# Introduction to **R**

Session 2 – Packages, functions, data cleaning and subsetting

For these exercises we will need the Lakes.csv data set read into **R**. Your code will look something like this:

```
lake.df = read.csv("PATH/TO/Lake.csv",
                   stringsAsFactors = FALSE)
```

## 1. Installing an R package

**R** packages are collections of user-defined functions. The function `std.error`, for example, is contained in the `plotrix` package.

1. Let's look at what happens when we try to use a function within a package before that package has been installed on our computer. Try to calculate the SEM of age using `std.error()`.

```
std.error(lake.df$pH)
```

```
#R:  Error in std.error(lake.df$pH): could not find function "std.error"
```

2. Install the package `plotrix` while in your **R** session by following the instructions below:
   (a) Select the **Packages** tab from the bottom right panel of your Rstudio interface.
   (b) Click on the *Install Packages* icon just below **Packages**.
   (c) Type `plotrix` in the blank space provided below "*Packages (separate multiple with space or comma):*"
   (d) Select *No* if you are asked you to restart **R**.
   (e) Submit the code `library(plotrix)` to the **R** console to make the functions contained in the `plotrix` package, available in the current **R** session.

```
library(plotrix)
```

3. Now, use `std.error` to calculate the standard error of the pH.

```
std.error(lake.df$pH)
```

```
#R:  [1] 0.1862989
```

4. Try writing your own code to calculate the standard error of the pH (don't write your own function for this… yet).

$$\hat{\text{SEM}} = \frac{s}{\sqrt{N}}$$

Hint: `sd()` calculates the standard deviation, and `length()` can be used to calculate N.

```
with(lake.df, sd(pH, na.rm = TRUE) / sqrt(length(pH)))
```

```
#R:  [1] 0.1809494
```

## 2. Write your own function

In Session 2 you were shown a simple function to calculate the standard error of the mean (SEM):

```
sem_function = function(input){
  s = sd(input, na.rm = TRUE) # Calc std. deviation
  N = length(input)           # Calc sample size
  s / sqrt(N)                 # Definition of SEM
}
```

1. Type the above code into your **R** script and submit it to the **R** console.
2. Modify the function in 2.1 so that the output will have only 2 decimal places.

```
sem_function = function(input){
       s = sd(input, na.rm = TRUE)
       N = length(input)
       round(s / sqrt(N), 2)
}
```

3. Calculate the SEM of `pH` using the function you created in 2.2.

```
sem_function(lake.df$pH)
```

```
#R:  [1] 0.18
```

## 3. Subsetting datasets

1. Find the following:

- The pH of the first lake.

```
lake.df$pH[1]
```

```
#R:  [1] 6.1
```

- The pH of the last lake.

```
lake.df$pH[53]
```

```
#R:  [1] 7.9
```

```
# OR:
lake.df$pH[nrow(lake.df)]
```

```
#R:  [1] 7.9
```

```
# Note: nrow() returns the number of rows in the data set
```

- The pH values of the first and last lakes.

```
lake.df$pH[c(1, 53)]
```

```
#R:  [1] 6.1 7.9
```

```
# OR:
lake.df$pH[c(1, nrow(lake.df))]
```

```
#R:  [1] 6.1 7.9
```

- All measurements made on the third lake.

```
lake.df[3, ]
```

```
#R:    ID   Lake  pH Calcium Chlorophyll
#R:  3  3 Apopka 9.1    High       128.3
```

- All pH values.

```
lake.df[, "pH"]
```

```
#R:   [1] 6.1 5.1 9.1 6.9 4.6 7.3 5.4 8.1 5.8 6.4 5.4 7.2 7.2  NA 7.6 8.2 8.7
#R:  [18] 7.8 5.8 6.7 4.4 6.7 6.1 6.9 5.5 6.9 7.3 4.5 4.8 5.8 7.8  NA 3.6 4.4
#R:  [35] 7.9 7.1 6.8 8.4 7.0 7.5 7.0 6.8 5.9 8.3  NA 6.2 6.2 8.9 4.3 7.0 6.9
#R:  [52] 5.2 7.9
```

2. Calculate:

- The average pH of lakes with low Calcium concentration.

```
with(lake.df, mean(pH[Calcium == "Low"]))
```

```
#R:  [1] 5.229412
```

- The average pH of lakes with high Calcium concentration.

```
with(lake.df, mean(pH[Calcium == "High"], na.rm = TRUE))
```

```
#R:  [1] 7.835714
```

- The average pH of lakes with low Calcium concentrations *and* Chlorophyll concentrations less than 10.

```
with(lake.df, mean(pH[Calcium == "Low" & Chlorophyll < 10], na.rm = TRUE))
```

```
#R:  [1] 4.933333
```

# 4. Challenge

Modify the function given in 2.1, so that the function will return a 95% confidence interval (with 2 decimal places).

Hint: A 95% confidence interval of a variable X is given by the average of X $\pm$ 1.96 $\times$ SEM of X.

```
sem_ci = function(input){
       avg = mean(input, na.rm = TRUE)
       stdev = sd(input, na.rm = TRUE)
       N = length(input)
       sem = stdev / sqrt(N)
       upperCI = round(avg + 1.96 * sem, 2)
       lowerCI = round(avg - 1.96 * sem, 2)
       c(lowerCI, upperCI)
}

sem_ci(lake.df$pH)
```

```
#R:  [1] 6.23 6.94
```