ST 551: Statistical Methods I
A Brief Introduction to R

1 Simple Operations

1.1 Simple Numerical Calculations

R can work as a simple calculator. Try the following examples (expected output shown in the line below, as it should appear on your screen). Note that preceding any line with the pound character "`#`" tells R to ignore that line; it is treated as a comment.

```
> 3 + 4
[1] 7

> (3 + 4) * 2
[1] 14

> 2 * 8
[1] 16

> 5^2
[1] 25

> 16/4
[1] 4

# Natural Logarithm
> log(2)
[1] 0.6931472

# Logarithm Base 2
> log(x=8, base=2)
[1] 3

# Exponential function: e^3
> exp(3)
[1] 20.08554
```

Try other more complicated examples to familiarize yourself with the arithmetic manipulations in R.

1.2 Assignment

If you want to save the result of a calculation or if you want to create an object, you use the assignment functions, `<-` or `=`. For example:

```
> rslt <- 3 + 4
> rslt
[1] 7
```

1.3 Create Vectors

Use the `c()` function to create vectors.

```
> cars.sold <- c(20, 18, 11, 6, 10)
> cars.sold
[1] 20 18 11  6 10

> sales.people <- c(6, 6, 3, 4, 2)
> sales.people
[1] 6 6 3 4 2
```

1.4 Combine Vectors into a Matrix

The `cbind()` function combines vectors into a matrix, by columns. The `rbind()` function combines vectors into a matrix, by rows.

```
> cars.data <- cbind(cars.sold, sales.people)
> cars.data
     cars.sold sales.people
[1,]        20            6
[2,]        18            6
[3,]        11            3
[4,]         6            4
[5,]        10            2
```

1.5 Test Equality

To test equality, you use double equal signs `==`. Try this example:

```
> rslt <- 3 + 4

# Testing equality: is rslt equal to 7?
> rslt == 7
[1] TRUE

# Testing equality: is rslt equal to 8?
> rslt == 8
[1] FALSE

# Assignment: Set rslt equal to 8
> rslt = 8
> rslt
[1] 8

# Testing equality: is rslt equal to 8?
> rslt == 8
[1] TRUE
```

1.6 Vector and Matrix Manipulation

When you add or multiply two vectors or matrices, the result is the component-wise sum or product of the arguments. To perform the usual matrix multiplication, use the `%*%` operator. You can also calculate the mean, variance, sum, and standard deviation of a vector or matrix. Try these examples:

```
> cars.sold <- c(20, 18, 11, 6, 10)
> sales.people <- c(6, 6, 3, 2, 4)
```

As an example, if we add 'cars' and 'sales' we get a vector of the sums of the individual components.

```
> cars.sold + sales.people
[1] 26 24 14  8 14
```

To calculate the average number of cars sold per salesperson, we can divide 'cars' by 'sales'.

```
> cars.sold/sales.people
[1] 3.333333 3.000000 3.666667 3.000000 2.500000
```

If we square 'cars.sold' we get a vector of the squared individual components.

```
> cars.sold^2
[1] 400 324 121  36 100
```

The function `length()` returns the number of elements in a vector.

```
> length(cars.sold)
[1] 5
> length(c(cars.sold, sales.people))
[1] 10
```

1.7 Mean, Standard Deviation, and Variance

The functions `mean()`, `sd()`, and `var()` compute the sample mean, standard deviation, and variance.

```
> mean(cars.sold)
[1] 13
> sd(cars.sold)
[1] 5.830952
> var(cars.sold)
[1] 34

> mean(sales.people)
[1] 4.2
> sd(sales.people)
[1] 1.788854
> var(sales.people)
[1] 3.2
```

1.8 Selecting Elements of a Vector or Matrix

Elements of a matrix or vector may be selected using square brackets " `[ ]`", as illustrated below. For a vector, the contents of the brackets should be a position or vector of positions that you would like to select. For a matrix, the contents of the brackets should be [row numbers, column numbers]. If you leave either row numbers or column numbers blank, all rows (respectively columns) are selected.

```
> cars.data
     cars.sold sales.people
[1,]        20            6
[2,]        18            6
[3,]        11            3
[4,]         6            2
[5,]        10            4

# Third element of the cars.sold vector
> cars.sold[3]
[1] 11

# Odd positions of the cars.sold vector
> cars.sold[c(1, 3, 5)]
[1] 20 11 10

# Second row of the cars.data matrix
> cars.data[2,]
   cars.sold sales.people
          18            6

# Third and fourth row, second column of the cars.data matrix
> cars.data[c(3,4), 2]
[1] 3 2
```

2 Reading in Data

You can read data into R from a text file using the `read.table` command (easiest for ordinary or tab-delimited text files) or the `read.csv` command (for .csv files). If you have data in an Excel file, you should save it as a tab delimited text file (.txt) or a comma delimited file (.csv), and then read it into R.

The `read.table` and `read.csv` commands require that you specify the file location of the file that you want to read in, for instance:

```
> read.table("/Users/SCE/Documents/ST 551 2023/Datasets/MathSAT.csv", header=T)
```

Or

```
> read.csv("/Users/SCE/Documents/ST 551 2023/Datasets/MathSAT.csv")
```

The `header=T` portion of this command (default for `read.csv`) tells R that the first line of the data file contains the variable names. The location information for a file may be found by right clicking on the file that you would like to use and selecting "Get Info" (Mac) or "Properties" (Windows) from the menu. In the dialog box that appears, there is a line labeled "Where" (Mac) or "Location" (Windows) that displays the location information for the file.

To make it easy to read in files (and not have to type the entire file location each time) you can set a working directory. Use the location found above to set your working directory (**note that on Windows machines you have to change the direction of the slashes in the file location!**).

```
> setwd("/Users/SCE/Documents/ST 551 2023/Datasets")
> mathSAT <- read.csv("MathSAT.csv", header=T)
```

It is a good idea to explore the dataset you read in without printing the entire thing. The first thing you might want to do is to find out the dimensions of the dataset, using the `dim()` function:

```
> dim(mathSAT)
[1] 61  4
```

You can also look at the header of the dataset, which is the first six rows, with column names, to get an idea of the data structure. The `head()` function prints the dataset header:

```
> head(mathSAT)
  Score Total Male Female
1   800 11959 8072   3887
2   790  3588 2327   1261
3   780  3770 2377   1393
4   770  9663 6340   3323
5   760  7016 4413   2603
6   750 10313 6721   3592
```

Or perhaps we just want to see the first 3 rows:

```
> mathSAT[1:3,]
   Score Total Male Female
1    800 11959 8072   3887
2    790  3588 2327   1261
3    780  3770 2377   1393
```

Or the first column:

```
> mathSAT[,1]
 [1] 800 790 780 770 760 750 740 730 720 710 700 690 680 670 660 650 640 630 620 610
[21] 600 590 580 570 560 550 540 530 520 510 500 490 480 470 460 450 440 430 420 410
[41] 400 390 380 370 360 350 340 330 320 310 300 290 280 270 260 250 240 230 220 210
[61] 200
```

It is also frequently useful to just see the names of the columns (variables) in a dataset, which can be done using the `names()` function.

```
> names(mathSAT)
[1] "Score"  "Total"  "Male"   "Female"
```

3 Where to Get Help

- Type "help" in an R session. Example: `> help(matrix)`
- CRAN website. The CRAN website contains help files and general information
- Search the web. If you are having trouble doing something in R, try a general search in Google. Example Google search: "R help creating matrix"
- Ask Instructor, TA, fellow student.