

Crackers

Getting hungry just thinking about learning some statistical analysis? If you are like many, you might reach for a box of crackers as a “healthy alternative” to more sugar-laden foods. But just how healthy are crackers? For instance, the Los Angeles Unified School District Obesity Prevention Motion identified Pepperidge Farm’s Cheddar Goldfish as an unapproved snack food. ¹ To investigate what is contained in a crackers box, we learn the tools of exploratory data analysis.

1 The crackers data set

We’ll use a data set containing various nutritional information (see Table 1) gleaned from the website www.dietfacts.com and the sides of cracker boxes at a supermarket in Oberlin Ohio. ²

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. 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. A warning: *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 seems to have the most cracker products on the shelves? Guess how many different products are on the shelves?

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)
```

¹<http://cafe-la.lausd.k12.ca.us/usnacks.htm>

² This data set has been contributed to the *Journal of Statistical Education*, <http://www.amstat.org/publications/jse/>, by Professor Carolyn Cuff of Westminster College. Many of the questions in this project follow those outlined in her accompanying paper. Some of the variable names were shortened from the original. The data is census data for all crackers sold at this supermarket. Can you tell what the supermarket was?

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

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"
```

For instance, the variable `Crackers` records the crackers per serving and the variable `Fat.Calories` records the calories due to fat per serving.

Nutrition Facts	
Serving Size 55 crackers (55g)	
Servings Per Container 12	
Amount Per Serving	
Calories 150	Calories From Fat 60
%Daily Values*	
Total Fat 6g	11%
Saturated Fat 1.5g	7%
Sodium 250mg	10%
Total Carbohydrates 19g	6%
Dietary Fiber 0g	0%
* Percent Daily Values are based on a 2,000 calorie diet.	

Table 1: Example of a nutritional label. This one for Pepperidge Farm's Cheddar Goldfish.

2 Numeric variables

Most of the variables in the data set are numeric. For such data, we have numeric summaries and graphical summaries available. Each has their place. Graphical summaries allow us to quickly see several features of a distribution at once, whereas numeric summaries allow us to quantitatively compare our data to some other data set or pre-conceived notion about our data.

2.1 Numeric summaries

Numeric summaries we use summarize the center (`mean()` and `median()`), the spread (`range()`, `sd()` and `IQR()`), or even the position within a data set (`quantile()` and `scale()`).

For example, looking at the `Crackers` variable, we can compare centers with

```
> mean(Crackers)
[1] 13.35870
> median(Crackers)
[1] 6.5
```

The difference leads us to believe the data set is not symmetric. The spread is summarized with


```
> sd(Crackers)
[1] 14.18603
> IQR(Crackers)
[1] 11
```


The IQR is computed from the 0.25 and 0.75 quantiles, which may be found with


```
> quantile(Crackers, c(0.25, 0.75))
25% 75%
5 16
```

The 95th percentile would be returned with

```
> quantile(Crackers, 0.95)
95%
46.35
```

 Question 3: Apply the functions `mean()`, `median()`, `range()`, `IQR()`, and `sd()` above to the `Calories` data set. What values do you get?

 Question 4: Of the five numbers found in the previous exercise, which one is not available from the output of `summary()` applied to the data set `Calories`.

 Question 5: The grams per serving variable, `Grams`, has missing data. You can verify this by typing the command

```
> Grams
```

The missing values are coded `NA`, read “not available.” When there are missing values, the extra argument `na.rm=TRUE` is often needed (this removes `NA` values). Verify that `mean(Grams)` is not what is wanted, but

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

computes the desired answer.

 Question 6: For the `Calories` data, find the 5th and 95th percentiles.

2.2 Graphical summaries of numeric data

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 center of the distribution, a sense of spread, 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 to produce these graphics are `stem()`, `stripchart()`, `hist()`, `density()`, and `boxplot()`

2.3 Stem and leaf diagrams

Stem and leaf diagrams are produced in R using `stem()`.³

For instance, a stem-and-leaf diagram of the calories per serving is produced with:

```
> stem(Calories)
```

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

```

 4 | 00
 6 | 00000000000000000000000000000000
 8 | 0000000000000
10 | 00
12 | 00000000000000
14 | 000000000000000000000000000000
16 | 000

```

From the diagram we can see that the smallest recorded number is 40, the largest 160, the range is 120. The “shape” of this data set is *bimodal*—that is, there are two distinct “peaks” (around 60 and 140).



Question 7: Issue the command

```
> stem(Grams)
```

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



Question 8: Verify, that the command `stem(Calories)` actually truncates some of the data, by comparing the diagram with the output of `range(Calories)`. Explain.





Question 9: The choice of stem is automatically determined by `stem()`. It may not always produce the best results. However, you can override the choice by using an extra `scale=` argument. For instance, try the command

³An alternative to this is to use the `stem.leaf()` function in the `Rcmdr` package. If present, this is loaded with the command `library(Rcmdr)`. This package provides a GUI to the R workspace. Alternatively, you can download a copy of the function with the command `source("http://www.math.csi.cuny.edu/st/R/stem.leaf.R")`.

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

Find the range of the data and compare to your previous answer. Does this make a better stem and leaf diagram than before? Explain why?

 Question 10: Remake a stem and leaf diagram of **Calories** finding a value for **scale=** that uses a stem recording the tens digits.

 Question 11: Make a stem-and-leaf diagram of the crackers per serving variable **Crackers**. What is the range of the data? This shape of this data set is skewed right. What type of cracker do you think has 55 or more per serving? Check your answer by quickly scrolling through the data set.

2.4 Dotplots

A dotplot can be produced with the `stripchart()` function.⁴

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:⁵

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

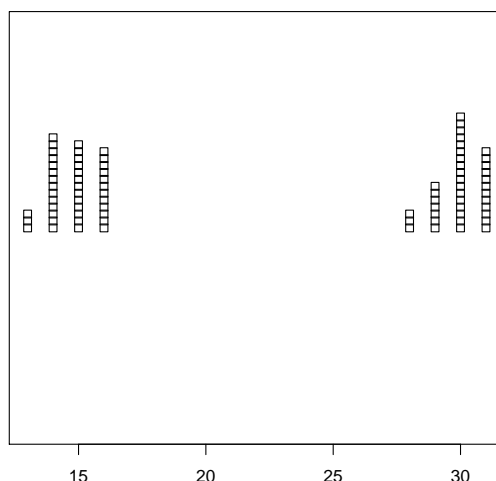


Figure 1: Dotplot of grams per serving

⁴ Alternatively, you can use the `DOTplot()` function that may be installed with the command `source("http://www.math.csi.cuny.edu/st/R/DOTplot.R")`

⁵ If you use `DOTplot()` the graphic is produced by `DOTplot(Grams)`.

Some things we can quickly see from Figure 1 are that the range is roughly 13 to 31, the mean is around 20 (balance point), the median is somewhere in the left cluster of values, and the shape is bimodal. To check our guesses on the mean and median we have:

```
> summary(Grams)
```

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	NA's
13.00	15.00	16.00	22.11	30.00	31.00	11.00



Question 12: 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.

2.5 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()`⁶, as in its use to produce a histogram of the amount of sodium per serving:

```
> hist(Sodium)
```



Question 13: 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 14: Produce a histogram of the number of crackers per serving. Based on the histogram do the following

1. Estimate the range of the data

⁶Or, one can use the function `truehist()` from the **MASS** package. This is available after loading the package, which can be done with the command `library(MASS)`

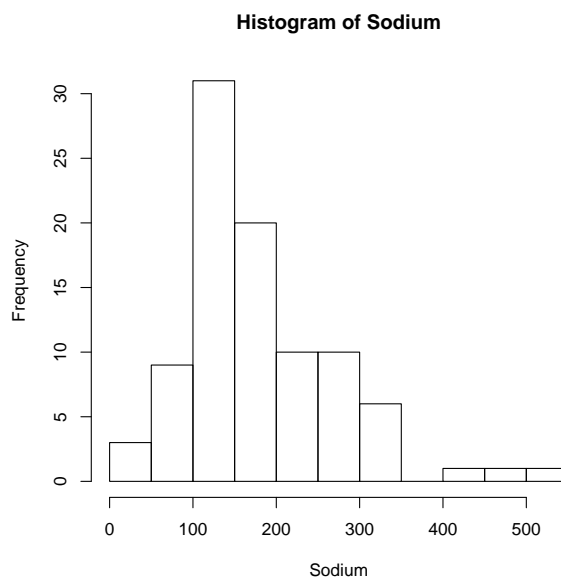


Figure 2: Histogram of sodium amount in crackers.

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))
```

To add data to a graphic ⁷ we use the function `points()` and `lines()`. The former adds the data as points, the latter connects the points with lines. We want to plot a curve, so we would use `lines()`:

⁷If you close the graphics window before adding the line, you will get an error message.

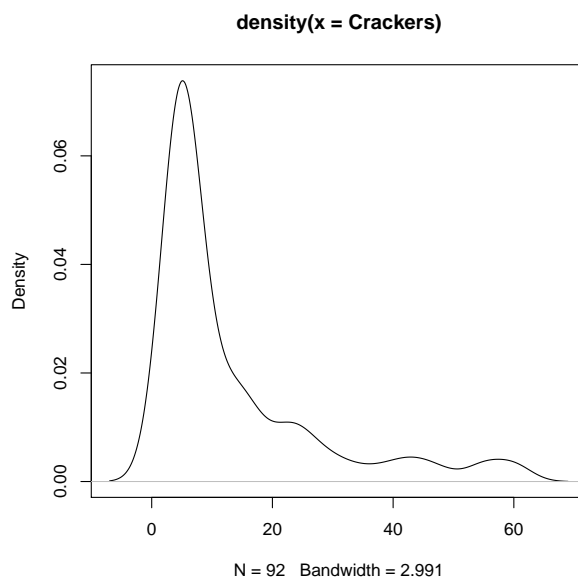


Figure 3: Density plot of crackers per serving

```
> lines(density(Crackers))
```

(That is, replace `plot()` with `lines()`.)

When adding a density on top of histogram, one first needs to make the histogram having a total area of 1. This is achieved using the extra argument `probability=TRUE` to `hist()`. We abbreviate this to `prob=T`.⁸

For instance, a histogram with a density estimate of the crackers per serving overlaid may be generated with (Figure 4):

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



Question 15: 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 16: Make a density plot of the variable `Sodium`. Based on this, is the population skewed or symmetric? Long tailed?

⁸This is unnecessary when using `truehist()` from the `MASS` package.

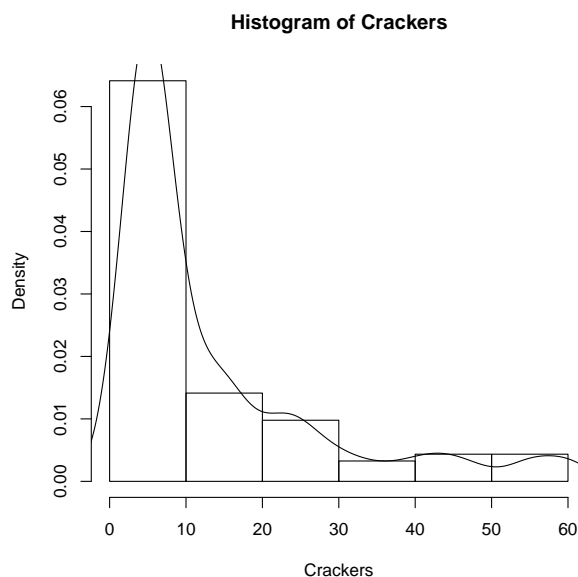



Figure 4: Histogram and density estimate of the number of crackers per serving

 Question 17: One can add the actual data points to the histogram and density estimate using the `rug()` function. To see what is done, with the density plot for `Sodium` still open issue the following command:


```
> rug(Sodium)
```

2.6 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")
```

 Question 18: 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.

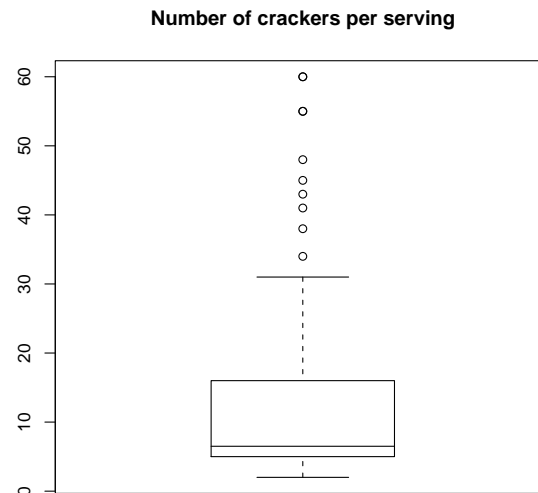




Figure 5: Boxplot of number of crackers per serving

5. Are there any “outliers?” If so, how many?

 Question 19: Make a boxplot of the `Fat.Grams` variable. Compare the values plotted in the boxplot with those produced by the command ⁹

```
> summary(Fat.Grams)
```

 Question 20: If there are two variables, one numeric and one categorical (numeric will also work) of the same size, it is easy to produce parallel boxplots for each level of the categorical variable. The syntax is to use `numeric ~ categorical` for the argument. For instance: ¹⁰

```
> 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?

⁹The `summary` command finds the actual quantiles, whereas the boxplot technically uses the related—but sometimes different—hinges.

¹⁰The 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.

3 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. The cracker data set lends itself to analysis of the sub-populations. For example, a histogram of calories per serving shows two distinct modes—one near 60 and one near 130. What causes this? Could it be the difference between low-fat crackers and non-dietetic crackers?

3.1 Extraction by index

A look at the variable `Product` in the data set shows that the following products (by index) are labeled by their manufacturer as low-fat or low-sodium crackers:

low fat: 10 12 19 23 28 30 42 50 51 54 58 68 73 86 90

low sodium: 8 24 29 31 41 55 56 66 71 80 92

We enter these into vectors for storage as follows:


```
> low.fat = c(10, 12, 19, 23, 28, 30, 42, 50, 51, 54, 58, 68, 73,
+            86, 90)
> low.sodium = c(8, 24, 29, 31, 41, 55, 56, 66, 71, 80, 92)
```


To refer to the values in the `Calories` variable that correspond to the low fat crackers is done by extraction. To extract just the specified indices is made possible using the square-bracket notation: `[]`. For instance:


```
> Calories[low.sodium]


[1] 70 80 70 70 150 60 60 140 150 150 60
```

This command returns the 11 values corresponding to the indices.

 Question 21: Make a stem-and-leaf diagram of the `Calories` variable for the low-sodium crackers. Is the data still bimodal?

 Question 22: Make a stem-and-leaf diagram of the `Calories` variable for the crackers labeled as low in fat. Is the data still bimodal?


 Question 23: You can combine indices using `c()`, or eliminate potential duplicates with `union(low.fat, low.sodium)`. Do so, then make a histogram of the `Calories` variable for this larger restricted data set. Describe the data set, and contrast it to the full data set stored in `Calories`.

 Question 24: Parallel boxplots of the calories per serving for low-fat crackers and other crackers can be produced as follows:

```
> boxplot(Fat.Grams[low.fat], Fat.Grams[-low.fat])
```

This uses the special convention for negative indices—to exclude the values.

Produce the boxplot. (The leftmost boxplot refers to the first variable, etc.) One would expect that the crackers labeled as low in fat would generally have less fat than those not labeled as such. Based on the boxplots, write a paragraph examining how the boxplots would actually show such an expected relationship, and comment on whether they do.

 Question 25: Produce parallel boxplots of the **Sodium** variable for the crackers labeled as low in sodium and those which aren't labeled this way.

Based on the boxplots, comment on whether the manufacturer's labeling of a product as low sodium implies the crackers are lower in sodium when compared with non-low-sodium labeled crackers.

 Question 26: The crackers with extra cheese or bacon flavor are


15, 37, 41, 42, 45, 46, 63, 65, 79, 80, 82, 85, 86, and 87

Make parallel boxplots of the crackers with these extra flavors and those without. Does it appear that the distribution of the number of Calories is dependent on whether the cracker has a “cheesy” or “bacony” flavor? Explain.

3.2 Extraction using logical data

Extraction may also be done in response to a “question.” A sample question would be “Which indices correspond to values where the calories per serving are less than 100?” R is a little more succinct. This is “asked” as:

```
> Calories < 100
```

 Question 27: Run the above command and interpret the output. What is computed for each observation in the **Calories** variable?

The actual indices are found in combination with the `which()` function:

```
> which(Calories < 100)
```

```
[1] 1 2 3 4 5 6 7 8 9 10 17 18 19 20 21 22 23 24 29 30 31 32 33 34 35
[26] 36 53 54 55 56 58 59 61 62 64 70 74 75 76 77 78 84 88 89 90 91 92
```

The `which()` function returns the index for each **TRUE** value. A vector of **TRUE** and **FALSE** values is called a logical vector. Many functions are adapted to use logical vectors. For instance, the `sum()` function adds up all the **TRUE** values:

```
> sum(Calories < 100)
```

```
[1] 47
```

For extraction, only those values corresponding to **TRUE** are returned:

```
> low.cal = Calories < 100
> Calories[low.cal]
```

```
[1] 60 60 60 60 70 60 70 70 70 70 80 80 60 80 80 80 70 80 70 50 70 60 70 80 80
[26] 80 60 60 60 60 70 80 60 70 80 70 80 60 60 60 70 60 70 70 50 60 60
```

Variables that have only two values (TRUE and FALSE), say, or 0 and 1 are referred to as *indicator variables*. They can often make many computations easier to organize.

The negation operator, `!`, reverses the TRUE's and FALSE's. For instance, the command

```
> Calories[!low.cal]
```

```
[1] 130 120 140 140 140 130 140 140 140 140 130 150 140 150 150 140 140 160 140
[20] 160 130 140 130 120 110 150 110 140 130 130 140 140 120 130 150 140 130 150
[39] 150 140 140 120 160 140 150
```

Returns values for which the calories per serving are 100 or more (when `Calories >= 100`).



Question 28: You don't need to use the same variable. For instance to look at the products which have just a few crackers per serving we have:

```
> Product[Crackers <= 3]
```

```
[1] Bretton          Vinta
[3] Vivant           Cabaret
[5] Harvest Bakery Cornbread Harvest Bakery Multigrain
[7] Harvest Bakery Rye Stoned Wheat Thins
92 Levels: Barnum's Animal crackers ... Zesta Soup & Oyster Crackers
```

Repeat a similar command, only showing those products with more than 50 per serving. Which crackers are these?



Question 29: Use extraction to find the products where the calories per serving are more than 140. Are there any surprises in the data set? Explain what you expected and describe what you see.



Question 30: The `boxplot()` function works well with indicator variables. For example, a histogram of `Crackers`, the crackers per serving, shows that many are 10 or fewer and others, the small crackers, have many more. To break the data up by the crackers per serving and then plot boxplots for each may be done as follows:

```
> small.serv = Crackers <= 10
> boxplot(Calories ~ small.serv)
```

(The labels on the boxplots are taken from the values in `small.serv`. This is in contrast to a more cumbersome expression like

```
> boxplot(Calories[small.serv], Calories[!small.serv])
)
```

Run these commands. Do the two boxplots look as though they represent the same population?



Question 31: Create an indicator variable `low.cal` as follows

```
> low.cal = Calories < 100
```

Now make boxplots of fat grams per serving (**Fat.Grams**) broken up by the indicator variable **low.cal**.

Are there any non-low calorie crackers with fewer grams of fat than some low-calories crackers? What part of the boxplots answers this?