Information and Coding, Fall 2014, K. Petersen

# **MatLab Project - Assignment 1**

Prepared by Todd Riley

This project is meant to familiarize you with Matlab and expose you to the strengths (and weaknesses) of this powerful computing tool in the area of Cryptology. While we have chosen MatLab as our computational tool, we could have also used Maple or Mathematica to achieve similar results. Please take some time and examine the MatLab Code written for this Lab. Open the files and try to see what they are doing. (The code is contained in the MatLab m-files.) These scripts and functions can be enhanced and used as a starting point for bigger and better things (like your project). You can modify the way they work, produce graphs, etc. Also, I recommend that you use MatLab 6 Release 12 or higher for this assignment. MatLab made major improvements in the user interface in this version (although more could still be made). You will find that the output will be easier to read and the environment and scripts will be easier to work with for this project in R12 or higher.

You should hand in a description of what you did, what you observed, what you concluded, along with the answers to the questions below and any relevant graphs, tables, and other data. As always for homework, you are encouraged to work together, but your final write-up you should do alone, and you should credit properly any sources or outside help.

The Random Poly-alphabetic Substitution Cipher Builder and Cracker

Copy the set of MatLab m-files and text files to a common folder on your computer. The functions and commands will be used in the MatLab Session to cryptanalyze ciphertexts. (You will notice in the scripts that I used my h: drive and named my folder ‘crypto’. You can name your folder differently as long as you also change the commands.) Note: Copy all the files to this folder, not just folders that contain the files.

Start MatLab and set the “current directory” to your newly created folder where the m-files are located. This informs MatLab where to find the scripts to execute.

Initialize the Global Variables for this Lab by executing: “initEnvironment”. Note: Do not type (or copy) these quotation signs: you are giving a command to Matlab to execute all the commands that are contained in the m-file. But when referring to text files, you SHOULD use the single quotes around the filename. If you get error messages, it may be because the computer cannot find the files called for, either because they are not in the directory to which you are currently set, or because there are misprints in the way you typed the filename or the command.

Load Cipher Text 6 by executing: “loadText('ciphertext6.txt')”. Notice that this ciphertext is actually just the plaintext in this example.

Now perform Friedman’s Index of Coincidence calculation to see if the Cipher Text contains frequencies comparable to a Standard English Text. Remember that the closer this index is to 0.065, the closer the ciphertext resembles the output of a simple monoalphabetic substitution cipher performed on a Standard English Text. Execute the command: “displayCoincidenceIndex(OriginalTextData);”.

Alternatively, you can use the m-file friedman.m by typing “friedman(OriginaltextData)”.

Pretend that this cipher text was really encrypted and that you believe that it was encrypted with a random poly-alphabetic substitution cipher with a keyword-length of 2. (Notice that in a random poly-alpha cipher there is no “keyword” in the real sense, so the keyword-length refers to the number of random substitution ciphers. A Random Poly-Alpha with a keyword-length the same size as the cipher/plain text is a one time pad.). So you would intitialize the PASCC with keyword-length 2 (PASCC stands for Poly-Alphabetic Substitution Cipher Cracker – say that ten times fast!) Execute command: “initPASCC(2);”. This command produces the CipherMatrix with the column size set equal to the argument passed in (“2” in this case). Thus each column in the CipherMatrix represents a mono-alphabetic substitution cipher.

**There are two return results from “initPASCC(2)”. The first gives the L1 distance between the frequencies of the characters in rs in the two alphabets.) Thus, the closer this value is to 0, then the closer the fit to the frequencies of a Standard English Text. Notice here that this accumulated differthe first cipher (column) in the poly-cipher and the frequencies found in a typical Standard English Text. (This is the sum of the absolute values of the differences of the frequencies of the corresponding characteence is pretty close to zero. The second return value is Friedman's Index of Concidence calculated on the first cipher (column) of the Cipher Text. Again, remember that the closer this index is to 0.065 the closer the letters of the domain set resemble the frequencies found in a Standard English Text.**

Now you can actually graph the frequencies and compare them to the frequencies of a S.E.T. by executing : “graphFrequencies”. Notice that the last vertical bar in each column is the frequency for that character in a S.E.T. The frequencies are also sorted, so if the keyword-length (ie – the number of ciphers) is correct then the first "grouping of columns" should be the letter ‘z’ and the last "grouping of columns" should be the letter ‘e’ for each individual cipher. (However, this is not always true.) The columns are colored so that each color represents one of the mono-alphabetic substitution ciphers.

You can view the Cipher Matrix in your MatLab environment by executing: "displayCipherMatrix"

You can also view the original Cipher Text in your MatLab environment by executing: "displayCipherText"

If you look at the ciphertext file with a word processor, such as MS Word, you can view it in columns of a given length by setting the margins.

You can view the candidate Plain Text based on the L1 frequency analysis by executing: "displayPlainText". The candidate Plain Text is obtained by replacing letters in each mono-alphabetic cipher with the letters of a Standard English Text that have the same frequency rank. If for example the letter 'x' had the greatest frequency in a particular cipher, then this letter would be replaced by the letter that had the greatest frequency in a S.E.T. (the letter 'e'). Then the letter with the second greatest frequency in the cipher would be replaced by the letter of greatest frequency in a S.E.T. and so forth.

Notice that this candidate Plain Text needs “some help”. You can provide this help by analyzing the text and performing “swaps” in the substitution matrix in order to correct the misassignments of "cipher letters" to "plain letters" due to the rankings of the frequencies of the letters. Many of the letters of a S.E.T. have nearly equal frequencies. So, depending on the size of the ciphertext, some of the frequencies of the corresponding letters in the Cipher Text may switch places in the ranking of the frequencies. Therefore we may need to perform "swaps" to correct the misranking of letters that (usually) have frequencies of nearly equal value. (Can you see why this text needs so much help?)

Execute: swap(1, 'o', 'i')

swap(2, 'o', 'n')

swap(2, 'm', 'g')

swap(2, 'r', 'h')

swap(2, 'c', 'f')

swap(2, 'b', 'v')

swap(2, 'm', 'y')

swap(1, 'g', 'w')

swap(2, 'f', 'm')

Observe what’s happening with the candidate Plain Text. Once you are pleased with your results, you can save the candidate text by executing: “saveText('h:\crypto\plaintext6.txt')”

A normal decrypting attempt sequence of commands has been combined into a single m-file in AndrewPASCUtility.m, also in PASCCscript.m, which includes some swaps. (Organizing the swaps, and keeping a record of them, can reduce confusion.) These files can save some time when making repeated cracking attempts, as in the second part of this assignment.

Congrats! You have just cracked your first Cipher Text. (Even though it wasn’t encrypted in the first place, you still performed Cryptanalysis based on frequency analysis.)

The following part of this assignment is due LATER IN THE SEMESTER:

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.

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.

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

It is the copyright terms of usage for WinZip.

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!