Contents

1 Exercise 1.1

```
205 + 106
2 * 6 + 9 / 3
2 * (6 + 9) / 3
3^4 - 4^3
28 %% 6
```

```
## [1] 311
## [1] 15
## [1] 10
## [1] 17
## [1] 4
```

There is a general order of operations, which follows regular math.

2 Exercises 1.2

```
red_apples <- 4
green_apples <- 5
basket_apples <- red_apples + green_apples
18 * basket_apples
```

```
## [1] 162
```

While you could do (c) with `basket_apples <- 4 + 5`, this makes the code less flexible when the variables `red_apples` and `green_apples` are representing a specific quantity, i.e. the number of red apples and green apples in the basket.

For example, we can easily change the values of `red_apples` and `green_apples` and get the total sum, changing the values at *only one place*.

3 Exercise 1.3

```
my_course <- "MH1234"
my_score <- 76
my_grade <- "A-"
my_core <- TRUE

class(my_course)
```

```
## [1] "character"
```

```
class(my_score)
```

```
## [1] "numeric"
```

```
class(my_grade)
```

```
## [1] "character"
```

```
class(my_core)
```

```
## [1] "logical"
```

4 Exercise 1.4

```
seq(10)
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```
seq(0, 10, length = 10)
```

```
##  [1]  0.000000  1.111111  2.222222  3.333333  4.444444  5.555556  6.666667
##  [8]  7.777778  8.888889 10.000000
```

```
seq(0, 10, by = 1)
```

```
##  [1]  0  1  2  3  4  5  6  7  8  9 10
```

```
rep(2, 3)
```

```
## [1] 2 2 2
```

```
rep(-1, 4)
```

```
## [1] -1 -1 -1 -1
```

```r
rep(1:3, 2)
```

```
## [1] 1 2 3 1 2 3
```

```r
rep("A", 4)
```

```
## [1] "A" "A" "A" "A"
```

```r
c(rep("A", 3), rep("B", 3))
```

```
## [1] "A" "A" "A" "B" "B" "B"
```

```r
rep(c("A", "B"), 3)
```

```
## [1] "A" "B" "A" "B" "A" "B"
```

```r
vector1 <- seq(3, 9, 2)
vector2 <- rep(2:4, 2:4)
vector3 <- c(rep(T, 2), rep(F, 3))
vector4 <- rep(c("High", "Medium", "Low"), 2)
```

5 Exercise 1.5

```r
poker_winnings <- c(140, -50, 20, -120, 240)
roulette_winnings <- c(-24, -50, 100, -350, 10)
days <- c("Mon", "Tue", "Wed", "Thur", "Fri")

poker_winnings
```

```
## [1]  140  -50   20 -120  240
```

```r
names(poker_winnings) <- days
poker_winnings
```

```
##  Mon  Tue  Wed Thur  Fri
##  140  -50   20 -120  240
```

```r
names(roulette_winnings) <- days

poker_winnings[poker_winnings > 0]
```

```
## Mon Wed Fri
## 140  20 240
```

```r
daily_winnings <- poker_winnings + roulette_winnings
daily_winnings[daily_winnings > 0]
```

```
## Mon Wed Fri
## 116 120 250
```

```r
sum(poker_winnings)
```

```
## [1] 230
```

```r
sum(roulette_winnings)
```

```
## [1] -314
```

(j): Based on the summations, we can clearly see poker had a net gain, while roulette had a net loss. Clearly, don't play roulette.

```r
sum(daily_winnings)
```

```
## [1] -84
```

(k): There was a net loss of $84 in the week.

6 Exercise 1.6

```r
matrix_4x2 <- matrix(1:8, ncol = 2)
matrix_4x2
```

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

```
matrix_2x4 <- matrix(c(1:4, 11:14), nrow = 2, byrow=T)
matrix_2x4
```

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

```
matrix_4x2 %*% matrix_2x4
```

```
##      [,1] [,2] [,3] [,4]
## [1,]   56   62   68   74
## [2,]   68   76   84   92
## [3,]   80   90  100  110
## [4,]   92  104  116  128
```

```
matrix_2x4 %*% matrix_4x2
```

```
##      [,1] [,2]
## [1,]   30   70
## [2,]  130  330
```

```
matrix_4x2
```

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

```
matrix_2x2 <- matrix_4x2[1:2,]
matrix_2x2
```

```
##      [,1] [,2]
## [1,]    1    5
## [2,]    2    6
```

Note that matrix multiplication is done using `%*%` in R. The `solve()` function attempts to find the inverse of a square matrix. To verify it is the inverse, we can perform matrix multiplication.

```r
solve(matrix_2x2)
```

```
##      [,1]  [,2]
## [1,] -1.5  1.25
## [2,]  0.5 -0.25
```

```r
matrix_2x2 %*% solve(matrix_2x2)
```

```
##      [,1] [,2]
## [1,]    1    0
## [2,]    0    1
```

```r
diag(3)
```

```
##      [,1] [,2] [,3]
## [1,]    1    0    0
## [2,]    0    1    0
## [3,]    0    0    1
```

```r
diag(c(2, 3, 5))
```

```
##      [,1] [,2] [,3]
## [1,]    2    0    0
## [2,]    0    3    0
## [3,]    0    0    5
```

7 Exercise 1.7

```r
apple <- c(350, 150)
orange <- c(400, 230)
watermelon <- c(180, 90)

fruit_matrix <- matrix(c(apple, orange, watermelon), nrow = 3, byrow = T)

fruit_matrix
```

```
##      [,1] [,2]
## [1,]  350  150
## [2,]  400  230
## [3,]  180   90
```

```r
rownames(fruit_matrix) <- c("Apple", "Orange", "Watermelon")
colnames(fruit_matrix) <- c("Weekday", "Weekend")
fruit_matrix
```

```
##            Weekday Weekend
## Apple          350     150
## Orange         400     230
## Watermelon     180      90
```

```r
sum_fruit_matrix <- cbind(fruit_matrix, Total=rowSums(fruit_matrix))
sum_fruit_matrix
```

```
##            Weekday Weekend Total
## Apple          350     150   500
## Orange         400     230   630
## Watermelon     180      90   270
```

Above, I add the weekday and weekend totals into a new matrix just so it is easier to run repeatedly.

```r
sum_fruit_matrix[,"Total"] * c(1, 1.5, 4)
```

```
##      Apple     Orange Watermelon
##        500        945       1080
```

```r
sum(sum_fruit_matrix[,"Total"] * c(1, 1.5, 4))
```

```
## [1] 2525
```

```r
sum_fruit_matrix[,"Total"] %*% c(1, 1.5, 4)
```

```
##      [,1]
## [1,] 2525
```

```r
c(1, 1.5, 4) %*% sum_fruit_matrix[,"Total"]
```

```
##      [,1]
## [1,] 2525
```

The %*% operator here is acting on two *vectors*, so the inner product of the two vectors is obtained instead.

8 Getting help on matrix multiplication

As a side note, to get help on matrix multiplication (or any other operator), you have to surround it with backticks (“‘).

```
?`%*%`
help(`%*%`)
```