In [10]:

```python
import math
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker
from matplotlib.ticker import PercentFormatter
from collections import Counter

import numpy as np


def bernford(digit):
    temp = 1 + (1 / digit)
    return math.log(temp, 10)
```

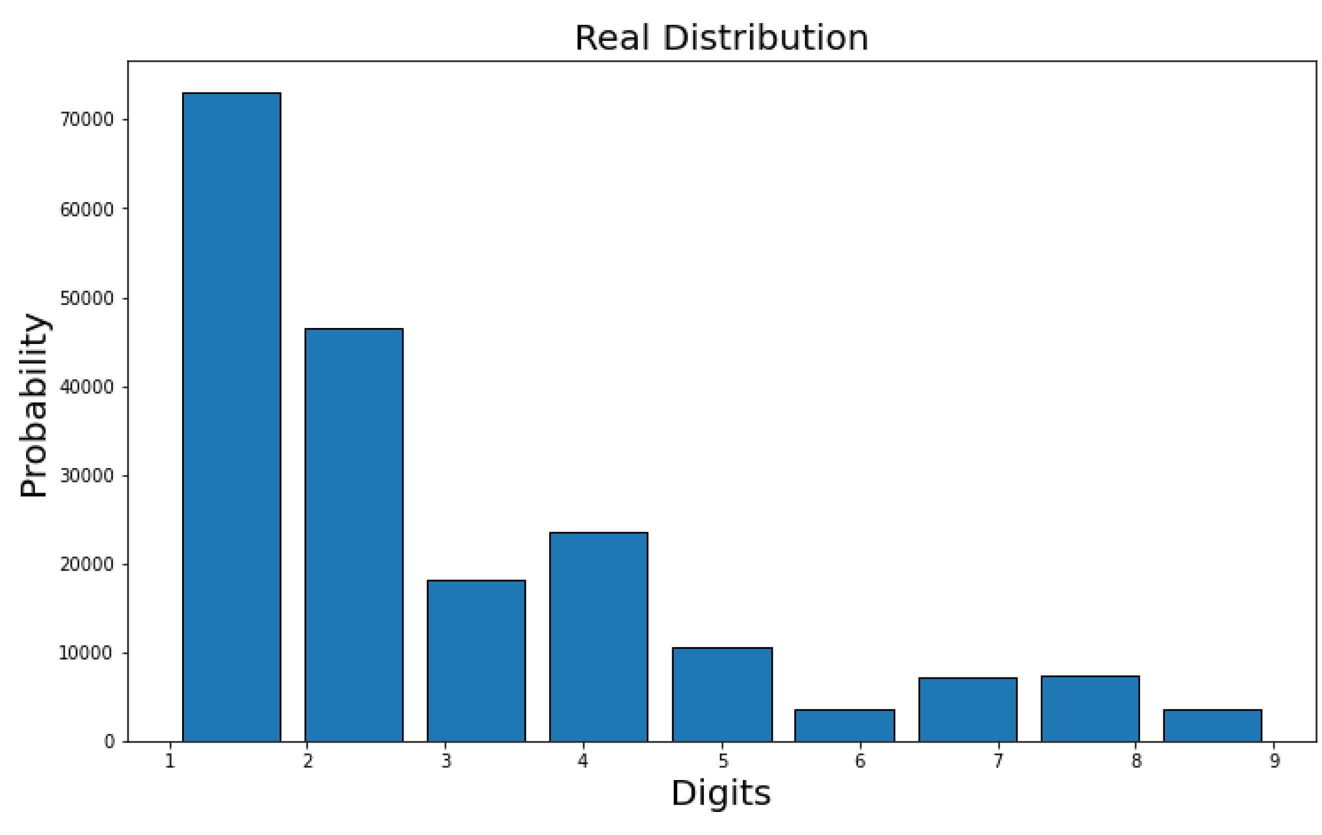
In [11]:

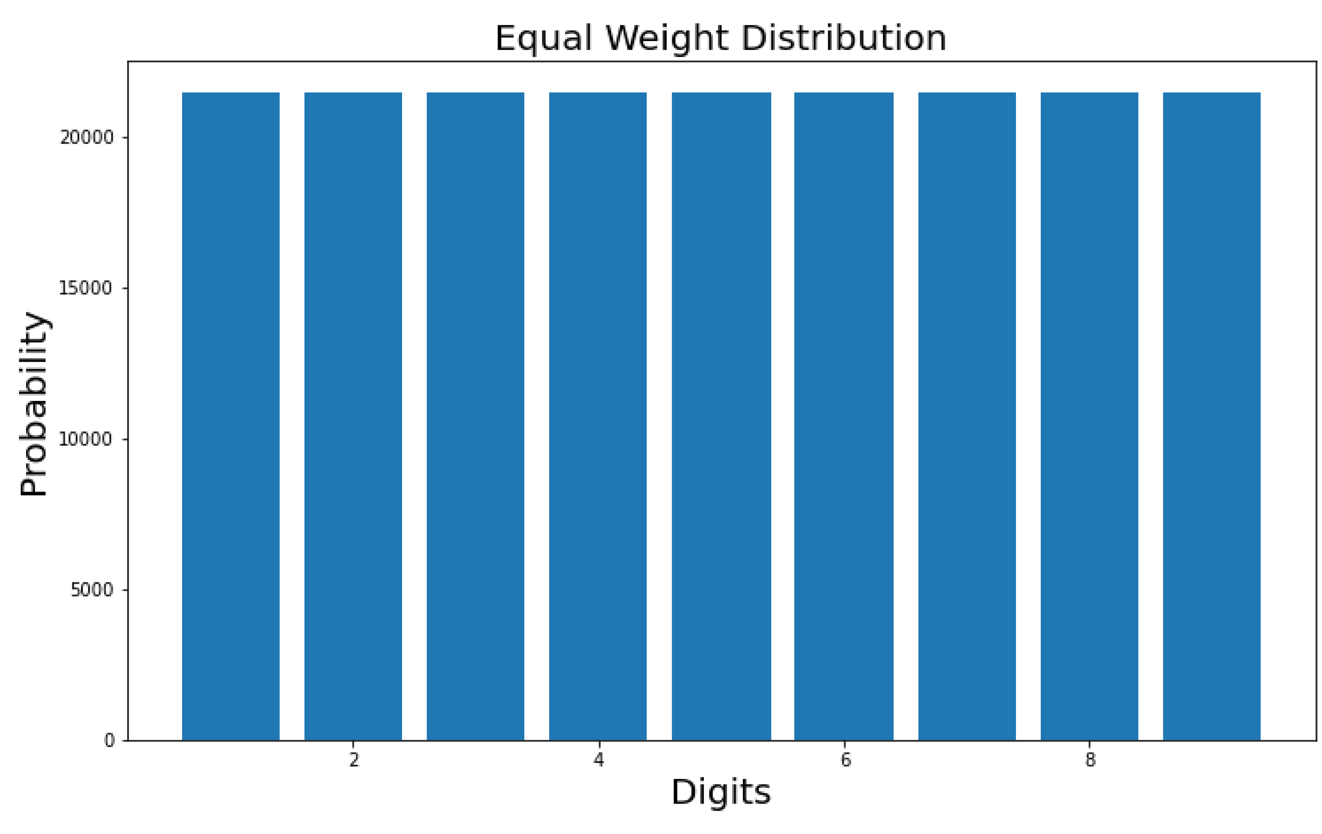
```python
# Extract the data and extract the first digits
df = pd.read_csv('online_retail.csv')
df = df[['UnitPrice', 'Country']]
temp = []
for p in df.UnitPrice:
    p = str(p)
    temp.append(int(p[0]))
df['PriceDigit'] = temp

# Drop data with PriceDigit == 0
df.drop(df[df.PriceDigit == 0].index, inplace=True)
```

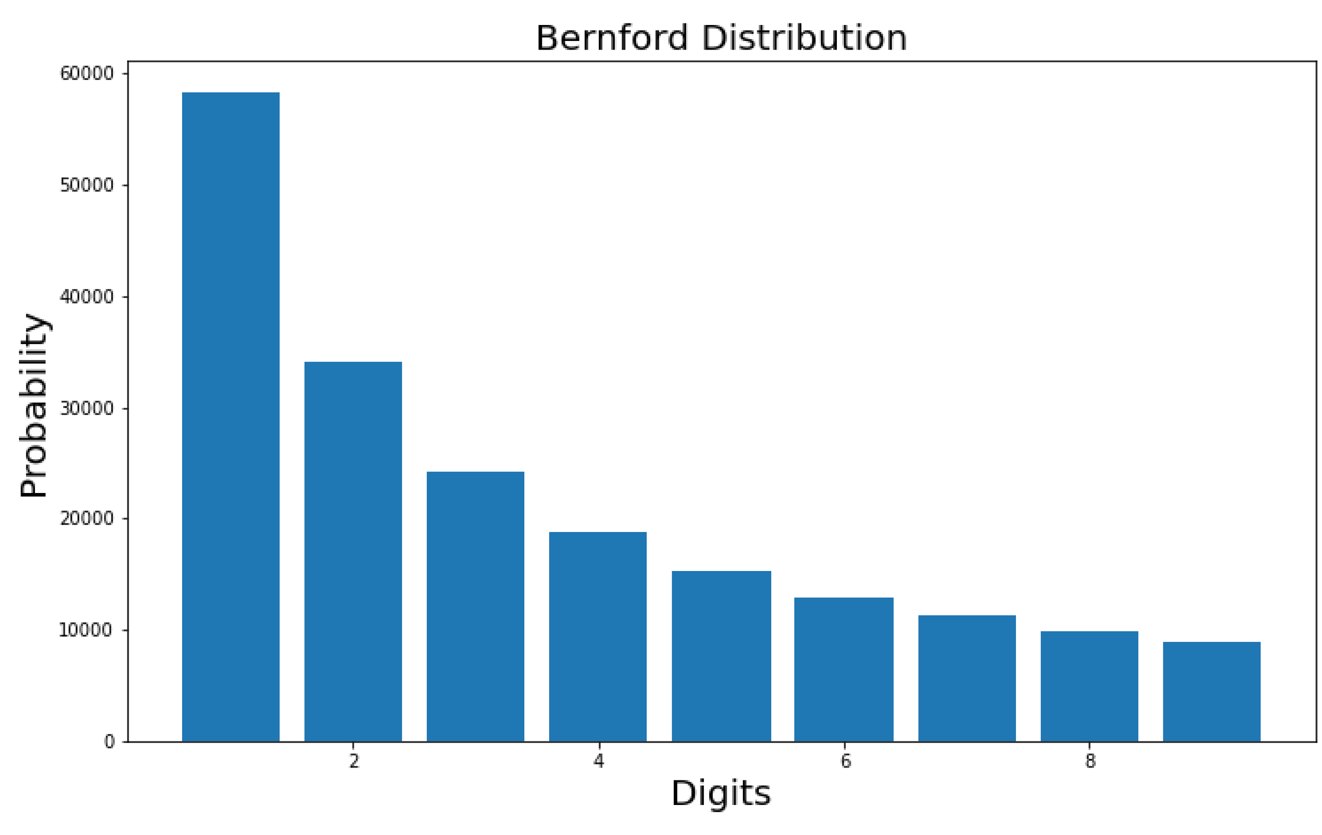
In [12]:

```python
# Create the figure size
f = plt.figure()
f.set_size_inches(12,24)

# Subplot of real distrubution
data = df.PriceDigit
reaHis = f.add_subplot(3,1,1)

n, bins, patches = reaHis.hist(data, bins = 9, histtype='bar', ec='black', rwidth=0.8)

reaHis.set_title('Real Distribution', fontsize=20)
reaHis.set_xlabel('Digits', fontsize=20)
reaHis.set_ylabel('Probability', fontsize=20)

# Subplot of Equal distrubution
x = [ i+1 for i in range(9)]
y = [ 1/9 * len(data) for i in x]
ewHis = f.add_subplot(3,1,2)

ewHis.bar(x, y, width = 0.8)

ewHis.set_title('Equal Weight Distribution', fontsize=20)
ewHis.set_xlabel('Digits', fontsize=20)
ewHis.set_ylabel('Probability', fontsize=20)

# Subplot of Bernford distrubution
x = [ i+1 for i in range(9)]
y = [bernford(i) * len(data) for i in x]
bfHis = f.add_subplot(3,1,3)

bfHis.bar(x, y, width = 0.8)

bfHis.set_title('Bernford Distribution', fontsize=20)
bfHis.set_xlabel('Digits', fontsize=20)
bfHis.set_ylabel('Probability', fontsize=20)
```

Out[12]: `Text(0, 0.5, 'Probability')`

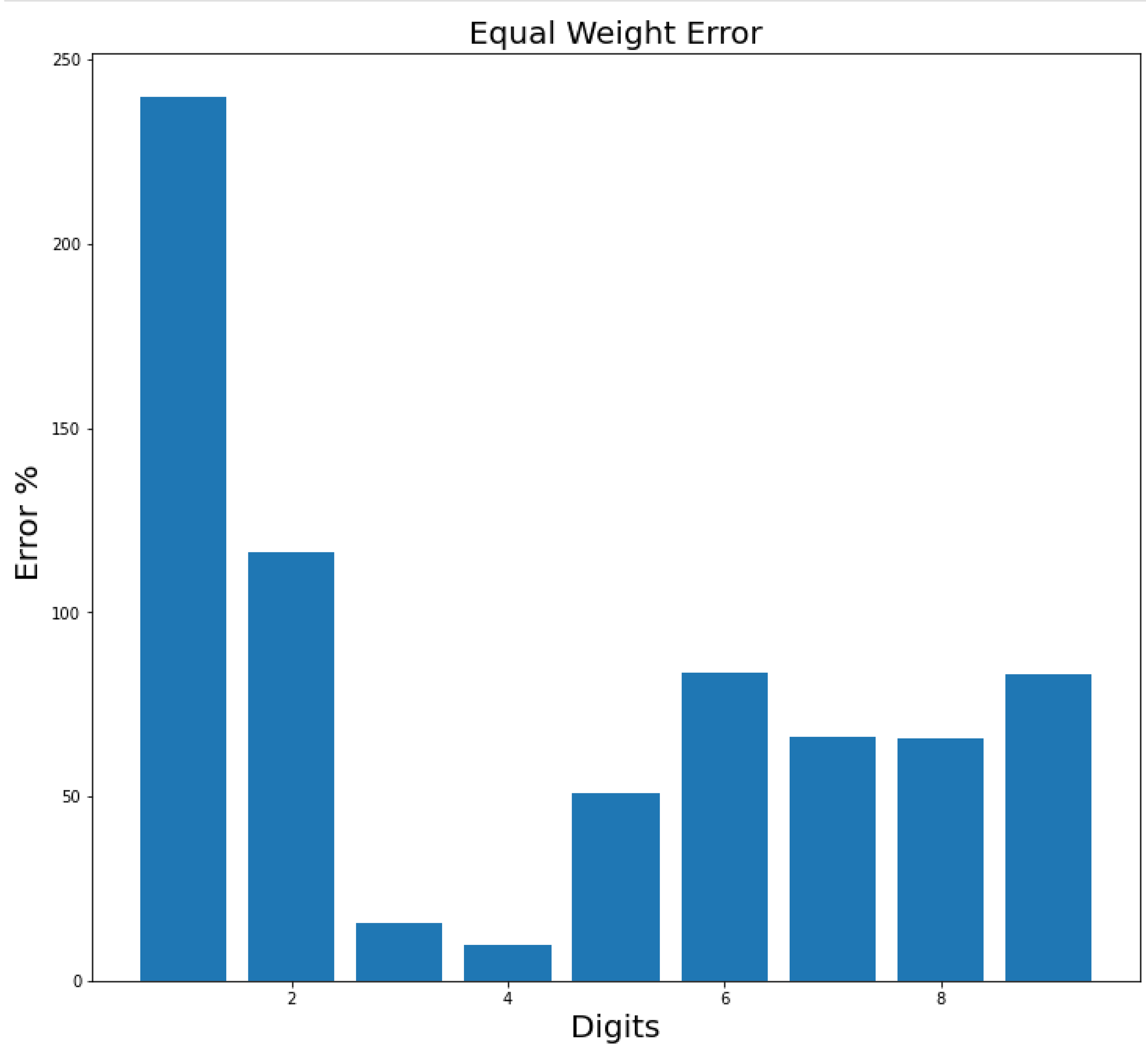


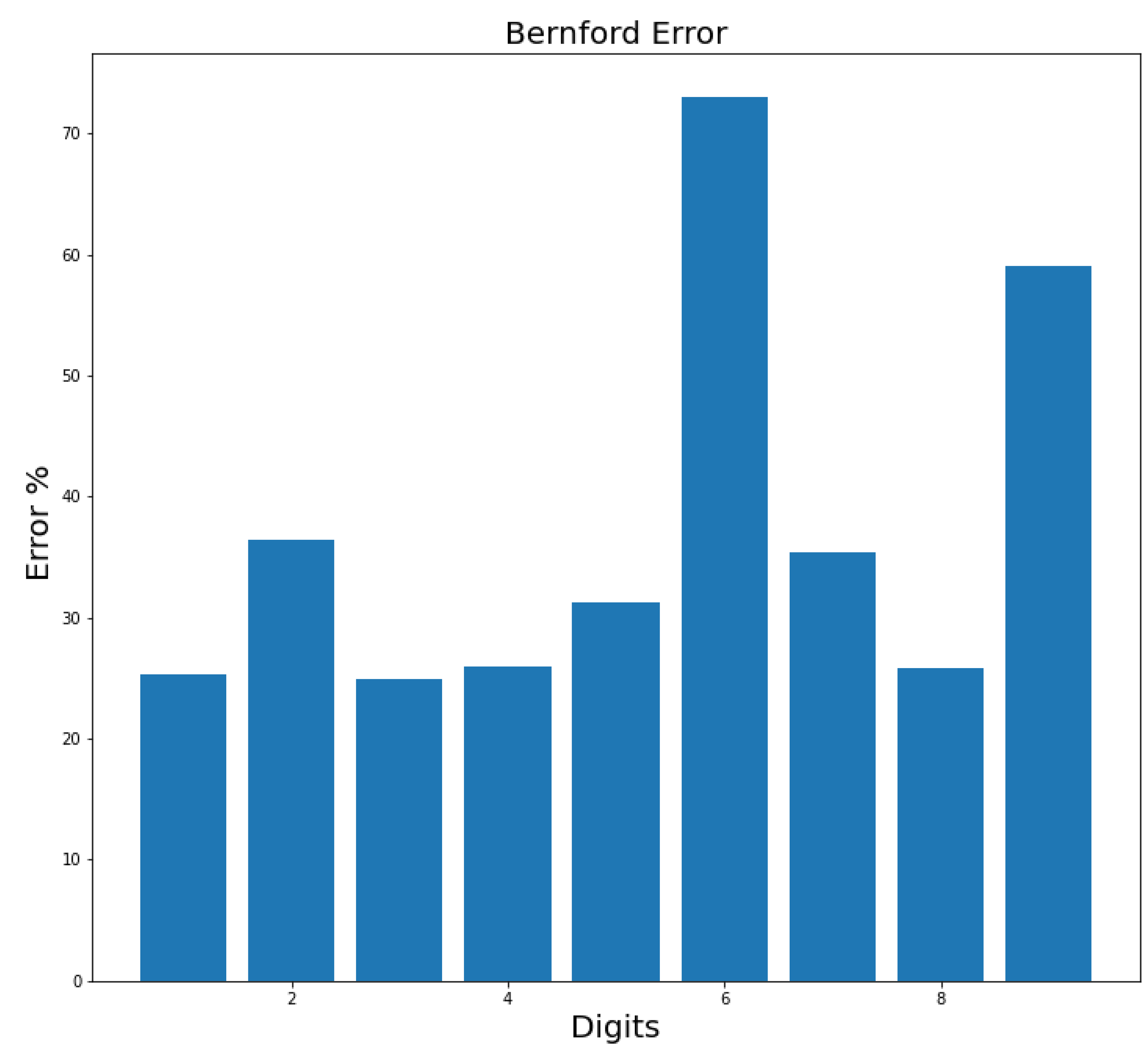




In [13]:

```python
# Relative error
def relativeErr(true, predict):
    ans = abs((true-predict) * 100) / true
    return ans

if __name__ == "__main__":
    reD = np.array(n)
    ewD = np.array([ 1/9 * len(df.PriceDigit) for i in range(9)])
    bfD = np.array([bernford(i+1) * len(df.PriceDigit) for i in range(9)])

    rErr_ewD = relativeErr(ewD, reD)
    rErr_bfD = relativeErr(bfD, reD)
    x = [i+1 for i in range(9)]


    # Plotting
    f = plt.figure()
    f.set_size_inches(12,24)

    ## Plot equal weight
    a = f.add_subplot(2,1,1)
    a.bar(x, rErr_ewD, width = 0.8)

    a.set_title('Equal Weight Error', fontsize=20)
    a.set_xlabel('Digits', fontsize=20)
    a.set_ylabel('Error % ', fontsize=20)


    ## plot bernford
    a = f.add_subplot(2,1,2)
    a.bar(x, rErr_bfD, width = 0.8)

    a.set_title('Bernford Error', fontsize=20)
    a.set_xlabel('Digits', fontsize=20)
    a.set_ylabel('Error % ', fontsize=20)
```





In [14]:

```python
# RMSE
def rmse(true, predict):
    temp = (true-predict)
    n = len(true)
    ans = sum(temp*temp/n)**0.5
    return ans

if __name__ == "__main__":
    rmse_ewD = rmse(ewD, reD)
    rmse_bfD = rmse(bfD, reD)

    print("remse for model 1 to real distribution is {:.1f}".format(rmse_ewD))
    print("remse for model 1 to benford distribution is {:.1f}".format(rmse_bfD))
```

```
remse for model 1 to real distribution is 22256.6
remse for model 1 to benford distribution is 8120.7
```

Take 3 countries of my choice

In [15]:

```python
# Check how many countries we have
country = set(df.Country)
country
```

Out[15]:

```
{'Australia',
 'Austria',
 'Bahrain',
 'Belgium',
 'Brazil',
 'Canada',
 'Channel Islands',
 'Cyprus',
 'Czech Republic',
 'Denmark',
 'EIRE',
 'European Community',
 'Finland',
 'France',
 'Germany',
 'Greece',
 'Hong Kong',
 'Iceland',
 'Israel',
 'Italy',
 'Japan',
 'Lebanon',
 'Lithuania',
 'Malta',
 'Netherlands',
 'Norway',
 'Poland',
 'Portugal',
 'Saudi Arabia',
 'Singapore',
 'Spain',
 'Sweden',
 'Switzerland',
 'USA',
 'Unit',
 'United Arab Emirates',
 'United Kingdom',
 'Unspecified'}
```

In [16]:

```python
# Choose Isreal, Finland, Japan
dfIsr = df[df.Country == 'Israel']
dfFin = df[df.Country == 'Finland']
dfJap = df[df.Country == 'Japan']
```

In [17]:

```python
def countFrequency(input):
    countList = [0 for i in range(9)]
    for d in input:
        countList[d-1] = countList[d-1] + 1
    return countList

if __name__ == "__main__":
    # Compute F P pi
    ## Compute F, real distribution
    fIsr = np.array(countFrequency(dfIsr.PriceDigit))
    fFin = np.array(countFrequency(dfFin.PriceDigit))
    fJap = np.array(countFrequency(dfJap.PriceDigit))


    ## Compute P, equal weight distribution
    pIsr = np.array([1/9 * len(dfIsr) for i in range(9)])
    pFin = np.array([1/9 * len(dfFin) for i in range(9)])
    pJap = np.array([1/9 * len(dfJap) for i in range(9)])
    ## Compute pi, Bernford distribution
    piIsr = np.array([bernford(i+1) * len(dfIsr) for i in range(9)])
    piFin = np.array([bernford(i+1) * len(dfFin) for i in range(9)])
    piJap = np.array([bernford(i+1) * len(dfJap) for i in range(9)])
```

In [18]:

```python
# Using RMSE as a 'distance' metric
print("RMSE of fIsr with pIrs is {:.1f}".format(rmse(pIsr, fIsr)))
print("RMSE of fFin with pFin is {:.1f}".format(rmse(pFin, fFin)))
print("RMSE of fJpa with pJap is {:.1f}".format(rmse(pJap, fJap)))

# Closest to equal weight
print('Isreal is the closed to equal weight P')
```

```
RMSE of fIsr with pIrs is 3.6
RMSE of fFin with pFin is 22.7
RMSE of fJpa with pJap is 25.4
Isreal is the closed to equal weight P
```

In [19]:

```python
print(len(dfIsr))
print(len(dfFin))
print(len(dfJap))
```

```
34
215
171
```