

# Problem Set 1 Solutions

Introduction to R | University of Oxford Sociology

Casey Breen

## Problem Set 1

Complete the following questions in R within a Quarto document.

### Exercise 1: Assignment, Arithmetic, Logical Expressions

#### 1a

Assign `x` and `y` to take values 3 and 4.

```
# Assign x and y to take values 3 and 4
x <- 3
y <- 4
```

#### 1b

Create a new variable `z` as the product of `x` and `y`.

```
# Create a new variable z as the product of variables x and y
z <- x * y
```

#### 1c

Calculate the square of 3 and assign it to a variable called `three_squared`.

```
# Write code to calculate the square of 3 and assign it to a variable three_squared
three_squared <- 3^2
```

### 1d

Write a logical expression to check if `three_squared` is greater than 10. Discuss scenarios where such a filter could be useful.

```
# Write a logical expression to check if `three_squared` is greater than 10
three_squared > 10
```

```
[1] FALSE
```

### 1e

Write a logical expression to test whether `three_squared` is not greater than 10. Use the `negate` symbol.

```
# Write a logical expression to check if `three_squared` is not greater than 10
!(three_squared <= 10)
```

```
[1] FALSE
```

## Exercise 2: Sequencing

### 2a

Generate vectors containing the numbers 100 to 105 using three different methods (`c()`, `seq()`, `:`). Discuss the convenience of each method.

```
# Generate a vector using c() method
vector_c <- c(100, 101, 102, 103, 104, 105)

# Generate a vector using seq() method
vector_seq <- seq(100, 105, by = 1)

# Generate a vector using : operator
vector_colon <- c(100:105)
```

**Answer:** The first method, generating a vector using `c()` is convenient when you are only including a few elements in your sequence or there is no clear pattern. The second method, using `seq`, is convenient when the numbers follow a pattern but not necessarily just increment by. The third method is most convenient to generate numbers in a sequence increasing or decreasing by exactly 1.

## 2b

Generate a sequence of all even numbers between 0 and 100.

```
# Generate a sequence of all even numbers between 0 and 100
even_seq <- seq(0, 100, by = 2)
even_seq
```

```
[1] 0 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36
[20] 38 40 42 44 46 48 50 52 54 56 58 60 62 64 66 68 70 72 74
[39] 76 78 80 82 84 86 88 90 92 94 96 98 100
```

```
# Create a descending sequence of numbers from 100 to 1
desc_seq <- seq(100, 1, by = -1)
desc_seq
```

```
[1] 100 99 98 97 96 95 94 93 92 91 90 89 88 87 86 85 84 83
[19] 82 81 80 79 78 77 76 75 74 73 72 71 70 69 68 67 66 65
[37] 64 63 62 61 60 59 58 57 56 55 54 53 52 51 50 49 48 47
[55] 46 45 44 43 42 41 40 39 38 37 36 35 34 33 32 31 30 29
[73] 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11
[91] 10 9 8 7 6 5 4 3 2 1
```

## 2c

Create a descending sequence from 100 to 1 and assign it to a variable.

```
# Create a descending sequence of numbers from 100 to 1
desc_seq_decrease <- seq(100, 1, by = -1)
desc_seq_decrease
```

```
[1] 100 99 98 97 96 95 94 93 92 91 90 89 88 87 86 85 84 83
[19] 82 81 80 79 78 77 76 75 74 73 72 71 70 69 68 67 66 65
[37] 64 63 62 61 60 59 58 57 56 55 54 53 52 51 50 49 48 47
[55] 46 45 44 43 42 41 40 39 38 37 36 35 34 33 32 31 30 29
[73] 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11
[91] 10 9 8 7 6 5 4 3 2 1
```

### Exercise 3: Data Generation and Basic Statistical Analysis

#### 3a

Generate a sample of 100 observations from a normal distribution with a mean of 10 and a standard deviation of 2. Use the `rnorm()` function.

```
sim_data <- rnorm(n = 100,  
                  mean = 10,  
                  sd = 2)
```

#### 3b

Calculate the mean of this generated sample. Relate this to the population mean. Note the population mean is 10.

```
mean(sim_data)
```

```
[1] 10.02163
```

*Answer: This is relatively close to, but not exactly, the population mean. This is because we are taking a random sample from the distribution.*

#### 3c

Calculate the difference between the sample mean and the population mean. Discuss the reason for the discrepancy.

```
diff_100 <- mean(sim_data) - 10
```

#### 3d

Repeat steps 1–3 with a sample size of 10,000. Discuss whether the difference between the sample and population mean decreased and its implications.

```
## simulate 10000 draws  
sim_data_10000 <- rnorm(n = 10000,  
                        mean = 10,  
                        sd = 2)
```

```
mean_sim_data_10000 <- mean(sim_data_10000)
```

```
##
```

```
mean_sim_data_10000 - 10
```

```
[1] -0.004936195
```

***Answer:** The difference between the sample mean and the population mean decreased as we increased our sample size. This is because as our sample size increases, the mean of the sample tends towards the population mean. This is a fundamental concept in statistics: the law of large numbers.*