03-08 Complex Conditional Subsets

1 - Purpose

- index a vector using multiple conditions
- index a string vector using patterns in the string values
- subset a vector using the index vector
- revisit linear regression models and scatterplots

2 - Concepts

3 - Gathering data

The script for this lesson is located here.

Like the last lesson, we are going to use the reformatted Lansing weather data frame and we are going to treat all the columns with string values as strings -- instead of factors.

```
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"];
```

4 - which() to subset vectors

As we saw in the last lesson, we can find out which days had an average temperature greater than 80 using the ***which()*** function:

```
daysOver80 = which(avgTemp > 80);
```

which() produces a vector of index values (**163, 204, 224, 225, 250**) and this vector is saved to the variable

daysOver80. The five vector values indicate that there were **5 days** with average temperatures greater than 80 (the 163rd day, the 204th day...).

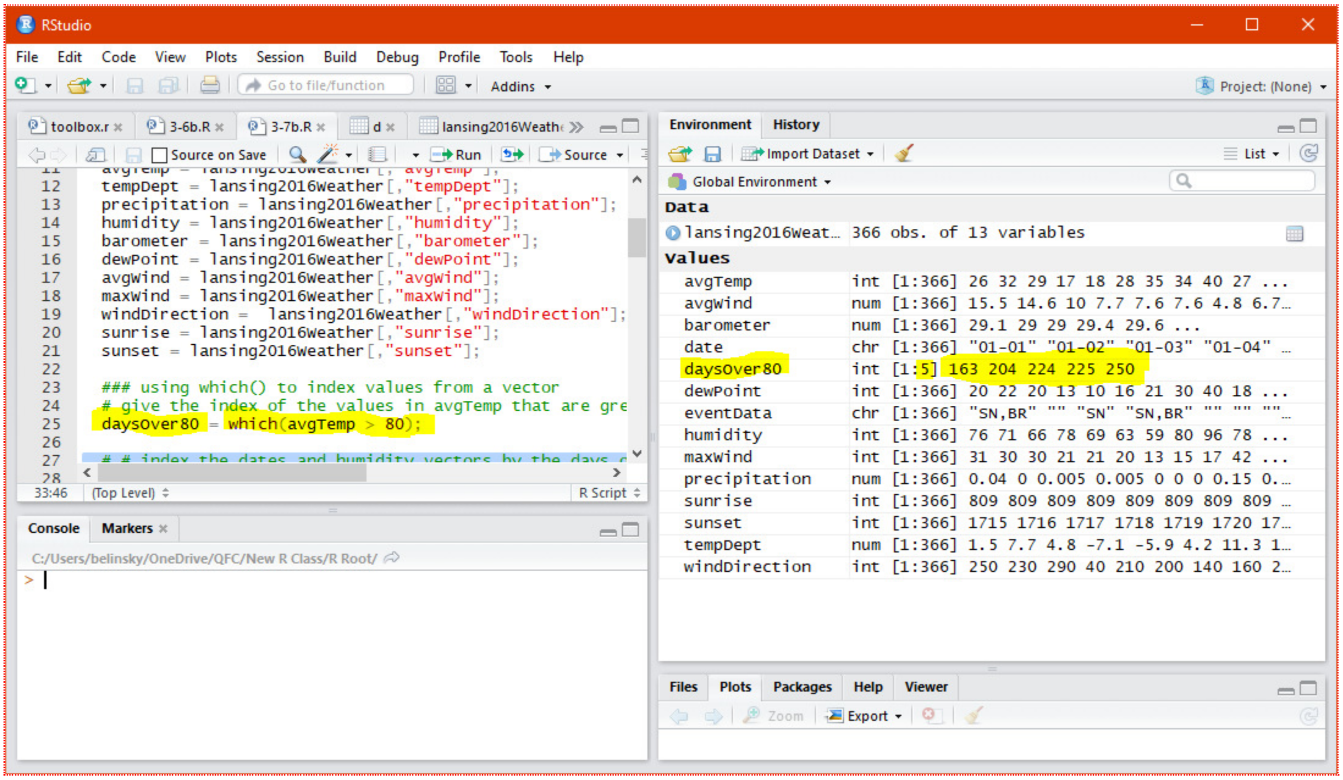

*Fig 1: Using **which()** to subset the vector **avgTemp**.*

4.1 - Indexing other vectors

daysOver80 is a vector of five index values with:

- ***daysOver80***[1] = 163
- ***daysOver80***[2] = 204
- ***daysOver80***[3] = 224
- ***daysOver80***[4] = 225
- ***daysOver80***[5] = 250

We can use the vector ***daysOver80*** to index other vectors and find out more information like:

What are the actual dates where the temperatures averaged over 80 degrees?

```
datesOver80 = dates[daysOver80];
```

answer: June 11, July 22, August 11, August 12, and September 6

Or, what was the humidity on the days the temperature averaged over 80 degrees?

```
humidityOver80 = humidity[daysOver80];
```

answer: 54%, 65%, 66%, 82%, and 70%

4.2 - Simple stats on subset vector

We could take this one step further and find the ***min()*** and ***max()*** humidity for the days the temperatures averaged over 80 degrees.

```
minHumOver80 = min(humidity[daysOver80]);
```

```
maxHumOver80 = max(humidity[daysOver80]);
```

answers: min=54%, max=82%

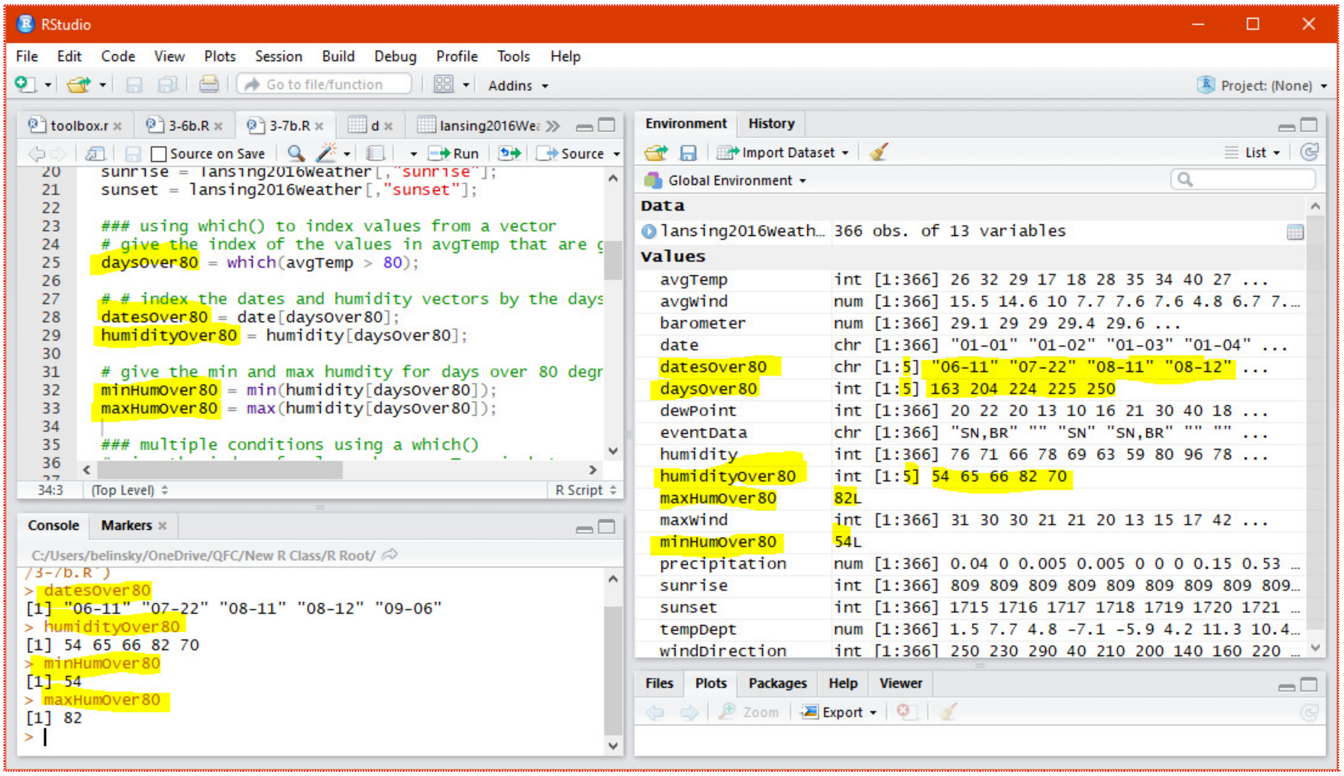

*Fig 2: Use the vector of index values (**daysOver80**) to subset other vectors (**dates** and **humidity**).*

4.3 - Creating index vector with multiple conditions (& and |)

Let's find the days that averaged temperatures in the **60s** (so, between **60** and **70**).

For *a single temperature value*, we could use an ***if()*** with the ***&&*** operator:

```
if(avgTemp > 60 && avgTemp < 70)
```

But we have a vector with multiple values. We can make a similar conditional statement using ***which()***, except we use a single ***&*** instead of the double ***&&***.
Extension: Difference between & and &&

To find temperatures between **60** and **70** using ***which()*** we look for ***avgTemp*** greater than **60** and less than **70.** As with the ***if()*** conditional statement, we need to explicitly give both conditions.

```
tempsIn60s = which(avgTemp > 60 & avgTemp < 70);
```

tempIn60s has 56 values (*Fig 3*), so there were 56 days that averaged 60 to 70 degrees. The earliest day was day **69** (early March) and the latest was day **306** (early November).

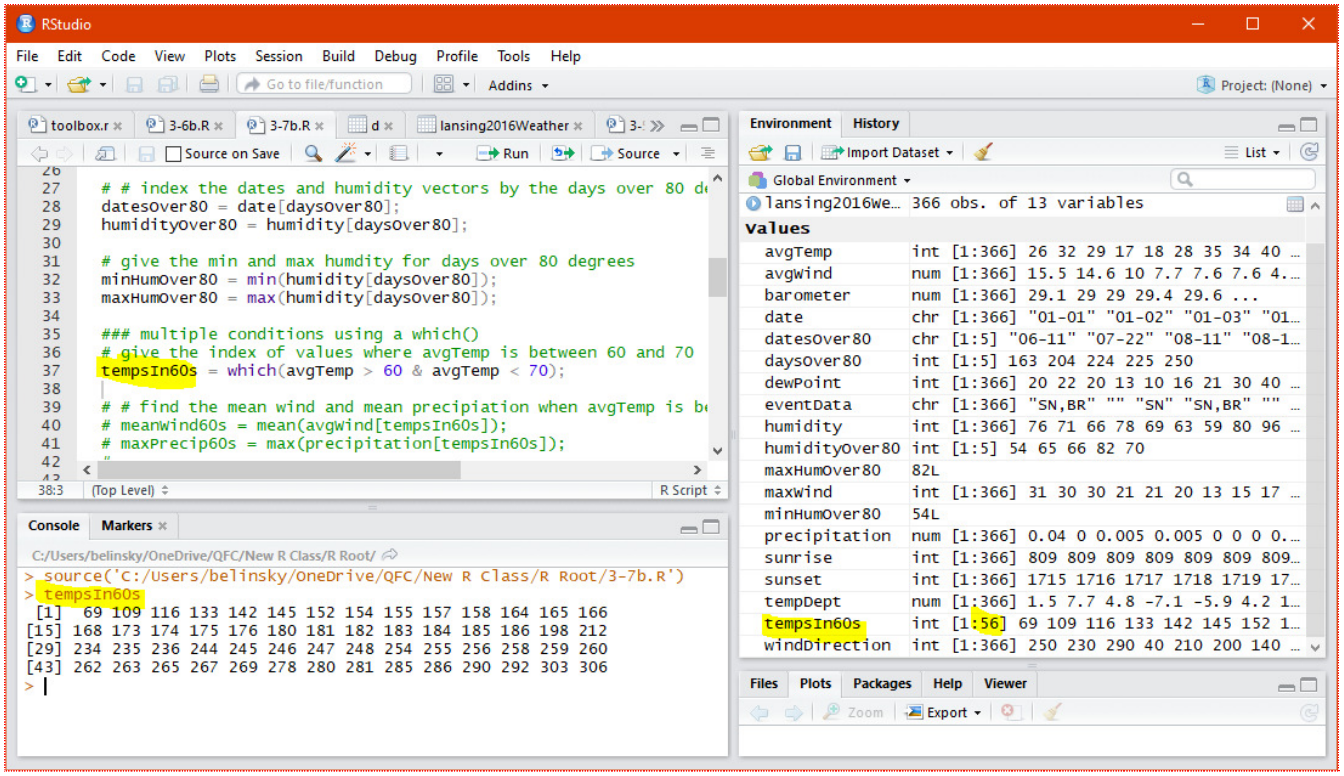

*Fig 3: Using the **&** operator in **which()**.*

Now we can use the vector ***tempsIn60s*** to find the average winds for the **56** days with temperature between **60** and **70:**

```
meanWind60s = mean(avgWind[tempsIn60s]);
```

or the maximum precipitation for the **56** days with high temperature between **60** and **70:**

```
maxPrecip60s = max(precipitation[tempsIn60s]);
```

We can see that (*Fig 4*):

- there were **56** days with ***avgTemp*** values in the **60s**
- the average wind speed *for those **56** days* was about **7.3mph**
- and the maximum precipitation for those **56** days was **1.01 inches**.

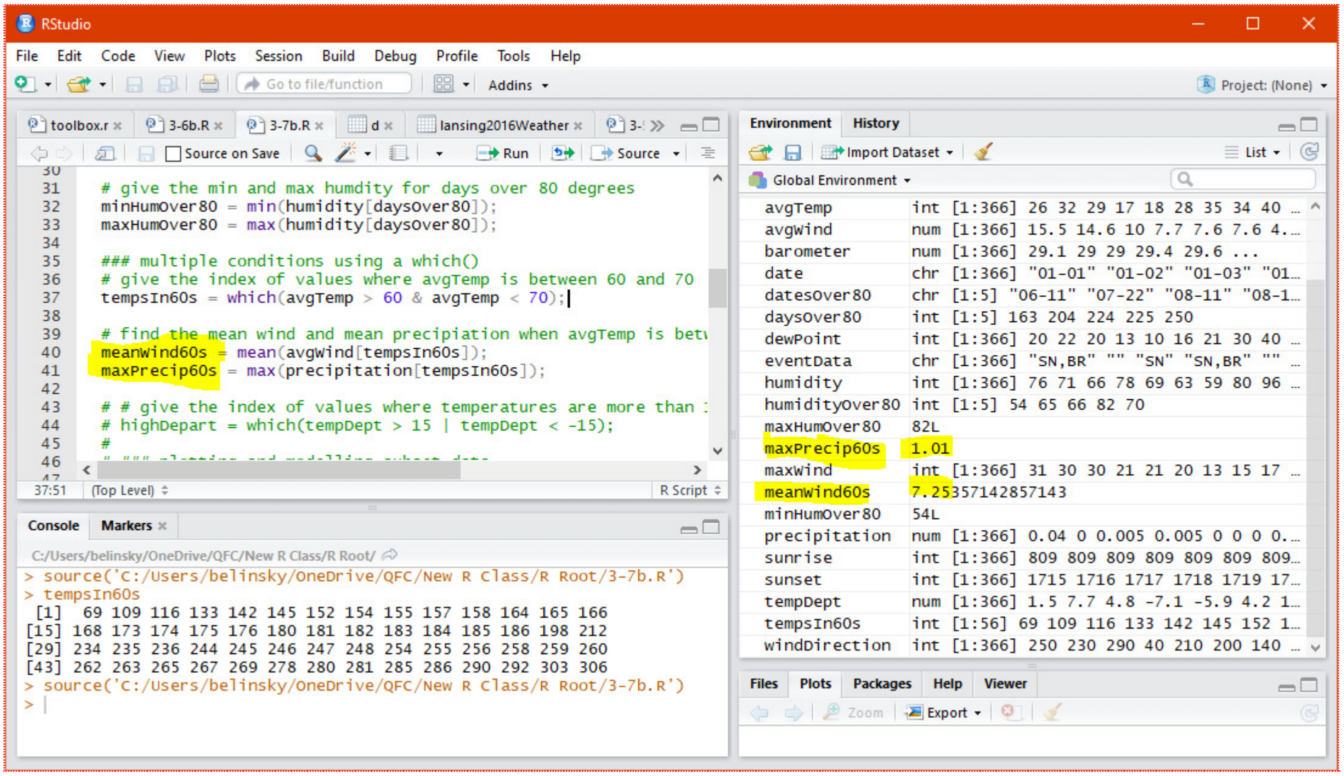

*Fig 4: Using the index vector (**tempsIn60s**) to subset the vectors **avgWind** and **precipitation**.*

5 - Linear models using subsetted vectors

In the last lesson we initially found that there was no relationship between **avgTemp** and **barometer** (pressure). In this lesson, we will try to find relationships by looking at ***avgTemp*** vs ***barometer*** while considering other factors, like departure from average temperature (**tempDept**).

tempDept measures the deviation between the actual temperature and the average temperature.

Let's create a vector that holds the indexes for the days that had more than a 15 degree departure from normal (in either the positive and negative direction). So we want ***tempDept*** that is *less than -15 or greater than 15*. This means we need to use the ***or*** (*|*) operator.

```
highDepartTemp = which(tempDept <  -15 | tempDept > 15);
```

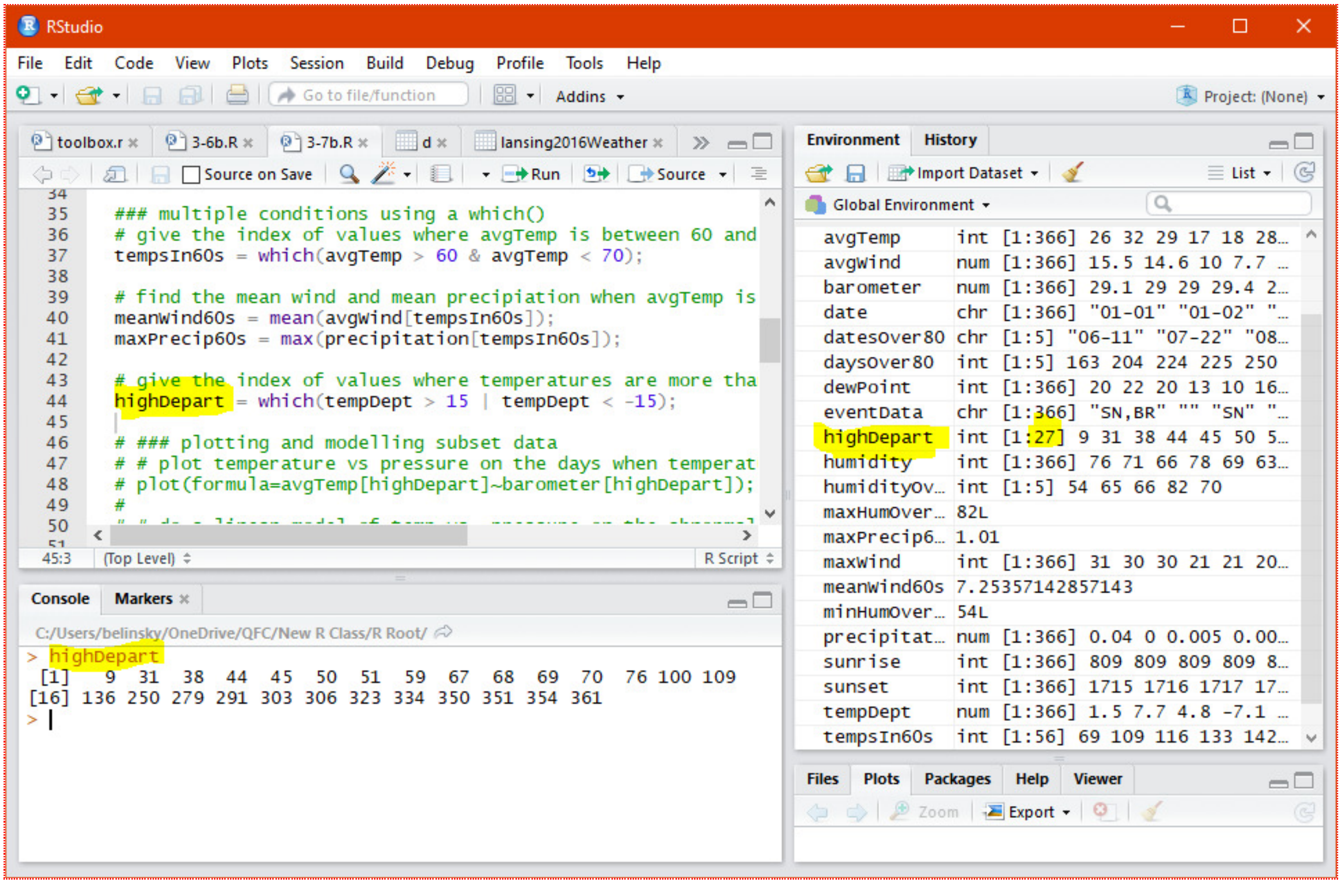

*Fig 5: Subsetting a vector using **which()** and the **or** (|) conditional operator.*

5.1 - Set up scatterplot and linear model

Next we make a scatterplot of ***avgTemp*** vs ***barometer*** on days with high departure temperatures (*Fig 6*):

```
plot(formula=avgTemp[highDepart]~barometer[highDepart]);
```

And make a linear model that indexes temperature and pressure on the days with more than a 15 degree deviation from normal (*Fig 6*):

```
model = lm(formula=avgTemp[highDepart]~barometer[highDepart]);
```

Then we add the regression line from the model to the scatterplot (*Fig 6*):

```
abline(model, col="blue");
```

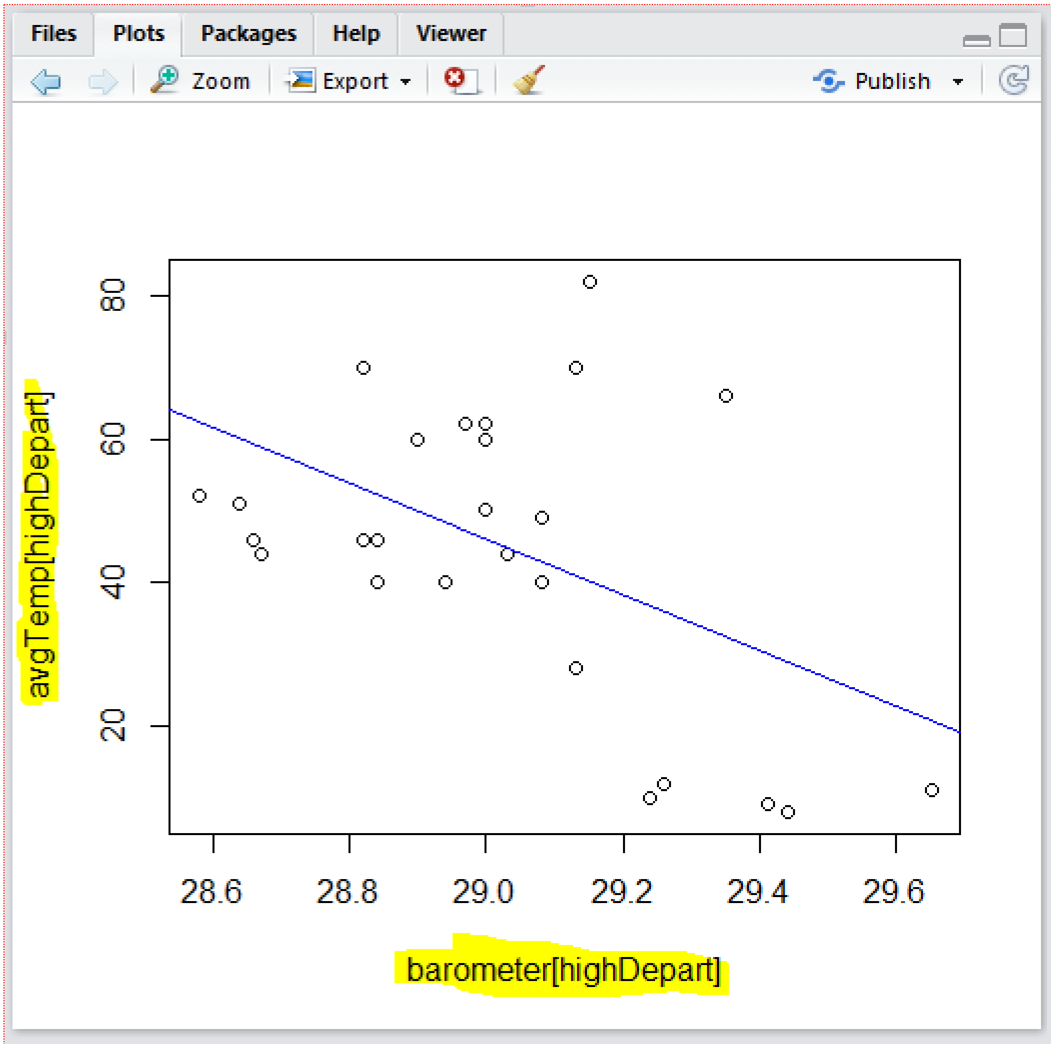

*Fig 6: Scatterplot of **avgTemp** vs **barometer** on days with abnormal temperatures -- with regression line added*

The scatterplot and regression suggest a relationship between ***avgTemp*** and ***barometer*** when temperatures deviates by more than 15 degrees from normal.

5.2 - Summarizing the linear model

We can print the linear model summary to the Console Window

```
print(summary(model));
```

```
Console   Markers
C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/

Call:
lm(formula = avgTemp[highDepart] ~ barometer[highDepart])

Residuals:
   Min     1Q Median     3Q    Max
-26.75 -12.63  -6.24  14.37  41.76

Coefficients:
                      Estimate Std. Error t value Pr(>|t|)
(Intercept)            1168.81     401.68   2.910  0.00749 **
barometer[highDepart]   -38.72      13.84  -2.797  0.00977 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 18.31 on 25 degrees of freedom
Multiple R-squared:  0.2384,	Adjusted R-squared:  0.2079
F-statistic: 7.826 on 1 and 25 DF,  p-value: 0.009769

>
```

*Fig 7: Summary of the linear model that suggests a relationship between **avgTemp** and **barometer**.*

5.3 - A second linear model

Wind directions plays a big role in weather conditions so we will repeat the analysis with southerly winds (***windDirection*** between **90** ***and*** **270** degrees). Note: **0** degrees is due north, **180** degrees is due south.

```
southWinds = which(windDirection > 90 & windDirection < 270);
plot(formula=avgTemp[southWinds]~barometer[southWinds]);
model2 = lm(formula=avgTemp[southWinds]~barometer[southWinds]);
abline(model2, col="blue");     # add the regression line to the plot
print(summary(model2));         # summary shows a relationship
```

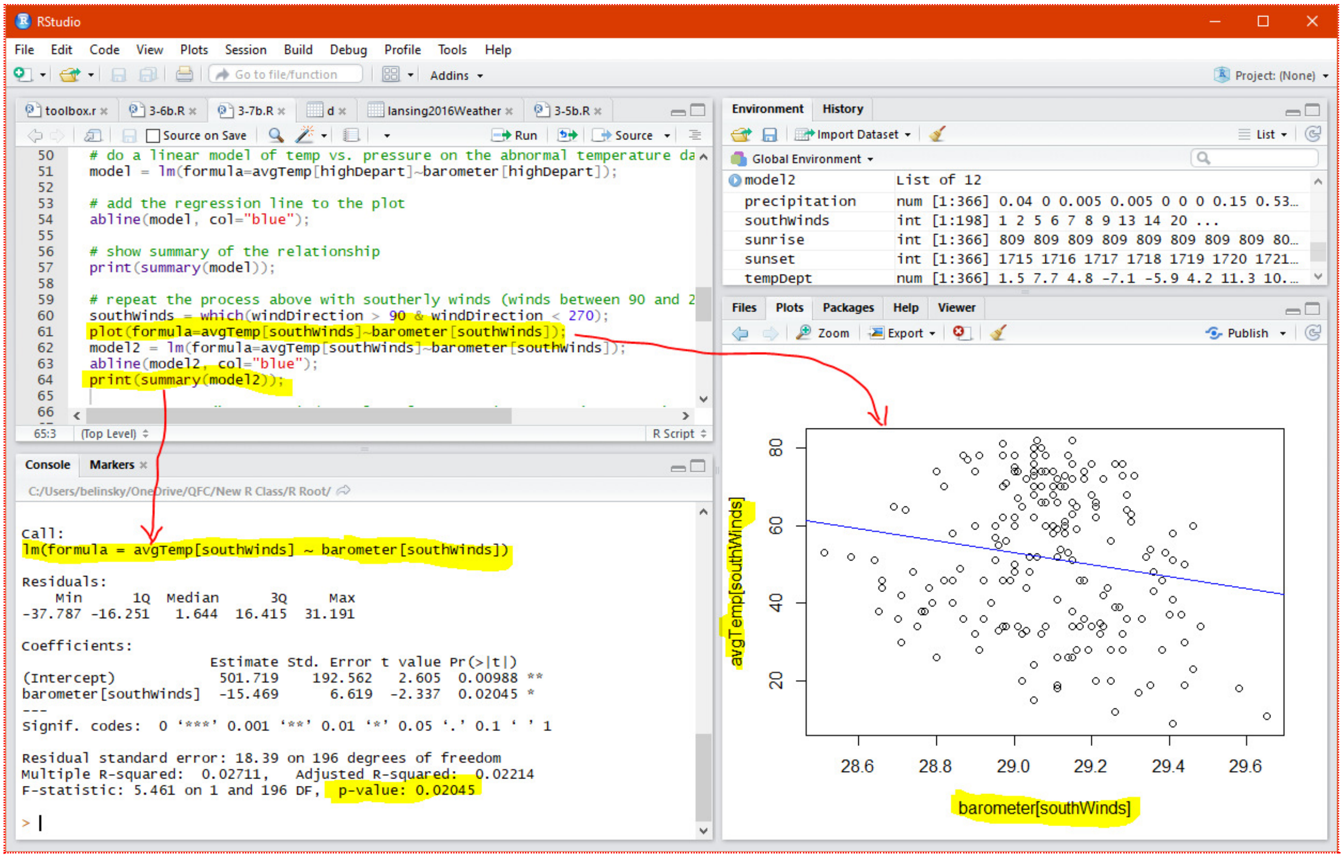

*Fig 8: Summary of linear model showing relationship between **avgTemp** and **barometer** on days with southerly winds*

6 - Finding patterns in string value with grep()

which() works well for subsetting vectors that have numeric values but ***which()*** is not equipped to deal flexibly with string values. As we saw in last lesson, ***eventData*** has values like: **SN, FG, HZ**, indicating **snow**, **fog**, and **haze** on that day.

We can use ***grep()*** to find all ***eventData*** *values that contain* **SN** (snow) (*Fig 9*):

```
daysWithSnow = grep("SN", eventData);
```

or use ***grep()*** find all ***eventData*** *values that contain* **RN** (rain) (*Fig 9*):

```
daysWithRain = grep("RN", eventData);
```

6.1 - grep() with the | condition

And we can use the ***or*** (**/**) operator to find all values in ***eventData*** *that contain either* **SN** or **RN** (*Fig 9*):

```
daysWithPrecip = grep("RN|SN", eventData);
```

There are **124** days with only rain, **65** with only snow, and **179** days with rain ***or*** snow.
124 + **65** = **189**, so there are **10** days (**189-179**) that had ***both*** rain ***and*** snow.

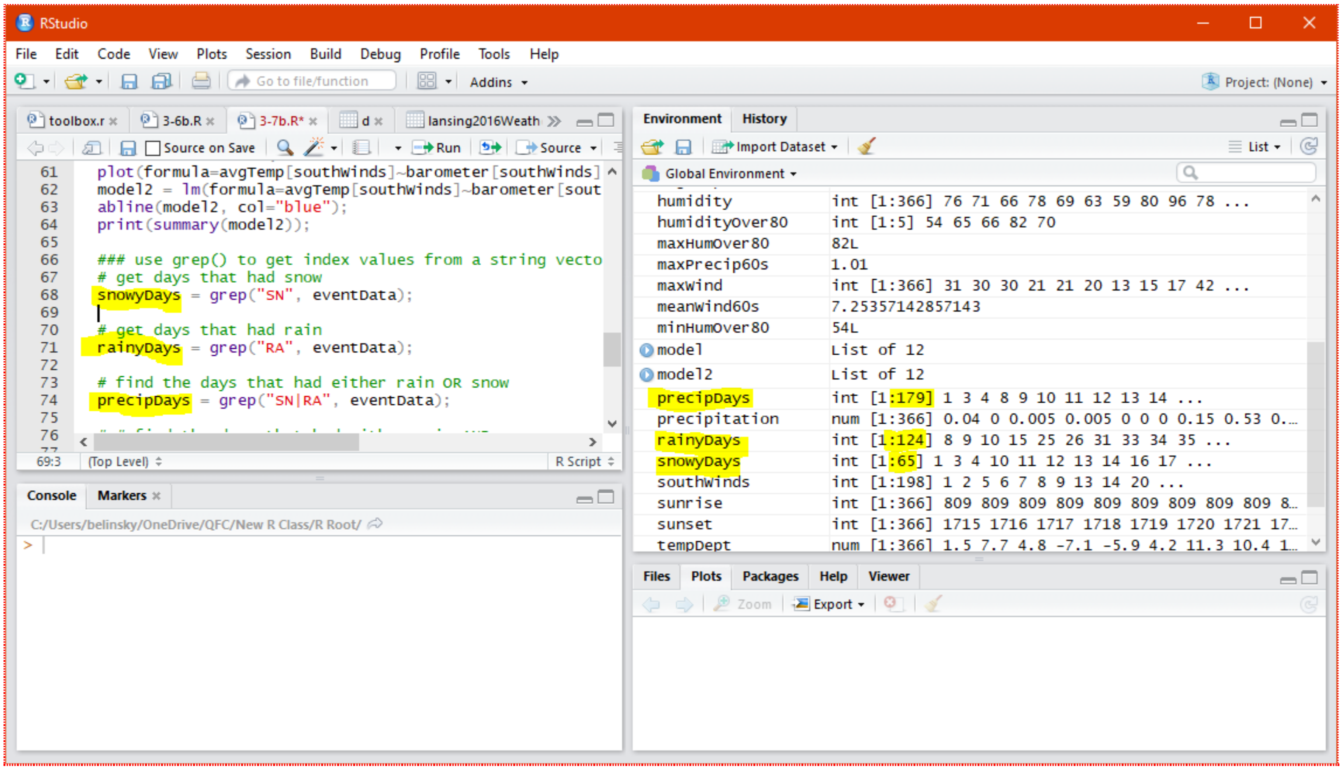

*Fig 9: Using the or (|) condition within a **grep()**.*

We can return days with rain or snow using ***grep()*** (the | operator) but we cannot use ***grep()*** to return days with ***both*** rain ***and*** snow (the ***&*** operator)

```
daysWithRainAndSnow1 = grep("RN&SN", eventData);  # this does not work!
```

The reason for this requires a lengthy explanation of how ***grep()*** (and regular expressions, in general) are executed.

6.2 - An alternative to & when using grep()

We have:

- ***daysWithSnow:*** the indexes of all days with snow, and
- ***daysWithRain:*** the indexes of all days with rain.

What we want is the *index values that are in both* ***daysWithRain*** *and* ***daysWithSnow***.

Luckily, there is a function for that called ***intersect()***. ***intersect***() produces a vector of values that occur in both vectors.

```
daysWithRainAndSnow2 = intersect(daysWithRain, daysWithSnow);
```

daysWithRainAndSnow2 has **10** values, representing the **10** values in ***eventData*** that had both ***RN*** (rain) and ***SN*** (snow).

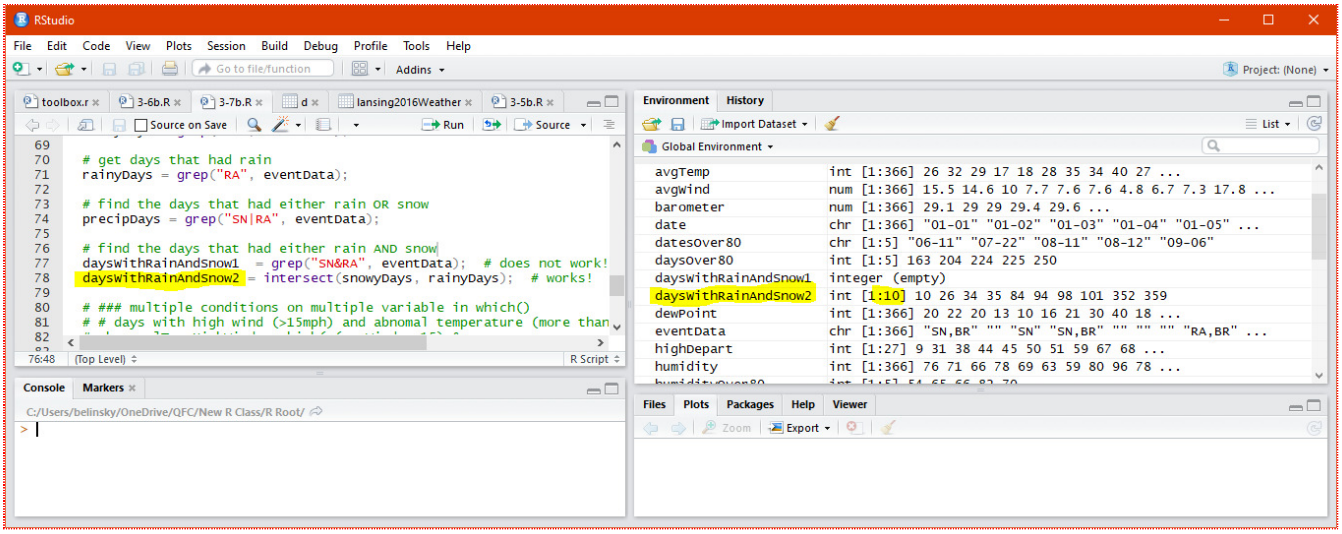

*Fig 10: Using **intersect()** to find values that occur in two vectors.*

Extension: ***union()*** -- sister function to ***intersect()***

7 - Multiple conditions on multiple variables

Similar to ***if()***, we can use ***which()*** to look at *multiple conditions on multiple variables*. The trick is to make sure you have the parentheses correct, because parentheses determine the order of operations.

abnormalTempHighWind gives the index of values where there were strong winds (greater than **15mph**) and the temperature deviated more than **10** degrees from average.

```
abnormalTempHighWind = which( (avgWind > 15) &
                              (tempDept < -10 | tempDept > 10 ) );
```

chillyLightRain gives the index for temperature values between **40 *and* 50** degrees where there was a light precipitation (between **0.1 *and* 0.2** inches)

```
chillyLightRain = which( (avgTemp > 40 & avgTemp < 50) &
                         (precipitation > 0.1 & precipitation < 0.2 ));
```

We see there are:

- **9** windy days with temperatures deviating more than 10 degrees from average and
- **3** days that were chilly with light rain.

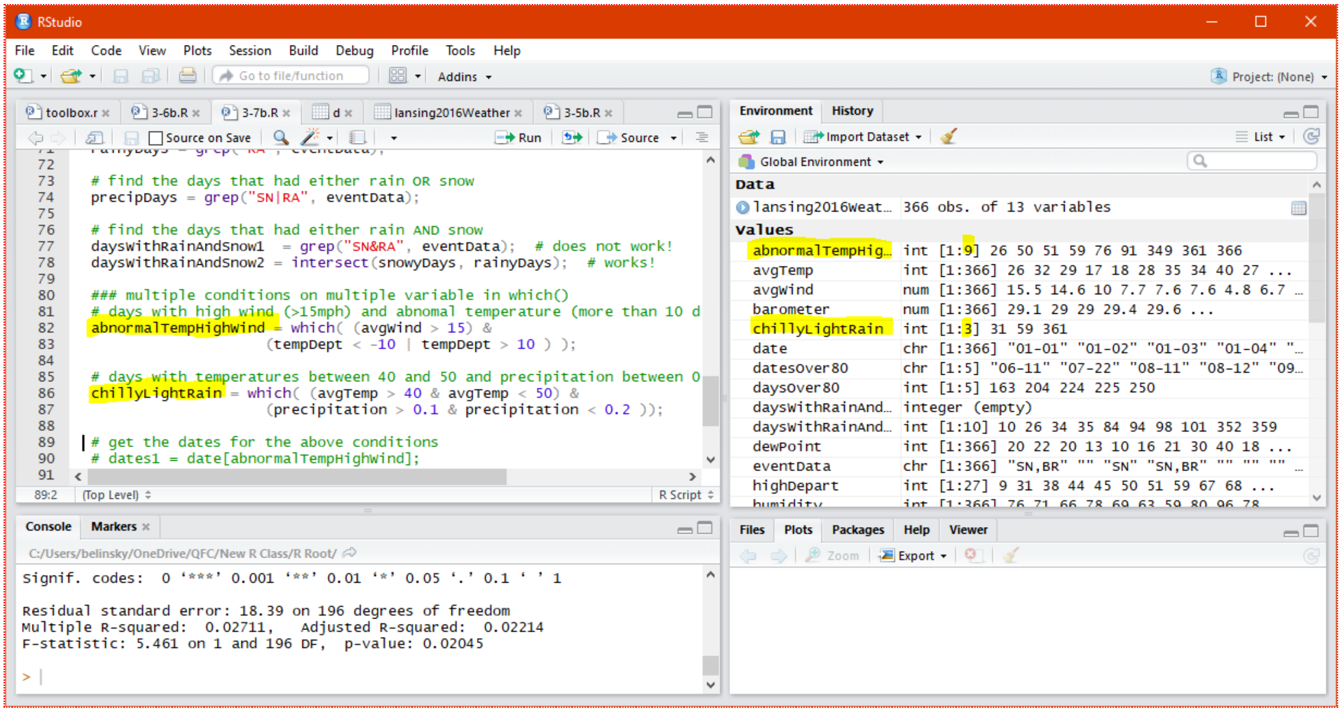

*Fig 11: Using **which()** with multiple conditions on multiple variables*

7.1 - Using the index vector to subset another vector

Let's get the actual dates for the conditions above.

We can use the subset vectors to find the dates for ***abnormalTempHighWind*** and ***chillyLightRain:***

```
dates1 = date[abnormalTempHighWind];
dates2 = date[chillyLightRain];
```

And we will print this information to the Console Window. Note: ***cat()*** is easier to use than ***print()*** for printing out string-only content.

```
cat("Abnormal temps and high winds:\n");
print(dates1);
cat("\nChilly days and light rain:\n");
print(dates2);
```

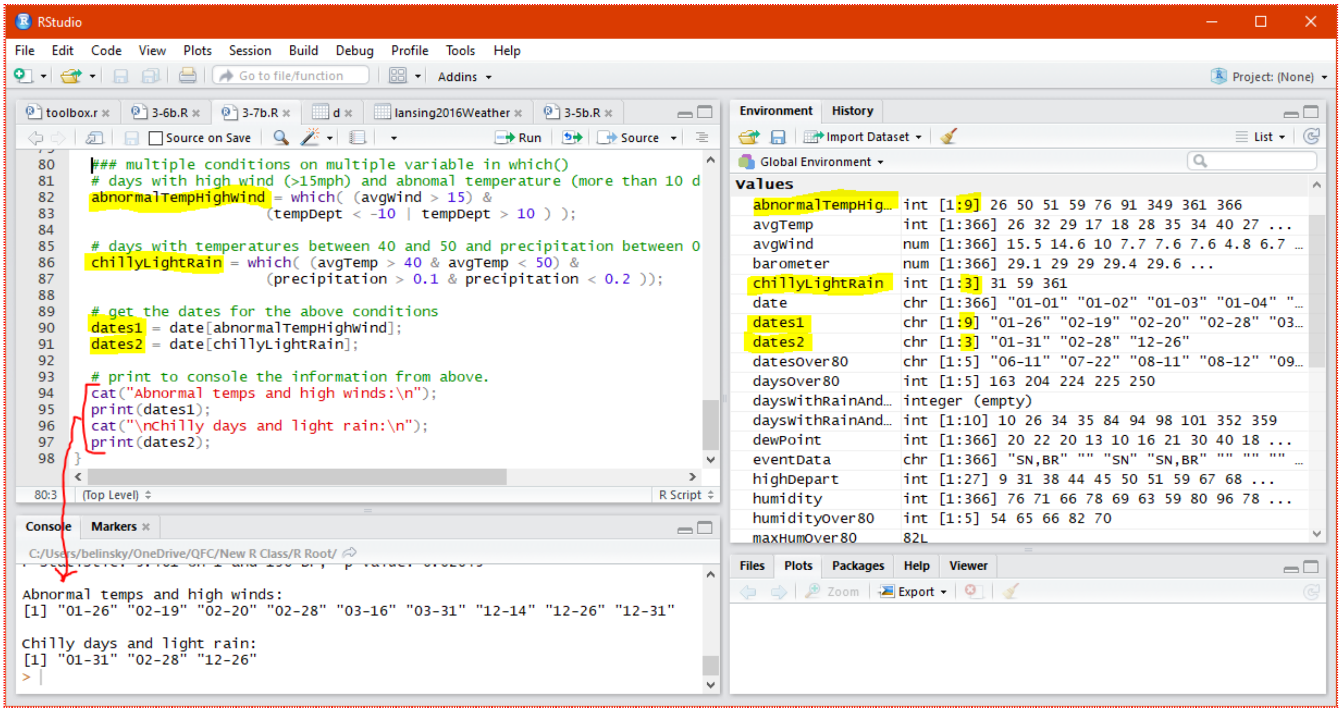

Fig 12: Printing out the results of subsetting vectors

8 - Application

Use the data from ***LansingNOAA2016Formatted.csv*** for this application.

1) Using ***grep()***, create a vector with indices for every day in the months of February and March.
hint: all December values are in the form **12-##**, all November values are in the form **11-##...**

2) Find the number of days the maximum wind speed was high (greater than 40) and for these days give:
   - the date
   - the amount of precipitation
   - the maximum wind speed

3) Print out the dates that have
  - northerly winds (less than 90 degrees or greater than 270 degrees)
  - northerly winds and precipitation greater than 0.5 inches

4) Does ***hoursOfSun*** (from last lesson's application) always correlate with temperature?
  - Make a scatterplot of ***avgTemp*** vs ***HoursOfSun*** for every two months (6 in all)
  - Create a linear models of ***avgTemp*** vs ***hoursOfSun*** for every two months (6 in all)
  - Add the regression lines to the scatterplots

9 - Extension: Differences between & and &&

Note: everything in this extension is also true for the differences between | and ||.

If you are comparing single values then ***&*** and ***&&*** will always function exactly the same so:

```
testNum = 4

```

```
if(testNum > 3 && testNum < 7)  # evaluates to TRUE
{
  # do something here
}

if(testNum > 3 & testNum < 7)   # also evaluates to TRUE
{
  # do something here
}
```

Most R programmers will use ***&*** in these circumstances. For this class, I chose to use ***&&*** because *in most other programming languages you need to use **&&*** when you are comparing single values. However, you could go back to lessons 1-9 and 1-10 and replace every ***&&*** with ***&*** and nothing will functionally change.

When dealing with vectors in R, ***&*** and ***&&*** will function differently. ***&&*** checks to see if *every value in the vector meets the condition* and produces one **TRUE** or **FALSE** statement whereas ***&*** *checks each value individually* and produces a ***TRUE/FALSE*** statement for each value.

Let's look at an example using the vector ***ages = c(2,7,3,9,6,3,5):***

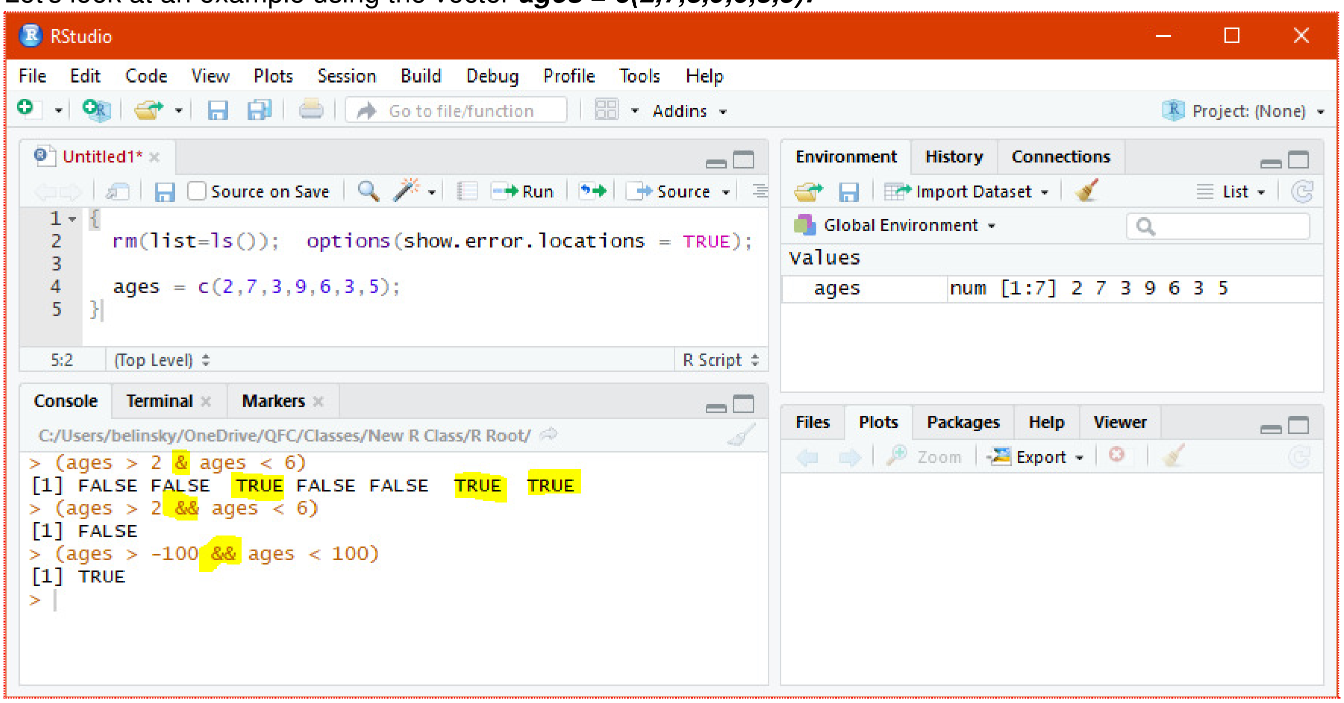

*Fig 13: Comparing **&** to **&&** in conditional statements with vectors.*

In the Console Window we see that ***(age > 2 & age < 6)*** produces seven values, representing each value of the ***ages*** vector. Since the 3rd, 6th, and 7th values in ***ages*** are between **2** and **6**, the 3rd, 6th, and 7th values are **TRUE**, the rest are **FALSE**.

(age > 2 && age < 6) produces one **FALSE** value because all the values in ***ages*** *do not meet the condition of being between 2 and 6*.

(age > -100 && age < 100) produces one **TRUE** value because all the values in ***ages*** *do meet the condition of being between -100 and 100*.

Note: In most other programming language, ***&*** functions differently -- it is a *bitwise operator*, which is an advanced topic.

10 - Extension: union(), the sister to intersect()

intersect() on two vectors returns the values that are represented in *both vectors*,
union() on two vectors returns the values that are represented in *either vector.*

So, if:

```
aVector = c(3, 4, 5, 6, 7);
bVector = c(6, 7, 8, 9, 10);
```

then:

```
intersect(aVector, bVector);
```

returns a vector with the values **6** and **7**, because **6** and **7** occurred in both ***aVector*** and ***bVector***

and:

```
union(aVector, bVector);
```

returns a vector with the values **3,4,5,6,7,8,9,10** -- or all the values that occur in either ***aVector*** or ***bVector***

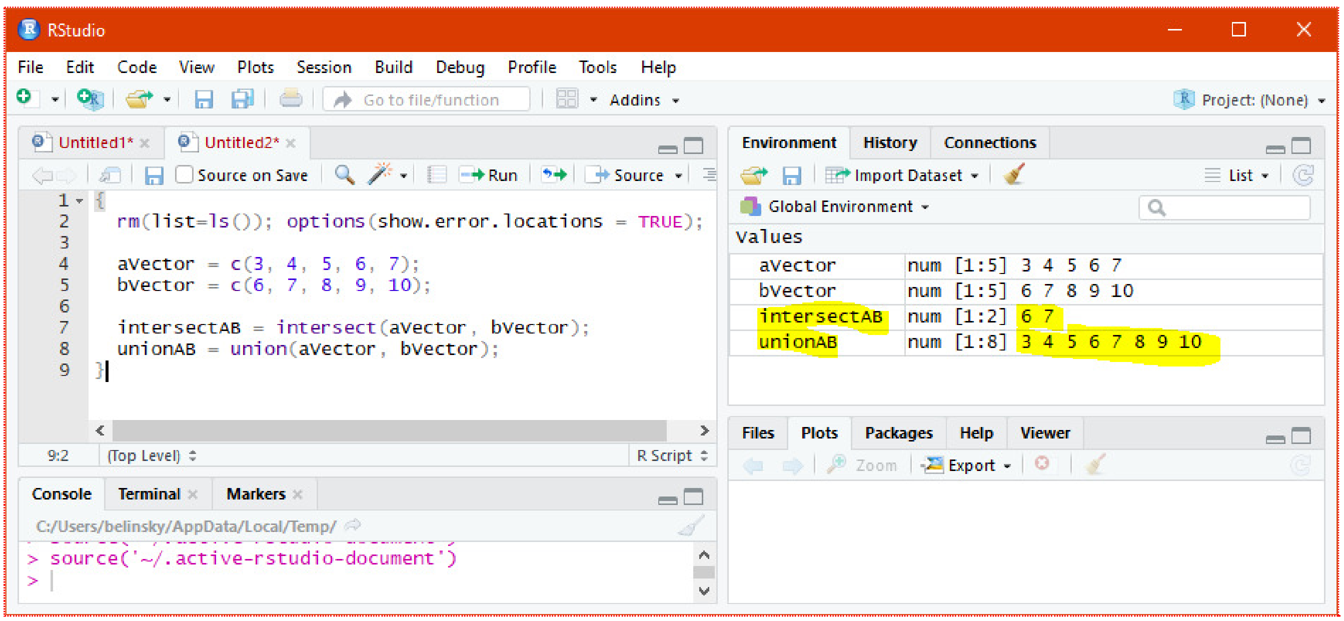

Fig 14: The intersection and union of two vectors.