03-04 t-tests and ANOVAs

1 - Purpose

- Perform a t-test on temperature data from two different years
- Set up data frame for use with an ANOVA
- Perform an ANOVA on temperature data from multiple years
- Display temperatures in a boxplot

2 - Class Project 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

You can get the script for this lesson here. The lesson goes in order on the script. In RStudio, you can comment out large parts of the script using Control-Shift-C (Windows) or Command-Shift-C (MAC).

You will also be need the data file, ***lansingJanTemps.csv***, created in lesson 3-3. If you do not have this file, download it here and *save it to your data folder*.

4 - Comparing means using a t-test

We will be using the matrix we saved to ***lansingJanTemps.csv*** from the last lesson. So, first we need to read in the file, ***lansJanTemp.csv***, and save the data to a variable.

```
1 lansJanTempsDF = read.csv(file = "data/lansingJanTemps.csv");
```

By default, ***read.csv()*** saves the data in the csv file as a data frame, whereas we want to data to be in matrix. We can convert the data frame into a matrix because all the values in the data frame are numeric. To convert the data frame to a matrix, we use the function ***as.matrix()***. The only parameter we need to use is ***x***, which is the data frame.

```
1 lansJanTemps = as.matrix(x=lansJanTempsDF);
```

4.1 - Creating a boxplot of the data

Let's first create a boxplot for each of the six years of January temperature using ***boxplot()***. In other words, we want one boxplot for each column in the ***lansJanTemps*** matrix.

There are two parameters we will use:

- ***x***: the data

- ***use.cols***: if ***TRUE***, group data by columns; if ***FALSE***, group data by rows.

```
boxplot(x=lansJanTemps, use.cols = TRUE);
```

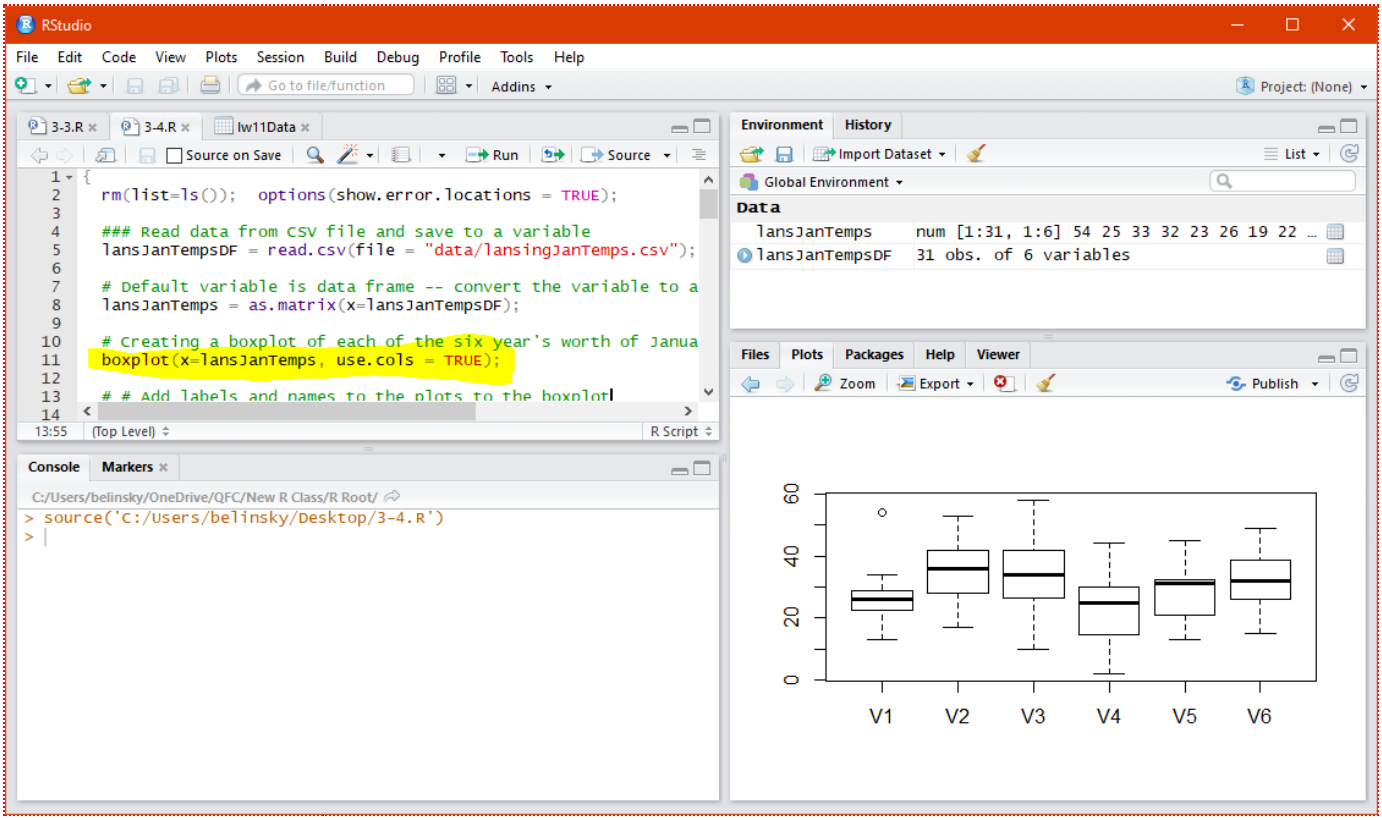


Fig 1: Boxplot of January temperatures.

The parameter ***use.cols*** has a default value of ***TRUE***. So ***boxplot()*** would do the same thing if you left ***use.cols*** out. If you set ***use.cols*** = ***FALSE***, then ***boxplot()*** will produce one boxplot for each row. In other word, you will get a boxplot for each of the 31 days in January. Try it

4.2 - Adding labels to the boxplot

We are going to add a few more parameters to the ***boxplot()*** call:

- ***main***: title of the plot
- ***xlab***, ***ylab***: labels on the x and y axes
- ***names***: a vector that holds the names of the boxplots -- in this case, six values for the six displayed boxplots

```
boxplot(x = lansJanTemps, use.cols = TRUE,
        names = c(2011,2012,2013,2014,2015,2016),
        main = "January 2011 through January 2016",
        xlab = "Years",
        ylab = "Temperature (Fahrenheit)"
);
```

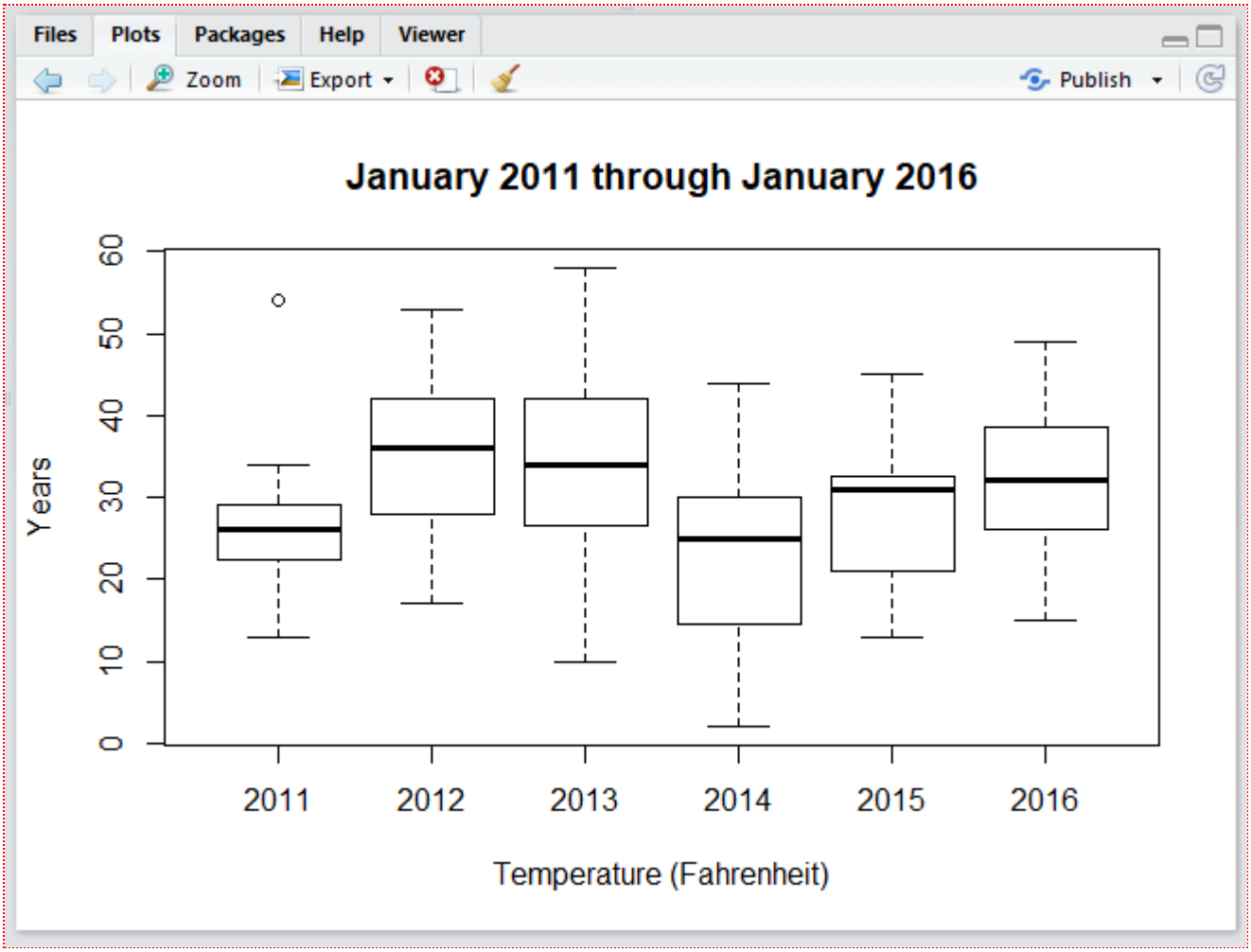


Fig 2: Boxplots of January temperatures by year with labels.

The boxplots suggest quite a bit of variation in temperature between the years. 2014 seems to be the coldest year and 2012 is the warmest year. 2011 seems to have the least variation.

Reminder: You can use the arrow keys in the upper-right of the Plot Window to cycle through the plots.

5 - t-tests

Let's be more formal about analyzing the data and do a *two-sample t-test.* In statistics, a two-sample t-test is used to determine if there is evidence that the mean from two different sets of values, in this case columns (years) in the matrix, are statistically different.

We are going to compare the means from January 2013 (column 3) and January 2016 (column 6). To perform a t-test between these two years, we call the function ***t.test()*** and use the parameters ***x*** and ***y***, which are vectors (i.e., the columns in the matrix we are comparing). In this case, columns 3 and 6 represent the years 2013 and 2016, respectively.

```
1 tTest1 = t.test(x=lansJanTemps[,3], y=lansJanTemps[,6]);
```

In the Environment Window, ***tTest1*** is labelled as a "List of 9". This is not the most intuitive way to read the results. We can get a much more helpful summary of the results using the function ***print()*** and passing it the results of the t-test using the parameter ***x***.

```
print(x=tTest1);
```

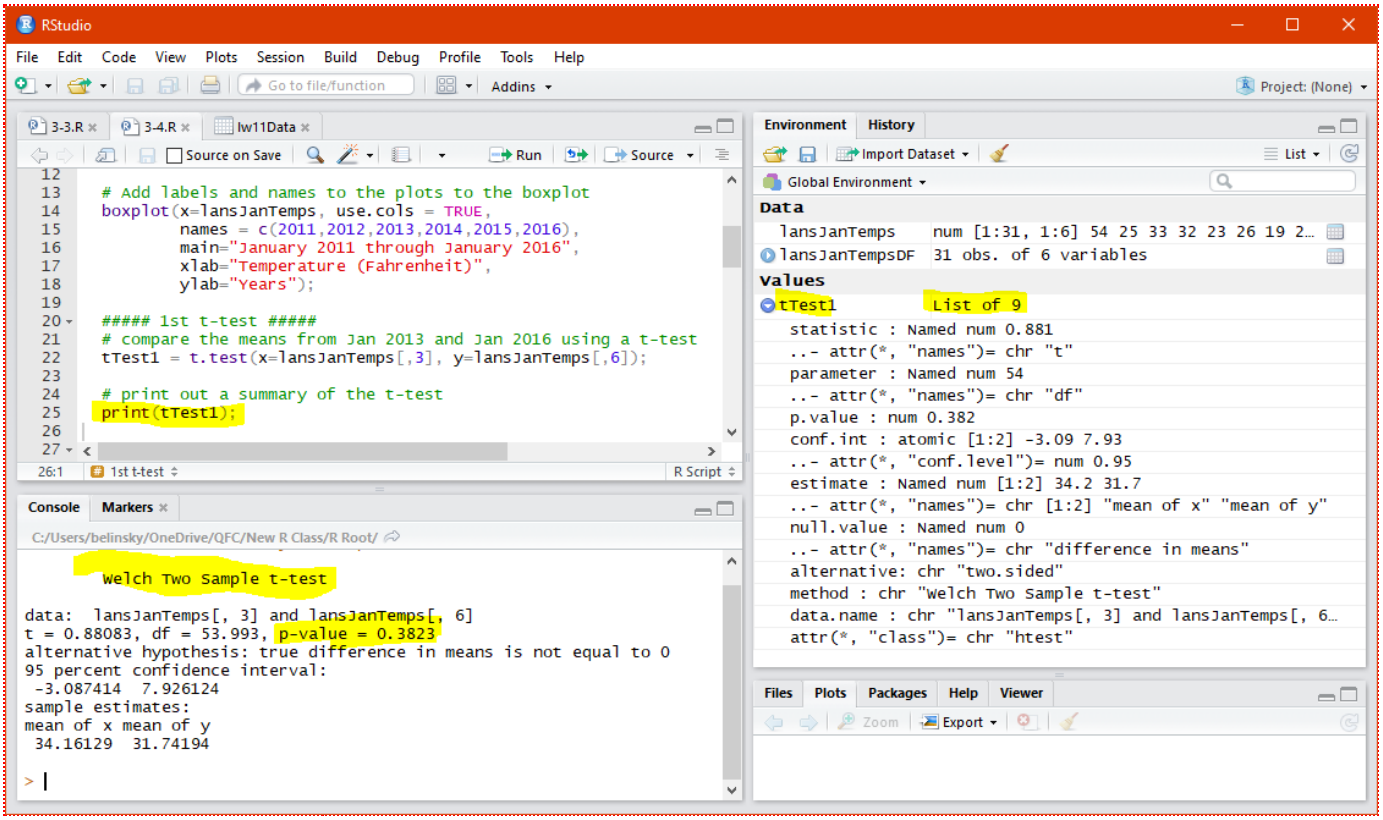


Fig 3: Performing a t-test and printing the summary of the t-test

In the summary of the t-test it is stated that

1. The mean of ***x*** (Jan 2013) is **34.2** and the mean of ***y*** (Jan 2016) is **31.7**
2. If the "experiment" could be repeated, we would expect to see a difference of this size between the means **38.23%** of the time (from the p-value of 0.3823)
3. We are 95% sure that the means of ***x*** and ***y*** have a difference between **-3.09** and **7.93**
4. We would not reject the null hypothesis that the means of ***x*** and ***y*** are equal

5.1 - A second t-test

We will do a second t-test between years that do not look as similar -- 2012 (column 2) and 2014 (column 4):

```
tTest2 = t.test(x=lansJanTemps[,2], y=lansJanTemps[,4]);
print(x=tTest2);
```

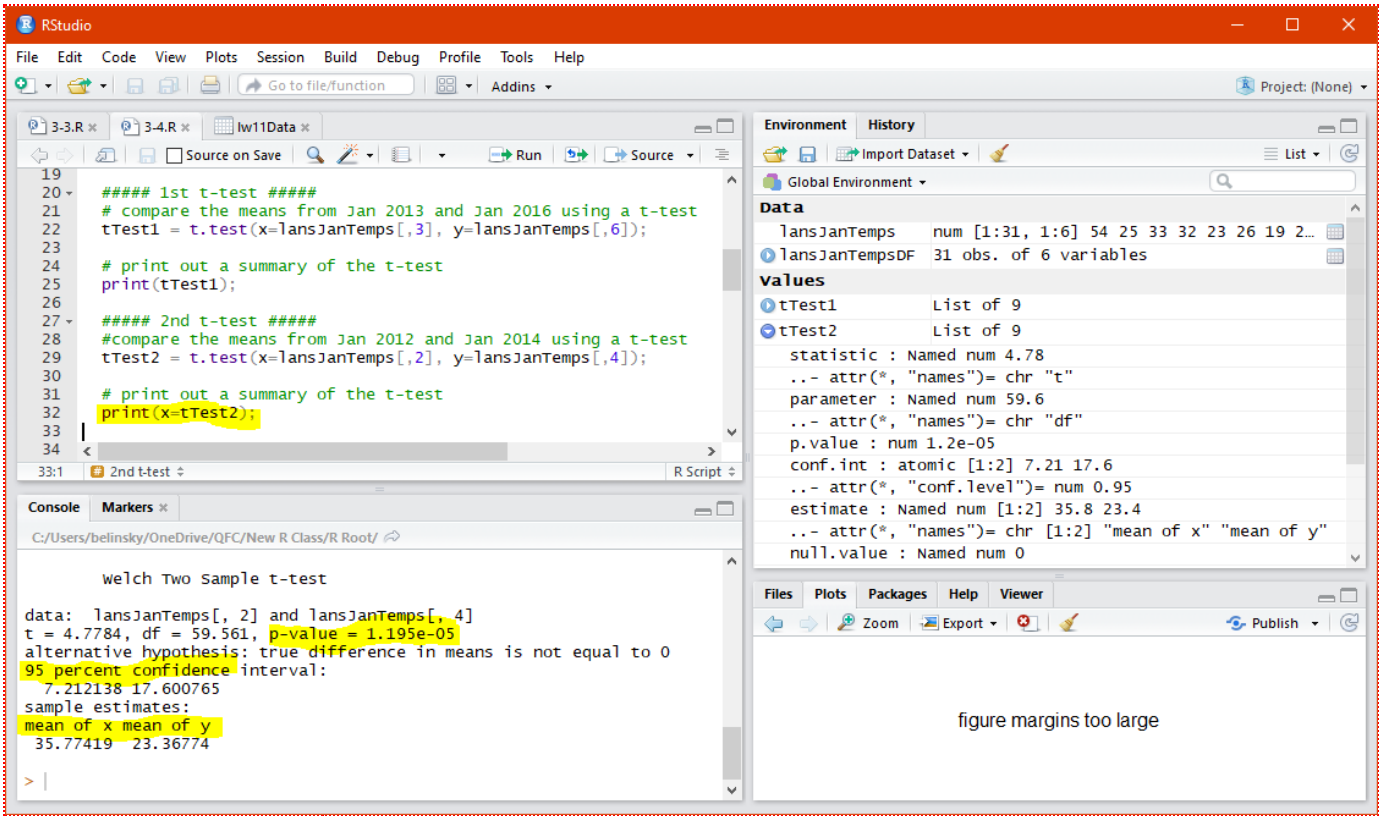


Fig 4: Performing a second t-test and printing the summary of the t-test.

In the summary of the t-test it is stated that

1. The mean of ***x*** (Jan 2012) is **35.8** and the mean of ***y*** (Jan 2014) is **23.4**
2. If the "experiment" could be repeated, we would expect to see a difference of this size between the means **0.001195%** of the time (from the p-value of 1.195e-05 or 0.00001195)
3. We are 95% sure that the means of ***x*** and ***y*** have a difference between **7.21** and **17.6**
4. In this case, we would reject the null hypothesis that the means of ***x*** and ***y*** are equal.

Note: the "*figure margins too large*" message in the Plot Window. This happens when the Plot Window is sized too small to hold a plot (the plot was from earlier in the lesson). This script error given is: *Error in plot.new() (from 3-3b.R#11) : figure margins too large*

6 - ANOVA using a data frame

ANOVAs, while similar functionally to a t-test, require that data be structured differently. ANOVAs compare values in one column from a data frame (e.g., temperatures) based on groupings in another column (e.g., years). The first column contains the data and the second column contains the groupings (also called *categories, factors*).

In our case, the data are the temperatures whereas the groupings are the years. So, we want to create a data frame with two columns -- one that has the temperature values and one that has the associated year.

Note: For now, we are going to create data frames that only have the values we need to perform the ANOVAs. Later we will learn how to perform an ANOVA on a portion, or subset, of a data frame.

6.1 - Setting up the data frame

Let's perform an ANOVA on the two years 2012 and 2014 -- just like the last t-test. Note: typically we would use more than two factors in an ANOVA, which we will do later.

The data frame will have two columns:
1. a vector with the temperatures values from the years 2012 and 2014
2. a vector with the associated grouping/category (either 2012 or 2014)

Temp	Year
##	2012
##	2012
...	...
##	2012
##	2014
...	...
##	2014

To create a vector of temperature values from both 2012 and 2014, we just need to create a vector that combines the second and fourth columns of our matrix:

```
JanTemps1 = c(lansJanTemps[,2], lansJanTemps[,4]);
```

For the grouping vector, we want the first 31 values to be 2012 and the second 31 values to be 2014. We could create a vector with all 62 values:

```
c(2012,2012,2012,....,2014,2014,2014);  # Don't do this!
```

but that can be a bit painful.

6.2 - the repeat function: rep()

A shortcut to creating the year vector is to use the ***rep()*** function. ***rep()*** takes two parameters:
- ***x***: the value(s) to repeat
- ***times***: the number of times repeat it the value(s)

We will test ***rep()*** in the Console Window -- this is a good way to test out code that you do not necessarily want in your main script.
The following ***rep()*** commands will repeat "*llama*" 5 times and *8754* 9 times

```
rep(x="llama", times=5);  # create a vector with 5 "llama"

rep(x=8754, times=9);     # create a vector with 9 "8754"
```

We can also use ***rep()*** to create a vector of vectors. In this case, we are combining 4 "*a*", 5 "*b*", and 2 "*c*" into one vector.

```
c(rep("a", 4), rep("b", 5), rep("c", 2));
```

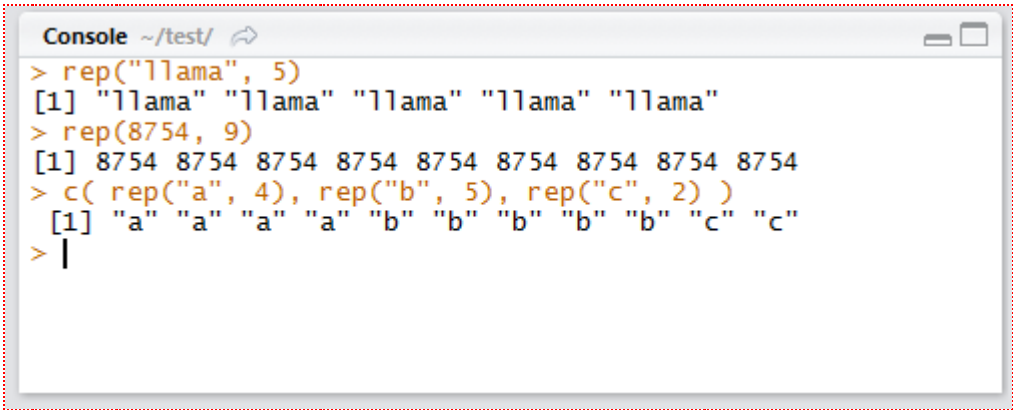


*Fig 5: Using the **rep()** function to repeat values*

So, to create the category vector with 31 ***2012*** and 31 ***2014***:

```
1 JanYears1 = c(rep(2012,31), rep(2014,31));
```

Note: for the above example, I took out the parameter names. This can be done if the parameters are given in the correct order.

*Extension: the **each** parameter*

6.3 - Combine temperature and year vectors into a data frame

Now we want to create a data frame with the vector that has the data (***JanTemps1***) and the vector that has the categories (***JanYears1***).

```
1 JanDataFrame1 = data.frame(JanTemps1, JanYears1);
```

Extension: parameters names for data frame columns.

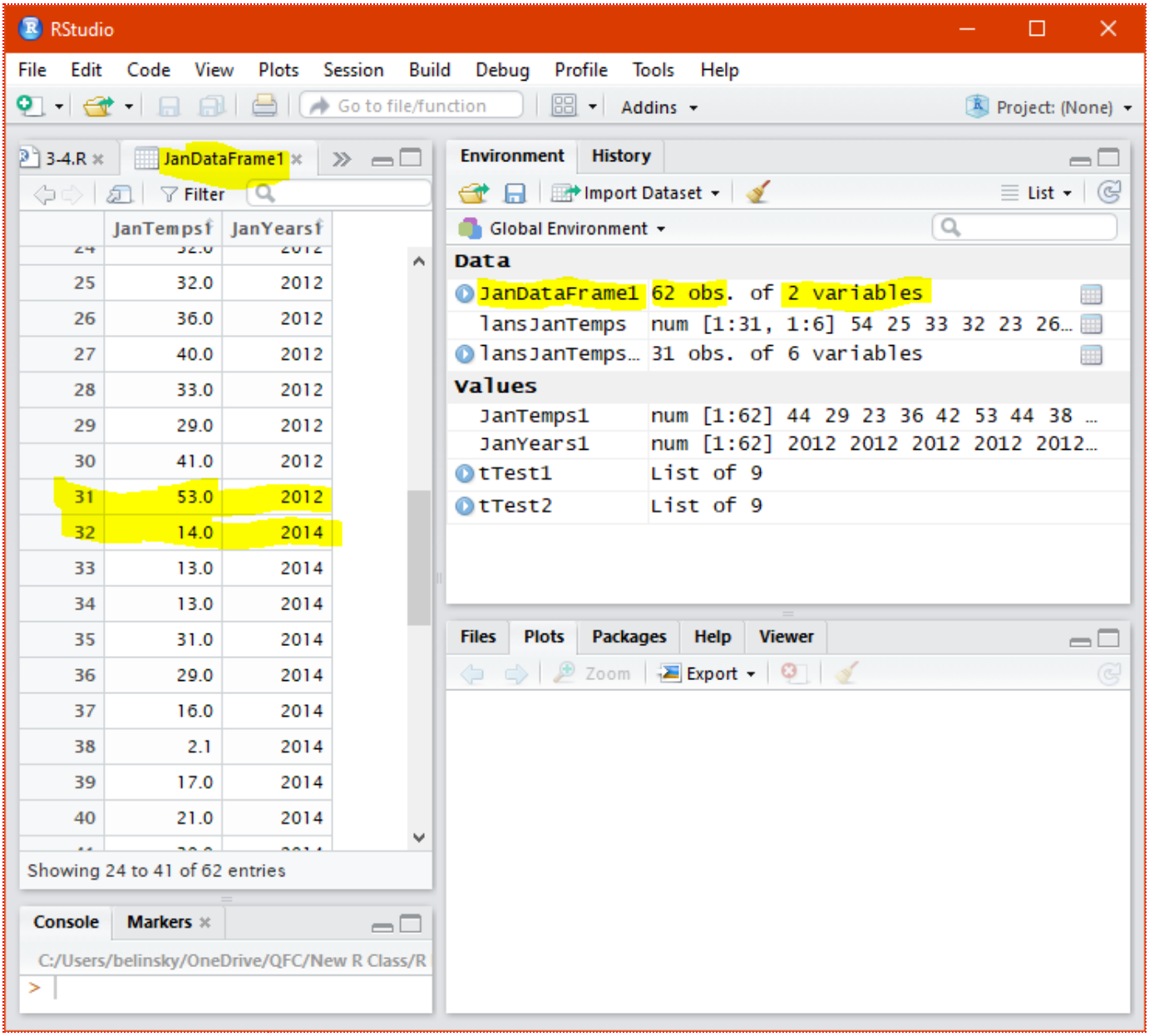


Fig 6: Data frame with values and grouping columns (temperature and year)

6.4 - Boxplot using values and categories

Before we run our ANOVA, let's look at the data again. We can create a boxplot using this data frame like we did using the matrix, but the setup for the parameters for ***boxplot()*** is different. In this case, we need to use the *formula* style.

The parameters for this type of ***boxplot()*** are:

- ***data:*** data frame the boxplot is getting data from
- ***formula***: ***y~x***
  - ***y*** is the column from the data frame that goes on the ***y-axis*** of the boxplot (temperatures)
  - ***x*** is the column from the data frame that goes on the ***x-axis*** of the boxplot (years)

```
1 boxplot(formula=JanTemps1~JanYears1, data=JanDataFrame1);
```

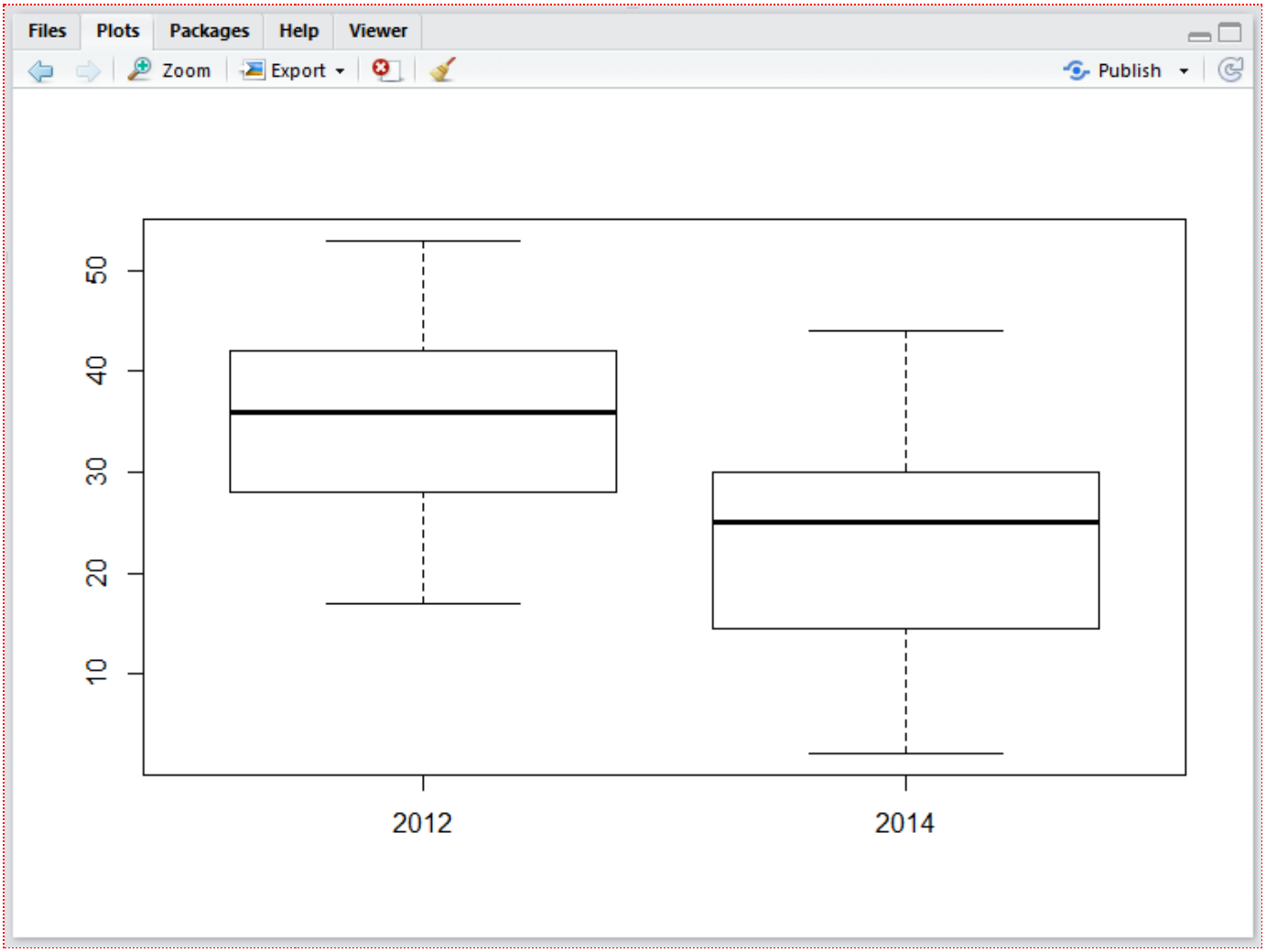


Fig 7: Boxplot of the January temperatures (data) broken down by year (category)

7 - Performing an ANOVA

We will perform an ANOVA on the data frame using the function ***aov()***-- in other words, we are going to check whether the 2 different months of January temperatures are likely to be from the same distribution. We use the same parameters for ***aov()*** as we did for the previous ***boxplot()***.

```
1 JanAnova1 = aov(formula=JanTemps1~JanYears1, data=JanDataFrame1);
2 print(summary(JanAnova1));
```

Just like with t-tests, the results are stored to a list that is not very readable. To summarize the results with t-test we used the function ***print()***. To summarize the important results for an ANOVA we use ***print(summary()).***

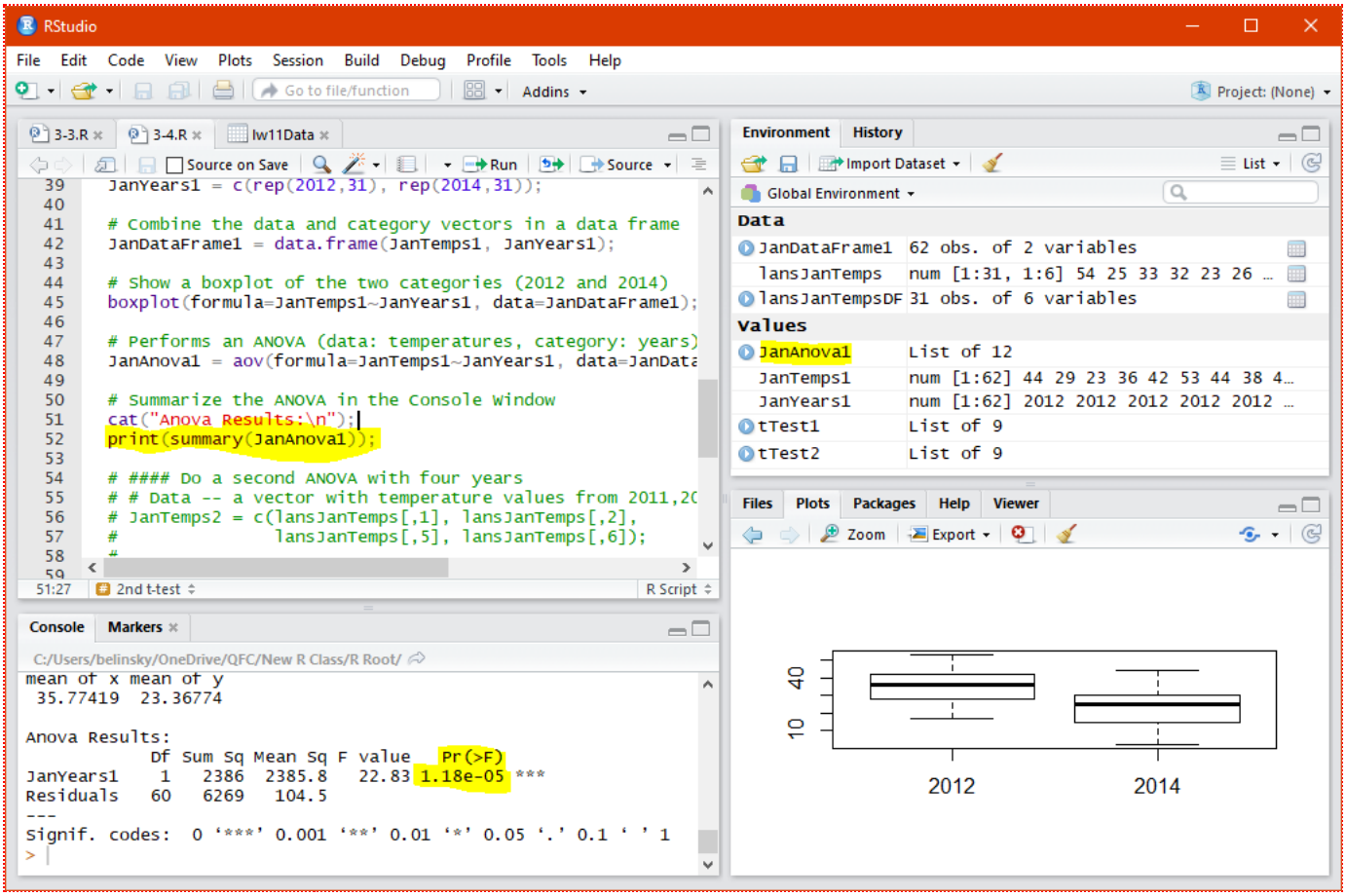


Fig 8: ANOVA of the temperatures from 2012 and 2014

The results of the ANOVA show the probability that the temperatures come from the same distribution is 1.18e-05 (0.0000118). This, as expected, is the exact same probability that we got from doing the t-test between 2012 and 2014.

7.1 - A Second ANOVA

We are going to perform a second ANOVA on 4 years worth of January temperatures. The steps will be the same as above.

Create a data vector with the temperature values from 2011, 2012, 2015, and 2016 (columns 1, 2, 5, and 6)

```
JanTemps2 = c(lansJanTemps[,1], lansJanTemps[,2],
              lansJanTemps[,5], lansJanTemps[,6]);
```

Create a category vector with 31 2011s, 2012s, 2015s, and 2016s:

```
JanYears2 = c(rep(2011,31), rep(2012,31),
              rep(2015,31), rep(2016,31));
```

Combine the data and category vector in a data frame:

```
JanDataFrame2 = data.frame(JanTemps2, JanYears2);
```

Perform an ANOVA -- the formula is temperature (data) vs year (category) and the data is from the data frame just created

```
JanAnova2 = aov(JanTemps2~JanYears2, data=JanDataFrame2);
```

Summarize the ANOVA in the Console Window

```
print(summary(JanAnova2));
```

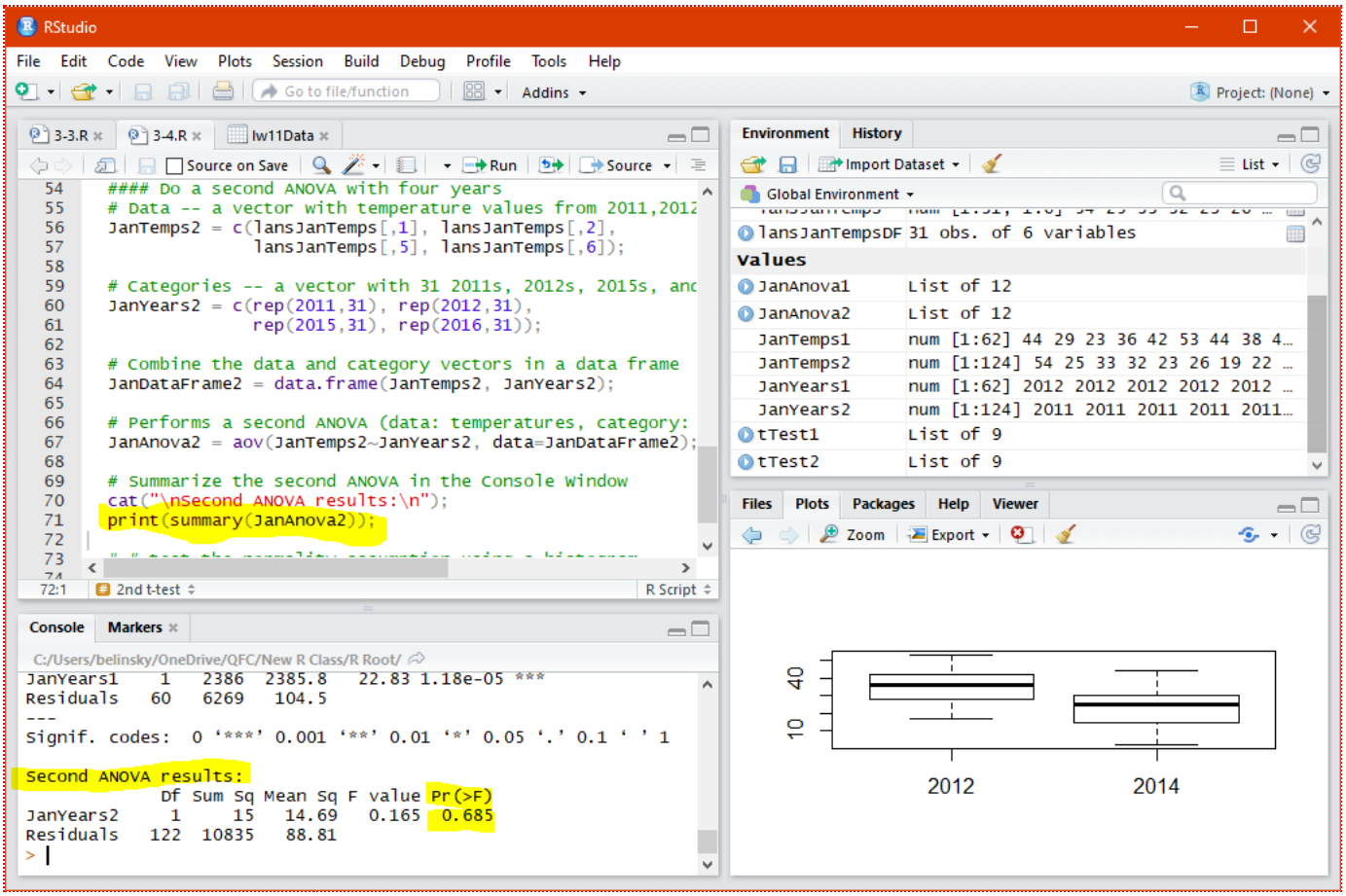


Fig 9: ANOVA on the temperatures from 2012, 2012, 2015, and 2016

The results of the ANOVA show that the probability of the annual temperatures coming from the same distribution is **0.685**, or **68.5%**.

7.2 - Histogram on the residuals

Lastly, we can perform a histogram on the residuals of the ANOVA to show the we have not violated normality assumptions.

The ***residuals()*** function gets the residuals from ***JanAnova*** and ***hist()*** plots the residuals.

```
hist(x=residuals(JanAnova2));
```

... and this histogram looks fairly normal so we probably did not violated the normality assumption.

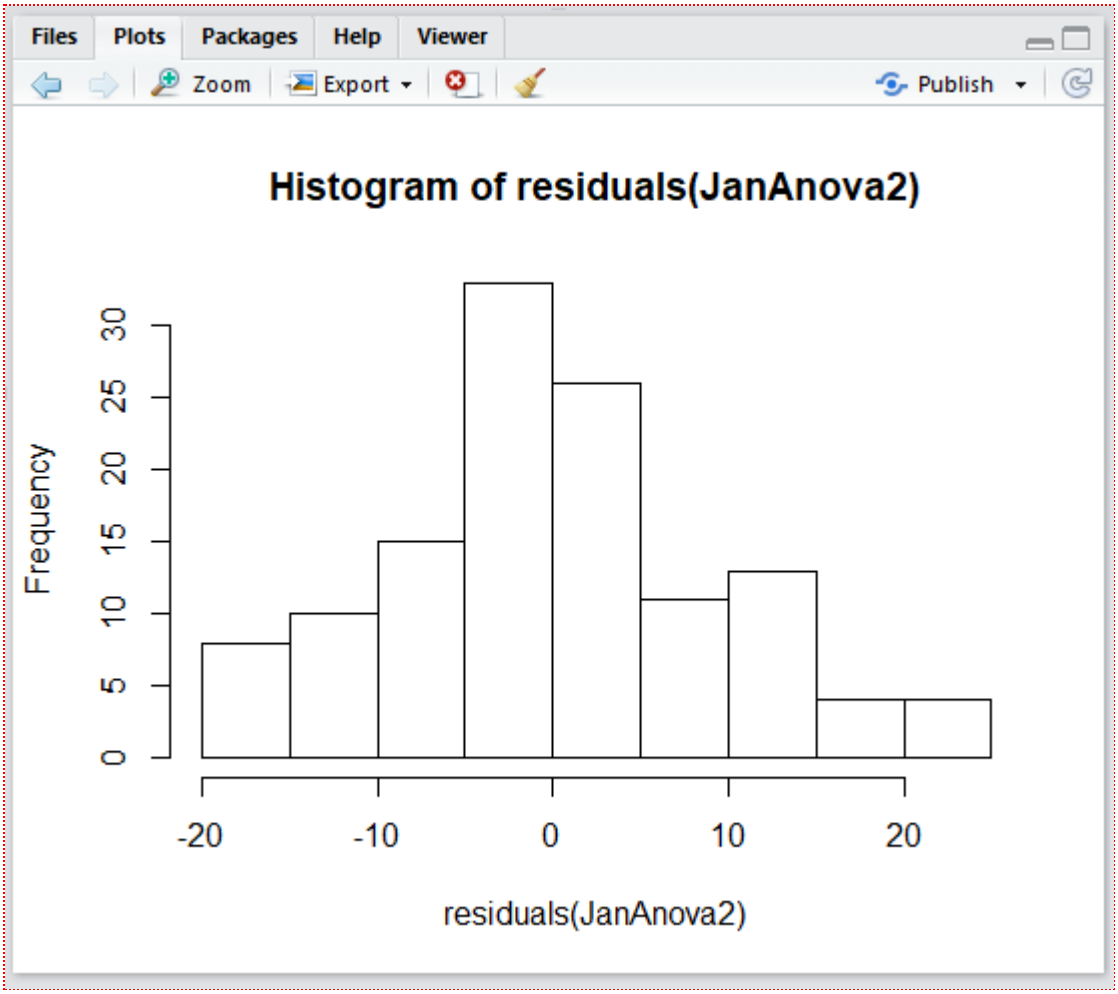


Fig 10: Histogram of the residuals from the second ANOVA

8 - 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.

1) Using the data from NOAA/NCDC obtained in this lesson: Find out which years' temperatures are most statistically similar (using t-tests) to the temperatures from 2014 (other than 2014).

2) Perform an ANOVA using all 6 months.
- Create a boxplot from the data frame that shows each of the years
- Add labels to the boxplot that give the year for each plot
- Make a histogram of the residuals from the ANOVA
- Save the data frame created in part 2 to a file called JanTempDF.csv -- we will use this file later.

*Save you script file as **app3-4.r** in the **scripts** folder of your RStudio Project for the class.*

9 - Extension: parameters names for data frame columns

When we look at the parameters for a data frame, we see that the data is entered in a different format from all other parameters, signified by the (...)

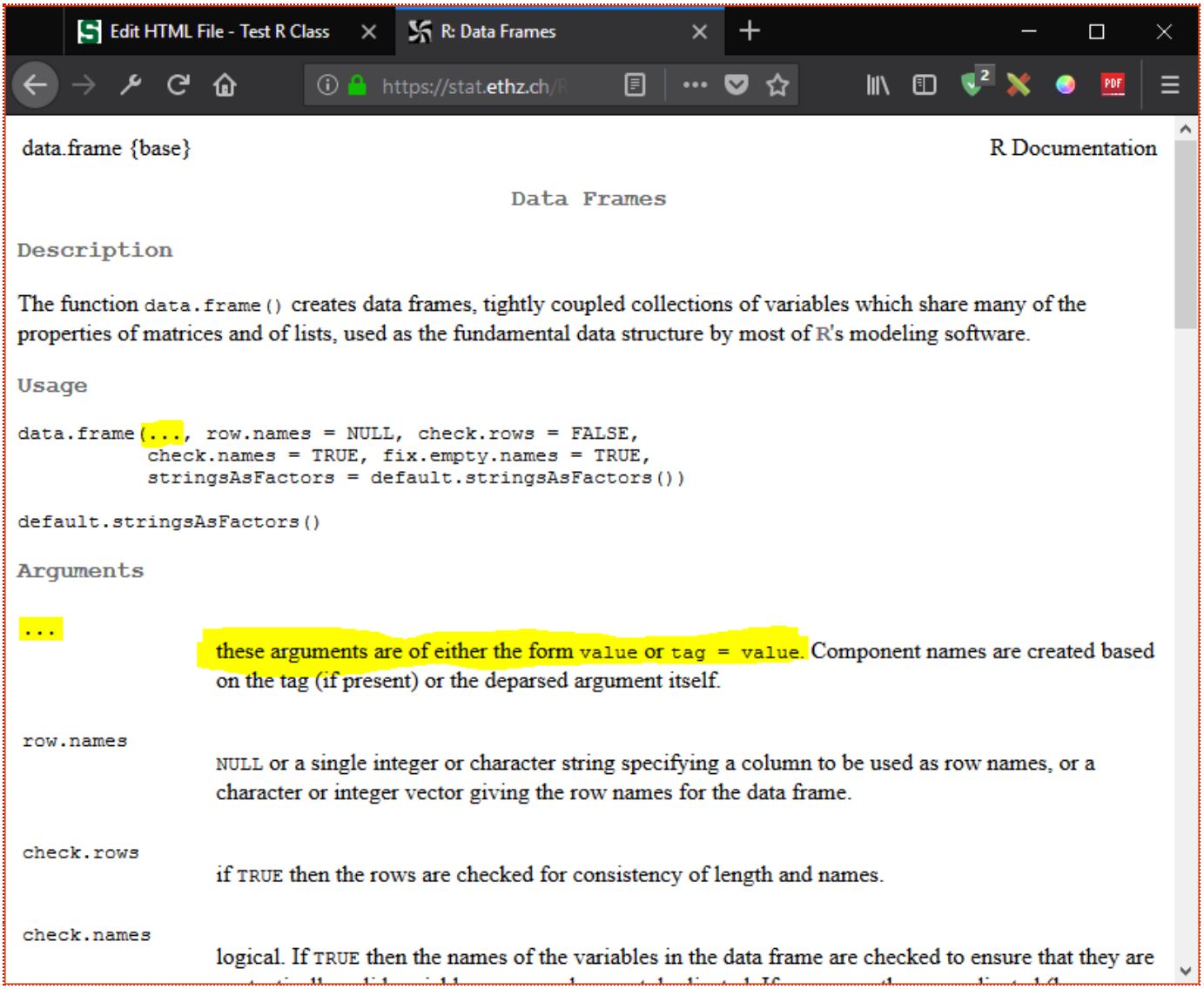


Fig 11: From the R documentation: data for a data frame is part of the (...) in ***data.frame()***

In the case of a data frame, *tag = value* (*Fig.11*) means you can use "parameter" name as the column header.

So, instead of building the data frame like this:

```
1 JanDataFrame1 = data.frame(JanTemps1, JanYears1);
```

You could do this:

```
1 JanDataFrame1 = data.frame(temperatures = JanTemps1, years = JanYears1);
```

and the data frame will have the column header ***temperatures*** and ***years*** instead of ***JanTemps1*** and ***JanYears1***

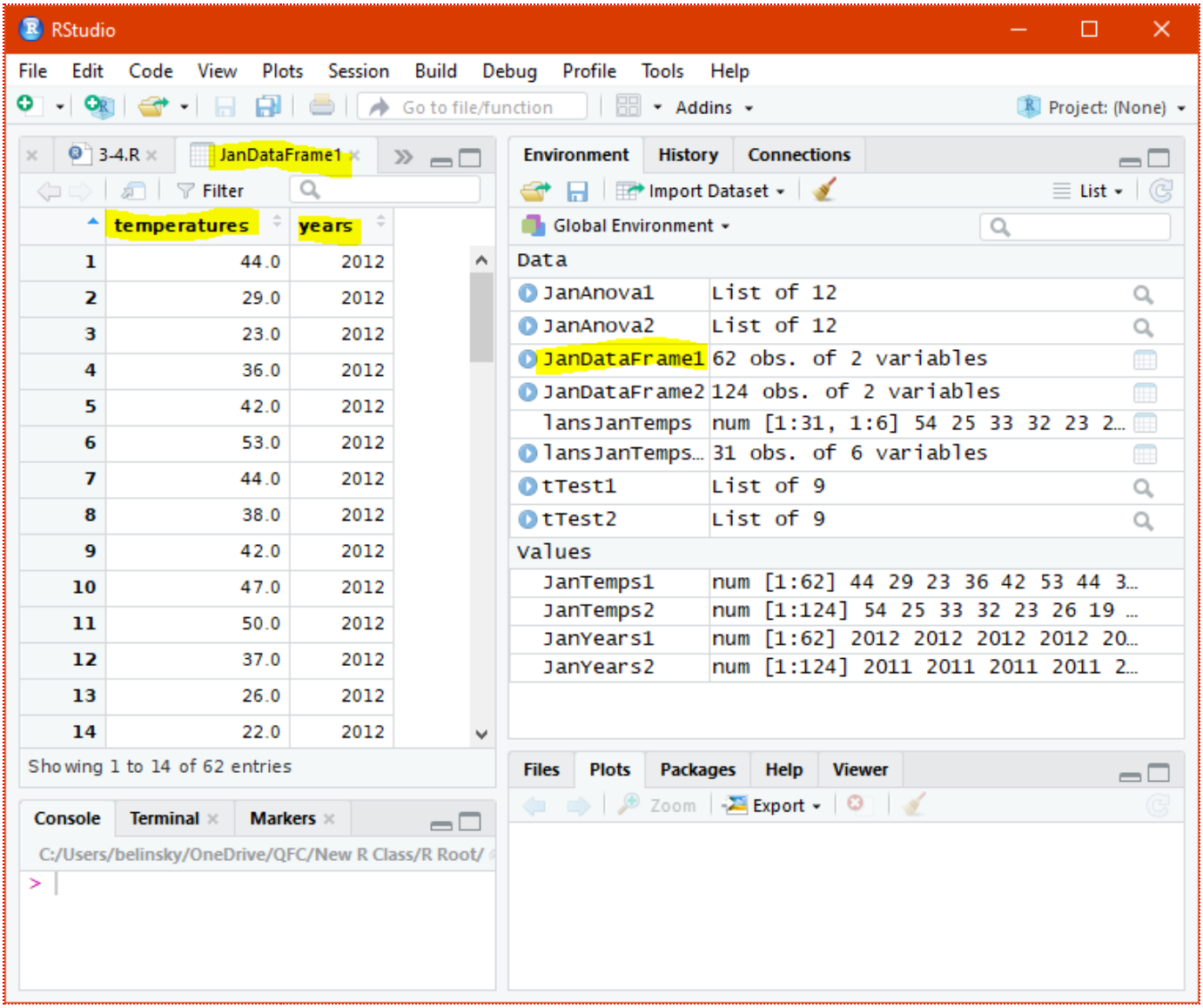


*Fig 12: Changing the column headers using **data.frame()***

10 - Extension: the *each* parameter

The line of code:

```
1 JanYears1 = c(rep(2012,31), rep(2014,31));
```

can be equivalently coded as:

```
1 JanYears2 = rep( c(2012, 2014), each = 31);
```

which can be read as "repeat the values in the vector 31 times"

There is a small advantage to using this method when you have multiple values you want to repeat the same number of times.

For instance, if we wanted to repeat the years **2010** to **2017** 31 times then we could explicitly use ***rep()*** 8 times:

```
1 JanYears3 = c(rep(2010,31), rep(2011,31), rep(2012,31), rep(2013,31),
```

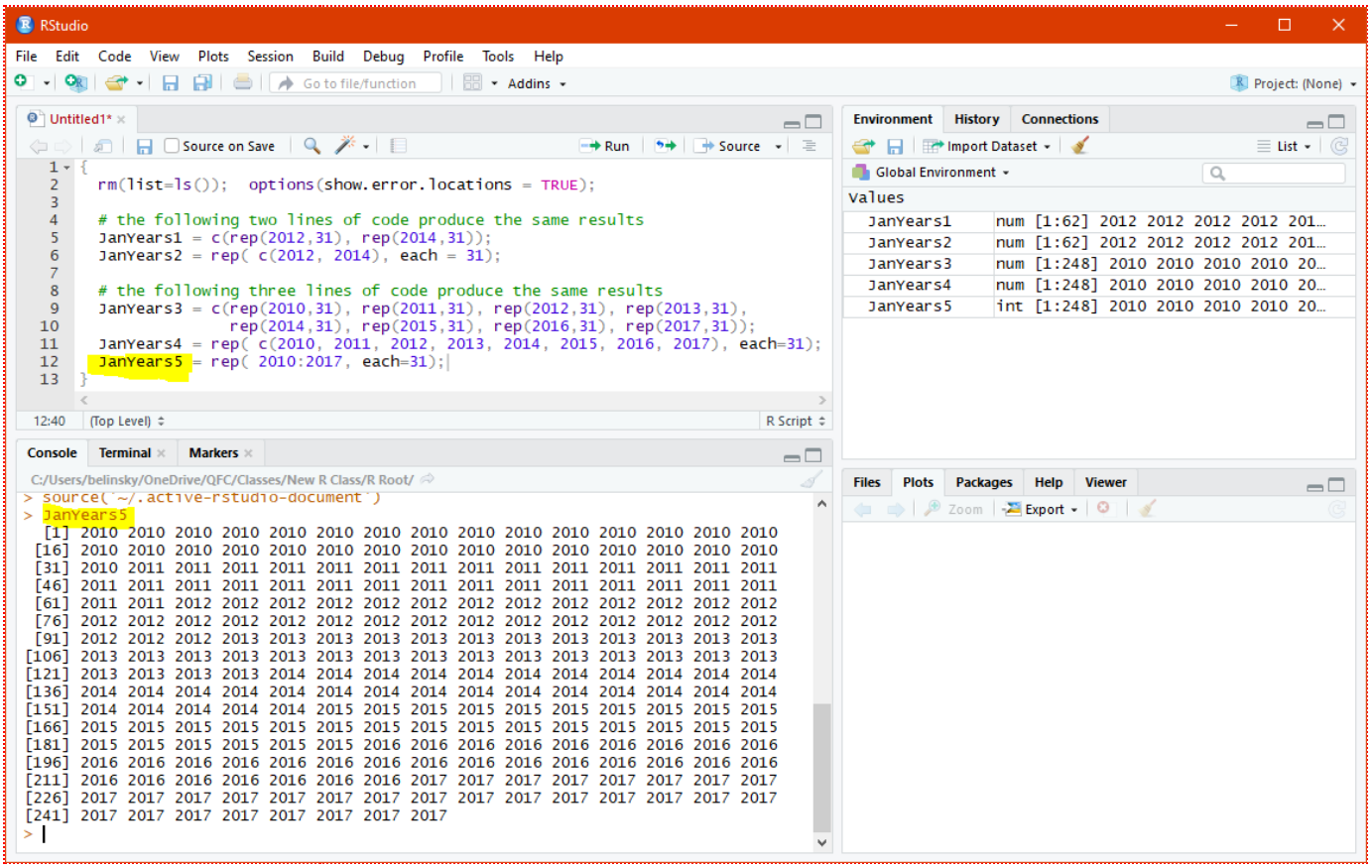
```
2                rep(2014,31), rep(2015,31), rep(2016,31), rep(2017,31));
```

or we could make a vector from **2010** to **2017** and use the ***each*** parameter:

```
1 JanYears4 = rep( c(2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017), each=31);
```

or, to reduce even more code, we could using a sequence to represent all years from **2010** to **2017**:

```
1 JanYears5 = rep( 2010:2017, each=31);
```



Using the ***each*** parameter and sequences to reduce code in ***rep()***