functions `mean()` and `median()`. Were you close?

Question 14: Make a density plot of the variable `Sodium`. Based on this, is the population skewed or symmetric? Long tailed?

3.4 Boxplots

Boxplots succinctly describe a numeric data set in a manner that lends itself to multiple comparisons. From a boxplot we can quickly identify all of the following: the center, the spread, the range, symmetry or skew, and tail length.

Boxplots are produced with the `boxplot()` function. For example, a boxplot of the number of crackers per serving (Figure 5) may be made with

```
> boxplot(Crackers)
> title("Number of crackers per serving")
```

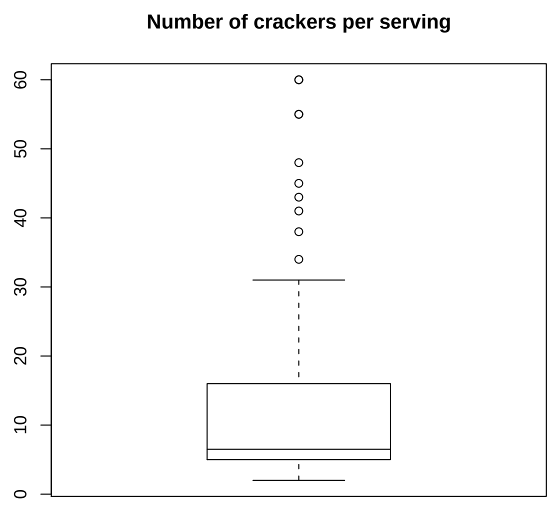

Figure 5: Boxplot of number of crackers per serving

Question 15: From Figure 5 answer the following:

1. What is the "center" of the variable?
2. What is the spread?
3. Is the data set skewed?
4. Do you expect the mean or median to be the largest? Why? Check it.
5. Are there any "outliers?" If so, how many?

Question 16: Make a boxplot of the `Fat.Grams` variable. From your figure answer the following:

1. What is the "center" of the variable?
2. What is the spread?
3. Is the data set skewed?
4. Do you expect the mean or median to be the largest? Why? Check it.
5. Are there any "outliers?" If so, how many?

Question 17: If there are two variables, one numeric and one categorical of the same size, it is easy to produce parallel boxplots for each level of the categorical variable. The syntax is to use the argument `numeric ~ categorical` for the argument. For instance: [5]

```
> boxplot(Calories ~ Company,las=2)
```

Based on your plot, answer the following:

1. Which company has the widest range?
2. Which company has the largest median?
3. Which company has the largest IQR?
4. Why do some companies have just a short horizontal line?

4 Using just some of the data

A nice feature of R is the ability to easily extract just part of the data in a variable. For instance a histogram of the grams per serving variable shows two distinct modes. How can we get information about the left one, say? Looking at the histogram, we see that 22 separates the two modes, so we only want the values of `Grams` less than 22.

Question 18: The command

```
> Grams < 22
```

has an interesting output. Explain what it is, and why one should read this command like a question.

The special syntax

```
> Grams[ Grams < 22 ]
```

extracts just the part of the variable `Grams` for which `Grams < 22`. The "question" is answered with `TRUE` and `FALSE` and the square brackets, `[]`, allow one to use these answers to extract the corresponding values from the variable `Grams`.

Question 19: You don't need to use the same variable. For instance to look at the calories per serving for the low-gram crackers we have

```
> x = Calories[Grams < 22]
```

Run this command, then make a histogram of `x`. Describe its shape.

[5]The extra argument `las=2` is optional. It turns the labels so that they are easier to read. You can also enlarge the plot window to see more text.