AWS-Market Share Analysis

June 9, 2019

1 Collection of Simple Random Sample from Population of Websites

[13]:
```
from random import randint
import pandas as pd
import socket

# List of top one million sites according to Alexa Analytics/Website Ranking
# https://s3.amazonaws.com/alexa-static/top-1m.csv.zip
top_sites = pd.read_csv('top-1m.csv', header=None)[1]

# n is sample size
n=100

# Dictionary used for stored sample data
sample = {
    'Website Domain' : [],
    'IPv4 Address' : []
}

def create_sample(n):
    i = 0
    while i < n:
        i += 1
        # Get random number between 0 and 999,999
        random_index = randint(0, len(top_sites) - 1)

        # If the site has not already been selected, add it to our data set
        if not top_sites[random_index] in sample['Website Domain']:
            try:
                # print("\033[0mGetting IPv4 Address for %s..." %␣
↪top_sites[random_index])
                ipv4 = socket.gethostbyname(top_sites[random_index])
            # If we can't resolve the IP from the host name, replace it with a␣
↪different host name
            except:
                # print("\033[1mFailed. Selecting new site for sample.")
                i -= 1
                continue
            sample['Website Domain'].append(top_sites[random_index])
            sample['IPv4 Address'].append(ipv4)

create_sample(n)

# Save sample to a CSV file
dataset = pd.DataFrame.from_dict(sample)
dataset.to_csv('website_sample.csv')

dataset
```

[13]:

```
                  Website Domain     IPv4 Address
0                     vexere.com   125.212.247.90
1                        gdx.net     97.82.82.244
2          filmygore.blogspot.com    172.217.6.161
3                         occ.bg     104.31.85.57
4                   raboo-co.ir    51.77.184.210
5             myfirstfarmers.com       23.185.0.3
6                petstock.com.au    165.160.15.20
7                      laheia.gr   185.79.189.178
8                    syncplay.pl    91.121.132.98
9                   krskstate.ru    185.211.0.210
10              homescherish.com    74.63.240.162
11       americanfoodbloggers.com     72.14.191.82
12                  botanic06.com     62.210.16.62
13                   momofuku.com    35.196.138.31
14                  kashanedu.ir   185.153.208.53
15              jakesembassy.com   67.220.188.162
16            hoodsforheroes.org   162.144.223.31
17                    sibroid.com     89.42.211.83
18                     dquail.com    185.143.232.5
19                      sdmts.com    12.236.147.10
20                   mitrphol.com   203.146.102.21
21                   asforme.org   162.241.225.30
22               mirzhivotnye.ru   185.253.33.126
23         agorapublicaffairs.com    66.198.240.13
24                  kenyabuzz.com    34.249.215.96
25                   fortytwo.sg    104.20.154.42
26            changonerias.com.mx  74.208.236.225
27                  sehatsegar.net   104.27.168.176
28        srpskijezickiatelje.com   107.20.139.176
29                   hctorpedo.ru     79.174.76.38
..                            ...              ...
70                    yingmoo.com    103.41.53.194
71               pacificard.com.ec     157.100.71.4
```

```
72                      pandzee.com     104.18.49.60
73               ipm-mathemagic.com  192.124.249.110
74            leahberman.tumblr.com       66.6.33.21
75  transitofloridablanca.gov.co   204.93.177.191
76                   zombicity.info   193.109.247.65
77                   euthemians.com    104.25.190.34
78                 hindi-kavita.com   139.162.47.194
79                     myeslsca.com     213.32.31.32
80              techmediasquare.com     94.23.201.37
81                       greece.com      104.28.8.97
82                 lafirstdates.com   38.130.197.118
83                   mc-complex.com     104.24.126.2
84            donmooreswartales.com      192.0.78.25
85              leveragetech.com.au     103.211.6.10
86                   soundoasis.com  192.124.249.109
87                        b-p.sale    193.233.63.11
88                    warsteiner.de   213.160.71.154
89                 mp2carnot.free.fr    212.27.63.111
90                   bizservices.ir   79.175.172.150
91      elaguijondelescorpion.com    198.54.116.10
92                       noel.gv.at   194.232.42.155
93               mychameleon.com.au     43.230.64.65
94                 dappercodes.com   104.24.117.154
95                     koinwnia.com   66.147.240.199
96                    lockvista.com  184.168.131.241
97                       ggpc.co.nz   119.47.116.121
98      from-template.appspot.com    172.217.9.180
99                 pornavigator.com      37.1.205.41

[100 rows x 2 columns]
```

2 Determining Proportion of Websites Running AWS

[14]:

```python
import json, requests, ipaddress

# List of IP Ranges (IPv4 and IPv6) owned by Amazon and used for AWS
# https://ip-ranges.amazonaws.com/ip-ranges.json
aws_ip_ranges = json.loads(requests.get('https://ip-ranges.amazonaws.com/
 ↪ip-ranges.json').text)

# Determine if given IP address (ip_input) shows uo in AWS IPv4 Range
def check_aws(ip_input):
    # Compare given IP to all AWS IP addresses within AWS IPv4 Subnetwork
    for i in range(len(aws_ip_ranges['prefixes'])):
        # Parse IPv4 address for comparison
        site_ip = ipaddress.ip_address(ip_input)
        # Parse AWS IPv4 Subnet
        aws_subnet = ipaddress.
 ↪ip_network(aws_ip_ranges['prefixes'][i]['ip_prefix'])

        # If IP is within the AWS IPv4 Range, the website is run on AWS
        if site_ip in aws_subnet:
            return True
    # If the website is not within the range, the
    # website operates independnetly of AWS
    return False

# List of websites using AWS
websites_using_aws = []

def get_aws_domains():

    # Check every IP within our sample against AWS IPv4 Range
    for i in range(len(dataset)):
        if check_aws(dataset['IPv4 Address'][i]):
            websites_using_aws.append(dataset['Website Domain'][i])

get_aws_domains()

# Save dataset of AWS websites to a CSV file
aws_df = pd.DataFrame({'AWS Websites':websites_using_aws})
aws_df.to_csv('websites_using_aws.csv')

aws_df
```

[14]:
```
               AWS Websites
0            kenyabuzz.com
1  srpskijezickiatelje.com
2   atlasophthalmology.net
3        basketballking.jp
4           2appstudio.com
```

3 1-Proportion Z-Test for Proportion of AWS to non-AWS Websites

[15]:
```
import math
import scipy.stats as st

# Creating initial values from datatset/claim
claimed_marketshare = 0.31

p = claimed_marketshare
```

```
q = 1-claimed_marketshare

# Success/Failure Condition, exception raised if np or nq is less than 10
assert n*p >= 10, True
assert n*q >= 10, True

# Calculate Z-Score & P-Value
z = ((len(websites_using_aws)/n) - p)/math.sqrt((p*q)/n)
p_value = st.norm.cdf(z)

print('P: %f\tQ: %f\nNP: %f\tNQ: %f\n\nP-Hat: %f\n\nZ-Score: %f\nP-Value: %f\n'
      % (p, q, n*p, n*q, (len(websites_using_aws)/n), z, p_value))

# Hypothesis Testing
confidence_level = 0.95

if p_value <= (1-confidence_level): print('\033[1mReject H0')
else: print('\033[1mFail to reject H0')
```

```
P: 0.310000     Q: 0.690000
NP: 31.000000   NQ: 69.000000

P-Hat: 0.050000

Z-Score: -5.621704
P-Value: 0.000000
```