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 - Class Project issues or other questions...

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

3 - Script and data for this lesson

The script for this lesson is located here.

We are using the data file, ***LansingNOAA2016Formatted.csv*** created in lesson 3-6. It you do not have the file, you can download it here.

4 - Gathering data

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"];
```

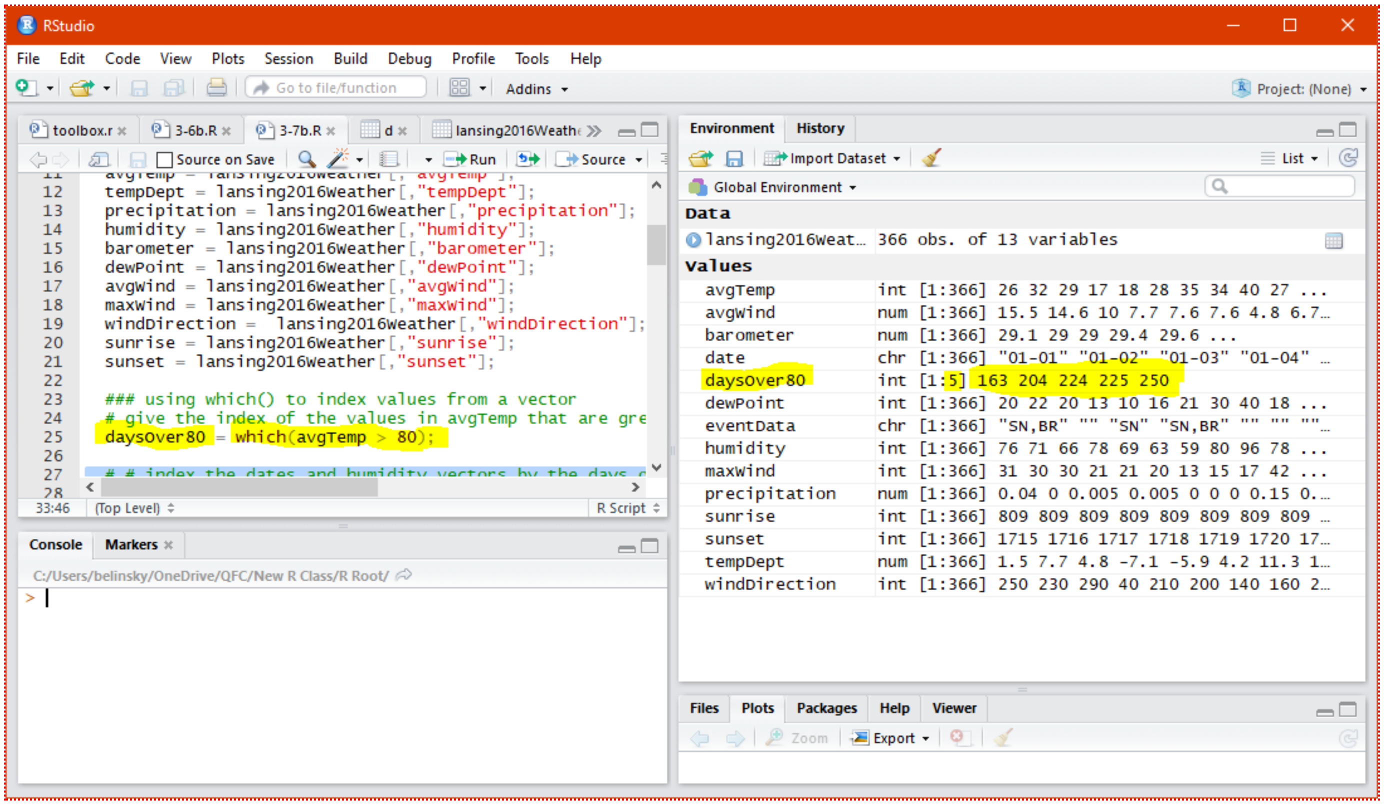
```
windDirection = lansing2016Weather[,"windDirection"];
sunrise = lansing2016Weather[,"sunrise"];
sunset = lansing2016Weather[,"sunset"];
```

5 - 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...).



```
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"];

### using which() to index values from a vector
# give the index of the values in avgTemp that are gre
daysOver80 = which(avgTemp > 80);

# # index the dates and humidity vectors by the days c
```

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

```
Data
lansing2016Weat…  366 obs. of 13 variables
Values
  avgTemp         int [1:366] 26 32 29 17 18 28 35 34 40 27 ...
  avgWind         num [1:366] 15.5 14.6 10 7.7 7.6 7.6 4.8 6.7…
  barometer       num [1:366] 29.1 29 29 29.4 29.6 ...
  date            chr [1:366] "01-01" "01-02" "01-03" "01-04" …
  daysOver80      int [1:5] 163 204 224 225 250
  dewPoint        int [1:366] 20 22 20 13 10 16 21 30 40 18 ...
  eventData       chr [1:366] "SN,BR" "" "SN" "SN,BR" "" "" ""…
  humidity        int [1:366] 76 71 66 78 69 63 59 80 96 78 ...
  maxWind         int [1:366] 31 30 30 21 21 20 13 15 17 42 ...
  precipitation   num [1:366] 0.04 0 0.005 0.005 0 0 0 0.15 0.…
  sunrise         int [1:366] 809 809 809 809 809 809 809 809 …
  sunset          int [1:366] 1715 1716 1717 1718 1719 1720 17…
  tempDept        num [1:366] 1.5 7.7 4.8 -7.1 -5.9 4.2 11.3 1…
  windDirection   int [1:366] 250 230 290 40 210 200 140 160 2…
```

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

5.1 - 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?

```r
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?

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

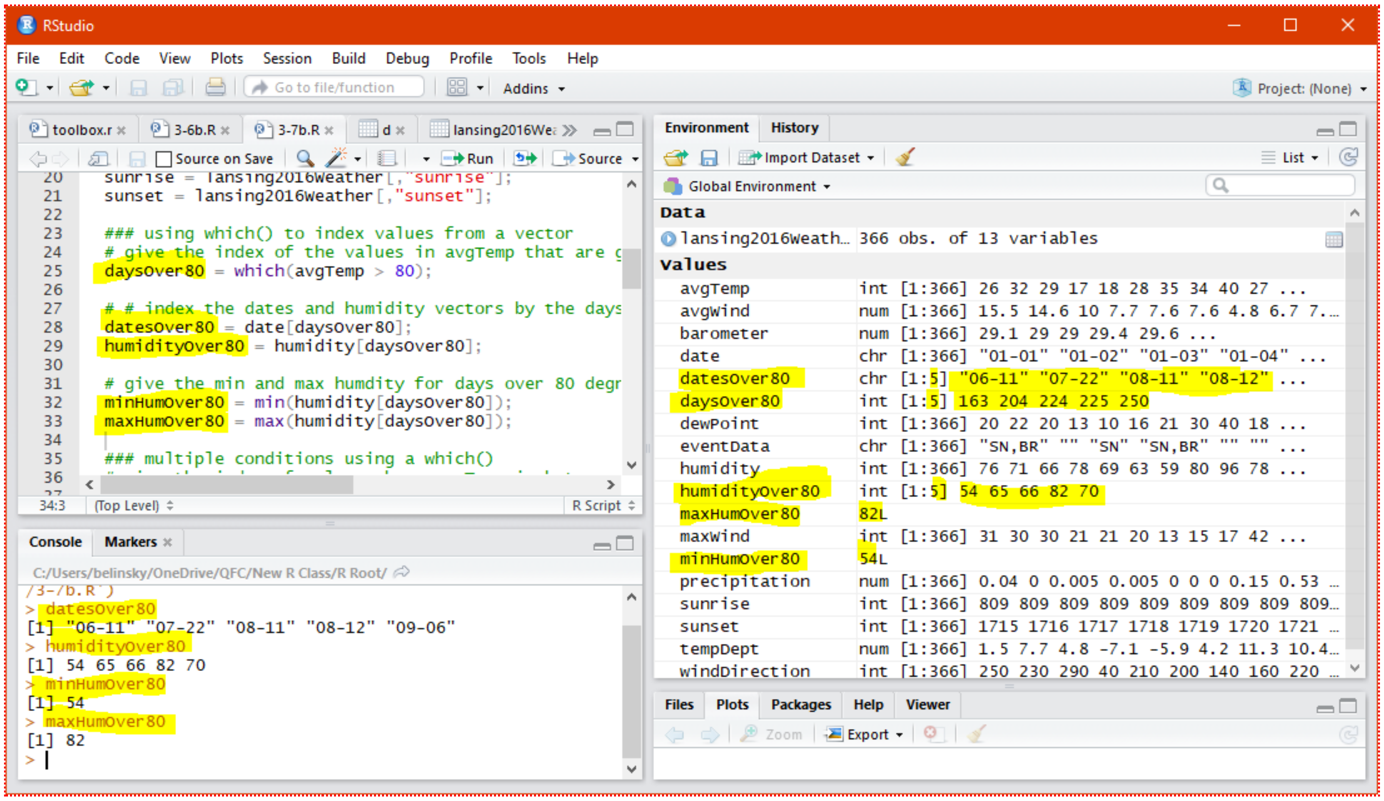
answer: 54%, 65%, 66%, 82%, and 70%

5.2 - 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.

```r
minHumOver80 = min(humidity[daysOver80]);
maxHumOver80 = max(humidity[daysOver80]);
```

answers: min=54%, max=82%



```r
sunrise = lansing2016Weather[,"sunrise"];
sunset = lansing2016Weather[,"sunset"];

### using which() to index values from a vector
# give the index of the values in avgTemp that are g
daysOver80 = which(avgTemp > 80);

# # index the dates and humidity vectors by the days
datesOver80 = date[daysOver80];
humidityOver80 = humidity[daysOver80];

# give the min and max humdity for days over 80 degr
minHumOver80 = min(humidity[daysOver80]);
maxHumOver80 = max(humidity[daysOver80]);

### multiple conditions using a which()
```

```
C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/
/3-7b.R')
> datesOver80
[1] "06-11" "07-22" "08-11" "08-12" "09-06"
> humidityOver80
[1] 54 65 66 82 70
> minHumOver80
[1] 54
> maxHumOver80
[1] 82
>
```

```
Data
lansing2016Weath…  366 obs. of 13 variables
Values
  avgTemp          int [1:366] 26 32 29 17 18 28 35 34 40 27 ...
  avgWind          num [1:366] 15.5 14.6 10 7.7 7.6 7.6 4.8 6.7 7.…
  barometer        num [1:366] 29.1 29 29 29.4 29.6 ...
  date             chr [1:366] "01-01" "01-02" "01-03" "01-04" ...
  datesOver80      chr [1:5] "06-11" "07-22" "08-11" "08-12" ...
  daysOver80       int [1:5] 163 204 224 225 250
  dewPoint         int [1:366] 20 22 20 13 10 16 21 30 40 18 ...
  eventData        chr [1:366] "SN,BR" "" "SN" "SN,BR" "" "" ...
  humidity         int [1:366] 76 71 66 78 69 63 59 80 96 78 ...
  humidityOver80   int [1:5] 54 65 66 82 70
  maxHumOver80     82L
  maxWind          int [1:366] 31 30 30 21 21 20 13 15 17 42 ...
  minHumOver80     54L
  precipitation    num [1:366] 0.04 0 0.005 0.005 0 0 0 0.15 0.53 …
  sunrise          int [1:366] 809 809 809 809 809 809 809 809 809…
  sunset           int [1:366] 1715 1716 1717 1718 1719 1720 1721 …
  tempDept         num [1:366] 1.5 7.7 4.8 -7.1 -5.9 4.2 11.3 10.4…
  windDirection    int [1:366] 250 230 290 40 210 200 140 160 220 …
```

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

5.3 - 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:

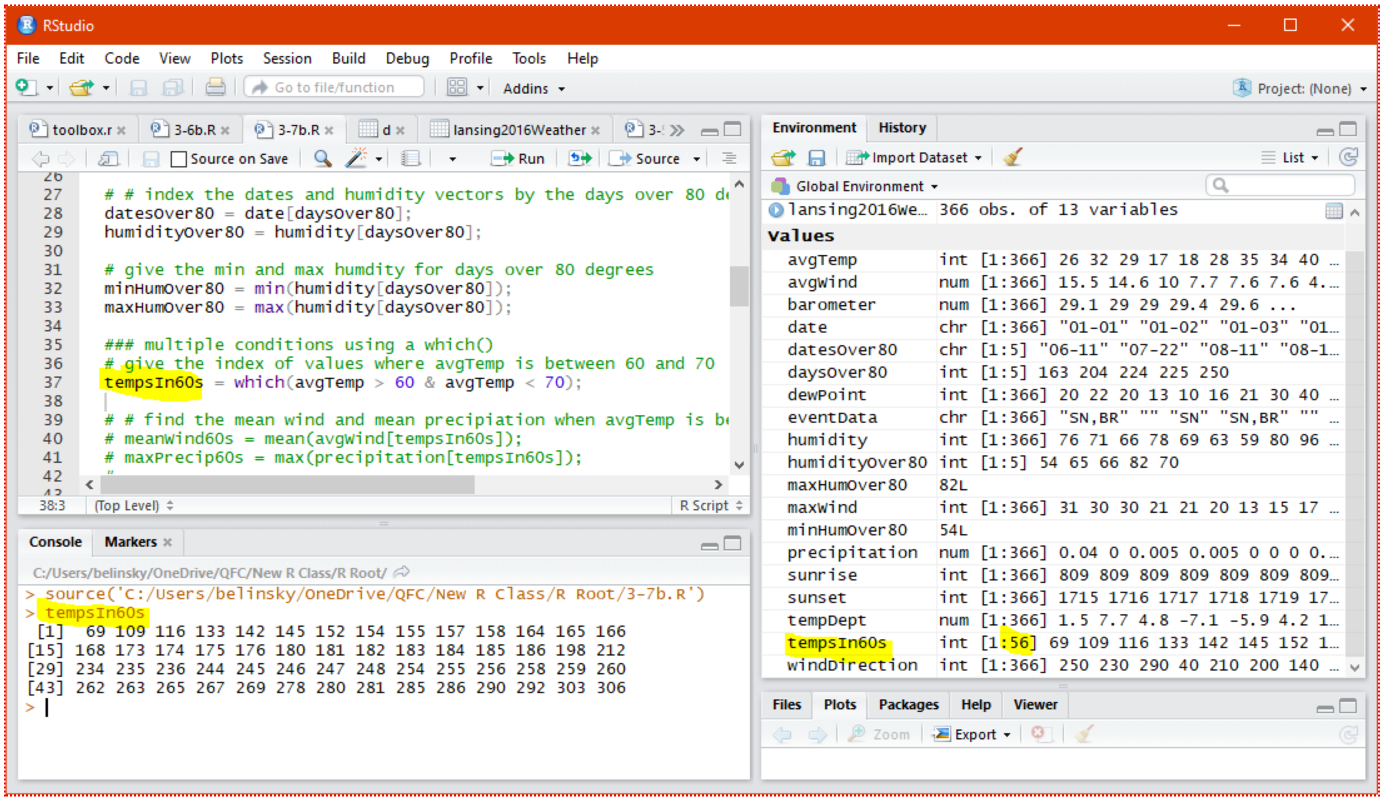
```r
if(avgTemp > 60 && avgTemp < 70)
```

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).



```
# # index the dates and humidity vectors by the days over 80 de
datesOver80 = date[daysOver80];
humidityOver80 = humidity[daysOver80];

# give the min and max humdity for days over 80 degrees
minHumOver80 = min(humidity[daysOver80]);
maxHumOver80 = max(humidity[daysOver80]);

### multiple conditions using a which()
# give the index of values where avgTemp is between 60 and 70
tempsIn60s = which(avgTemp > 60 & avgTemp < 70);

# # find the mean wind and mean precipiation when avgTemp is be
# meanWind60s = mean(avgWind[tempsIn60s]);
# maxPrecip60s = max(precipitation[tempsIn60s]);
```

```
C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/3-7b.R')
> tempsIn60s
 [1]  69 109 116 133 142 145 152 154 155 157 158 164 165 166
[15] 168 173 174 175 176 180 181 182 183 184 185 186 198 212
[29] 234 235 236 244 245 247 248 254 255 256 258 259 260
[43] 262 263 265 267 269 278 280 281 285 286 290 292 303 306
>
```

```
lansing2016we…   366 obs. of 13 variables
Values
  avgTemp          int [1:366] 26 32 29 17 18 28 35 34 40 …
  avgWind          num [1:366] 15.5 14.6 10 7.7 7.6 7.6 4.…
  barometer        num [1:366] 29.1 29 29 29.4 29.6 ...
  date             chr [1:366] "01-01" "01-02" "01-03" "01…
  datesOver80      chr [1:5] "06-11" "07-22" "08-11" "08-1…
  daysOver80       int [1:5] 163 204 224 225 250
  dewPoint         int [1:366] 20 22 20 13 10 16 21 30 40 …
  eventData        chr [1:366] "SN,BR" "" "SN" "SN,BR" "" …
  humidity         int [1:366] 76 71 66 78 69 63 59 80 96 …
  humidityOver80   int [1:5] 54 65 66 82 70
  maxHumOver80     82L
  maxWind          int [1:366] 31 30 30 21 21 20 13 15 17 …
  minHumOver80     54L
  precipitation    num [1:366] 0.04 0 0.005 0.005 0 0 0 0.…
  sunrise          int [1:366] 809 809 809 809 809 809 809…
  sunset           int [1:366] 1715 1716 1717 1718 1719 17…
  tempDept         num [1:366] 1.5 7.7 4.8 -7.1 -5.9 4.2 1…
  tempsIn60s       int [1:56] 69 109 116 133 142 145 152 1…
  windDirection    int [1:366] 250 230 290 40 210 200 140 …
```

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]);
```

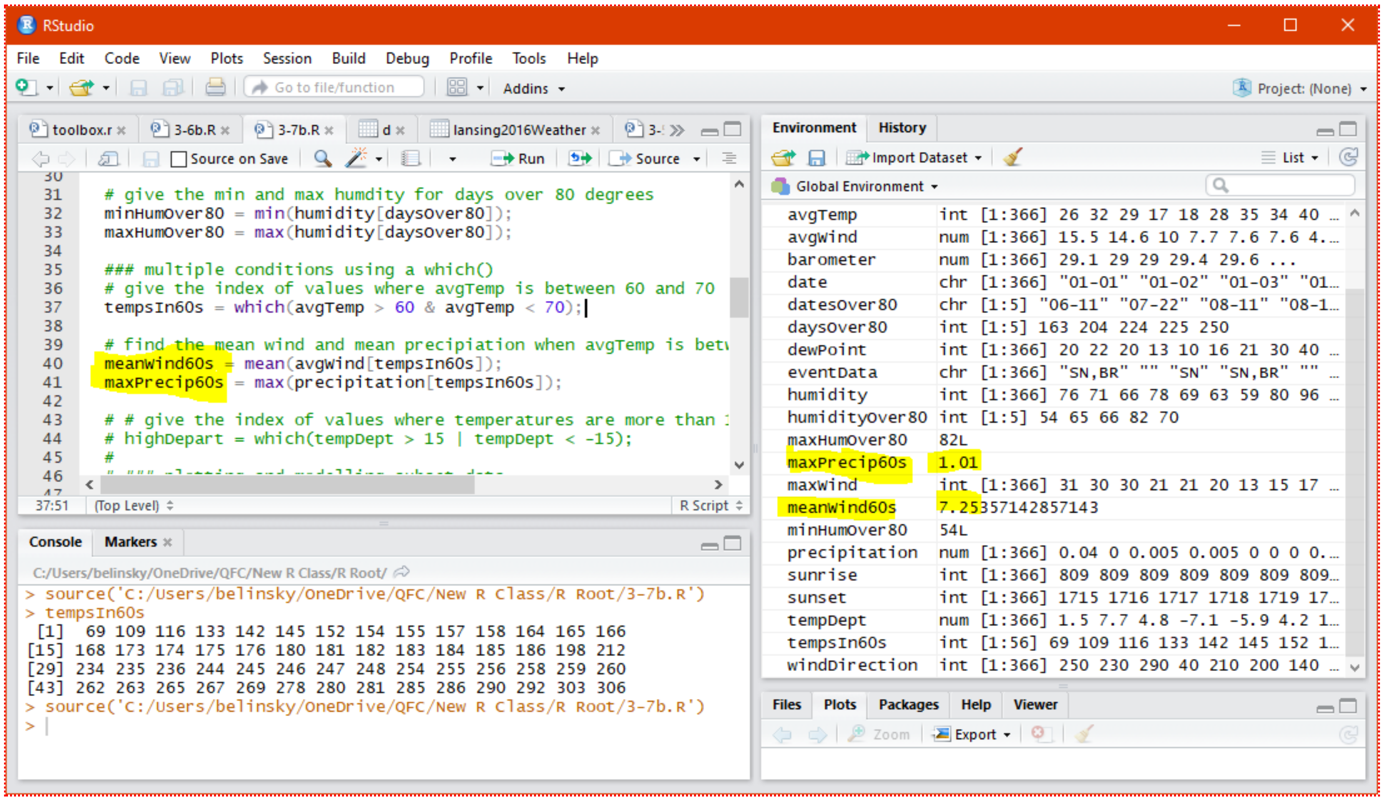
or the maximum precipitation for the **56** days with high temperature between **60** and **70:**

```
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**.



```
# give the min and max humdity for days over 80 degrees
minHumOver80 = min(humidity[daysOver80]);
maxHumOver80 = max(humidity[daysOver80]);

### multiple conditions using a which()
# give the index of values where avgTemp is between 60 and 70
tempsIn60s = which(avgTemp > 60 & avgTemp < 70);

# find the mean wind and mean precipiation when avgTemp is betw
meanWind60s = mean(avgWind[tempsIn60s]);
maxPrecip60s = max(precipitation[tempsIn60s]);

# # give the index of values where temperatures are more than 1
# highDepart = which(tempDept > 15 | tempDept < -15);
#
```

```
C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/3-7b.R')
> tempsIn60s
 [1]  69 109 116 133 142 145 152 154 155 157 158 164 165 166
[15] 168 173 174 175 176 180 181 182 183 184 185 186 198 212
[29] 234 235 236 244 245 246 247 248 254 255 256 258 259 260
[43] 262 263 265 267 269 278 280 281 285 286 290 292 303 306
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/3-7b.R')
> 
```

```
avgTemp          int [1:366] 26 32 29 17 18 28 35 34 40 ...
avgWind          num [1:366] 15.5 14.6 10 7.7 7.6 7.6 4....
barometer        num [1:366] 29.1 29 29 29.4 29.6 ...
date             chr [1:366] "01-01" "01-02" "01-03" "01...
datesOver80      chr [1:5] "06-11" "07-22" "08-11" "08-1...
daysOver80       int [1:5] 163 204 224 225 250
dewPoint         int [1:366] 20 22 20 13 10 16 21 30 40 ...
eventData        chr [1:366] "SN,BR" "" "SN" "SN,BR" "" ...
humidity         int [1:366] 76 71 66 78 69 63 59 80 96 ...
humidityOver80   int [1:5] 54 65 66 82 70
maxHumOver80     82L
maxPrecip60s     1.01
maxWind          int [1:366] 31 30 30 21 21 20 13 15 17 ...
meanWind60s      7.25357142857143
minHumOver80     54L
precipitation    num [1:366] 0.04 0 0.005 0.005 0 0 0 0....
sunrise          int [1:366] 809 809 809 809 809 809 809...
sunset           int [1:366] 1715 1716 1717 1718 1719 17...
tempDept         num [1:366] 1.5 7.7 4.8 -7.1 -5.9 4.2 1...
tempsIn60s       int [1:56] 69 109 116 133 142 145 152 1...
windDirection    int [1:366] 250 230 290 40 210 200 140 ...
```

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

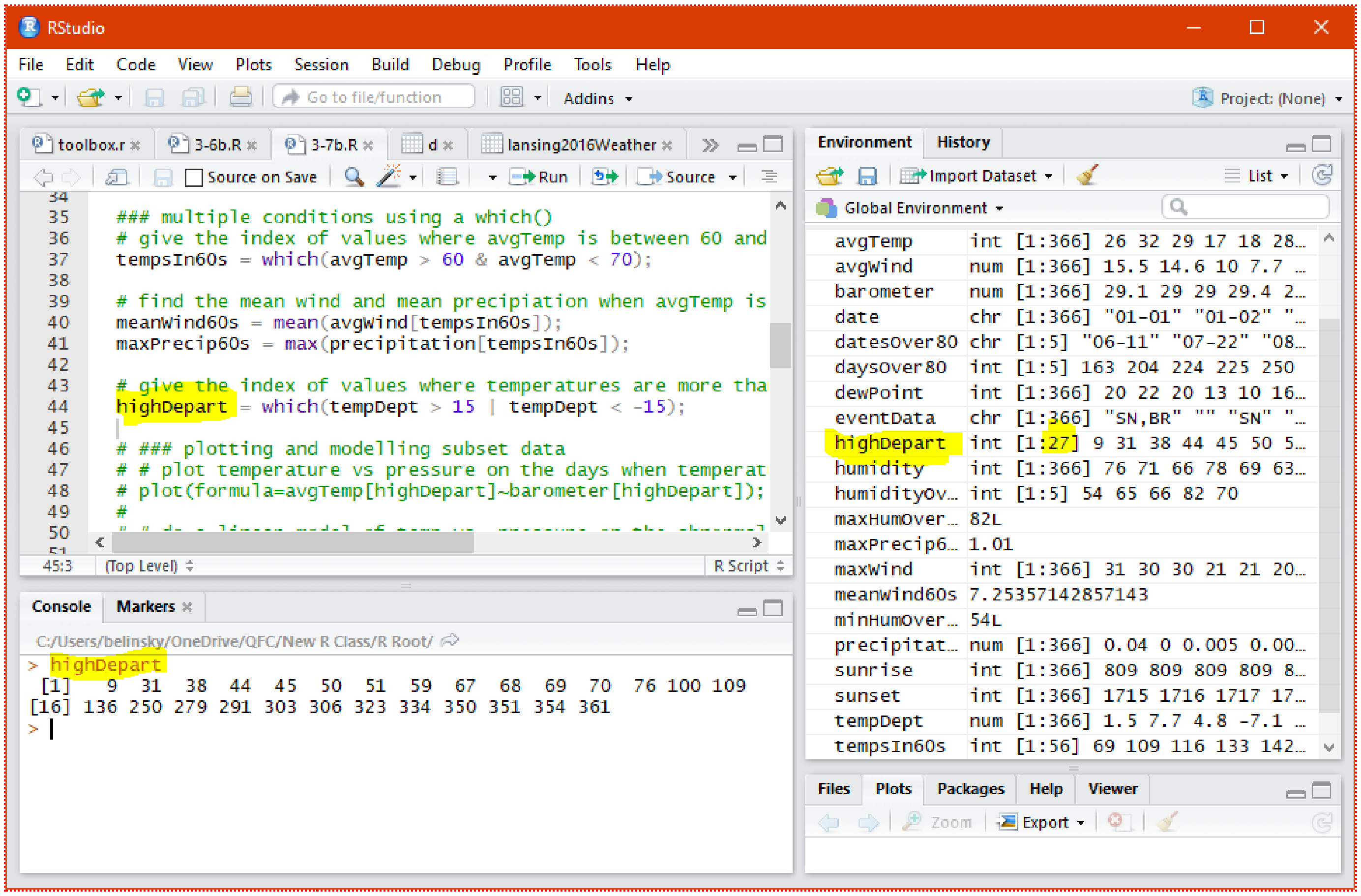
6 - Linear models using subsetted vectors

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);
```



```
### multiple conditions using a which()
# give the index of values where avgTemp is between 60 and
tempsIn60s = which(avgTemp > 60 & avgTemp < 70);

# find the mean wind and mean precipiation when avgTemp is
meanWind60s = mean(avgWind[tempsIn60s]);
maxPrecip60s = max(precipitation[tempsIn60s]);

# give the index of values where temperatures are more tha
highDepart = which(tempDept > 15 | tempDept < -15);

# ### plotting and modelling subset data
# # plot temperature vs pressure on the days when temperat
# plot(formula=avgTemp[highDepart]~barometer[highDepart]);
#
```

```
C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/
> highDepart
 [1]   9  31  38  44  45  50  51  59  67  68  69  70  76 100 109
[16] 136 250 279 291 303 306 323 334 350 351 354 361
> |
```

```
avgTemp        int  [1:366] 26 32 29 17 18 28…
avgWind        num  [1:366] 15.5 14.6 10 7.7 …
barometer      num  [1:366] 29.1 29 29 29.4 2…
date           chr  [1:366] "01-01" "01-02" "…
datesOver80    chr  [1:5] "06-11" "07-22" "08…
daysOver80     int  [1:5] 163 204 224 225 250
dewPoint       int  [1:366] 20 22 20 13 10 16…
eventData      chr  [1:366] "SN,BR" "" "SN" "…
highDepart     int  [1:27] 9 31 38 44 45 50 5…
humidity       int  [1:366] 76 71 66 78 69 63…
humidityOv…    int  [1:5] 54 65 66 82 70
maxHumOver…    82L
maxPrecip6…    1.01
maxWind        int  [1:366] 31 30 30 21 21 20…
meanWind60s    7.25357142857143
minHumOver…    54L
precipitat…    num  [1:366] 0.04 0 0.005 0.00…
sunrise        int  [1:366] 809 809 809 809 8…
sunset         int  [1:366] 1715 1716 1717 17…
tempDept       num  [1:366] 1.5 7.7 4.8 -7.1 …
tempsIn60s     int  [1:56] 69 109 116 133 142…
```

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

6.1 - 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");
```

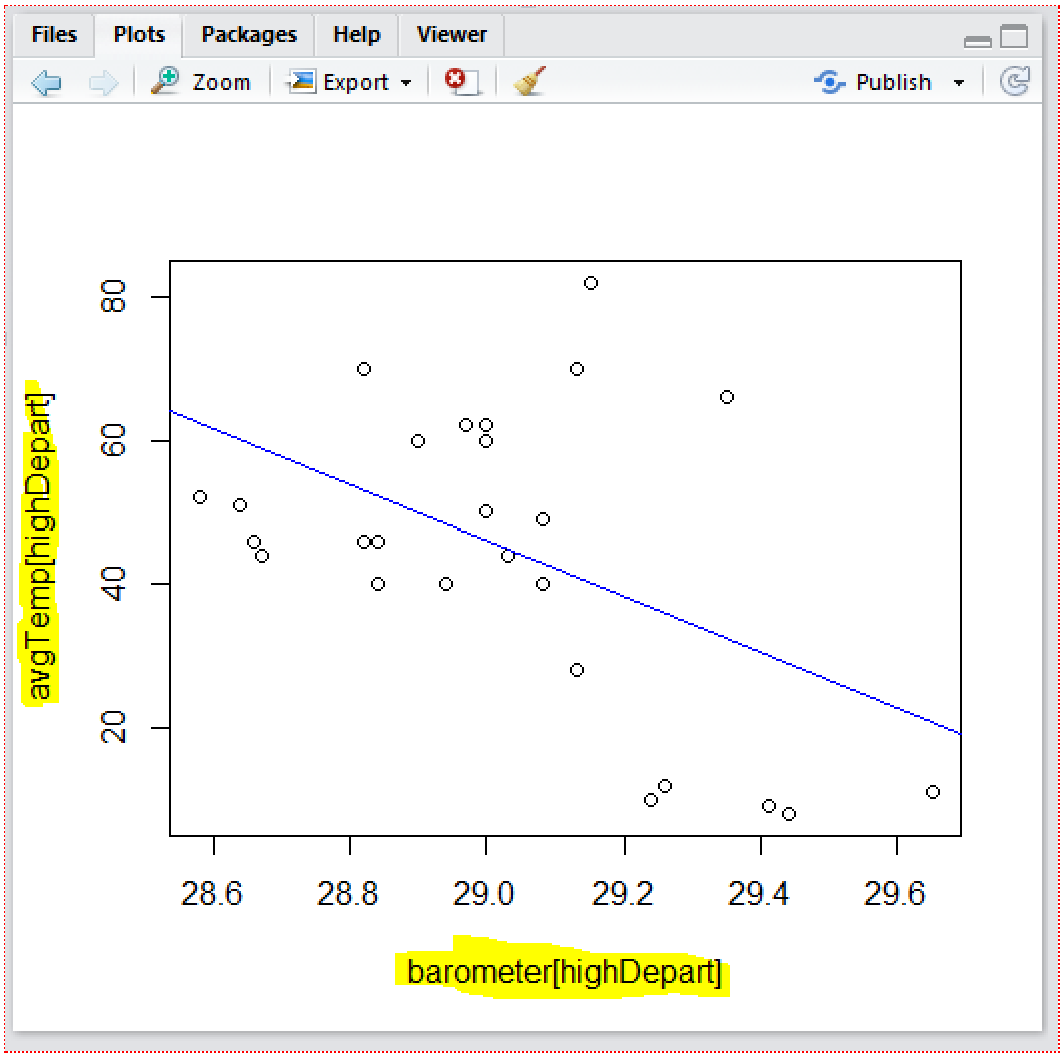


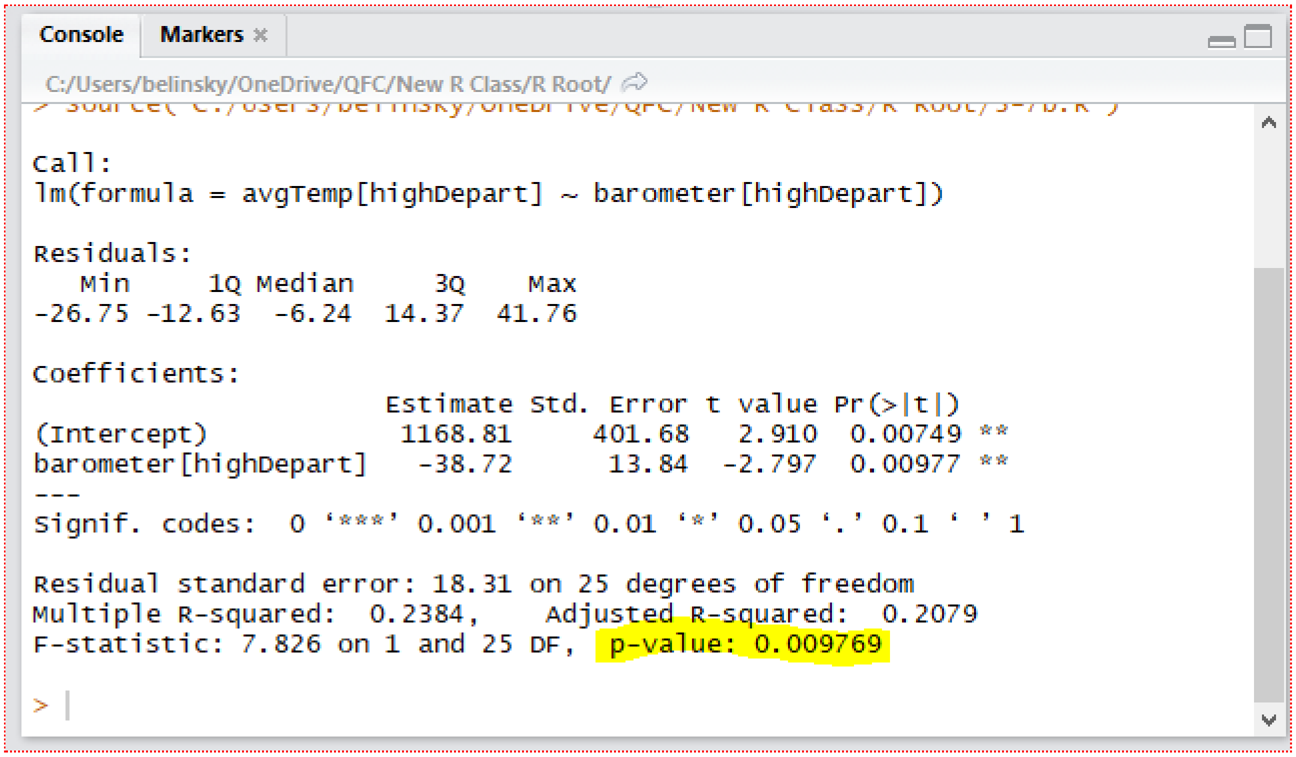
*Fig 6: Scatterplot of **avgTemp** vs **barometer** on days with abnormal temperatures -- with regression line added*

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

6.2 - Summarizing the linear model

We can print the linear model summary to the Console Window

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



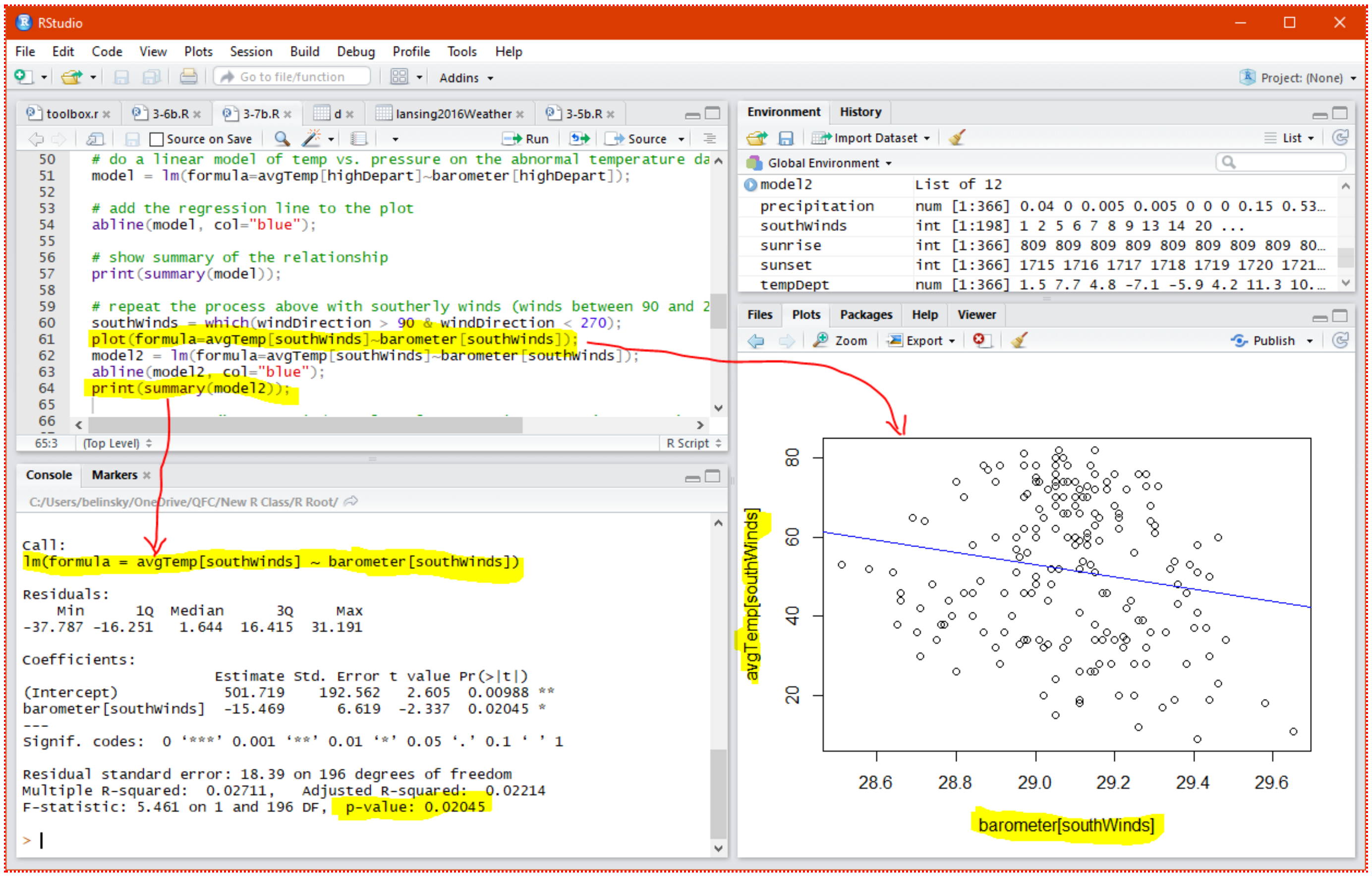
```
Call:
lm(formula = avgTemp[highDepart] ~ barometer[highDepart])

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**.*

6.3 - 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
```



```r
# do a linear model of temp vs. pressure on the abnormal temperature da
model = lm(formula=avgTemp[highDepart]~barometer[highDepart]);

# add the regression line to the plot
abline(model, col="blue");

# show summary of the relationship
print(summary(model));

# repeat the process above with southerly winds (winds between 90 and 2
southWinds = which(windDirection > 90 & windDirection < 270);
plot(formula=avgTemp[southWinds]~barometer[southWinds]);
model2 = lm(formula=avgTemp[southWinds]~barometer[southWinds]);
abline(model2, col="blue");
print(summary(model2));
```

```
model2          List of 12
  precipitation num [1:366] 0.04 0 0.005 0.005 0 0 0 0.15 0.53…
  southWinds    int [1:198] 1 2 5 6 7 8 9 13 14 20 ...
  sunrise       int [1:366] 809 809 809 809 809 809 809 809 80…
  sunset        int [1:366] 1715 1716 1717 1718 1719 1720 1721…
  tempDept      num [1:366] 1.5 7.7 4.8 -7.1 -5.9 4.2 11.3 10.…
```

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

Residuals:
    Min      1Q  Median      3Q     Max 
-37.787 -16.251   1.644  16.415  31.191 

Coefficients:
                      Estimate Std. Error t value Pr(>|t|)   
(Intercept)            501.719    192.562   2.605  0.00988 **
barometer[southWinds]  -15.469      6.619  -2.337  0.02045 * 
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 18.39 on 196 degrees of freedom
Multiple R-squared:  0.02711,	Adjusted R-squared:  0.02214 
F-statistic: 5.461 on 1 and 196 DF,  p-value: 0.02045

> |
```

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

7 - 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*):

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

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

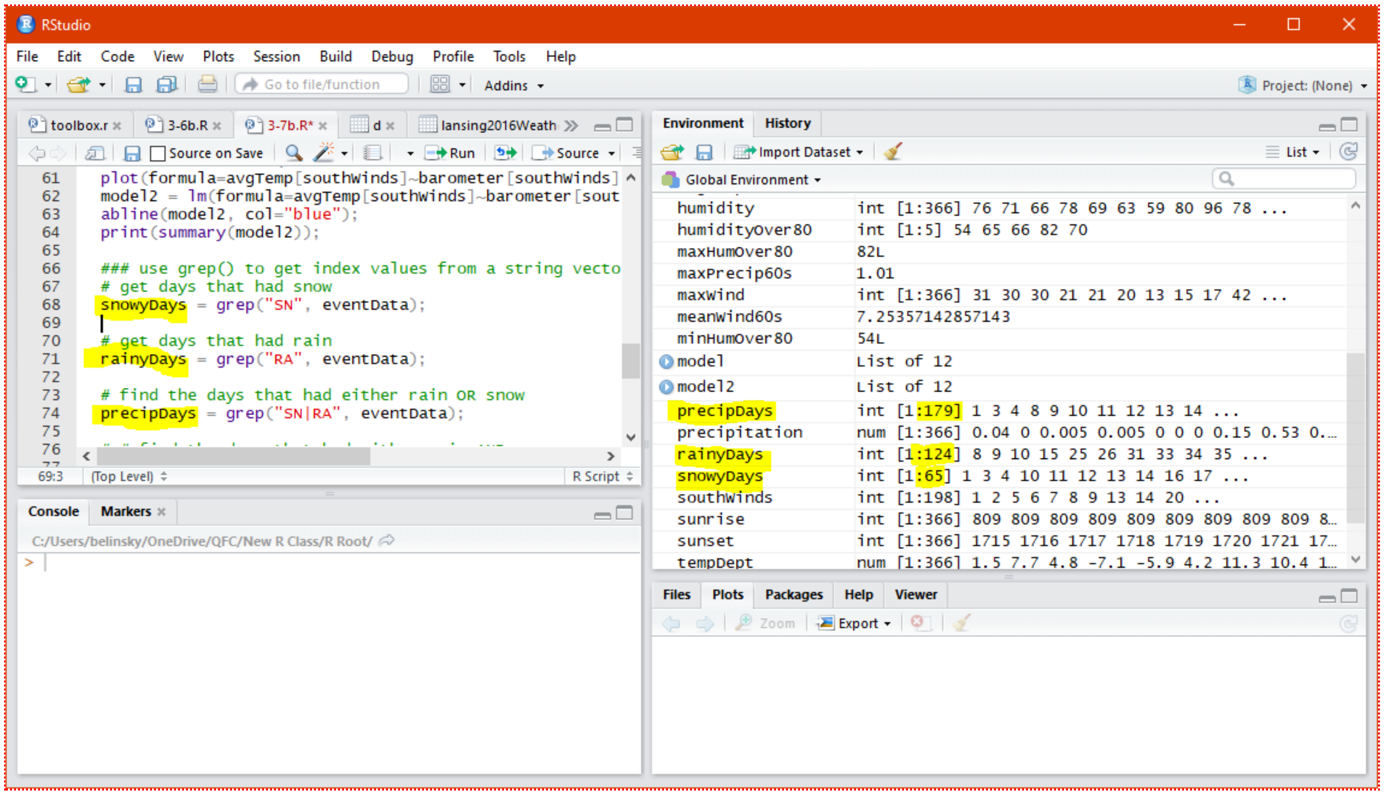
```r
daysWithRain = grep("RN", eventData);
```

7.1 - 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*):

```r
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.



```
plot(formula=avgTemp[southWinds]~barometer[southWinds]
model2 = lm(formula=avgTemp[southWinds]~barometer[sout
abline(model2, col="blue");
print(summary(model2));

### use grep() to get index values from a string vecto
# get days that had snow
snowyDays = grep("SN", eventData);

# get days that had rain
rainyDays = grep("RA", eventData);

# find the days that had either rain OR snow
precipDays = grep("SN|RA", eventData);
```

```
humidity          int [1:366] 76 71 66 78 69 63 59 80 96 78 ...
humidityOver80    int [1:5] 54 65 66 82 70
maxHumOver80      82L
maxPrecip60s      1.01
maxWind           int [1:366] 31 30 30 21 21 20 13 15 17 42 ...
meanWind60s       7.25357142857143
minHumOver80      54L
model             List of 12
model2            List of 12
precipDays        int [1:179] 1 3 4 8 9 10 11 12 13 14 ...
precipitation     num [1:366] 0.04 0 0.005 0.005 0 0 0 0.15 0.53 0...
rainyDays         int [1:124] 8 9 10 15 25 26 31 33 34 35 ...
snowyDays         int [1:65] 1 3 4 10 11 12 13 14 16 17 ...
southWinds        int [1:198] 1 2 5 6 7 8 9 13 14 20 ...
sunrise           int [1:366] 809 809 809 809 809 809 809 809 809 8...
sunset            int [1:366] 1715 1716 1717 1718 1719 1720 1721 17...
tempDept          num [1:366] 1.5 7.7 4.8 -7.1 -5.9 4.2 11.3 10.4 1...
```

*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.

7.2 - An alternative to & when using grep()

We have:

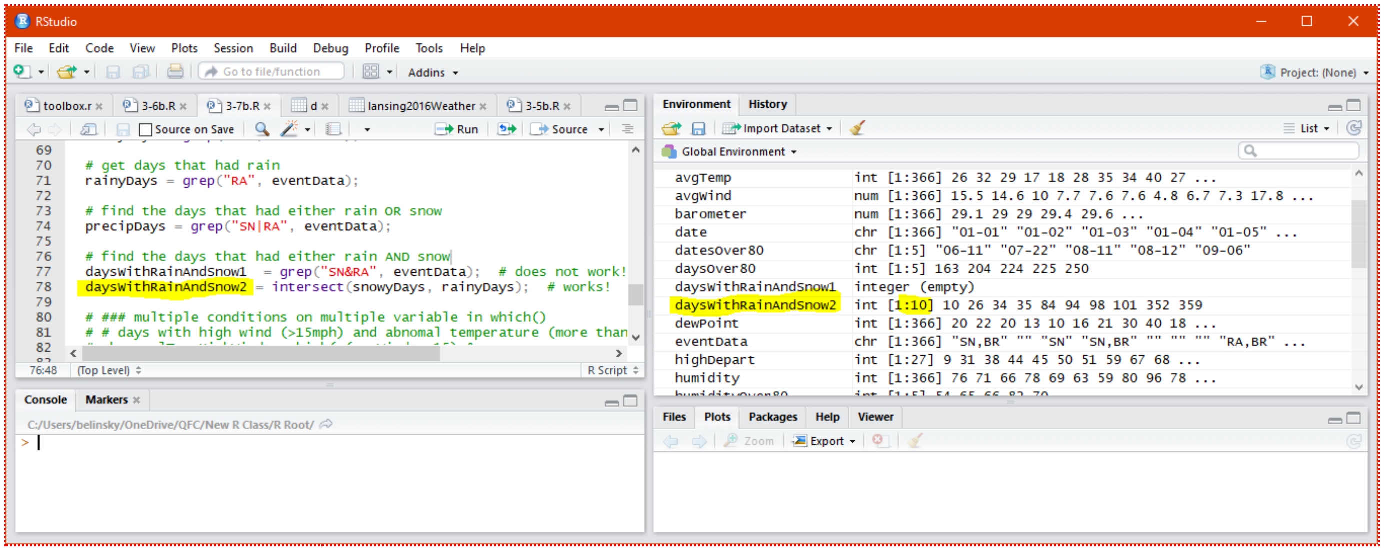
- ***daysWithSnow:*** the indexes of all days with snow, and
- ***daysWithRain:*** the indexes of all days with rain.

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).



```
# get days that had rain
rainyDays = grep("RA", eventData);

# find the days that had either rain OR snow
precipDays = grep("SN|RA", eventData);

# find the days that had either rain AND snow
daysWithRainAndSnow1  = grep("SN&RA", eventData);  # does not work!
daysWithRainAndSnow2 = intersect(snowyDays, rainyDays);  # works!

# ### multiple conditions on multiple variable in which()
# # days with high wind (>15mph) and abnomal temperature (more than
```

```
avgTemp               int [1:366] 26 32 29 17 18 28 35 34 40 27 ...
avgWind               num [1:366] 15.5 14.6 10 7.7 7.6 7.6 4.8 6.7 7.3 17.8 ...
barometer             num [1:366] 29.1 29 29 29.4 29.6 ...
date                  chr [1:366] "01-01" "01-02" "01-03" "01-04" "01-05" ...
datesOver80           chr [1:5] "06-11" "07-22" "08-11" "08-12" "09-06"
daysOver80            int [1:5] 163 204 224 225 250
daysWithRainAndSnow1  integer (empty)
daysWithRainAndSnow2  int [1:10] 10 26 34 35 84 94 98 101 352 359
dewPoint              int [1:366] 20 22 20 13 10 16 21 30 40 18 ...
eventData             chr [1:366] "SN,BR" "" "SN" "SN,BR" "" "" "" "RA,BR" ...
highDepart            int [1:27] 9 31 38 44 45 50 51 59 67 68 ...
humidity              int [1:366] 76 71 66 78 69 63 59 80 96 78 ...
```

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

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

8 - 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.

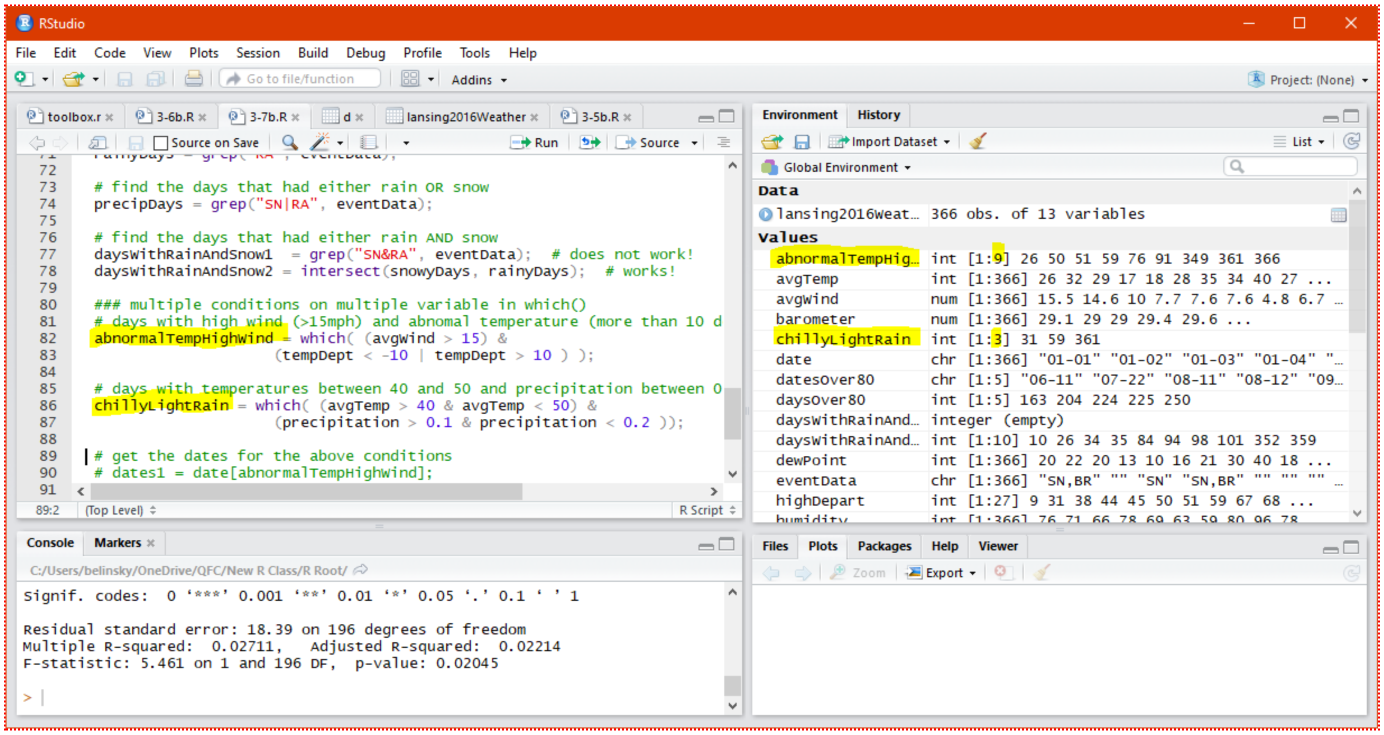
```
abnormalTempHighWind = which( (avgWind > 15) &
                              (tempDept < -10 | tempDept > 10 ) );
```

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.



```
# find the days that had either rain OR snow
precipDays = grep("SN|RA", eventData);

# find the days that had either rain AND snow
daysWithRainAndSnow1  = grep("SN&RA", eventData);   # does not work!
daysWithRainAndSnow2 = intersect(snowyDays, rainyDays);   # works!

### multiple conditions on multiple variable in which()
# days with high wind (>15mph) and abnomal temperature (more than 10 d
abnormalTempHighWind = which( (avgWind > 15) &
                             (tempDept < -10 | tempDept > 10 ) );

# days with temperatures between 40 and 50 and precipitation between 0
chillyLightRain = which( (avgTemp > 40 & avgTemp < 50) &
                             (precipitation > 0.1 & precipitation < 0.2 ));

# get the dates for the above conditions
# dates1 = date[abnormalTempHighWind];
```

```
Data
lansing2016Weat…    366 obs. of 13 variables
Values
abnormalTempHig…    int [1:9] 26 50 51 59 76 91 349 361 366
avgTemp             int [1:366] 26 32 29 17 18 28 35 34 40 27 ...
avgWind             num [1:366] 15.5 14.6 10 7.7 7.6 7.6 4.8 6.7 …
barometer           num [1:366] 29.1 29 29 29.4 29.6 ...
chillyLightRain     int [1:3] 31 59 361
date                chr [1:366] "01-01" "01-02" "01-03" "01-04" "…
datesOver80         chr [1:5] "06-11" "07-22" "08-11" "08-12" "09…
daysOver80          int [1:5] 163 204 224 225 250
daysWithRainAnd…    integer (empty)
daysWithRainAnd…    int [1:10] 10 26 34 35 84 94 98 101 352 359
dewPoint            int [1:366] 20 22 20 13 10 16 21 30 40 18 ...
eventData           chr [1:366] "SN,BR" "" "SN" "SN,BR" "" "" "" …
highDepart          int [1:27] 9 31 38 44 45 50 51 59 67 68 ...
```

```
C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 18.39 on 196 degrees of freedom
Multiple R-squared:  0.02711,	Adjusted R-squared:  0.02214 
F-statistic: 5.461 on 1 and 196 DF,  p-value: 0.02045

> 
```

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

8.1 - Using the index vector to subset another vector

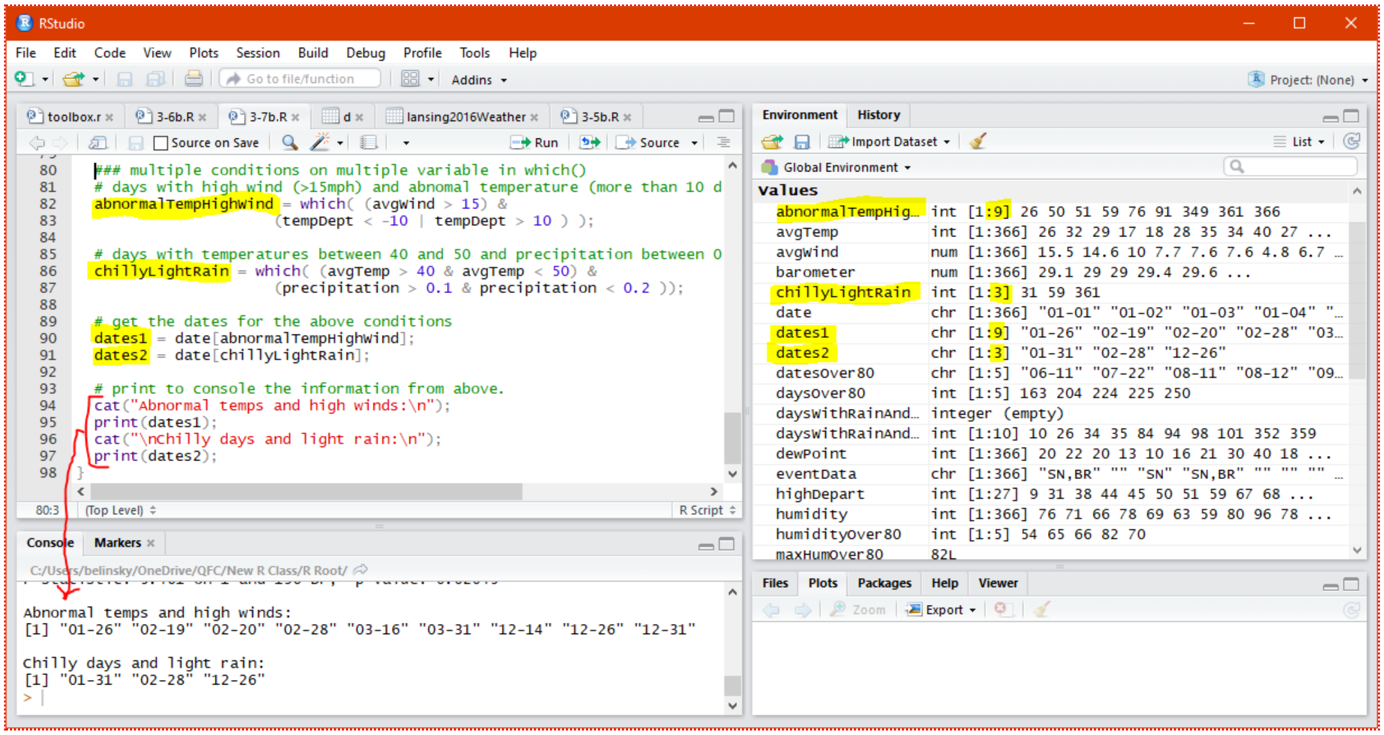
Let's get the actual dates for the conditions above.

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);
```



```r
### multiple conditions on multiple variable in which()
# days with high wind (>15mph) and abnomal temperature (more than 10 d
abnormalTempHighWind = which( (avgWind > 15) &
                              (tempDept < -10 | tempDept > 10 ) );

# days with temperatures between 40 and 50 and precipitation between 0
chillyLightRain = which( (avgTemp > 40 & avgTemp < 50) &
                              (precipitation > 0.1 & precipitation < 0.2 ));

# get the dates for the above conditions
dates1 = date[abnormalTempHighWind];
dates2 = date[chillyLightRain];

# print to console the information from above.
cat("Abnormal temps and high winds:\n");
print(dates1);
cat("\nChilly days and light rain:\n");
print(dates2);
}
```

```
Abnormal temps and high winds:
[1] "01-26" "02-19" "02-20" "02-28" "03-16" "03-31" "12-14" "12-26" "12-31"

Chilly days and light rain:
[1] "01-31" "02-28" "12-26"
>
```

```
Values
  abnormalTempHig…  int [1:9] 26 50 51 59 76 91 349 361 366
  avgTemp           int [1:366] 26 32 29 17 18 28 35 34 40 27 ...
  avgWind           num [1:366] 15.5 14.6 10 7.7 7.6 7.6 4.8 6.7 …
  barometer         num [1:366] 29.1 29 29 29.4 29.6 ...
  chillyLightRain   int [1:3] 31 59 361
  date              chr [1:366] "01-01" "01-02" "01-03" "01-04" "…
  dates1            chr [1:9] "01-26" "02-19" "02-20" "02-28" "03…
  dates2            chr [1:3] "01-31" "02-28" "12-26"
  datesOver80       chr [1:5] "06-11" "07-22" "08-11" "08-12" "09…
  daysOver80        int [1:5] 163 204 224 225 250
  daysWithRainAnd…  integer (empty)
  daysWithRainAnd…  int [1:10] 10 26 34 35 84 94 98 101 352 359
  dewPoint          int [1:366] 20 22 20 13 10 16 21 30 40 18 ...
  eventData         chr [1:366] "SN,BR" "" "SN" "SN,BR" "" "" "" …
  highDepart        int [1:27] 9 31 38 44 45 50 51 59 67 68 ...
  humidity          int [1:366] 76 71 66 78 69 63 59 80 96 78 ...
  humidityOver80    int [1:5] 54 65 66 82 70
  maxHumOver80      82L
```

Fig 12: Printing out the results of subsetting vectors

9 - Application

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

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) For the whole year, find the number of days the peak wind speed was high (greater than 40) and for these days give:
   - the date
   - the amount of precipitation
   - the peak wind speed

3) For the whole year, print out the dates that have
  - northerly winds (between 90 degrees and 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

*Save you script file as **app3-8.r** in the **scripts** folder of your RStudio Project for the class.*

10 - Extension: Differences between & and &&

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

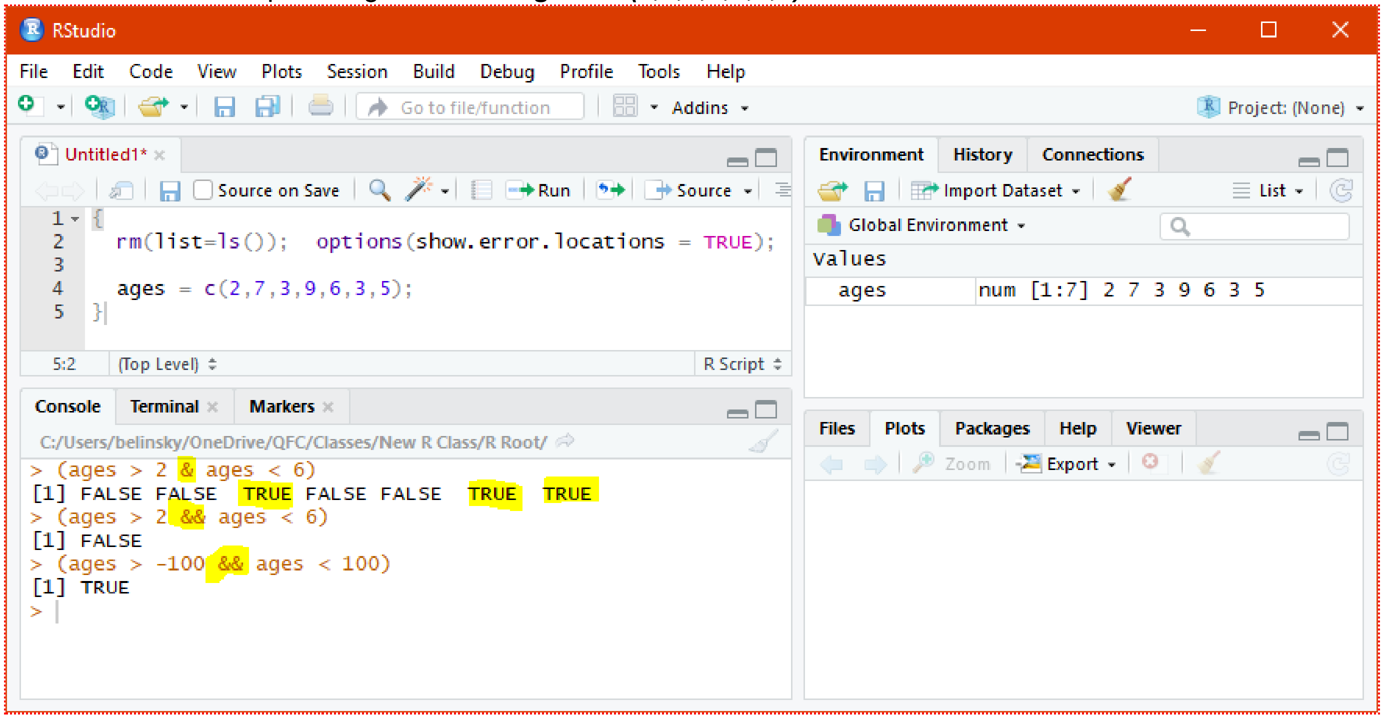
If you are comparing single values then **&** and **&&** will always function exactly the same so:

```
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):***



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

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

```
Values
 ages          num [1:7] 2 7 3 9 6 3 5
```

```
C:/Users/belinsky/OneDrive/QFC/Classes/New R Class/R Root/
> (ages > 2 & ages < 6)
[1] FALSE FALSE  TRUE FALSE FALSE  TRUE  TRUE
> (ages > 2 && ages < 6)
[1] FALSE
> (ages > -100 && ages < 100)
[1] TRUE
>
```

*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.

11 - 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:

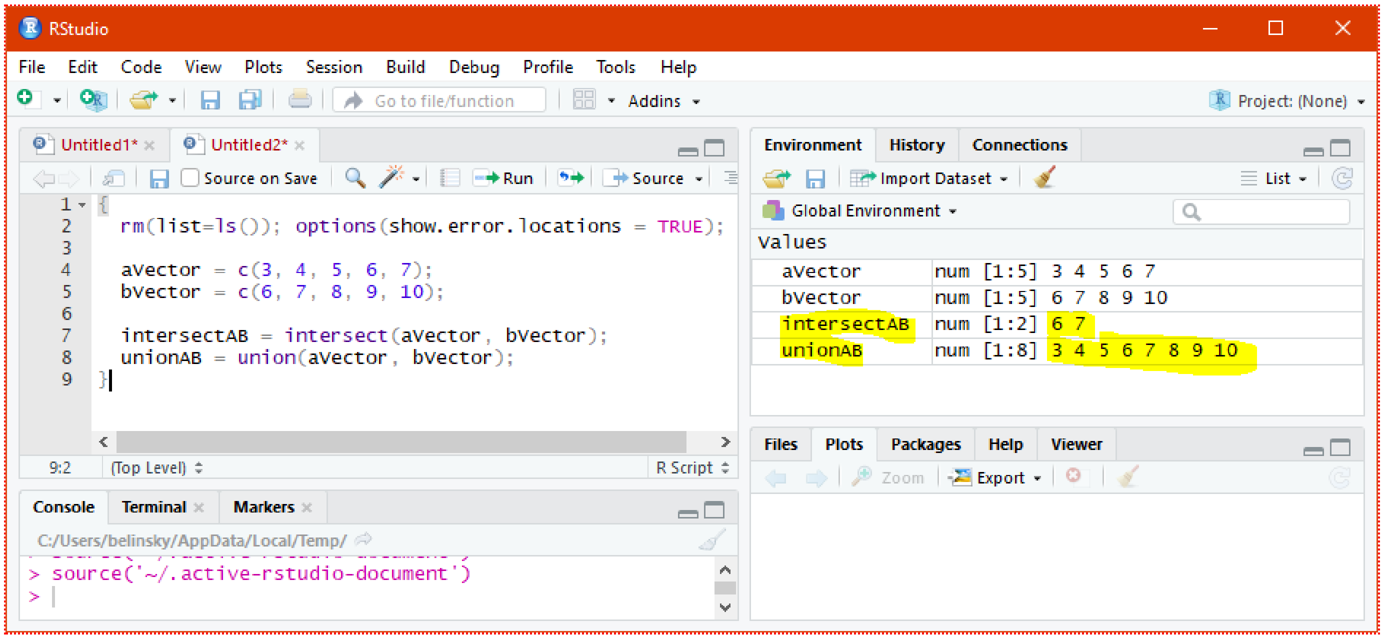
```
intersect(aVector, bVector);
```

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***


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Untitled1* Untitled2*
Source on Save Run Source
Environment History Connections
Import Dataset List
Global Environment
Files Plots Packages Help Viewer
Console Terminal Markers
C:/Users/belinsky/AppData/Local/Temp/

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

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

  intersectAB = intersect(aVector, bVector);
  unionAB = union(aVector, bVector);
}
```

```
Values
  aVector      num [1:5] 3 4 5 6 7
  bVector      num [1:5] 6 7 8 9 10
  intersectAB  num [1:2] 6 7
  unionAB      num [1:8] 3 4 5 6 7 8 9 10
```

```
> source('~/.active-rstudio-document')
> 
```

Fig 14: The intersection and union of two vectors.