02-02: Subset Operations

1 - Purpose

- Access specific values within a data frame
- Access a range of values within a data frame and save to a vector
- Access values within a vector

2 - Questions about the material...

If you have any questions about the material in this lesson or your Class Project *feel free to email them to the instructor here*.

3 - Accessing specific values within a data set

For this lesson we will use an expanded version of the Lansing weather data that adds two more columns: ***date*** and ***precipitation.***

```
date,highTemp,lowTemp,precipitation
Mar27,57,45,0.01
Mar28,50,43,0.005
Mar29,54,42,0.04
Mar30,40,38,1.11
Mar31,39,37,0.12
Apr1,58,45,0
Apr2,60,46,0.005
Apr3,53,50,0.49
Apr4,55,48,0.45
Apr5,44,40,0.30
Apr6,39,36,1.13
Apr7,53,43,0.004
Apr8,61,45,0
Apr9,75,63,0
```

Create a new text file in RStudio (***File -> New File -> Text File***) and paste the above data inside the file. We will save this file as ***LansingWeather2.csv*** so we do not overwrite the old file. Save ***LansingWeather2.csv*** to the ***data*** folder inside the default working directory, ***R Root*** (*Fig.1*).

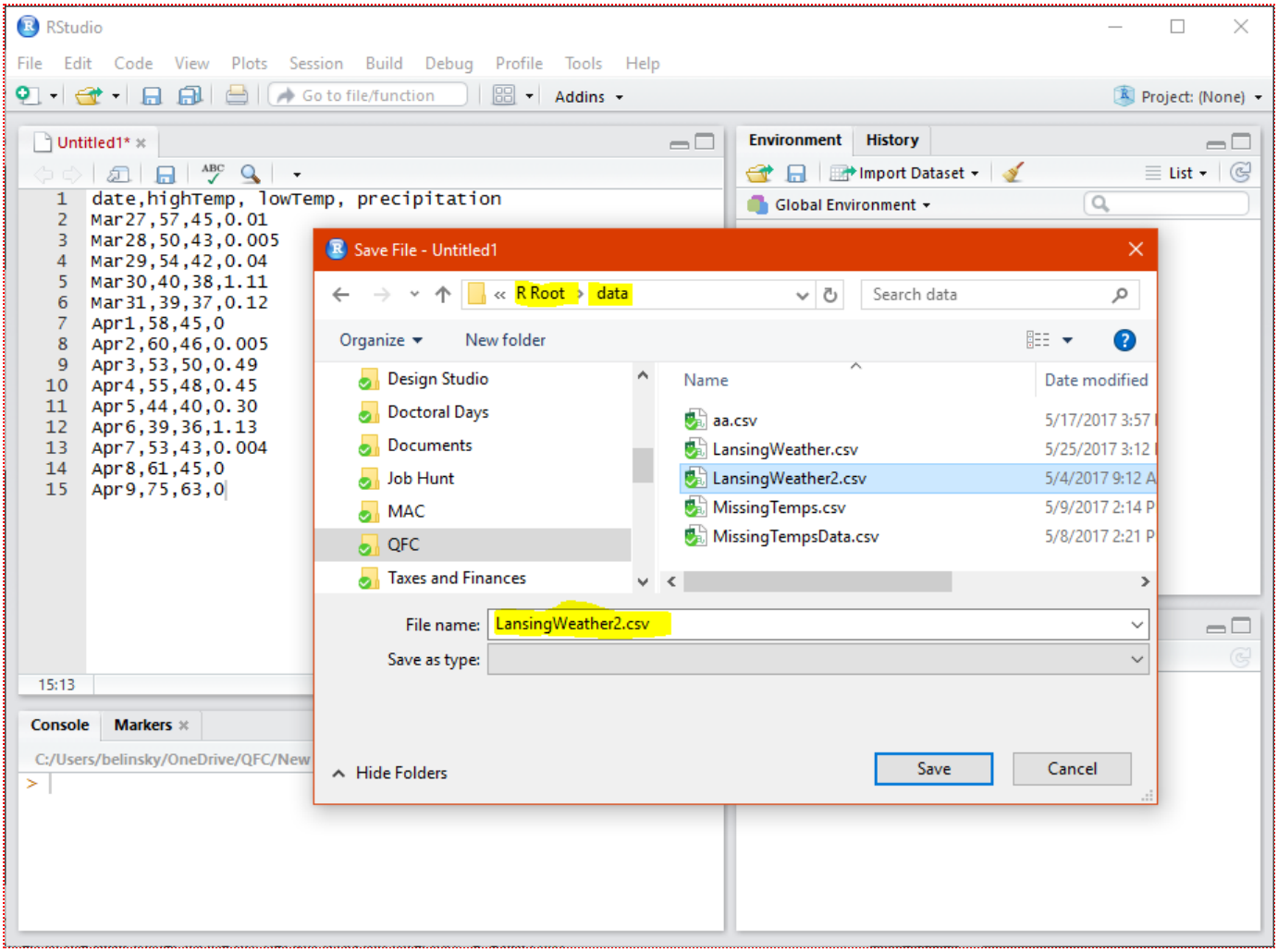

*Fig 1: Save **LansingWeather2.csv** file to the **data** folder*

3.1 - Input data from a file to a DataFrame

We will use a similar script from last lesson to read in the data from ***LansingWeather2.csv*** and save it to the variable ***weatherData***.

```
1 {
2   rm(list=ls());  options(show.error.locations = TRUE);
3
4   # read data from LansingWeather2.csv and save to the variable weatherData
5   weatherData = read.csv("data/LansingWeather2.csv");
6 }
```

Line 4 of the script opens the ***LansingWeather2.csv*** file and saves the information inside the ***LansingWeather2.csv*** file to the variable named ***weatherData***. ***weatherData*** is called a *data frame* in R. A data frame is a collection of related values.

Looking at the Environment Window, we see ***weatherData*** *now* has "**14** obs. of **4** variables". Double click on ***weatherData*** in the Environment Window to open the table in a window (*Fig.2*).

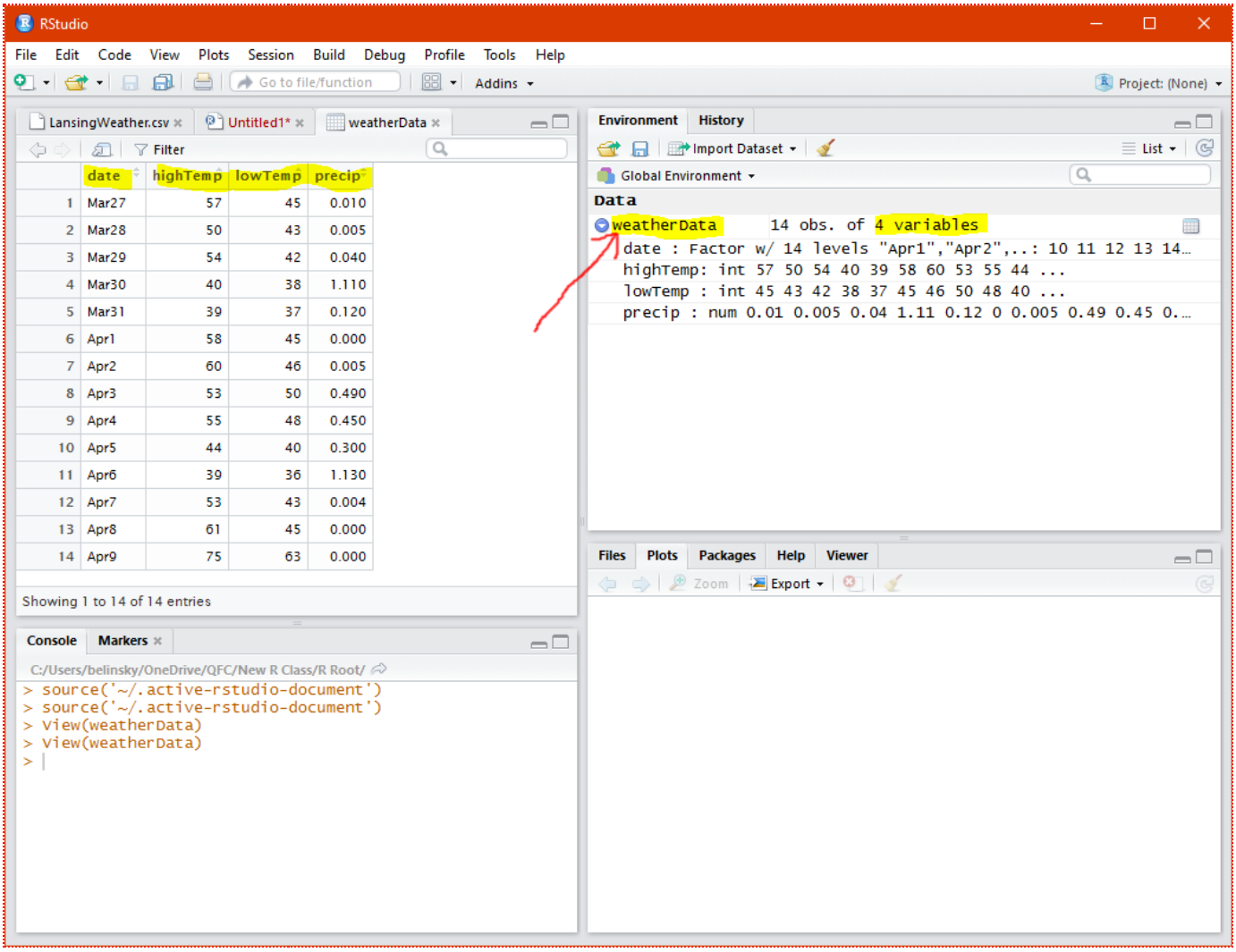

*Fig 2: Viewing the full **weatherData** data frame in RStudio*

4 - Getting values from the data frame

In *Fig.2*, the script shows that ***weatherData*** has "14 obs. of 4 variables". "variables" here refers to the *column headers* in the data frame -- *this is not the same as a programming variable.*

Data frames, in general, have columns that consist of related data collected from an experiment. Each column in a data frame consists of a *specific type of data* (e.g., date, high temp, low temp, precipitation) and the rows are *instances of each data type* (e.g., day1, day2...)

4.1 - The subset operator

Data frames are two dimensional objects -- every value in the Data Frame can referenced by two values: a column and a row. In R, we can extract data from a data frame using a subset operator ([ ]). The subset operator allows us to index the data frame using the rows and columns. The simplest way to use the subset operator is to extract one value from the data frame by the value's row and column placement.

```
1 dataPoint = weatherData[rowNumber, colNumber];
```

where ***dataPoint*** is the name of the variable that will store the value from the ***weatherData*** DataFrame

rowNumber is the row that the data point is on
colNumber is the column that the data point is on

So, if we want to get the ***highTemp*** (column 2) for March 31st (row 5), the code is:

```
dataPoint = weatherData[5, 2];
```

Adding this to the earlier script gives this:

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

  weatherData = read.csv("data/LansingWeather2.csv");
  dataPoint = weatherData[5,2];
}
```

The output we see in the Environment Window is "39L" (*Fig.3*). The "L" stands for "Long Integer" and refers to how the number is stored in memory -- for our sake, the "L" can be ignored. If we check the table (*Fig.2*), we see that 39 is indeed the high temperature on March 31st.

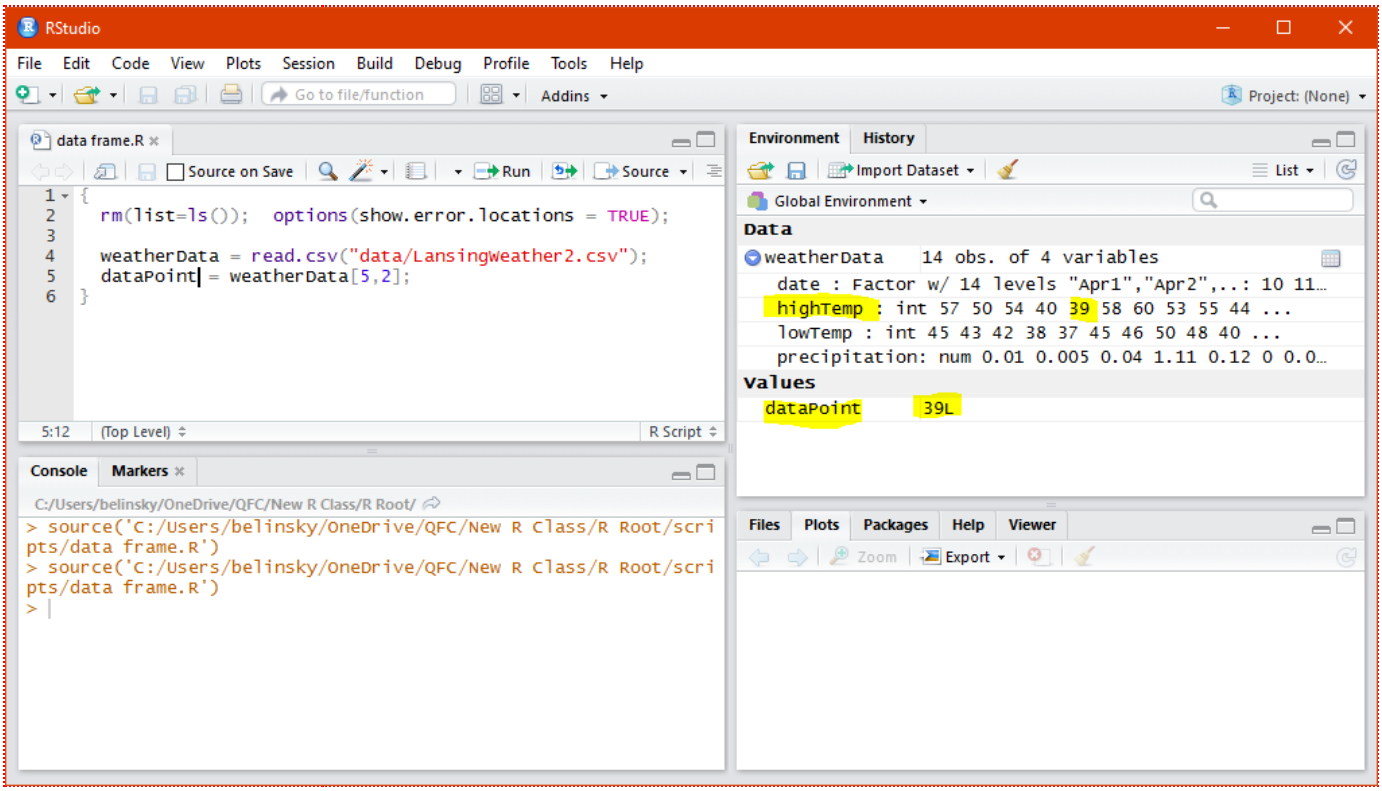

Fig 3: Subset a data frame to get a specific value

We can assign this data point to a variable and (like we did last unit) can perform a conditional operation on it:

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

```

```
  weatherData = read.csv("data/LansingWeather2.csv");
  dataPoint = weatherData[5,2];

  if(dataPoint < 40)
  {
    cat("It was cold that day!");
  }
}
```

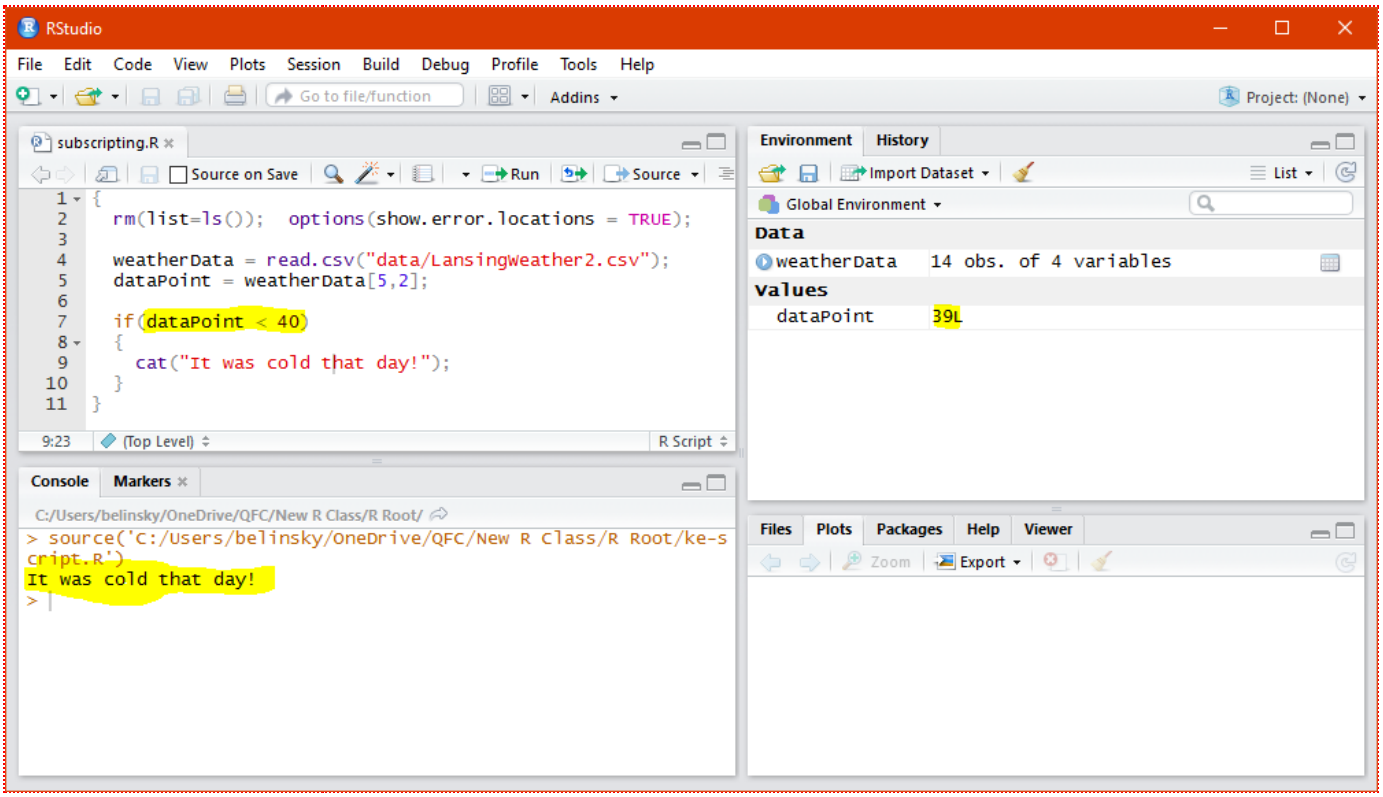

*Fig 5: Using **if** statement on individual values from a data frame*

Extension: Data vs Values in the Environment Window

4.2 - Subset by name

We can also use the column headers to subset the data frame

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

  weatherData = read.csv("data/LansingWeather2.csv");
  dataPoint1 = weatherData[5, "highTemp"]; # 5th value from highTemp column
  dataPoint2 = weatherData[7, "lowTemp"];  # 7th value from lowTemp column

}
```

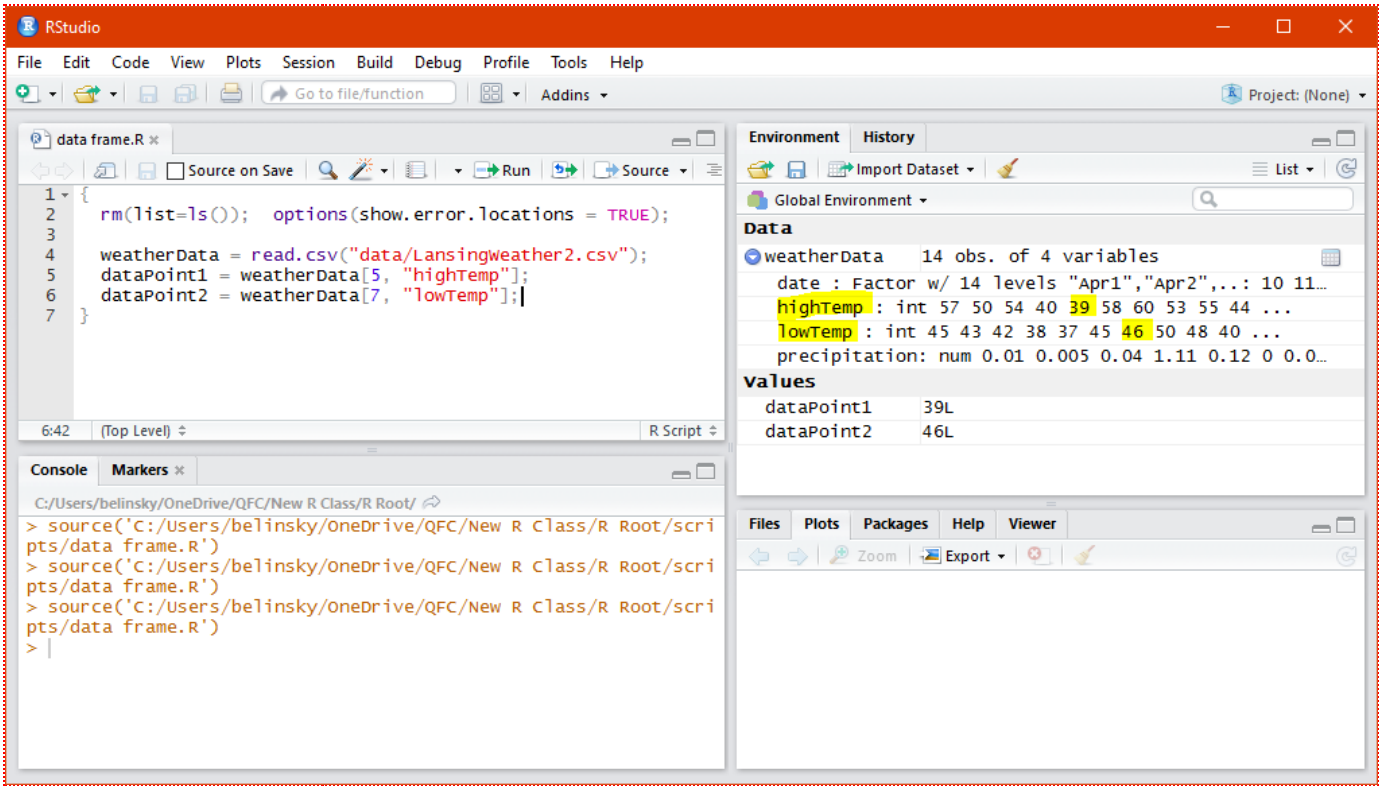

Fig 6: Subset a data frame using the column header

Notice that ***highTemp*** and ***lowTemp*** are in quotes. This is because ***highTemp*** and ***lowTemp*** are values *not variables*. If you take out the quotes you will get the error: "*Object highTemp not found*"

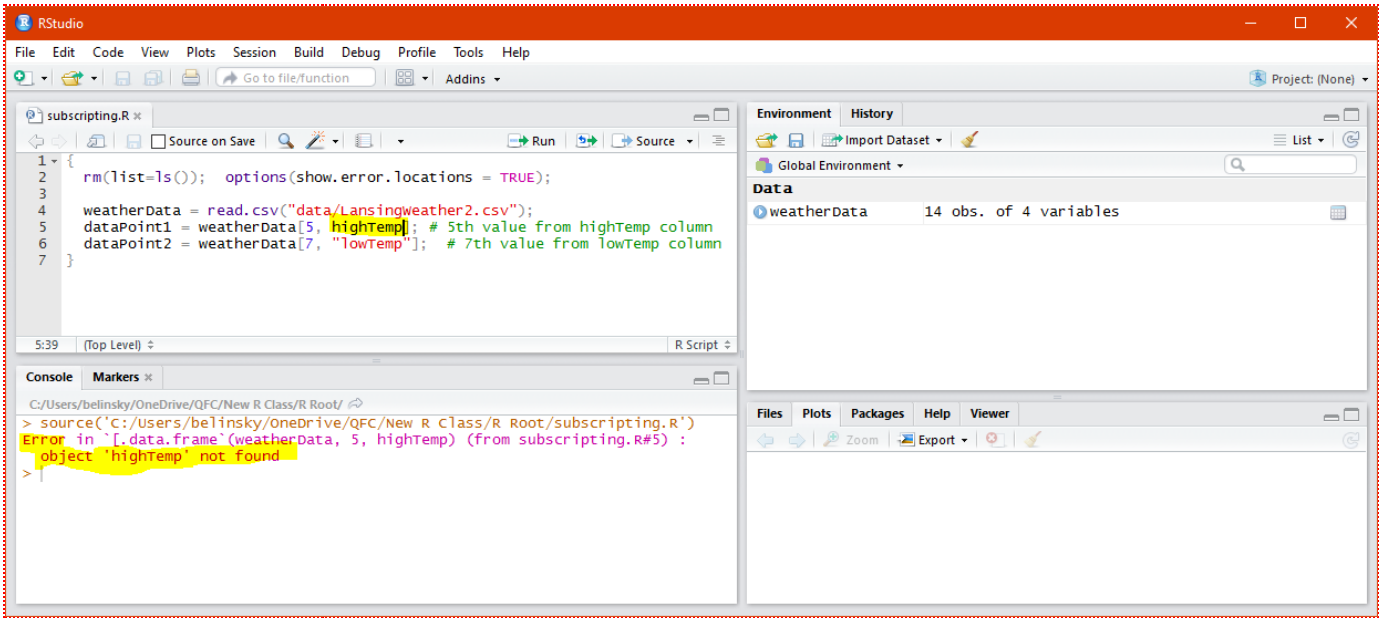

*Fig 7: Error: did not put **highTemp** in quotes*

5 - Retrieving a whole column of data using subset operator ([ ])

We can also use the subset operator (***[ ]***) to grab a whole column of data and save those data to a variable. Remember the subset operator takes two parameters: ***row*** and ***col***.

```
1 dataPoints = weatherData[rowNumber, colNumber];
```

The ***highTemp*** data is in column 2 of ***weatherData*** so we want the values *from every row* in column 2.

To return the value *of every row from a column*, we leave the *row number blank* and just put in a column number.

```
highTempData = weatherData[ , 2];           # column 2 is the "highTemp" data
```

Or we could use the column header instead of the column number:

```
highTempData = weatherData[ , "highTemp"];  # this the equivalent to the above line
```

This code accesses the ***highTemp*** data using the *column number* and the ***lowTemp*** data using the *column header*:

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

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

  # get all highTemps and save to highTempData
  highTempData = weatherData[ ,2];        # same as weatherData[ ,"highTemp"]
  # get all lowTemps and save to highTempData
  lowTempData = weatherData[ ,"lowTemp"]; # same as weatherData[ ,3]
}
```

highTempData and ***lowTempData*** are both shown in the Environment Window prefaced by ***int [1:14]*** (*Fig.8*). This means that there are **14** values associated with both ***highTempData*** and ***lowTempData*** and all the values are **integers**.

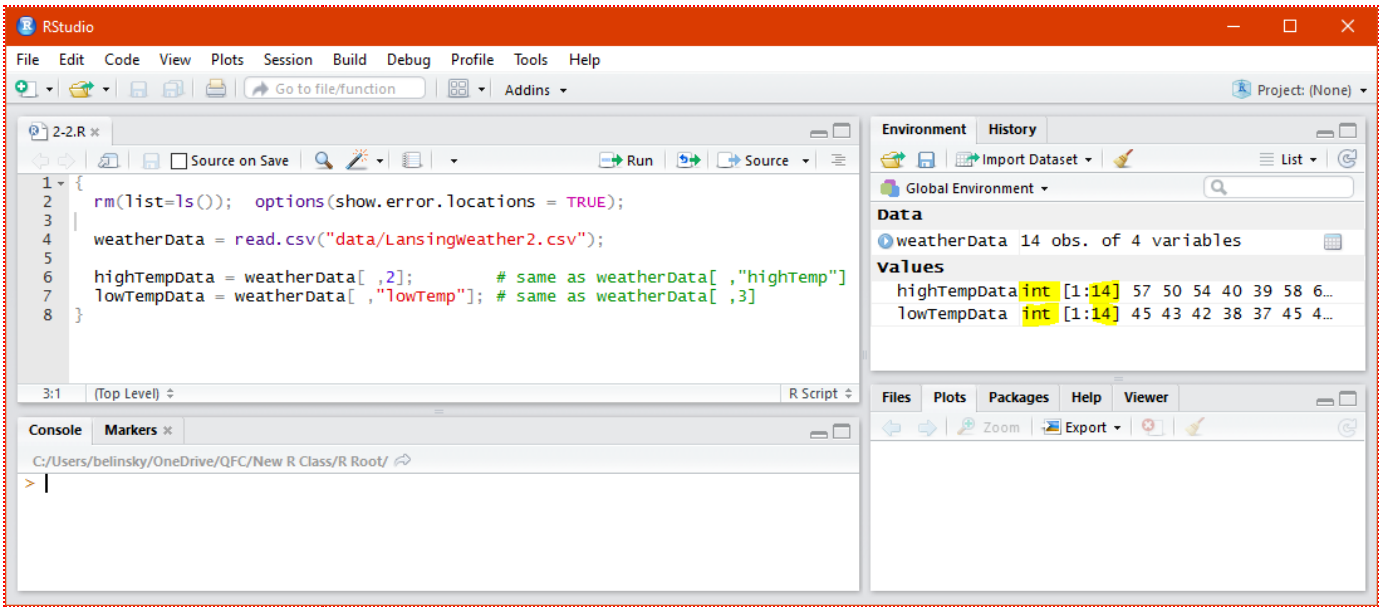

Fig 8: Saving values to a vector

highTempData and ***lowTempData*** are both vectors. *Vectors, in R, are variables that hold multiple values* -- in this case, multiple temperature readings.

5.1 - Accessing specific values in a vector

We can use the subset operator (***[ ]***) to access specific values within vectors. The difference from accessing values in a data frame is that the vectors (***highTempData***, ***lowTempData***) are one-dimensional objects whereas data frames like ***weatherData*** are two-dimensional. In other words, *only one index position is needed to subset a vector*, whereas two index positions are needed to subset a data frame. The value used to subset a vector is often called an *index value*.

```
highTempValue = highTempData[index];
```

The following code returns the ***3rd*** value in ***highTemps*** and the ***9th*** value in ***lowTemps***

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

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

  highTempData = weatherData[ ,"highTemp"];  # same as weatherData[ ,2]
  lowTempData = weatherData[ ,3];            # same as weatherData[ ,"lowTemp"]

  cat("Third value of highTemps: ", highTempData[3]);
  cat("\nNinth value of lowTemps: ", lowTempData[9]);
}
```

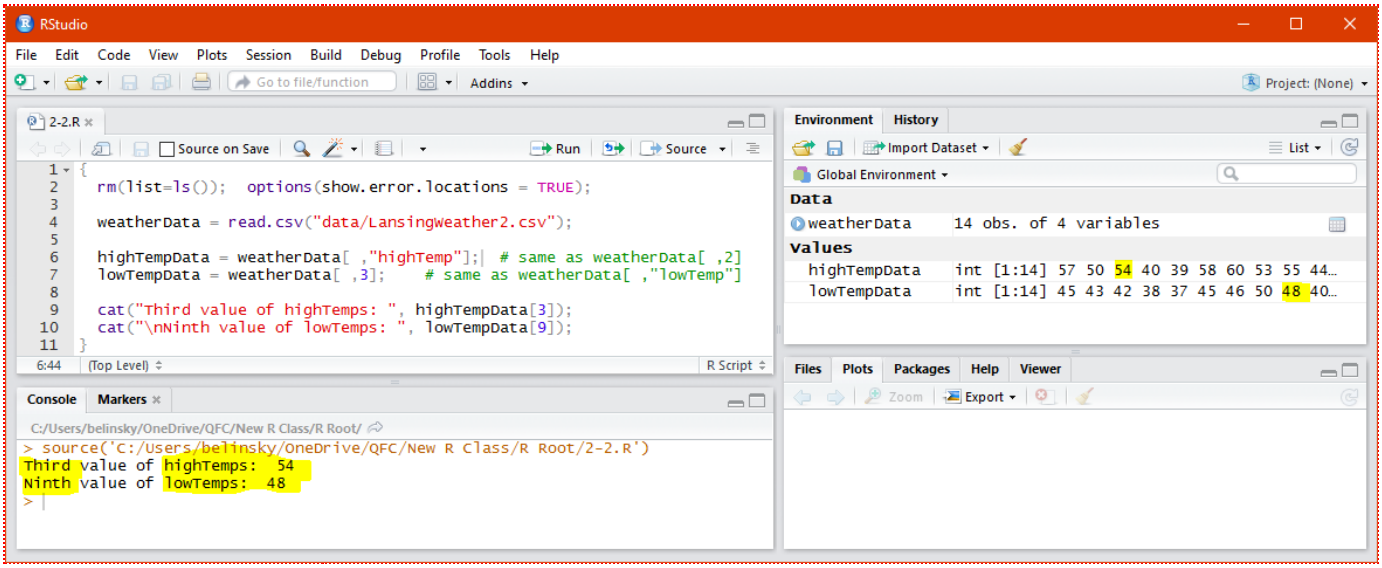

Fig 9: Subset (index) vectors to get individual values

highTempData[3] is the 3rd value in the variable ***highTempData***
lowTempData[9] is the 9th value in the variable ***lowTempData***

More formally, a *vector is a variable that has multiple values that are of the same type* (e.g., integers or characters).
Both ***highTempData*** and ***lowTempData*** are ***vectors*** because they are one-dimensional objects that hold multiple values of the same type.

6 - Application

If you have any questions regarding this application or your Class Project, feel free to email them to the instructor here.
You can attach the whole Root Folder as a zipped file. Instructions for zipping the Root Folder are here.

A) Save the data from LansingWeather2.csv to a data frame object

B) *From the data frame* you just created*:* output to the Console Window the ***precipitation*** for the 3rd day and the 10th day
Note: referencing a single value in a data frame requires two reference points because a data frame is two-dimensional

C) Save the ***precipitation*** column from the data frame to a vector named ***precipData***

D) *From the vector you just created in part C,* ***precipData***: output to the Console Window the ***precipitation*** for the 5th day and the 13th day
Note: referencing a single value in a vector requires only one point because a vector is one-dimensional

E) Challenge: calculate the temperature difference (highTemp - lowTemp) for the 4th day and the 6th day -- save the answer to variables.

Save you script file as ***app2-2.r*** *in the* ***scripts*** *folder of your RStudio Project for the class.*

7 - Extension: Data vs Values in the Environment Window

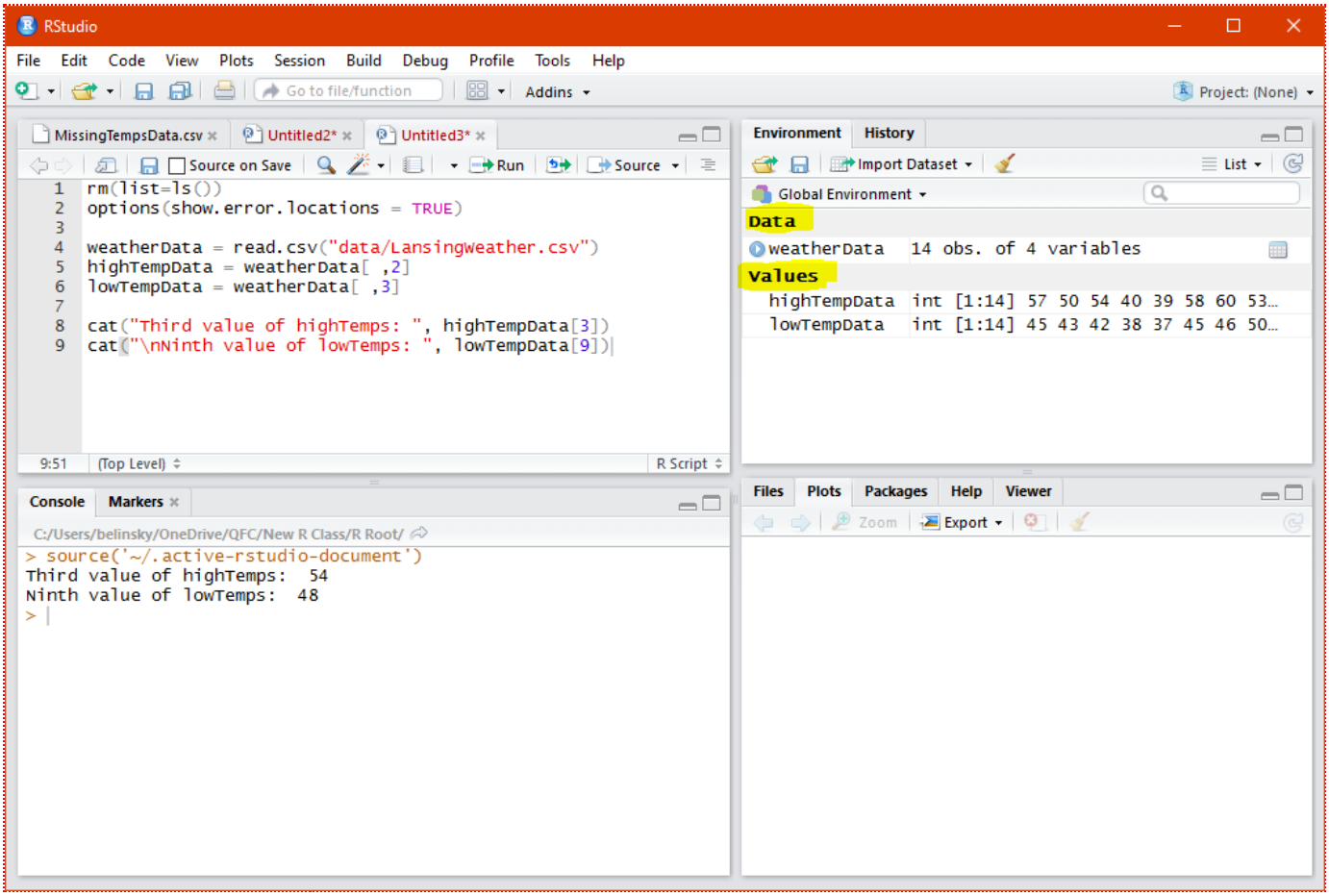

*Fig 10: **Data** and **Values** in the RStudio Environment Window*

The RStudio Environment window displays all the variables in a script and RStudio separates those variables into two categories: ***Data*** and ***Values*** (*Fig.10*). The functional difference between ***Data*** and ***Values*** is that any variable under ***Data*** can be double-clicked and viewed in the Main Window.

The technical difference between ***Data*** and ***Values*** is that ***Data*** contains all multi-dimensional variables (e.g., data frames), whereas ***Values*** contains all one-dimensional variables (e.g., vectors) and point variables. I would argue that this is not intuitive and that you would be better off ignoring the distinction between ***Data*** and ***Values*** -- the Environment Window is the only place where you will see it.