R Programming and Data Visualizations

Noah Greski and Andrew Jop

Abstract

Throughout this project we explore and investigate the programming language R. R is a powerful programming language when it comes to data visualization. First we explore R's basic features like data types and data structures. Then we dive deeper into the data side of R. We looked into what plots, graphs and maps R can create. Next we created some small programs that test the plots and maps of the language. Finally we created a larger program that creates an interactive map using R. All code is written and run in RStudio.

Introduction

In this paper the programming language R will be examined and explored. There are several different sections detailing different topics of the R programming language. Before diving into programming in R we learned the features and fundamentals of R. While going through these features we compared and contrasted them to other languages we have gone in depth with in the Programming Languages Course. After analyzing many features, we then started testing out some of these features in R Studio. R Studio is the development environment we used to write, run and test all of our code. We created a small program testing out the basics of R in *example1.R*. After creating this small program we looked into the data visualization portions of R. We created a few small programs, gathering data from Kaggle and created charts, graphs and maps using the data. The final portion of this project included a larger more detailed programming section that allows for highly detailed and interactive data visualization using the R programming language.

Context

R was first released in 2000 by Ross Ihaka and Robert Gentleman at the University of Auckland (Worsley, 2023). R is a programming language for computing statistics and creating graphics. R is well known for its ease of use and ability to generate detailed plots and graphics. R is open source, meaning it is accessible to anyone who would like to use it. R offers a wide range of tools when it comes to statistical analysis and graphics (R Core Team, 2023).

Data types

The first thing we examined was the different data types in R. R is a dynamically typed language meaning that data types do not need to be declared when storing a value in a variable (Intellipaat, 2023). Instead R uses an inference based on the value provided. This is similar to Python in a sense that data types do not need to be defined when creating variables. However, this is different from other languages like C and Java, where the data type needs to be explicitly defined when creating a variable. The advantages to this approach is that there is more flexibility in allowing the programmer to focus on the behavior of the code. The disadvantage to type inference is the potential for error prone code.

The first data type is the numeric data type. The numeric data type is all real numbers, including values with or without decimals (Programiz, 2023). The numeric data type can be integers or floating point values. This is different compared to other languages we are familiar with like in C or Java where there are separate types of integers and floating point values.

The next data type is the integer data type. This data type can specify a value to be an integer (no decimal places) with a suffix "L" (Programiz, 2023). This means that integers can be stored as a numeric data type or can be specified to be an integer by adding the correct suffix onto the value. For example "`21L`" would make 21 an integer type. The numeric type seems more useful in most cases, but there can be times where specifying the value to be an integer type can also be useful within the R programming language.

The next data type in R is complex data type. This data type is used to define imaginary values in R. Again R uses its suffix feature to define this data type. The suffix used for a complex number is "i" (Programiz, 2023). An example of a complex number in R would be "5+3i". This data type is not the most useful data type as complex numbers are not common occurrences when creating programs in R, complex numbers are unique and don't seem to be useful in many situations. Therefore this is not typically supported directly in many programming languages.

The next data type that we explored is the character data type. The character data type in R is used to define character or string values. A string is a set of characters, as it is in mostly every programming language(all of them we have discussed in the Programming Languages Course). In R double quotes (" ") or single quotes (' ') can be used to represent the character data type (Programiz, 2023). For good coding convention in R, single quotes are used for single characters and double quotes are used for strings. Both strings and characters fall under the character data type. This differs from a language like Java, where there is a char type for character and a String type for strings.

Next we looked into the logical data type. This is also known as the boolean data type. The logical data type works in a similar way to the boolean data type in Java. There are only two values that can be stored as a logical data type, "`TRUE`" and "`FALSE`" (Programiz, 2023). Additionally, `TRUE` and `FALSE` values can also be defined as just a single letter, T or F. The logical data type in R is similar to most boolean data types in other languages, defining true and false. However, some languages do not have a logical or boolean type, like C. C just uses integers 1 and 0, for true and false. The logical data type is useful in many programming scenarios and is a valuable addition to the language.

The final data type that we looked at in R is the raw data type. This data type specifies values as raw bytes in R. There are two methods that can be used to convert characters to raw and raw to characters, `charToRaw()` and `rawToChar()`. An example of this would be converting "W" to raw would make it 57, and vice versa (Programiz, 2023). The raw data type is not useful in many situations. One practical use of the raw data type is for efficient memory storage. In some cases where memory is crucial, the raw data type takes up less memory therefore making it more efficient.

Variable Names

Next we looked into variable names in the R programming language. There are a few rules when it comes to naming variables in R. First, a name must start with a letter and it can be a combination of letters, digits, period(.) and underscore(_). If the variable starts with a period, it can have a digit after it. Names cannot start with underscores or numbers. Names are case sensitive. Lastly, reserved words cannot be used as variable names (W3Schools, 2023). The variable name rules are similar to C based languages and many other programming languages.

Operators

After variables, we explored all of the operators within the R programming language. First we looked at the arithmetic operators, there are 7 of them. They include, `+, -, *, /,` which are operators that work as they would in basic math(W3Schools, 2023). Next are the last three operators that are a bit different from other languages. The first is `^`, which is called exponent. In a language like Java, Math.pow is used for exponents, using the Math Library. Next is `%%`, which is similar to `%` from Java or C which is called Modulus. This gets the remainder from division. Lastly we have the operator `%/%` which is integer division (W3Schools, 2023). The operator allows for the result of an expression to result in an integer instead of a floating point number.

There are 2 assignment operators in R: `->` and `<-`. Either one can be used to assign a value to a variable, always making sure that the value is pointing to the variable name. An example of this would be `8 -> x` is the same as C's or Java's `x = 8`. There is another version of the assignment operator which is `->>` or `<<-`, which is a global assigner(W3Schools, 2023). Next we have comparison/relational operators. These operators are the exact same as several other languages that we have studied in class. These operators include equal (`==`), not equal (`!=`), greater than (`>`), less than (`<`), greater than or equal to (`>=`) and less than or equal to (`<=`). These work as intended and the same ways as most programming languages(W3Schools, 2023).

Next we have logical operators. These are also similar to most languages and can be used in conditional statements. First we have bitwise `&` and `&&`. The single AND operator is used for elements and the double is used for statements. The same goes for the `|` and `||`, OR operators which are also in R. Lastly is the Logical NOT (`!`) operator. This returns a `FALSE` if the statement is `TRUE` (W3Schools, 2023).

Lastly we have a few miscellaneous operators that can be used to manipulate data. First we have the : operator. This creates a series of numbers in a sequence. This can be used for several applications in the R language. An example of this would be, `a <- 1:5`, making the variable a now a sequence of numbers 1 through 5. Next we have, `%in%`, which finds out if an element belongs to a vector. An example of this would be, `a %in b`, seeing if element a is in vector b. Lastly we have `%*%` which is matrix multiplication which is often used for data science. It does what it is called, it multiplies matrices together, instead of having to iterate through them one by one and multiply them (W3Schools, 2023). Most operators in R are similar and work similarly to languages like Java and C, but have some other operators that are for specific scenarios that can be very useful and are unique in R.

Data Structures

After exploring through the different types of operators in R, when then decided to look at the different data structures in R. The first data structure we investigated is vectores. A vector is a list of items that are of the same type (W3Schools, 2023). Vectors can be any data type in R. Vectors can be created in a few different ways. First, the most common way to make a vector is by using the c() function or called the concatenate function. This function takes a list separated by commas of values of the same type. For example the code segment, `a <- c(1, 2, 3)`, creates a vector with the values 1, 2 and 3 in it. Again, this can be done for any data type. Another way to create a vector is by creating a sequence of values, this can be done by using the : operator or by using the seq() function. Vectors can be accessed by using brackets ([]). For example, by using the vector about with 1, 2 and 3 in it, you can do, `a[c(1,2)]`, to get the first two elements of the vector only (W3Schools, 2023). Vectors are a very useful data structure that have various applications. Vectors also use functions like sort() and length() to manipulate the data inside the vectors and get information about the vector. Vectors are similar to an array in Java because the programmer has direct access making it easy to manipulate data in the vector.

The next data structure that we explored in R is lists. Lists are very similar to vectors and are created and accessed in similar ways. However, in lists there can be several different data types that can be stored in a list, compared to a vector that can only have one data type. Lists can be created using the `list()` function and elements can be added into the list using the `append()` function. As with vectors, lists can be accessed using brackets([]). Additionally, the `length()` function can be used to return the length of a list (W3Schools, 2023). Lists are another useful data structure in R that can be used in various applications.

The last data structure that we looked into in R is data frames. A data frame is not similar to any data structure that we have looked at in the Programming Languages Course, or like any well known data structure in familiar languages. Data frames are data displayed in the format of a table. Data frames can have different data types within the tables, the first column can be numeric but the second could be logical. Each column needs to have the same data(W3Schools, 2023). The function `data.frame()` can be used to create a data frame. This data structure is complex and has many applications. There are various functions to manipulate the data within the data frame including, `summary()`, `cbind()`, `rbind()`, `length()` and several others(W3Schools, 2023). Data frames are a useful data structure that has various applications when creating complex programs in R. More specifically when working with small sets of data, data frames can be extremely useful.

In R there are several other data structures including matrices, arrays and factors. These structures are useful and can provide a better understanding of the language. However, matrices and arrays work similarly to how they do in all other programming languages, therefore there is no need for a further explanation of them. Additionally the factor data structure has its benefits but explaining it is beyond the scope of this project and is not necessary to move forward when learning about R.

Garbage Collection

The last basic feature that we examined in R is the garbage collection. In many programming languages memory needs to be explicitly freed within the program. In R, garbage collection is used. Garbage collection automatically releases memory when an object is no longer being used. R does this by tracking how many names point to an object and when nothing is pointing to that object, it is freed and deleted from memory (W3Schools, 2023). This is similar to Java, how memory does not need to explicitly be freed in the program. However, this is very different from a language like C where the `free()` function must be used.

Data Visualization

After looking through the basic features of R we decided to focus more on the data visualization features that R has to offer. These features are what separates R from other programming languages. In the grand scheme of things, R has similar features to most regular and well known programming languages, but in the data visualization part, R stands out. R offers several areas of graphics within their libraries that allows for extensive data manipulation and analysis using its visualization features. First we will go through what R has to offer, then provide a few examples using its features of data visualization.

R shines from its powerful visualization capabilities through various libraries that offer tools to represent various models. We decided to use the “ggplot2” open source data visualization library. This library features declarative syntax, which allows users to build plots by specifying the data, aesthetics (like color, size), and layers through a clear and intuitive grammar. Not to mention the rich variety of plots the library offers. There are a wide range for different uses. For example, scatter plots are ideal for showing a relationship between two variables whereas line charts are good at displaying trends over time. ggplot2 offers a user-friendly experience making it accessible for statisticians and analysts. Its intuitive syntax and clear structure helps users to create a wide array of visualizations, allowing for effective communication of complex data insights without steep learning curves.

Data Visualization Examples

Now we will be providing examples of using ggplot2 in action. We gathered data (in csv format) from the website Kaggle (Kaggle). A CSV file is a file with comma separated values. We got csv files and added these files to our Github repository. We created a few examples showing what R can do with this data and how it can create these visualizations. We created three small example programs with three different sets of data. These programs read in a csv file then output a plot we created based on that data.

Boston Example

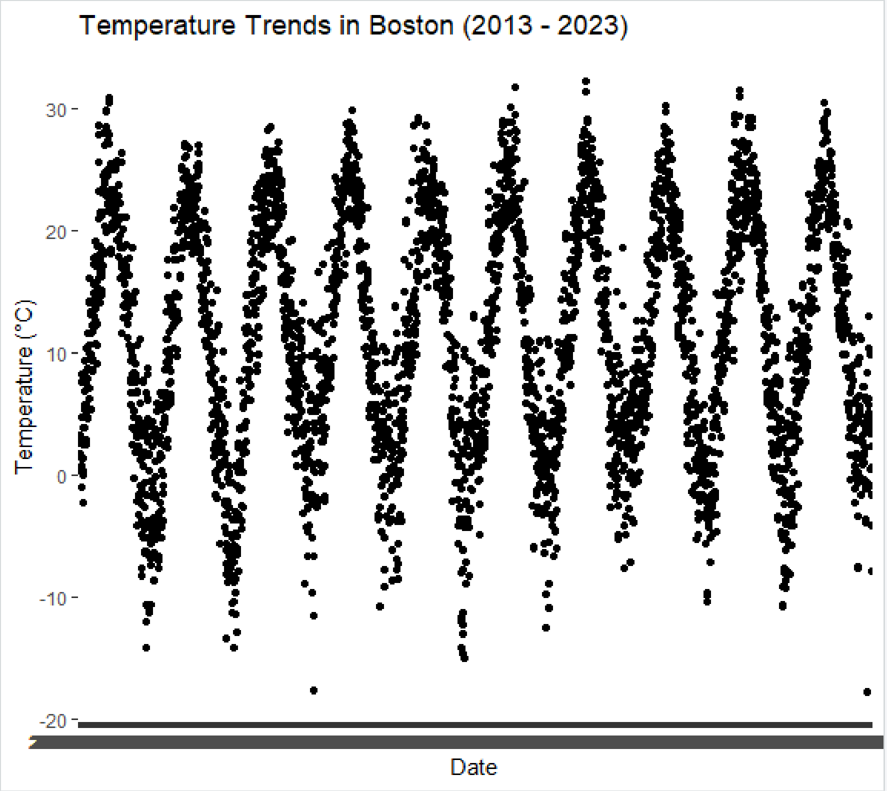
In the example Boston.R, we read a comma separated value (csv) file which has data about weather in Boston, Massachusetts from 2013 - 2023 using the read.csv() built in function.

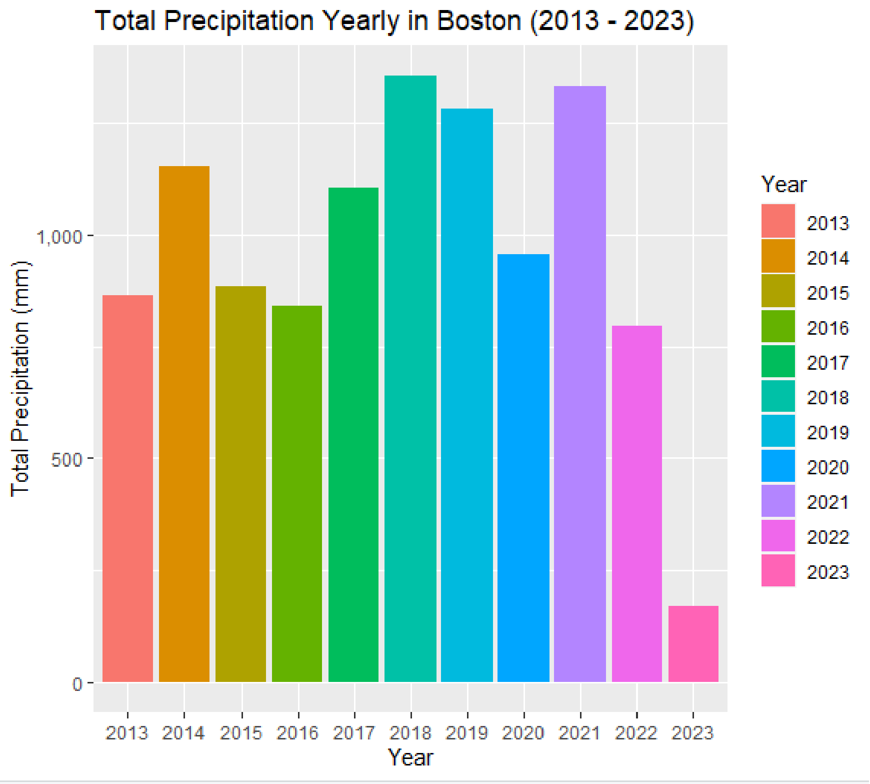
```
data <- read.csv("Boston Example/boston_weather_data.csv")
```

We retrieved this data from Kaggle (Meher 2023). From the data we read in we will then create a scatter plot using the ggplot() function with the first argument as our data, we then use aes() which is a ggplot2 function to set the label of our x and y axes. We then add geom_point() to add points to the plot and set labels for the plot.

```
ggplot(data, aes(x = time, y = tavg)) +
  geom_point() +
  labs(title = "Temperature Trends in Boston (2013 -
2023)",
       x = "Date",
       y = "Temperature (°C)")
```

There are many functions that take care of the underlying work.We created a bar chart based on the precipitation values in the csv file, before we did this we had to create a date object to then create a year variable which takes the year value. This is done using the `format()` function from the time column in the csv specifying “`%Y`” to extract a four digit format. We can use the ggplot() as before and specify the parameters for a bar chart.

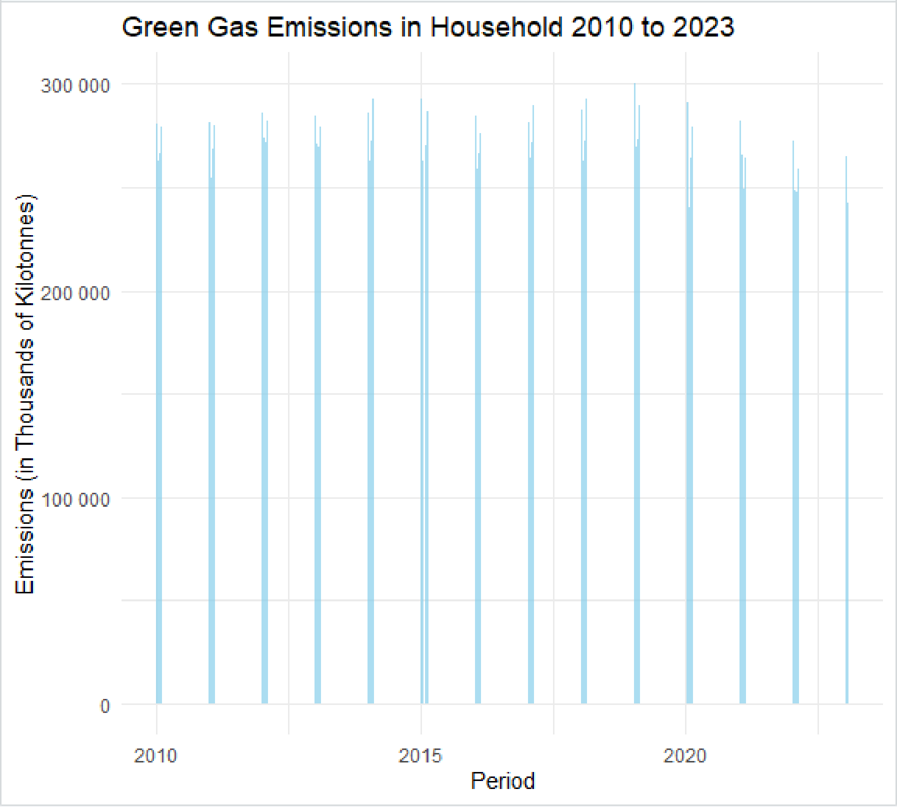




Emissions Example

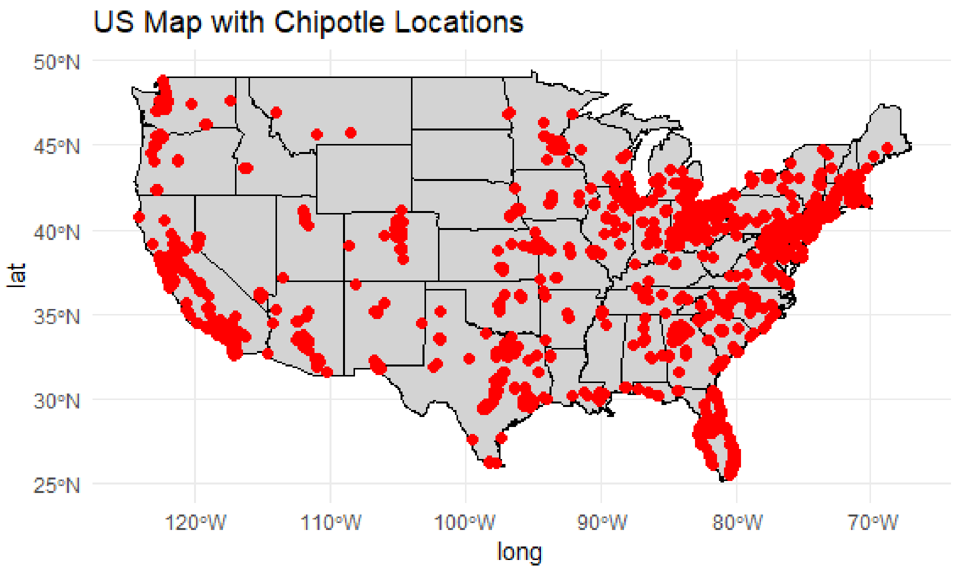
The next example we created was emissions.R (Greenhouse, 2023). First we installed the ggplot2 package and called the library function, so that the necessary functions are available for use throughout this program. Read the data from the CSV file. We used the ggplot function and the aes function to take in the x and y values that we wanted from the data. Then we used the `geom_bar()` function to create the bar chart using several other features like color and adding titles to the bar chart. Lastly we called a function to scale the data as the numbers were too big to fit within the bar chart for the y axis.Resulting in a bar chart showing household emission values from 2010-2023.

```
     ggplot(data, aes(x = Period, y = Data_value)) +
       geom_bar(stat = "identity", fill = "skyblue", alpha =
0.7) +
       labs(title = "Green Gas Emissions in Household 2010 to
2023", x = "Period", y = "Emissions (in Thousands of
Kilotonnes)") +
       theme_minimal() +
       scale_y_continuous(labels = scales::label_number())
```



Chipotle Example

Lastly, we created another example for data visualization in Chipotle.R. This example uses several packages to use a map to display locational data. The dataset used for this example is called “chipotle_stores.csv.” This dataset contains coordinates with every Chipotle restaurant in the US. First as other examples did, the csv file is read into the program using the `read.csv()` function. Next objects are created for each location on the map using the longitude and latitude from the dataset. Then the US map is imported using the maps package and the `map_data()` function. Lastly the map is created with all the locations. Several of the map's features are edited including colors, sizes and titles. This was done by using the `ggplot()`, `geom_polygon()` and `goem_sf()` function. The final map is displayed in the "Plots" section in RStudio. All examples done with these smaller datasets have a similar pattern. Install and load the necessary packages, read in the dataset and then edit the features and display the plot. We got this data from Kaggle (Braun 2020).

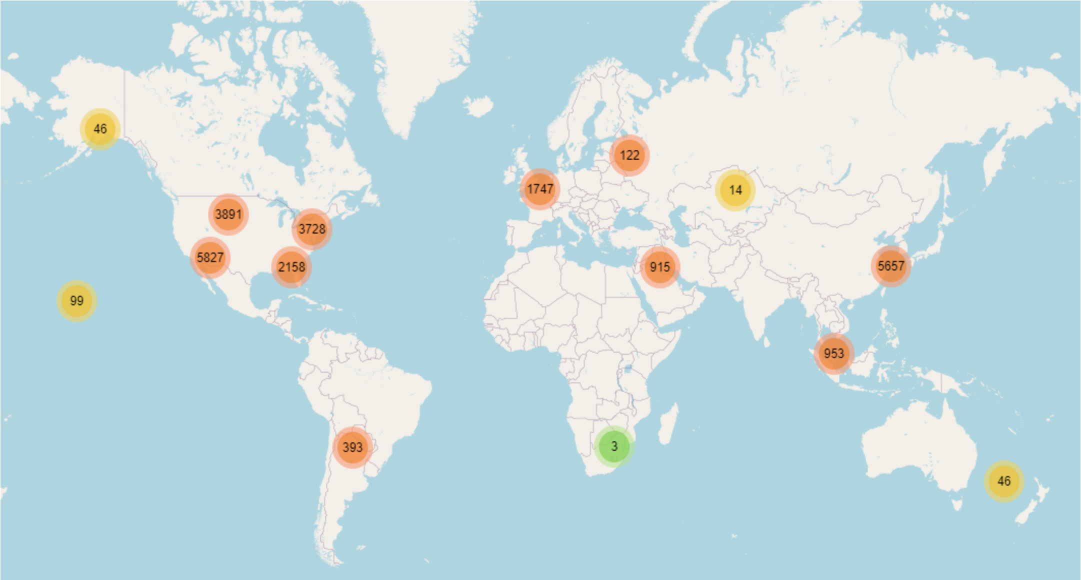


After creating these three data visualization examples in R, we had a better understanding of the language and what it provides with its data visualization and manipulation capabilities. In a way R is very similar to the programming language Python. Both allow for the importation of packages that can display graphics based on a dataset. In the programming languages course we talked about Python frequently but not much about its data manipulation and visualization capabilities. Nonetheless these languages are very similar in this way. When programming in R, the amount of coding can be relatively short in most cases, but the output of the program can be detailed and large. Through these examples we learned the basics of the data visualization portion of the R programming language.

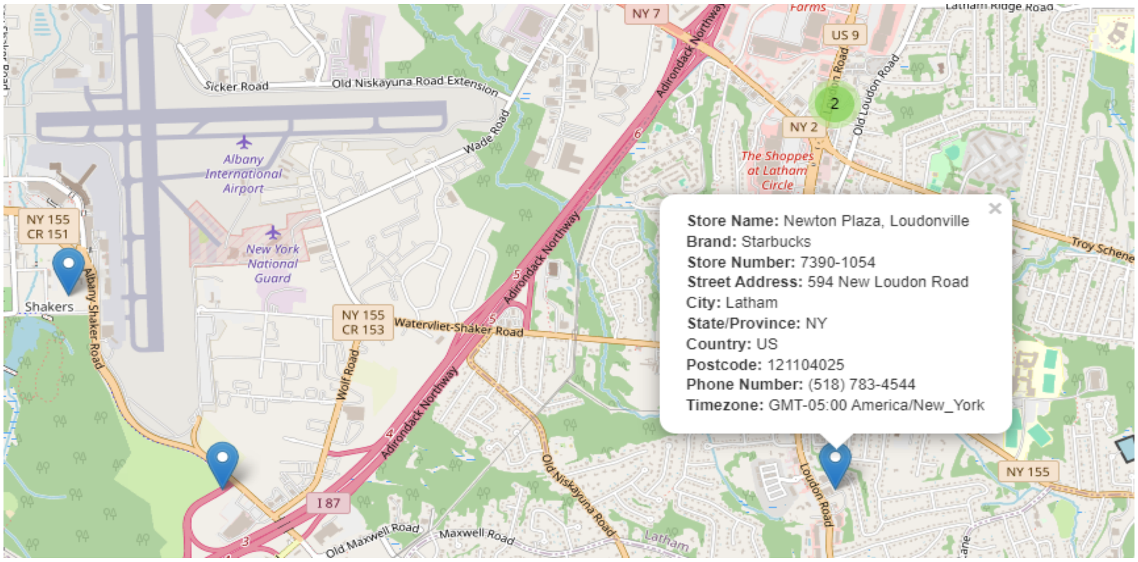
Starbucks Interactive Map

After going through all the basic features of R and doing some basic programming in the language we decided to end our project by taking on a larger programming task that will really show off the capabilities of R. We decided to create an interactive map within RStudio that shows all world Starbucks locations. This program can be seen in Starbucks.R. We wanted to do something with a lot of location data, worldwide of course. We went to Kaggle and found world Starbucks locations(VEERESHELANGO, 2016) and made our program around that. This data set includes not just the location of each Starbucks but it also has information about each location including store name, address, city, country as well as several other pieces of information. Next we wanted to research a bit more about interactive maps in R. There are several ways to create these maps but we found that using the Leaflet package was the most effective way (Ba Tran, 2018). Leaflet is an open street map that allows users to place data points on the map based on longitude and latitude (Hahn, 2020). The map itself is very detailed and shares similarities with Google Maps.

After exploring the Leaflet package we got to programming. We had to install three packages into our code, including Leaflet, dplyr and sf. Dplyr is a package that is great for data manipulation that helps us filter some of the locations within our dataset. And sf handles spatial data, like overlaying maps with a lot of data points. Then we loaded those packages into our program. As done in every example so far, we read the data using the `read.csv()` function. Next we filtered out all the rows that had no coordinates in them. This was causing us errors because some of the data did not have valid locations and the map was not processing them correctly. Here we used the `filter()` function to do this, this function used the dplyr package. Next we converted the data into sf objects which maps the map run much better. Spatial indexing allows for more efficient spatial operations. Next we started to create the actual map with the data points. The Leaflet package takes care of most of the dirty work here using its function to edit how we want to see our map. Using functions like addTiles and addMarkers, to add tiles and makers at each location. The popup and label section adds what you will see when you hover over or click on a Starbucks location on the map. Finally we print the map by just putting "starbucks_map_world."

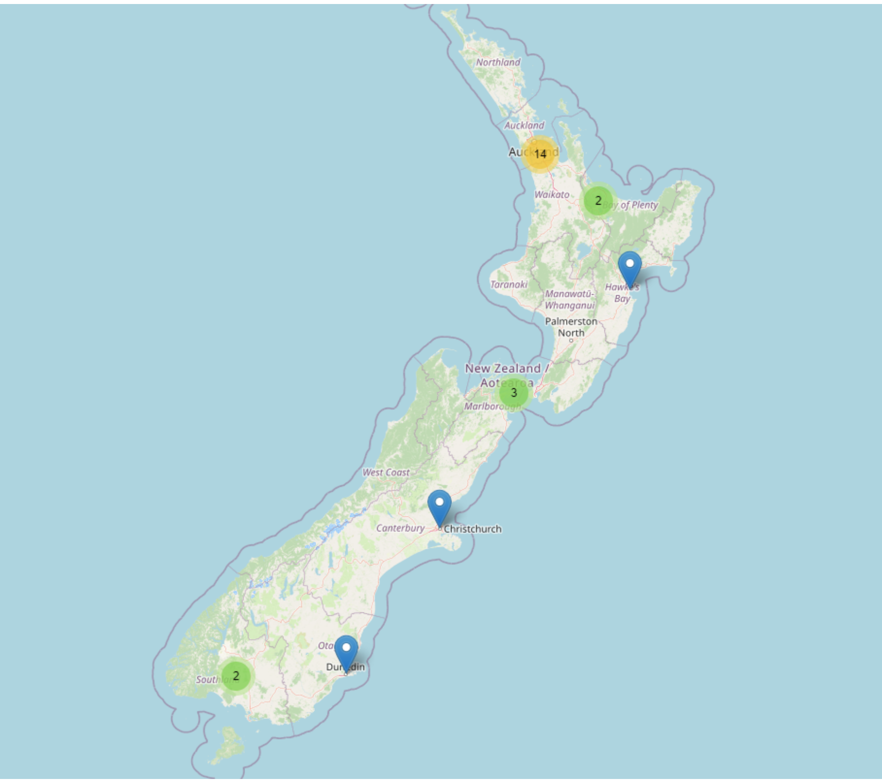


After running the program in RStudio our interactive map pops up on the right side of the screen. The zoom button can be used to see a bigger version of the map. The map can be zoomed in and out to go all over the world to see every Starbucks location. The points are shown in clusters until being zoomed in closer. The reason for this was for the map to be able to run faster. Showing all the points on the screen makes the map very slow and almost unusable. The clusters are created by the line, `clusterOptions = markerClusterOptions()`, in the program. When zooming all the way into a data point it can be clicked on to show more information. This information includes the store name, address, city, phone number and several other pieces of information about that specific Starbucks location.



Store Name: Newton Plaza, Loudonville
Brand: Starbucks
Store Number: 7390-1054
Street Address: 594 New Loudon Road
City: Latham
State/Province: NY
Country: US
Postcode: 121104025
Phone Number: (518) 783-4544
Timezone: GMT-05:00 America/New_York

This map is simple and is user friendly. It also displays a large amount of data. When looking at the map you can see what parts of the world have more Starbucks locations than others. See what places do not have Starbucks at all. This geographical data allows for users to do what they want with it and see what different Starbucks locations that they can find. When zooming in only partly you can better see a specific city or country without the data still being clustered. Here we have a screenshot of New Zealand's Starbucks locations.



Interactive Map Conclusions

After completing this programming section there are a few conclusions that we can make about the program we created. R is capable of creating an interactive map with less than 100 lines of code. This shows off the impressive efficiency of the language and the nature of the packages provided. This interactive map is something that neither of us have created in any programming language before. It took a lot of timing and understanding to create this map. What we have created takes a massive data set and turns it into an interactive map. This really shows the excellence of R when it comes to data visualization and manipulation.

Conclusion

After completing this project there are several conclusions that we can draw about R and what it can do as a programming language. We learned a lot about the programming language R, but we also learned a lot about programming languages in general. R is a powerful programming language that has many capabilities when it comes to data visualization. Plots and maps are R's strong suits. R's other features are not as impressive and they remain similar to other programming languages that we have discussed in the Programming Languages course. R is an efficient programming language, as the amount of code to write is not a lot, when it comes to the result of the program(the interactive map shows this well). This project gave us a small taste of what programming languages can do. Creating an interactive map is something that seems insignificant, that it already has been done millions of times, but creating it yourself is a whole different story. This project has opened our minds on what programming can do. In the future we hope to work with R again, creating more detailed maps. We also hope to use larger data sets with R and try to push its limits on what we can see through creating maps in R.

Citations:

Ba Tran, A. (2018). Interactive Maps with Leaflet. *In R for Journalists*. https://learn.r-journalism.com/en/mapping/leaflet_maps/leaflet/.

Braun, Jeffrey. 2020, Chipotle Locations, Kaggle, https://www.kaggle.com/datasets/jeffreybraun/chipotle-locations.

Greenhouse gas emissions (industry and household): March and June 2023 quarters | Stats NZ. https://www.stats.govt.nz/information-releases/greenhouse-gas-emissions-industry-and-household-march-and-june-2023-quarters/.

Hahn, N. (2020). Making Maps with R. https://bookdown.org/nicohahn/making_maps_with_r5/docs/leaflet.html.

Intellipaat. (2023). R Data Types and Variables. In R Programming Tutorial - Learn R Programming Basics. https://intellipaat.com/blog/tutorial/r-programming/basic-syntax-data-types-and-variables/#:~:text=Variables%20in%20R%20programming%20can,use%20it%20in%20our%20program.

Meher, Swaroop. 2023, Boston Weather 2013-2023. Kaggle, https://www.kaggle.com/datasets/swaroopmeher/boston-weather-2013-2023/.

Programiz. (2023). R Data Types. https://www.programiz.com/r/data-types#google_vignette.

R Core Team. (2023). What is R? Introduction to R. R Project. https://www.r-project.org/about.html.

VEERESHELANGO. (2016). Starbucks Worldwide. https://www.kaggle.com/code/veereshelango/starbucks-world.

W3Schools. (2023). R Data Frames. https://www.w3schools.com/r/r_data_frames.asp.

W3Schools. (2023). R Lists. https://www.w3schools.com/r/r_lists.asp.

W3Schools. (2023). R Operators. https://www.w3schools.com/r/r_operators.asp.

W3Schools. (2023). R Variable Names (Identifiers). https://www.w3schools.com/r/r_variables_name.asp#:~:text=A%20variable%20can%20have%20a,and%20underscore(_).

W3Schools. (2023). R Vectors. https://www.w3schools.com/r/r_vectors.asp.

Wickham, H. (2023). Advanced R (2nd ed.). http://adv-r.had.co.nz/memory.html#:~:text=R%20uses%20an%20alternative%20approach,object%2C%20it%20deletes%20that%20object.

Worsley, S. (2023). What is R? - An Introduction to The Statistical Computing Powerhouse. *DataCamp*. https://www.datacamp.com/blog/all-about-r.