# Lab Two – Basic Tasks in R

- Use `R` (R Core Team 2025) to complete the tasks below. Make sure to start your solutions in on a new line that starts with "**Solution**:".
- Make sure to use the Quarto Cheatsheet. This will make completing and writing up the lab *much* easier.

## 1 Libraries and Packages

### 1.1 Part a

Install the `esquisse` package for `R`. Report the code below, ensuring to set the `eval` option to `false`, using "#| eval: false", so that we don't install the package every time you compile your Quarto document.

### 1.2 Part b

Load the `esquisse` package for `R`. Report the code below. It does not matter whether you set "#| eval: false" because loading the package does not add a lot of time. Note you will not evaluate any code that uses the `esquisse` package, so there is no need to load it as you compile your Quarto document.

### 1.3 Part c

How can you ask for help about the `esquisse` package for `R`? Report the code below, ensuring to set the `eval` option to `false`, using "#| eval: false", so that the help document isn't loaded every time you compile your Quarto document.

### 1.4 Part d

Is a demo or vignette available for the `esquisse` package for `R`? Report the code below, ensuring to set the `eval` option to `false`, using "#| eval: false", so that the you don't get errors when you compile your Quarto document.

### 1.5 Part e

Add the BibTeX citation for `esquisse` package for `R` to your `.bib` file and cite it in a sentence describing what the package does below. Note you can reference a citation named `esquisse` using "[@esquisse]".

### 1.6 Part f

Run `esquisser()` in your console – not in a chunk of `R` code in your Quarto document.

i. Select `palmerpenguins` from the "Select an environment in which to search:" dropdown. This should automatically select `penguins` in the "Select a data.frame:" dropdown. Click "Import Data".
ii. Drag `body_mass_g` to the X box and `species` to the fill box.
iii. Describe what you see in words. What can you conclude about Adelie, Chinstrap, and Gentoo penguins based on the resulting plot?

#### 1.6.1 Solution

According to the histogram, there are the most of the Adelle species of penguin in total. In terms of body mass in grams, both Adelle and Chinstrap penguins have a similar range of body mass from around 2000-5000 with medians around 3500+. In contrast, the Gentoo species of penguin has a greater range of body mass, from around 4000-6000.

# 2 Objects and Vectors

Create the following vectors in `R`. In some cases, you may be able to use `seq()` or `rep()` while in others you cannot. Use these functions when possible, otherwise manually create the vector and explain why that was necessary.

**Some Snowday Notes**

There are three ways we will create a vector. Most simply, we can create one from scratch. For example, I create a vector of odds, and evens less than 10 below.

```
(odds <- c(1, 3, 5, 7, 9))
```

```
[1] 1 3 5 7 9
```

```
(evens <- c(2, 4, 6, 8))
```

```
[1] 2 4 6 8
```

There are also built-in functions like `seq(...)` for doing this:

```
(odds <- seq(from=1, to=9, by=2))
```

```
[1] 1 3 5 7 9
```

```
(evens <- seq(from=2, to=8, by=2))
```

```
[1] 2 4 6 8
```

Either approach is easy enough with a small number of elements, but what if I wanted odds less than 100? 1000? 1 million? The `rep(...)` and `seq()` approaches would be far more efficient.

We can also create repeating sequences by hand

```
(repeating.seq1 <- c(1, 2, 3, 1, 2, 3, 1, 2, 3))
```

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

```
(repeating.seq2 <- c(1, 1, 1, 2, 2, 2, 3, 3, 3))
```

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

or using a built-in function `rep(...)`

```
(repeating.seq1 <- rep(c(1,2,3), times=3))
```

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

```
(repeating.seq2 <- rep(c(1,2,3), each=3))
```

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

There are some vectors for which we can't use a `seq()` or `rep()`. For example, consider the prime numbers less than 10. The primes are not sequential (e.g., jump by a fixed amount), nor are they repeating.

```
primes <- c(2, 3, 5, 7)
```

**Note**: There is a `primes` package for `R` (Keyes and Egeler 2025) that contains a `generate_primes(min, max)` function for generating a vector of primes from `min` to `max`.

## 2.1 Part a

The Fibonacci Sequence is a recursive formula:

$$F_n = F_{n-1} + F_{n-2}$$

where $F_0 = 0$ and $F_1 = 1$.

Create a vector of the first 10 Fibonacci numbers.

**Solution**

```
(fibonacci <- c(0, 1, 1, 2, 3, 5, 8, 13, 21, 34))
```

```
 [1]  0  1  1  2  3  5  8 13 21 34
```

I used c() to form this vector manually, because there is no regular pattern that rep() or seq() could use to easily form the vector.

## 2.2 Part b

Triangular Numbers are the cumulative sums of natural numbers:

$$F_n = \frac{n(n+1)}{2}.$$

Create a vector of the first 10 Triangular Numbers.

**Solution**

```
n <- 1:10
(triangular <- n * (n + 1) / 2)
```

```
 [1]  1  3  6 10 15 21 28 36 45 55
```

Here, I made $n = 1 : 10$, along with the equation $n * (n + 1)/2$ to easily generate the first 10 terms that follow the given pattern of triangular numbers.

```
(triangular <- c(1, 3, 6, 10, 15, 21, 28, 36, 45, 55))
```

```
 [1]  1  3  6 10 15 21 28 36 45 55
```

Alternatively, you could use c() to form this vector manually (like above), but this becomes impractical if we were trying to generate a larger vector.

## 2.3 Part c

Suppose I were designing a repeated meausres experiment with three treatment conditions. Each of $n = 10$ participants (with `ID` 1 to 10) will receive *all* experimental conditions, call them "Control", "Treatment A", and "Treatment B".

Consider setting up data entry for this experiment.

i. Create a vector containing each `ID` repeated three times, once for each treatment.
ii. Create a vector containing each `Treatment` repeated for each participant ID.

**Solution**

```
(id.repeating.seq <- rep((1:10), times=3))
```

```
 [1]  1  2  3  4  5  6  7  8  9 10  1  2  3  4  5  6  7  8  9 10  1  2  3  4  5
[26]  6  7  8  9 10
```

```
(treatment.repeating.seq <- rep(c("Control", "Treatment A", "Treatment B"), each=10))
```

```
 [1] "Control"     "Control"     "Control"     "Control"     "Control"
 [6] "Control"     "Control"     "Control"     "Control"     "Control"
[11] "Treatment A" "Treatment A" "Treatment A" "Treatment A" "Treatment A"
[16] "Treatment A" "Treatment A" "Treatment A" "Treatment A" "Treatment A"
[21] "Treatment B" "Treatment B" "Treatment B" "Treatment B" "Treatment B"
[26] "Treatment B" "Treatment B" "Treatment B" "Treatment B" "Treatment B"
```

For part i it is sufficient to use repeating.seq from 1 to 10 to show how each ID received 3 treatments. Part ii, then, assigns the 3 treatments to the corresponding participants using each.seq, but this function needs to do each 10 times to match the 30 IDs.

## 2.4 Part d

Create a vector containing the `character` "MATH" and the `numeric` 240. What is the resulting class? Explain why in a sentence.

**Solution**

```
(course.name <- c("MATH", 240))
```

```
[1] "MATH" "240"
```

The result shows both "MATH" and 240 as characters, since this is the rightmost class of objects. This is because vectors are atomic and will only report one class type. Characters are the only class type that could kind of fit the data inputted here, so R will return both "MATH" and 240 as characters.

### 2.5 Citations

I used R for the entirety of this assignment (R Core Team 2025). The primes (Keyes and Egeler 2025) package was used for specific vector functions with prime numbers and the esquisse package (Meyer and Perrier 2025) was used to create and analyze the histogram.

### 2.6 Discussion

I used different strategies of creating vectors to form the vectors in parts a through d. At times, vectors needed to be created manually, while others could be made more efficiently using rep() functions or further object assignments. Questions such as part c where there is a lot of repetitive data cater much better to the rep() function, and questions such as part d show the necessity of class type agreement in a vector. The vector in part d gave back two characters, rather than a character and a number, because vectors are atomic and return the class type furtherst right in the progression. The vectors in parts a and b, were short and followed less regular patterns, so in this case, it was easiest to create them manually, although in part b a short cut could be used to make this code more effective especially if we had wanted to make a vector with more entries. Then, as for the initial histogram question, I used esquisser() along with the palmerpenguins package to observe the distribution of 3 different species of penguin's body masses and explained what I saw above. Each part here clearly shows some different basic functions of R.

## References


Keyes, Os, and Paul Egeler. 2025. *Primes: Fast Functions for Prime Numbers*. https://doi.org/10.32614/CRAN.package.primes.

Meyer, Fanny, and Victor Perrier. 2025. *Esquisse: Explore and Visualize Your Data Interactively*. https://doi.org/10.32614/CRAN.package.esquisse.

R Core Team. 2025. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. https://www.R-project.org/.