Should I Learn Esperanto?

Willow Shipperley

May 15, 2019

Why Esperanto?

There are hundreds of languages currently in use around the world today. Right now, in so many different tongues, people are making friends, conducting business, navigating the country they are in, and so much more. As citizens of the world, it can be difficult to know that even though there are billions of people out there, we are limited to only being able to communicate with people who know the same languages as us. Because of this, many people have decided to learn a second language, then perhaps a third, then perhaps a fourth, and so on. As an avid lover of languages, sometimes it can be difficult to figure out which language one should learn next. Common factors to help decide this usually include (but are definitely not limited to) “what places can I go if I speak this language,” “how many new people will I be able talk to,” and “how long will it take me to get a good understanding of this language.”

Today, I will show that, whether you are preparing to learn your seventeenth language or your second, it should be Esperanto (https://en.wikipedia.org/wiki/Esperanto). Esperanto is the world’s most used conlang (https://en.wikipedia.org/wiki/Constructed_language), or constructed language. That means it did not come from natural usage and evolution, like Spanish came from Latin roots. Instead, it was built from the ground up by a single person (https://en.wikipedia.org/wiki/L._L._Zamenhof), who defined the rules of its grammar and pronunciation. Esperanto was originally created in the late 1880s, with the hopes that it would become a universal language that all of the world’s people could speak. Although it has not reached its original goal, in part due to issues involving the world wars, there are many speakers today, and more starting to learn all the time.

Goals

In order to prove that Esperanto should be the next language you learn, we are going to look at the three factors I listed above, and compare them for Esperanto and the official languages of other countries. To clarify, we will be finding out:

1: How many new people you will be able to talk to? 2: How long it will take you to have a good grasp of the language? 3: What places you will be able to go to once you speak Esperanto?

Gathering Data

To answer the above 3 questions, we are going to need to collect some data. To begin, we will need to include some libraries to aid us.

tidyverse (https://www.tidyverse.org/) will be the first. This library contains many packages that will help us prepare, model, and visualize data.

rvest (https://github.com/tidyverse/rvest) will help us get data from the web.

dplyr (https://cran.r-project.org/web/packages/dplyr/vignettes/dplyr.html) is helpful for data manipulation.

ggplot2 (https://ggplot2.tidyverse.org/) will be useful when we make different graphs and plots.

Note that if you do not have any of the packages listed above, you can run install.packages(“”) to get them.

```
library(tidyverse)
```

```
## -- Attaching packages ------------------------------------- tidyverse 1.2.1 --
```

```
## v ggplot2 3.1.0     v purrr   0.3.0
## v tibble  2.0.1     v dplyr   0.7.8
## v tidyr   0.8.2     v stringr 1.3.1
## v readr   1.3.1     v forcats 0.3.0
```

```
## -- Conflicts ---------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(rvest)
```

```
## Loading required package: xml2
```

```
## 
## Attaching package: 'rvest'
```

```
## The following object is masked from 'package:purrr':
## 
##     pluck
```

```
## The following object is masked from 'package:readr':
## 
##     guess_encoding
```

```
library(dplyr)
library(ggplot2)
```

Great. Now that we are all set, we can begin.

The first thing we need to do is find a list of countries and their main language. There is one CSV file (https://www.howtogeek.com/348960/what-is-a-csv-file-and-how-do-i-open-it/), or “Comma Separated Value” file easily accessable online, which does what we need. You can find it here (http://www.fullstacks.io/2016/07/countries-and-their-spoken-languages.html). After you download it, you can load it from your computer using the local path name.

```
csv <- "C:/Users/willo/country_list.csv"
codes <- read_csv(csv) %>% as.data.frame()
```

```
## Parsed with column specification:
## cols(
##   ID = col_double(),
##   country_name = col_character(),
##   country_code_name = col_character(),
##   country_code = col_double(),
##   lang_name = col_character(),
##   lang_code = col_character()
## )
```

```
head(codes)
```

```
##   ID   country_name country_code_name country_code         lang_name
## 1  1    Afghanistan                af           93            Pashto
## 2  2        Albania                al          355          Albanian
## 3  3        Algeria                dz          213 Tamazight (Latin)
## 4  4 American Samoa                as         1684              <NA>
## 5  5        Andorra                ad          376              <NA>
## 6  6         Angola                ao          244              <NA>
##   lang_code
## 1        ps
## 2        sq
## 3       tzm
## 4      <NA>
## 5      <NA>
## 6      <NA>
```

The next thing we need to do is change the lowercase country codes into uppercase ones, because we will need them to match another dataframe later! Let's fix that now.

```
codes$country_code_name <- sapply(codes$country_code_name, toupper)
codes <- codes[ -c(1, 4, 5, 6)] # removed unnecessary columns as well
names(codes) <- c("country", "code")
codes <- codes[-c(60, 61),]
head(codes)
```

```
##          country code
## 1    Afghanistan   AF
## 2        Albania   AL
## 3        Algeria   DZ
## 4 American Samoa   AS
## 5        Andorra   AD
## 6         Angola   AO
```

Great! However, some of the languages are missing, and some of them are written as a native would pronounce it. For example, "French" is written as "Occitan." The way countries are spelled is going to be important later on, so this isn't going to be enough on it's own. That's why I removed that column from the dataframe.

A good site we can supplement our data with is this site (https://www.infoplease.com/world/countries/languages-spoken-in-each-country-of-the-world).

We will do this by scraping (https://en.wikipedia.org/wiki/Web_scraping), or gathering, information from the website above.

```
url <- "https://www.infoplease.com/world/countries/languages-spoken-in-each-country-of-the-worl
d"

countryLangs <- read_html(url) %>%
  html_node("table") %>%
  html_table()
head(countryLangs)
```

```
##                    X1
## 1         Afghanistan
## 2             Albania
## 3             Algeria
## 4             Andorra
## 5              Angola
## 6 Antigua and Barbuda
##                                                                   X2
## 1 Dari Persian, Pashtu (both official), other Turkic and minor languages
## 2                        Albanian (Tosk is the official dialect), Greek
## 3                            Arabic (official), French, Berber dialects
## 4                     Catalán (official), French, Castilian, Portuguese
## 5              Portuguese (official), Bantu and other African languages
## 6                                    English (official), local dialects
```

Those simple lines of code allow us to go onto the linked website, and find the table that is embedded into the page. html_node() is a function in rvest that finds the part of the website we tell it to, in this case, “table”. Since it is already in the correct format, all we have to do after that is give the command html_table(), which converts the html into a dataframe we can use!

Now that we have access to it here, lets clean it up a little. I am going to rename the columns, and remove all but the first language listed for each row, since we will only need one per country.

```
names(countryLangs) <- c("country", "language") #this sets the columns names
countryLangs$language <- gsub(",.*", "", countryLangs$language) # removes everything after a com
ma (inclusive)
countryLangs$language <- gsub(";.*", "", countryLangs$language) # removes everything after a sem
icolon (inclusive)
countryLangs$language <- gsub("\\(.*", "", countryLangs$language) # removes everything after an
 opening parenthesis (inclusive)
countryLangs$language <- gsub("[0-9].*", "", countryLangs$language) # removes everything after a
number (inclusive)
countryLangs$language <- gsub("/.*", "", countryLangs$language) # removes everything after a for
ward slash (inclusive)
countryLangs$language <- gsub(" and.*", "", countryLangs$language) # removes everything after th
e word " and" (inclusive)
countryLangs$language <- gsub(" less.*", "", countryLangs$language) # removes everything after t
he word " less" (inclusive)
countryLangs$language <- gsub("Catalán", "Catalan", countryLangs$language) # we need to remove t
he special character so we can merge datasets later
countryLangs$language <- gsub("Bokmål Norwegian", "Norwegian", countryLangs$language) # we need
 to remove special characters so we can merge datasets later
countryLangs$language <- gsub("Standard Chinese", "Mandarin", countryLangs$language) # for mergi
ng, again
countryLangs$language <- gsub("\\s+", "", countryLangs$language) #remove any sneaky white spaces

countryLangs$country <- gsub(" and.*", "", countryLangs$country)
countryLangs$country <- gsub("\\(.*", "", countryLangs$country)

head(countryLangs)
```

```
##       country    language
## 1 Afghanistan DariPersian
## 2     Albania    Albanian
## 3     Algeria      Arabic
## 4     Andorra     Catalan
## 5      Angola  Portuguese
## 6     Antigua     English
```

Good work! Now we are going to combine the edited CSV file and our countryLangs, by merging the rows that have the same value in the country column. We will do this via inner_join().

```
countryLangs <- countryLangs %>%
  inner_join(codes, by="country")
head(countryLangs)
```

```
##       country    language code
## 1 Afghanistan DariPersian   AF
## 2     Albania    Albanian   AL
## 3     Algeria      Arabic   DZ
## 4     Andorra     Catalan   AD
## 5      Angola  Portuguese   AO
## 6   Argentina     Spanish   AR
```

Great! Next up, we are going to do some very similar data scraping for another dataset on another website, which you can find here (https://kalkulinda.com/2016/12/10/percountry-rates-of-esperanto-speakers/). This dataset is going to help us figure out where Esperanto speakers live, and how many there are in each country.

There have been many attempts over the years to find out exactly how many people actually speak Esperanto. Ethnologue (https://www.ethnologue.com/language/epo) is a reference publication which tracks information on the languages of the world. It currently reports roughly 2,000,000 (two million) speakers worldwide, in about 115 countries. Between 1,000 (one thousand) and 2,000 (two thousand) people are considered to be native speakers, having learned the language from birth as taught by their parents/guardians. Unfortunately, there are very few solid resources detailing exactly where these speakers are located.

The dataset we are using today was created by a statistician/bioinformatician and Esperanto enthusiast named Svend. The explanation of how he calculated the per country rates is quite thorough, although a few years old. The number he came up with as a total, 62983.9, is significantly less than the 2,000,000 (two million) reported in many other places. His calculations take into account the number of active users on many Esperanto-based platforms, such as Lernu (https://lernu.net/en), a site for learning and talking to others, and Pasporta Servo (https://www.pasportaservo.org/), a site where Esperanto speakers can temporarily offer their homes to other speakers as they travel the globe. So, although there is a large discrepancy between his result and the accepted number of speakers worldwide, his results are sound enough to use for our purposes.

Let's scrape his data into another dataset, as we did before.

```
url <- "https://kalkulinda.com/2016/12/10/percountry-rates-of-esperanto-speakers/"

countries <- read_html(url) %>%
  html_nodes("table") %>%
  html_table()
#we use html_nodes with an s this time because there are multiple tables to be read
countries <- countries[[3]] #the one we want is the third one
names(countries) <- c("country", "rank", "frequency", "total", "proportion")
head(countries)
```

```
##         country     rank                                frequency
## 1       Andorra  1 [1,2] 620.03 [277.3,1039.9]/ [180.4,2246.5]
## 2     Lithuania  2 [1,6]   249.32 [156.7,482.6]/ [119.2,936.9]
## 3       Iceland  3 [2,8]      210.07 [109.7,375]/ [71.5,729.9]
## 4       Hungary  4 [2,6]    203.82 [151.1,283.8]/ [98.3,695.9]
## 5    Luxembourg 5 [2,10]     196.82 [93.9,285.1]/ [60.4,681.1]
## 6 New Caledonia 6 [3,16]     122.85 [64.1,234.2]/ [47.2,450.3]
##                          total             proportion
## 1         48.4 [22,81]/ [14,175] 0.0008 [0.0003,0.0013]
## 2     748 [470,1448]/ [357,2811] 0.0119 [0.0072,0.0225]
## 3        67.9 [35,121]/ [23,236] 0.0011 [0.0006,0.0019]
## 4 1997.5 [1481,2781]/ [963,6820] 0.0317 [0.0242,0.0449]
## 5       112.2 [53,163]/ [34,388] 0.0018 [0.0008,0.0026]
## 6         32.9 [17,63]/ [13,121]  0.0005 [0.0003,0.001]
```

Perfect. Now we have his dataset with the total number of people per country. However, he has included confidence intervals in brackets, and we don't need those. Let's clean up the table by removing those. He has also included rank, proportion, and frequency, which we do not need. So, we will remove those as well.

```
keep <- c("country", "total")
countries <- countries[keep]
countries$total <- as.numeric(gsub(" .*", "", countries$total))
countries$country <- gsub(" and.*", "", countries$country)

#These are some edits to help our mergins later
countries$country <- gsub("United States of America", "United States", countries$country)
countries$country <- gsub("The Bahamas", "Bahamas", countries$country)

head(countries)
```

```
##         country  total
## 1       Andorra   48.4
## 2     Lithuania  748.0
## 3       Iceland   67.9
## 4       Hungary 1997.5
## 5    Luxembourg  112.2
## 6 New Caledonia   32.9
```

Okay, now we have two of the three datasets we need. The last dataset we need to get in order to answer our main question is going to be one that shows, roughly, how much time (in hours) a person must study a language in order to become moderately proficient in it. Of course, putting a hard number on something like that is difficult, because there are so many factors that can influence how well you learn, such as age, how many previous languages you have learned, and how similar the language is to others that you know. We will treat these numbers as well researched averages, as they come from the Foreign Service Institute (https://www.state.gov/bureaus-offices/under-secretary-for-management/foreign-service-institute/).

This (https://en.wikibooks.org/wiki/Wikibooks:Language_Learning_Difficulty_for_English_Speakers) is the link we will be using to gather this information.

```
url <- "https://en.wikibooks.org/wiki/Wikibooks:Language_Learning_Difficulty_for_English_Speaker
s"

scrapeHours <- read_html(url) %>%
  html_node(".wikitable")
head(scrapeHours)
```

```
## $node
## <pointer: 0x0000000019e52c30>
##
## $doc
## <pointer: 0x0000000019b6df90>
```

The rows of the table are not clearly defined, so we are going to have to do some work to clean it up! Since each language name in each time category has the attribute “title”, we can use the CSS selector [title]

```
languages <- scrapeHours %>%
  html_nodes("[title]") %>%
  html_text() %>%
  as.data.frame()
names(languages) <- c("language")
head(languages)
```

```
##    language
## 1 Afrikaans
## 2   Catalan
## 3    Danish
## 4     Dutch
## 5    French
## 6  Galician
```

Now that we have all the languages as rows, we can add a column to represent the number of hours it takes to learn that language. Just by looking at the wiki we took the information from, it is easy to see that the first 12 languages are in the 600 hours category, the 62 languages after that are in the 1100 hours category, etc. So, the code below will fill those values in for our new column, “hours”.

```
languages$hours <- c(rep(600, 12), rep(1100, 62), rep(2200, 7), rep(750, 1), rep(900, 5))
head(languages)
```

```
##    language hours
## 1 Afrikaans   600
## 2   Catalan   600
## 3    Danish   600
## 4     Dutch   600
## 5    French   600
## 6  Galician   600
```

Perfect! Now, using research explained here (https://en.wikipedia.org/wiki/Propaedeutic_value_of_Esperanto), we can see that it is estimated that to learn Esperanto, one will need about 150 hours of study. Note that the research which determined this number of 145 to 150 hours of studying Esperanto found that it was more closely equivalent to 1500 hours of studying English, or 2000 hours of studying German, as opposed to the 750 hours given by our above dataset. However, because the research may not be as thorough as the FSI’s research from above, we will keep the 150 hours the same instead of potentially scaling it down. Let’s add that to our languages data set now.

```
temp1 <- data.frame("Esperanto", 150)
names(temp1) <- c("language", "hours")
languages <- rbind(temp1, languages)
languages$language <- gsub("\\s+", "", languages$language)
head(languages)
```

```
##    language hours
## 1 Esperanto   150
## 2 Afrikaans   600
## 3   Catalan   600
## 4    Danish   600
## 5     Dutch   600
## 6    French   600
```

Great! Now we have a list of countries and their main language, a list of countries and how many Esperanto speakers are in them, and a list of languages and how many hours it takes to a decent level of fluency. We just need to collect one more set of data: a list of countries that la Pasporta Servo (mentioned earlier) is active in. However, there is one caveat: the countries on the site are written in Esperanto. So, instead of saving the names of the countries, I will be saving the country codes, which are also listed. This is why we used the CSV at the beginning, and then supplemented with the html scraped countries and languages. We are going to need something in common in order to merge at the end, and the country codes are that thing.

For ease of reference, the data I will be collecting can be found here (https://www.pasportaservo.org/lo/).

```
url <- "https://www.pasportaservo.org/lo/"
pasporta <- read_html(url) %>%
  html_nodes("span") %>%
  html_text() %>%
  as.data.frame()
pasporta <- pasporta[-c(1, 2, 3, 289:296),] #remove unnecessary rows
pasporta <- pasporta[-grep("[0-9]+", pasporta)] # remove any row that has a number in it

pasporta <- as.data.frame(subset(pasporta, nchar(as.character(pasporta)) <= 2)) # only keep the
 two letter country codes

names(pasporta) <- "code"
head(pasporta)
```

```
##   code
## 1   AL
## 2   DZ
## 3   AR
## 4   AU
## 5   AT
## 6   AZ
```

We are now ready to begin combining our three small datasets into one larger dataset! We will do this through a process called joining (https://www.rdocumentation.org/packages/plyr/versions/1.8.4/topics/join), which will help us merge our dataframes together. This is when the library dplyr, which we included earlier, will be very useful!

Let's start with our first two tables, countryLangs and countries. We know that they both have a column called "country", so that is what we will be focusing on. Any rows with the same country name will be merged! I will be using a full_join () on them. This is so countries with no Esperanto speakers don't necessarily get lost. They will have a null value in the "total" column, but we can fix that easily.

Then, we will merge the result of the first merge with our last dataset, with the number of hours to learn a certain language. In this case, I used a left_join (), so all of the data already in our new dataset will stay there, and any new data that matches will get added. This does leave some null values, unfortunately, they can be easily dealt with.

```
languages$language <- as.character(languages$language)

data <- countryLangs %>% full_join(countries, by="country")

temp1 <- data.frame(NA, "Esperanto", NA, NA)
names(temp1) <- c("country", "language", "code", "total")
data <- rbind(temp1, data)

data <- data %>% left_join(languages, by="language")
```

```
## Warning: Column `language` joining factor and character vector, coercing
## into character vector
```

```
head(data)
```

```
##       country    language code total hours
## 1        <NA>   Esperanto <NA>    NA   150
## 2 Afghanistan DariPersian   AF  11.2    NA
## 3     Albania    Albanian   AL  67.9  1100
## 4     Algeria      Arabic   DZ  98.8  2200
## 5     Andorra     Catalan   AD  48.4   600
## 6      Angola  Portuguese   AO  14.9   600
```

Notice that I had to add Esperanto to the dataframe data before I merged it with the dataframe languages. This is because otherwise, the column for Esperanto would have been removed during the final merge, since it didn't previously exist in both dataframes. This same reason also could have caused some of our rows to disappear. With inner join, only those rows with matching column values will be merged. The rest will be ignored. This has a positive benefit, which is that there will be no null values to worry about, except with the Esperanto entity.

We are now done collecting our data!

Analysis

Now that we have all of our data, we can begin to answer the three "factor" questions mentioned earlier.

How Many People Will You Be Able to Talk To?

This is a simple question to answer, as it comes straight from our "countries" dataset. While sources point to there being around 2 million (two million) speakers around the world currently, definitive proof comes from the calculations done by the creator of the dataset. It is simple to sum up the totals.

```
sum(data$total, na.rm=TRUE)
```

```
## [1] 62984
```

While this number can’t even hold a candle to the number of people who speak Mandarin or English, for example, it is still a large number. In addition, these people can be found all over the world, and can be talked to through apps like Amikumu (https://amikumu.com/), or even visited during international events such as la Universala Kongreso de Esperanto (https://en.wikipedia.org/wiki/World_Esperanto_Congress), which has been held nearly every year since 1905.

How Long Will It Take To Learn Esperanto?

It is easy to see just by looking at our data that Esperanto only takes 150 hours to become proficient in versus a minimum of 600 hours for other languages. But how much less is it than all other languages in our dataset? Luckily, some simple math will show us this.

```
150/unique(data$hours, na.rm=TRUE)
```

```
## [1] 1.00000000         NA 0.13636364 0.06818182 0.25000000 0.20000000
## [7] 0.16666667
```

This simple result shows that learning Esperanto will take 25% of the time a 600 hour language would take, 20% of that of a 750 hour language, 16.67% of a 900 hour language, and so on.

But how useful is it to learn a language faster until you know where you can speak it? That leads to the final factor of this article’s question.

What places you will be able to go to once you speak Esperanto?

To answer this, let’s reflect on what the code attribute of our dataframe “data” means. They are the country codes, which match both pasporta and our CSV file from the beginning. In order to find out how many countries we can go to, all we need to do is look at how many rows pasporta has - every country code in there is unique, and signifies a different country that an Esperanto speaker could go to, explore, and be sure to have a place to sleep in.

```
nrow(pasporta)
```

```
## [1] 115
```

115 (one hundred fifteen) unique countries that could be visited as long as you were a speaker of Esperanto. Don’t forget, you can’t use Pasporta Servo to find a place to stay unless you speak only Esperanto to your host. Often, hosts will also give you a tour around their country, or at least the city in which they live. That sounds great, and is definitely a helpful and cheap way to travel the world and meet interesting people along the way! However, before we celebrate, let’s make absolutely sure that we are getting the most benefit by choosing Esperanto instead of another language.

Let’s take a look at our data again.

If we divide our data into groups, we should be able to make a graph that can show us how many countries we will be able to visit and talk to people in depending on which language we use. Let’s see how many countries we can visit depending on what language we learn!

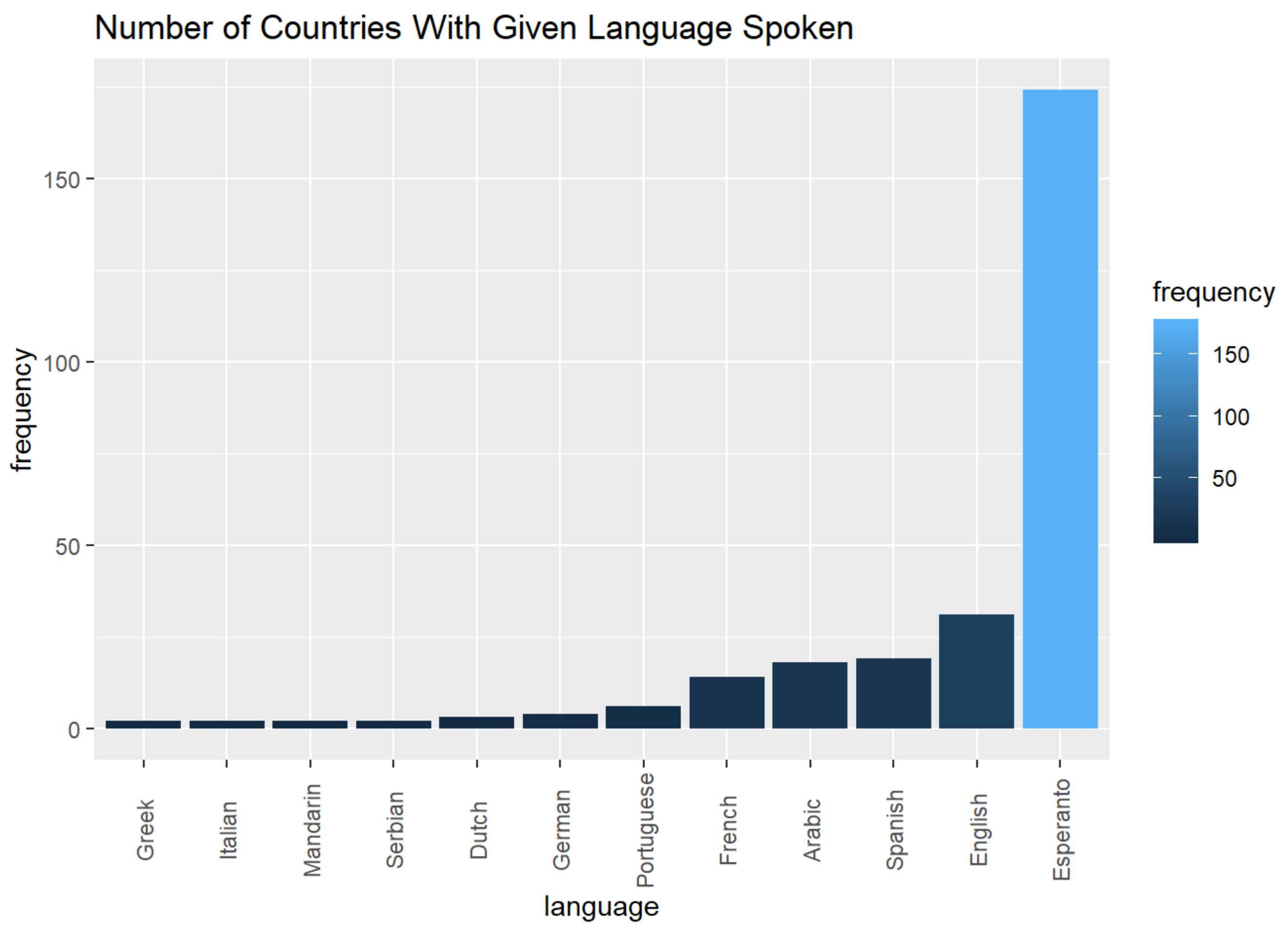
We will start by counting the number of times each language is shown, and then use that to create a histogram plot, so we can compare each language next to each other. We can do this through the table (http://www.datasciencemadesimple.com/table-function-in-r/) function. For the sake of space, any language that has a frequency of one will be removed.

```
uniqueLangs <- as.data.frame(table(data$language))
uniqueLangs <- uniqueLangs[uniqueLangs$Freq != 1,]
names(uniqueLangs) <- c("language", "frequency")
#Esperanto is a unique case; it only shows up once in data, but any entity/row with a total > 0.
0 has at least one esperanto speaker, so we will total that up now

#which returns a vector of indicies where there is a non null value > 0.0, length tells how man
y.
temp <- data.frame("Esperanto", length(which(data$total > 0)))
names(temp) <- c("language", "frequency")
uniqueLangs <- rbind(temp, uniqueLangs)
uniqueLangs <- uniqueLangs[order(uniqueLangs$frequency),]
ordered <- uniqueLangs$language #this gets the order I need for the graph to look more pleasing
 to the eye
head(uniqueLangs)
```

```
##    language frequency
## 32    Greek         2
## 38  Italian         2
## 55 Mandarin         2
## 69  Serbian         2
## 22    Dutch         3
## 31   German         4
```

```
uniqueLangs %>%
  ggplot(mapping=aes(x= language, y= frequency)) +
  geom_bar(aes(fill=frequency), stat="identity") +
  scale_x_discrete(limits=ordered) +
  theme(axis.text.x = element_text(angle=90, vjust=0.5)) +
   ggtitle("Number of Countries With Given Language Spoken")
```



Now we can make a plot, and it is clear to see that the number of countries you can visit and meet new people in is significantly greater if a person can speak Esperanto. In addition, they will also have a host who speaks the same language as you offering their house for free during your stay.

This certainly shows how many places one can speak Esperanto, but what about seeing where, exactly?

We will begin by fixing things like country names that don't fit the dataset. After that, we will combine the official world map data with our own via a left join, and then tell ggplot that we want it to fill the map with colors based on the number of Esperanto speakers per country.

```
data <- data %>% mutate(country = if_else(country == "United States", "USA", if_else(country ==
"United Kingdom", "UK", country)))

map.world <- map_data("world")
```

```
## Warning: package 'maps' was built under R version 3.5.3
```

```
##
## Attaching package: 'maps'
```

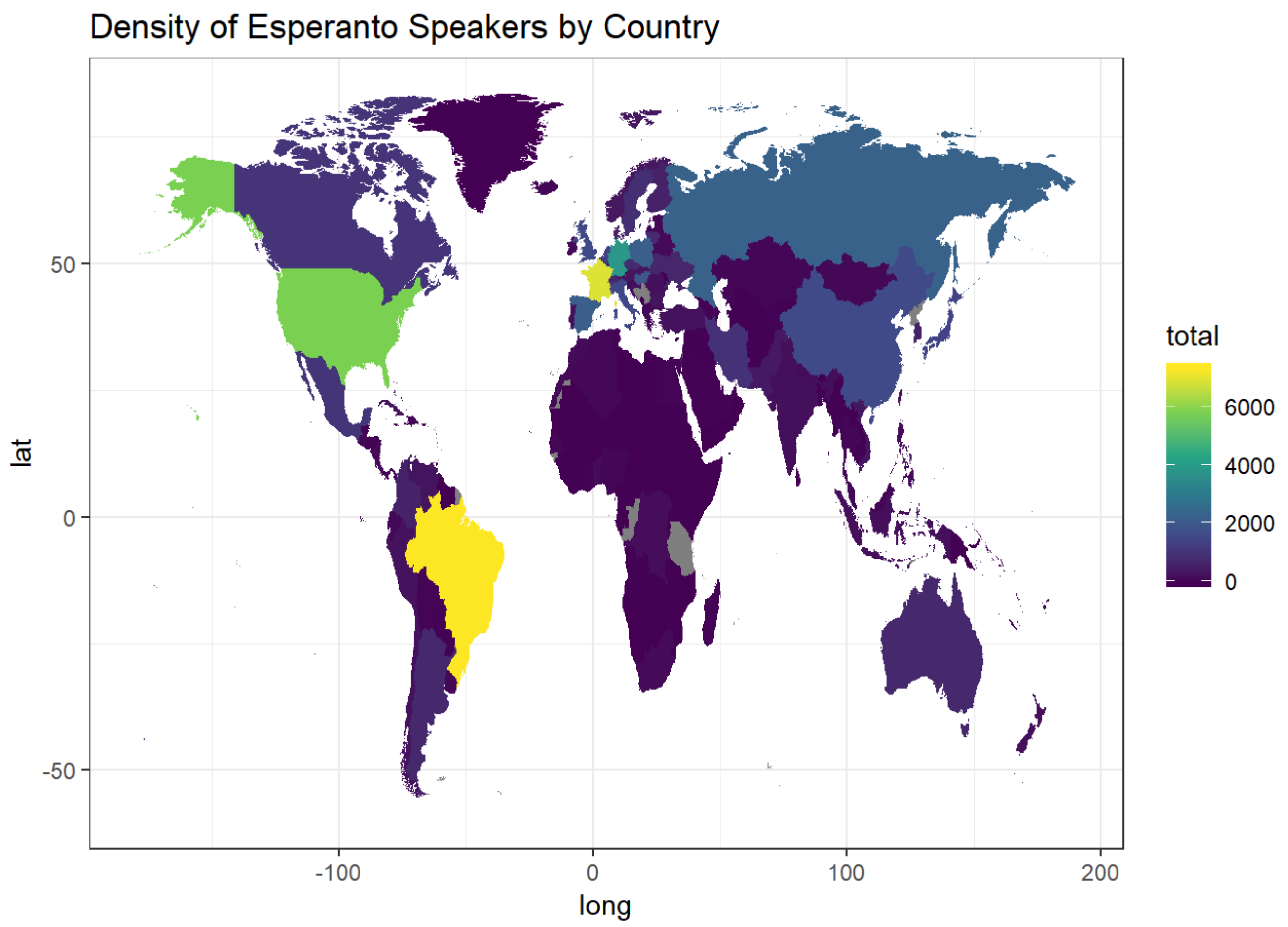
```
## The following object is masked from 'package:purrr':
##
##     map
```

```
map_data("world") %>% group_by(region) %>% summarise() %>% print(n = 6) #this gets us a list of
 all the countries that map_data knows about, so we can join and make sure there willbe no confu
sion.
```

```
## # A tibble: 252 x 1
##   region
##   <chr>
## 1 Afghanistan
## 2 Albania
## 3 Algeria
## 4 American Samoa
## 5 Andorra
## 6 Angola
## # ... with 246 more rows
```

```
newData <- left_join(map.world, data, by= c("region" = "country"))

newData %>%
  filter(region != "Antarctica") %>%
  ggplot(aes(x=long, y=lat, group=group)) +
  geom_polygon(aes(fill=total)) +
  scale_fill_viridis_c() + theme_bw() +
  ggtitle("Density of Esperanto Speakers by Country")
```



Any place in the world on this map that is not colored gray has people who speak Esperanto.

Although there are not as many people in the world who speak Esperanto as who speak other languages, it is faster to learn, and it was designed to be easy by its founder. There are unique people all over the world who you will be able to talk to once you learn this simple language. Instead of boxing yourself in and learning a language that will only grant you access to a few countries at most, learn Lingvo Internacia, the language of the world, Esperanto.

Free resources exist at Duolingo (https://www.duolingo.com/) and Lernu (https://lernu.net/en). Already know some Esperanto? Come join us for a chat on Amikumu (https://amikumu.com/), or on our subreddit (https://www.reddit.com/r/esperanto)!