Intro to R: Week 1

Topics Covered: Variables, Operations, Data Types, Indexing, Which and If Statements

Task 1: Create a variable called `a` to store the number 5

- Variables store values or objects so that they can be accessed later
- In R, we use `<-` to assign values or objects to variables and `=` to set function arguments. The shortcut for `<-` in RStudio is alt-.
- In RStudio you can run the current line of code with the shortcut Ctrl+Enter or Command+Enter

```
a <- 5 # this statement is pronounced "a gets 5"

# When you run this code, what happens?  Where did that number go?

a # typing a into your console returns the value stored in that variable
```

Task 2: Translate the statement "b gets 10" to code. Is `a` equal to `b`? Which is greater, `a` or `b`?

```
b <- 10

a == b
a != b

b > a
a < b
```

Task 3: Add, subtract, multiply, and divide a and b and store the reults as new variables

```
add.vals <- a+b

sub.vals <- a-b

mul.vals <- a*b

div.vals <- a/b
```

Task 4: Use the function `c()` to create a variable to store the numbers from 1 to 10.

- In R, round braces are used for function calls: `function(arguments go here)`

```
x <- c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10) # c is for combine

x # x is now a vector of integers from 1 to 10
```

Task 5: use `seq()` and `:` to re-generate identical vectors

```
?seq() # the seq function requires arguments for from, to, and by

x <- seq(from=1, to=10, by=1)

?':' # the colon operator requires arguments from and to and only steps by 1

x <- 1:10
```

Task 6: Use 'seq()' to create a vector of odd numbers between 1 and 20. How many numbers are in this new vector?

```
y <- seq(1, 20, by=2)

length(y)
```

Task 7: Add, subtract, multiply, and divide `x` and `y`. What happens?

```
x+y

x-y

x*y

x/y
```

- These are all element-wise operations, so they operate on each element of the vector individually

---

Task 8: Multiply `a` by `x`. What happens?

- Create a variable `c` that stores the values 1 and 2 and multiply it by `x`. What happens?
- Create a variable `d` that stores the values 1, 2, and 3 and multiply it by `x`. What happens?

```
a*x # each element of x is multiplied by a

c <- 1:2
c*x # the 1st element of x is multiplied by 1, the 2nd by 2, the 3rd by 1

d <- 1:3
d*x # the 1st element is multiplied by 1, the 2nd by 2, the 3rd by 3, and the 4th by 1
```

- If you get a warning message, this is an FYI from R that what you meant to do might not be what actually happened. It's a heads up, but doesn't impact the result. Error messages are different. They halt the execution of the code.

Task 9: Add all of the numbers in x and y together into a single value using `sum()`. Multiply the sum of `x` by the sum of `y`.

```
sum(x+y)

sum(x) + sum(y)

sum(x) * sum(y)
```

Task 10: Store your name in a variable called `name`. Multiply your name by 5.

```
my.name <- "Eiren"

my.name*5 # this doesn't work.
```

- The variable `my.name` is not numeric, so it can't be multiplied by 5
- There are four basic types of data: numeric, integer, character, logical

---

Task 11: Create four vectors, one of each data type, and check their type using the function 'class()'

```
my.numeric <- c(1.5, 23, 8850.2)

class(my.numeric)

my.integer <- c(1, 2, 3)

class(my.integer)

my.character <- c("Eiren", "Jess", "Peter")

class(my.character)

my.logical <- c(TRUE, TRUE, FALSE)

class(my.logical)
```

Task 12: You just got back from a marine mammal survey and you saw 5 fin, 2 blue, 14 humpback, 0 minke, and 1 gray whale. Create vectors to store the species names and the number of whales seen. Combine these vectors into a single object. From this variable, extract the number of humpback whales seen.

- In R, square brackets are used for indexing within objects: `object[index]`

```
species <- c("Fin", "Blue", "Humpback", "Minke", "Gray")

whales <- c(5, 2, 14, 0, 1)

names(whales) <- species

whales["Humpback"] # square brackets are used for indexing within objects

whales[3] # this accomplishes the same thing
```

Task 13: What was the maximum number of whales seen and which species was it? Did you see zero of any species? Which?

```
max(whales) # returns the max no. of whales seen

which.max(whales) # returns the vector position of the max no. of whales seen

# We can use this vector position as an index to return the species name we want

species[which.max(whales)] # returns the species name of the max no. of whales

any(whales==0) # returns T/F whether any values are equal to zero

which(whales==0) # returns the vector position of which species was seen 0 times

species[which(whales==0)] # Find the species name of the value equal to zero
```

Task 14: Whenever you see more than 15 whales in total, you report that you saw many, but when you see less than 15, you report that you saw few. Use an `ifelse()` statement to indicate whether you should report few or many whales.

- Curly braces are used for statements, mostly in conjunction with `if`, `for`, and function definitions.

```
report <- if (sum(whales)>15) {"many"}

report <- ifelse(sum(whales)>15, "many", "few")
```

Task 15: Store your information about whales as RData and .csv files

```
save(whales, file="whaleSurvey") # save the vector as an RData file

rm(whales) # remove whales from the workspace

# whales no longer exists because we removed it :(

load("whaleSurvey") # now the data are available in the workspace again

write.csv(whales, "whaleSurvey.csv") # store the data to a .csv file
```