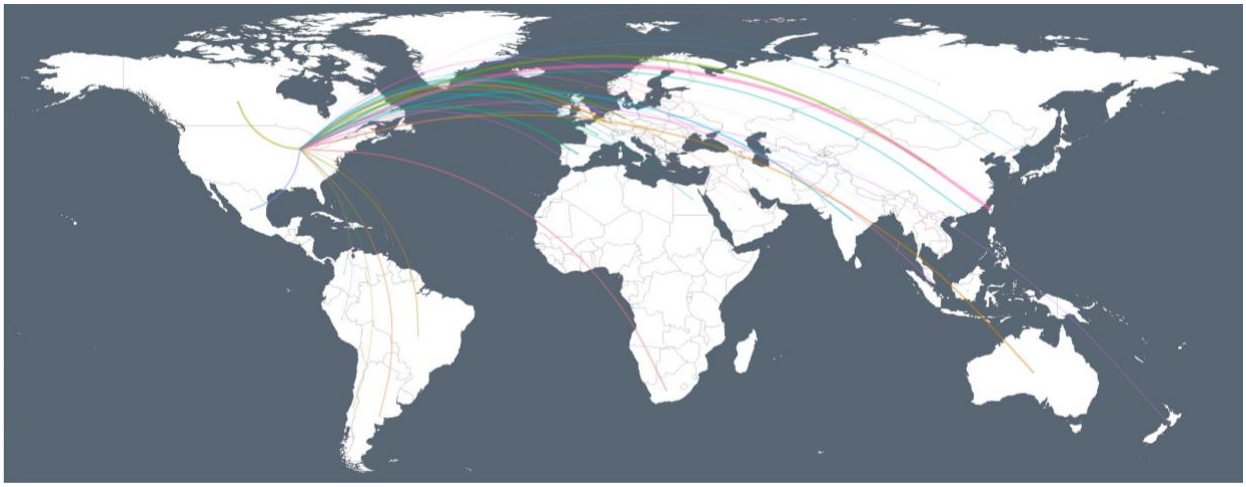
Cybersecurity Through DNS Traffic Project Report

Shiny App team: Haoda Song, Donna Nguyen, Brian Geoghegan;
Geographic team: Nate VanCuyk, Caroline Denecke, Faith Obringer;
Time-Series team: Bretho Danzy, Matthew Bandusky, George Sacco



Executive Summary

In an effort to better protect websites from cyber attacks, a large amount of data was collected through corp.com, a digital honeypot. From there, malware researchers could investigate hackers' behaviors based on these data. To help the researchers understand the data easily and efficiently, we sought to explore some key insights of the data, analyze and visualize the data spatially and temporally, while also displaying our results by an interactive shiny app.

In this report, we mainly show our results in four sections. The introduction provides backgrounds, motivations and goals of our project. There are 2 sections showcasing time series and geography findings. Finally, the description of our shiny app introduces how users could manipulate the app to explore aspects they would like to know from the data. Besides time series and geography insights, this report also shows how the original data were manipulated and the methodology used, including details of how we handled and analyzed the data. At the end of the report, users could find the codes that we used to analyze and visualize these information.

Introduction

Background

Today, almost any software can be misused or abused. Often, there is a profit to be made, or a political goal to be reached, by attacking computer networks and exploiting their programs. To protect the softwares/websites from cyberattacks, theft of data and operational fraud or to minimize these risks, cybersecurity is significant to many companies and government entities. Through corp.com (a digital honeypot which informs malware researchers on hackers' behaviors), DNS traffic data was collected from April 26, 2017 to October 15, 2019. This contains roughly 2.4 billion network traffic observations that will be explored.

Motivation

Corp.com, the honeypot, collects the records left by users and helps malware researchers learn about hackers' behaviors. Based on these DNS records, our project seeks to assist the researchers by answering questions such as: How can researchers understand the flow and pattern of DNS traffic directly in a few minutes, instead of just getting tons of complex information? Is there a way that allows researchers to manipulate the information and quickly see what they would like to concentrate on? Keeping these questions in mind, we work together to create the app which we will go into details later.

Goals and Questions for Each Group

The overall goal of this project is to create a shiny dashboard (shiny app) that can display the entire scope of DNS data for the researchers. It should take advantage of the temporal, spatial and categorical nature of the data. The researchers should be able to quickly absorb the recent history of communications going through the honeypot.
To complete the overall goal, we separate the tasks into three teams,
Geographic (Spatial) team: Nate VanCuyk, Caroline Denecke, Faith Obringer;
Time-Series (Temporal) team: Bretho Danzy, Matthew Bandusky, George Sacco;
Shiny App team: Donna Nguyen, Brian Geoghegan, Haoda Song.
The goal of the geographic team is to explore the geographic relationships within DNS and to create visualizations, such as a map with DNS characters or elements which can be customized by the researchers to detect geographic patterns. Similarly, the goal of the time-series team is to convert that DNS characters/elements into time series plots and find observations into how the traffic behaves. Finally, the goal of Shiny app team is to combine and modify all visualizations provided by the previous team into one interactive Shiny dashboard which can be easily manipulated by the researchers. The data can be more fully explored using the app.

Cleaning and Organizing the Data

Brief Explanation

In cleaning the data we wanted to capture time series data showcasing the following aspects:

- Amount of traffic to the server per day or hour.
- Amount of unique IP traffic to server per day or hour.
- Amount of traffic by DNS resource record type per day or hour.
  - Resource record types indicate the format of the data and it gives a hint of its intended use (e.g. *MX* indicates e-mail server traffic).
- Amount of traffic by country per day or hour.

Raw Data

We received the data in 886 text files in the following format:

```
10-Aug-2019 23:59:56.607 queries: info: client 4.7.22.65*#63190
(101.64.10.10.in-addr.arpa): query: 101.64.10.10.in-addr.arpa IN PTR -
(72.28.99.90)
10-Aug-2019 23:59:56.634 queries: info: client 3.83.62.177*#17571 (IP-
C0A8F365.oakriver.corp.com): query: IP-C0A8F365.oakriver.corp.com IN
SOA -EDC (72.28.99.91)
10-Aug-2019 23:59:56.650 queries: info: client 156.154.12.235*#37181
(alv.corp.com): query: alv.corp.com IN A -EDC (72.28.99.90)
```

Data Processing

We used SAS to parse each line and merge the text files into one resulting in the following dataset:

Obs	client	client_port	record_type	date	time	record_class	destination	num	continent_name	country_code	country_name
1	(pki.corp.com):	.	-EDC	21042	34846.32	0	0	1	NA	NA	NA
2	1.0.252.134	44812	MX	21702	28779.88	1	1	2	Asia	TH	Thailand
3	1.0.69.57	16387	A	21066	37926.64	1	1	3	Asia	JP	Japan
4	1.0.69.84	34010	A	20941	79652.66	1	2	4	Asia	JP	Japan
5	1.0.69.84	36463	A	21017	36417.58	1	1	4	Asia	JP	Japan
6	1.0.69.84	47056	A	21019	35393.40	1	2	4	Asia	JP	Japan
7	1.0.69.84	5531	A	21041	75282.73	1	1	4	Asia	JP	Japan
8	1.1.255.230	50008	A	21013	44396.85	1	1	5	Asia	TH	Thailand
9	1.1.255.230	50004	A	21013	44397.35	1	2	5	Asia	TH	Thailand
10	1.1.255.230	50025	A	21013	44397.36	1	2	5	Asia	TH	Thailand

From here we used Structured Query Language (SQL) to aggregate the merged data into more reasonable datasets. We then read our data into R refining mainly the Date variable into an intelligible format. This leaves our data in following generic format.

Generalized Data Structure

- Date: date or date-time of event
- Group: Group variable which contains categorical data (i.e. Country or Resource Record Type)
- Count: amount of server traffic that occurred within the time period

R Sample Data

Here we have printed the first few lines of our data to give an idea of our finalized data structure. All counts are given in thousands.

Daily DNS Traffic

Date	Count
2017-04-26	1404.456
2017-04-27	1407.740
2017-04-28	1443.900
2017-04-29	1142.731
2017-04-30	1121.184
2017-05-01	1328.932

Hourly Country Data

Date	Group	Count
2017-04-26 02:00:00	Antigua and Barbuda	0.007
2017-04-26 02:00:00	Argentina	0.038
2017-04-26 02:00:00	Australia	0.354
2017-04-26 02:00:00	Austria	0.012
2017-04-26 02:00:00	Bangladesh	0.031
2017-04-26 02:00:00	Barbados	0.001

Time Series Analysis

This section contains background to the different generated plots and methods that follow.

What is a Time Series?

A time series is a sequence of data collected over time at uniform intervals. Time series data are statistically dependent observations with the theory that the current observation is determined by past observations. In that the most important aspect of time series modeling is correlation. In this report we will not perform any modeling, but we will utilize certain time series modeling tools. The two tools we will utilize are the autocorrelation function and periodogram.

Diagnostic Tools

The Autocorrelation Function

The autocorrelation function, ACF, of a time series is a function of the autocorrelation of arbitrary current observation Y_t with a given prior observation Y_s, where s is less than t. The ACF utilizes autocovariance, the linear relationship between two time points $Cov(Y_t, Y_s)$. What ACF is finding is the correlation between observation Y_t and previous observations in time. The formula of the ACF is:

$$Cov(Y_s, Y_t)/\sqrt{Var(Y_s)Var(Y_t)}$$

The ACF plot will list a lag number on the x-axis and the ACF value on the y. The value at each lag indicates the correlation between Y_t and a prior observation indicated in the lag number; for example lag 1 is the correlation between Y_t and Y_{t-1}. There is a dashed horizontal line above and below the x-axis, if a lag surpasses one of the dashed lines there is a strong correlation.

One of the things we will see in the ACF plots of DNS traffic is the existence of a seasonal trend, a type of mean trend that is periodic in nature. A seasonal time series is nonstationary[1]. In modeling time series you need to account for a mean trend and in a seasonal time series the mean trend is modeled via $\beta_1 cos(2\pi wt) + \beta_2 sin(2\pi wt)$ where w is the frequency. In order to find the frequency we use the periodogram.

The Periodogram

Any seasonal time series can have any of the following harmonic frequencies $\frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \ldots, \frac{1}{n}$ where n is the total number of observations. The periodogram is a tool that calculates the significance of all possible frequencies in a seasonal time series and plots them using the following method:
Let $j = 1,2, \ldots, \frac{n}{2}$ and $w_j = j/n$. Representing the time series as:

$$X_t = \sum_{j=1}^{n/2} \beta_{j1} cos(2\pi w_j t) + \beta_{j2} sin(2\pi w_j t)$$

[1] This indicates the time series has a nonconstant mean or non constant variance

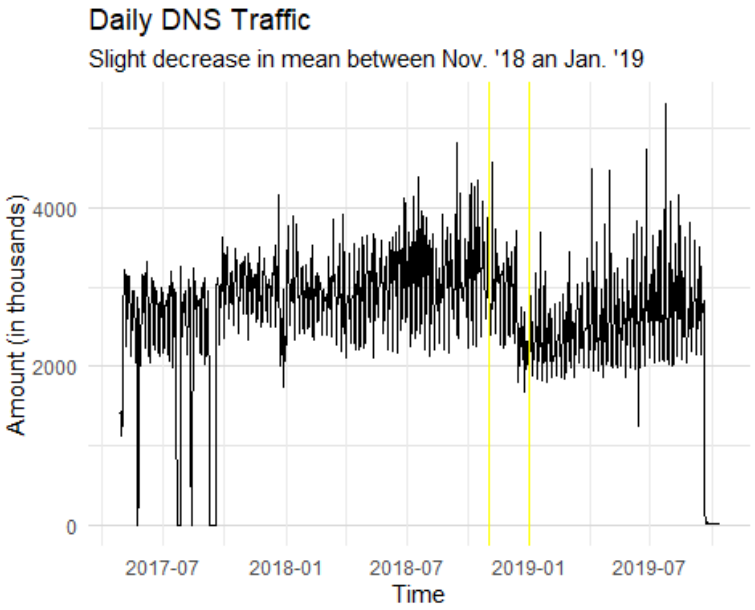
The betas are treated as regression parameters for each frequency. The amount of variation at a certain frequency in the periodogram is found using the fitted regression parameter coefficients:

$$P(\frac{j}{n}) = \widehat{\beta_{1J}^2} + \widehat{\beta_{2J}^2}$$

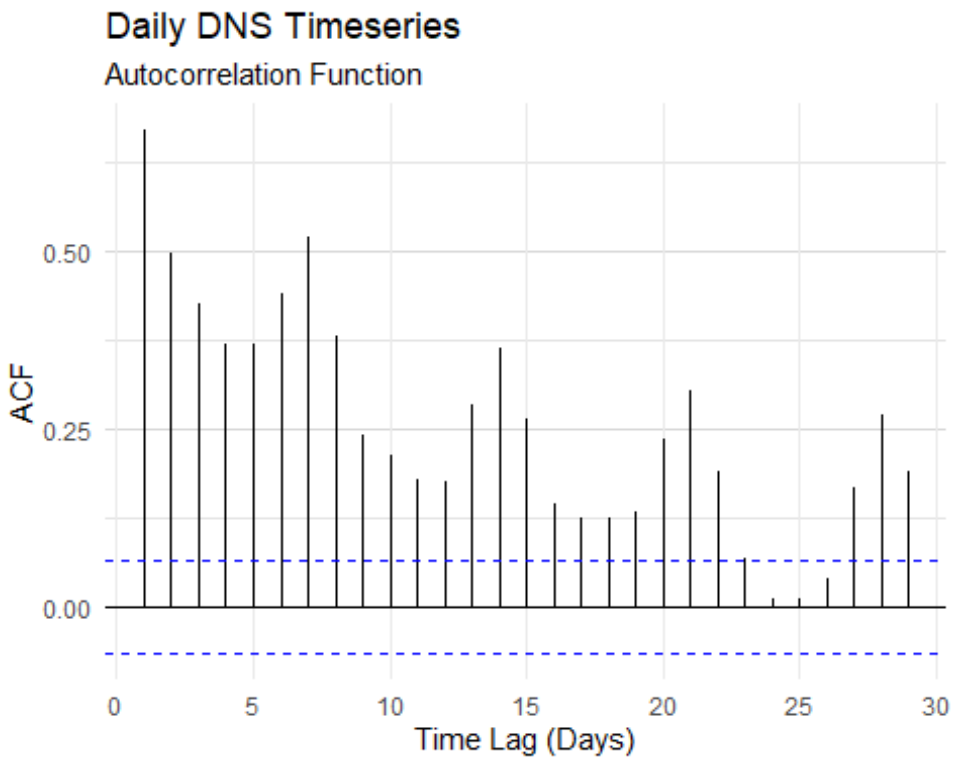
The periodogram then graphs each $P(\frac{j}{n})$. Based upon the ACF plot if there is a seasonal trend there will be a large spike at one of the frequencies in the periodogram. The frequency with the large spike is then used to model the seasonal trend in a time series. More detail on the periodogram can be found in the references.

Plots and Observations

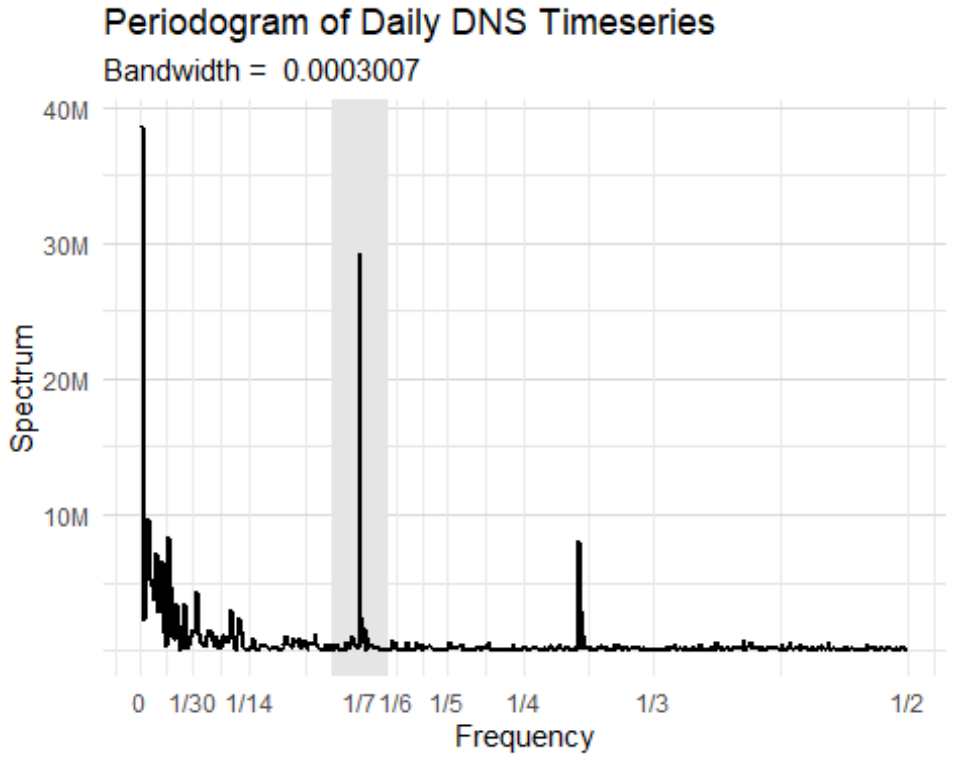
To begin with our plots, we first are going to plot the totality of traffic by day and by hour.



We can see a few interesting observations. First the daily traffic has an ever so slight mean increase from the start until about November 2018. The mean of traffic then falls off until January 2019 where it subsequently increases again. This time series does not appear stationary, but it could be. We will plot the ACF plot to see if it is stationary and detect other underlying trends.

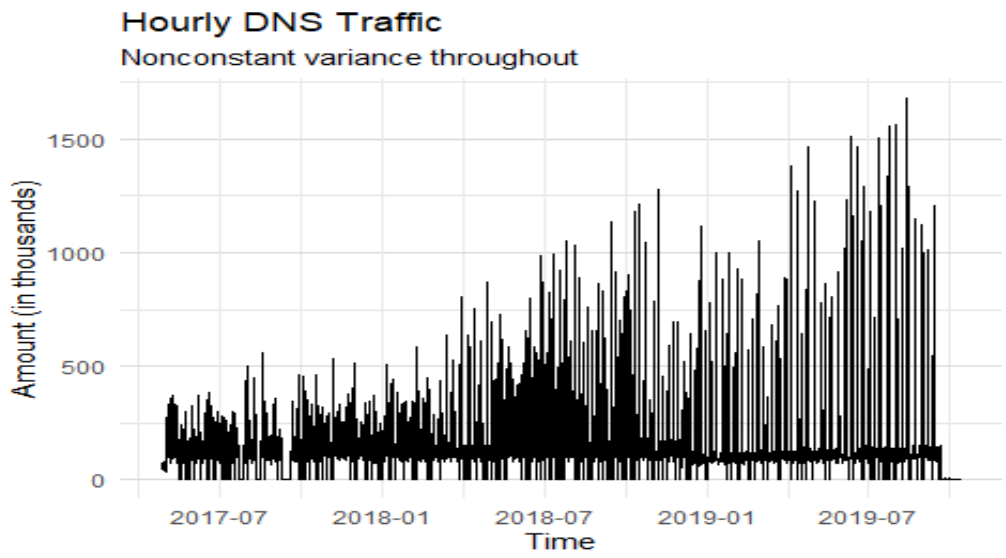


In the ACF we can see that there is some periodic trend in the time series; the lags have a periodic decrease to increase repeating every 7th lag. To investigate this further we will draw the periodogram to try and determine the frequency of the trend.

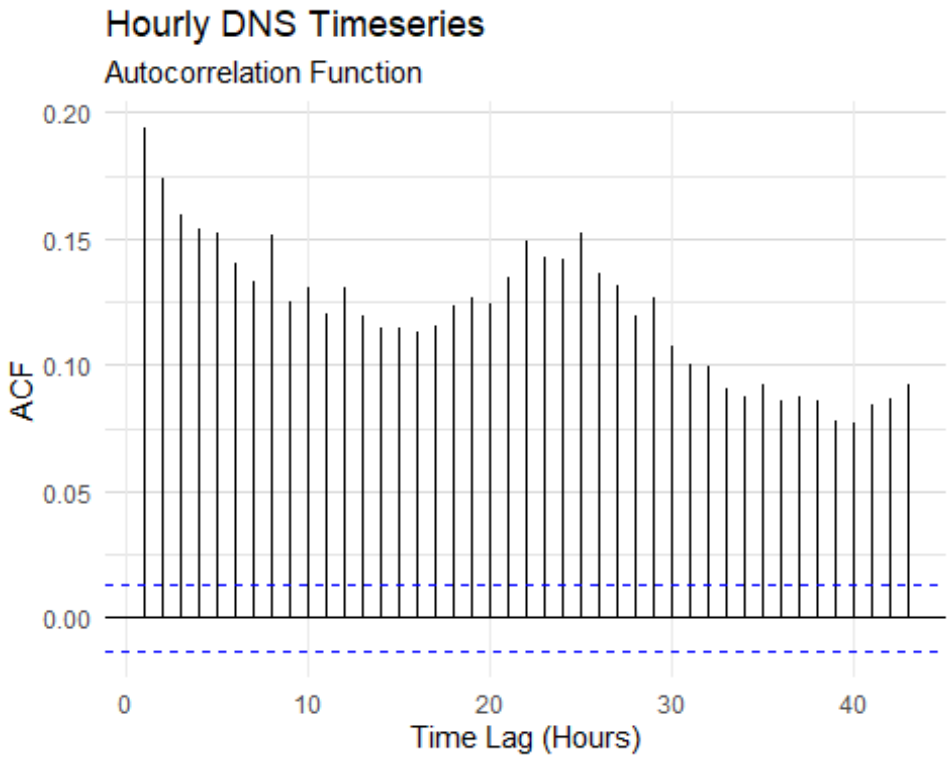


We can see the periodogram has a spiking frequency at 1/7. This suggests that the period of the time series occurs every 7 days or weekly. When you zoom in on the time series so you only see a handful of weeks the weekly period is apparent. The periodogram also has a spike very close to zero possibly indicating a yearly trend, but it is unclear what the harmonic frequency that spike is indicating.

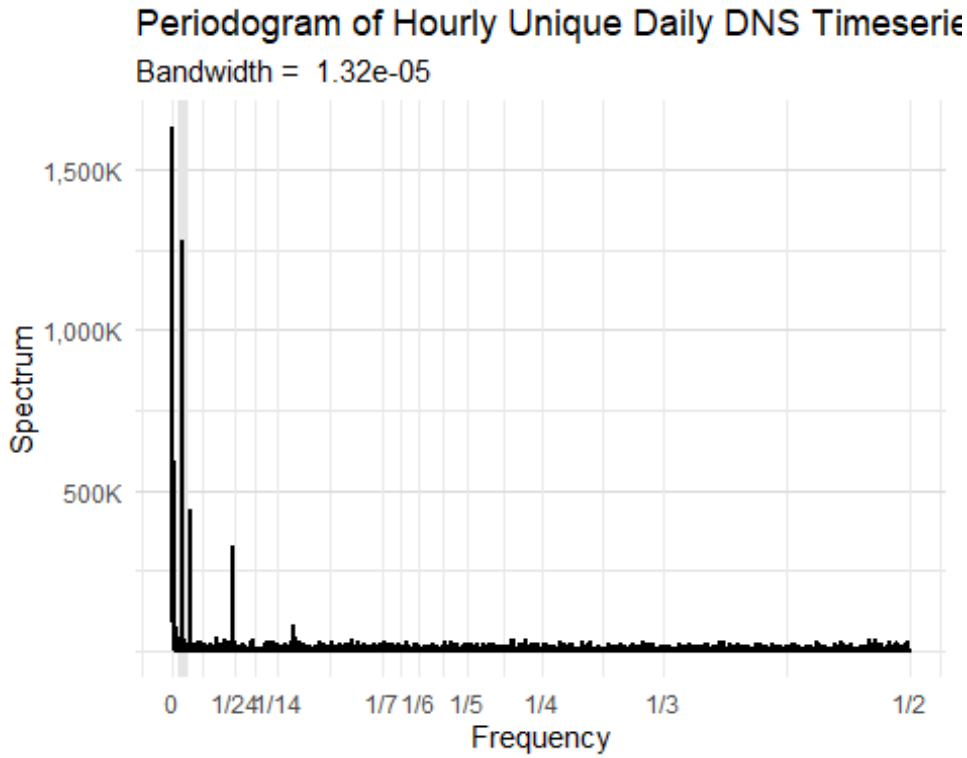
Next, we will plot the total hourly DNS traffic and see if this periodic trend is present as well.



What we can see first is that the variance is clearly nonconstant. The DNS traffic has very large spikes at certain hours then the count returns to a normal level of traffic. The traffic spikes also get larger in magnitude over the course of time. We will plot the ACF for the hourly plot to further explore this behavior.

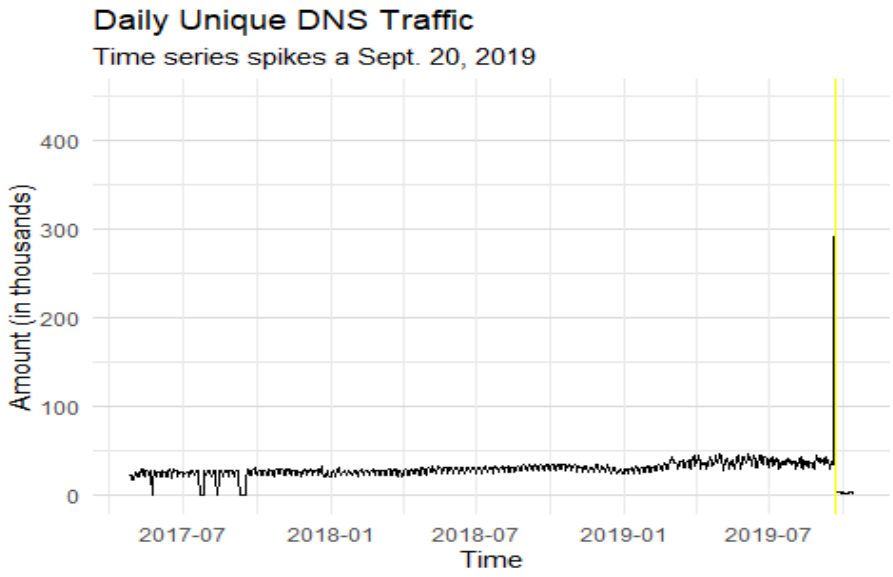


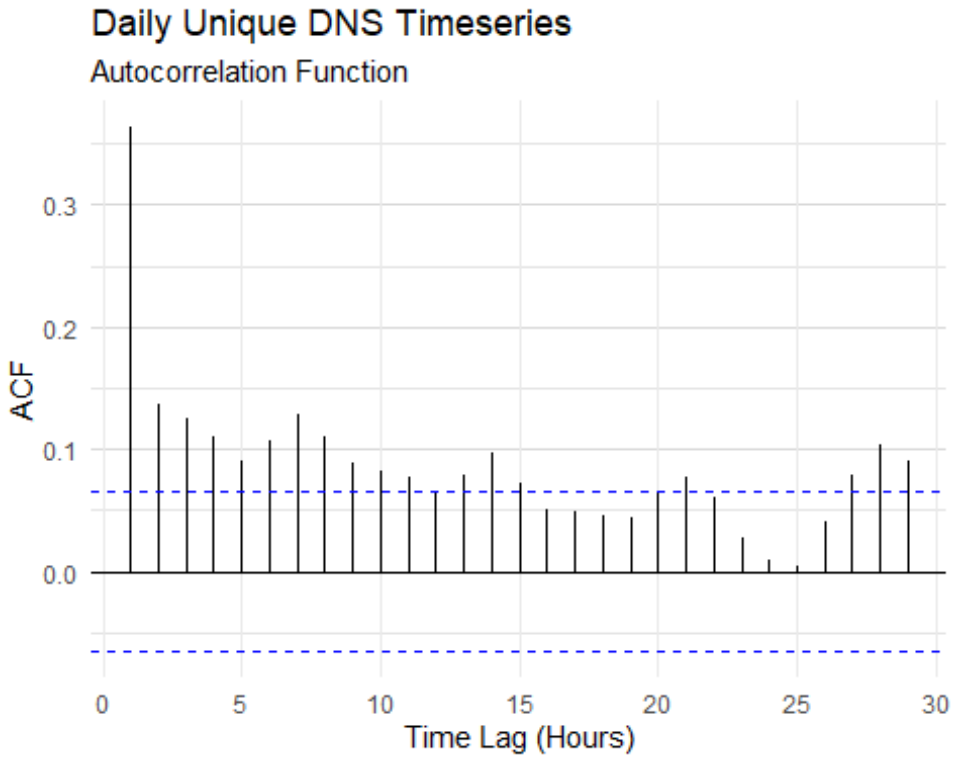
We once again can see that periodic trend in the ACF, but the period appears to be much larger, possibly every 24th lag or so.



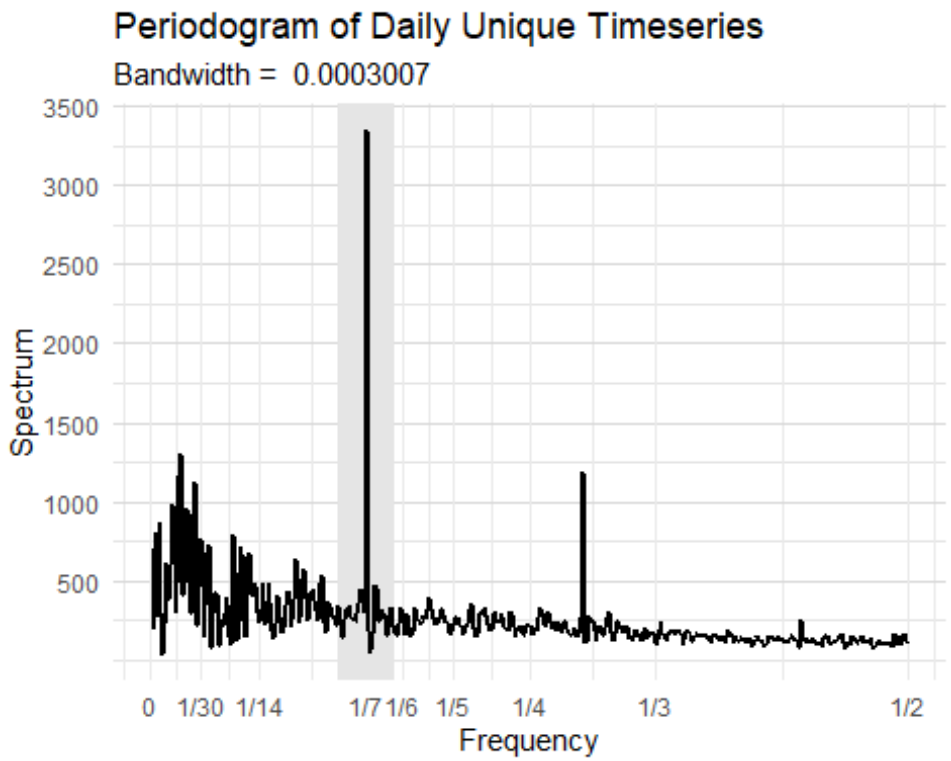
First we can see there is one very distinct spike in the periodogram. This occurs at 1/168 and a smaller one at 1/24, which just so happens to coincide with the number of hours in a week and a day. The weekly periodic trend is occurring in the hourly data as well. Now that we have delved into the total traffic let's break it down into smaller data sets.

Below are the daily and hourly time series of unique DNS traffic.

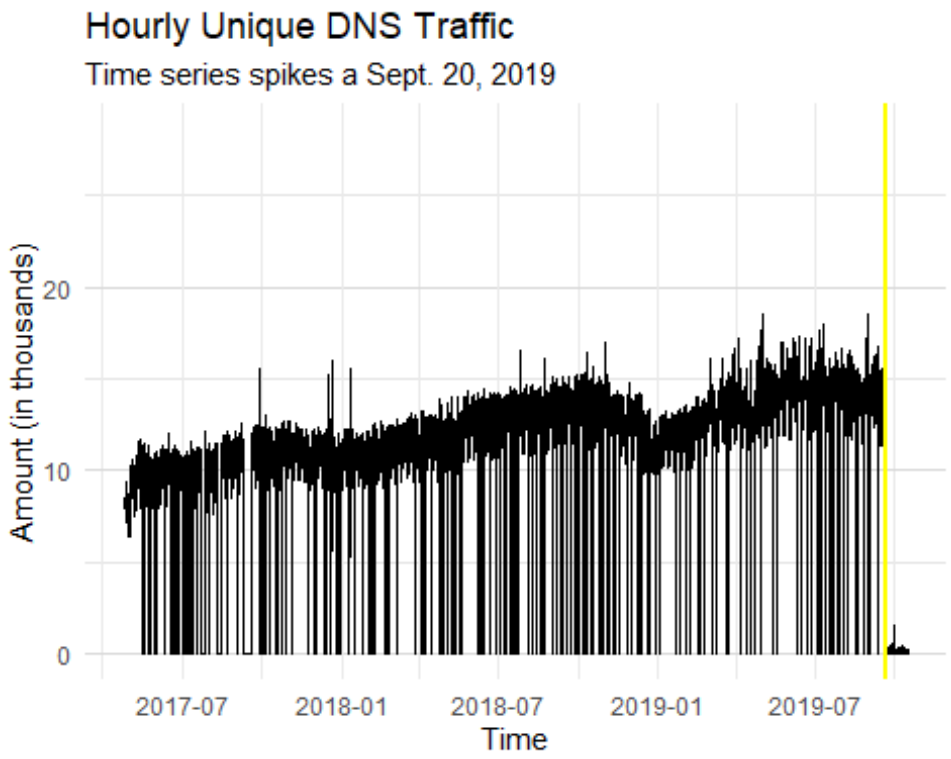


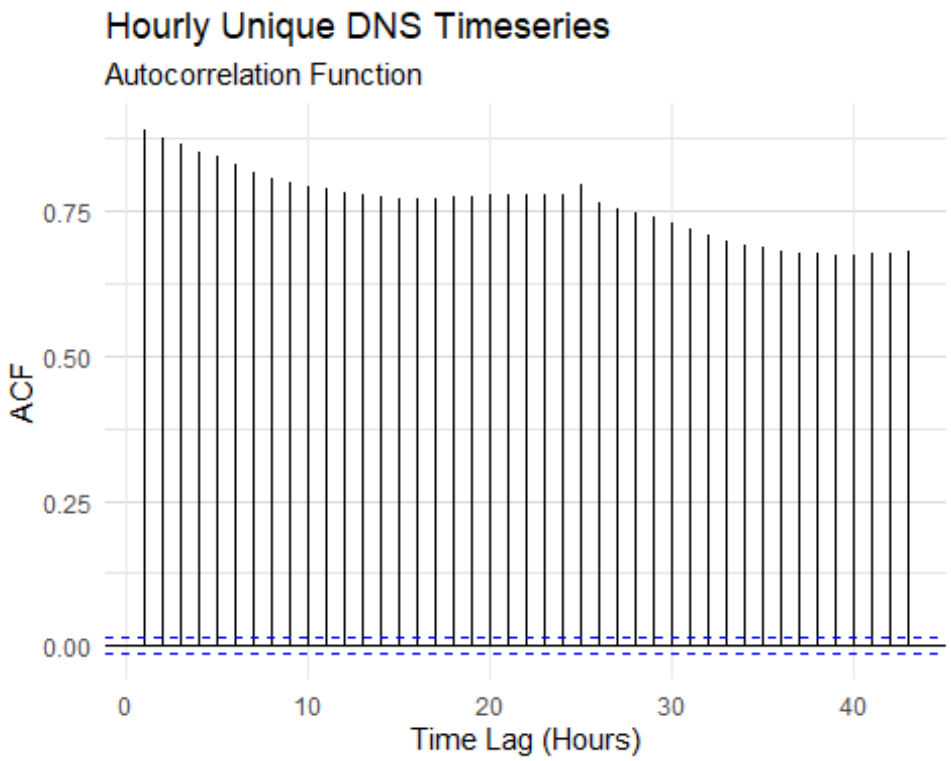


We see one major spike occurring on September 20th 2019. We also see the resemblance of a periodic trend in the ACF.

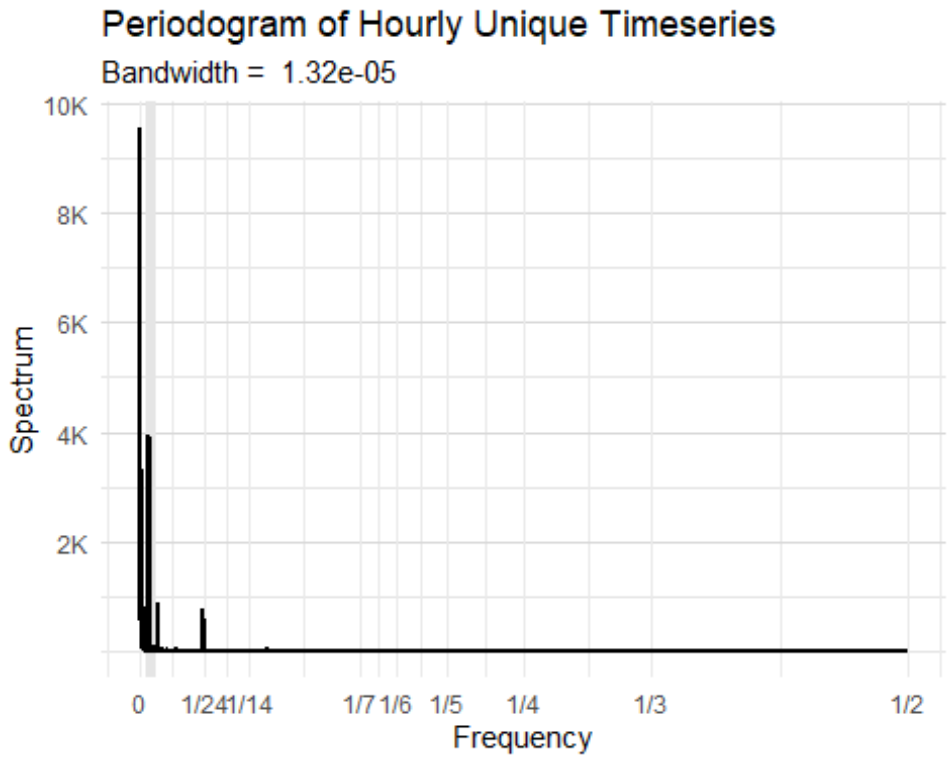


Drawing the periodogram we once again find the spike at 1/7 repeating weekly. These results are very similar to the total daily traffic. The hourly unique DNS traffic time series is also very similar to total hourly traffic.





The volatility is present but less extreme than in the total traffic. We also see the clear periodic trend once again in the ACF.



We see that periodic trend once again occurring at 1/168 and a smaller spike at 1/24.

Observations Regarding Resource Record Type

We also generated time series for the daily and hourly observations of each DNS record type. Due to the incredibly large number of record types we will not be fully breaking down every single record type. Instead we will look at more broad observations and talk about some of the larger sources of traffic.

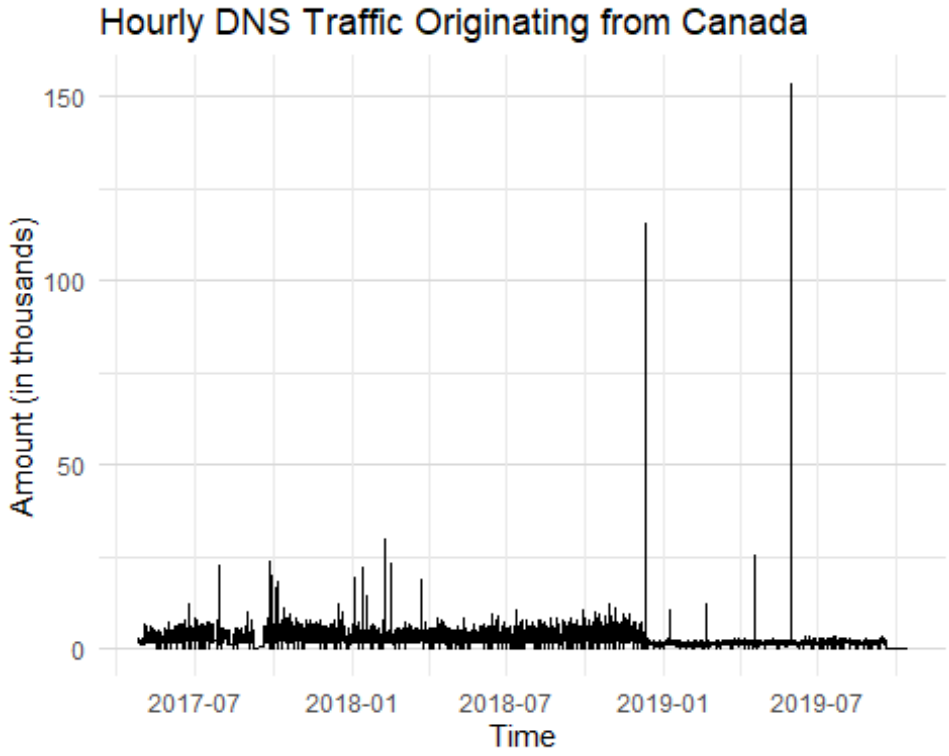
Address record comprised the majority of DNS traffic. Record type IPv6 address record has the exact same form as type Address record, but IPv6 address record has fewer observations. A lot of other record types have very infrequent sporadic usage such as Address Prefix List, Certificate record, DHCP identifier, DNSSEC Lookaside Validation record, DNAME, DNS Key record, Host Identity Protocol, and many others. Record type Mail exchange record has a strong dependence; this means observations above the mean are likely to follow other observations above the mean; likewise observations below the mean are likely to follow other observations below the mean. Start of authority record has a general increase in traffic over time. Type Service locator follows a similar traffic pattern to daily total traffic.

All these can be viewed under the time series tab of the app. In that tab there is a report types section. In the report types section, you can view the daily and hourly time series of any of the DNS record types, and you can plot up to five different record types on one plot.

Observations Regarding Country

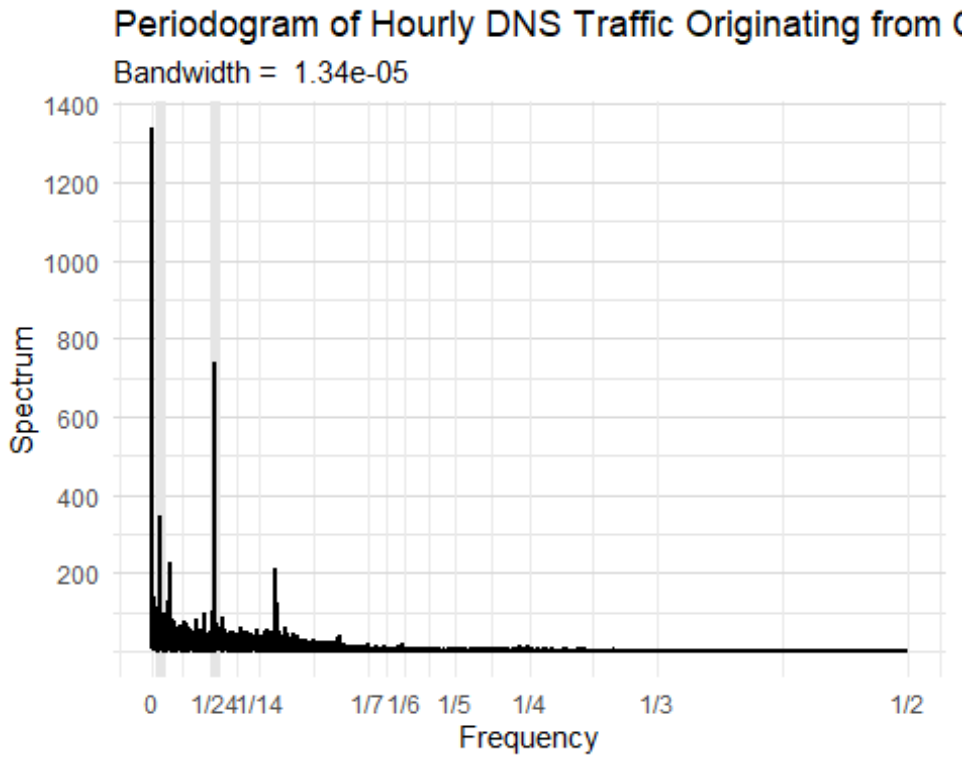
Like record type there are far too many countries to do a full breakdown of each. Instead we will give broad observations and focus on where most traffic came from.

First, the overwhelming majority traffic comes from the United States. As expected, the traffic patterns for the United States mirrors the total daily and hourly traffic patterns already observed. Most small countries or less developed countries have small sporadic levels of traffic; some examples are Albania, Kosovo, Bermuda, Somalia, and Syria. In countries with more consistent traffic, such as Canada, UK, China, Taiwan, Japan, Germany, and Mexico, you can see in the hourly traffic a clear cyclic pattern within a day. This can best be explored within the app. We will try our best to illustrate by looking at the static Canadian hourly traffic.



In this time series you very cyclic nature often peaking in the afternoon hours (3 - 8 pm)[2]. We will also plot the periodogram to see the period within a day.

[2] If you zoom in on the app time series the cyclic nature is clear, it is hard to visualize in the full plot.



And we see yet another periodic trend occurring at 1/168. We also see this accompanied by a peak at 1/24 indicating there is also a daily trend.

In the time series section of the app there is a subsection labeled countries. In that section you are able to plot the daily and hourly time series of any country that has observable traffic. Additionally you can plot up to 5 different countries on the same plot for comparison.

Geographic Analysis

Data Handling

Initially, the data from just a week was used in order to be more manageable in size. The files for the week were read into R, homogenized as some of the data would read in with an extra column, and then combined into one dataframe to be worked with. The IP addresses then had to be extracted from the given column as there were other characters in the same column. Once the IP addresses were formatted correctly, we used the ip_api() function to find the geographic information; we later realized that this way of collecting geographic information was flawed in multiple ways. The first problem was that this function was virtually impossible to use on large datasets. Since this function had to ping a database, the number of IP address requests we could make in a

given time period was capped, rendering it impossible to get the needed information in a timely manner. To compensate for the first issue, a subset of the data was used to at least get some of the geographic information to test visuals; this led to the realization of another major issue. Even if the dataset was small enough to get the geographic information, the ip_api() function only gave geographic information on about 10% of the IP addresses. This was tested multiple times to see if it was a problem with a certain subset of the data, but after repeated testing it was seen that the results were around 10% every time.

Thanks to feedback from another group we switched over to the maxmind() function, which was much more effective at both getting the geographic information quickly, but also at getting much more of the information. This was used with the week's worth of data to better tune visuals and do some EDA.

For the final visuals, we are using aggregate data that summarised overall frequencies of country activity and DNS types. This aggregate data was then combined with mapping data from the *maps* package in R. This presented some issues on its own as the data wasn't initially compatible to be combined, so further steps had to be taken to ensure visuals generated properly. Once these issues were fixed and the data combined, it was then usable for the maps to be made.

Visualizations

The goal of the geographic part of this project is to look for trends in locations for the DNS traffic. In order to do this, we'll be generating choropleth heat maps using R. We originally planned to include lines showing the path between the beginning and ending points of the network, but this proved to create too busy of plots to be readable, and therefore were virtually useless in conveying the necessary information.

R Packages Being Used

- assertthat
- dplyr
- gapminder
- ggmap
- ggplot2
- ggraph
- ggthemes
- igraph
- iptools
- maps
- purrr
- readr
- rgeolocate
- tidyverse

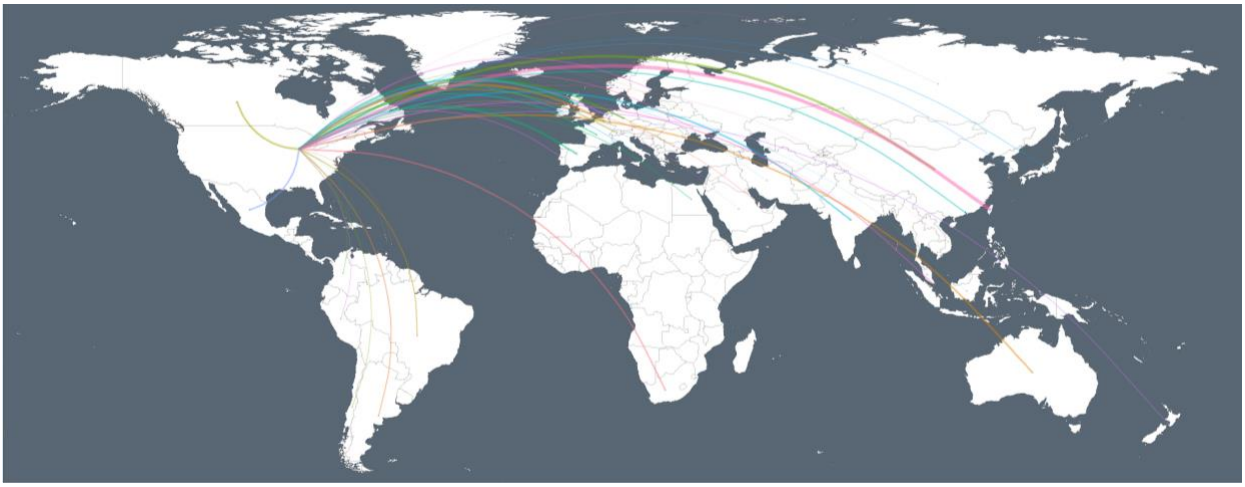
Exploratory Data Analysis

The geographic portion of the project went through multiple stages of EDA as we found better ways to read-in the ip addresses as discussed earlier.

Visualizations

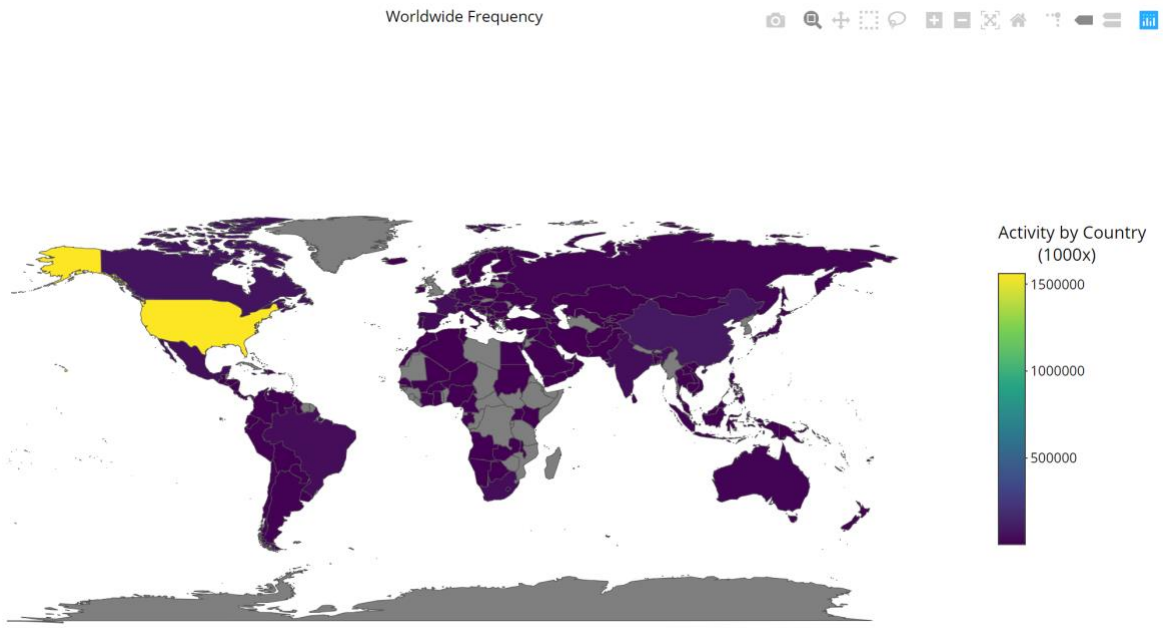
As stated earlier, our original idea was to show connections between countries to convey the amount of activity having taken place. As can be seen in Figure 1, it was quickly realized that this visual was both overwhelming and hard to follow, prompting us to change our approach.

Figure 1



After scrapping the idea of a connecting map, it was decided to use heat maps to better convey the information. Figure 2 is the final version of this map.

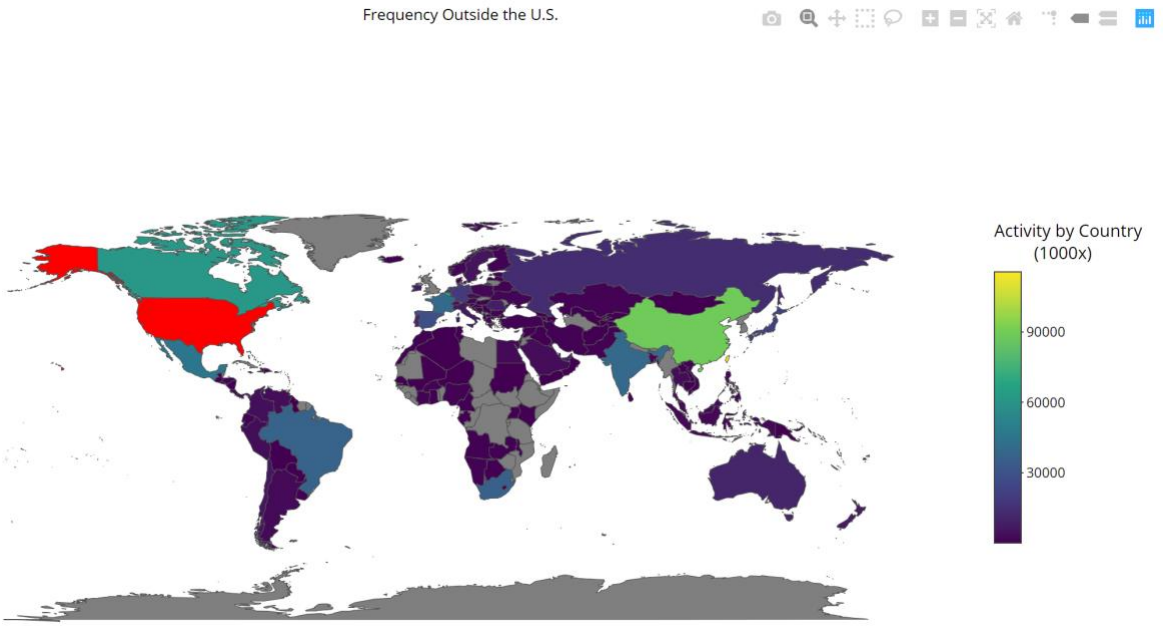
Figure 2



This type of visual was much less busy, and much better at conveying the information to someone viewing it. The issue we ran into with this is that it's easy to see the United States has far more activity than everywhere else, but difficult to determine the frequency among the rest of the world.

After digging a little further, it was found that the United States has approximately 1.89 times as much activity as the rest of the world combined, so another visual was made excluding the United States in order to see international information.

Figure 3



Countries on grey have no data (no interaction with the honey pot) and the U.S. is red to show exclusion from this visual

In Figure 3 it is much easier to see which countries outside of the United States are most active, the top five being: Taiwan, China, Belgium, Canada, and Mexico. A problem with this style of visual is geographically large countries are easy to see, while smaller countries are much more difficult, so we changed to using a plotly map. Figures 4, 5, and 6 are some examples of this functionality:

Figure 4

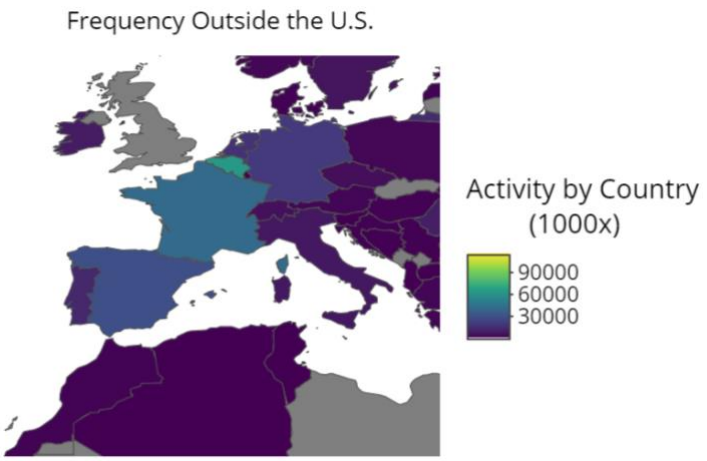


Figure 5

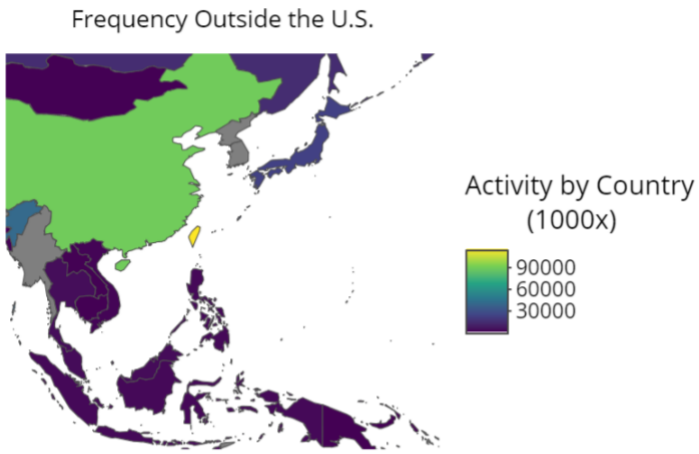
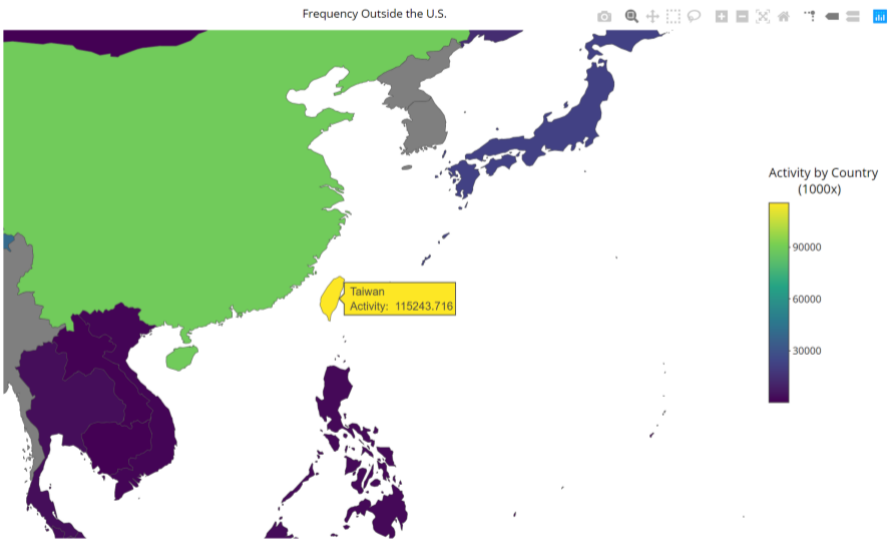


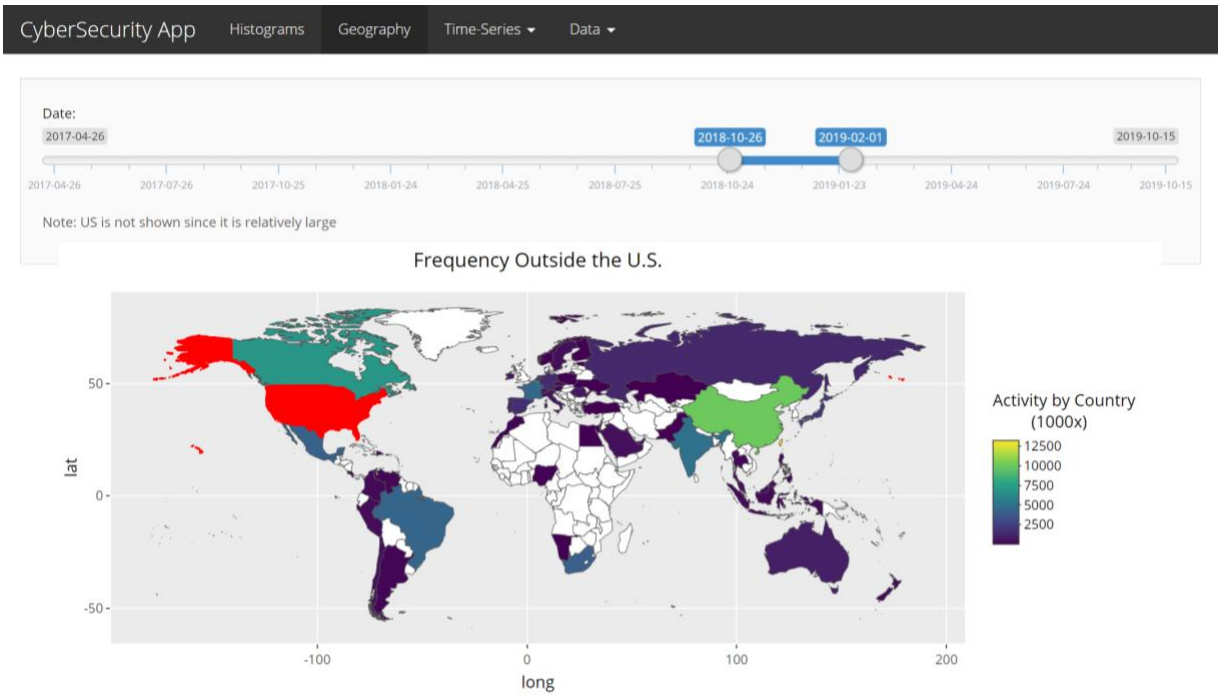
Figure 6



Using an interactive plot helps to mitigate the issue of some countries being harder to see, and also lends the ability to get information from hovering over each country. We chose to include the country name and the amount of activity, as these were the two main things being looked at.

Another advantage to this approach is it makes having a reference table unnecessary. Initially, some kind of accompanying table would have been necessary to see the actual numbers to compare each country, but this way you can quickly compare at a glance and then get more information through interacting with the map. Within the app itself, you are also able to pick a time frame in which you can view the same data (as shown in Figure 7), in order to see how the activity differed during different periods of time.

Figure 7



Ultimately, what we learned from this is that most of the activity stems from just a few countries (Primarily the U.S. followed by Taiwan, China, Belgium, and Canada) , relative to the rest of the world. Although looking at time periods there is some variation in country activity, the top countries during any period are fairly consistent, not showing any drastic changes at any given point in time. Another thing to note is the distribution of activity across the globe; most areas in the world had at least some activity except for noticeable areas of Africa being the main outlier. Perhaps the most interesting discovery was that outside of the U.S., Taiwan has the most activity by a large margin and Belgium has the 3rd most activity. As these countries are relatively small, it was not expected to see them so high in the list. Another interesting find was how far down the

list Russia is, having the 17th most activity. Given they have been in the news for infamously interfering in the 2016 U.S. Presidential Election, it was assumed they would have a much higher rate of activity.

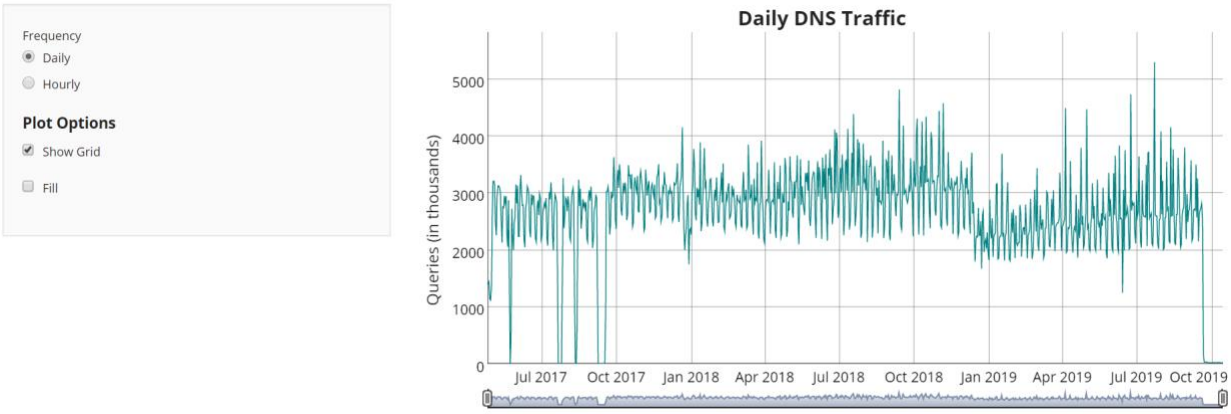
Caveats to consider about these results are the fact that this is data from just one honey pot, so it's almost impossible to say how representative this is of general global activity of this sort, and whether it can be said to be directly related to cyber security. Another is this data was collected during a limited time frame. Perhaps if we had data from 2016, Russia would be higher on the list, but the data is only from April 2017 - October 2019, so this is something to keep in mind. Another inherent issue, especially with this type of data, is one can look at the results and see what activity occurred, but in this context, we cannot know what exactly spurred the activity from each country at a given point in time.

Description of the Shiny App

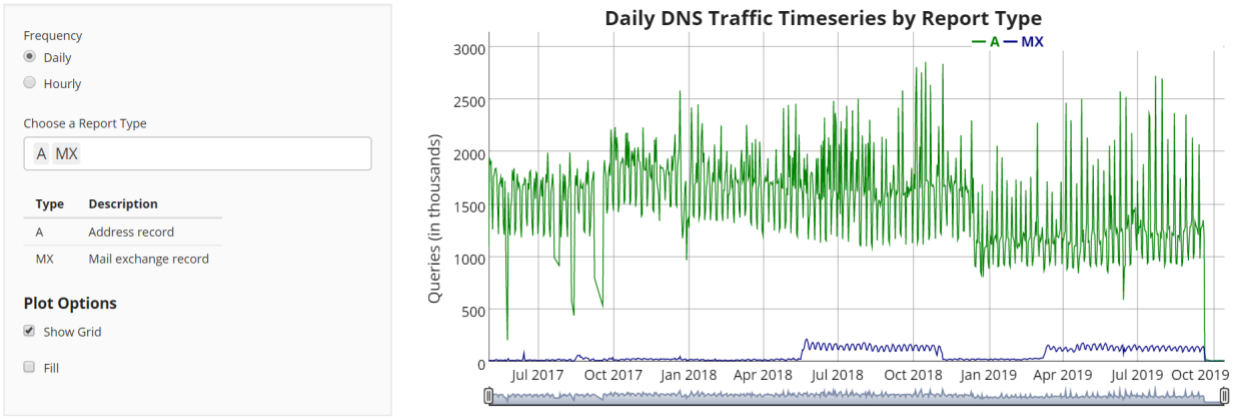
https://danzybm.shinyapps.io/dns_app/

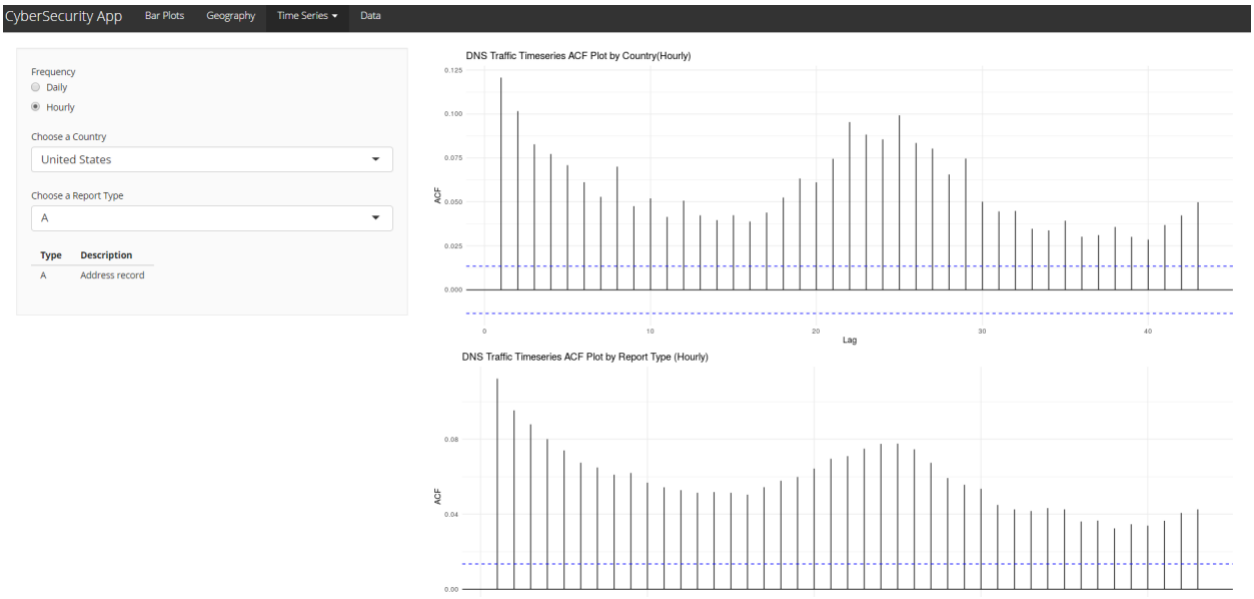
The goal of this CyberSecurity app is to provide an easy-to-use and powerful software tool to explore patterns of DNS records from two aspects: time series and geography. The application supports detailed time series analysis of the DNS records using either daily or hourly data as well as a geographic map of the daily queries by country. There are 4 main tabs on the horizontal navigation bar, namely: Bar Plots, Geography, Time Series and Data. There is also a drop down menu for the Time Series tab.

With regards to Time Series, users have the options to further explore in terms of Countries, Report Types, Unique IPs, ACF Plots, Periodogram and general DNS Traffics. Each of these six represents a drop down menu option and once clicked on, will present users with interactive time series graphs. The slider bar at the bottom allows users to specify which date range they want to focus on. Given the vast amount of data, all of these six can be summarized by either daily or hourly traffic. For example:

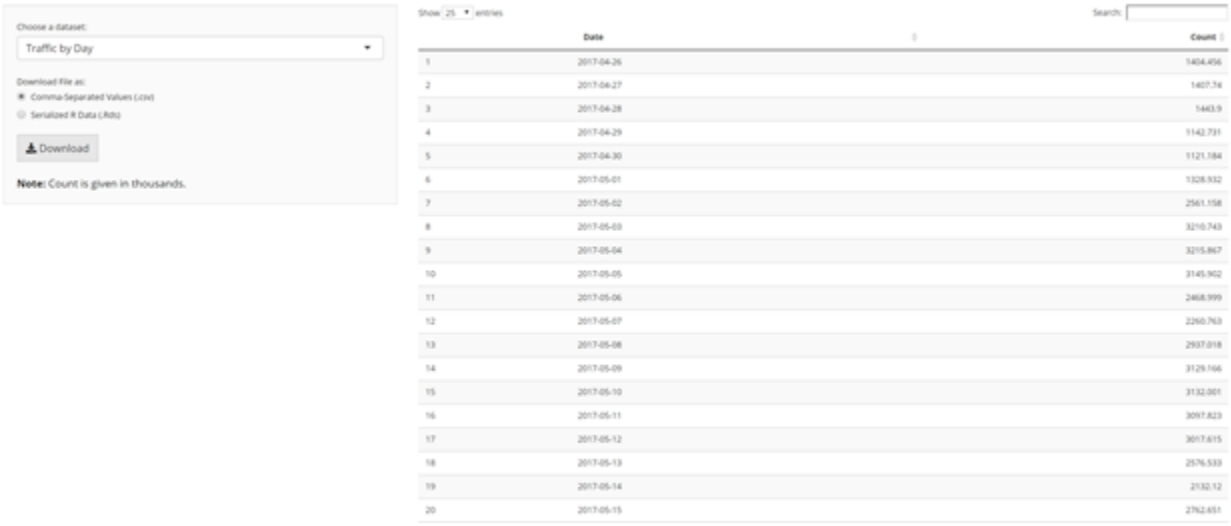


Furthermore, in the Report Types and Countries plots, users can choose which Types and Countries they want to display (limit up to 5 Report Types or 5 Countries per graph). ACF and Periodogram plots have options for users to choose both a Country and a Report type.



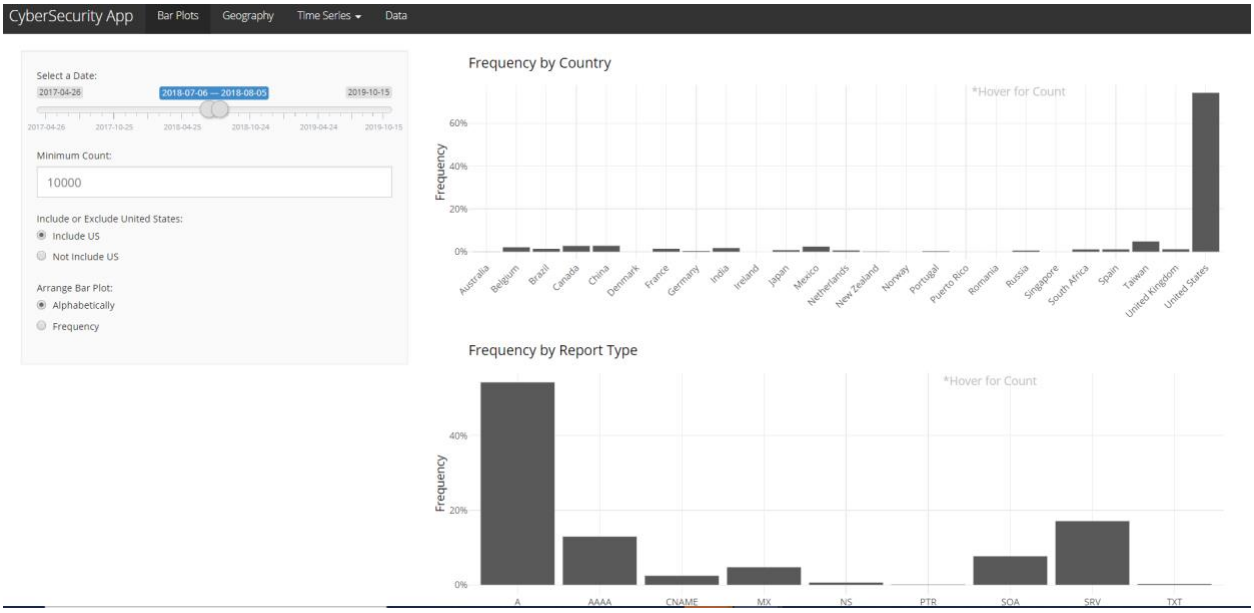


The Data tab stores some summary data tables such as Data Type dictionary, Traffic by Day, Traffic by Hour etc… It shows 25 entries per page for these tables with a search box on the right corner that users can type in. A new feature has been added where one can download the data as a .csv or .Rds file.

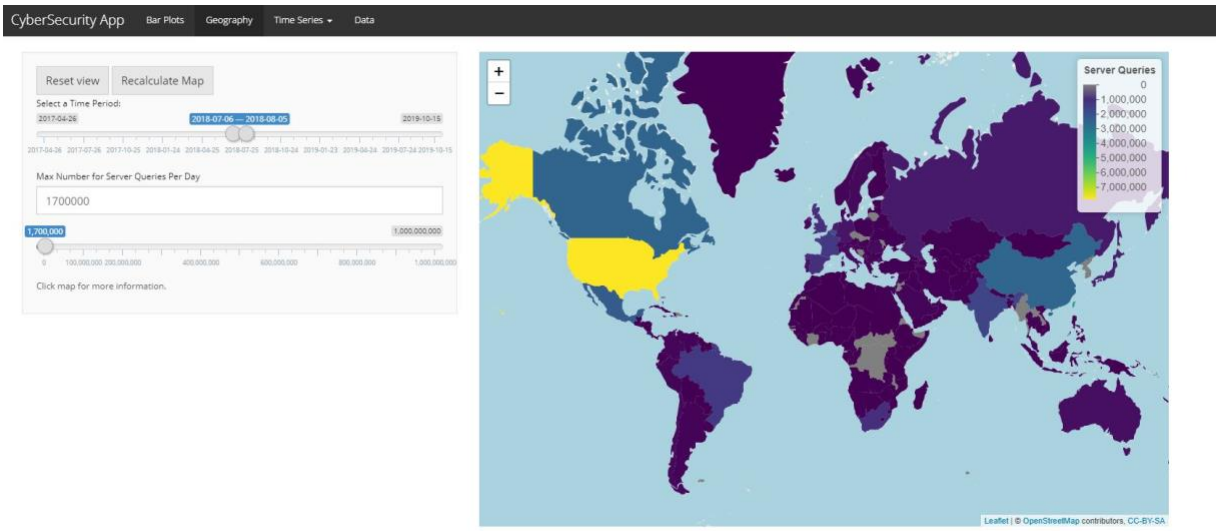


The Bar Plots tab will show the user two bar plots; the top one is showing you the frequency in percentage term by country (how much percent does that country represent in the total number of traffic for all countries) and the bottom shows you the frequency by report type. Users can adjust the date range they want to look at using the slider on the left. The United States' total number of transmissions is large compared to every other country, so we decided to add a toggle for users to decide whether to take it out. If you hover over each individual bar, you will be presented with the actual count number (or total transmission amount) besides the percent information. Minimum Count

is the smallest amount of traffic a country needs to have to be featured on the graph. Below is a screenshot that you can take a look at.



The final tab is the Geography tab which shows a map of the world detailing how many queries were recorded by country during the selected interval of time. You will also be able to click on a specific country and see the count along with the country name. You can also change the date range of queries by using the slider on the left side. Max Number for Server Queries Per Day is an input that helps mitigate the effect of the US skewing the color scale. The legend in the upper right-hand corner gives the user the cumulative number of queries over the time period limited by the daily maximum number of queries setting. There are two buttons in this tab: one is the “Reset View” button. This will rescale the map back to its default settings. The other button, “Recalculate Map”, initiates the creation of a new map based on the current user settings. The map is not refreshed until the “Recalculate Map” button is pressed. Below is a screenshot for this tab.



References

Penn State Eberly College of Science, The Periodogram, accessed 30 April 2020, <https://online.stat.psu.edu/stat510/lesson/6/6.1>

Appendix

Comprehensive Data Processing Description

This will show all data cleaning and code inline.

Getting Started

The data cleaning can be accomplished and many ways but this walkthrough assumes the following:

- You are operating on a Windows Computer

- 7-Zip is installed
- SAS is installed
- You have downloaded GeoLite2-Country.mmdb from MaxMind

- Lastly, you have your files arranged as follows:

File Structure

```
STA475
├ dns_compressed
├ dns_csv
├ dns_maxmind
|  └ GeoLite2-Country.mmdb
├ dns_RData
├ dns_sas7bdat
└ dns_uncompressed
```

Obtaining Data

This data uses a Tidyverse package, googledrive to download the data. Then it prepares the data (which is currently a folder of text files) for SAS.

```
# install necessary libraries
library(googledrive)    # to download files from a Google Drive

# download Data.7z from Google Drive
# URL is fixed. You can download it using a browser as well.
URL <- "https://drive.google.com/file/d/1madd-
kNPY0ml8L1PFysCAfJvGwvCKs2P/edit"
# point it to dns_compressed as Data.7z
FILE_DEST <- "C:/Users/Bretho M. Danzy
III/Documents/STA475/dns_compressed/Data.7z"
drive_download(URL, FILE_DEST)

# extract Data.7z as Data
# location of 7-Zip executable following step can also be accomplished
w/ GUI
EXE_7ZIP <- "C:\\Program Files\\7-Zip\\7z.exe"
FILE <- "C:\\Users\\Bretho M. Danzy
III\\Documents\\STA475\\dns_compressed\\Data.7z"
DIR_DEST <- "C:\\Users\\Bretho M. Danzy
III\\Documents\\STA475\\dns_compressed"
INPUT <- paste0('\"', EXE_7ZIP, '\" x \"', FILE, '\" -o\"', DIR_DEST,
'\"')
system(INPUT)

# extract the dns files
FILES <- "C:\\Users\\Bretho M. Danzy
III\\Documents\\STA475\\dns_compressed\\Data\\*.xz"
# point it to dns_uncompressed
DIR_DEST <- "C:\\Users\\Bretho M. Danzy
III\\Documents\\STA475\\dns_uncompressed"
INPUT <- paste0('\"', EXE_7ZIP, '\" x \"', FILES, '\" -o\"', DIR_DEST,
'\"')
system(INPUT)
```

```
# change the file suffix from nothing to .txt
file_list <- list.files(DIR_DEST, full.names = T)
new_file_list <- paste(file_list, ".txt", sep = "")
file.rename(file_list, new_file_list)
```

Cleaning Data

Now we use this SAS code to load all text files in dns_uncompressedfolder and save the parsed text strings in
dns_sas7bdats folder.

```
/*  clean_data.sas - 12APR2020

    Directory: C:\Users\Bretho M. Danzy III\Documents\STA475
[Windows]
    Author: Bretho Danzy III
    Purpose: Parse dns text files and create a nicely
formatted .sas7bdat

    Input data files ------------------------------------------------
-------------
    *.txt           [Source: JAS Advisors]

    Output data files -----------------------------------------------
-------------
    clean_data.sas7bdat

*/
*************************************************************************
************;
*                                    CLEANING
*************************************************************************
************;
* setting up macro variables;
* STA library or file location to save datasets and formats;
libname STA "C:\Users\Bretho M. Danzy
III\Documents\STA475\dns_sas7bdat";
* dir directory for all *.txt files [Note: data should be uncompressed
to save time];
%let dir = C:\Users\Bretho M. Danzy
III\Documents\STA475\dns_uncompressed\*.txt;

* PROC STEP
    formats the destination IP so it takes up less space;
proc format library = STA;
    value $dest
    '(72.28.99.90)' = '1'
    '(72.28.99.91)' = '2'
    '(72.28.99.92)' = '3'
    '(72.28.99.93)' = '4'
    '(2607:fb58:c001:1ab5::90)' = '5'
```

```
    '(2607:fb58:c001:1ab5::91)' = '6'
    '(2607:fb58:c001:1ab5::92)' = '7'
    other = '0';
run;

* PROC STEP
    formats record_class so that is takes up less space
    info gleaned from
https://en.wikipedia.org/wiki/Domain_Name_System#Resource_records;
proc format library = STA;
    value $class
    /* internet */
    'IN' = '1'
    /* chaos */
    'CH' = '2'
    /* hesoid */
    'HS' = '3'
    other = '0';
run;

* load previously saved formats;
OPTIONS FMTSEARCH = (STA);

* DATA STEP
    cleans the data using SAS magic;
data STA.clean_data (compress = yes);
    * variables used for data processing;
    drop dummy_var preclient prerecord_class predestination;
    * input text files from designated directory;
    infile "&dir";
    * declare length of variables;
    length dummy_var $1 client $37 client_port 5. preclient $46
           prerecord_class $2 record_type $9 predestination $25;
    * load the variables;
    input date ANYDTDTE11. time TIME12. 4*dummy_var preclient
$ 3*dummy_var
          prerecord_class $ record_type $ dummy_var $ predestination
$;
    * parse and format few strings;
    record_class = input(put(prerecord_class, $CLASS.), 1.);
    client_port = scan(preclient,2,"*#");
    client = scan(preclient,1,"*#");
    destination = input(put(predestination, $DEST.), 1.);
run;

************************************************************************
************;
*                                DATASET CREATION
************************************************************************
************;
* PROC STEP
```

```
    manipulates clean_data;
proc sql;

* returns a vector of unique ip addresses;
create table STA.uniq_ip_vector as
    select distinct client as unique_ip
    from STA.clean_data;
quit;
```

Creating an IP Dictionary

Here we use a MaxMind database to create and IP dictionary. The MaxMind database is a log of IPs and their locations. The database is structured in such a way that it allows us to query the roughly 1.5 million unique IPs in our data in a matter of seconds.

```
# load libraries ----------------------------------------------------
----------
library(rgeolocate) # to interact with MaxMind database
library(sas7bdat)   # to read sas7bdats

# file/directory locations ------------------------------------------
----------
HOME_DIR <- getwd()
DIR_MM <- "C:\\Users\\Bretho M. Danzy
III\\Documents\\STA475\\dns_maxmind"
FILE <- "C:\\Users\\Bretho M. Danzy
III\\Documents\\STA475\\dns_sas7bdat\\uniq_ip_vector.sas7bdat"
CSV <- "C:\\Users\\Bretho M. Danzy
III\\Documents\\STA475\\dns_csv\\uniq_ip_dictionary.csv"
SAVE <- "C:\\Users\\Bretho M. Danzy
III\\Documents\\STA475\\dns_RData\\uniq_ip_dictionary.RData"

# setup database ----------------------------------------------------
----------
DATABASE <- system.file("extdata","GeoLite2-Country.mmdb", package =
"rgeolocate")

# prepping uniq_ips dictionary --------------------------------------
----------
# set working directory to dns_maxmind
setwd(DIR_MM)

# read in SAS file
uniq_ip_vector <- read.sas7bdat(FILE)

# coerce all values to into a vector of strings
ip_addresses <- as.character(uniq_ip_vector$unique_ip)

# we now have locations [Note: This is a 59% answer due to
missingness.]
location_ips <- maxmind(ip_addresses, DATABASE,
```

```
                          c("continent_name", "country_code",
"country_name"))

# get percentage of IPs we did not obtain locations for
sum(!is.na(location_ips[,1]))/dim(location_ips)[1]

# merge the ip_addresses with location_ips
uniq_ip_dictionary <- cbind(ip_addresses, location_ips)

# save as csv to read into SAS
write.csv(uniq_ip_dictionary,file = CSV)

# save uniq_ip_dictionary to RData file
save(uniq_ip_dictionary, file = SAVE)

# return to home directory
setwd(HOME_DIR)
```

Finalizing Datasets

SAS Portion

Now we join ip_dictionary with clean_data and creates finalized sas7bdats.

```
/*  merged_data.sas - 13APR2020

    Directory: C:\Users\Bretho M. Danzy III\Documents\STA475
[Windows]
    Author: Bretho Danzy III
    Purpose: Merge uniq_ip_dictionary with clean_data and create
finalized sas7bdats

    Input data files ----------------------------------------------
--------------------------------
    clean_data.sas7bdat
    uniq_ip_dictionary.csv

    Output data files ---------------------------------------------
--------------------------------
    uniq_ip_dictionary.sas7bdat

    merged_data.sas7bdat

    freq_day.sas7bdat
    freq_hour.sas7bdat

    uniq_day.sas7bdat
    uniq_hour.sas7bdat

    record_type_day.sas7bdat
    record_type_hour.sas7bdat
```

```
    country_day.sas7bdat
    country_hour.sas7bdat

*/
*************************************************************************
*************;
*                                 MERGING DATA
*************************************************************************
*************;
* setting up macro variables;
* STA library or file location to save datasets and formats;
libname STA "C:\Users\Bretho M. Danzy
III\Documents\STA475\dns_sas7bdat";
* location of uniq_ip_dictionary.csv;
%let uniq_csv "C:\Users\Bretho M. Danzy
III\Documents\STA475\dns_csv\uniq_ip_dictionary.csv";

* DATA STEP
    load the csv;
data sta.uniq_ip_dictionary;
    infile &uniq_csv dlm = ',' dsd firstobs = 2;
    length num $1 client $37 continent_name $13  country_code $2
country_name $33;
    input num client $ continent_name $ country_code $ country_name $;
run;

*************************************************************************
*************;
*                                 DATASET CREATION
*************************************************************************
*************;
* PROC STEP
    manipulates clean_data;
proc sql;

* compress data the data;
options compress = yes;

* merged_data.sas7bdat;
create table sta.merged_data as
    select *
    from sta.clean_data
    left join sta.uniq_ip_dictionary
    on clean_data.client = uniq_ip_dictionary.client;

* R cannot read in compressed SAS dataset so the rest must be
uncompressed;
options compress = no;

* frequency datasets;
```

```
create table sta.freq_day as
select date, count(date)as count
    from sta.merged_data
    group by date;

create table sta.freq_hour as
select date,
        ceil(time/3600) as hour,
        count(*) as count
    from sta.merged_data
    group by date, hour;

* unique ip addresses datasets;
create table sta.uniq_day as
select date,
    count(distinct client) as count
    from sta.merged_data
    group by date;

create table sta.uniq_hour as
select date,
        ceil(time/3600) as hour,
        count(distinct client) as count
    from sta.merged_data
    group by date, hour;

* record types datasets;
create table sta.record_type_day as
select date, record_type,
        count(*) as count
    from sta.merged_data
    group by date, record_type;

create table sta.record_type_hour as
select date, record_type,
        ceil(time/3600) as hour,
        count(*) as count
    from sta.merged_data
    group by date, hour, record_type;

* country datasets;
create table sta.country_day as
select date, country_name,
        count(*) as count
    from sta.merged_data
    group by date, country_name;

create table sta.country_hour as
select date, country_name,
        ceil(time/3600) as hour,
        count(*) as count
```

```
    from sta.merged_data
    group by date, hour, country_name;

quit;
```

This converts the sas7bdats into R-friendly tibbles.

```
# load libraries ---------------------------------------------------
----------
library(tidyverse)         # how R works
library(sas7bdat)          # how R reads .sas7bdat's
library(lubridate)         # how R tells time

# file locations ---------------------------------------------------
----------
sas_freq_day <-
  "C:/Users/Bretho M. Danzy
III/Documents/STA475/dns_sas7bdat/freq_day.sas7bdat"
sas_freq_hour <-
  "C:/Users/Bretho M. Danzy
III/Documents/STA475/dns_sas7bdat/freq_hour.sas7bdat"

sas_uniq_day <-
  "C:/Users/Bretho M. Danzy
III/Documents/STA475/dns_sas7bdat/uniq_day.sas7bdat"
sas_uniq_hour <-
  "C:/Users/Bretho M. Danzy
III/Documents/STA475/dns_sas7bdat/uniq_hour.sas7bdat"

sas_report_day <-
  "C:/Users/Bretho M. Danzy
III/Documents/STA475/dns_sas7bdat/record_type_day.sas7bdat"
sas_report_hour <-
  "C:/Users/Bretho M. Danzy
III/Documents/STA475/dns_sas7bdat/record_type_hour.sas7bdat"

sas_country_day <-
  "C:/Users/Bretho M. Danzy
III/Documents/STA475/dns_sas7bdat/country_day.sas7bdat"
sas_country_hour <-
  "C:/Users/Bretho M. Danzy
III/Documents/STA475/dns_sas7bdat/country_hour.sas7bdat"

# data processing function -----------------------------------------
----------
data_processing <- function(sas_file,
                            daily = T,
                            group = F) {
  # range of project
```

```
  range_day <-
    seq(ymd("2017-04-26"), ymd("2019-10-15"), by = 'day')
  range_hour <- seq(
    as_datetime("2017-04-26 02:00:00 UTC"),
    as_datetime("2019-10-15 24:00:00 UTC"),
    by = 'hour'
  )
  # load data
  data <- read.sas7bdat(sas_file)
  if (daily == T) {
    if (group == T) {
      names(data) <- c('Date', 'Group', 'Count')
    }
    if (group == F) {
      names(data) <- c('Date', 'Count')
    }
    tibble <- (as_tibble(data) %>%
                 mutate(Date = as_date(Date, origin = "1960-01-
01"))) %>%
      right_join(expand.grid(Date = range_day), by = "Date")
  }
  if (daily == F) {
    if (group == T) {
      names(data) <- c('date', 'Group', 'hour', 'Count')
      selection <- c("Date", "Group", "Count")
    }
    if (group == F) {
      names(data) <- c('date', 'hour', 'Count')
      selection <- c("Date", "Count")
    }
    tibble <- (
      as_tibble(data) %>%
        mutate(
          date1 = as.character(as_date(date, origin = "1960-01-01")),
          date2 = paste0(date1, " ", str_pad(hour, 2, pad = "0"),
":00:00 UTC"),
          Date = as_datetime(date2)
        )
    ) %>%
      right_join(expand.grid(Date = range_hour), by = "Date") %>%
      select(all_of(selection))
  }
  return(tibble)
}

# create tibbles ------------------------------------------------------
----------
freq_day <- data_processing(sas_freq_day)
freq_hour <- data_processing(sas_freq_hour, daily = F)

uniq_day <- data_processing(sas_uniq_day)
```

```
uniq_hour <- data_processing(sas_uniq_hour, daily = F)

report_day <- data_processing(sas_report_day, group = T)
report_hour <- data_processing(sas_report_hour, daily = F, group = T)

country_day <- data_processing(sas_country_day, group = T)
country_hour <- data_processing(sas_country_hour, daily = F, group =
T)
```

Code for map visuals

```
setwd("C:/Users/sur/Desktop/School Documents/Senior Year/STA 475/Nikitin Project")

library(tidyverse)
library(iptools)
library(readr)
library(rgeolocate)
library(assertthat)
library(purrr)
library(igraph)
library(ggraph)
library(ggmap)

load("C:/Users/sur/Desktop/School Documents/Senior Year/STA 475/Nikitin
Project/map_making_one_week.Rdata")

maptheme <- theme(panel.grid = element_blank()) +
  theme(axis.text = element_blank()) +
  theme(axis.ticks = element_blank()) +
  theme(axis.title = element_blank()) +
  theme(legend.position = "none") +
  theme(panel.grid = element_blank()) +
  theme(panel.background = element_rect(fill = "#FFFFFF")) +
  theme(plot.margin = unit(c(0, 0, 0.5, 0), 'cm')) +
  theme(plot.title = element_text(hjust = 0.5, size = 10, lineheight = 10))

mapcoords <- coord_fixed(xlim = c(-160, 180), ylim = c(-52, 80))

total <- country_day %>%
  group_by(Group) %>%
  mutate(total=sum(Count))

total_no_us <- country_day %>%
```

```
  group_by(Group) %>%
  mutate(total=sum(Count)) %>%
  filter(!Group=="United States")

total4 <- total2 %>%
  distinct(Group, total)

total <- total[seq(1,229),]

us <- total4 %>%
  filter(Group == "United States")

not_us <- total4 %>%
  filter(!Group == "USA")

us$Group <- "USA"

total2 <- rbind(not_us, us)

world_data <- map_data("world")

total4$region <- as.character(total4$Group)

total4 <- total4 %>%
  filter(!Group=="NA")

world_data3 <- left_join(x=world_data, y=total4, by= "region")

world_data4 <- world_data3 %>%
  distinct(lat, long, total, region, group, order)

#For no US

not_us$region <- as.character(not_us$Group)

world_data5 <- left_join(x=world_data, y=not_us, by= "region")

world_data6 <- world_data5 %>%
  distinct(lat, long, total, region, group, order)

world_data7 <- world_data6 %>%
  filter(!region == "USA")


#Fill map using frequency instead of line weights
#With US

ggplot() + geom_polygon(aes(x = long, y = lat, group = group,
                            fill = total),
```

```
                data = world_data3, color = "#515151",
                size = 0.15) + mapcoords + scale_fill_continuous(type = "viridis") + labs(title = "Worldwide
Frequency") + maptheme

save(world_data3, file="country_data_w_US.Rdata")

#No US
ggplot() + geom_polygon(aes(x = long, y = lat, group = group,
                    fill = total),
                data = world_data5, color = "#515151",
                size = 0.15) + maptheme + mapcoords + scale_fill_continuous(type = "viridis") +
ggtitle("Frequency Outside the U.S.") + theme(plot.title = element_text(hjust = 0.5))

save(world_data5, file="country_data_no_US.Rdata")

save(total4, file="aggregate_total_activity_country.Rdata")


total5 <- total4 %>%
  filter(!region == "USA")

total6 <- total4 %>%
  filter(region == "USA")

total6$total/sum(total5$total)

dns_types <- report_day %>%
  group_by(Group) %>%
  mutate(total=sum(Count)) %>%
  distinct(Group, total)

dns_types1 <- dns_types %>%
  filter(Group == "A")

dns_types2 <- dns_types %>%
  filter(!Group == "A")

dns_types1$total/sum(dns_types2$total)
```