Principles of Neuroimaging

Problem Set 1

MATLAB practice. This is more of a worksheet than a problem set and should be very easy.

I am assuming that you have installed MATLAB on some computer somewhere, and have it running. In my notes, the stuff you enter is in **`bold face`**, and the stuff that MATLAB responds with is in `plain text`. The >> at the left is the MATLAB command prompt. Do not type this.

1. We will use MATLAB to create a plot of $S=e^{-te/T2}$ (which was one of the questions on the intro quiz).

A. Create two new variables, T2 and te, and set them to vales of 100 and 10 respectively, At the command line, type (typing <return> at the end of each command line):

```
>> T2 = 100
T2 =
     100
>> te = 10
te =
     10
```

Assuming that $S = e^{-te/T2,}$ use MATLAB to find S:

```
>> S=exp(-te/T2)
```

What is the value of S?

B. Create an array of values for te:

```
>> te = 0:5:250;
```

This command creates a one-dimensional array (vector) with values of te from 0 to 250 in increments of 5. The semicolon at the end of the line suppresses the MATLAB output, which otherwise would have been a list of all of the values of te. If you want to see that list, just type:

```
>> te
```

C. Create an array of values for S according to the equation:

```
>> S=exp(-te ./ T2);
```

The notation **`./`** tells MATLAB to divide each element in the te vector by the constant, T2, creating the S vector.

What are the values of S?

D. Make a plot of S as a function of te:

```
>> plot(te, S)
```

The plot command makes a 2D plot, in a new window, of the second variable as a function of the first. If only one variable is listed (e.g., **`plot(S)`**) you just get a plot of that array in

evenly spaced steps. An interesting feature of MATLAB (as opposed to other languages, such as C) is that the arrays typically start from 1 instead of 0. This means that **`plot(S)`** has x values that start from 1. Turn in a copy of the plot of S vs. te. Notice that the **`;`** is not needed at the end of the line, because the plot command does not produce a text output.

E. Make a plot of a Gaussian curve: $G = \exp(-x^2)$, with x ranging from -3 to 3 in steps of 0.1. The MATLAB notation for x^2 is **`x^2.`**

2. The t-statistic is used to determine if the mean value of two samples is different. The t-statistic is defined as:

$$t = (\bar{x}_2 - \bar{x}_1)/\sqrt{(s_1^2 + s_2^2)/N}$$

where x_1 and x_2 are the sampled values of the data from two populations and s is the standard deviation of each sample (in this case, we assume explicitly that the number of samples is the same for both populations. If *t* is large compared to the variance, the populations are most likely to have a different mean.

We will use MATLAB to show that the t-statistic (at least qualitatively) has the same distribution even if the distribution of the variables being sampled does not.

First, use the built-in random function to select numbers from your choice of distributions.

A. Create a vector of 1000 normally distributed numbers with a mean of 0 and a standard deviation of 1.5.
```
>> Norm=random('norm',0,1.5,[1 1000]);
```

The string, **`'norm'`**, tells MATLAB to select random numbers from a normal distribution. The second argument is the mean and the third argument is the standard deviation.

B. Plot these numbers as a histogram:
```
>> hist(Norm)
```

C. The default histogram has ten bins. Use the following command to make a histogram with 50 bins
```
>> hist(Norm,50)
```

Repeat these steps, but create an array with the name **Unif** with uniformly-distributed numbers numbers (**`'unif'`** instead of **`'norm'`**). Create an array, **Gamma**, of random numbers from the gamma distribution (**`'gam'`** instead of **`'norm'`**). Plot histograms of these distributions. Note that the uniform distribution is described by two parameters: the upper and lower bounds, so the command,

```
>> Unif=random('unif',0,5,[1 1000]);
```

Creates a collection of random numbers from 0 to 5, whose mean is 2.5. The Gamma distribution also accepts two parameters, whose product is the average value. For the time being, use **`Gamma=random('gam',2,10,[1 1000])`**.

To find more about the random command, you can type:

```
>> help random
```

Or you can use the help window.

We will be performing a Monte Carlo experiment, where we simply run the analysis 10000 times to look at the values of t when the data are sampled from entirely different underlying distributions.

D. Create 10000 different arrays of 1000 normally-distributed random numbers with a mean of ten and a standard deviation of 5 (be sure to type a semi-colon at the end of the line or your screen will become cluttered with random numbers)

```
>> NA = random('norm',10,5,10000,1000);
```

Depending on how MATLAB is set up on your computer, there should be a window or window pane that states:

```
NA    <10000x1000 double>
```

To show that the NA array exists as an array of 10000 rows and 1000 columns.

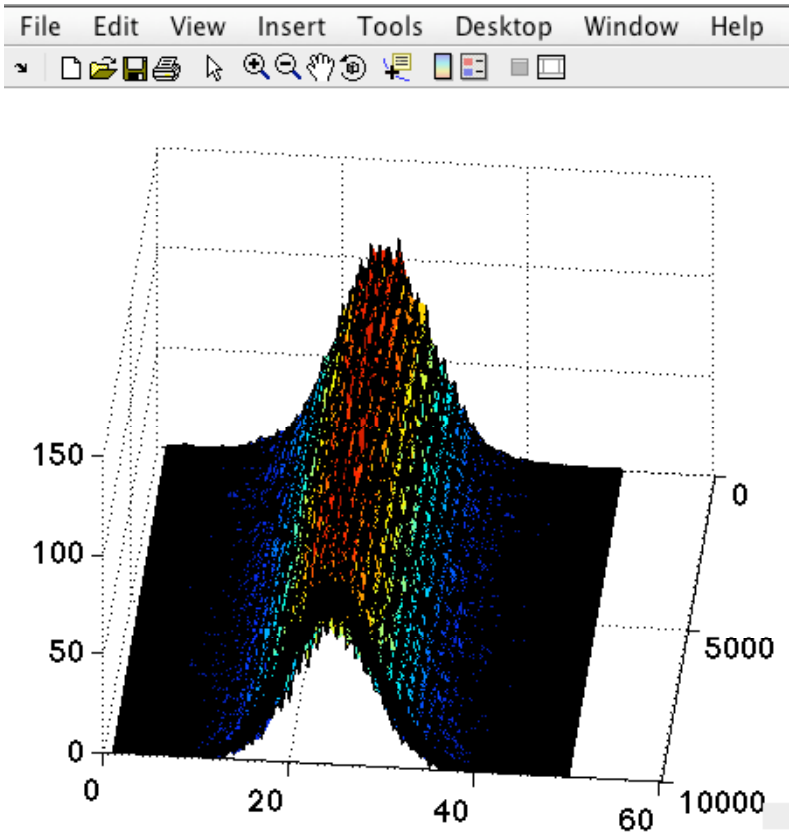
E. Make a series of 10000 histograms each with 50 bins, one for each of the 10000 rows of normally-distributed numbers

```
>> NAH = hist(NA',50);
```

The ' operator creates the matrix transpose, flipping the rows and columns of NA. We do this because the hist command otherwise would form a histogram of all of the column values rather than the row values. You will see that MATLAB creates a variable, NAH, with dimensions 50x10000. You can use the surf command to see this as a surface plot:

```
>> surf(NAH);
```

select the  icon in the graphing window. Click anywhere on the surface plot and drag to see the plot from different views.



Notice when you do this, that the x axis is no longer labeled correctly, as it reverts to 50 values from 1 to 51. NAH is a 2 dimensional array that is easily represented as an image. MATLAB can present this as an image with the command:

```
>> image(NAH);
```

The range of values for NAH is rather large. In cases like this, taking the logarithm of these values may make things easier to see:

```
>> image(log(NAH));
```

F. Create another array, NB, with normally distributed numbers having a mean of **10.5** and a standard deviation of **5**.

G. Find the mean and standard deviation of each row in NA and NB:

```
>> NAM = mean(NA,2);
>> NBM = mean(NB,2);
>> NAstd = std(NA,1,2);
>> NBstd = std(NB,1,2);
```

In the std command, the first argument is the name of the input array, the second is a flag with a value of 1 or 0 (0 means divide the deviations by N, whereas 1 means divide the deviations by N-1). The third argument is the dimension on which to create the standard deviations, 2 meaning the standard deviation of each row.

Plot histograms of NAM, NBM, NAstd and NBstd. Notice that there is a range of both standard deviations and means.

H. Calculate 10,000 t statistics of the difference of means:

```
>> tN = (NBM - NAM) ./ sqrt((NAstd .^2 + NBstd .^2)/1000);
```

The operator **.^2** tells MATLAB to take the square of each element in the preceding array. **NAstd .^2** means the square of each element in the vector, NAstd.

I. Make and send me a histogram of the values of tN.

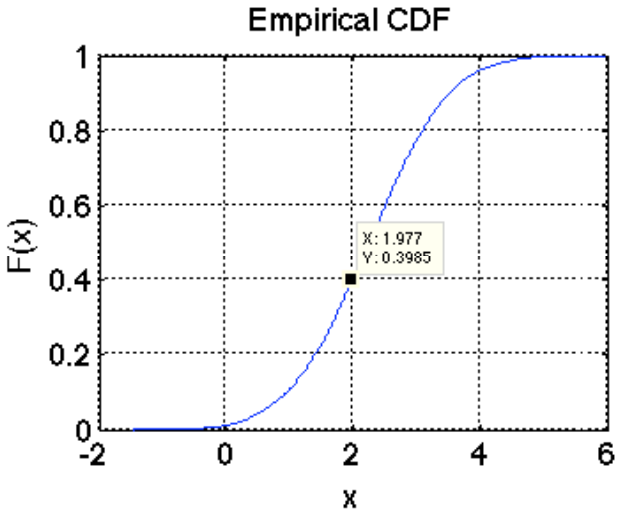
We know by design that the nominal value of *t* should be:

$$t = (\bar{x}_2 - \bar{x}_1)/\sqrt{(s_1^2 + s_2^2)/N}$$
$$= (10.5 - 10)/\sqrt{50/1000}$$
$$\approx 2.24$$

What is the mean of your experimentally determined tN? What is the standard deviation of tN? Approximately what fraction of the time would your experimentally measured t statistic be greater than 3.24?

The *degrees of freedom* for this test is N-2, because we have computed two means already. It is known that when there are 30 or more degrees of freedom, the distribution of the *t*-statistic is extremely close to the normal distribution. In this case, for example, there would be about a 5% probability of the *t*-statistic exceeding 1.96 by chance. Our *false positive* or *type I* error rate is the likelihood of exceeding the observed *t* value by chance. We say that $\alpha = 0.05$

J. To get an intuition for the *type II* error rate, consider this: we reject the null hypothesis with 95% confidence that the means are the same, if we get a t statistic of 1.96 or more. The *false negative rate* is the likelihood of us getting a *t* of less than 1.96 by chance when the means are different. There is a nice MATLAB function to get at this: **cdfplot(tN)** plots the "cumulative density function" of tN. This is the fraction of the area under histogram of tN at any value of tN.



The CDF of tN, as plotted above, shows that about 40% of the observed *t* values will be less than 1.96. This implies that 40% of the time, our test will come out negative, even though we know that the means of our two samples differ by 5%. The term, β, is the false negative rate. The *power* of a test is defined as 1- β. In our case, the power is about 0.6 when the value of α is 5%. Your plots should look very slightly different. Because we performed such a very large number of tests (10,000), the curves should be pretty close.

This lopsided relationship between type I and type II error rates is the norm. To lower the false negative rate to say, 5%, you must set an extremely high type I rate with many false positives.

K. Make 10000 sets of 1000 uniformly distributed random numbers ranging from 0.5 to 20.5, and from 0 to 20. Make a histogram of the t-statistics for the difference in mean of these two distributions. Send these to me.

L. Make 10000 sets of 1000 gamma-distributed random numbers with gamma parameters of 2 and 5, and gamma parameters of 2 and 5.25. Make a histogram of the t-statistics for the difference in mean of these two distributions. Send these to me.

With all of these, you should have learned some key principles in MATLAB. I will hand out a solution set in the form of a 'script' which essentially captures all of the individual commands you have used into a single file that MATLAB can run all at once.

To think about:

- Under what conditions can you have both a very low false negative rate and a low false positive rate?
- What is it about the *t* statistic that seems to make it immune to assumptions about the underlying differences in the distribution of the two populations?
- What is a good number of Monte Carlo tests to run in order to get an estimate of the CDF?