

# Class Activity 2

Your name here

March 26 2024

## Problem 1

### Create and Save Your Name Trend Plot

In this activity, you will use R to explore the popularity of your name over time using the **babynames** package. You will then save this plot into a folder. Follow the steps below:

1. Replace "Dee" with your own name in the code below.
2. Run the code chunk to generate and save the plot in the **img/** folder.

```
library(ggplot2)
library(dplyr)
library(babynames)
library(stringr)

# Replace 'Dee' with your name
your_name <- "Dee"
your_name_data <- babynames %>% filter(name == your_name)

ggplot(data=your_name_data, aes(x=year, y=prop)) +
  geom_point(size = 3, alpha = 0.6) +
  geom_line(aes(colour = sex), size = 1) +
  scale_color_brewer(palette = "Set1") +
  labs(x = 'Year', y = str_c('Proportion of Babies Named ', your_name),
       title = str_c('Trends in Names: ', your_name))
```

### Load and Display Your Saved Plot

After saving your plot in the **img/** folder, the next task is to load and display the saved image. This demonstrates how to reuse images in your R Markdown documents.

1. Ensure your plot was saved in the previous task.
2. Use the code below to load and display your saved plot image.

```
knitr::include_graphics("img/your-plot-filename.png")
```

## Problem 2

In this activity, we'll explore some basic data assignments and manipulations in R. Understanding these fundamental concepts will help you work effectively with data in R. Let's dive into some practical exercises.

### a. Creating a Simple Vector

Vectors are one of the most basic data types in R. They hold elements of the same type. Let's create a vector containing all integers from 4 to 10. Call it **a1**.

*Answer:*

```
a1 <- 4:10
a1
[1] 4 5 6 7 8 9 10
```

## b. Creating a Vector of Even Integers

Now, let's create a vector that only contains even integers from 4 to 10. Call it **a2**.

*Answer:*

```
a2 <- seq(4, 10, by=2)
a2
[1] 4 6 8 10
```

## c. Adding Two Vectors

What do you think happens when we add two vectors of the same length in R? Let's find out by adding **a1** and **a2**.

*Answer:*

```
a1_plus_a2 <- a1 + a2
a1_plus_a2
[1] 8 11 14 17 12 15 18
```

## d. Summing Up Vector Elements

The `sum()` function calculates the total sum of all the elements in a vector. Let's see how it works with our vector **a1**.

*Answer:*

```
sum(a1)
[1] 49
```

## e. Finding the Length of a Vector

To find out how many elements a vector has, we can use the `length()` function. Let's apply it to **a1**.

*Answer:*

```
length(a1)
[1] 7
```

## f. Calculating the Average

f. Use the `sum` and `length` commands to calculate the average of the values in **a1**.

*Answer:*

```
average_a1 <- sum(a1) / length(a1)
average_a1
[1] 7
```

## g. Conditional Operations with `ifelse()`

The `ifelse()` function is useful for performing conditional operations on vectors. It takes a condition, a result for TRUE values, and a result for FALSE values.

Answer:

```
# Example: Replace even numbers with 1, and odd numbers with 0
ifelse(a1 %% 2 == 0, 1, 0)
[1] 1 0 1 0 1 0 1
```

## h. Combining Strings with paste()

The `paste()` function concatenates strings together. Let's combine some text with the elements of a vector.

Answer:

```
paste("Value is", a1)
[1] "Value is 4" "Value is 5" "Value is 6" "Value is 7" "Value is 8"
[6] "Value is 9" "Value is 10"
```

## i. Creating a Matrix

A matrix in R is a two-dimensional array that holds data of a single basic type. Let's create a simple matrix.

Answer:

```
my_matrix <- matrix(1:9, nrow=3, ncol=3)
my_matrix
      [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
```

## j. Making a DataFrame using cbind or rbind

Data frames are used to store tabular data in R. They can be created using the `cbind()` (column-bind) or `rbind()` (row-bind) functions. Here's how:

Answer:

```
df_cbind <- data.frame(cbind(a1, a2))
df_cbind
  a1 a2
1  4  4
2  5  6
3  6  8
4  7 10
5  8  4
6  9  6
7 10  8

df_rbind <- data.frame(rbind(a1, a1))
df_rbind
  X1 X2 X3 X4 X5 X6 X7
a1  4  5  6  7  8  9 10
a1.1 4  5  6  7  8  9 10
```

## k. Creating a List

Lists in R can hold elements of different types. They are incredibly versatile. Let's create a simple list.

Answer:

```
my_list <- list(numbers=a1, evenNumbers=a2, average=average_a1)
my_list
$numbers
[1] 4 5 6 7 8 9 10

$evenNumbers
[1] 4 6 8 10

$average
[1] 7
```

1. (*Bonus*) What would the following evaluate to? Could you think of a reason.

```
x <- c(7, 5, 3, 9)
y <- c(FALSE, factor(c("cellar", "door")), 2)
x - y
[1] 7 4 1 7
```

*Answer:*