Now you are ready for some challenges…

**Pretend that ciphertext1 was really encrypted and that you didn’t really know the keyword-length. Analyze the frequencies for keyword-lengths of length 3,4,5, and 6. What’s happening to our frequency analysis? Why is this happening? Please provide the L1 Distances and Friedman Indexes used for your analysis.**

Keyword length (3)

L1 Distance = 0.1479

index: 0.0706

Keyword length (4)

L1 Distance = 0.0672

index: 0.0672

Keyword length (5)

L1 Distance = 0.0953

index: 0.0696

Keyword length (6)

L1 Distance = 0.1338

index: 0.0678

If ciphertext1.txt was cipher, the keylength would be of length 4 because that is closest to 0.065. The frequency analasis of this text shows that ciphertext1.txt is written in English

**Apply the formula for approximate keyword length in terms of the Friedman index (see page 138 of Barr). A cool way to do this would be to open the m-file displayCoincidenceIndex.m in the Matlab editor (or any other word processor), add a line calculating this expression, and save it under a name such as friedman.m. Then whenever you give Matlab the command “friedman(OriginalTextData)”, it will compute and display both the Friedman index and the estimated keyword length.**

When applying the function, friedman.m to ciphetext1.txt, it returned an estimated Keylength of 0.9129 and a CoincidenceIndex of 0.0675, implying that ciphertext1.txt is written in English.

**Load and Cryptanalyze ciphertext4.txt. What is “wrong” with this text? (Hint: How is this text different from other texts?)**

ciphertext4.txt has an unusual amount of “z’s,” setting off the frequency analysis.

**Now try your hand at a real cipher text. Load and Cryptanalyze ciphertext5.txt. Make sure that you have a current version and not one with an old date on it. Please turn in a record of your work: the L1 distances and Friedman Indexes you obtained, keyword lengths tried, frequency graphs analyzed, swaps performed, etc. Also, save your final plaintext to plaintext5.txt. For practice, or more fun, or extra credit, or if ciphertext5.txt turns out not to work well for you, you can repeat all this for ciphertext3.txt—but it's not necessary. Good luck and have fun!**

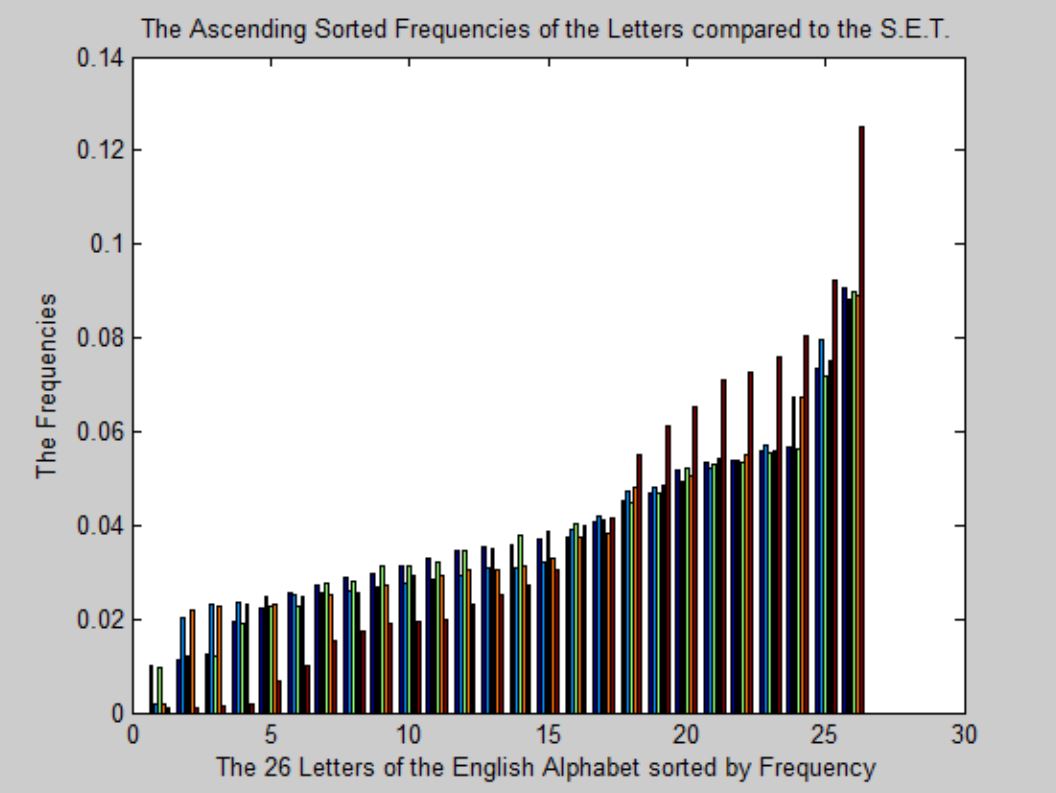
By applying friedman(OriginalTextData), MATLAB returned, a KeyLength of 4.4500, so by that I can assume that the keyword length is most likely 4, 5, or 6. From here I would apply the input initPASCC(4), initPASCC(5), initPASCC(6).

**Keyword Length (4)**

L1 Distance = 0.3498

CoincidenceIndex = 0.0471

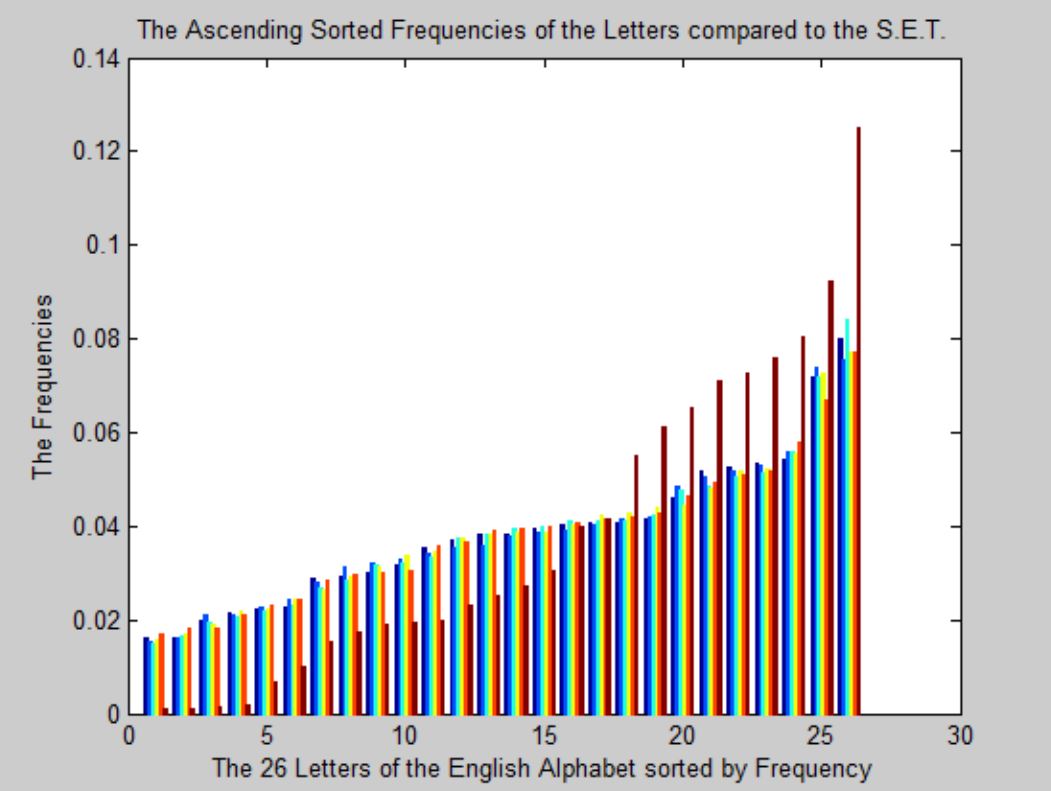
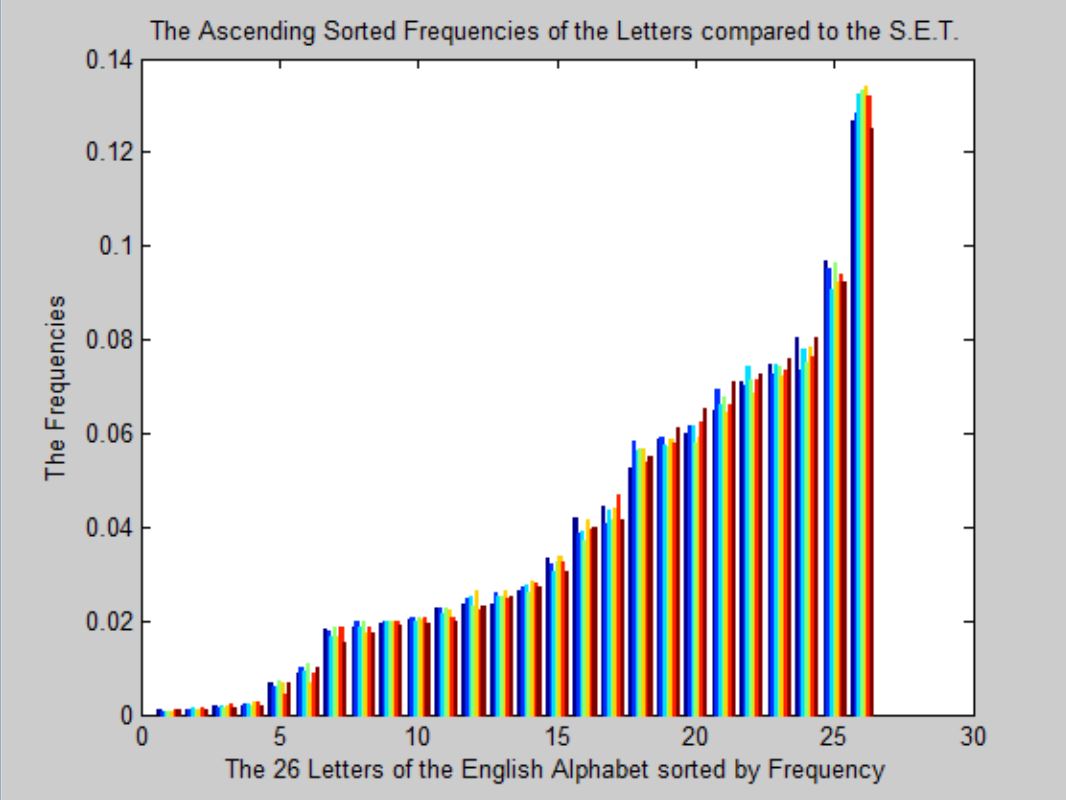
Frequency Analysis:



**Keyword Length (5) Keyword Length (6)**

L1 Distance = 0.4149 L1 Distance = 0.0460

CoincidenceIndex = 0.0446 CoincidenceIndex = 0.0652

Frequency Analysis: Frequency Analysis: 

Because of the CoincidenceIndex of Keyword Length (6) is 0.0652, which is the closest to 0.065, showing that the keyword length of ciphertext5.txt is of length **6**.

From here, I display the cipher matrix with a keylength of 6, and then display the plaintext. From here, I get the below text, in which I will start swapping:

1. swap(1,’p’,’w’);
2. swap(1,’d’,’l’);
3. swap(4,’p’,’w’);
4. swap(5,’o’,’i’);
5. swap(4,’u’,’c’);
6. swap(4,’y’,’g’);
7. swap(4,’g’,’y’);
8. swap(1,’y’,’g’);
9. swap(2,’u’,’m’);
10. swap(5,’e’,’c’);
11. swap(6,’g’,’f’);
12. swap(2,’a’,’e’);
13. swap(3,’n’,’r’);
14. swap(2,’e’,’a’);
15. swap(3,’r’,’n’);
16. swap(4,’l’,’d’);
17. swap(2,’i’,’n’);
18. swap(4,’n’,’i’);
19. swap(5,’r’,’h’);
20. swap(6,’i’,’o’);
21. swap(4,’p’,’y’);
22. swap(5,’w’,’g’);
23. swap(6,’l’,’d’);
24. swap(2,’f’,’w’);
25. swap(4,’m’,’f’);
26. swap(1,’r’,’h’);
27. swap(4,’r’,’h’);
28. swap(6,’r’,’h);
29. swap(5,’s’,’n’);
30. from this point I got the string of text **whathowhathothisfellowisdancingmad** which, after searching on a popular search engine, I was able to determine is out of the Edgar Allan Poe story, *The Golden Bug*. Knowing this, I compare this to the original text of the Poe story and make more swaps to the PlainText.
31. swap(2,’i’,’o’);
32. swap(3,’m’,’n’);
33. swap(6,’b’,’p’);
34. swap(3,’m’,’n);
35. swap(3,’c’,’u’);
36. swap(2,’c’,’u’);
37. swap(3,’d’,’l’);
38. swap(5,’c’,’e’);
39. swap(4,’p’,’g’);
40. swap(1,’u’,’c’);
41. swap(5,’u’,’m’);
42. swap(3,’j’,’x’);
43. swap(2,’j’,’x’);
44. swap(2,’g’,’y’);
45. swap(3,’p’,’w’);
46. swap(3,’h’,’r’);
47. swap(5,'d','l');
48. swap(5,’u’,’c’);
49. swap(6,’w’,’g’);
50. swap(2,’p’,’f’);
51. swap(3,’c’,’m’);
52. swap(5,’n’,’s’);
53. swap(1,’f’,’m’);
54. swap(6,’c’,’u’);
55. swap(2,’p’,’g’);
56. swap(6,’q’,’z’);
57. swap(6,’y’,’b’);

The final document is saved as **plaintext6.txt**. It is Edgar Allan Poe’s, *The Golden Bug*.