Chapter 1

1.1

Store the values *-20, -15, -5, 8, 12, 9, 2, 23, 19* in the R variable *x* and verify the sum is 33.

```
x <- c(-20, -15, -5, 8, 12, 9, 2, 23, 19)

sum(x)
```

```
[1] 33
```

1.2

For the data in E1, verify the average is 3.67.

```
mean(x)
```

```
[1] 3.666667
```

1.3

What commands can be used to compute an average without using the R command *mean*?

```
n <- length(x)

sum(x) / n
```

```
[1] 3.666667
```

1.4

Sum the positive values:

```
sum( x[x >= 0] )
```

```
[1] 73
```

1.5

Use *which* command to get the average of the values ignoring the largest.

```
mean( which( x != max(x) ) )
```

```
[1] 4.625
```

1.6

Speculate about the values corresponding to *x[abs(x) >= 8 & x < 8]*.

[T, T, F, T, T, F, T, T] & [T, T, T, F, F, F, F, F] = [T, T, F, F, F, F, F, F]

Verify.

```
x[ abs(x) >= 8 & x < 8]
```

```
[1] -20 -15
```

1.7

Your recorded time to commute for 10 days is:

23, 18, 29, 22, 24, 27, 28, 19, 28, 23

```
commute <- c(23, 18, 29, 22, 24, 27, 28, 19, 28, 23)

mean(commute)
```

```
[1] 24.1
```

```
min(commute)
```

```
[1] 18
```

```
max(commute)
```

```
[1] 29
```

1.8

Verify:

```
y <- c(2, 4, 8)
z <- c(1, 5, 2)
2*y
```

```
[1]  4  8 16
```

returns the values *4, 8, 16*.

1.9

Let *x = c(1, 8, 2, 6, 3, 8, 5, 5, 5, 5)*.

```
x <- c(1, 8, 2, 6, 3, 8, 5, 5, 5, 5)

n <- length(x)

stopifnot( (sum(x) / n) == mean(x))
```

1.10

For the values in *E9*, use R to subtract the average from each value, then sum the results.

```
xbar <- mean(x)
sum(x - xbar)
```

```
[1] 1.776357e-15
```

1.11

Imagine a matrix having 100 rows and 2 columns. Further imagine that some of the values in the first column are *NA*.

Use *is.na* to remove missing.

```
values <- matrix(nrow = 100, ncol = 2)
rval <- runif(100)
values[, 1] <- ifelse(rval <= .25, NA, rval)


values[!is.na(values[,1]),]
```

```
           [,1] [,2]
 [1,] 0.6228954   NA
 [2,] 0.5341502   NA
 [3,] 0.4623623   NA
 [4,] 0.8545606   NA
 [5,] 0.6585149   NA
 [6,] 0.5442241   NA
 [7,] 0.8719583   NA
 [8,] 0.8880862   NA
 [9,] 0.5737093   NA
[10,] 0.5039815   NA
[11,] 0.4392406   NA
[12,] 0.3704456   NA
[13,] 0.7970290   NA
[14,] 0.4729621   NA
[15,] 0.9681068   NA
[16,] 0.9346614   NA
[17,] 0.5553968   NA
[18,] 0.2725059   NA
[19,] 0.4424126   NA
[20,] 0.8597343   NA
[21,] 0.8037248   NA
[22,] 0.7444269   NA
[23,] 0.8337538   NA
[24,] 0.4630863   NA
[25,] 0.7617848   NA
[26,] 0.3676779   NA
[27,] 0.4698599   NA
[28,] 0.6590073   NA
[29,] 0.8487626   NA
[30,] 0.7174182   NA
[31,] 0.4067656   NA
[32,] 0.2779783   NA
[33,] 0.8866678   NA
[34,] 0.9568841   NA
[35,] 0.8504638   NA
[36,] 0.4169639   NA
[37,] 0.8445439   NA
[38,] 0.9058939   NA
[39,] 0.3574626   NA
[40,] 0.3975024   NA
[41,] 0.4256570   NA
[42,] 0.9609642   NA
[43,] 0.5355832   NA
[44,] 0.9313943   NA
```

```
[45,] 0.4019153   NA
[46,] 0.3375918   NA
[47,] 0.4349616   NA
[48,] 0.3291224   NA
[49,] 0.4019925   NA
[50,] 0.9726433   NA
[51,] 0.7913068   NA
[52,] 0.2737922   NA
[53,] 0.5553981   NA
[54,] 0.5803151   NA
[55,] 0.9376537   NA
[56,] 0.2926755   NA
[57,] 0.6791216   NA
[58,] 0.6771001   NA
[59,] 0.8755614   NA
[60,] 0.9552153   NA
[61,] 0.3892541   NA
[62,] 0.7801718   NA
[63,] 0.3218550   NA
[64,] 0.9069957   NA
[65,] 0.6465472   NA
[66,] 0.9892382   NA
[67,] 0.7955869   NA
[68,] 0.4047784   NA
[69,] 0.8125634   NA
[70,] 0.9696048   NA
[71,] 0.7583709   NA
[72,] 0.5211029   NA
[73,] 0.4433115   NA
[74,] 0.8500371   NA
[75,] 0.3028957   NA
[76,] 0.8946830   NA
[77,] 0.2542547   NA
[78,] 0.9660887   NA
```

1.12

R has a built-in data set ChickWeight. Verify the mean.

```
mean(ChickWeight[, 1])
```

```
[1] 121.8183
```

```
mean(ChickWeight[, 3])
```

```
Warning in mean.default(ChickWeight[, 3]): argument is not numeric or
logical: returning NA
```

```
[1] NA
```

```
mean(as.numeric(ChickWeight[, 3]))
```

```
[1] 26.25952
```

1.13

Create a matrix with two rows and five columns with some of the entries stored as NA.

```
rval <- runif(10, 0, 1)
values <- matrix(ifelse(rval <= .25, NA, rval), nrow = 2)
```

1.14

Use R to compute the average weight among chicks that were fed horsebean.

```
mean(chickwts[chickwts$feed == "horsebean",]$weight)
```

```
[1] 160.2
```

1.15

Let *x = 1, 8, 2, 6, 3, 8, 5, 5, 5, 5*

Sum values not equal to 2 or 3, two ways.

```
x <- c(1, 8, 2, 6, 3, 8, 5, 5, 5, 5)

sum( x[!x %in% c(2, 3)] )
```

```
[1] 43
```

```
sum( x[x != 2 & x != 3] )
```

```
[1] 43
```

1.16

For the values used in the previous exercise, use two different R commands to sum all the values != 5.

```
sum( x[ x != 5] )
```

```
[1] 28
```

```
sum( x[ !x %in% c(5) ])
```

```
[1] 28
```

1.17

Use a single command to change all values equal to 8 to 7.

```
x[x == 8] <- 7
x
```

```
 [1] 1 7 2 6 3 7 5 5 5 5
```

1.18

Create a matrix with four rows and two columns with the values: *1, 2, 3, 4* in the first column, and *5, 6, 7, 8* in the second column.

```
matrix(seq(1, 8, by = 1), nrow = 4, ncol = 2)
```

```
     [,1] [,2]
[1,]    1    5
[2,]    2    6
[3,]    3    7
[4,]    4    8
```

1.19

Create a matrix with four columns and two rows with the values *1, 2, 3, 4* in the first row and *11, 12, 13, 14* in the second row.

```
matrix(c(seq(1,4), seq(11, 14)), ncol = 4, byrow = T)
```

```
     [,1] [,2] [,3] [,4]
[1,]    1    2    3    4
[2,]   11   12   13   14
```