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.

Solution: I used the install.packages(FILLER) to install the esquisse package. I used quotation marks as to not confuse it with an object.

```
install.packages("esquisse") # this installs the package
```

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.

Solution

```
library("esquisse") # this allows me to use the package previously installed during a new session.
```

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.

Solution

```
help("esquisse") # typing this into the console directs you to a description of the package, the authors, two useful links, a place to report bugs, and
```

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.

Solution There is no demo or vignette.

```
demo("esquisse")
vignette("esquisse") #If this had a demo/vignette it would have directed me there, but it has neither.
```

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 Meyer and Perrier (2025) using "[@esquisse]".

Solution The package, esquisse, (Meyer and Perrier 2025) is useful for exploring and visualizing data interactively, specifically using "ggplot2" figures. These figures allow you to drag-and-drop to map your variables to different aesthetics.

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?

Solution There is a histogram that shows all three population's data. There appears to be a greater amount of data collected for the Adelle population. Each population's data is relatively symmetric with a general peak. The Adelle population has a more steep peak, while the Chinstrap and Gentoo populations are more moderately sloped. The Adelle and Chinstrap populations' body mass appear to have similar centers and ranges, indicating that these populations have very similar masses, while the Gentoo population is heavier on average.

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 I used a manual formula because there are few repeating terms and the difference between terms is not uniform.

```
(Fibonacci.Sequence = c(0, 1, 1, 2, 3, 5, 8, 13, 21, 34)) # using the manual function was necessary without clear repitions and uniform changes. Becaus
```

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

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 I used a manual formula because again, there were no repeating terms and the difference between terms increased by one with every new term.

```
(Triangular.Numbers = c(1, 3, 6, 10, 15, 21, 28, 36, 45, 55)) # this formula is difficult to create as a sequential vector because terms are based on t
```

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

2.3 Part c

Suppose I were designing a repeated measures 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.

Solution

```
(ID = rep(seq(from=1, to=10, by=1), each=3)) # a sequential vector is ideal for the inside as it assigns participants numbers one through ten. Making i
```

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

ii. Create a vector containing each `Treatment` repeated for each participant ID.

Solution

```
Treatment = rep((treatment = c("treatment.a", "treatment.b", "control")), times=10) # I manually entered the characters in for the data to be repeated,
factor(Treatment) # I made the vector into a factor vector because the information is categorical, and this is preferable if I were to do future manipu
```

```
 [1] treatment.a treatment.b control     treatment.a treatment.b control
 [7] treatment.a treatment.b control     treatment.a treatment.b control
[13] treatment.a treatment.b control     treatment.a treatment.b control
[19] treatment.a treatment.b control     treatment.a treatment.b control
[25] treatment.a treatment.b control     treatment.a treatment.b control
Levels: control treatment.a treatment.b
```

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 The resulting class is a character. Vectors are atomic, thus they can only work with one data type. The integar became a character because it can easily be conveyed as such, while a character is much more difficult to convey as a number.

```r
(part.d = c("MATH", 240)) # this is a vector with two different types of data.
```

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

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/.