Exercise 4

Unknown Author

March 23, 2014

1 Task 1

$$f(\text{Agatha}) = 30$$
$$f(\text{Christie}) = 117$$
$$f(\text{Agatha Christie}) = 20$$

with

$$p(a) = \frac{f(a)}{n}$$

.

We can compute the probability of “Agatha Christie” appearing under the null hypothesis (“Agatha” and “Christie” are independent) as

$$p_{ac} = p(\text{Agatha}) \cdot p(\text{Christie})$$

This is very easy to implement in python:

In [2]:
```
n = 1000000 # one million words
f_christie = 117
f_agatha = 30

p_christie = f_christie/n
p_agatha = f_agatha/n
p_ac = p_agatha * p_christie
print("p_ac = {}".format(p_ac))
```

```
p_ac = 3.51e-09
```

We want now to apply a t-test to find out, if the null hypothesis is met. For this we model this as a Bernoulli process with

$$\mu = p_{ac}$$
$$\bar{x} = p(\text{Agatha Christie}) = \frac{20}{n}$$
$$\sigma^2 = p(\text{Agatha Christie}) \cdot (1 - p(\text{Agatha Christie})) \approx p(\text{Agatha Christie})$$

And compute then the t value we observed

$$t_{obs} = \frac{\overline{x} - \mu}{\sqrt{\frac{\sigma^2}{n}}} = \frac{\overline{x} - \mu}{\sqrt{\sigma^2}} \cdot \sqrt{n}$$

.

Again this can be easily realized for our example in python

In [3]:
```
from math import sqrt

f_agatha_christie = 20
p_agatha_christie = f_agatha_christie/n
print("x_bar = {}".format(p_agatha_christie))

mu = p_ac
sigma2 = p_agatha_christie*(1 - p_agatha_christie)

t_obs = (p_agatha_christie - mu)/sqrt(sigma2)*sqrt(n)
print("t_obs = {}".format(t_obs))
```

```
x_bar = 2e-05
t_obs = 4.471395809321142
```

We then compare this value to the t-test distribution for with

$$\text{dof} = n \approx \infty$$

for a certainty of 99%:

In [4]:
```
t_obs < 2.576
```

Out [4]:
```
False
```

So we come to the conclusion, that "Agatha Christie" appears more in the text than just by random occurences. For Task b) adapt the numbers in the following way:

In [5]:
```
f_christie = 200
f_agatha = 300

p_christie = f_christie/n
p_agatha = f_agatha/n
p_ac = p_agatha * p_christie
print("p_ac = {}".format(p_ac))

from math import sqrt

f_agatha_christie = 6
p_agatha_christie = f_agatha_christie/n
print("x_bar = {}".format(p_agatha_christie))

mu = p_ac
sigma2 = p_agatha_christie*(1 - p_agatha_christie)

t_obs = (p_agatha_christie - mu)/sqrt(sigma2)*sqrt(n)
print("t_obs = {}".format(t_obs))
```

```
p_ac = 6e-08
x_bar = 6e-06
t_obs = 2.4250021203726195
```

And again compare it to the same bound as before:

In [6]:
```
t_obs < 2.576
```

Out [6]:
```
True
```

And come to the conclusion that it may very well possible that the "Agatha Christie" we found are only random occurences and the two terms are most likely not related.

2 Task 2

Make a chi-square test based on the observations in table a:

In [7]:
```
n_woman = 20000
n_all = 60000
n_not_woman = 40000

# As tuples, observed value to the left, expected to the right
# first column
n_woman_you = (350, 1000/n_all*n_woman)
n_woman_not_you = (19650, 59000/n_all*n_woman)

# second column
n_not_woman_you = (650, 1000/n_all*n_not_woman)
n_not_woman_not_you = (39350, 59000/n_all*n_not_woman)

def make_summand(tupel):
    return (tupel[0] - tupel[1])**2/tupel[1]

chi2a = make_summand(n_woman_you) + make_summand(n_woman_not_you) + make_summand(n_not_
print("chi square = {}".format(chi2a))
```

```
chi square = 1.2711864406779663
```

We then compare this value with the value found for 1 degree of freedom and 99% certainty:

In [8]:
```
chi2a < 6.6353
```

Out [8]:
```
True
```

And come to the conclusion that there is no correlation. For task b) we do the same with different numbers:

In [9]:
```
# first column
n_woman_I = (450, 1000/n_all*n_woman)
n_woman_not_I = (19550, 59000/n_all*n_woman)

# second column
n_not_woman_I = (550, 1000/n_all*n_not_woman)
n_not_woman_not_I = (39450, 59000/n_all*n_not_woman)

chi2b = make_summand(n_woman_I) + make_summand(n_woman_not_I) + make_summand(n_not_wom
print("chi square = {}".format(chi2b))
```

```
chi square = 62.288135593220346
```

And make again the comparison with our bound:

In [10]:
```
chi2b < 6.635
```

Out [10]:
```
False
```

And come the conclusion that there is some correlation.

3 Task 3

For this task we first parse the file and find all tokens and their respective frequency

In [11]:
```
from collections import Counter
import re

with open('input_ex3.txt', encoding='iso-8859-15') as f:
    # split() method splits on any whitespace by default
    arry = f.read().split()
    arry = list(filter(lambda x: not re.match(r'[<>\d]+', x), arry))

counter = Counter()
```

```
n_words = 0
for i in arry:
    # remove trailing dots
    if i[-1] == '.':
        i = i[0:-1]
    counter[i] += 1
    n_words += 1

n_types = len(counter)
most_common = counter.most_common(100)
print('Found {} words and {} types. The 100 most common are:'.format(n_words, n_types)
print(most_common)
```

```
Found 192046 words and 14079 types. The 100 most common are:
[('the', 16569), ('of', 11796), ('to', 6840), ('and', 4876), ('in',
4130), ('a', 3826), ('be', 3748), ('that', 2677), ('is', 1970),
('which', 1961), ('it', 1884), ('by', 1710), ('as', 1596), ('The',
1266), ('have', 1215), ('would', 1206), ('or', 1191), ('for', 1187),
('will', 1178), ('not', 1151), ('this', 1107), ('their', 1078),
('with', 1011), ('from', 1004), ('are', 981), ('on', 907), ('an',
885), ('they', 786), ('been', 786), ('may', 778), ('all', 646),
('its', 624), ('has', 577), ('more', 569), ('State', 561), ('at',
552), ('other', 550), ('than', 549), ('government', 544), ('any',
538), ('It', 495), ('power', 488), ('States', 488), ('one', 475),
('no', 449), ('can', 447), ('those', 447), ('them', 431), ('but',
428), ('must', 427), ('we', 396), ('most', 389), ('who', 384),
('such', 384), ('so', 374), ('upon', 371), ('these', 369), ('I', 367),
('his', 359), ('people', 348), ('there', 344), ('same', 337), ('if',
336), ('against', 328), ('should', 327), ('was', 322), ('every', 320),
('national', 314), ('might', 305), ('federal', 303), ('In', 303),
('under', 301), ('our', 299), ('into', 296), ('only', 279), ('public',
273), ('were', 265), ('had', 261), ('But', 256), ('States,', 256),
('ought', 254), ('some', 246), ('between', 244), ('general', 243),
('authority', 238), ('great', 238), ('shall', 234), ('could', 232),
('This', 229), ('less', 228), ('New', 225), ('each', 220),
('government,', 217), ('Constitution', 204), ('part', 204),
('different', 202), ('United', 200), ('particular', 198), ('two',
195), ('well', 193)]
```

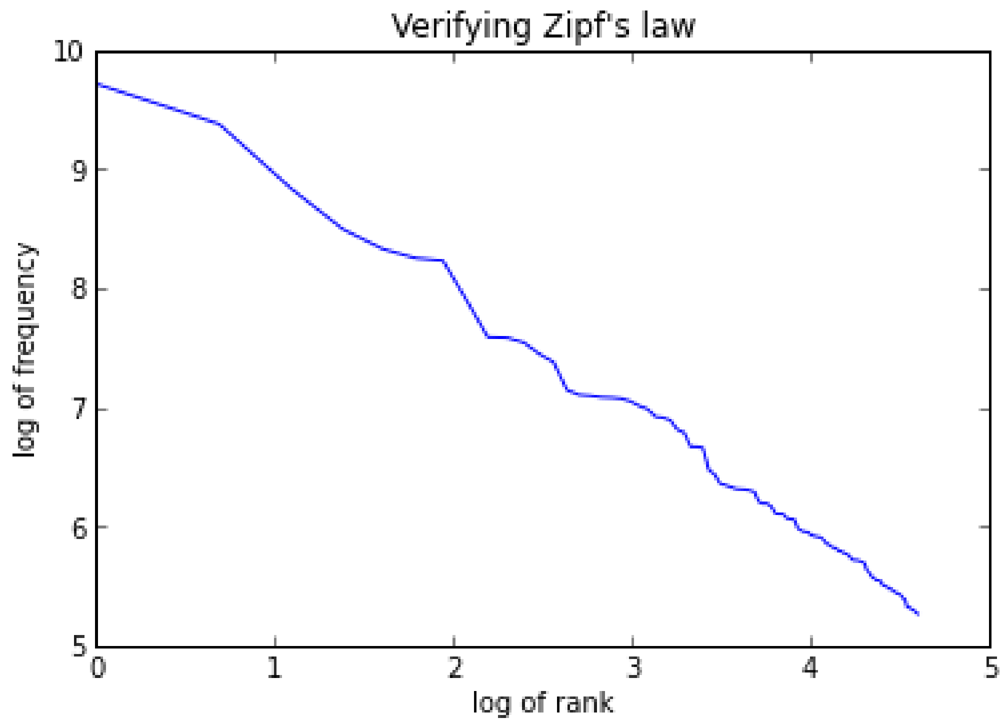
And then visualize this data to verify Zipfs law

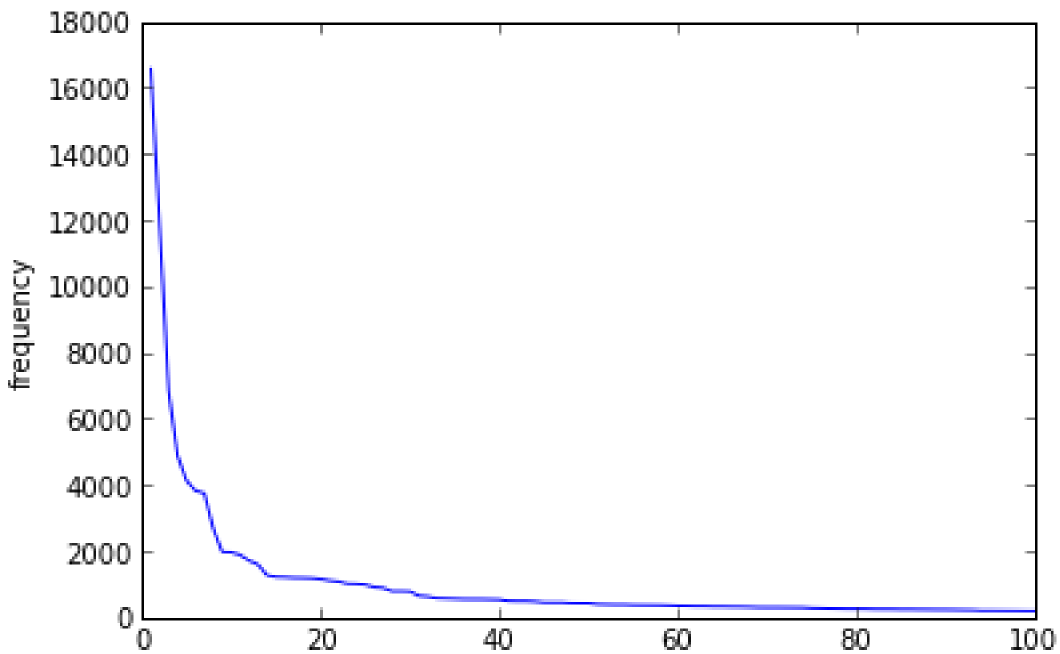
In [20]:

```
%matplotlib inline
import matplotlib.pyplot as plt

from math import log
plt.figure()
plt.title("Verifying Zipf's law")
plt.plot([log(rank) for rank in range(1,101)], [log(freq) for (word,freq) in most_comm
plt.xlabel('log of rank')
plt.ylabel('log of frequency')
plt.show()

plt.figure()
plt.plot(range(1,101), [freq for (word,freq) in most_common])
plt.ylabel('frequency')
plt.show()
```


Verifying Zipf's law
log of frequency
log of rank
10
9
8
7
6
5
0
1
2
3
4
5


frequency
18000
16000
14000
12000
10000
8000
6000
4000
2000
0
0
20
40
60
80
100

In [ ]: