TESTE DE DUPLICAÇÃO DE NÚMERO

O teste de duplicação de números (*Number Duplication Test (NTD) do inglês*) não é especificamente um teste da lei de Benford; é um teste associado que pode ser usado para fornecer mais informações resultantes dos testes da lei de Benford. O NDT pode gerar números específicos que causaram picos nos testes de primeira ordem (o primeiro teste de dois dígitos é um exemplo) e o teste de soma, que é um teste avançado da Lei de Benford.

Os picos de qualquer teste de primeira ordem são causados por números que ocorrem com mais frequência do que o esperado, enquanto os picos no teste de soma são geralmente devidos a altos volumes dos mesmos números que se repetem com mais frequência do que o normal. Mark Nigrini escreveu: “O teste de duplicação de números foi desenvolvido como parte da minha dissertação de doutorado quando eu procurava valores de contribuintes duplicados de maneira anormal em frequência. Eu acreditava que essas duplicações anormais existiam porque os contribuintes estavam inventando números e que, como pensamos da mesma forma, as pessoas gravitavam para formar os mesmos números.”

Mark J. Nigrini, Benford’s Law: Applications for Forensic Accounting, Auditing, and Fraud Detection (Hoboken, NJ: John Wiley & Sons, 2012).

O NDT deve ser aplicado a diferentes critérios no conjunto de dados.

- Maior ou igual a 10
- De 0,01 a 9,99
- De –0,01 a –9,99
- Menor ou igual a –10

A saída deve mostrar a quantidade duplicada e a contagem para cada quantidade.

A classificação das contagens apresenta aquelas com o maior número de valores duplicados próximo ao topo dos resultados.

Para ilustração, usaremos um arquivo de dados de despesas de viagem para executar o NDT. O arquivo de despesas de viagem possui campos para os vários tipos de despesas de viagem, como transporte, passagem aérea, acomodação, refeições e assim por diante. Testaremos os valores de acomodação e valores iguais ou superiores a US 10,00.

Vamos importar as bibliotecas necessárias para executar as análises:

In [1]:

```
import pandas as pd
import matplotlib.pyplot as plt
import benford as bf

%matplotlib inline
```

In [2]:

```
# Importando o dataset

travel_expenses = pd.read_csv('Travel Expenses.csv', parse_dates = ['START_DATE_DATE','END_DATE_DATE',
                                                                    'START_DATE_TIME','END_DATE_TIME'])
travel_expenses.head()
```

Out[2]:

	EMPLOYEE_NO	START_DATE_DATE	START_DATE_TIME	END_DATE_DATE	END_DATE_TIME	AIR_FARE	OTHER_TRANSPORTATION	ACCOMMOD
0	1453	2009-10-30	2019-09-19	2010-10-15	2019-09-19	0.0	177.50	
1	1044	2009-11-18	2019-09-19	2009-11-18	2019-09-19	0.0	0.00	
2	1044	2009-11-24	2019-09-19	2009-11-24	2019-09-19	0.0	0.00	
3	1470	2009-11-24	2019-09-19	2010-01-07	2019-09-19	0.0	24.00	

	EMPLOYEE_NO	START_DATE_DATE	START_DATE_TIME	END_DATE_DATE	END_DATE_TIME	AIR_FARE	OTHER_TRANSPORTATION	ACCOMMOD
4	1209	2009-12-15	2019-09-19	2010-03-17	2019-09-19	0.0	327.04	

In [ ]:

```
travel_expenses.shape
```

Como irei utilizar somente a coluna `ACCOMMODATION` criarei uma nova tabela apenas com a coluna de acomodação, com valores maiores ou iguais a U$ 10.00

In [3]:

```
travel_acomodação = travel_expenses[travel_expenses["ACCOMMODATION"]>=10]
travel_acomodação.head()
```

Out[3]:

	EMPLOYEE_NO	START_DATE_DATE	START_DATE_TIME	END_DATE_DATE	END_DATE_TIME	AIR_FARE	OTHER_TRANSPORTATION	ACCOMMO
6	1453	2009-12-17	2019-09-19	2009-12-17	2019-09-19	2978.57	0.00	
15	1236	2010-01-27	2019-09-19	2010-01-28	2019-09-19	0.00	158.74	
16	1440	2010-02-01	2019-09-19	2010-05-03	2019-09-19	1159.25	136.00	
17	1009	2010-02-03	2019-09-19	2010-06-07	2019-09-19	0.00	0.00	
19	1048	2010-02-04	2019-09-19	2010-06-03	2019-09-19	0.00	79.75	

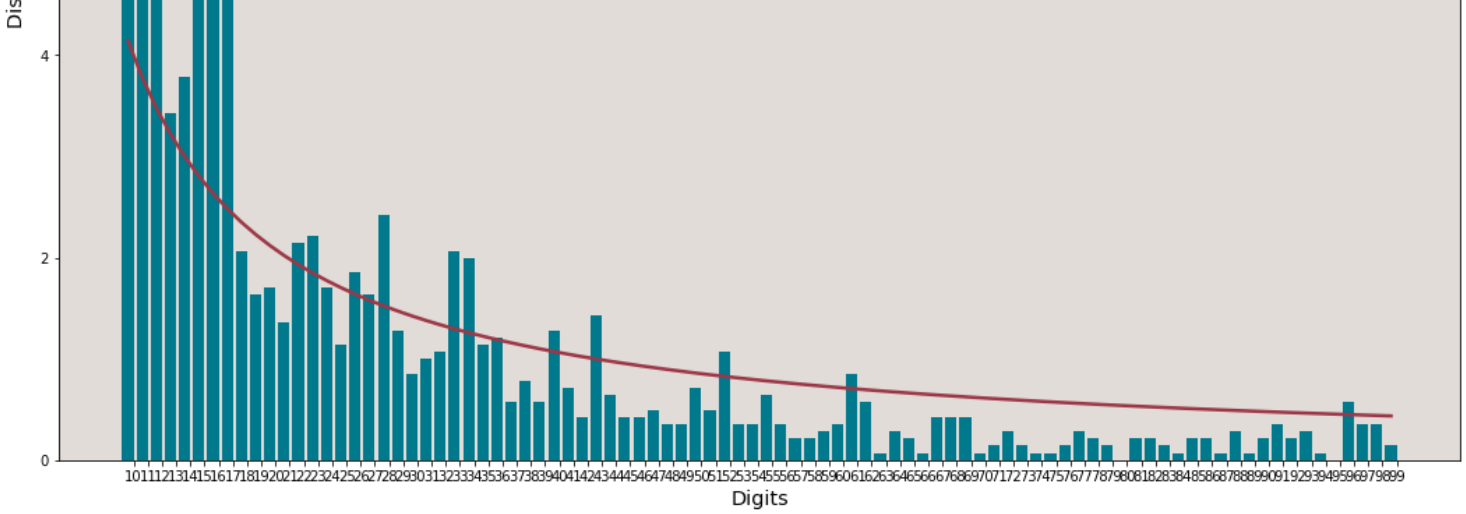
Primeiro vamos ver como a categoria a ser analisada se comporta no teste dos 2 primeiros digítos da Lei de Benford.

In [4]:

```
teste_2digitos = bf.first_digits(travel_acomodação.ACCOMMODATION, digs= 2)
```

Initialized sequence with 1403 registries.

Test performed on 1403 registries.
Discarded 0 records < 10 after preparation.



In [5]:

```
# Verificando os 10 itens com maiores contagens nos 2 primeiros digítos.
teste_2digitos.sort_values('Counts',ascending= False ).head(10)
```

Out[5]:

First_2_Dig	Counts	Found	Expected
12	135.0	0.096222	0.034762
11	117.0	0.083393	0.037789
17	91.0	0.064861	0.024824
10	76.0	0.054170	0.041393
16	73.0	0.052031	0.026329
15	66.0	0.047042	0.028029
14	53.0	0.037776	0.029963
13	48.0	0.034212	0.032185
28	34.0	0.024234	0.015240
23	31.0	0.022096	0.018483

Vemos que existem vários picos no teste dos 2 primeiros digítos. Vamos ver como o dataset se comporta no teste da soma.

O teste de soma analisa os primeiros dois dígitos nos dados agrupando os registros dos primeiros dois dígitos e depois calculando a soma de cada grupo.

Montantes com os mesmos dois primeiros dígitos, como 1200, 125, 12, 1234 por exemplo, são somados. Usando os valores calculados e somados, o processo determina se uma distribuição uniforme é seguida.

O teste de soma identifica números excessivamente grandes em comparação com o restante dos dados. O teste é baseado em somas e não em contagens, como nos outros testes da lei de Benford. Em teoria, as somas de números com os mesmos dois primeiros dígitos devem ser iguais em distribuição. No entanto, em conjuntos de dados normais, existem duplicações anormais regulares de grandes números que podem ser causadas por alguns números muito grandes ou um volume alto de números moderadamente grandes. Análises adicionais serão necessárias nesse caso.

In [6]:

```
teste_soma = bf.summation(travel_acomodação.ACCOMMODATION,top= 10) # Mostrar as 10 maiores diferenças.
```

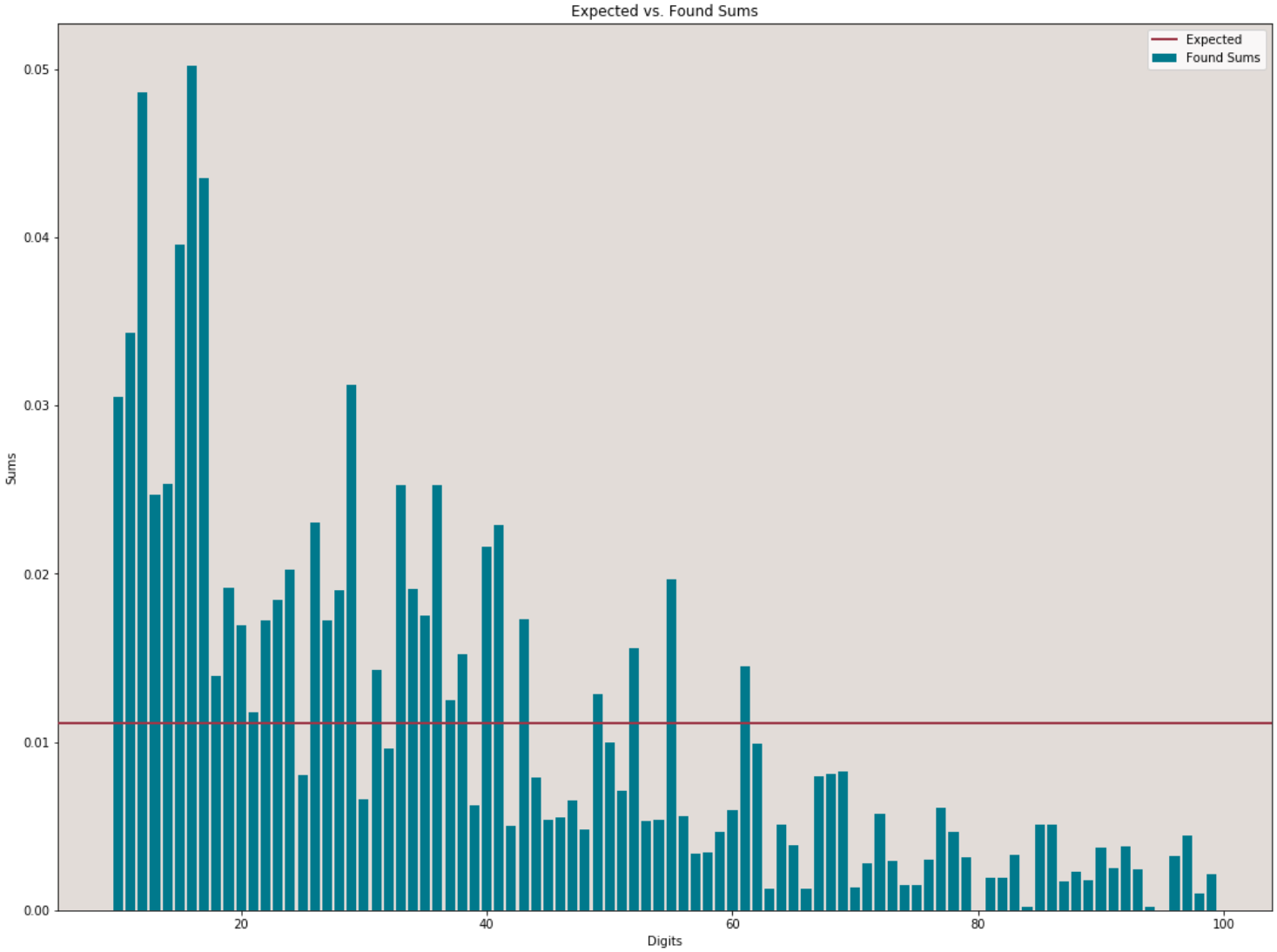
Initialized sequence with 1403 registries.

Test performed on 1403 registries.

The top 10 diferences are:

```
       Sum   Percent    AbsDif
F2D
10   1548276  0.030522  0.019411
11   1742079  0.034343  0.023232
12   2466108  0.048616  0.037505
13   1252169  0.024685  0.013574
14   1285125  0.025335  0.014224
```

14 1283123 0.025333 0.014224
15 2006808 0.039562 0.028451
16 2546904 0.050209 0.039098
17 2208102 0.043530 0.032419
18 705314 0.013904 0.002793
19 973333 0.019188 0.008077



Agora vamos ver a aplicação do teste de duplicação na mesma categoria para ver o seu comportamento. O resultado mostrará as 10 entradas com mais incidências.

In [7]:

```
travel_dupl = bf.duplicates(travel_acomodação.ACCOMMODATION,top_Rep= 10)
```

Found 170 duplicated entries
The entries with the 10 highest repitition counts are:
Entries
174.02 52
123.17 39
124.30 27
168.37 27
283.68 22
111.87 22
157.07 21
113.00 20
134.47 15
271.20 12
Name: Count, dtype: int64

Podemos usar como exemplo para futura averiguação, o número 12, onde no teste dos 2 primeiros digítos ele foi o que teve o maior pico. Junte-se a isso, o número 12 também foi o 3º com a maior diferença no teste da soma e no teste de duplicação, os valores 123.17 e 124.30 (2 primeiros digitos = 12) apareceram juntos 39 e 27 vezes respecitvamente.

Podemos usar esses 2 valores como filtros para uma análise mais aprofundada, como demonstrado a seguir, onde, com esse filtro, podemos fazer outros tipos de análises mais aprofundadas para os 2 valores discrepantes.

In [8]:

```
travel_acomodação[(travel_acomodação['ACCOMMODATION'] == 123.17)|
                  (travel_acomodação['ACCOMMODATION'] ==124.30)]
```

Out[8]:

	EMPLOYEE_NO	START_DATE_DATE	START_DATE_TIME	END_DATE_DATE	END_DATE_TIME	AIR_FARE	OTHER_TRANSPORTATION	ACCOMM
28	1160	2010-02-24	2019-09-19	2010-03-22	2019-09-19	832.58	123.20	
356	1013	2010-05-06	2019-09-19	2010-05-07	2019-09-19	578.35	156.00	
630	1420	2010-06-08	2019-09-19	2010-06-09	2019-09-19	0.00	0.00	
678	1084	2010-06-13	2019-09-19	2010-06-14	2019-09-19	794.60	0.00	
695	1413	2010-06-14	2019-09-19	2010-06-14	2019-09-19	0.00	0.00	
861	1090	2010-07-04	2019-09-19	2010-07-29	2019-09-19	0.00	15.00	
942	1334	2010-07-14	2019-09-19	2010-07-15	2019-09-19	682.99	0.00	
955	1009	2010-07-19	2019-09-19	2010-07-19	2019-09-19	0.00	0.00	
957	1027	2010-07-19	2019-09-19	2010-07-20	2019-09-19	0.00	0.00	
963	1334	2010-07-19	2019-09-19	2010-07-20	2019-09-19	1033.16	16.95	
964	1339	2010-07-19	2019-09-19	2010-07-19	2019-09-19	0.00	0.00	
1097	1141	2010-08-10	2019-09-19	2010-08-11	2019-09-19	535.90	64.50	
1103	1377	2010-08-10	2019-09-19	2010-08-11	2019-09-19	637.60	70.00	
1216	1092	2010-08-17	2019-09-19	2010-08-31	2019-09-19	0.00	0.00	
1234	1202	2010-08-19	2019-09-19	2010-08-20	2019-09-19	763.11	6.00	
1289	1248	2010-08-26	2019-09-19	2010-08-27	2019-09-19	0.00	131.08	
1295	1063	2010-08-27	2019-09-19	2010-08-28	2019-09-19	0.00	0.00	
1297	1342	2010-08-27	2019-09-19	2010-08-27	2019-09-19	0.00	0.00	
1449	1122	2010-09-20	2019-09-19	2010-09-20	2019-09-19	0.00	0.00	
1464	1416	2010-09-20	2019-09-19	2010-09-20	2019-09-19	0.00	0.00	
1467	1442	2010-09-20	2019-09-19	2010-09-21	2019-09-19	0.00	152.80	
1505	1300	2010-09-22	2019-09-19	2010-09-23	2019-09-19	355.75	0.00	
1519	1235	2010-09-23	2019-09-19	2010-09-24	2019-09-19	1140.51	146.95	

	EMPLOYEE_NO	START_DATE_DATE	START_DATE_TIME	END_DATE_DATE	END_DATE_TIME	AIR_FARE	OTHER_TRANSPORTATION	ACCOMM
1555	1302	2010-09-27	2019-09-19	2010-09-28	2019-09-19	603.70	45.00	
1700	1334	2010-10-13	2019-09-19	2010-10-14	2019-09-19	0.00	0.00	
1703	1443	2010-10-13	2019-09-19	2010-10-14	2019-09-19	0.00	0.00	
1707	1215	2010-10-14	2019-09-19	2010-10-15	2019-09-19	1255.80	0.00	
1711	1277	2010-10-14	2019-09-19	2010-10-15	2019-09-19	1255.80	0.00	
1730	1070	2010-10-18	2019-09-19	2010-10-18	2019-09-19	521.21	69.35	
1797	1377	2010-10-23	2019-09-19	2010-10-24	2019-09-19	579.97	65.60	
...	...	...	...	...	...	...	...	
1932	1139	2010-11-08	2019-09-19	2010-11-09	2019-09-19	0.00	180.40	
1946	1430	2010-11-08	2019-09-19	2010-11-09	2019-09-19	1027.79	58.30	
2007	1013	2010-11-18	2019-09-19	2010-11-19	2019-09-19	0.00	46.49	
2033	1421	2010-11-22	2019-09-19	2010-11-23	2019-09-19	1065.59	71.00	
2072	1129	2010-11-29	2019-09-19	2010-11-30	2019-09-19	0.00	0.00	
2073	1165	2010-11-29	2019-09-19	2010-11-30	2019-09-19	0.00	0.00	
2079	1402	2010-11-29	2019-09-19	2010-11-30	2019-09-19	0.00	0.00	
2090	1360	2010-11-30	2019-09-19	2010-12-01	2019-09-19	0.00	156.01	
2145	1376	2010-12-08	2019-09-19	2010-12-09	2019-09-19	0.00	80.00	
2150	1115	2010-12-09	2019-09-19	2010-12-10	2019-09-19	0.00	199.60	
2156	1360	2010-12-09	2019-09-19	2010-12-10	2019-09-19	0.00	84.62	
2248	1334	2011-01-11	2019-09-19	2011-01-12	2019-09-19	700.83	35.00	
2260	1430	2011-01-12	2019-09-19	2011-01-13	2019-09-19	0.00	102.89	
2284	1038	2011-01-17	2019-09-19	2011-01-18	2019-09-19	0.00	0.00	
2287	1111	2011-01-17	2019-09-19	2011-01-18	2019-09-19	0.00	0.00	
2377	1022	2011-01-27	2019-09-19	2011-02-01	2019-09-19	1761.19	200.87	

	EMPLOYEE_NO	START_DATE_DATE	START_DATE_TIME	END_DATE_DATE	END_DATE_TIME	AIR_FARE	OTHER_TRANSPORTATION	ACCOMM
2378	1060	2011-01-27	2019-09-19	2011-01-28	2019-09-19	0.00	180.12	
2384	1241	2011-01-27	2019-09-19	2011-01-28	2019-09-19	0.00	146.47	
2401	1013	2011-01-31	2019-09-19	2011-02-01	2019-09-19	1142.66	64.95	
2405	1102	2011-01-31	2019-09-19	2011-02-01	2019-09-19	1142.66	60.00	
2407	1169	2011-01-31	2019-09-19	2011-02-01	2019-09-19	673.62	30.00	
2435	1333	2011-02-02	2019-09-19	2011-02-03	2019-09-19	0.00	76.28	
2466	1013	2011-02-07	2019-09-19	2011-02-09	2019-09-19	0.00	139.57	
2566	1102	2011-02-18	2019-09-19	2011-02-19	2019-09-19	0.00	381.16	
2606	1241	2011-02-24	2019-09-19	2011-02-24	2019-09-19	0.00	0.00	
2609	1333	2011-02-24	2019-09-19	2011-02-24	2019-09-19	0.00	176.11	
2672	1329	2011-03-03	2019-09-19	2011-03-04	2019-09-19	682.80	0.00	
2706	1022	2011-03-10	2019-09-19	2011-03-10	2019-09-19	0.00	0.00	
2734	1285	2011-03-17	2019-09-19	2011-03-19	2019-09-19	0.00	241.56	
2757	1139	2011-03-23	2019-09-19	2011-03-24	2019-09-19	0.00	61.60	

66 rows × 18 columns

O NDT é um bom provedor de informações detalhadas para identificar picos no teste de somatória e no teste dos dois primeiros dígitos. Isso ajuda a determinar se é necessária uma maior investigação.

Com relação a documentação para a biblioteca benford_py, onde é possível realizar outros testes da Lei de Benford como alterar alguns parâmetros que se adequem melhor a cada necessidade, favor consultar a documentação do pacote clicando aqui.