Intro to R: Week 2

Topics Covered: Datasets, indexing, and if/else statements

Last week we talked about how to store data in vectors, which are one-dimensional objects. With more complex data, we can use matrices, data frames, and lists to store information in multiple dimensions.

- A **matrix** is a collection of elements of the same data type (numeric, character, or logical) arranged in a fixed number of rows and columns. An array is an n-dimensional matrix.
- A **data frame** has variables as columns and observations as rows and can contain elements of multiple data types
- A **list** can contain different kinds of objects (data frames, vectors, matrices, etc.) with different dimensions

Task 1: Use the `matrix()` function to create a matrix of the numbers 1 to 100 in ten rows and ten columns. Can you get the numbers to read from left to right instead of from top to bottom?

Elements within matrices can be accessed using square brackets in two ways:

- Element-wise indexing: `mat[e]` returns the eth element within the matrix (just like a vector!)
- Row-and-column indexing: `mat[r, c]` returns the element in the rth row and the cth column

Task 2: Extract the number 50 from your matrix using both row-and-column and element-wise indexing

Task 3: Use the functions `colSums()` and `rbind()` to add a row to the bottom of your matrix that contains the mean of the numbers in each column

---

Matrices are great, but most of the time, biologists collect data of several different types, so data frames are more appropriate.

Task 4: Use the `data.frame()` function to store the following data (stolen from Sandin et al. 2008). What happens if you store the data in a matrix instead? Why might this be a problem?

Island	Inhabited	Coral Cover (%)	Fish Biomass (mT haˆ-1)
Kingman	No	44	5
Palmyra	No	20	3
Tabuaeran	Yes	20	2
Kiritimati	Yes	15	1

In addition to element-wise and row-and-column indexing, we can also use column and row names to index within matrices and data frames. `island.mat[,"coral"]` and `island.df[,"coral"]` will return only the coral columns of these objects. Additionally, data frames can be indexed using the `$` to access named columns, so `island.df$coral` also returns the coral column of this data frame.

Task 5: What is the average coral cover of islands that are uninhabited?

Task 6: Your advisor asks to see only the data from inhabited islands. Use the `subset()` function to extract this information from your data frame.

Task 7: Now imagine that you want to know whether fish biomass is greater on inhabited or uninhabited islands.

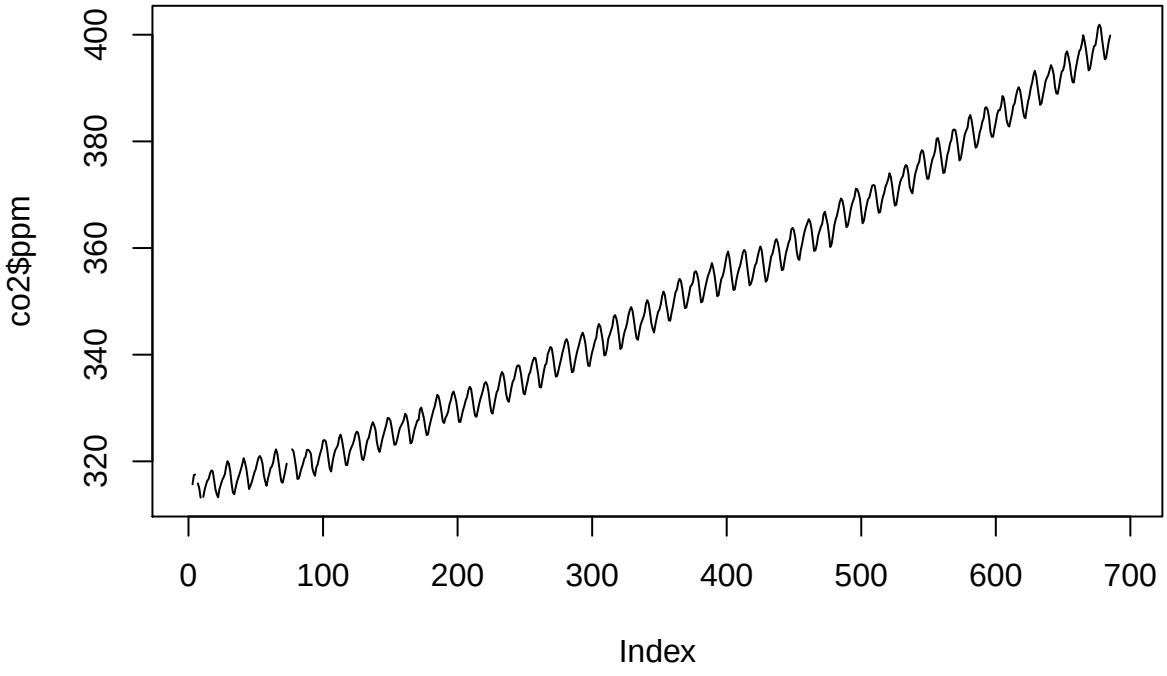
Step 7.1 Calculate the mean fish biomass on inhabited and uninhabited islands

Step 7.2 Use an `if()` statement with a logical test on the mean fish biomass calculated above to print a message indicating whether fish biomass is greater on inhabited or uninhabited islands. Your code should have the following structure:

```
if (logical test goes here) {thing to do if logical test is true} else {
                             thing to do if logical test is false
}
```

Task 8: Use the function `read.csv()` to read in the file 'monthly_mlo.csv'.
What kind of data object is it?

Task 9: Not surprisingly, this data set is not perfect. Replace any missing values with NA and use the `plot()` function to create a line graph of CO2 values over time.



Sometimes during research projects we collect many different types of data that aren't necessarily directly linked. For example, perhaps you collect CTD data AND count organisms collected in mid-water trawls AND have logs of whales detected in sonobuoy recordings all from the same location. These data can't be combined into a single data frame, but it is useful to have them all in the same place. This is where **lists** come in handy. Think of them as R databases that can be stored as single objects (.RData files) and can contain a bunch of different information.

Task 10: Use the `list()` function to create a list that contains the whale survey data from Week 1, the islands data set we created earlier today, and the Mauna Loa CO2 data. Use the `str()` function to examine the structure of the list.

Lists use a combination of double and single square brackets to access nested elements of a list. The `$` can also be used to access elements of a list. From the output of `str()` you can see how to access elements using `$`.

Task 11: There are at least nine different ways to extract the coral column of the islands data set from this list. Can you find three of them?