4.2: Using R to Summarize Data

One of R's strengths is its `Stats` package, which provides access to a rich body of tools for analyzing data. The package is part of R's base installation and is available whenever you use R without the need to use `library()` to make it available. Almost all of the statistical functions we will use in this textbook are included in the `Stats` package.

Bringing Your Data Into R

This section uses the M&M data in Table 1 of Chapter 3.1. You can download a copy of the data as a .csv spreadsheet using this link. Before we can summarize our data, we need to make it available to R. The code below uses the `read.csv` function to read in the data from the file `MandM.csv()` as a data frame. The text `"MandM.csv"` assumes the file is located in your working directory.

```
mm_data = read.csv("MandM.csv")
```

Finding the Central Tendency of Data Using R

To report the mean of a data set we use the function `mean(x)` where `x` is the object that holds our data, typically a vector or a single column from a data frame. An important argument to this, and to many other functions, is how to handle missing or `NA` values. The default is to keep them, which leads to an error when we try to calculate the mean. This is a reasonable default as it requires us to make note of the missing values and to set `na.rm = TRUE` if we wish to remove them from the calculation. As our vector of data is not missing any values, we do not need to include `na.rm = TRUE` here, but we do so to illustrate its importance.

```
mean(mm_data$net_weight, na.rm = TRUE)
```

```
[1] 48.9803
```

To report the median of a data set we use the function `median(x)` where `x` is the object that holds our data, typically a vector or a single column from a data frame.

```
median(mm_data$net_weight, na.rm = TRUE)
```

```
[1] 48.7345
```

Finding the Spread of Data Using R

To report the variance of a data set we use the function `var(x)` where `x` is the object that holds our data, typically a vector or a single column from a data frame.

```
var(mm_data$net_weight, na.rm = TRUE)
```

```
[1] 2.052068
```

To report the standard deviation we use the function `sd(x)` where `x` is the object that holds our data, typically a vector or a single column from a data frame.

```
sd(mm_data$net_weight, na.rm = TRUE)
```

```
[1] 1.432504
```

To report the range we have to be creative as R's `range()` function does not directly report the range. Instead, it returns the minimum as its first value and the maximum as its second value, which we can extract using the bracket operator and then use to compute the range.

```
range(mm_data$net_weight, na.rm = TRUE)[2] - range(mm_data$net_weight, na.rm = TRUE)
[1]
```

```
[1] 5.325
```

Another approach for calculating the range is to use R's `max()` and `min()` functions.

```
max(mm_data$net_weight) - min(mm_data$net_weight)
```

```
[1] 5.325
```

To report the interquartile range we use the function `IQR(x)` where `x` is the object that holds our data, typically a vector or a single column from a data frame. The function has nine different algorithms for calculating the *IQR*, identified using `type` as an argument. To obtain an *IQR* equivalent to that generated by R's `boxplot()` function, we use `type = 5` for an even number of values and `type = 7` for an odd number of values.

```
IQR(mm_data$net_weight, na.rm = TRUE, type = 5)
```

```
[1] 1.841
```

To find the median absolute deviation we use the function `mad(x)` where `x` is the object that holds our data, typically a vector or a single column from a data frame. The function includes a scaling constant, the default value for which does not match our description for calculating the *MAD*; the argument `constant = 1` gives a result that is consistent with our description of the *MAD*.

```
mad(mm_data$net_weight, na.rm = TRUE, constant = 1)
```

```
[1] 0.818
```
