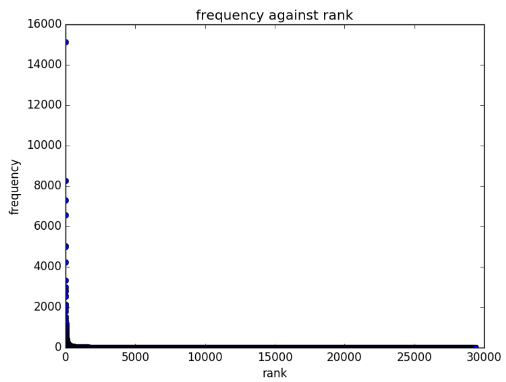
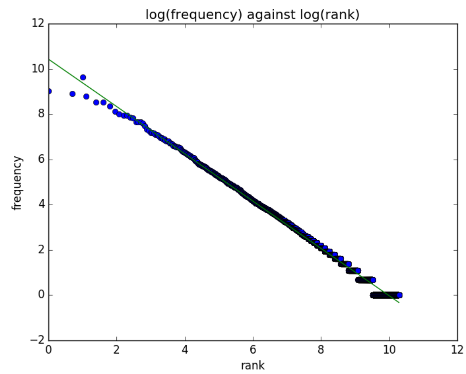
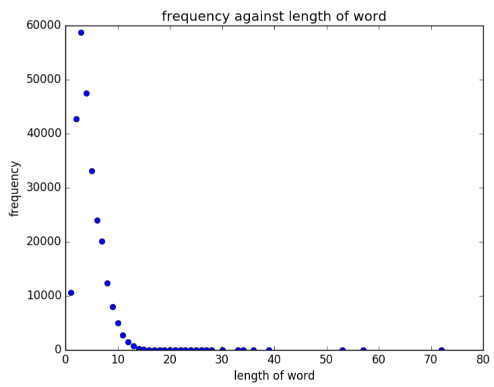
Jin Young Jeon
23051612
Cog Sci 190

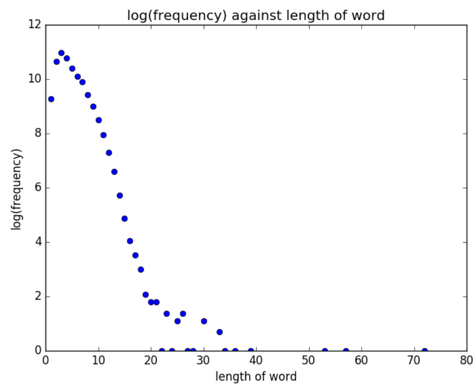
In Class Lab

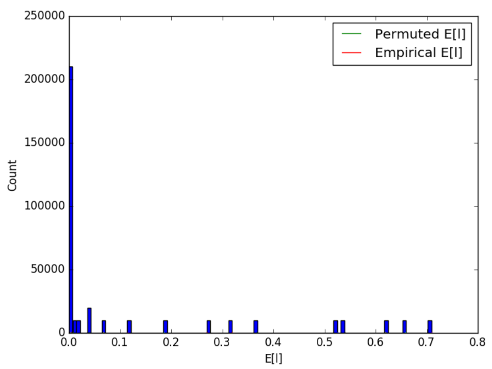
------- -----
1.the 15130
2.of 8261
3.and 7287
4.a 6578
5.to 5042
6.in 5006
7.he 4227
8.his 3331
9.i 2996
10.s 2828
11.that 2795
12.with 2562
13.it 2530
14.was 2134
15.on 2129
16.you 2084
17.for 1963
18.her 1785
19.him 1525
20.is 1462
------- -----











Slope: -1.047
Pearson Correlation: 5.2939570792049181e-09
E-l value: 4.48533 | P-value: 0
The calculated E-l value is similar to the peak shown at log(frequency) against length of word graph. Words with letters from 3-7 are used most frequently.
The calculated p-value was 0 which is significantly low meaning that the result E[l] values are different.

(Appendix)

```
# Import relevant libraries
import string
import numpy as np
from scipy.stats import pearsonr
import matplotlib.pyplot as plt

import collections
import re
from tabulate import tabulate
from math import log
######################
# ________________________________________________________________
# Table 1

# Task 1: Report the most frequent 20 words in "txt" and their frequencies

words = re.findall(r'\w+', open('ulysses.txt').read().lower())
frequency = collections.Counter(words).most_common(20)
# print(frequency)

table1 =
[["1.the",15130],["2.of",8261],["3.and",7287],["4.a",6578],["5.to",5042],["6.in",5006],["7.he",4227],["8.his",3331],["9.i",2996],["
10.s",2828],["11.that",2795],["12.with",2562],["13.it",2530],["14.was",2134],["15.on",2129],["16.you",2084],["17.for",1963],["1
8.her",1785],["19.him",1525],["20.is",1462]]
print(tabulate(table1))
# [.5pt]
# -------  -----
# 1.the    15130
# 2.of      8261
# 3.and     7287
# 4.a       6578
# 5.to      5042
# 6.in      5006
# 7.he      4227
# 8.his     3331
# 9.i       2996
# 10.s      2828
# 11.that   2795
# 12.with   2562
# 13.it     2530
# 14.was    2134
# 15.on     2129
# 16.you    2084
# 17.for    1963
# 18.her    1785
# 19.him    1525
# 20.is     1462
# -------  -----

frequency2 = collections.Counter(words).most_common(29379)#max number of words found
# ________________________________________________________________
# Figure 1 (2 subplots) - Replicating Zipf's law (Zipf, 1949)

# Task 2a: Scatter plot word frequency (y-axis) against rank (x-axis)
lst_y = []
for i in range(0, len(frequency2)):
        lst_y.append(frequency2[i][1])
lst_x = []
for i in range(0, len(lst_y)):
        lst_x.append(i)

plt.plot(lst_x, lst_y, 'o')
plt.xlabel('rank')
```

```
plt.ylabel('frequency')
plt.title('frequency against rank')
plt.show()

# Task 2b: Scatter plot log(frequency) against log(rank)
new_x = []
for i in range(0, len(lst_x)):
	if lst_x[i] == 0:
		new_x.append(1)
	else:
		new_x.append(log(lst_x[i]))
new_y = []
for i in range(0, len(lst_y)):
	new_y.append(log(lst_y[i]))

plt.plot(new_x, new_y, 'o')
plt.xlabel('rank')
plt.ylabel('frequency')
plt.title('log(frequency) against log(rank)')
plt.plot(np.unique(new_x), np.poly1d(np.polyfit(new_x, new_y, 1))(np.unique(new_x)))
plt.show()

# Task 2c: Compute and report slope in Task 2b, i.e. slope of log(frequency) vs log(rank)
# Hint: Use numpy.polyfit function of order 1
# [.5pt]
slope = (-0.03 -10.44)/(10-0) #arbitrary points on best fit line from 2b
print(slope) #-1.047


# ____________________________________________________________________
# Figure 2 (2 subplots) - Word length and frequency

# Task 3a: Scatter plot word frequency against word length
#lists all the counts for each len of word
counts = collections.Counter([len(word.strip('?!,.')) for word in txt.split()])

len_word = []
for i in counts:
	len_word.append(i)
len_word = sorted(len_word)
len_frequency = (10589, 42712, 58804, 47521, 33178, 24055, 20095, 12323, 8038, 4965, 2806, 1495, 738, 303, 132, 58, 34, 20,
8, 6, 6, 1, 4, 1, 3, 4, 1, 1, 3, 2, 1, 1, 1, 1, 1, 1)

plt.plot(len_word, len_frequency, 'o')
plt.xlabel('length of word')
plt.ylabel('frequency')
plt.title('frequency against length of word')
plt.show()
# Task 3b: Scatter plot log(frequency) against word length
log_frequency = []
for i in range(0, len(len_frequency)):
	log_frequency.append(log(len_frequency[i]))

plt.plot(len_word, log_frequency, 'o')
plt.xlabel('length of word')
plt.ylabel('log(frequency)')
plt.title('log(frequency) against length of word')
plt.show()

# Task 3c: Compute Pearson correlation between log(frequency) and word length
# Hint: Use scipy.stats.pearsonr function; first output is correlation
# [.5pt]
print(pearsonr(len_word, log_frequency))
# 5.2939570792049181e-09

# ____________________________________________________________________
```

```
# Figure 3 (1 plot) - Further analysis of word (coding) length and frequency

# Task 4: Near-optimal word length analysis
# Compare empirical expected word length E[l] = sum_w p(w) l(w)
# to scrambled expected word length
len_word
g = []
for i in len_word:
        g.append(counts[i])
summ = sum(g)
prob = []
for i in range(0, len(g)):
        prob.append(g[i]/summ)

El = []
for i in range(0, len(prob)):
        El.append(prob[i]*len_word[i])

E_l = sum(El)
#4.48533

# Permutation test by shuffling word identities and recomputing E[l]
npermutation = 10000;
# E_l_perm = [None] * npermutation;
E_l_perm = El * npermutation;

def permtest(length, freq, nperm):
        d = np.abs(np.mean(length) - np.mean(freq))
        i = 0
        n = len(length)
        sets = np.concatenate([length, freq])
        lst_permdiff = []
        for k in range(nperm):
                np.random.shuffle(sets)
                permdiff = np.abs(np.mean(sets[:n]) - np.mean(sets[n:]))
                i += d < permdiff
                lst_permdiff.append(permdiff)
        pvalue = i / nperm
        return pvalue
print(permtest(len_word, log_frequency, 10000))
# Report p-value
# 0 #think I did this incorrectly

# Uncomment the following when you obtain E_l = E[l] and E_l_perm from above
# Histogram permuted E[l] and compare with empirical E[l]

plt.figure()
plt.hist(E_l_perm,100)
plt.plot([El,El],[0,500])
plt.ylabel('Count')
plt.xlabel('E[l]')
plt.legend(['Permuted E[l]','Empirical E[l]'])
plt.show()
```