Title:	Utilizing Weak Indicators to Detect Anomalous Behaviors in Networks
Author(s):	Egid, Adin Ezra
Intended for:	Report
Issued:	2018-02-06

Utilizing Weak Indicators to Detect Anomalous Behaviors in Networks

Adin Egid,
Graduate Research Assistant,
A-4 Advanced Research in Cyber Systems
Los Alamos National Laboratory

October 20, 2017

Abstract

We consider the use of a novel weak indicator alongside more commonly used weak indicators to help detect anomalous behavior in a large computer network. The data of the network which we are studying in this research paper concerns remote log-in information (Virtual Private Network, or VPN sessions) from the internal network of Los Alamos National Laboratory (LANL). The novel indicator we are utilizing is something which, while novel in its application to data science/cyber security research, is a concept borrowed from the business world. The Herfindahl-Hirschman Index (HHI) is a computationally trivial index which provides a useful heuristic for regulatory agencies to ascertain the relative competitiveness of a particular industry. Using this index as a lagging indicator in the monthly format we have studied could help to detect anomalous behavior by a particular or small set of users on the network.

Additionally, we study indicators related to the speed of movement of a user based on the physical location of their current and previous logins. This data can be ascertained from the IP addresses of the users, and is likely very similar to the fraud detection schemes regularly utilized by credit card networks to detect anomalous activity. In future work we would look to find a way to combine these indicators for use as an internal fraud detection system.

1 Introduction

1.1 Outline of the Problem

The problem entails using weak indicators to detect anomalous data from within the database of user remote logins, or VPN sessions, from among the lab's employees. A weak indicator is one which by definition, does not by itself immediately and with certainty identify anomalous or malicious behavior. Instead, a weak indicator is one which when combined with other weak indicators and/or a deeper manual inspection, may yield this result. An example of such an indicator may be when one's credit card is used unexpectedly at a gas station a few states away, and the credit card company, while certainly not being able to immediately detect a fraud, will likely call or text the owner of the card to inquire if this purchase was indeed legitimate.

In this paper we are looking for such indicators which, due to their deviation from some normal statistical pattern, present a raised flag that may warrant at least a cursory inspection into the underlying data.

When multiple weak indicators are triggered via a set of data, this may present a red flag which will require a deeper inspection.

1.2 Data

The data for this study originated in LANL's internal databases. After importing the data, we used the Python programming language, particularly the pandas data analysis toolkit, to analyze the data in the desired methods described in the paper. The data used provides a user identification number, a session start time and end time, an external IP address as well as whether the user is a US citizen or foreign national. The external IP addresses will be very critical to our study, as we can feed these into the Python geoip library to yield location information about the VPN sessions.

The IP address data comes from MaxMind, Inc. When called via a few simple commands,

```
from geolite2 import geolite2
reader=geolite2.reader()
reader.get('50.130.223.86')
```

we can input an IP address from the internal list into the reader function, and the geolite2 library returns us a nested dictionary of relevent location data, as seen below:

```
{'city': {'geoname_id': 5476825,
   'names': {'de': 'Los Alamos',
     'en': 'Los Alamos',
     'es': 'Los  lamos',
     'fr': 'Los Alamos',
     'ja': '              ',
     'pt-BR': 'Los Alamos',
     'ru': '      -       ',
     'zh-CN': '
        '}},
  'continent': {'code': 'NA',
   'geoname_id': 6255149,
   'names': {'de': 'Nordamerika',
     'en': 'North America',
     'es': 'Norteam rica',
     'fr': 'Am rique du Nord',
     'ja': '          ',
     'pt-BR': 'Am rica do Norte',
     'ru': '
                ',
     'zh-CN': '      '}},
  'country': {'geoname_id':
     6252001,
   'iso_code': 'US',
   'names': {'de': 'USA',
     'en': 'United States',
     'es': 'Estados Unidos',
     'fr': ' tats-Unis',
     'ja': '
        ',
     'pt-BR': 'Estados Unidos',
     'ru': '      ',
     'zh-CN': '      '}},
  'location': {'accuracy_radius':
      100,
   'latitude': 35.8366,
   'longitude': -106.3093,
   'metro_code': 790,
   'time_zone': 'America/Denver
      '},
  'postal': {'code': '87544'},
  'registered_country': {'
     geoname_id': 6252001,
   'iso_code': 'US',
   'names': {'de': 'USA',
```

With the use of a few for-loops and append functions, we can build lists of the most relevant pieces of data derived from the IP addresses, namely city, state, country, latitude, and longitude, and then append these lists back to the initial pandas DataFrame we are working with. It is the first three pieces of data, particularly the state and country for VPN sessions, that we will use for our novel indicator, the Herfindahl-Hirschman Index. The last two pieces of data, the latitude and the longitude, will be used to track the rate of changes of user locations between sessions in much the same way that a credit card company could look for anomolous behavior as described above.

Radius (in km)	Correctly Resolved	Incorrectly Resolved	Unresolved
10	54%	40%	5%
25	73%	22%	5%
50	81%	14%	5%
100	87%	8%	5%
250	91%	4%	5%

Figure 1: GeoLite2 City Accuracy

It is important to note that accuracy of these geoip services are not infallible, as the provider clearly states on their website[1]. The chart on the website shows that for the United States in particular, while an IP address can be correctly resolved to a city within a 250 km radius (i.e., when the stated city retrieved from the nested dictionary is within 250 km of the actual location) with 91% accuracy, this accuracy decreases to less than 54% for resolution within a 10 km radius. Please see Figure 1 above for more detailed information.

While the small percentage of incorrectly resolved/unresolved IP addresses for the 250 km radius are unlikely to have a significant affect on the HHI calculations, as will see later on they have the potential to cause significant differences and thus raise inaccurate flags for the speed-of-movement calculations.

1.3 The Herfindahl-Hirschman Index (HHI)

The purpose of the index in its industry application is to measure the concentration of firms by some metric of market share, most likely revenue. The *higher* the index value, the *less* competitive an industry is deemed to be. The index value is obtained by 1) multiplying the market share of each firm in the industry by 100, 2) squaring those numbers, and 3) summing the squared values:

$$HHI = \sum_{i=1}^{n} (100 \times s_i)^2$$

where s_i = the market share, as a percentage, of a firm in the industry.

To elucidate the calculation, let's walk through a real world example[2]. The desktop search engine market is primarily divided between four major players: Google ($\sim$ 79%), Bing($\sim$ 8%), Baidu($\sim$ 8%), and Yahoo($\sim$ 6%). The HHI calculation is as follows:

$$(79\% \times 100)^2 + (8\% \times 100)^2 + (8\% \times 100)^2 + (6\% \times 100)^2 = 6,241 + 64 + 64 + 36$$

which yields a result of **6,405**.

As we can see, the calculation is bounded by 10,000 as a maximum (100% market share for one monopolistic firm), and asymptotically approaches zero on the lower bound (an example of an industry that fits the market share criterion for perfect competition could have 1000 firms each having $\frac{1}{10}$ of a percent market share; this would yield an HHI of **1**). The conventional thought on the HHI is that a value above 2500 represents a relatively oligopolistic industry while a value below 1500 represents a relatively competitive industry. The example clearly demonstrates that Google's dominance of the search engine industry makes it a relatively uncompetitive industry.

We will discuss the novel application of this index as a weak indicator in section 2.

1.4 Speed of Movement

The speed of movement indicators, which will be discussed in greater depth in sections 4 and 5, have an underlying theme similar to those in use by credit card companies. Namely, we will aggregate all of a user's sessions over a given time period, and then using the geographic coordinates (latitude and longitude) for the user's previous and current VPN session we will find the distance traveled from the previous session's IP address to the current session's IP address (this calculation obviously hinges on the accuracy of the data provided by the GeoIP library, so as we can see from Figure 1, there will certainly be some errors in this calculation). Next, using the session start and session end times, we can find the speed at which the employee would have had to have changed location to have the two VPN sessions be legitimate from that user; a speed of travel faster than what a commercial aircraft could travel should certainly raise a flag.

2 Application of the Herfindahl-Hirschman Index

The application of the HHI to VPN sessions in our study is based upon treating the number of VPN sessions for a given user in a given time period as the market share numbers in the original industry application of the index. Specifically, we create a pandas DataFrame for a given month, where each row is a separate VPN session, and then find out what share of the total number of VPN sessions a particular user accounts for. We can calculate an HHI number for any given month to see the concentration of VPN sessions across users for that month (i.e are many different users each accounting for a small number of VPN sessions, or are one or a small number of users accounting for a significant portion of those sessions). Once we have HHI numbers for a sizable sample of different months, we can look at how the Herfindahl number changes across months as a possible indication of changes in user behavior.

It is important to note that given the 10,000+ employee size of the LANL, the 'market share' of a specific laboratory employee in VPN sessions is likely to be much smaller than what we would see as a leading market share in even a highly competitive industry. Thus, the HHI numbers given as guidelines in subsection 1.3 which define the competitiveness of an industry are not going to be appropriate for our study. Instead, we will look to create confidence intervals based on the mean and standard deviation of monthly HHI numbers, and will look for deviations outside of a given confidence interval.

There were two main methods of applying the HHI that we used in this study: 1) long duration sessions for each month, and 2) sessions of all durations for each month in the study. We defined long duration sessions as those of either 18,000 or 36,000 seconds (5 or 10 hours). The main advantages of focusing on long duration sessions is that 1) these sessions make up a smaller percentage of all VPN sessions as most employees will not intentionally stay logged in for such long periods of time consecutively unless they have a serious amount of work to do, thus the HHI numbers will reflect a higher concentration of users and likely be more significant, and 2) we will ignore many of the extremely short duration sessions, that while we can't confirm directly from the data, likely represent intermittent session breaks and reconnects due to loss of network connectivity, a user temporarily stepping away from their computer, etc.

However, looking at sessions of all lengths does provide the advantage of having many more sessions to look at which will likely, as confirmed by this study, lead to lower HHI numbers as these various length VPN sessions are spread out amongst more users. With lower HHI numbers for each month, a heavy concentration of usage by one user will now potentially be more likely to stand out as anomalous, and it will thus be easier to have a flag raised to suspicious activity on the network.

2.1 Code and Implementation

The first step is to take a count of the number of cumulative sessions by user in our

DataFrame and to normalize these so we can have a pandas series object with the user id as the index and the market shares as the series value:

```
user_market_shares=sessions_df.
    user.value_counts(normalize=
    True)
user_market_shares[:10]
```

The second line of code will output the top ten users by market share for of VPN sessions; in this case for a particular month in 2017 we looked at sessions of all length:

```
XXX759      0.022244
XXX500      0.015717
XXX927      0.012601
XXX965      0.010576
XXX861      0.010363
XXX843      0.010096
XXX520      0.009031
XXX099      0.008152
XXX939      0.008045
XXX039      0.007885
Name: user, dtype: float64
```

This series is sorted by descending market share. Next, we will feed this series into an HHI function which also outputs the cumulative HHI value after every new user's market share is fed in. Having the output shown will reiterate the fact that in a given data sample such as the month we are looking at (which had 743 unique users initiate over 37,000 VPN sessions), the largest few user market shares will have the largest impact on the HHI value, while the users with the least market share (a number of users had only one VPN session connection) will barely push the index at all. For ease of viewing, only the HHI value after the first 5 (5 largest market shares) and after the last 5 (all single VPN session users) will be shown, along with the final HHI of 41.89.

The HHI function:

```
def HHI(x):
    HHI=0
    for value in x:
        HHI+=(value*100)**2
        print(HHI)

    return(HHI)
```

The user market share Series being fed into the function and it's output:

```
HHI(user_market_shares)

4.94801139385
7.41838002119
9.00612234416
10.1246294656
11.1985124046
...........
41.8879707003
41.887977797
41.8879848937
41.8879919904
41.8879990872

41.88799908716873
```

2.2 Sample Monthly Data

Before moving on to the HHI results, I thought it would be helpful to look at the distribution of VPN sessions by length for the same given month that the preceding examples in Section 2 came from.

```
from datetime import timedelta
    as dt
seconds_plot=sessions_df['
    duration'].dt.total_seconds()
    .plot(kind='hist',bins=300,
    log=True,title='Month␣X␣VPN␣
    Session␣Durations')
seconds_plot.set_xlabel('
    Duration␣in␣Seconds')
```

The log format was necessary (see Figure 2 due to the fact that two distinct time frames seemed to have an inordinately high numbers of sessions bounded within a few

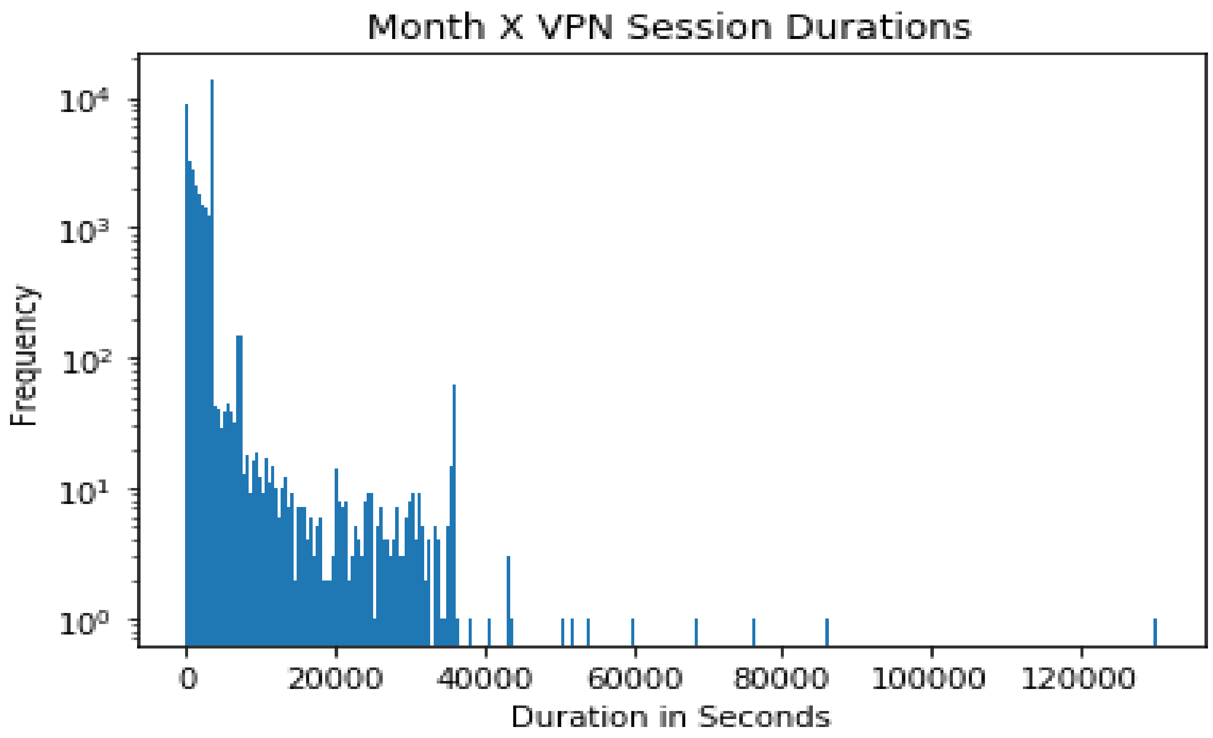


Figure 2: Distribution of VPN sessions by duration

seconds of duration of them. Specifically, the bin containing 3,600 seconds (1 hour) duration and the bin containing 36,000 seconds (10 hour) duration window had such spikes in the number of sessions (1-2 orders of magnitude higher than surrounding bins), that the these spike were still clearly visible even in a log format histogram. This leads us to surmise that there may be some sort of default automatic log-off of VPN sessions around both of these time frames .

```
sessions_df['duration_seconds']=
    sessions_df['duration'].dt.
    total_seconds()
sessions_df['duration_seconds'].
    describe()
```

```
count        37538.000000
mean          2331.160797
std           3142.384160
min              0.000000
25%            493.000000
50%           2142.000000
75%           3613.000000
max         130122.000000
Name: duration_seconds, dtype:
    float64
```

```
percentiles=sessions_df['
    duration_seconds'].quantile
    ([.05,.1,.15,.2,.25,.3,.35,.4,.45,.5,.55
```

```
percentiles
```

```
0.05        1.0
0.10       24.0
0.15      119.0
0.20      281.0
0.25      493.0
0.30      757.0
0.35     1015.0
0.40     1318.0
0.45     1707.0
0.50     2142.0
0.55     2676.0
```

```
0.60        3290.0
0.65        3610.0
0.70        3613.0
0.75        3613.0
0.80        3614.0
0.90        3615.0
0.95        3616.0
Name: duration_seconds, dtype:
    float64
```

The two previous snippets of code show 1) the output of the pandas describe method, which seems to indicate something interesting occurring around the aforementioned 3600 second mark, and 2) when we drill down into the percentiles, we see that the 65th to the 95th percentiles of session durations includes sessions that end shortly after the 1 hour mark, bolstering our previous suspicion that there is some sort of default termination of VPN sessions just after 1 hour.

3 Results of HHI and Hypothetical Example

3.1 Results of HHI

For the study we looked at a period of 15 consecutive months each with VPN session data similar to what we explored in Section 2. Running the entire Python script on the file for each month yielded the results displayed in Figure 3 for sessions of all durations:

	HHI for sessions of **ALL** lengths	Number of unique users	Number of sessions
Month 1	75.327	509	26957
Month 2	58.733	533	29187
Month 3	50.644	615	37101
Month 4	53.298	593	35964
Month 5	55.966	596	34924
Month 6	63.638	599	32443
Month 7	91.241	623	36606
Month 8	91.477	635	35634
Month 9	56.542	668	36663
Month 10	53.261	639	31828
Month 11	50.444	675	37625
Month 12	49.713	699	38711
Month 13	50.697	737	42624
Month 14	41.888	743	37538
Month 15	46.185	745	35253
Mean	**59.27**		
St. Dev	**15.16**		

Figure 3: HHI for all session lengths by month

We can create a 95% confidence interval on the monthly HHI in the following manner:

$$59.27-(1.96)*15.16 \Longleftrightarrow 59.27+(1.96)*15.16$$

$$= 29.56 \Longleftrightarrow 88.98$$

Months 7 and 8 fall just above this confidence interval, signifying a higher concentration of VPN sessions among a small group of users, potentially something worth looking in to. This is a prime example though of how our novel application of the HHI is clearly *not predictive*, it is instead what in economic terms would be considered a lagging indicator, as a full 9 months passed in this case from the time activity in Month 7 began to the point where our analysis would have detected it.

Figure 4 is the result of creating new DataFrames consisting of sessions of only a certain duration or longer. Those two specific durations, 18,000 and 36,000 seconds, were used as a filter for creating the new DataFrames in the following manner:

```
long_duration_sessions=
    sessions_df[sessions_df['
    duration_seconds']>18000]
```

After creating a pandas Series of normalized counts of VPN sessions of long duration, we feed this Series into the HHI function as described in Section 2 and are given HHI numbers for each month for long duration sessions. This is done for both durations, again for all 15 months, with the results summarized in Figure 4: As expected, the number of both sessions and unique users drops significantly as we switch from looking at all sessions to sessions of 18,000+ seconds, and then again as we switch from 18,000+ seconds to

	HHI for **18,000+** second duration VPN sessions	Number of unique users	Number of sessions	HHI for **36,000+** second duration VPN sessions	Number of unique users	Number of sessions
Month 1	300.22	75	373	687.4	35	107
Month 2	286.51	82	366	705.3	34	111
Month 3	342.97	71	320	727.98	25	95
Month 4	443.77	58	244	606.25	30	80
Month 5	403.16	61	288	521.54	35	84
Month 6	392.99	66	287	909.91	27	94
Month 7	353.23	61	303	600.32	33	86
Month 8	371.15	67	298	516.35	39	89
Month 9	459.93	62	297	764.85	35	114
Month 10	466.16	54	242	870.86	26	71
Month 11	410.09	54	322	867.2	29	92
Month 12	416.66	49	276	661.73	29	90
Month 13	353.59	62	310	1003.4	30	84
Month 14	377.88	72	290	532.54	39	78
Month 15	366.61	85	249	371.21	62	107
Mean	**382.99**			**689.79**		
St. Dev	**52.68**			**172.95**		

Figure 4: Long duration sessions HHI by month

36,000+ seconds. The drop in the number of unique users naturally corresponds to larger 'market shares' for each, which means significantly higher HHI numbers.

We again repeat the process of the construction of confidence intervals, first for 18,000+ second sessions:
$382.99 - (1.96) * 52.68 \iff 382.99 + (1.96) * 52.68$

$$= 279.74 \iff 486.24$$

It appears that none of the months have an HHI number outside of the 95% confidence interval. Next we well construct the same size confidence intervals for 36,000+ seconds:
$689.79 - (1.96) * 172.95 \iff 689.79 + (1.96) * 172.95$

$$= 350.81 \iff 1028.77$$

While Month 13 comes close with an HHI number of 1003, no month falls above the 95% confidence interval. However, we will via example see how when used in such a manner, the HHI indicator would be able to detect anomalous behavior definitively with only a 1 month lag.

3.2 Hypothetical Example of Clear Detection

In this example, let's look at hypothetical activity in a particular month, for example, Month 12. we will look at 36,000+ second sessions. With the current real data, we have 29 users accounting for 90 VPN sessions of 10 hours or longer, and a resultant HHI of 661.73, slightly below the average HHI for the 15 months we looked at.

Now, let's imagine that a certain malicious user, denoted by user ID 999999, has an intent of performing some malicious activity towards the lab's internal network, perhaps monitoring something, which requires this user to be constantly logged in. For simplicity, let's imagine the user logs in twice in a 24-hour period and leaves the VPN session running. After perhaps a default logout that may appear to occur just after 10 hours, the user logs back in. This is performed twice a day for 30 days, resulting in 60 VPN sessions of 36,000+ seconds for this user. The abnormally high concentration of long duration

VPN sessions by the user causes a severe spike in the HHI number for Month 12. The calculations of this hypothetical malicious user's affect on the HHI (60 sessions) are shown in step by step detail, being added on to the real user activity for 36,000+ second VPN sessions (90 sessions) for Month 12 in Figure 5:

With this new hypothetical HHI value of 1,838.22, we would have had a HHI mean for the 15 months of 768.22 and a standard deviation of HHI numbers of 342.74. The value of 1,838.22 would lie above the upper bound for even the 99% confidence interval,

$$768.22 + (2.58) * 342.74 = 1652.49$$

thus definitely raising a flag of suspicious activity.

4 Speed of Movement Indicator

4.1 Methodology

In order to track the speed of movement for each user between their VPN sessions, we will have to create a separate DataFrame for each user. This is done in two steps: 1) We first create a list of all the user id's who have had VPN sessions in a given month (sessions of ALL lengths),

```
user_list=
    user_session_count_list.index
```

and then 2) we use a for loop to loop over that list (it is actually a pandas 'Index' object) and create a new DataFrame with each row from the master DataFrame for that month that matches the user id in the list being copied to the specific DataFrame for that user.

```
for row in user_list:
      individual_user=sessions_df[
          sessions_df['user']==row
          ].copy()
```

Once we have the DataFrame created for each user, but still inside the for loop which iterates over the user list, we will create tuples of the current and previous latitude and longitude pairs for each VPN session, and then will apply the GeoPy vincenty function [3] to calculate the distance between the previous session's coordinates and the current session's coordinates.

```
for row in range(len(
    individual_user['latitude_
    change'])):
            if pd.isnull(
                individual_user.iloc[
                row,18])==True:
                  distance_output=0
            else:
                  distance_output=
                      vincenty(
                      individual_user.
                      iloc[row,20],
                      individual_user.
                      iloc[row,21]).
                      miles
            distance_list.append(
                distance_output)
```

(Columns 20 and 21 in the individual user DataFrames represent tuples for the previous and current session's coordinates). Once we have the distance between sessions, it is straight forward to find the time between the end of the previous session and the start of the current session given that we have both session start times and session durations.

It is now just a matter of finding speed in miles per hour as $\frac{distance}{timebetweensessions}$, setting a speed limit, and creating a 'high speed' DataFrame where the rows represent VPN sessions where the speed of movement since the previous session is over the speed limit:

```
speed_limit=600

speed_list=[]
      for index in range(len(
          distance_list)):
```

User ID	VPN session count	Market Share	Step 1: Multiply by 100	Step 2: Square	Cumulative Sum of HHI
999999	60	0.40	40.00	1,600.00	1,600.00
XXX520	11	0.07	7.33	53.78	1,653.78
XXX510	9	0.06	6.00	36.00	1,689.78
XXX209	9	0.06	6.00	36.00	1,725.78
XXX864	8	0.05	5.33	28.44	1,754.22
XXX960	7	0.05	4.67	21.78	1,776.00
XXX165	7	0.05	4.67	21.78	1,797.78
XXX677	5	0.03	3.33	11.11	1,808.89
XXX835	4	0.03	2.67	7.11	1,816.00
XXX218	3	0.02	2.00	4.00	1,820.00
XXX194	2	0.01	1.33	1.78	1,821.78
XXX880	2	0.01	1.33	1.78	1,823.56
XXX599	2	0.01	1.33	1.78	1,825.33
XXX451	2	0.01	1.33	1.78	1,827.11
XXX745	2	0.01	1.33	1.78	1,828.89
XXX360	2	0.01	1.33	1.78	1,830.67
XXX209	2	0.01	1.33	1.78	1,832.44
XXX245	1	0.01	0.67	0.44	1,832.89
XXX693	1	0.01	0.67	0.44	1,833.33
XXX213	1	0.01	0.67	0.44	1,833.78
XXX352	1	0.01	0.67	0.44	1,834.22
XXX163	1	0.01	0.67	0.44	1,834.67
XXX245	1	0.01	0.67	0.44	1,835.11
XXX726	1	0.01	0.67	0.44	1,835.56
XXX970	1	0.01	0.67	0.44	1,836.00
XXX735	1	0.01	0.67	0.44	1,836.44
XXX038	1	0.01	0.67	0.44	1,836.89
XXX969	1	0.01	0.67	0.44	1,837.33
XXX963	1	0.01	0.67	0.44	1,837.78
XXX418	1	0.01	0.67	0.44	1,838.22
Total Sessions:	**150**		Step 3: Sum	**1,838.22**	

Figure 5: Month 12 36,000+ second VPN sessions with overlay of hypothetical malicious user's VPN sessions and calculation of new HHI

```
        speed_output=3600*(
            distance_list[index]/
                time_change_list[
            index]) #need to
            make miles/sec into
            miles/hr
        speed_list.append(
            speed_output)

speed=np.asarray(speed_list)
    individual_user['speed']=
        speed

user_high_speed_df=
    individual_user[
    individual_user['speed']>
    speed_limit]
```

We then create a master high speed DataFrame by concatenating all of the individual user high speed DataFrames:

```
df_high_speed_list=[]
df_high_speed_list.append(
    user_high_speed_df)
master_high_speed_df=pd.concat(
    df_high_speed_list)
```

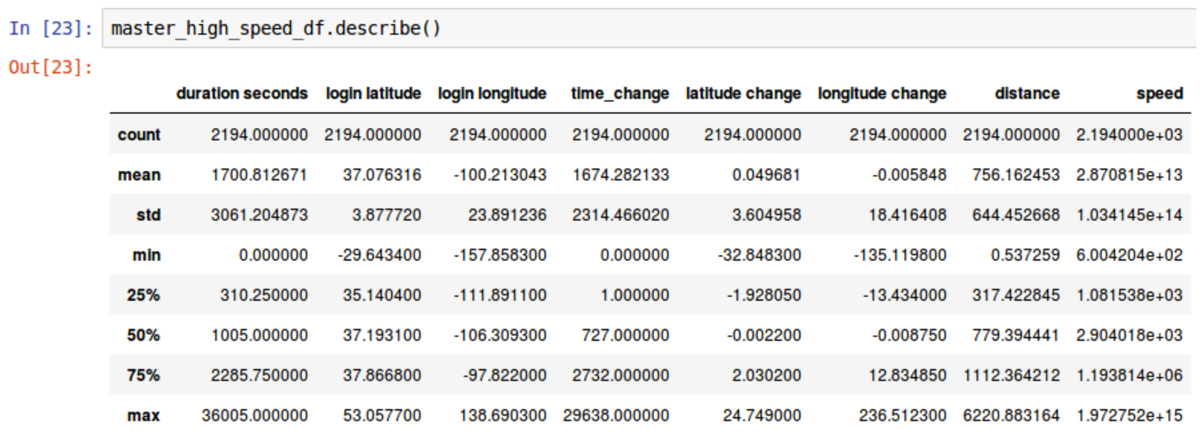
4.2 Results of Concatenating High Speed DataFrames

To encompass inter-session activity that spanned the end of one month and beginning of the next month, we decided to look at the first 9 months of 2017. Out of a total of 383,142 VPN sessions of all lengths, we found that 2194, or just over half of one percent, violated the 600 mph speed limit that modern commercial air travel would seem to logically set. Furthermore, there are a small number of absurdly high speeds that would violate any sort of common sense, and may at first glance indicate some sort of compromised credentials. Figure 6 shows a max speed that fits this descriptions, as well as mean speed orders of magnitude above the median speed, indicating the extreme right skew of the distributions of inter-session speeds in the master high speed DataFrame.

Yet when we look back at Figure 1, we are reminded that approximately 9% of all IP addresses in the US are not resolved correctly to within 250km when using the library behind the GeoLite2 database (thereare two other related databases, GeoIP2 and GeoIP2 Precision, but with both a still significant portion of IP addresses are not able to be properly resolved). This suspicion of the IP address not accurately resolving to location coordinates was corroborated by our manual input of the latitude and longitude into third party tools on the internet which seem to map latitude and longitude coordinates quite accurately. In one instance, a number of users were repeatedly mapped to a spot in the middle of a lake of a non-neighboring state [4] [5] just moments after being connected to a network within the state of New Mexico. So in summary, while the location pinpointing of a VPN session is promising and may serve to warrant manual inspection in some cases, it still lacks some of the accuracy that a credit card network's point of sale location based fraud system may currently have. One worthy note from MaxMind, Inc. the provider of these databases, is that IP geolocation is more accurate from broadband than from cellular networks, however that likely would not apply in most of the cases which we studied.

5 Conclusion and Future Work

Following off of the previous section, an area of improvement of this study is the accuracy of Geolocation IP services. This is something which we would naturally expect to increase over time, and is largely in the hands of those specific service providers. Other types of geolocation include the use of MAC addresses, but if the VPN session is from a portable laptop (very likely in our case), this proba-



```
In [23]: master_high_speed_df.describe()
Out[23]:
```

	duration seconds	login latitude	login longitude	time_change	latitude change	longitude change	distance	speed
count	2194.000000	2194.000000	2194.000000	2194.000000	2194.000000	2194.000000	2194.000000	2.194000e+03
mean	1700.812671	37.076316	-100.213043	1674.282133	0.049681	-0.005848	756.162453	2.870815e+13
std	3061.204873	3.877720	23.891236	2314.466020	3.604958	18.416408	644.452668	1.034145e+14
min	0.000000	-29.643400	-157.858300	0.000000	-32.848300	-135.119800	0.537259	6.004204e+02
25%	310.250000	35.140400	-111.891100	1.000000	-1.928050	-13.434000	317.422845	1.081538e+03
50%	1005.000000	37.193100	-106.309300	727.000000	-0.002200	-0.008750	779.394441	2.904018e+03
75%	2285.750000	37.866800	-97.822000	2732.000000	2.030200	12.834850	1112.364212	1.193814e+06
max	36005.000000	53.057700	138.690300	29638.000000	24.749000	236.512300	6220.883164	1.972752e+15

Figure 6: Descriptive statistics for the master high speed DataFrame

bly won't help this particular study much but may be broadly applicable if someone's VPN session originates from another machine.

As far as our novel (application of) indicator, the Herfindahl-Hirschman Index, we have considered using both different time periods and different metrics. Performing the HHI calculation on shorter time periods such as a weekly basis may allow us to detect anomalies with shorter lag times. Further, a weekly HHI calculation may then adjust for certain users' travel patterns better than a monthly indicator; more consistent patterns may emerge when looking at say, a Sunday to Saturday period than a period, than a period that starts on an arbitrary day of the week such as our monthly study. We can also apply the HHI to different metrics. We considered login states, login country, and potentially even HHI numbers for different hour segments of the day as considerations of possible further study.

Finally, we think the idea of combining these two indicators may potentially yield some promising results. Some manual inspection would be required to differentiate between high speed session changes which result from incorrect IP address resolution and those which are legitimate, but then we could look to combine user movements over a more modest speed limit with HHI numbers that fall a certain number of standard deviations above the mean over a certain time period.

Acknowledgements

I would like to thank my mentor at Los Alamos, Juston Moore, for his patience while I got up to speed with pandas and for giving me constant technical help and guidance, particularly with regular expressions. Additionally, I would like to thank another scientist in our group, Melissa Turcotte, for teaching me that a proper scientist cannot write reports or perform research with Microsoft Office products, and for thus answering all of my questions about Python and LaTeX in detail and with great patience.

References

[1] MaxMind, Inc., GeoIP2 City Accuracy, URL='maxmind.com/en/geoip2-city-database-accuracy'

[2] netmarketshare.com,
Search Engine Desktop Share,
URL='netmarketshare.com'

[3] python^{tm}
geopy 1.11.0
URL='pypi.python.org/pypi/geopy'

[4] Wikipedia,
Geolocation software,
URL='en.wikipedia.org/wiki/Geolocation_software'

[5] Wikipedia,
MaxMind,
URL='en.wikipedia.org/wiki/MaxMind'