02-05: Vector Operations and NA

1 - Purpose

- performing mathematics on vectors
- dealing with missing value in a vector
- R shortcuts for vector mathematics

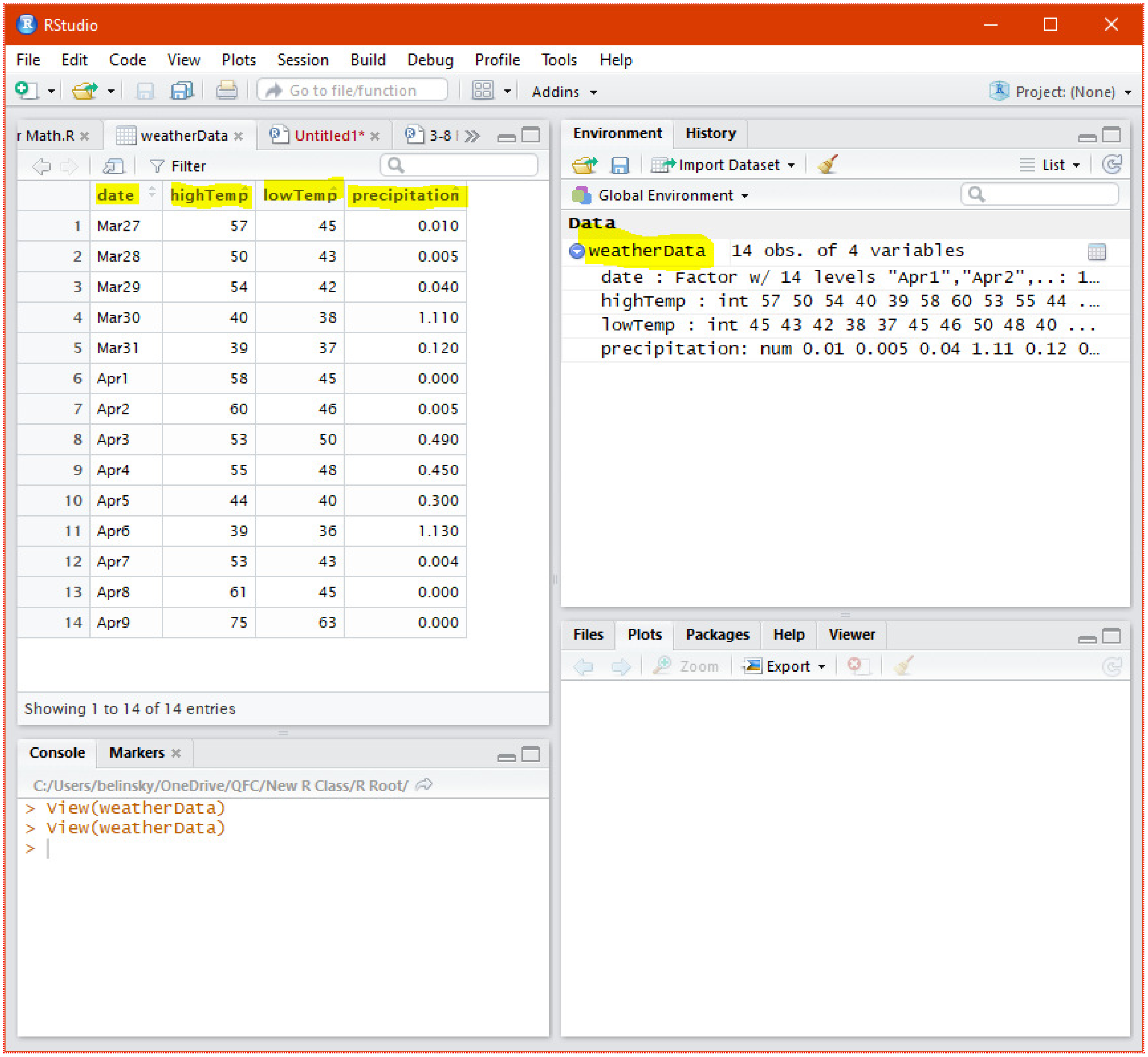
2 - Concepts

3 - Vector math

For this lesson, we are going to use the data from the file **LansingWeather2.csv**. First we need to open the CSV file and save the data in the file to a data frame, which we will call ***weatherData***.

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

  # read data from LansingWeather2.csv and save to the variable weatherData...
  weatherData = read.csv("data/LansingWeather2.csv");
}
```

The data frame, ***weatherData***, has the date, high temperature, low temperature, and precipitation for 14 days (March 27-April 9, 2017) in Lansing, MI.



	date	highTemp	lowTemp	precipitation
1	Mar27	57	45	0.010
2	Mar28	50	43	0.005
3	Mar29	54	42	0.040
4	Mar30	40	38	1.110
5	Mar31	39	37	0.120
6	Apr1	58	45	0.000
7	Apr2	60	46	0.005
8	Apr3	53	50	0.490
9	Apr4	55	48	0.450
10	Apr5	44	40	0.300
11	Apr6	39	36	1.130
12	Apr7	53	43	0.004
13	Apr8	61	45	0.000
14	Apr9	75	63	0.000

Showing 1 to 14 of 14 entries

```
Data
weatherData    14 obs. of 4 variables
   date : Factor w/ 14 levels "Apr1","Apr2",..: 1…
   highTemp : int 57 50 54 40 39 58 60 53 55 44 ...
   lowTemp : int 45 43 42 38 37 45 46 50 48 40 ...
   precipitation: num 0.01 0.005 0.04 1.11 0.12 0…
```

```
C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/
> View(weatherData)
> View(weatherData)
> 
```

*Fig 1: Looking at the value in the **weatherData** vector in the Main Window.*

3.1 - Finding change in temperature

We want to find the change in temperature for each day -- in other words, the high temperature minus the low temperature. Basically this means subtracting each day's low temperature from the high temperature and saving that value to a vector.

First we will extract the low and high temperature values from ***weatherData***. There are two ways we can do this:

1) by column number

```
# get all values from the 3rd column in weatherData
highTemp = weatherData[, 3];
```

2) or by column name

```
# get all values from the column named "lowTemp" in weatherData
```

```
lowTemp = weatherData[, "lowTemp"];
```

Note: the row number is left blank to indicate we are getting values *from all rows.*

We will use a ***for()*** to iterate through each value (i.e., the highTemp - lowTemp for each day) and solve for the temperature difference. To do this we also need a vector that will hold the change in temperature values, which we will call ***changeInTemp***.

```
changeInTemp = c();  #declared a vector
```

changeInTemp acts as a state variable because:
1) ***changeInTemp*** gets initialized before the ***for()***
2) ***changeInTemp*** gets populated during the iterations of the ***for()***
3) changeInTemp final state is the full set of temperature changes

3.2 - Subtracting values from two different vectors

We are going to use a ***for()*** loop that iterates through each value in ***highTemp*** and ***lowTemp***, subtracts the values, and saves the answer to ***changeInTemp***. We need to know the length of the vectors to do this. In this case, we know there are **14** values, but we will use ***length()*** to get this value so that it works for vectors of any size.

```
vectorLength = length(lowTemp);
```

Note: since the length of all columns in a data frame are, by definition, the same, we only need to get the length of one vector.

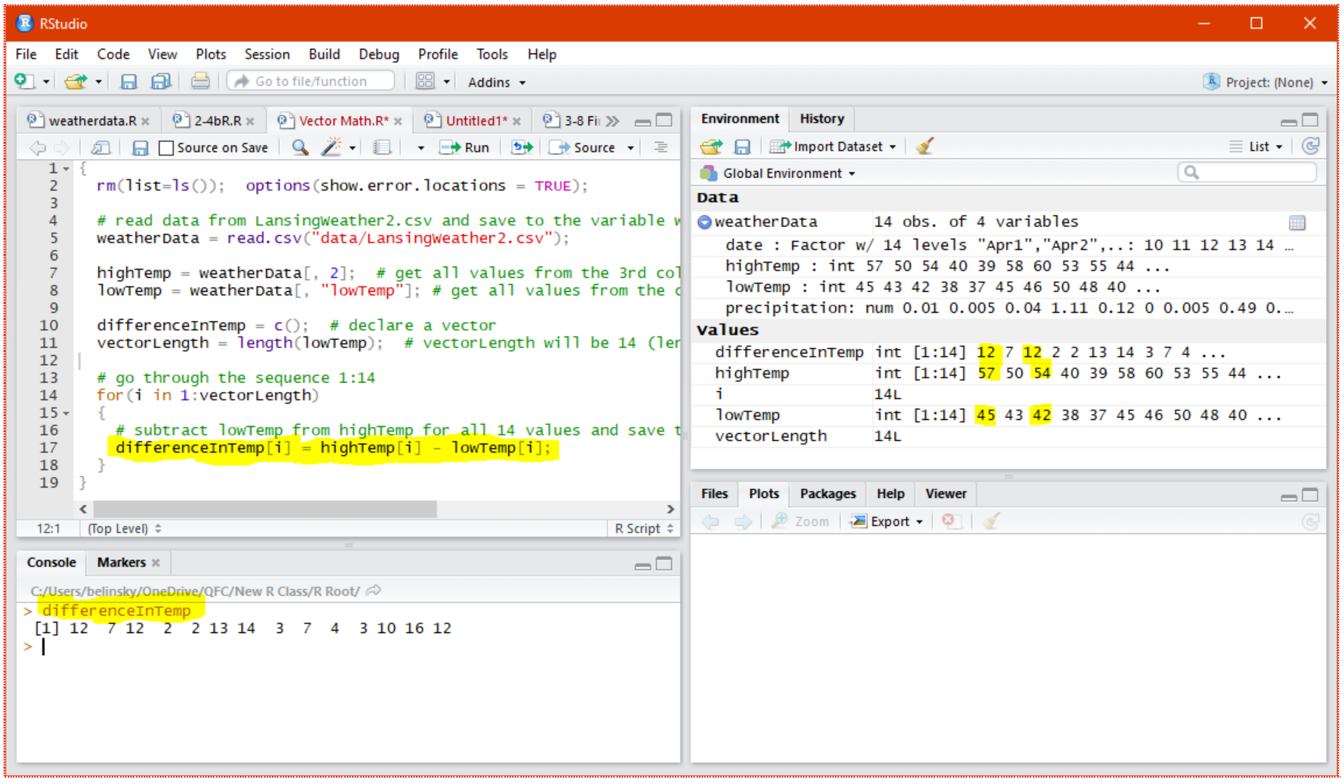
We use ***vectorLength*** to create a sequence that the ***for()*** iterates through. The ***for()*** iterates through the sequence **1:vectorLength***,* or **1:14**, and assigns the value of the **14** ***highTemp*** - ***lowTemp*** operations to the **14** ***changeInTemp*** values.

```
for(i in 1:vectorLength)
{
  changeInTemp[i] = highTemp[i] - lowTemp[i];
}
```

Let's put all this code together:

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

  # read data from LansingWeather2.csv and save to variable weatherData
  weatherData = read.csv("data/LansingWeather2.csv");
  # get all values from the 3rd column in weatherData
  highTemp = weatherData[, 2];
  # get all values from the column named "lowTemp" in weatherData
  lowTemp = weatherData[, "lowTemp"];

  changeInTemp = c(); # declare a vector
  vectorLength = length(lowTemp); # vectorLength will be 14 (length of data)

  # go through the sequence 1:14
  for(i in 1:vectorLength)
  {
     # subtract lowTemp from highTemp for all 14 values and save to changeInTemp
     changeInTemp[i] = highTemp[i] - lowTemp[i];
  }
}
```

The Environment Window shows that the values in the vector ***changeInTemp*** are ***highTemp*** - ***lowTemp*** but the Environment Window only shows up to 10 values. If you want to see all the values in ***changeInTemp***, you can type *changeInTemp* in the Console Window and the 14 values in ***changeInTemp*** will appear on the next line.



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

  # read data from LansingWeather2.csv and save to the variable w
  weatherData = read.csv("data/LansingWeather2.csv");

  highTemp = weatherData[, 2];  # get all values from the 3rd col
  lowTemp = weatherData[, "lowTemp"]; # get all values from the c

  differenceInTemp = c();  # declare a vector
  vectorLength = length(lowTemp);  # vectorLength will be 14 (len

  # go through the sequence 1:14
  for(i in 1:vectorLength)
  {
    # subtract lowTemp from highTemp for all 14 values and save t
    differenceInTemp[i] = highTemp[i] - lowTemp[i];
  }
}
```

```
Data
weatherData        14 obs. of 4 variables
  date : Factor w/ 14 levels "Apr1","Apr2",..: 10 11 12 13 14 …
  highTemp : int 57 50 54 40 39 58 60 53 55 44 ...
  lowTemp : int 45 43 42 38 37 45 46 50 48 40 ...
  precipitation: num 0.01 0.005 0.04 1.11 0.12 0 0.005 0.49 0.…
Values
  differenceInTemp  int [1:14] 12 7 12 2 2 13 14 3 7 4 ...
  highTemp          int [1:14] 57 50 54 40 39 58 60 53 55 44 ...
  i                 14L
  lowTemp           int [1:14] 45 43 42 38 37 45 46 50 48 40 ...
  vectorLength      14L
```

```
C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/
> differenceInTemp
 [1] 12  7 12  2  2 13 14  3  7  4  3 10 16 12
> 
```

Fig 2: Using a ***for()*** *to perform iterative mathematical operations on vectors.*

4 - Dealing with missing values

In the previous example, we have an idealized situation where every day had a high and low temperature associated with it. In the real world, especially with large amounts of data, there is often missing data. In R, missing data is represented by ***NA,*** but the CSV file could designated missing data according to a number of different convention.

First lets create a data set with missing values and save this as ***MissingTemps.csv*** in the ***Data*** directory

```
date,highTemp, lowTemp, precipitation
Mar27,57,45,0.01
```

```
Mar28,50,43,0.005
Mar29,54, ,0.04
Mar30,40,38,1.11
Mar31,39,NA,0.12
Apr1,58,45,0
Apr2,60, ,0.005
Apr3,53,50,0.49
Apr4,55,48,0.45
Apr5,44,40,0.30
Apr6,39,36,1.13
Apr7,NULL,43,0.004
Apr8,61,45,0
Apr9,75,63,0
```

In the above data set, the ***lowTemp*** for **Mar29** and **Apr2** are left blank, the ***lowTemp*** for **Mar31** is given as ***NA,*** and the ***highTemp*** for **Apr7** is given as ***NULL***. So, there are four missing values in ***missingTemps.csv***.

4.1 - Viewing blank or NA values

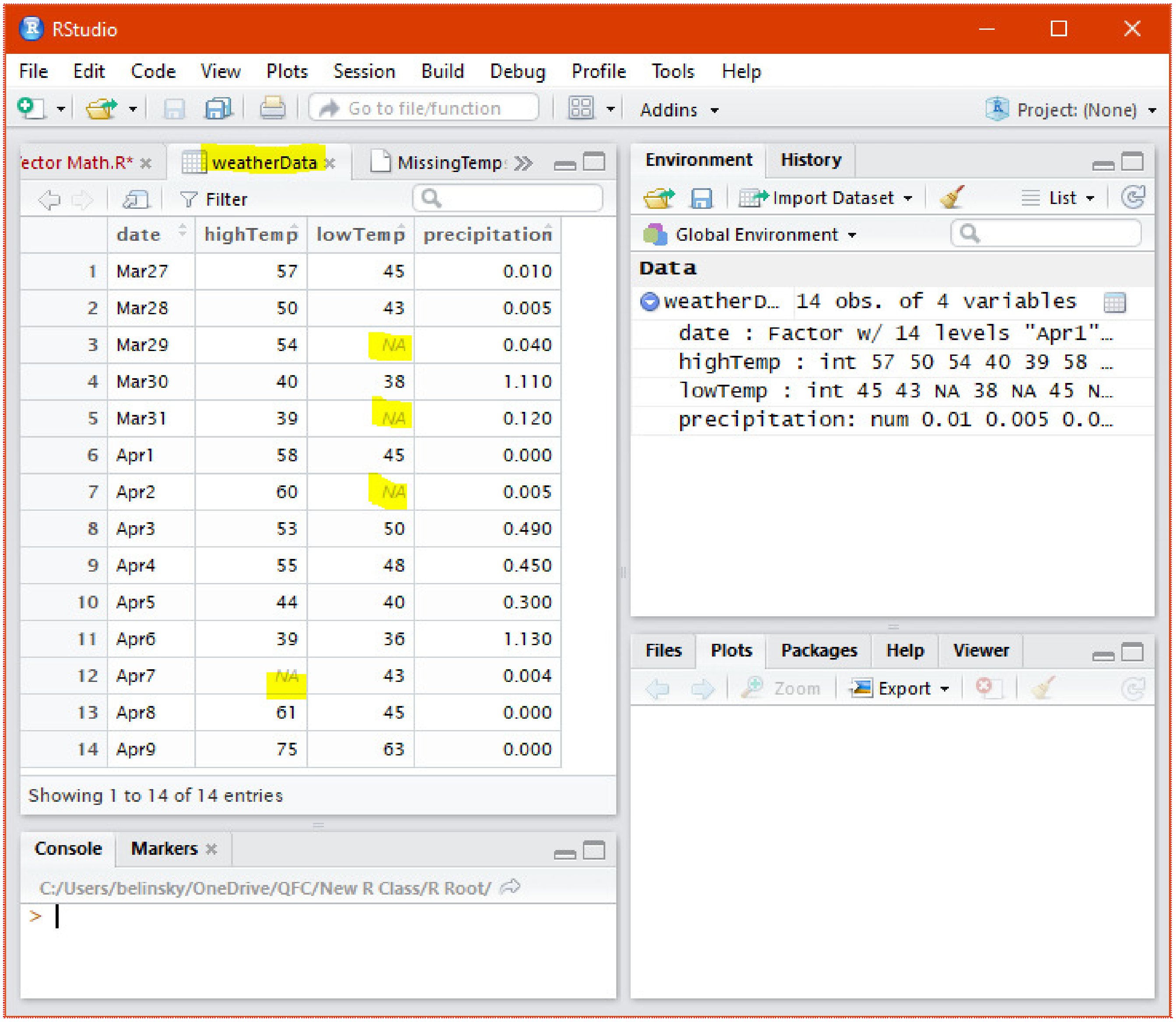
We are going to open a new script and save ***missingTemps.csv*** data to a Data Frame called ***weatherData.*** But, we want to standardized the way that missing values are recorded. So, in ***read.csv()*** we add the parameter ***na.strings***:

```
na.strings=c("", " ", "NULL", NULL, "NA", "na", NA, "null")
```

Essentially, ***na.strings*** is a vector that contains all the value that you want R to assign as ***NA***. It is good to be paranoid here and think of all the possible ways in which CSV files will present these values!

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

  weatherData = read.csv("data/missingTemps.csv",
            na.strings=c("", " ", "NULL", NULL, "NA", "na", NA, "null"));
}
```

Execute the code above and double-click on ***weatherData*** in the ***Environment Window*** so that it appears in the Main Window. We see that all four missing values in ***weatherData*** are labelled as ***NA***.



	date	highTemp	lowTemp	precipitation
1	Mar27	57	45	0.010
2	Mar28	50	43	0.005
3	Mar29	54	NA	0.040
4	Mar30	40	38	1.110
5	Mar31	39	NA	0.120
6	Apr1	58	45	0.000
7	Apr2	60	NA	0.005
8	Apr3	53	50	0.490
9	Apr4	55	48	0.450
10	Apr5	44	40	0.300
11	Apr6	39	36	1.130
12	Apr7	NA	43	0.004
13	Apr8	61	45	0.000
14	Apr9	75	63	0.000

Showing 1 to 14 of 14 entries

```
Data
weatherD…  14 obs. of 4 variables
  date : Factor w/ 14 levels "Apr1"…
  highTemp : int 57 50 54 40 39 58 …
  lowTemp : int 45 43 NA 38 NA 45 N…
  precipitation: num 0.01 0.005 0.0…
```

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

*Fig 3: **NA** values in a data frame*

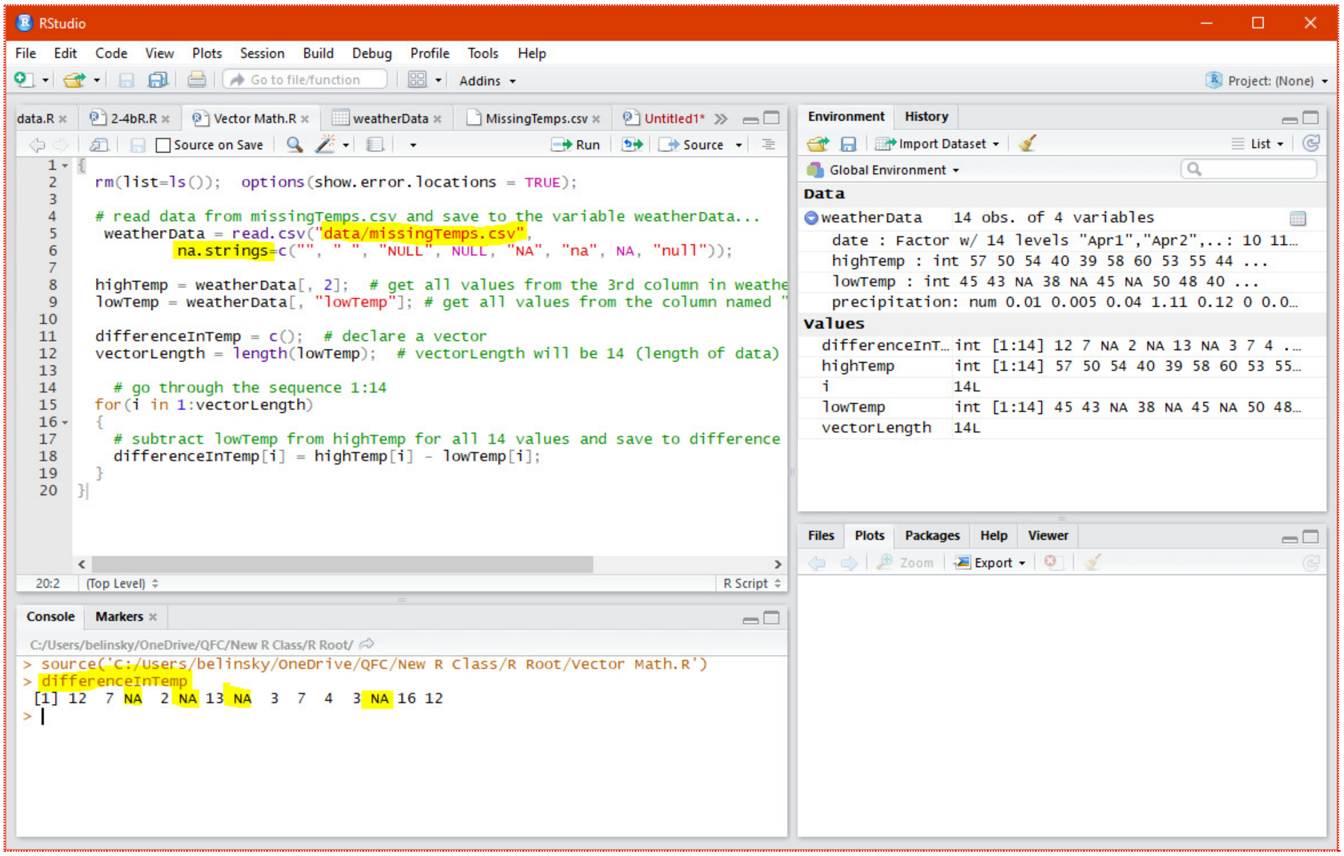
4.2 - Mathematical operation on vectors with NA values

We are going to run the same change in temperature script as before except with NA values in the data frame:

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

  # read data from missingTemps.csv and save to the variable weatherData...
  weatherData = read.csv("data/missingTemps.csv",
       na.strings=c("", " ", "NULL", NULL, "NA", "na", NA, "null"));

  # get all values from the 3rd column in weatherData
  highTemp = weatherData[, 2];
```

```
   # get all values from the column named "lowTemp" in weatherData
   lowTemp = weatherData[, "lowTemp"];

   changeInTemp = c(); # declare a vector
   vectorLength = length(lowTemp); # vectorLength will be 14 (length of data)

   # go through the sequence 1:14
   for(i in 1:vectorLength)
   {
     # subtract lowTemp from highTemp for all 14 values and save to changeInTemp
     changeInTemp[i] = highTemp[i] - lowTemp[i];
   }
}
```

Note: anytime there is an ***NA*** in a calculation (e.g., 3rd element in ***differenceInTemp***) , the answer is going to be ***NA***. This is a special feature of ***NA***, if you used any other value to represent missing data then you would get an error in the calculation.



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

  # read data from missingTemps.csv and save to the variable weatherData...
   weatherData = read.csv("data/missingTemps.csv",
            na.strings=c("", " ", "NULL", NULL, "NA", "na", NA, "null"));

  highTemp = weatherData[, 2];  # get all values from the 3rd column in weathe
  lowTemp = weatherData[, "lowTemp"]; # get all values from the column named "

  differenceInTemp = c();  # declare a vector
  vectorLength = length(lowTemp);  # vectorLength will be 14 (length of data)

    # go through the sequence 1:14
  for(i in 1:vectorLength)
  {
    # subtract lowTemp from highTemp for all 14 values and save to difference
    differenceInTemp[i] = highTemp[i] - lowTemp[i];
  }
}
```

```
Data
weatherData     14 obs. of 4 variables
  date : Factor w/ 14 levels "Apr1","Apr2",..: 10 11…
  highTemp : int 57 50 54 40 39 58 60 53 55 44 ...
  lowTemp : int 45 43 NA 38 NA 45 NA 50 48 40 ...
  precipitation: num 0.01 0.005 0.04 1.11 0.12 0 0.0…
Values
 differenceInT… int [1:14] 12 7 NA 2 NA 13 NA 3 7 4 .…
 highTemp       int [1:14] 57 50 54 40 39 58 60 53 55…
 i              14L
 lowTemp        int [1:14] 45 43 NA 38 NA 45 NA 50 48…
 vectorLength   14L
```

```
C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/Vector Math.R')
> differenceInTemp
 [1] 12  7 NA  2 NA 13 NA  3  7  4  3 NA 16 12
>
```

*Fig 4: Doing mathematical operation with **NA** values*

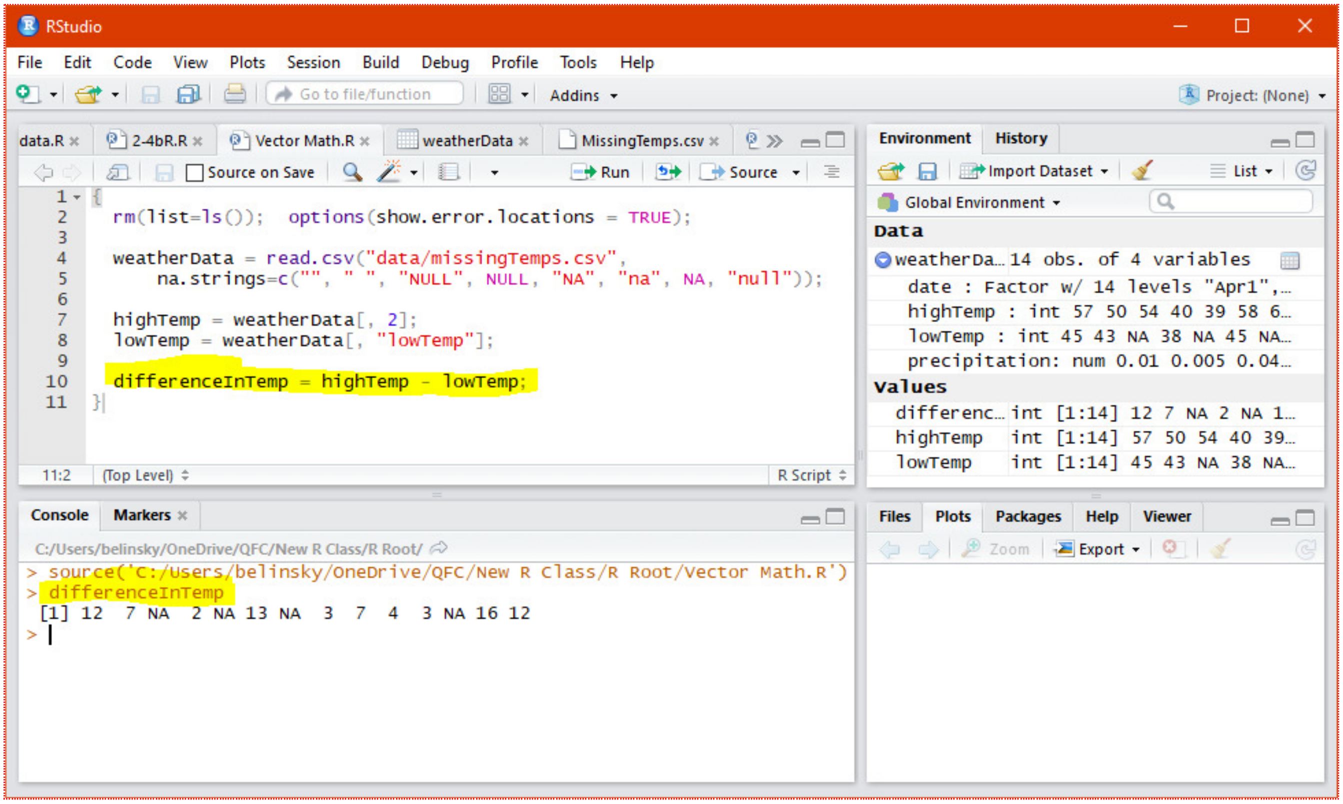
Extension: What counts as NA

5 - Vector Operation Shortcuts

So far we have used ***for()*** loops to perform iterative operations on a vector or multiple vectors. However, R has built in functions that do all these operations using a lot less code. For example, we can perform the above change in temperature calculation without a ***for().***

```
1  {
2    rm(list=ls()); options(show.error.locations = TRUE);
3
4    weatherData = read.csv("data/missingTemps.csv",
5           na.strings=c("", " ", "NULL", NULL, "NA", "na", NA, "null"));
6
7    highTemp = weatherData[, 2];
8    lowTemp = weatherData[, "lowTemp"];
9
10   changeInTemp = highTemp - lowTemp;
11 }
```

Line 10 does all the work of iterating through the values in the vector, subtracting the values, and saving the answer to the vector ***changeInTemp***. The result is the same results as the script with the ***for()*** loop.



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

  weatherData = read.csv("data/missingTemps.csv",
      na.strings=c("", " ", "NULL", NULL, "NA", "na", NA, "null"));

  highTemp = weatherData[, 2];
  lowTemp = weatherData[, "lowTemp"];

  differenceInTemp = highTemp - lowTemp;
}
```

```
Data
weatherDa… 14 obs. of 4 variables
  date : Factor w/ 14 levels "Apr1",…
  highTemp : int 57 50 54 40 39 58 6…
  lowTemp : int 45 43 NA 38 NA 45 NA…
  precipitation: num 0.01 0.005 0.04…
Values
  differenc… int [1:14] 12 7 NA 2 NA 1…
  highTemp   int [1:14] 57 50 54 40 39…
  lowTemp    int [1:14] 45 43 NA 38 NA…
```

```
C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/Vector Math.R')
> differenceInTemp
 [1] 12  7 NA  2 NA 13 NA  3  7  4  3 NA 16 12
>
```

Fig 5: Subtracting a vector in R.

Obviously, this method is easier and more often used than ***for()*** loops for subtracting two vectors. However, this method is really a shortcut and a shortcut specific to the R language. It is important to understand what is going on in with the ***for()*** example because, eventually, you will find a situation where you need to use a ***for()*** to solve a problem.

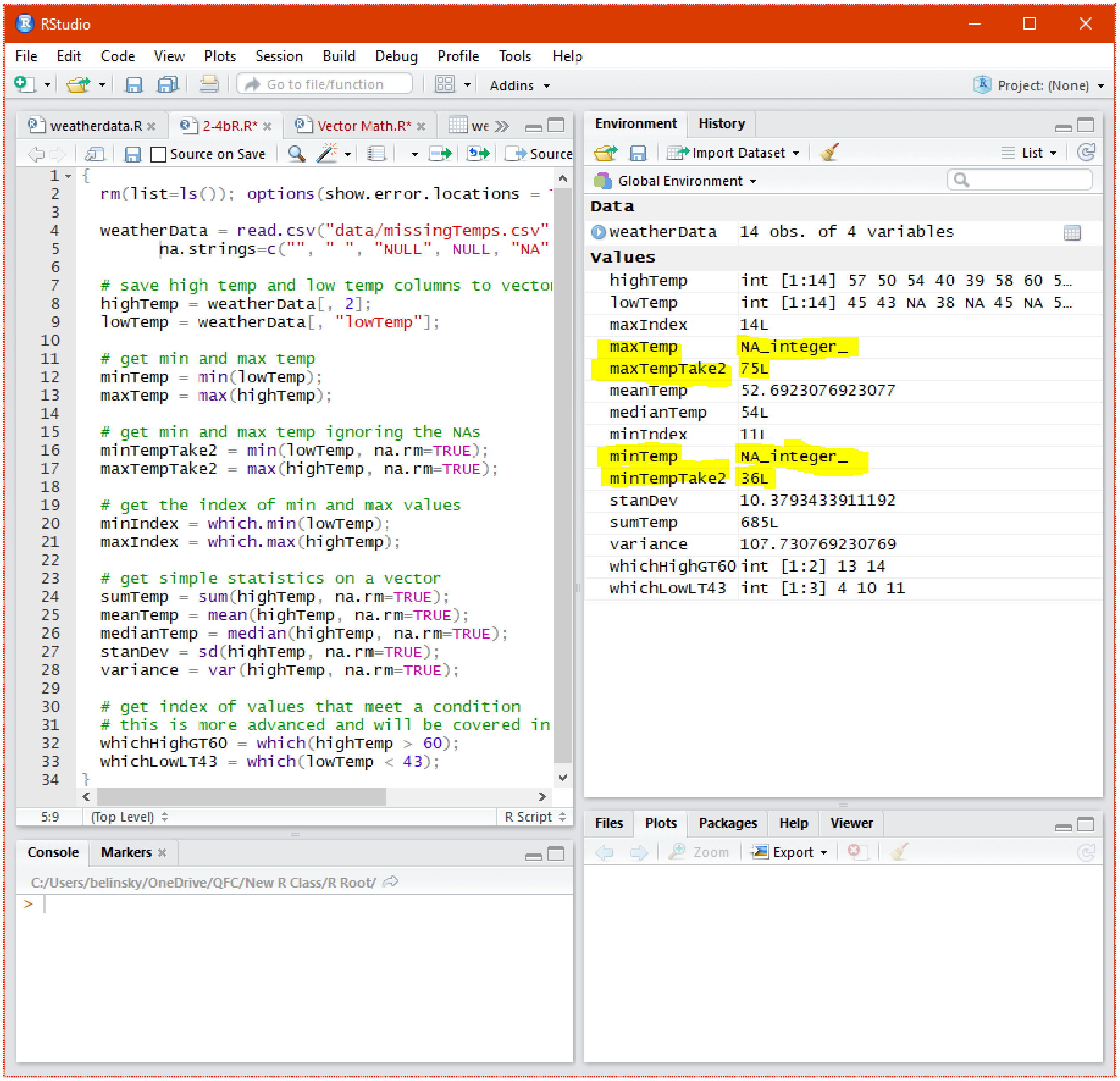
5.1 - Many other shortcuts

R has many functions that can quickly perform the most common operations on vectors like ***sum(), max(), min(),*** and ***mean()***.

One important parameter used in all of these functions is ***na.rm:***
na.rm = TRUE tells R to exclude the ***NA*** values from the vector before performing the operation.
na.rm = FALSE tells R to return ***NA*** if there are ***NA*** values in the vector.

The default value for ***na.rm*** is ***FALSE***.

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

  weatherData = read.csv("data/missingTemps.csv",
          na.strings=c("", " ", "NULL", NULL, "NA", "na", NA, "null"))

  # save high temp and low temp columns to vectors
  highTemp = weatherData[, 2];
  lowTemp = weatherData[, "lowTemp"];

  # get min and max temp (na.rm = default = FALSE)
  minTemp = min(lowTemp);
  maxTemp = max(highTemp);

  # get min and max temp ignoring the NAs
  minTempTake2 = min(lowTemp, na.rm=TRUE);
  maxTempTake2 = max(highTemp, na.rm=TRUE);

  # get the index of min and max values
  minIndex = which.min(lowTemp);
  maxIndex = which.max(highTemp);

  # get simple statistics on a vector
  sumTemp = sum(highTemp, na.rm=TRUE);
  meanTemp = mean(highTemp, na.rm=TRUE);
  medianTemp = median(highTemp, na.rm=TRUE);
  stanDev = sd(highTemp, na.rm=TRUE);
  variance = var(highTemp, na.rm=TRUE);

  # get index of values that meet a condition
  # this is more advanced and will be covered in detail later
  whichHighGT60 = which(highTemp > 60);
  whichLowLT43 = which(lowTemp < 43);
}
```



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

  weatherData = read.csv("data/missingTemps.csv"
        na.strings=c("", " ", "NULL", NULL, "NA"

  # save high temp and low temp columns to vecto
  highTemp = weatherData[, 2];
  lowTemp = weatherData[, "lowTemp"];

  # get min and max temp
  minTemp = min(lowTemp);
  maxTemp = max(highTemp);

  # get min and max temp ignoring the NAs
  minTempTake2 = min(lowTemp, na.rm=TRUE);
  maxTempTake2 = max(highTemp, na.rm=TRUE);

  # get the index of min and max values
  minIndex = which.min(lowTemp);
  maxIndex = which.max(highTemp);

  # get simple statistics on a vector
  sumTemp = sum(highTemp, na.rm=TRUE);
  meanTemp = mean(highTemp, na.rm=TRUE);
  medianTemp = median(highTemp, na.rm=TRUE);
  stanDev = sd(highTemp, na.rm=TRUE);
  variance = var(highTemp, na.rm=TRUE);

  # get index of values that meet a condition
  # this is more advanced and will be covered in
  whichHighGT60 = which(highTemp > 60);
  whichLowLT43 = which(lowTemp < 43);
}
```

```
Data
weatherData     14 obs. of 4 variables
Values
  highTemp      int [1:14] 57 50 54 40 39 58 60 5…
  lowTemp       int [1:14] 45 43 NA 38 NA 45 NA 5…
  maxIndex      14L
  maxTemp       NA_integer_
  maxTempTake2  75L
  meanTemp      52.6923076923077
  medianTemp    54L
  minIndex      11L
  minTemp       NA_integer_
  minTempTake2  36L
  stanDev       10.3793433911192
  sumTemp       685L
  variance      107.730769230769
  whichHighGT60 int [1:2] 13 14
  whichLowLT43  int [1:3] 4 10 11
```

```
>
```

Fig 6: Using R functions to perform common vector operations.

6 - Application

Create a vector called ***changeInHighTemp*** and using the data from **lansingWeather2.csv**, find the change in high temperatures from day-to-day. Add the vector to the data frame -- this means the vector must be the same size as the other columns.

So if you have four high temperatures: 40, 45, 35, 42
changeInHighTemp would be: ***NA***, 5, -10, 7

The NA says that there is not enough day to give the change in temperature for the first day.

7 - Extension: What counts as NA

R applies some intelligence when it moves data from a CSV to a data frame. If there is am empty value in a column that is all numbers, then R will convert the empty value to ***NA***.   ***NULL***, however, is not considered to be a valid data frame value and if there is a ***NULL*** in a column, it will be treated as the string value "NULL". R also does not consider lowercase ***na*** to be ***NA*** and this will also be treated as the string value "na".

There is now a second problem: if there is even one string in a column, all values in the column are treated as strings. This means that **40** will be seen as the string "40" and mathematical operations on strings will cause an error.