02-06: Vectors and Data Frames

1 - Purpose

- Creating vectors
- Adding vectors to a data frame
- Rearranging data frames
- Saving data frame to a CSV file

2 - Concepts

3 - Creating your own vector

In previous lessons, we pulled out a column from a data frame and saved that column to a vector such as ***highTempData*** or ***lowTempData***. There are also many situations where we want to create our own vector. For instance, we might want to add a humidity column to the data frame.

In R, a vector is a variable that has multiple associated values. If you want to declare a variable as a vector as opposed to a single value, you use the ***c()*** function. The values for the vector are put inside the parenthesis of ***c()*** and separated using commas.

In the following code, we declare four different vector variables and assign values to them.

```
{
  rm(list=ls()); options(show.error.locations = TRUE);

  vectorVar1 = c(5, 10, 20, 4, 12, 7, 2);
  vectorVar2 = c(-5.2, 7.3, 0.2, 21, -1.235);
  vectorVar3 = c("llama", "goat", "alpaca", "guanaco");
  vectorVar4 = c(7, "llama", 3, "alpaca");
}
```

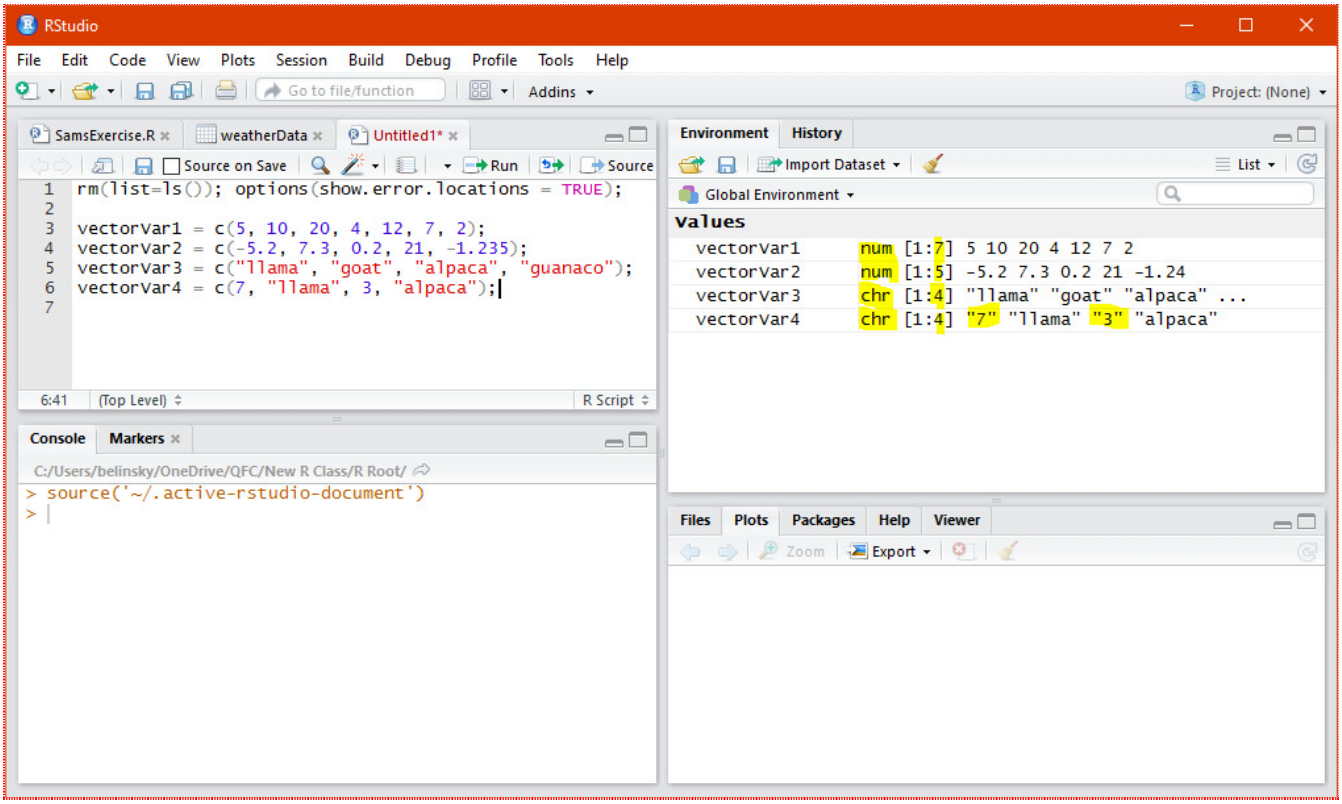

Fig 1: Creating four vector variables and assigning values to them

In the Environment Window, you can see three of the properties of the vectors

type: either string (chr) or numeric (num)
length: given by the second number in (1:x)
values: numbers or strings (in quotes)

All values in a vectors must be the same data type so if there are mixed strings and numbers in a vector, all the numbers will be converted to strings. Notice that for the 4th vector, R converted the numbers ***7*** and ***3*** into strings -- so 7 became "7" and 3 became "3".

3.1 - Subsetting vector

We can subset these vectors just like in previous lessons using the subset ([ ]) operator:

```
{
  rm(list=ls()); options(show.error.locations = TRUE);

  vectorVar1 = c(5, 10, 20, 4, 12, 7, 2);
  vectorVar2 = c(-5.2, 7.3, 0.2, 21, -1.235);
  vectorVar3 = c("llama", "goat", "alpaca", "guanaco");
  vectorVar4 = c(7, "llama", 3, "alpaca");

  cat("3rd value of the 1st vector:", vectorVar1[3], "\n");
  cat("2nd value of the 4th vector:", vectorVar4[2], "\n");
  cat("5th value of the 2nd vector:", vectorVar2[5], "\n");
```

```
12 }
```

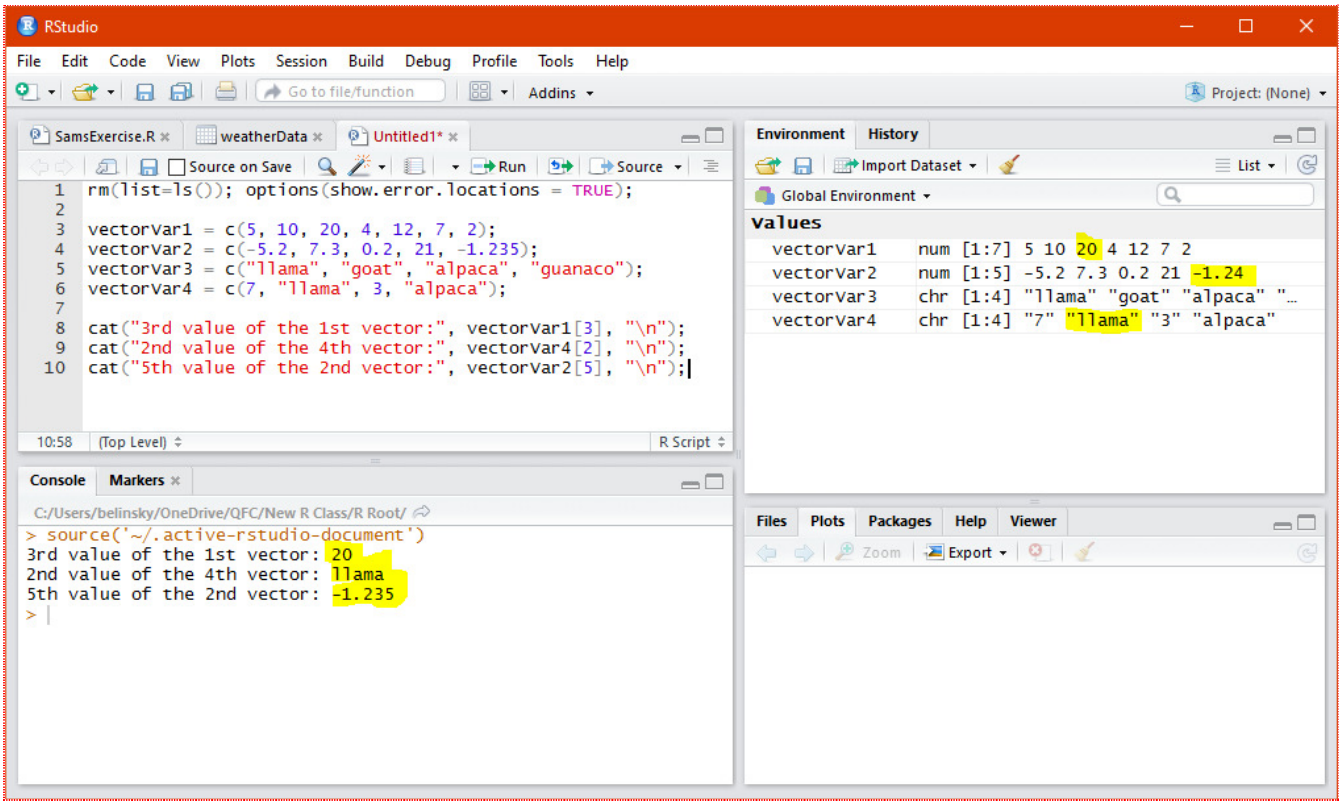

Fig 2: Subsetting the vector variables we created

4 - Manipulating data frames

In a previous lesson we calculated the values for the vector ***changeInTemp***, the difference between the high temperature and low temperature for each day.

Now let's say we want to add the ***changeInTemp*** vector to the CSV file. We would first need to save the ***changeInTemp*** vector to the ***weatherData*** data frame and then write the data frame to a CSV file.

4.1 - Add a vector to a data frame

In R,we can use the subsetting operator ([ ]) to add a vector to a data frame. The following script adds the ***changeInTemp*** vector to the ***weatherData*** data frame.

```
1 {
2   rm(list=ls()); options(show.error.locations = TRUE);
3
4   weatherData = read.csv("data/LansingWeather2.csv");
5   highTempData = weatherData[,2];
6   lowTempData = weatherData[,3];
7   changeInTemp = c(); # declare a vector variable
8
9   for(day in 1:length(highTempData))
```

```
  {
     changeInTemp[day] = highTempData[day] - lowTempData[day];
  }

  # add vector to data frame -- give it the name "deltaTemp"
  weatherData[,"deltaTemp"] = changeInTemp;
}
```

The line

```
  weatherData[,"deltaTemp"] = changeInTemp;
```

looks for a column header named "deltaTemp" and adds the values of the vector ***changeInTemp*** to the column. Since "deltaTemp" does not exist, R adds a column to the data frame, gives it the header "deltaTemp", and adds the values from ***changeInTemp*** to the column. If "deltaTemp" already existed, then the values would be overwritten with the values from ***changeInTemp***.

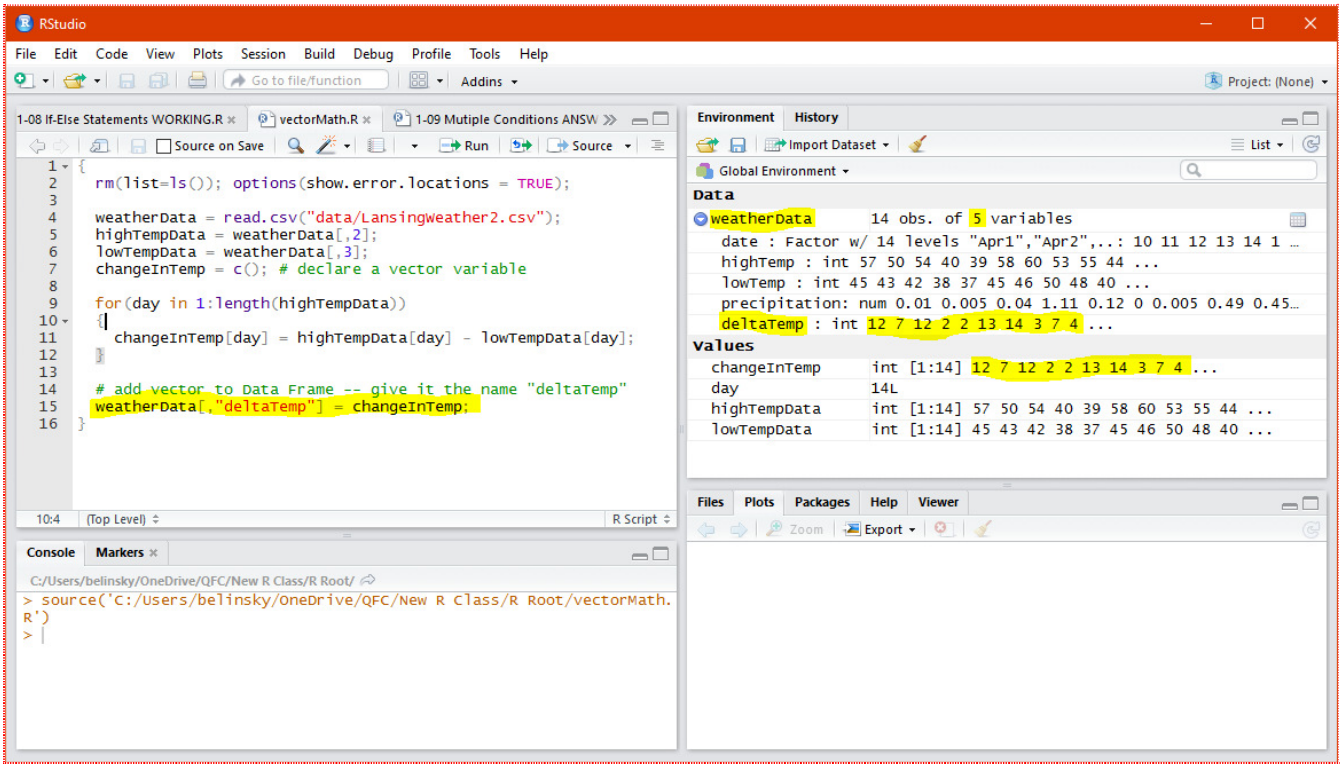

*Fig 3: Adding the **changeInTemp** vector to the **weatherData** data frame*

Note: ***weatherData*** now has 5 variables (instead of 4).

Double clicking on ***weatherData*** in the Environment Window displays the Data in visual form in the Main Window.

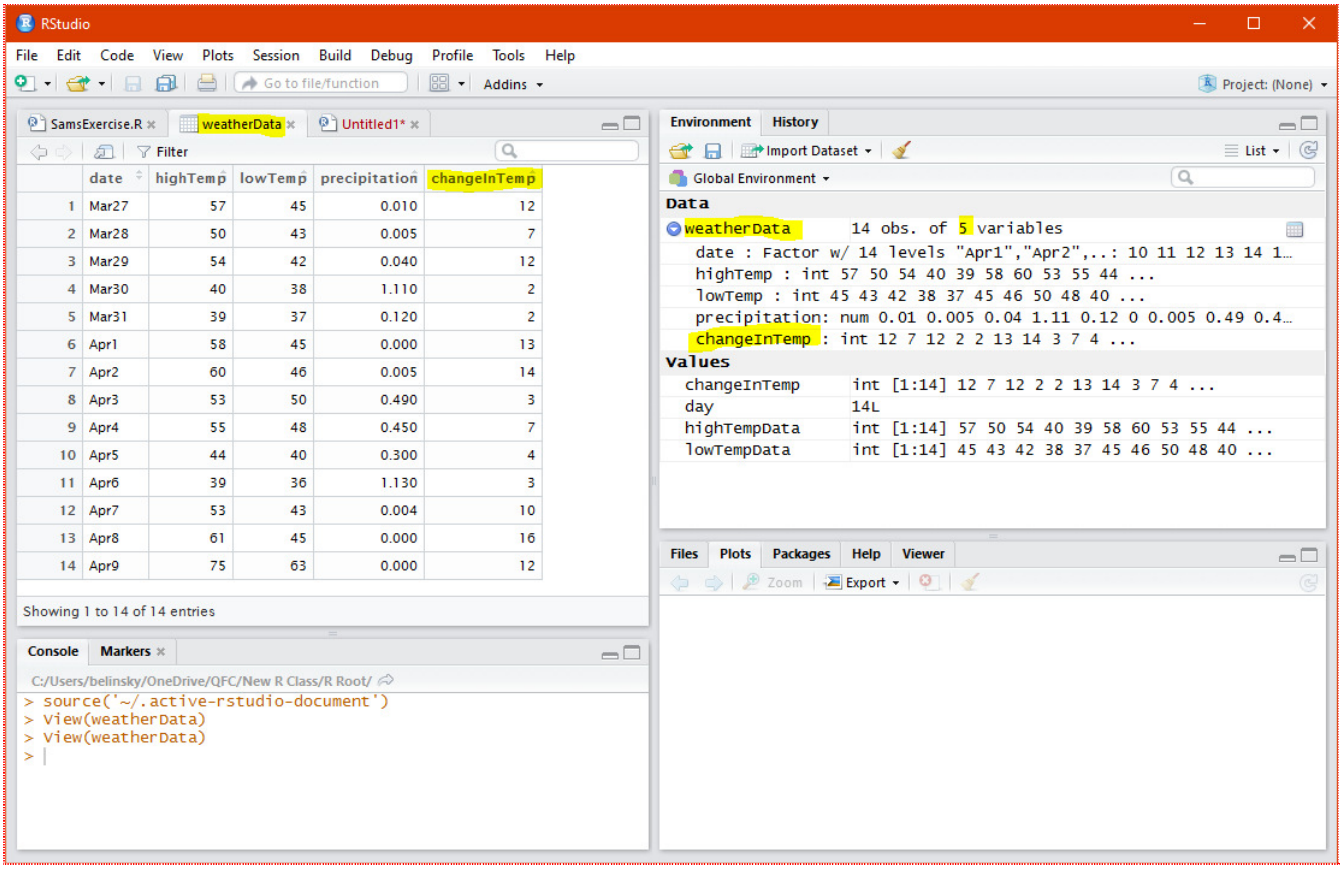

*Fig 4: Visual representation of **weatherData** data frame after **changeInTemp** vector is added.*

4.2 - Adding a created vector

You can create your own vector, put in values and add it to the data frame. One thing to note is that the length of the vector must be the same as the number of rows in the data frame. Luckily, R gives you a pretty convenient error if the length of your vector is incorrect. Let's try this by incorrectly adding an average humidity vector of a different length to ***weatherData***.

```
{
  rm(list=ls()); options(show.error.locations = TRUE);

  weatherData = read.csv("data/LansingWeather2.csv");

  # vector with 16 values in it
  aveHumidity = c(88,87,80,74,67,84,89,59,55,72,82,83,87,58,48,41);
  weatherData[,"Humidity"] = aveHumidity;
}
```

Note: the error state there are 16 values instead of the expected 14.

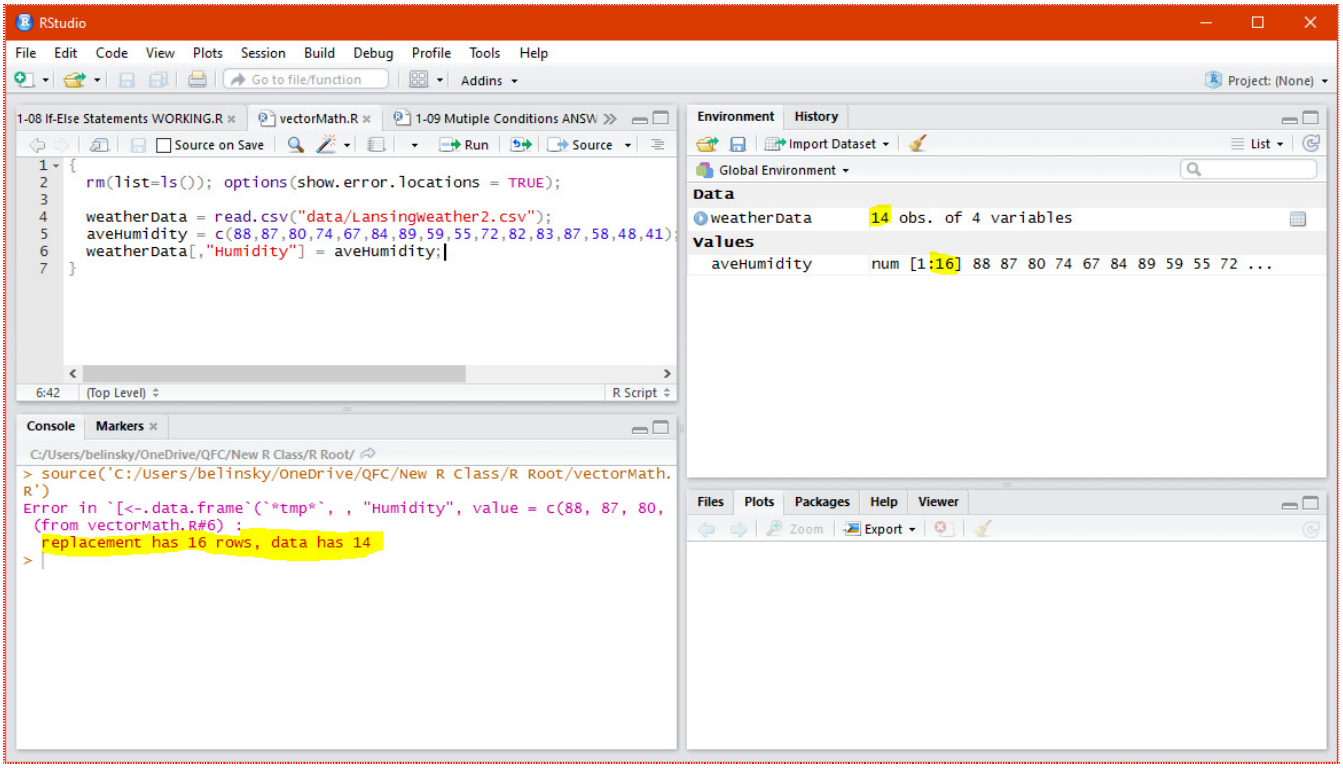

Fig 5: Putting a vector of wrong size in a data frame

Let's take off the last two values in ***aveHumidity*** to make it the right size vector to fit in ***weatherData***.

```
{
  rm(list=ls()); options(show.error.locations = TRUE);

  weatherData = read.csv("data/LansingWeather2.csv");
  aveHumidity = c(88,87,80,74,67,84,89,59,55,72,82,83,87,58);
  weatherData[,"humidity"] = aveHumidity;
}
```

In the Environment Window, we see that the vector ***aveHumidity*** has been added to ***weatherData*** with the column header "humidity"

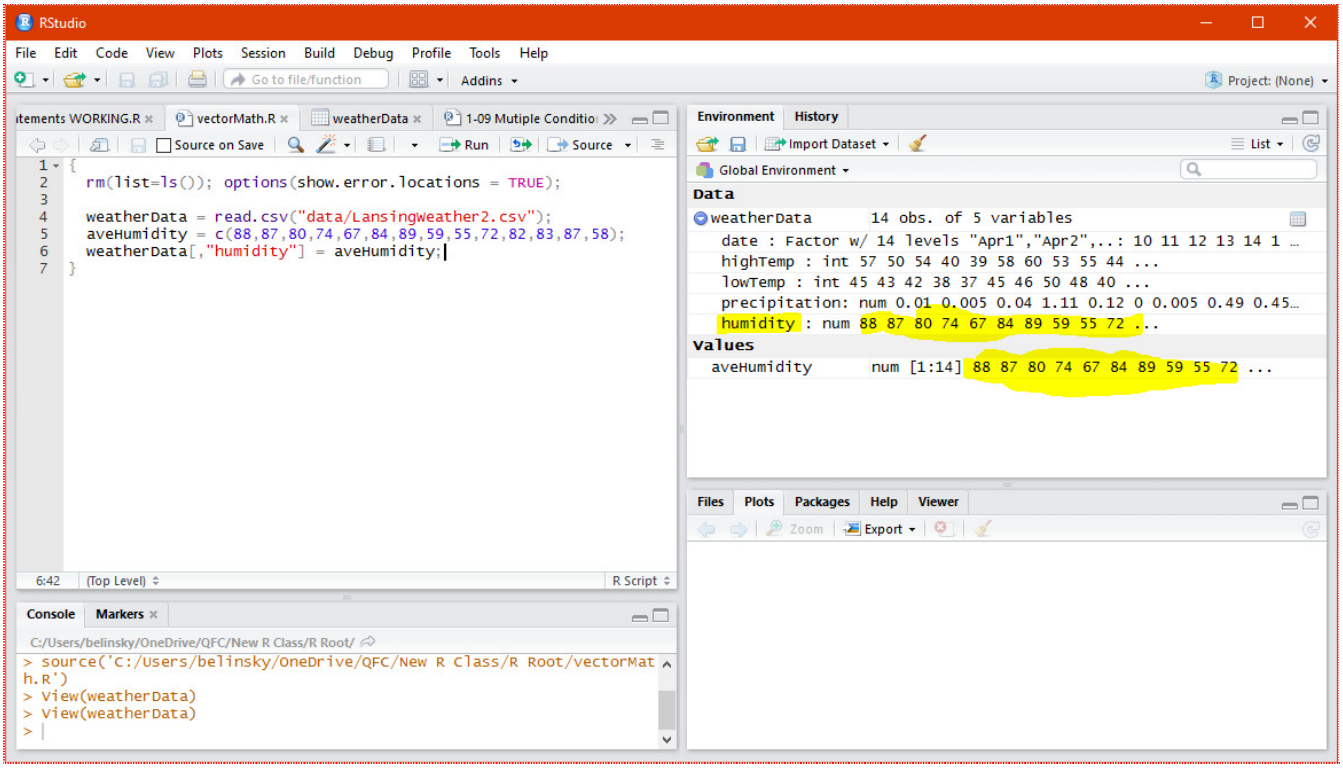

*Fig 6: Adding a vector to **weatherData***

Double clicking on ***weatherData*** and we see the updated data frame in the Main Window with "humidity" added

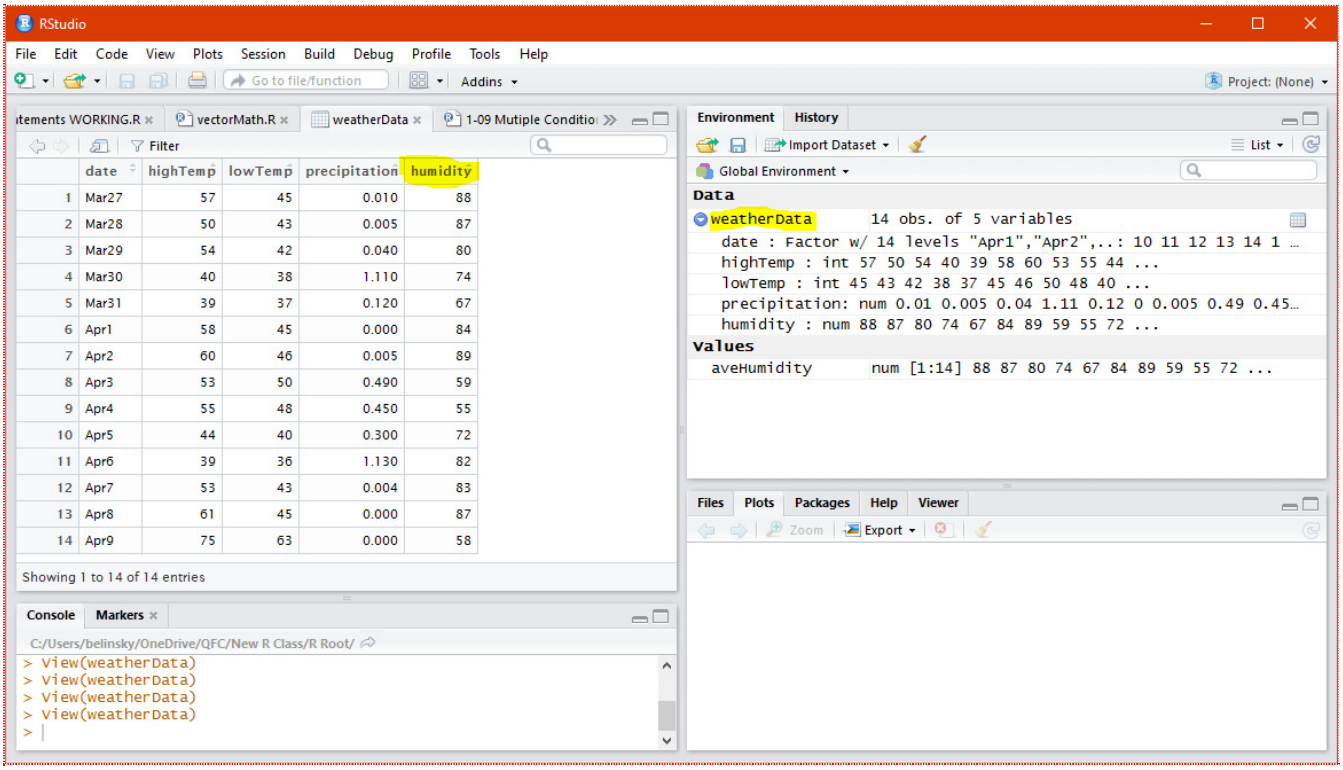

*Fig 7: Viewing the complete **weatherData** data frame*

4.3 - Rearranging columns in the data frame

The ***weatherData*** data frame now has 5 column variables with "humidity" added to the end of the data

frame. However, we might not want "humidity" as the last column in the data frame. Or, for that matter, we might want to change the order of multiple columns in ***weatherData***.

We can rearrange the columns in a data frame however we wish by using ***c()*** inside the subsetting operation (***[ ]***).

```
{
  rm(list=ls()); options(show.error.locations = TRUE);

  weatherData = read.csv("data/LansingWeather2.csv");
  aveHumidity = c(88,87,80,74,67,84,89,59,55,72,82,83,87,58);
  weatherData[,"humidity"] = aveHumidity;

  newWeatherData = weatherData[ ,c(1,5,4,2,3)];
}
```

There is a lot going on in the last line of the script. On the right side we are subsetting ***weatherData*** by its columns.
weatherData[, c(1,5,4,2,3)] says to gather the columns in this particular order: 1,5,4,2,3

We assigned the new column arrangement to ***newWeatherData***.

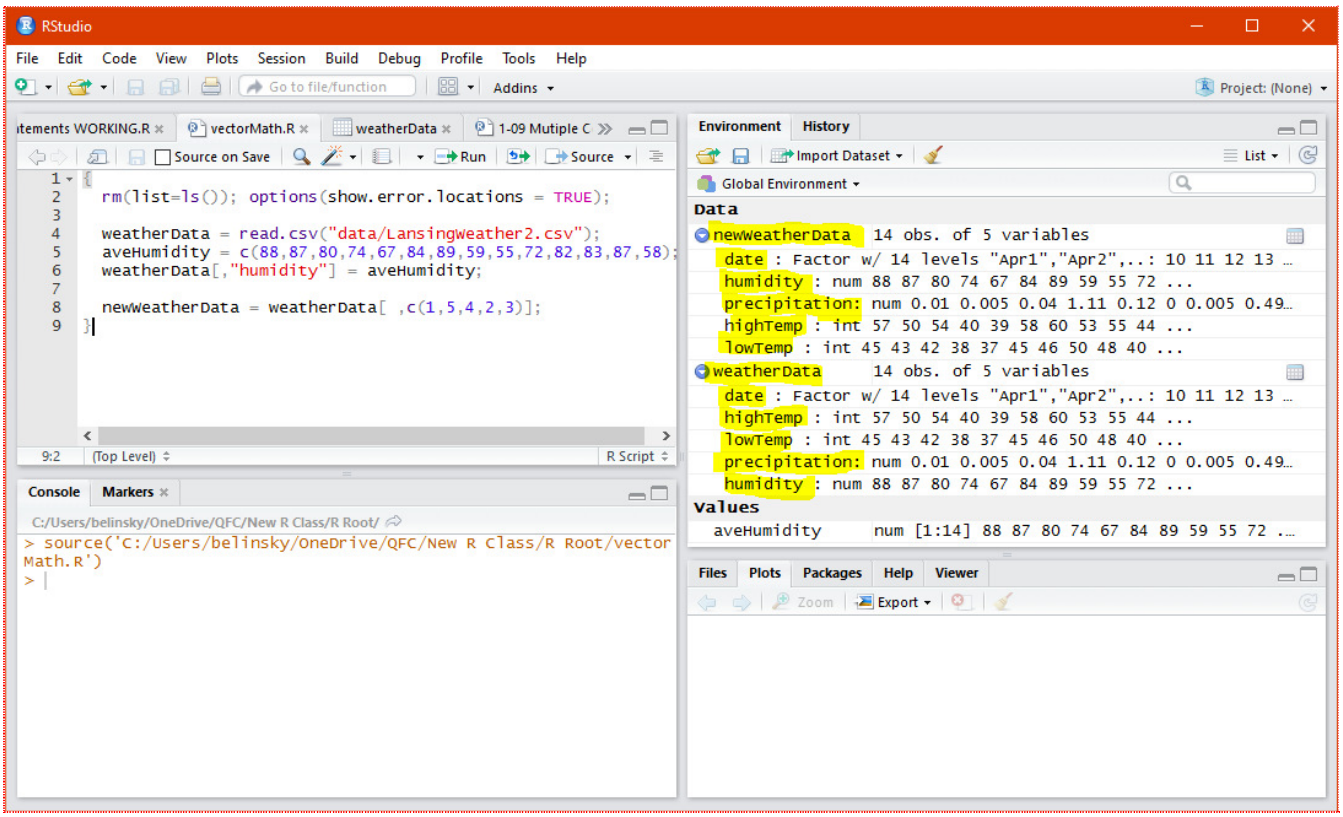

Fig 8: Rearrange the columns of a data frame

We could have also saved the rearranged column back to the same data frame, so the last line of the script would look like:

```
  weatherData = weatherData[ ,c(1,5,4,2,3)];
```

This line will overwrite ***weatherData*** and replace it with the new column arrangement.

4.4 - Removing columns

We can also use this technique to delete a column by not including the columns we don't want. So, if we do not want the precipitation column, we just avoid including precipitation's column number, which is 4

```
1  newWeatherData = weatherData[ ,c(1,2,5,3)];
```

Now we will have a new data frame that has only four columns: date, humidity, highTemp, and lowTemp -- precipitation is gone.

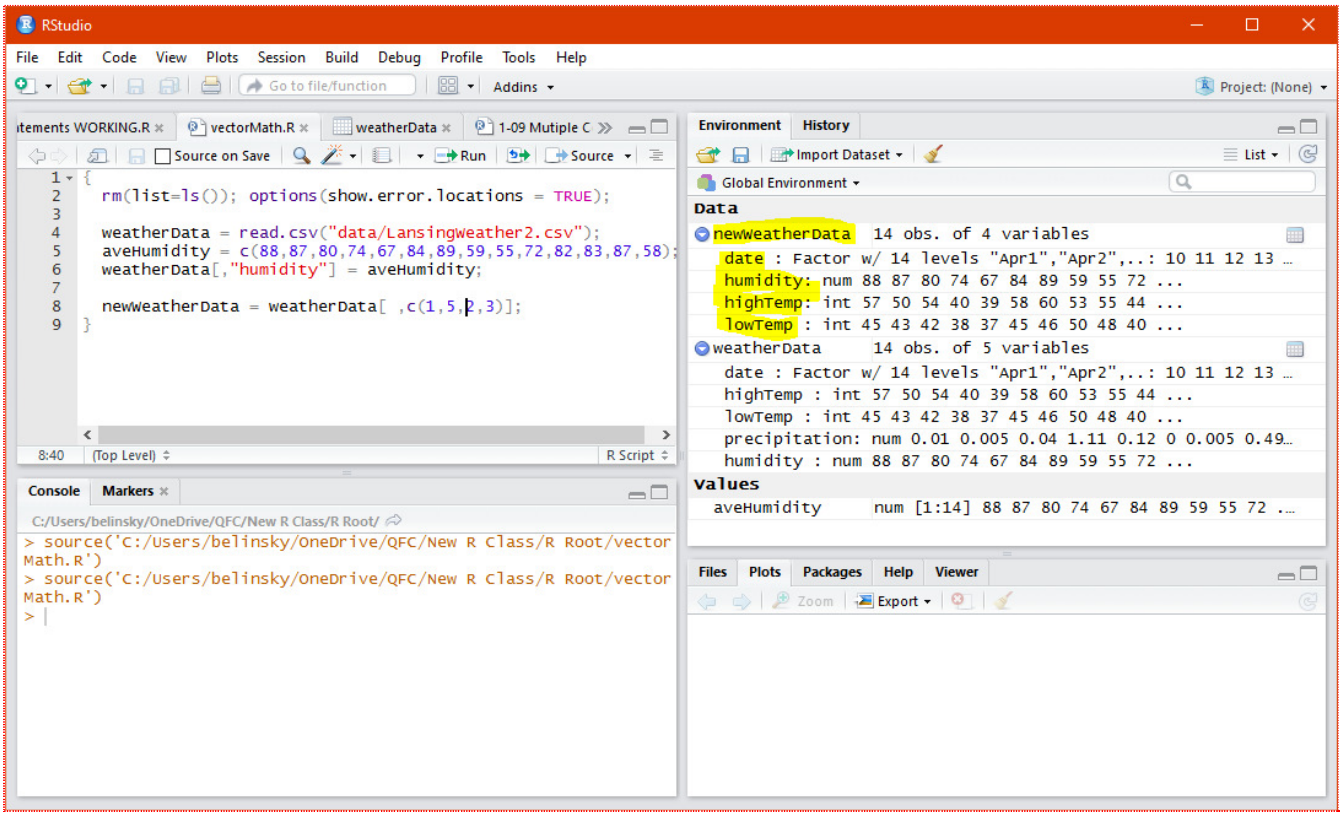

Fig 9: Rearranging and removing columns from a data frame

5 - Saving the data frame to a CSV file

Lastly, we might want to save a permanent copy of the data frame we created in the R script. To save the new data frame to a CSV file we use the command: ***write.csv()*** with the parameters:

1) ***x***: The data frame to save (in this case, ***newWeatherData***)
2) ***file***: The file name to save it as (in this case, **LansingWeather3.csv**) -- make sure the file name is in quotes

```
1 {
2   rm(list=ls()); options(show.error.locations = TRUE);
3
4   weatherData = read.csv("data/LansingWeather2.csv");
```

```
  aveHumidity = c(88,87,80,74,67,84,89,59,55,72,82,83,87,58);
  weatherData[,"humidity"] = aveHumidity;

  # weatherData with humidity added and rearranged column
  newWeatherData = weatherData[ ,c(1,5,4,2,3)];
  write.csv(x=newWeatherData, file="data/LansingWeather3.csv");
}
```

The above code saves ***newWeatherData*** as ***LansingWeather3.csv*** in the data folder inside your root R directory. After you execute the script above, you should notice that there is a new file called ***LansingWeather3.csv*** in the ***data*** folder. This new csv file can be opened in excel or in any text editor (including RStudio).

Note: it is (too) easy to overwrite files using the script above. If the last line has been:

```
  write.csv(x=newWeatherData, file="data/LansingWeather2.csv");
```

then the file ***LansingWeather2.csv*** would have been overwritten without any warning.

6 - Application

A) Add both the ***changeInTemp*** and ***humidity*** vectors to the ***weatherData*** data frame (form **data/lansingWeather2.csv**).

B) Rearrange the ***weatherData*** data frame so that the columns are in this order
1) date
2) lowTemp
3) change in temp
4) highTemp
5) humidity
6) precipitation

C) Remove the ***highTemp*** column and save the data to another data frame

D) Save the data frame from (C) to a new CSV file