

Chapter 2

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 $x[abs(x) \geq 8 \ \& \ x < 8]$.

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

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,]	0.6375472	NA
[2,]	0.9547545	NA
[3,]	0.5698609	NA
[4,]	0.7644303	NA
[5,]	0.7663828	NA
[6,]	0.5145734	NA
[7,]	0.7014108	NA
[8,]	0.3562794	NA
[9,]	0.3691317	NA
[10,]	0.9186638	NA
[11,]	0.8658385	NA
[12,]	0.5193081	NA
[13,]	0.5783889	NA
[14,]	0.3177765	NA
[15,]	0.5882129	NA
[16,]	0.5638461	NA
[17,]	0.5255802	NA
[18,]	0.9419415	NA
[19,]	0.8572786	NA
[20,]	0.6688114	NA
[21,]	0.3105101	NA
[22,]	0.6627235	NA
[23,]	0.5038343	NA
[24,]	0.8194848	NA
[25,]	0.7194572	NA
[26,]	0.4447876	NA
[27,]	0.8519157	NA
[28,]	0.3061646	NA
[29,]	0.9618567	NA
[30,]	0.8421975	NA
[31,]	0.8310775	NA
[32,]	0.7369533	NA
[33,]	0.7836635	NA
[34,]	0.3647067	NA
[35,]	0.6281713	NA
[36,]	0.9090098	NA
[37,]	0.6307505	NA
[38,]	0.8655927	NA
[39,]	0.3830484	NA
[40,]	0.5043155	NA
[41,]	0.8705381	NA
[42,]	0.5155331	NA
[43,]	0.5706433	NA
[44,]	0.2911739	NA

```
[45,] 0.2668346 NA
[46,] 0.9853353 NA
[47,] 0.9263314 NA
[48,] 0.4540517 NA
[49,] 0.4101492 NA
[50,] 0.9133918 NA
[51,] 0.5479068 NA
[52,] 0.5227746 NA
[53,] 0.3382899 NA
[54,] 0.3398138 NA
[55,] 0.6938258 NA
[56,] 0.4265168 NA
[57,] 0.7343853 NA
[58,] 0.7605570 NA
[59,] 0.9696664 NA
[60,] 0.4310507 NA
[61,] 0.6356616 NA
[62,] 0.3173934 NA
[63,] 0.6525155 NA
[64,] 0.4644148 NA
[65,] 0.2827141 NA
[66,] 0.8749185 NA
[67,] 0.4304272 NA
[68,] 0.7443630 NA
[69,] 0.9329234 NA
[70,] 0.9908767 NA
[71,] 0.9745048 NA
[72,] 0.7875264 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 $\neq 5$.

```
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
```