**DS 710**

**Homework 6**

**R assignment**

1. Can we detect when a marketing campaign has been successful?

1. On homework 4, you simulated data from the TableFarm salad chain before and after the implementation of a new marketing campaign.  Read the combined data (both before and after) into R.  (You could do this by saving the data as a .csv file and using read.csv(), or by copying the data into a text file, separating the values by commas, and enclosing the data in c( … ) to make a vector.)
2. Make a scatterplot of the data.  Add a vertical line to mark the month in which the new marketing campaign began, and add a legend to your plot.
3. Make a single graph with 2 side-by-side boxplots of the revenue before and after implementing the marketing campaign.  Write a few sentences describing and comparing the boxplots, and relating them to the underlying model you used to simulate the data.
4. Based on the way you simulated the data, you know that the marketing campaign was successful; that is, the data after implementing the marketing campaign was simulated from an underlying model with a higher mean than before the marketing campaign.  However, in real life we probably wouldn’t know this.  Based on the scatterplot and boxplots, would you be confident in claiming that the marketing campaign was successful?  Why or why not?
5. Write the null and alternative hypotheses for a test of whether the marketing campaign was successful.  (I.e., whether the mean revenue with the marketing campaign is higher than the mean revenue before the marketing campaign.)
6. In a few sentences, explain why a 2-sample, 1-sided t-test is appropriate for testing the hypotheses in part e.
7. Conduct a 2-sample, 1-sided t-test in R.  Include the R output and state your conclusion in the context of the problem.

2.  In homework 5, you counted the frequencies of letters in two encrypted texts.  In this problem, you will use statistical analysis to identify the language in which the text was written, and decrypt it.

Here’s the basic idea: Suppose that the language FakeEnglish has just 2 letters, E and S, with E occurring 55% of the time and S occurring 45% of the time. Also, suppose that the language FakeLatin also has just 2 letters, A (occurring 90% of the time) and M (occurring 10% of the time). Suppose your encrypted text uses the letter V 430 times and the letter X 570 times. Which language do you think it came from?

The encrypted text probably came from FakeEnglish, because the frequencies of each letter (43% and 57%) are much closer to the frequencies in FakeEnglish than to FakeLatin. We can also say that the encrypted letter X probably represents the FakeEnglish letter E, and encrypted letter V probably represents FakeEnglish letter S. *It doesn’t matter that V and X don’t occur in FakeEnglish or FakeLatin*, because the encrypted text is *encrypted*—it uses different letters to represent each letter in the language it came from.

So, our overall strategy to identify the language of each text will be as follows:

* Put the encrypted letter frequencies in order of increasing frequency. We will guess that the most common letter in the encrypted text represents the most common letter in the real language (English or Latin), the 2nd-most common letter represents the 2nd-most common letter, and so on. This is just like our guess in the example above, that X probably represents E.
* Use a chi-squared goodness-of-fit test to test whether the frequencies in the encrypted data are consistent with the proportions in English or Latin.
  + You may need to combine some letter categories to satisfy the assumptions of the chi-squared goodness-of-fit test.

Steps a-m, below, will walk you through this process step-by-step.

1. Read the letter frequencies from encryptedA into R.  Make a barplot of the letter frequencies, with the letters listed in order of increasing frequency. One way to do this is using the following code:  (Here I’ve assumed that your columns were named “key” and “frequency”.)

encrypt\_order = order(frequency)

barplot( frequency [encrypt\_order], names.arg = key[encrypt\_order] )

Be sure you understand what this code does.

1. The file Letter Frequencies.csv contains data on the frequencies of letters in different languages.  (Source:  <http://www.sttmedia.com/characterfrequency-english> and <http://www.sttmedia.com/characterfrequency-welsh>, accessed 21 August 2015. Used by permission of Stefan Trost.)  Read these data into R.
2. We’ll start by doing a visual comparison of the distributions: In a single graphing window, display two bar plots:  A plot showing the encrypted frequencies, and a plot showing the frequencies of letters in English.  Each plot should be sorted in order of increasing frequency.  Each plot should also have a title telling whether it is from the encrypted text or from plain English.
3. Based on the **shape** of the plots, do you think it is likely that the encrypted text came from English?  Explain.

(Note: The order of the letters along the horizontal axis of each plot will be quite different, because one plot shows the frequencies in plain English, and the other shows the frequencies in the encrypted text. So, you should ignore what letter is written below each bar when answering this question. Instead, look at things like the relative frequency of the most-common letter and the second-most common.)

1. We want to conduct a hypothesis test to be more precise about whether it is plausible that the text came from English.  To do this, we will pair up each letter in the encrypted text with a letter in English, based on the order of frequency.  So, encryptedA “r” is paired with English “e”, encryptedA “c” is paired with English “t”, etc.  Then we will test whether the resulting letter frequencies plausibly come from a random sample of English words.

To pair up the letters, sort the vector of counts from the encrypted text in order of increasing frequency, and store it as a new vector. Then do the same thing with the vector of frequencies from English.

* You already sorted the counts from the encrypted text in increasing order in part a) of this problem. This problem is asking you to store the sorted vector as a variable, and also to sort the theoretical English frequencies in increasing order.

1. To pair up the letters, we need the data (the counts of letters from encryptedA.txt) and the probability model (the theoretical frequencies from Letter Frequencies.csv) to have the same number of letters. Depending on how you formatted your output from Python, your letter counts may include 20 or 26 letters. This is due to the fact that some letters did not appear in the encrypted text, so they appeared 0 times. If necessary, prepend 6 zeroes to the *count* vector to make it the same length as the theoretical frequencies:

count = c( rep(0, 6), count )

1. State the null and alternative hypotheses for a chi-squared Goodness of Fit test of this question.
2. To satisfy the assumptions of a Goodness of Fit test, we need the expected counts of each category to be greater than or equal to 5. Find the total number of letters in the encrypted text. Then multiply this number by the probabilities from Letter Frequencies.csv to get the expected counts.
   * The expected counts are **only** to help you decide how many categories to combine. You will not be using the expected counts in any other steps of the analysis.
3. Combine categories (letters) to get expected counts that are greater than or equal to 5. **For example**, if you decided to combine the first two categories (the two least-common letters), you could use the code

sortEnglish\_combined = c( sum(sortEnglish[1:2]), sortEnglish[3:26] )

Combine the same categories in the encrypted counts.

* Note, in this step you should be combining categories of the actual counts of the letters in the encrypted text, and combining categories of the frequencies (percentages) of the letters in English. The expected counts from part h are **only** to help you decide how many categories to combine.

1. Use R to conduct the chi-squared Goodness of Fit test.

* If you get the warning message, “Chi-squared approximation may be incorrect,” one of two things has happened:

1. You did not combine enough categories in step i, or
2. You are using the wrong syntax for the chi-squared Goodness of Fit test.
   * Check that the degrees of freedom (df) are 1 less than the number of categories you used. If the degrees of freedom are > 100, then double-check the syntax demonstrated in the Goodness of Fit video.

* If either of these things is true, your results will not be reliable.

1. State your conclusion in the context of the problem.

* Note that the null hypothesis is that the observed counts of the most-frequent letter, 2nd-most frequent letter, etc. are *consistent* with the theoretical frequencies. Therefore, the null hypothesis is that the text *is* an encrypted piece of writing in English.

1. Repeat steps h-k for Welsh, and then repeat for both languages for encryptedB.  (It may help to use functions or *for* loops to help you organize your code.) Fill in the p-values you get in the following table:

|  |  |  |
| --- | --- | --- |
| Text | English | Welsh |
| EncryptedA |  |  |
| EncryptedB |  |  |

1. Based on the hypothesis tests, which text do you think came from which language?
   * This should be a reasonably clear decision. If all 4 of your p-values are near , or all 4 are near 0.5, double-check your work in steps h-j.
2. Optional: Try to decrypt the English text. Simon Singh’s Black Chamber website (<http://www.simonsingh.net/The_Black_Chamber/substitutioncrackingtool.html>) will automatically substitute letters for you, so you can test different possibilities for what English plaintext letter is represented by each letter in the ciphertext. Start by substituting the letter E for the most common letter in the ciphertext. Then use frequencies of letters in the ciphertext, common patterns of letters, and experimentation to determine other substitutions.

Submit a single .docx or .pdf file to GitHub containing your R code, R output and graphs, and your written interpretations and explanations. Include your name at the top of the file. Keep all portions of a problem together (don’t put all the R code at the end of the file).