2.2: Using R to Organize and Manipulate Data

The data in Table 2.2.1 should remind you of a data frame, a way of organizing data in R that we introduced in Chapter 1. Here we will learn how to create a data frame that holds the data in Table 2.2.1 and learn how we can make us of the data frame.

Creating a Data Frame

To create a data frame we begin by creating vectors for each of the variables. Note that `letters` is a constant in R that contains the 26 lower case letters of the Roman alphabet: here we are using just the first six letters for the bag ids.

```
bag_id = letters[1:6]
year = c(2006, 2006, 2000, 2000, 1994, 1994)
weight = c(1.74, 1.74, 0.80, 0.80, 10.0, 10.0)
type = c("peanut", "peanut", "plain", "plain", "plain", "plain")
number_yellow = c(2, 3, 1, 5, 56, 63)
percent_red = c(27.8, 4.35, 22.7, 20.8, 23.0, 21.9)
total = c(18, 23, 22, 24, 331, 333)
rank = c("sixth", "fourth", "fifth", "third", "second", "first")
```

To create the data frame, we use R's `data.frame()` function, passing to it the names of our vectors, each of which must be of the same length. There is an option within this function to treat variables whose values are character strings as factors—another name for a categorical variable—by using the argument `stringsAsFactors = TRUE`. As the default value for this argument depends on your version of R, it is useful to make your choice explicit by including it in your code, as we do here.

```
mm_data = data.frame(bag_id, year, weight, type, number_yellow, percent_red, total,
rank, stringsAsFactors = TRUE)
mm_data
```

```
bag_id year weight type number_yellow percent_red total rank
1 a 2006 1.74 peanut 2 27.80 18 sixth
2 b 2006 1.74 peanut 3 4.35 23 fourth
3 c 2000 0.80 plain 1 22.70 22 fifth
4 d 2000 0.80 plain 5 20.80 24 third
5 e 1994 10.00 plain 56 23.00 331 second
6 f 1994 10.00 plain 63 21.90 333 first
```

If we examine the structure of this data set using R's `str()` function, we see that bag_id, type, and rank are factors and year, weight, number_yellow, percent_red, and total arenumerical variables, assignments that are consistent with our earlier analysis of the data.

```
str(mm_data)
```

```
'data.frame': 6 obs. of 8 variables:
$ bag_id : Factor w/ 6 levels "a","b","c","d",..: 1 2 3 4 5 6
$ year : num 2006 2006 2000 2000 1994 ...
$ weight : num 1.74 1.74 0.8 0.8 10 10
$ type : Factor w/ 2 levels "peanut","plain": 1 1 2 2 2 2
$ number_yellow: num 2 3 1 5 56 63
$ percent_red : num 27.8 4.35 22.7 20.8 23 21.9
$ total : num 18 23 22 24 331 333
$ rank : Factor w/ 6 levels "fifth","first",..: 5 3 1 6 4 2
```

Finally, we can use the function `as.factor()` to have R treat a numerical variable as a categorical variable, as we do here for year. Why we might wish to do this is a topic we will return to in later chapters.

```
mm_year_as_factor = data.frame(bag_id, as.factor(year), percent_red, total)
str(mm_year_as_factor)
'data.frame': 6 obs. of 4 variables:
$ bag_id : Factor w/ 6 levels "a","b","c","d",..: 1 2 3 4 5 6
$ as.factor.year.: Factor w/ 3 levels "1994","2000",..: 3 3 2 2 1 1
$ percent_red : num 27.8 4.35 22.7 20.8 23 21.9
$ total : num 18 23 22 24 331 33
```

Creating a New Data Frame by Subsetting an Existing Data Frame

In Chapter 1.2 we learned how to retrieve individual rows or columns from a data frame and assign them to a new object. Here we learn how to use R's more flexible `subset()` function to accomplish the same thing. Here, for example, we retrieve only the data for plain M&Ms.

```
plain_mm = subset(mm_data, type == "plain")
plain_mm
bag_id year weight type number_yellow percent_red total rank
3 c 2000 0.8 plain 1 22.7 22 fifth
4 d 2000 0.8 plain 5 20.8 24 third
5 e 1994 10.0 plain 56 23.0 331 second
6 f 1994 10.0 plain 63 21.9 333 first
```

Note that `type == "plain"` uses a relational operator to choose only those rows in which the variable `type` has the value `plain`. Here is a list of relational operators:

Table 2.2.2. Relational Operators in R.

operator	usage	meaning
<	x < y	x is less than y
>	x > y	x is greater than y
<=	x <= y	x is less than or equal to y
>=	x >= y	x is greater than or equal to y
==	x == y	x is exactly equal to y
!=	x != y	x is not equal to y

We can string variables together using the logical & operator.

```
mm_plain10 = subset(mm_data, (weight == 10.0 & type == "plain"))
mm_plain10
bag_id year weight type number_yellow percent_red total rank
5 e 1994 10 plain 56 23.0 331 second
6 f 1994 10 plain 63 21.9 333 first
```

We also can narrow the number of variables returned using the `subset()` function's `select` argument. In this example we exclude samples collected before the year 2000 and return only the year, the number of yellow M&Ms, and the percentage of red M&Ms.

```
mm_20xx = subset(mm_data, year >= 2000, select = c(year, number_yellow, percent_red))
mm_20xx
year number_yellow percent_red
1 2006 2 27.80
2 2006 3 4.35
3 2000 1 22.70
4 2000 5 20.80
```