03-07 Conditional Subsets

1 - Purpose

- create conditional statements to evaluate the values in a vector
- use Regular Expression to find patterns in string variables
- get the index of vector values that meet a condition
- use indexing to subset another vector
- plot values from a subset index

2 - Concepts

3 - A conditional command for vectors: which()

In the last lesson we found no significant relationship between barometric pressure and high temperature for the year 2016. However, there might be a conflating variable in the data that is hiding a relationship (e.g., humidity). We want to do a temperature vs barometric pressure linear regression factoring in other conditions like humidity.

What this means is that we only want to look at temperature vs. pressure on days where humidity meets a certain condition (i.e., a subset).

3.1 - Subsetting with for() and if()

Our first task is to find which humidity values meet a condition. We could do this using a ***for()*** with an embedded ***if()***:

```
{
  rm(list=ls()); options(show.error.locations = TRUE);
  humidity = c(12,67,34,88,49,40);

  humiditySubset = c();


  # Go through each value in the humidity vector
  for(i in 1:length(humidity))
  {
     if(humidity[i] > 50) # if the value is greater than 50
     {
       humiditySubset = c(humiditySubset, i); # add i to the vector
     }
  }
}
```

humiditySubset has the values **2** and **4**. This means the **2nd** and **4th** values of ***humidity*** met the condition of being *greater than 50.*

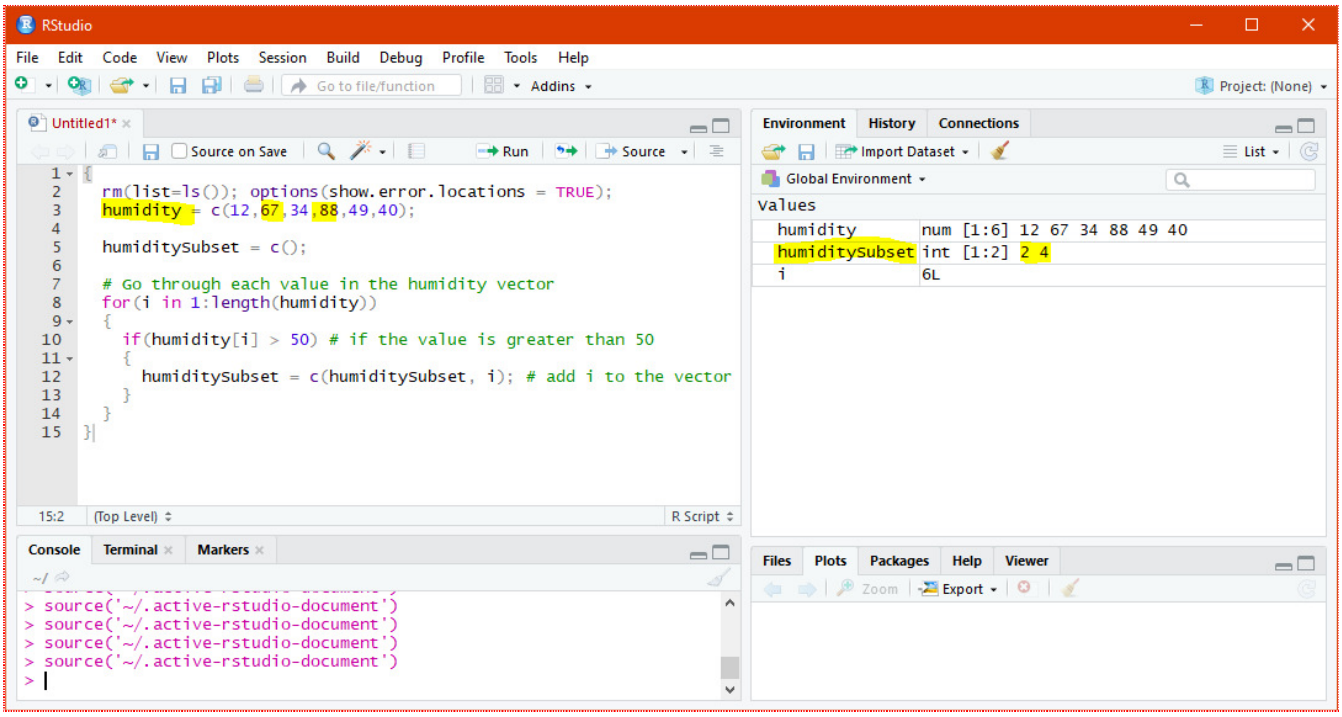

*Fig 1: Getting a subset vector of indices using **for()** and **if()**. We will describe how to use these indices in section 6.*

3.2 - Subsetting with which()

Or, the subset can be done much more succinctly with a ***which():***

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

  humidity = c(12,67,34,88,49,40);

  humiditySubset = which(humidity > 50);
}
```

humiditySubset once again has the values **2** and **4.**

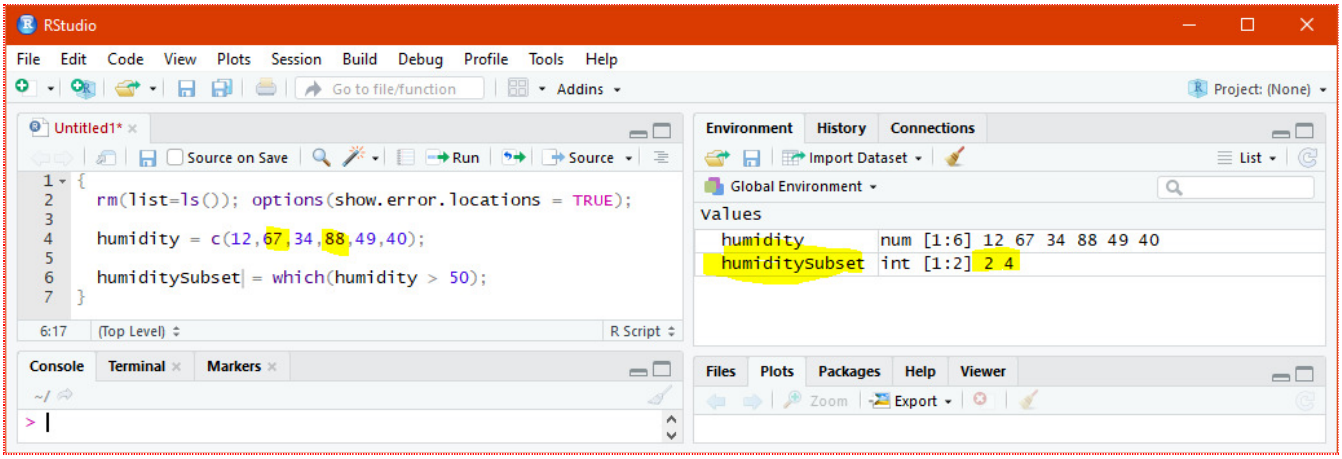

*Fig 2: Getting a subset vector using **which()**.*

For the rest of this lesson, we will focus on using ***which()*** to create subset vectors. The script for the rest of this lesson is located here.

4 - which() -- conditional statements for vectors

which() answers the question: Which of the values in this vector meet a specified condition? And the answer to the ***which()*** is presented as a *vector of index values* representing the values in the vector where the condition is ***TRUE***.

For instance the question could be:

1) Which of the following vector values *c(-2, 3, **7**, 0, **10**, **9**)* are greater than 4?
   Answer: ***c(3, 5, 6)***
   In other words, the *3rd, 5th, and 6th* values (7, 10, 9) in the vector are greater than 4.
2) Which of the following vector values *c("Swiss", **"Muenster"**, "American", **"Muenster"**)* are equal to "Muenster"?
   Answer: ***c(2, 4)***
   In other words, the *2nd* and *4th* value are equal to "Muenster".

The following script has two vectors (***ages, animals***) and four ***which()*** conditions that check the two vectors.

```
ages = c(2,7,3,9,6,3,5);
animals = c("llama", "alpaca", "goat", "llama", "guanaco");

index1 = which(ages > 4);  # holds index of ages greater than 4
index2 = which(ages < 4);  # holds index of ages less than 4

index3 = which(animals == "llama"); # holds index of animals that are "llama"
index4 = which(animals != "llama"); # holds index of animals that are not "llama"
```

which() outputs a vector of indexed values that meet the condition in parenthesis. This vector is saved to the variables ***index1, index2, index3, and index4.***

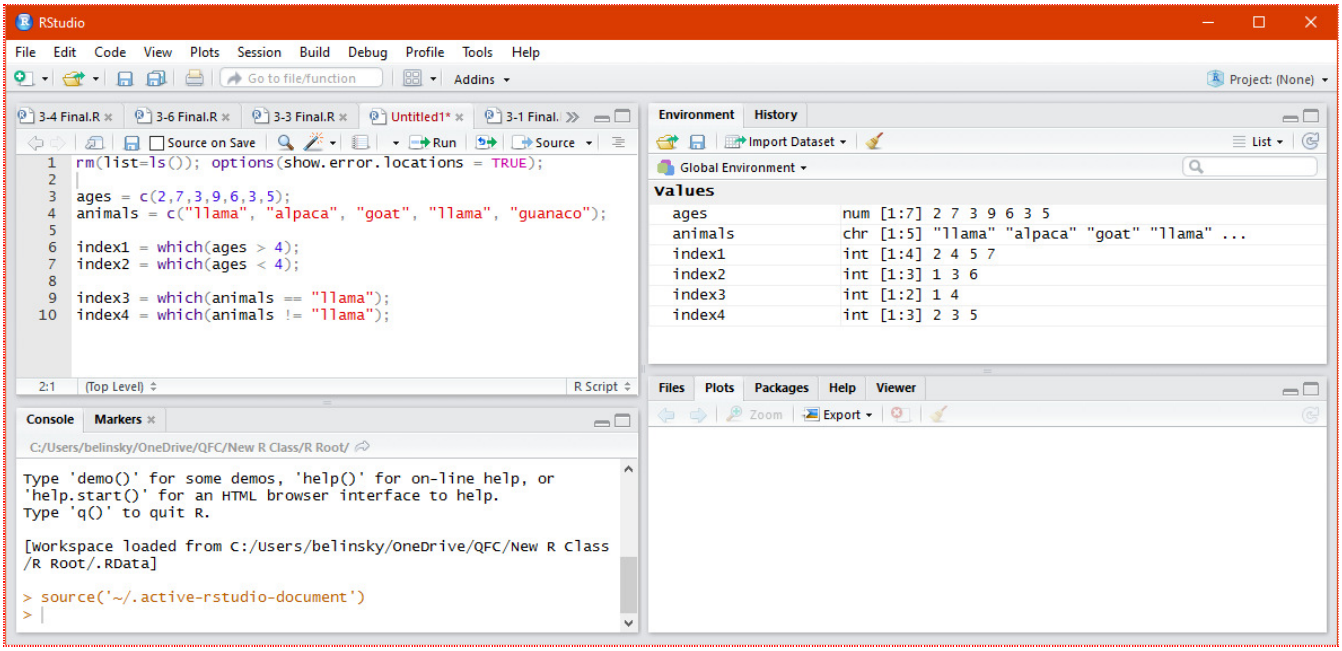

*Fig 3: Using **which()** to check all values in a vector against a condition*

Extension: The TRUE/FALSE vector

5 - Using which() on weather data

For the rest of this lesson, we are going to use the reformatted Lansing weather data frame from last lesson.

```
lansing2016Weather = read.csv(file="data/LansingNOAA2016Formatted.csv",
                              stringsAsFactors = FALSE);
```

And we are going to save the weather data columns to vectors.

```
date = lansing2016Weather[,"date"];
eventData = lansing2016Weather[,"eventData"];
avgTemp = lansing2016Weather[,"avgTemp"];
tempDept = lansing2016Weather[,"tempDept"];
precipitation = lansing2016Weather[,"precipitation"];
humidity = lansing2016Weather[,"humidity"];
barometer = lansing2016Weather[,"barometer"];
dewPoint = lansing2016Weather[,"dewPoint"];
avgWind = lansing2016Weather[,"avgWind"];
maxWind = lansing2016Weather[,"maxWind"];
windDirection = lansing2016Weather[,"windDirection"];
sunrise = lansing2016Weather[,"sunrise"];
sunset = lansing2016Weather[,"sunset"];
```

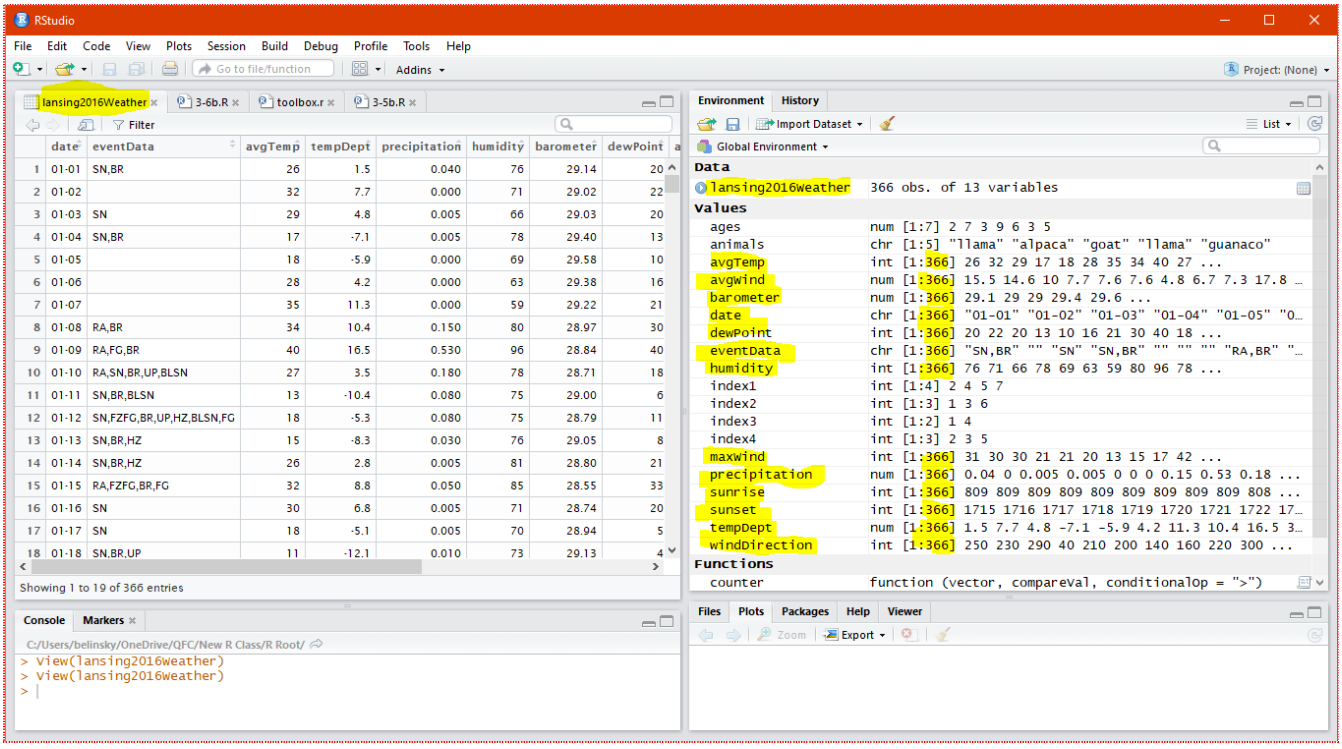

*Fig 4: The **13** vectors from the Lansing weather data frame.*

Extension: Using data frame columns instead of vector

5.1 - Identifying indices in weather data.

The vector ***eventData*** contains information about the type of weather that occurred during that day (e.g., Fog, Snow, Rain, etc.). If we want to find out which days were snowy we could use a ***for()*** to iterate through all 366 values and check each value to see if it is equal to snow ("SN").

First we need to create a vector that will hold all the index values that contain "SN". Note: we do not know at this point how long this vector is going to be.

```
snowyDays1 = c();
```

Then we iterate through all 366 values in ***eventData*** and use a conditional statement to see if each indexed value is equal to "SN"

```
for(i in 1:366)
{
  if(eventData[i] == "SN")
  {
    snowyDays1 = c(snowOnlyDays1, i);
  }
}
```

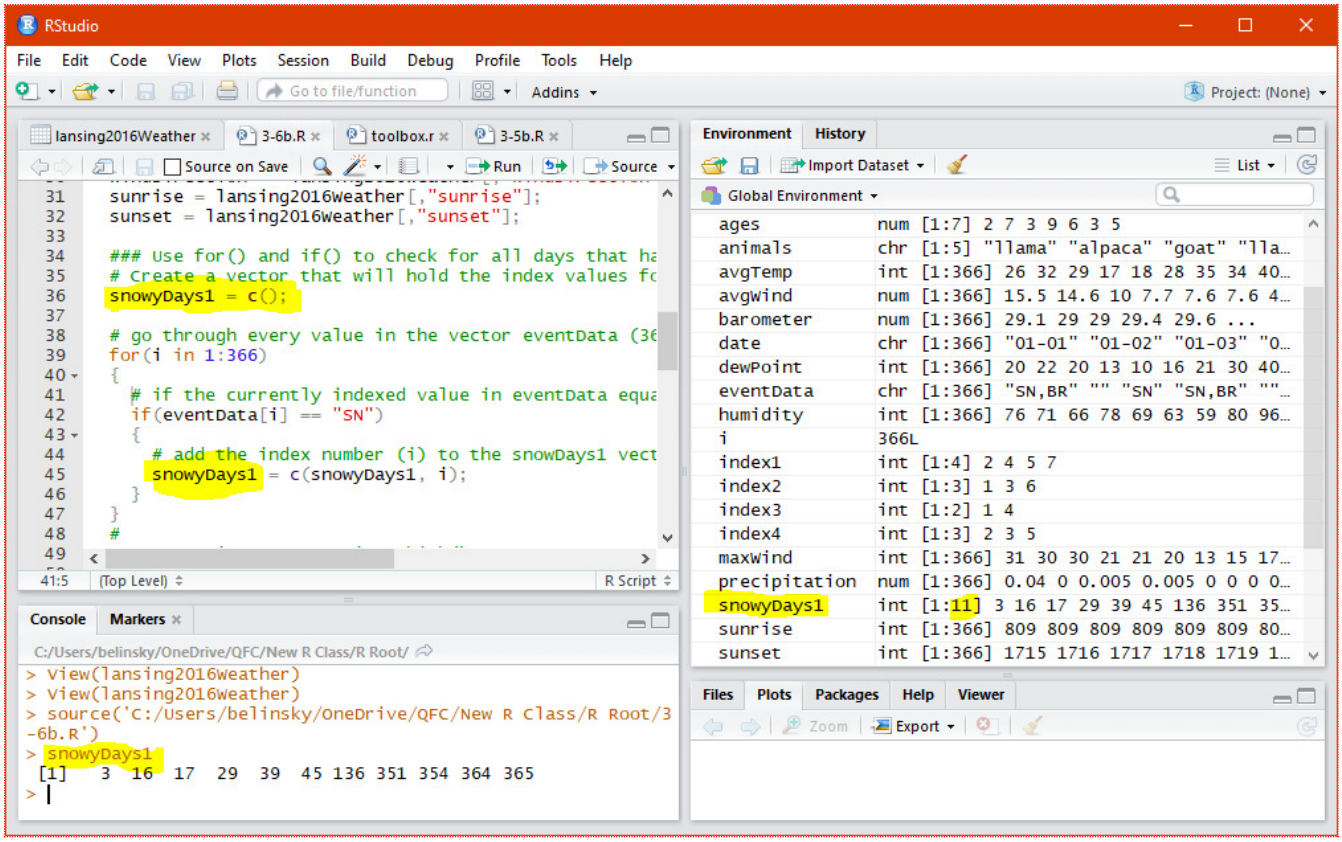

*Fig 5: Finding, and indexing, snowy days using **for()** and **if()***

At the end of the script, the vector ***snowyDays1*** holds the indexes for the values in ***eventData*** that are equal to "SN", of which there are 11.

5.2 - Identifying values using which()

Or, we could get the values in ***eventData*** that are equal to "SN" using ***which()***:

```
snowyDays2 = which(eventData == "SN");
```

The above code will:

1) Iterate through each value in the vector ***eventData***
2) Check if the value is equal to "SN"
3) Create a vector that has the indices of all the values in the vector equal to "SN"
4) Save that vector to ***snowyDays2***

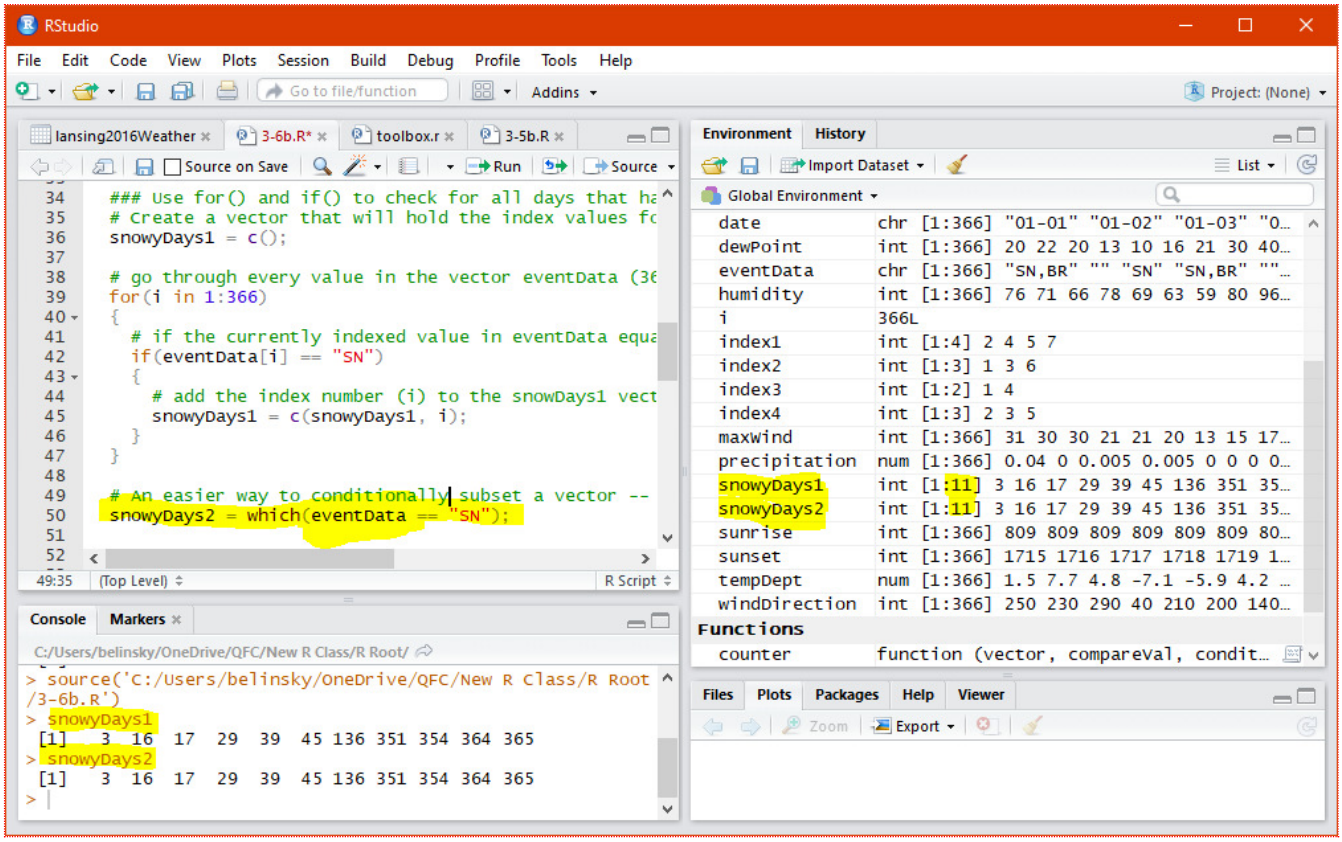

*Fig 6: Finding, and indexing, snowy days using **which()***

Looking at either ***snowyDays1*** or ***snowyDays2***, we see that there were 11 days that had snow -- and the days are identical.

6 - Using the indexed value generated by which()

Our next step is to use the vector of indexed values (***snowyDays2***) to index other vectors like ***avgTemp*** and ***avgWind***.

Let's find the average temperature and wind speed on days that it was snowy

```
1  snowyDayTemps = avgTemp[snowyDays2];
2  snowyDayWinds = avgWind[snowyDays2];
```

since ***snowyDays2*** = ***c(3, 16, 17, 29, 39, 45, 136, 351, 354, 364, 365)***

avgTemp[***snowyDays2***] is equivalent to ***avgTemp***[***c(3, 16, 17, 29, 39, 45, 136, 351, 354, 364, 365)***];
This gives the temperature on the 11 days indexed in the ***snowyDays2*** vector

avgWind[***snowyDays2***] is equivalent to ***maxWind***[***c(3, 16, 17, 29, 39, 45, 136, 351, 354, 364, 365)***];
This gives the winds on the 11 days indexed in the ***snowyDays2*** vector

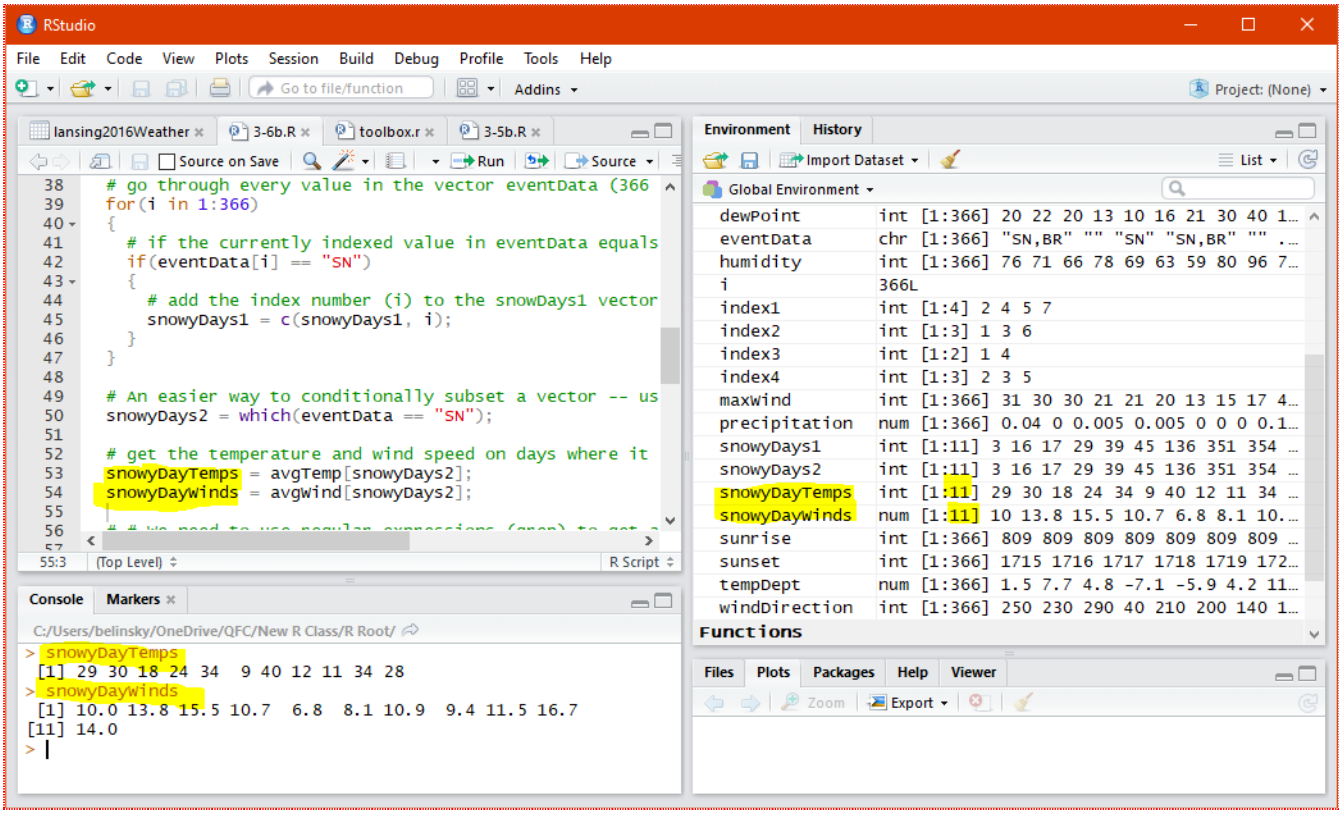

Fig 7: Subsetting the temperature and wind vectors using the snowy day index vector

So the average temperatures on the 11 snowy days were: **(29, 30, 18, 24, 34, 9, 40, 12, 11, 34, 28)** and the average wind on the 11 snowy days were: **(10.0, 13.8, 15.5, 10.7, 6.8, 8.1, 10.9, 9.4, 11.5, 16.7, 14.0)**

7 - Finding patterns within strings

But I lived in Lansing and I guarantee you that there were more than 11 days last year with snow so something is wrong here...

If we take a look at the ***eventData*** vector in the Console Window (by typing "eventData"), we can see that "SN" is in many more than 11 values. But, ***eventData*** gives every weather condition that happened on those days. On the 3rd and 16th day, there was just snow but on the first day there was both snow (SN) and mist (BR) and on the 14th day there was snow, mist, and haze (HZ).

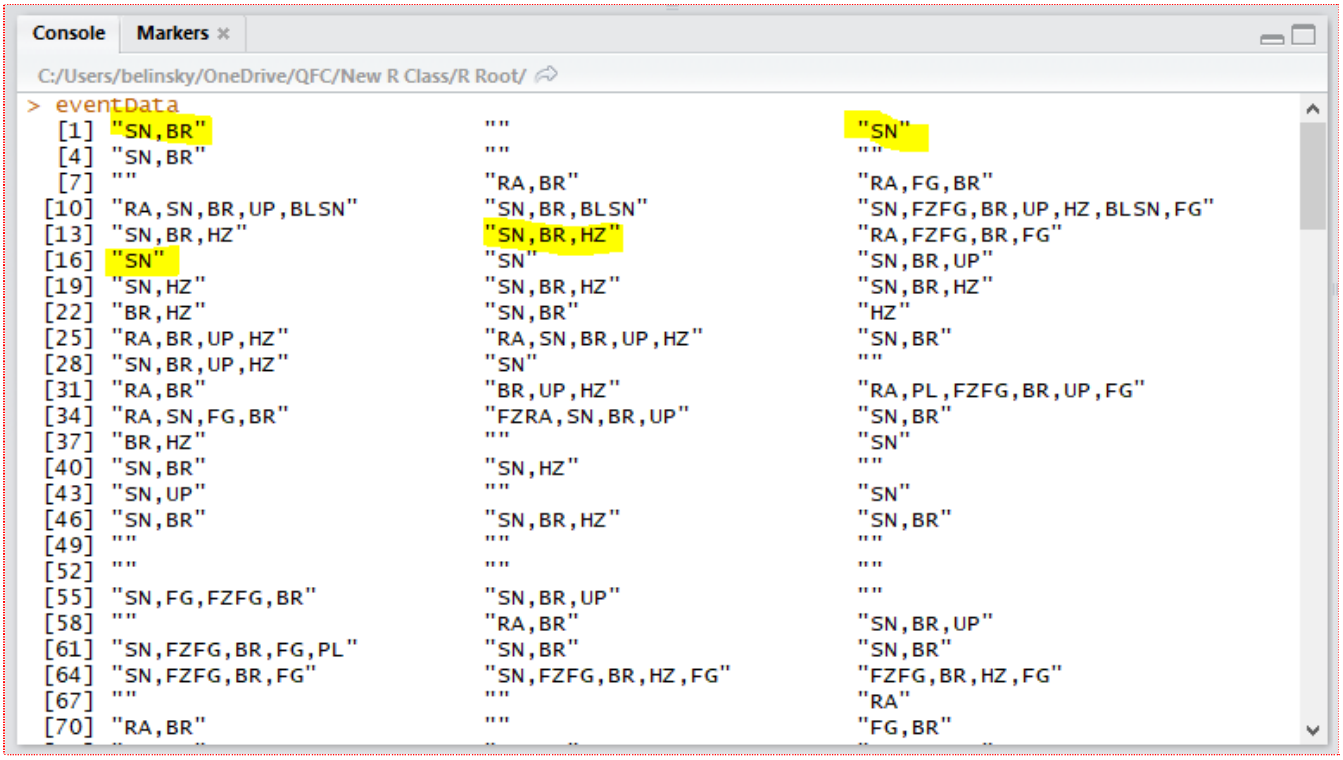

*Fig 8: The **eventData** vector.*

The conditional statement we used with ***which()*** only checks for values that are *exclusively "SN"*. We really want *every value that contains "SN"*. This cannot be easily done using ***which()*** but it can be done easily using ***grep(). grep()*** is a powerful tool that allows one to find patterns of all sorts of complexity within a string. We are going to use ***grep()*** to find the substring ("SN") contained within a vector of string values (***eventData***).

```
1  snowyDays3 = grep("SN", eventData);
```

grep(), like ***which()***, produces a vector of the index values that meets the condition (i.e., index of which ***eventData*** values contain "SN").

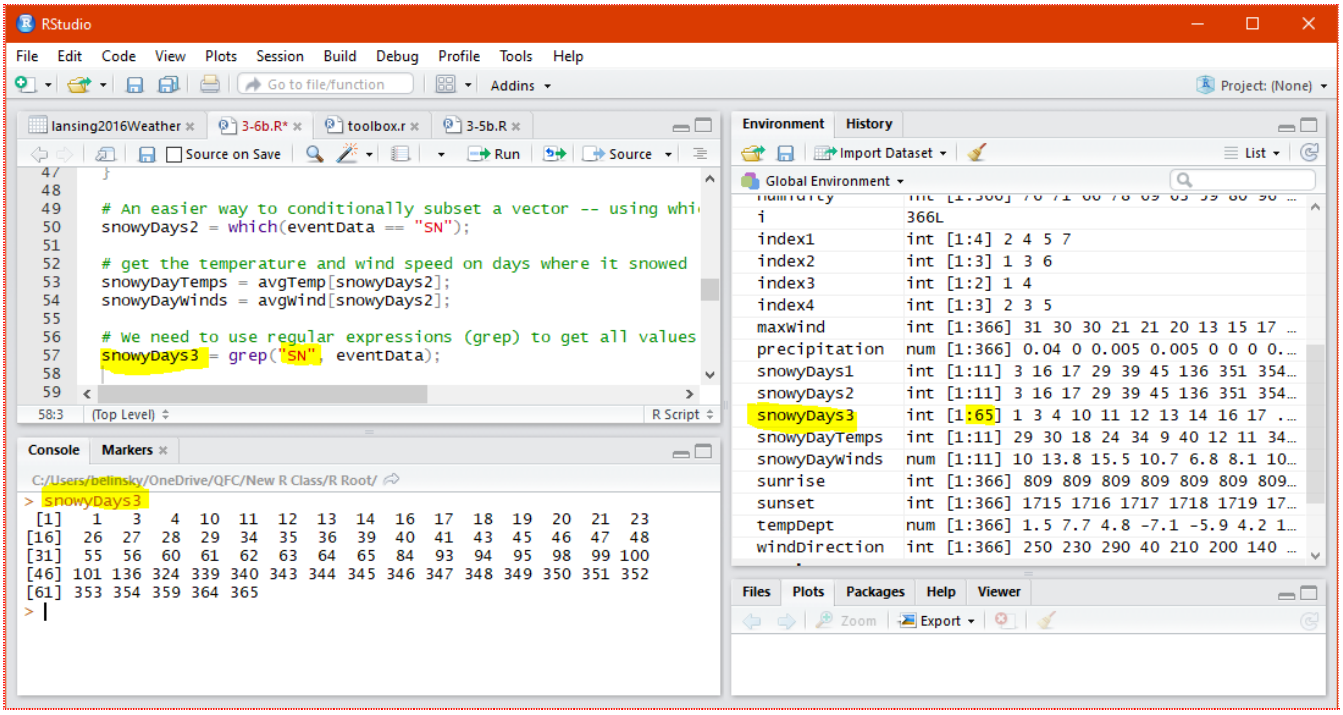

*Fig 9: Using **grep()** to find all days that had snow.*

7.1 - A quick grep() example

grep() is a very powerful tool for finding patterns within strings (the process is known as Regular Expressions, which could be a class on its own). We are only using the most basic functionality in ***grep()*** which is finding a substring within a string.

So if we have the vector:

```
testVector = c("one fish", "two fish", "one llama", "two llama",
               "red fish", "blue fish")
```

grep() return the indices of the vectors value that contain the substring specified

```
test1 = grep("two", testVector);   # test1 equals 2,4
test2 = grep("lla", testVector);   # test2 equals 3,4
test3 = grep("fish", testVector);  # test3 equals 1,2,5,6
```

7.2 - Subsetting vectors

snowyDays3 is a vector with 65 indexed values, representing the 65 days in Lansing that any snow fell. In the Console Window (*Fig 10*), we use ***snowyDays3*** to index the vectors ***avgTemp*** and ***avgWind***. This gives us the temperatures and the winds for the 65 days that it snowed.

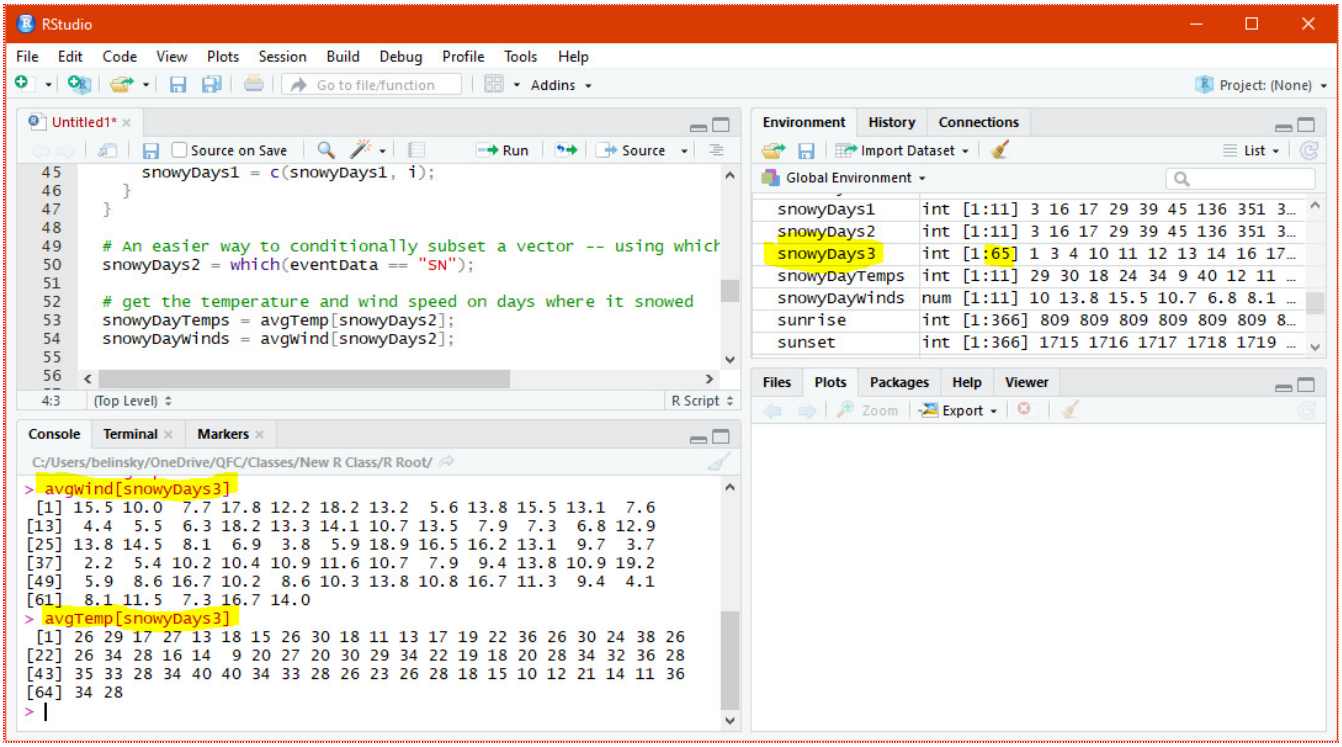

*Fig 10: **avgTemp** and **avgWind** values subsetted by **snowyDays3**.*

7.3 - Statistics on subsetted vectors

Maybe we want to know some statistical information like the minimum temperature of the 65 snow days or the mean wind speed of the 65 days. We can use the ***min()*** and ***mean()*** functions on the subsetted data.

```
1  snowDaysMinTemp = min(x=avgTemp[snowyDays3]);
2  snowDaysMeanWind = mean(x=avgWind[snowyDays3]);
```

Running this code shows that the minimum average temperature on snowy days was **9** (degrees Fahrenheit) and the mean average wind on snowy days was **10.9** (miles/hour).

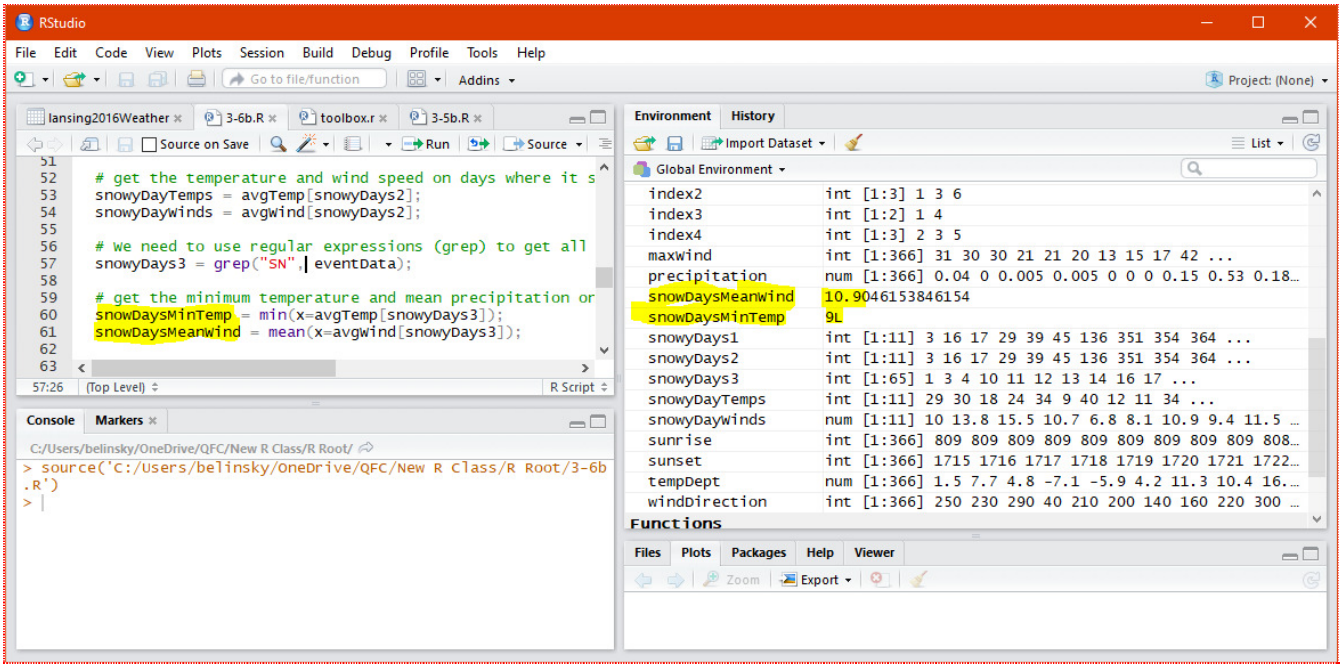

*Fig 11: Using subsetted vectors in functions **min()** and **mean()**.*

8 - Plotting subsetted vectors

We are going to produce two plots:
1) humidity vs average temperature on days without precipitation (***precipitation*** == **0.00**)
2) humidity vs average temperatures on days with a lot of precipitation (***precipitation*** > **1.00**).

First we need to create vectors that hold the days that have no precipitation and those that have hard rain.

Note: it is good practice to put in the decimal (**1.00** as opposed to **1**) to indicate that it is a real number -- the number just happens to have a value the looks like an integer.

```
1 daysNoPrecip = which(precipitation == 0.00);   # no precipitation days
2 daysHighPrecip = which(precipitation > 1.00);  # high precipitation days
```

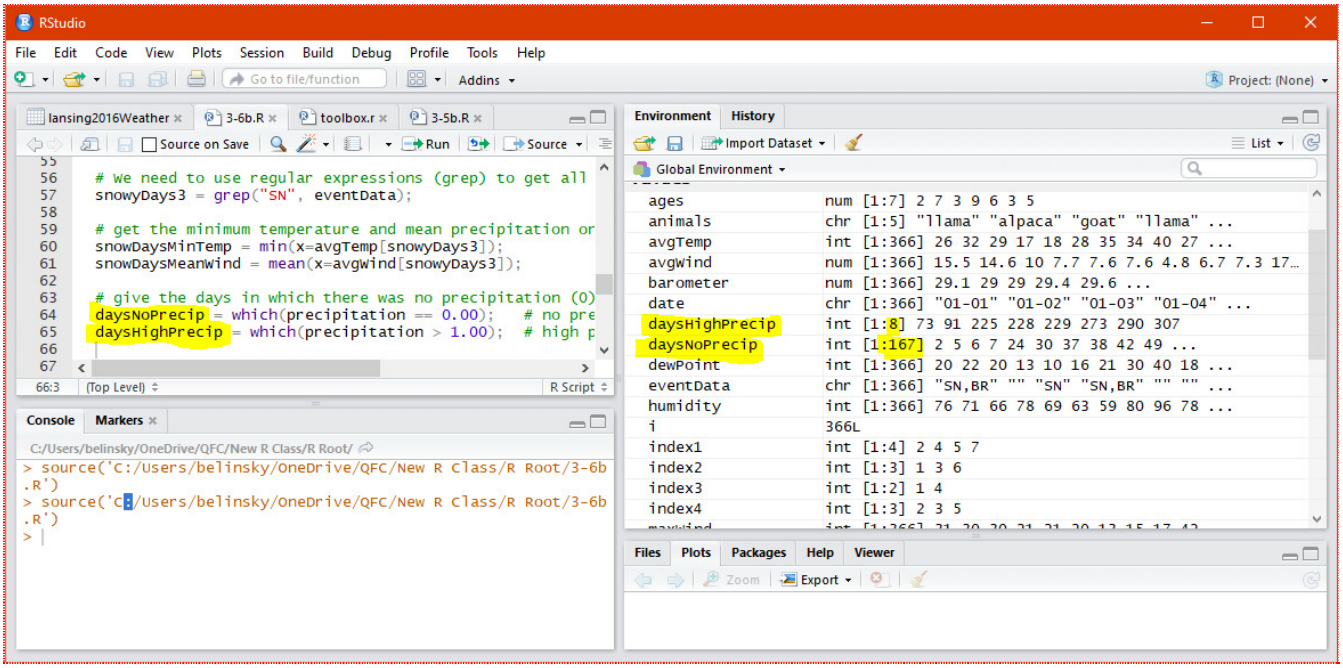

Fig 12: Creating vector indices of days based on precipitation amounts

We see in the above script that there were **167** days without any precipitation and **8** days with precipitation more than 1 inch.

Next we are going to plot a subset of ***avgTemp*** and ***humidity*** -- the subset being the **167** days where there was no precipitation**.**

```
1  plot(x=avgTemp[daysNoPrecip], y=humidity[daysNoPrecip]);
```

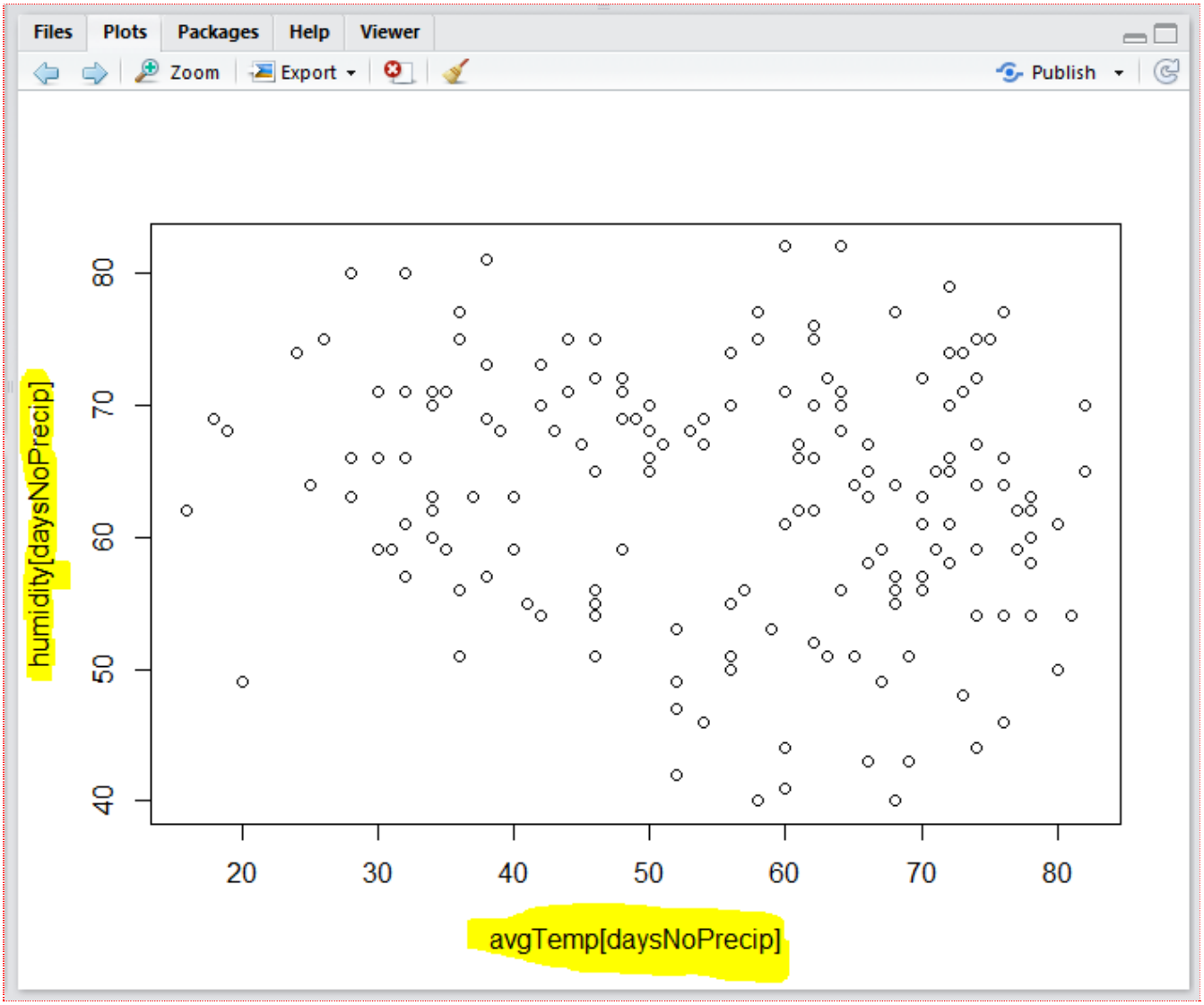

*Fig 13: Plot of **avgTemp** vs **humidity** on days with no precipitation*

Another plot of ***avgTemp*** and ***humidity*** -- this time we are only plotting the 6 days that had 1 inch or more of precipitation

```
1  plot(x=avgTemp[daysHighPrecip], y=humidity[daysHighPrecip]);
```

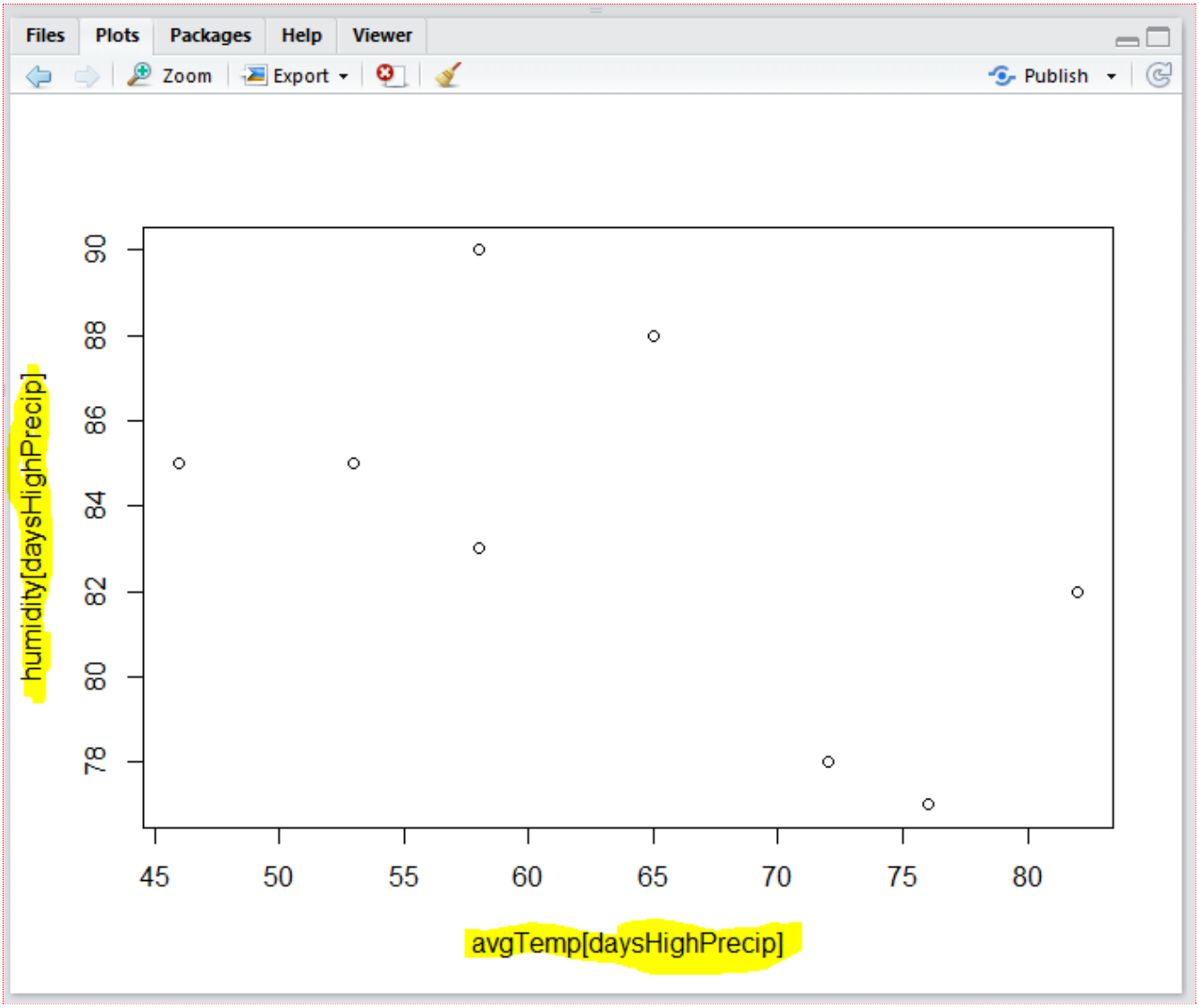

*Fig 14: Plot of **avgTemp** vs **humidity** on days with 1 inch or more of precipitation*

9 - Outputting values to the console

We can also output information about a vector as opposed to looking at it in the Environment Window. In this case we will print out to the Console Window the index for the days that had really strong winds (***maxWind*** > **45**).

```
1  windyDayIndex = which(maxWind > 45);
```

windyDayIndex is a vector with 7 values (representing the 7 days that had winds greater than **45mph**).

We can use the vector ***windyDayIndex*** as an index for the ***dates*** vector to get the actual dates that had really strong winds.

```
1  windyDays = dates[windyDayIndex];
2  print(windyDays);
```

So we see that ***2-19, 2-29, 3-16, 7-8, 7-12, 11-29,*** and ***12-26*** were the dates that had winds greater than **45mph**.

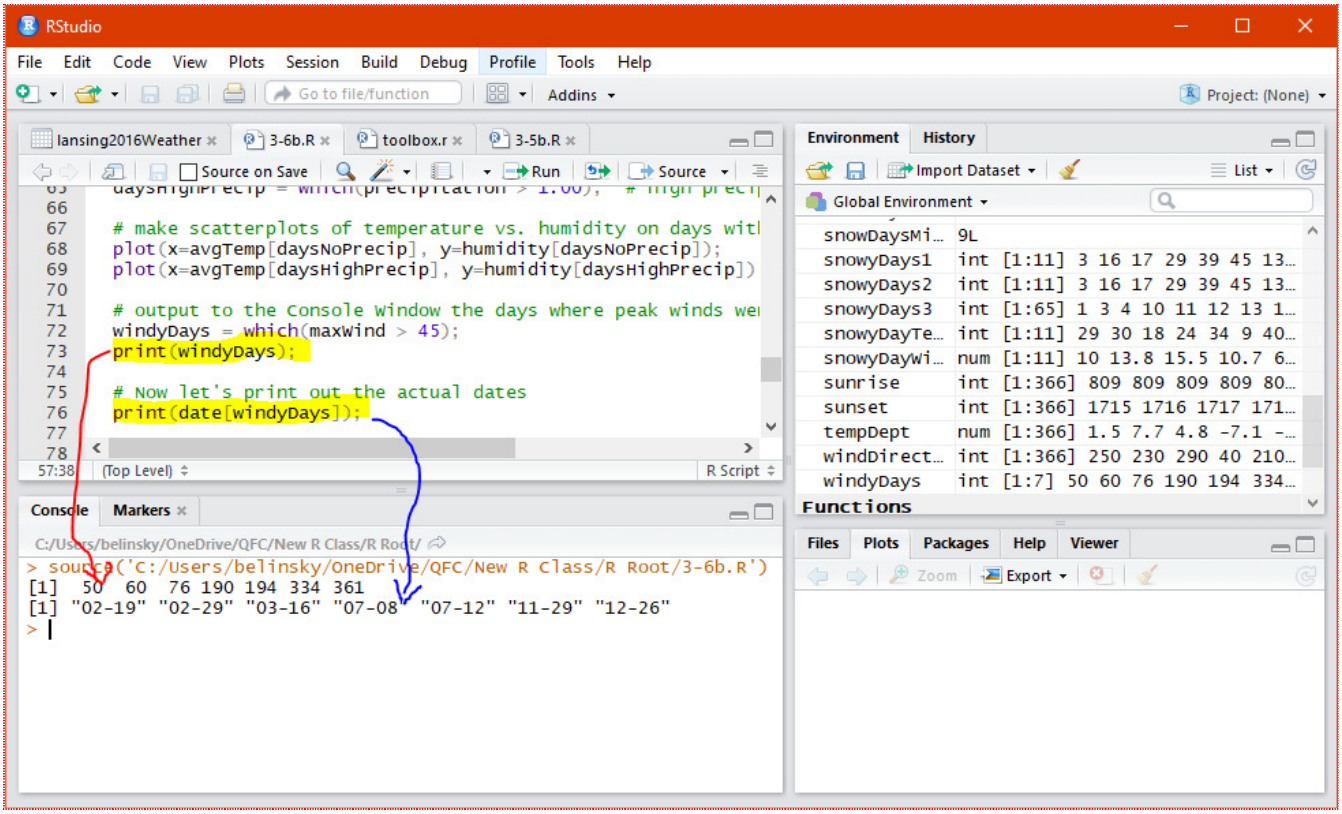

Fig 15: Printing out the subsetted windy dates to the Console Window

10 - Application

We will be using the data from ***LansingNOAA2016Formatted.csv***, created in the last lesson.

1) create a vector called ***hoursOfSun*** that is the ***sunset*** times minus the ***sunrise*** times (like last lesson, you might need to first convert ***sunset*** and ***sunrise*** times).

2) Find the average temperature for days that have less than 10 hours of sun (so, an average of the averages).

3) Find the average temperature for days that have more than 14 hours of sun.

4) Find the number of days the humidity was high (greater than 90%) and for these days output to the Console Window:
- the date
- the amount of precipitation
- the maximum wind speed

5) Find the dates that were hazy with other conditions (like snowy, windy, etc) and find the dates that were exclusively hazy (HZ).

11 - Extension: Data frame columns instead of vectors

The line of code:

```
1  humidity = lansing2016Weather[,"humidity"];
```

saves all the values from the column "humidity" in the data frame ***lansing2016Weather*** to the vector

humidity.

We then could use ***humidity*** to find which days had high humidity

```
highHumidity = which(humidity > 80);  # days with high humidity
```

But we could do the exact same thing with the column from the data frame

```
highHumidity = which(lansing2016Weather[,"humidity"] > 80);
```

While we can use both ***humidity*** and ***lansing2016Weather[,"humidity"]*** in the ***which()***, there are two advantages to using ***humidity***

1) The code is easier to read
2) We can modify ***humidity*** without modifying the data frame, ***lansing2016Weather***

12 - Extension: The TRUE/FALSE vector

Let's take another look at the ***which()*** code:

```
ages = c(2,7,3,9,6,3,5);
animals = c("llama", "alpaca", "goat", "llama", "guanaco");

index1 = which(ages > 4);  # holds index of ages greater than 4
index2 = which(ages < 4);  # holds index of ages less than 4

index3 = which(animals == "llama"); # holds index of animals that are "llama"
index4 = which(animals != "llama"); # holds index of animals that are not "llama"
```

There is an in between step that we skipped over here. Each ***which()*** has a conditional statement inside. That conditional statement gets evaluated by R first, and then ***which()*** is performed on the answer to the conditional statement. Using the Console Window, let's look at how the conditional statement get evaluated by R:

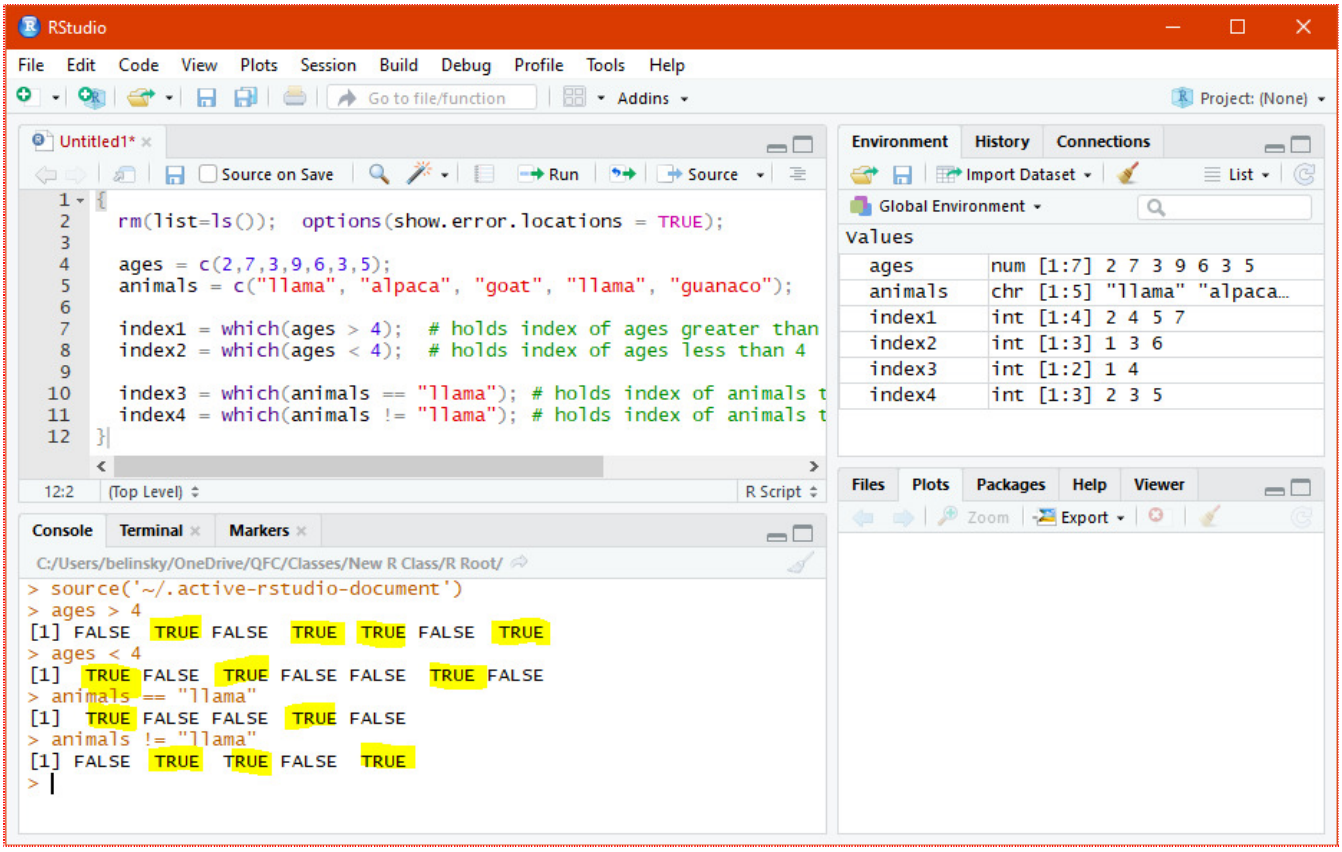

Fig 16: Looking at TRUE/FALSE vectors for conditional statements on a vector

Each conditional statement produces a vector that has the same length as the number of values in the vector (*Fig 16*). The vector has only two vales: **TRUE** and **FALSE**. The **TRUE** and **FALSE** values match the index of the values in the original index.

So if:

```
1  ages = c(2,7,3,9,6,3,5);
```

then:

***(ages* > 4)** is evaluated as: **c(FALSE, TRUE, FALSE, TRUE, TRUE, FALSE, TRUE)**

This indicates that the **2nd**, **4th**, **5th**, and **7th** values of the vector ***ages*** are greater than **4**

which() takes the **TRUE**/**FALSE** index and outputs the indices of the ***TRUE*** values

```
1  which(ages > 4);   # This evaluates to c(2, 4, 5, 7)
```