2.4 R IN FOCUS

Frequency Distributions for Quantitative Data

R can be used to construct frequency distributions. In this section, we will construct a frequency distribution table for the business safety data first listed in Table 2.3, which are reproduced here for reference.

Data reproduced from Table 2.3.

45	98	83	50	86
66	66	88	95	73
88	55	76	115	66
92	110	79	105	101
101	85	90	92	81
55	95	91	92	
78	66	73	58	
86	92	51	63	
91	77	88	86	
94	80	102	107	

Create a dataframe in R called "complaints" that has the above safety data observations listed in one column:

```
> complaints <- data.frame(Complaints=c(45,98,83,50,86,66,66,88,
+ 95,73,88,55,76,115,66,92,
+ 110,79,105,101,101,85,90,
+ 92,81,55,95,91,92,78,66,
+ 73,58,86,92,51,63,91,77,
+ 88,86,94,80,102,107))
```

First, let us order the data to see which businesses filed the fewest and most complaints by using the order function on the Complaint column data from the complaint dataframe:

```
> order(complaints$Complaints)
 [1]  1  4 36 12 26 33 37  6  7 15 31 10 32 13 39 30 18 43 25  3 22
[22]  5 34 41  8 11 40 23 28 38 16 24 29 35 42  9 27  2 20 21 44 19
[43] 45 17 14
```

The order function shows us that business observation 1 filed the fewest complaints, and business number 14 filed the most. This might be helpful for us if we knew which businesses those were. However, we may be more interested to know the most and fewest number of complaints filed. For that, we can use the sort function:

```
> sort(complaints$Complaints)
 [1] 45 50 51 55 55 58  63  66  66  66  66  73  73 76 77 78
[17] 79 80 81 83 85 86  86  86  88  88  88  90  91 91 92 92
[33] 92 92 94 95 95 98 101 101 102 105 107 110 115
```

We can go a step further and create a table that shows us, for each number of complaints filed, how many businesses filed that number:

```
> table(complaints$Complaints)
 45  50  51  55  58  63  66  73  76  77  78  79  80  81  83  85  86
  1   1   1   2   1   1   4   2   1   1   1   1   1   1   1   1   3
 88  90  91  92  94  95  98 101 102 105 107 110 115
  3   1   2   4   1   2   1   2   1   1   1   1   1
```

In the above output, we see that the most common numbers of complaints filed are 66 and 92: Four businesses filed 66 complaints and four businesses filed 92 complaints over the previous 3 years.

Next, we may want to know, for each value of number of complaints filed, what percentage of all of the values each represents. We can do this by using the prop.table (proportion) function:

```
> prop.table(table(complaints$Complaints))
        45         50         51         55         58         63
0.02222222 0.02222222 0.02222222 0.04444444 0.02222222 0.02222222
        66         73         76         77         78         79
0.08888889 0.04444444 0.02222222 0.02222222 0.02222222 0.02222222
        80         81         83         85         86         88
```

```
0.02222222 0.02222222 0.02222222 0.02222222 0.06666667 0.06666667
        90         91         92         94         95         98
0.02222222 0.04444444 0.08888889 0.02222222 0.04444444 0.02222222
       101        102        105        107        110        115
0.04444444 0.02222222 0.02222222 0.02222222 0.02222222 0.02222222
```

This output shows us that the value 45 makes up 2.22% of all the values. Of course, this output looks sloppy. First, there are too many values, and we can make it easier to read by rounding our output with the round() function:

```
> round(prop.table(table(complaints$Complaints)),4)

    45     50     51     55     58     63     66     73     76
0.0222 0.0222 0.0222 0.0444 0.0222 0.0222 0.0889 0.0444 0.0222
    77     78     79     80     81     83     85     86     88
0.0222 0.0222 0.0222 0.0222 0.0222 0.0222 0.0222 0.0667 0.0667
    90     91     92     94     95     98    101    102    105
0.0222 0.0444 0.0889 0.0222 0.0444 0.0222 0.0444 0.0222 0.0222
   107    110    115
0.0222 0.0222 0.0222
```

In the above code, we gave the round() function two arguments. First, we gave it the data that we wanted rounded, and second, we told it how many places we wanted our data rounded to (in this case, 4). We can make our data easier to read by multiplying our output by 100 to get the percentage:

```
> round(prop.table(table(complaints$Complaints)),4)*100

  45   50   51   55   58   63   66   73   76   77   78   79   80
2.22 2.22 2.22 4.44 2.22 2.22 8.89 4.44 2.22 2.22 2.22 2.22 2.22
  81   83   85   86   88   90   91   92   94   95   98  101  102
2.22 2.22 2.22 6.67 6.67 2.22 4.44 8.89 2.22 4.44 2.22 4.44 2.22
 105  107  110  115
2.22 2.22 2.22 2.22
```

Finally, let us assume we want to know the cumulative percentage for our values. We can use the cumsum() function. First, I saved our output and work so far by assigning it the name safety.data, and then I applied the cumsum() function to that object:

```
> safety.data <- round(prop.table(table(complaints$Complaints)),4)*100
> cumsum(safety.data)
   45    50    51    55    58    63    66    73    76    77    78
 2.22  4.44  6.66 11.10 13.32 15.54 24.43 28.87 31.09 33.31 35.53
   79    80    81    83    85    86    88    90    91    92    94
37.75 39.97 42.19 44.41 46.63 53.30 59.97 62.19 66.63 75.52 77.74
   95    98   101   102   105   107   110   115
82.18 84.40 88.84 91.06 93.28 95.50 97.72 99.94
```

You may notice that our cumulative percent does not equal 100%, and it should. Can you guess why? If you said it is due to rounding, you are correct! Remember, we rounded our data to four decimal places, which influenced how the data added up.

2.7 R IN FOCUS

Frequency Distributions for Categorical Data

R can be used to summarize categorical data that are ungrouped. We can use R to create a frequency distribution for the following hypothetical example: A group of health practitioners wants to classify children in public schools as being lean, healthy, overweight, or obese; this type of classification is common (Centers for Disease Control and Prevention, 2013; Privitera, 2016). To do this, the researchers calculated the body mass index (BMI) score of 100 children. Based on the BMI scores, they classified 15 children as lean, 30 as healthy, 35 as overweight, and 20 as obese.

As usual, let us begin by creating our dataframe:

```
> BMI <- data.frame(BMI=c("lean","healthy","overweight",
+ "obese"), Total=c(15,30,35,20))
> BMI
         BMI    Total
1       lean       15
2    healthy       30
3 overweight       35
4      obese       20
```

Similar to the previous section, we can calculate the percent (out of 100) for each category, convert that calculation to a number out of 100 (to make it easier to read), and then calculate the cumulative percentage:

```
> prop.table(BMI$Total)
[1] 0.15 0.30 0.35 0.20
> prop.table(BMI$Total)*100
[1] 15 30 35 20
> cumsum(prop.table(BMI$Total)*100)
[1] 15 45 80 100
```

As another example, instead of having the number of children classified as either lean, healthy, overweight, or obese, we may simply have classification types for each child, which we can enter into a dataframe:

```
> BMI.2 <-data.frame(BMI=c("lean","lean","healthy","overweight",
+ "lean","obese","obese","obese","lean",
+ "healthy","lean","obese","lean",
+ "overweight","lean"))
> BMI.2
          BMI
1        lean
2        lean
3     healthy
4  overweight
5        lean
6       obese
7       obese
8       obese
9        lean
10    healthy
11       lean
12      obese
13       lean
14 overweight
15       lean
> table(BMI.2)
BMI.2
  healthy    lean   obese overweight
        2       7       4          2
> prop.table(table(BMI.2))
BMI.2
   healthy      lean     obese overweight
 0.1333333 0.4666667 0.2666667  0.1333333
> cumsum(c(.47,.13,.13,.27))
[1] 0.47 0.60 0.73 1.00
```

In this example, we see that there are 7 lean children, 2 healthy children, 2 overweight children, and 4 obese children. The prop.table() function calculates the percentage for each of these categories, and we can again use the cumsum() function to calculate the cumulative percentage.

2.11 R IN FOCUS

Histograms, Bar Charts, and Pie Charts

To review, histograms are used for continuous or quantitative data, and bar charts and pie charts are used for discrete, categorical, or qualitative data. As an exercise to compare histograms, bar charts, and pie charts, we can construct these graphs for the same data by treating the data as a simple set of general values. Suppose we measure the data shown in Table 2.16.

Table 2.16 A Sample of 20 Values

1	4	5	7
2	3	6	8
3	6	7	9
2	6	5	4
4	5	8	5

Because we are not defining these values, we can just call the variable "numbers."

Creating a histogram in R is super simple! Of course, we are going to take advantage of this and learn some new tricks in R. First, create the dataframe:

```
> numbers <- data.frame(numbers=c(1,4,5,7,2,3,6,8,3,6,7,9,2,6,5,
+ 4,4,5,8,5))
```

Next, create a histogram using the hist() function:

```
hist(numbers$numbers)
```

This does the job, but the labels are not professional. For example, the *x*-axis label is numbers$numbers, which can be changed to our variable name, Numbers. We can also change the title of our histogram. These changes can be done by passing more information to the arguments of the hist() function. You can see the various arguments that the hist() function takes by looking them up in the help menu:

```
> help(hist)
```

Let us now improve our histogram:

```
> hist(numbers$numbers, n=20, main="Histogram",xlab="Numbers")
```

The "main" argument tells the hist() function what the title of the histogram should be, and the "xlab" argument renames the *x*-axis label. Experiment with the "n=20" argument by changing it to different values, such as "n=5" (note that the argument "break=5" will give you the same results). What do you notice? This argument can be used to specify the width of what are called bins. Be careful using this argument, because it can distort your histogram and therefore visualization of your data. There is no best value to use; experiment with it to see what best illustrates your data.

Finally, let us say we want to compare two histograms to see how different bin width values affect our histogram. We can divide our Plots window into two, where we see two plots in this one window:

```
> par(mfrow=c(2,1))
```

Now run this code:

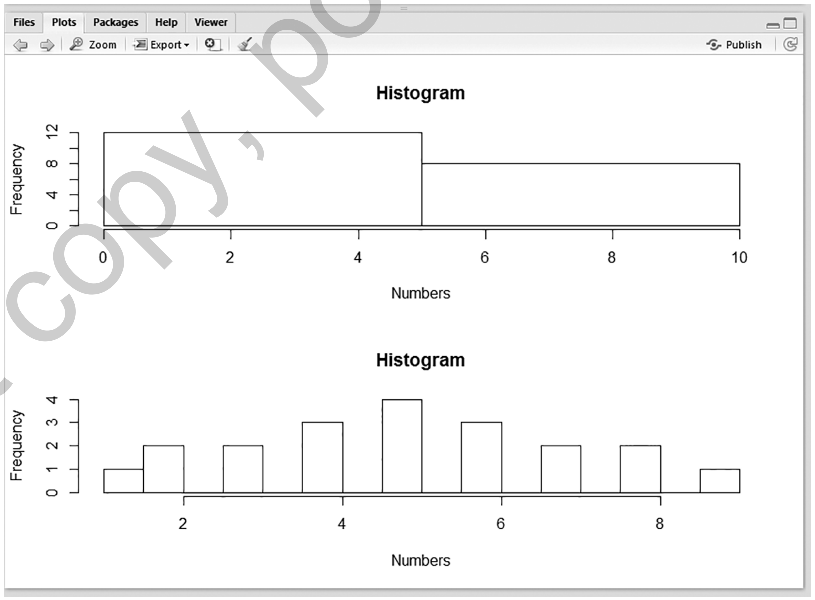
```
> hist(numbers$numbers, n=2, main="Histogram",xlab="Numbers")

> hist(numbers$numbers, n=15, main="Histogram",xlab="Numbers")
```

Now we have two histograms in one window to compare (see Figure 2.1)! To change our window back to its original view, type:

```
> par(mfrow=c(1,1))
```

Figure 2.1 Two Histograms With Different Bin Widths



We can use the par() function to view multiple plots in one window.

To become a better R user, it is important that you practice and experiment. Try using different functions and arguments to see how they influence your results. For example, can you change the colors of the bar graph using the arguments in the hist() function?