1 Rec 1: Random Data Simulation and Analysis

Directions: Your instructor will spend the the first 40 minutes of the recitation period working some review problems and going over one or more Matlab experiments in the following. During the last 10 minutes of recitation, your proctor will give you a "Lab Form" that your recitation team completes, signs, and turns in. See the last page for an indication of what you will be asked to do on the Lab Form.

Due to time limitations, only a part of the following can be covered during the recitation period. However, you might want in the future to try some of the uncovered experiments on your own. They could give skills useful on some future homework problems and could lend insight into your understanding of the course from an experimental point of view.

This Week's Topics.

- Matlab review
- Matlab functions `rand`, `hist`, `mean`, `median`
- Simulating Experiments and Estimating Probabilities

1.1 Exp 1: Matlab Review

You have had exposure to Matlab in EE 3015. This first experiment reviews some of the things you may have learned. If you already feel comfortable with your Matlab ability, you can skip the experiment and go to Experiment 2.

1.1.1 Use of the Dot

- Execute the command:

  ```
  [2 4 6].*[7 8 9]
  ```

  Do you understand what you see on the screen?

- Execute the command:

  ```
  [2 4 6].^2
  ```

  Do you understand what you see on the screen?

- Execute the command:

  ```
  2.^[2 4 6]
  ```

  Do you understand what you see on the screen?

1.1.2 Finding and Counting Things

- Execute the commands:

  ```
  x=[2 3 1 1 3 4 5 6 3 4];
  (x==3)
  ```

  Do you understand what you see on the screen?

- Execute the command:

  ```
  find(x==3)
  ```

  Do you understand what you see on the screen?

- Execute the command:

  ```
  sum(x==3)
  ```

  Do you understand what you see on the screen?

1.1.3 Differentiator and Integrator

- Execute the commands:

  ```
  x = [2 3 1 1 3 4 5 6 3 4];
  x(2:10)-x(1:9)
  ```

  Do you understand what you see on the screen?

- Execute the command:

  ```
  cumsum(x)
  ```

  Do you understand what you see on the screen?

1.1.4 Rows and Columns of Matrix

- Execute the command:

  ```
  M = [1 2 3; 4 5 6; 7 8 9]
  ```

  Do you understand what you see on the screen?

- Execute the command:

  ```
  M(2,:)
  ```

  Do you understand what you see on the screen?

- Execute the command:

  ```
  M(:,3)
  ```

  Do you understand what you see on the screen?

1.1.5 Manipulating Lists

- Execute the commands:

  ```
  2:8
  10:-1:1
  ```

  Do you understand everything you see on the screen?

- Execute the commands:

  ```
  y=[50 9 45 8 40 7 35 6 30 5];
  y(10:-1:1)
  ```

  Do you understand what you see on the screen?

- Execute the commands:

  ```
  y=[50 9 45 8 40 7 35 6 30 5];
  t=1:5;
  y(2*t)
  y(2*t-1)
  Q(1,1:5)=y(2*t);
  Q(2,1:5)=y(2*t-1);
  Q
  ```

  Do you understand everything you see on the screen?

1.2 Exp 2: Matlab function "`rand`"

- Execute the command:

  ```
  rand(1,10)
  ```

  You will see 10 real numbers on your screen chosen (pseudo)randomly from the interval $[0, 1]$.

- Execute the following command again:

  ```
  rand(1,10)
  ```

  The 10 numbers you get will probably all be different from before. This is as it should be: the outcomes are random (i.e., not known in advance).

- Execute the command:

  ```
  rand(3,3)
  ```

Then execute the command again to see if you get something else.

- Here is a test of your Matlab skills. Using a "Matlab for loop", generate the random 3×3 matrix 1000 times and then average up the results to see what you get. Talk to your proctor if you get stuck. You will get something that looks approximately like

```
0.5081    0.5149    0.5171
0.5016    0.5108    0.5009
0.4972    0.4991    0.4897
```

1.3 Exp 3: Mean and Median

Suppose you have a vector whose components are real numbers. Applying the Matlab function "`mean`" to the vector computes the average of all of the components of the vector. Applying the Matlab function `median` to the vector gives a real number x such that there are just as many components of the vector $\leq x$ as $\geq x$. The mean and the median are not necessarily the same thing.

- Run the Matlab script:

```
u=-log(rand(1,10))
mean(u)
median(u)
```

See if you understand the results by doing the following. First, use your calculator to average up the 10 components of vector `u` that you see on your screen; see if this value is the same as `mean(u)`. Then, execute the commands

```
sum(u<=median(u))
sum(u>=median(u))
```

to see if the median value is indeed the "middle value" of the 10 components.

1.4 Exp 4: Matlab function "`hist`"

This experiment teaches you a little bit about Matlab's histogram function "`hist`". Generate a vector x of 100 random integers between 1 and 100 by executing the following line of Matlab code:

```
x=ceil(100*rand(1,100))
```

You will see the 100 entries of x before you on the computer screen. Some integers between 1 and 100 will appear in the data sequence x and some will not. Stare at the 100 entries of x on the computer screen until you find an integer between 1 and 100 that *does not appear* in x. Then, stare at the 100 entries of x until you find an integer between 1 and 100 that *appears at least twice* in x. The matlab function "`hist`" allows you to answer questions of this sort much more painlessly. Now, execute the following two lines of Matlab code:

```
t=1:100;
hist(x,t)
```

You should see a plot appear on your Matlab screen. This plot is the *histogram* of x. For each integer between 1 and 100, the vertical height of the plot at that integer is the number of times that integer appears in x. Where the plot is zero, you have integers between 1 and 100 not appearing in x. Where the plot is at least two, you have integers between 1 and 100 appearing at least twice in x. It is a little hard to see from the horizontal axis which integers appear in x and which do not. Execute the following line of Matlab code:

```
f=hist(x,t)
```

The 100 entries of the vector f, printed out on your screen, give you the vertical heights of the histogram plot, which are the frequencies with which each integer 1 to 100 appear in x. We can use the frequency vector f to fulfill more easily the search tasks we did earlier by a laborious eyeball search of the computer screen. For example, execute the following line of Matlab code to see which integers between 1 and 100 do not appear in x:

```
find(f==0)
```

Then, execute the following line of Matlab code to see which integers between 1 and 100 appear in x at least twice:

```
find(f>=2)
```

Finish Experiment 4 by trying to answer the following questions:

- What line of Matlab code would you use to find those integers between 1 and 100 that appear in x at least once but less than 4 times?
- Which integers between 1 and 100 appear in x the most times? How many times does each of them appear in x?

1.5 Exp 5: Simulating Equiprobable Random Experiment

In this experiment, you will learn how to use the Matlab function `rand` to simulate the outcome of a random experiment having equally likely outcomes, and how the use the Matlab function "`mean`" to see whether the simulation is doing what it is supposed to do. First, we try to simulate the following random experiment, which defines a certain random variable X based on a coin flip:

Flip a fair coin: if heads, take $X = 1$; if tails, take $X = -1$.

We want to do a simulation in which we pseudorandomly generate 1 about half the time, and -1 the remaining times. If we were to form the pseudorandom number `ceil(2*rand(1,1))`, we would obtain 1 about half the time and 2 the remaining times. Then, we just have to do a linear transformation that will transform $1, 2$ into $-1, 1$, respectively. The random experiment is simulated by executing the Matlab script

```
u=ceil(2*rand(1,1));
x=a*u+b;
```

with the constants a and b chosen so that $au + b$ evaluated at $u = 1$ is -1 and $au + b$ evaluated at $u = 2$ is 1. A little bit of work shows that $a = 2$ and $b = -3$. (Do this.)

Our conclusion is that the Matlab one-liner

```
x=2*ceil(2*rand(1,n))-3;
```

should create a vector x whose entries simulate n independent trials in which the random experiment is performed and the value of X is noted. Let's do some data analysis on x for large number of trials n, to see whether our simulation seems to be working. Execute the one-liner

```
x=2*ceil(2*rand(1,100))-3
```

The 100 entries of x printed out simulate 100 observations of the random variable X. About 50 of the entries should be $= +1$ and 50 of the entries should be -1. Execute the lines

```
sum(x==1)
sum(x==-1)
```

to obtain the number of times $1, -1$ actually occur in vector x, respectively. Is each value near 50?

Do a plot of x as a bar chart by executing the line

```
bar(1:100,x)
```

Do about half the bars in the bar chart point upward, and about half downward?

Now, we obtain more convincing evidence. Execute the lines

```
clear;
x=2*ceil(2*rand(1,10000))-3;
```

The vector x in Matlab memory simulates 10000 independent observations of the random variable X. The probabilities $P[X = 1]$ and $P[X = -1]$ are both 1/2. If our simulation is working properly, the empirical frequency with which 1 appears in vector x should be about 1/2 (this frequency is the number of entries of x equal to 1, divided by 10000, the length of x—in other words, we count the number of trials in which event $\{X = 1\}$ occurred and divide by the total number of trials). Similarly, the empirical frequency with which -1 appears in x should also be about 1/2. These two empirical frequencies can be computed using the Matlab "`mean`" function, as follows:

```
mean(x==1)
mean(x==-1)
```

Execute these two lines to see if the two frequencies do indeed approximate the probabilities $P[X = 1] = 1/2$ and $P[X = -1] = -1/2$, respectively. Now average the empirical frequencies over 10 runs as follows to see if you get even better approximations to $P[X = 1]$ and $P[X = -1]$:

```
for i=1:10
x=2*ceil(2*rand(1,10000))-3;
empfreq1(i)=mean(x==1);
empfreq2(i)=mean(x==-1);
end
mean(empfreq1)
mean(empfreq2)
```

It is highly likely that when you round off the approximations you see on your screen to two decimal places, you will obtain 0.50 and 0.50. Do you?

1.6 Exp 6: Simulating Fair Die Tosses

We consider another experiment with equally likely outcomes. Toss a fair die, see what number comes up, and let the random variable X be this number. The RV X has 6 equally likely values $1, 2, 3, 4, 5, 6$. The following line of Matlab code simulates n observations of the random variable X:

```
x=ceil(6*rand(1,n));
```

Generate vector x of length 10000 in this way. For each $i = 1, 2, 3, 4, 5, 6$, count how many entries of vector x are equal to i with the line of Matlab code

```
 sum(x==i)
```

Then, approximate the probabilities

$$P[X = i] = 1/6, \quad i = 1, 2, 3, 4, 5, 6 \tag{1}$$

by means of the empirical frequencies

```
 sum(x==i)/10000
```

Equivalently, you can obtain this same empirical frequency with the line

```
 mean(x==i)
```

computed for $i = 1, 2, 3, 4, 5, 6$. Are the empirical frequencies all around $1/6$? If you have time, average the empirical frequencies from 10 runs of 10000 observations each, to see if the approximations to the probabilities (1) get even better.

Tossing Two Fair Dies: Consider 10000 trials of the experiment in which you toss a pair of fair dice and record the total. Run the following Matlab code, which approximates the probability that you get a total of 7:

```
    x=ceil(6*rand(1,10000)); %10000 flips of first die
    y=ceil(6*rand(1,10000)); %10000 flips of second die
    z=x+y; %The 10000 totals
    mean(z==7) %The estimate of the prob that the total is 7
```

1.7 Exp 7: Simulating Three Fair Coin Tosses

Consider the random experiment in which three fair coins (coin 1, coin 2, and coin 3) are each tossed once and the results recorded. Let X be the random variable equal to the total number of heads on the three coins. The values $0, 1, 2, 3$ of X are not equally likely. In fact,

$$P[X = 0] = P[X = 3] = 1/8, \quad P[X = 1] = P[X = 2] = 3/8.$$

However, X is based on an experiment with equally likely outcomes, so we can easily simulate observations of X using the ideas from Experiments 5-6. First, create the following three vectors which simulate the results of tosses 1,2,3, respectively.

```
x1=ceil(2*rand(1,10000))-1;
x2=ceil(2*rand(1,10000))-1;
x3=ceil(2*rand(1,10000))-1;
```

The entries of vectors $x1, x2, x3$ are each $0, 1$. For example, if entry i of vector $x1$ is 1, this means that the i-th toss of coin 1 came up heads; if entry i of vector $x1$ is 0, this means that the i-th toss of coin 1 came up tails. We simulate observations of the value of X by adding up the vectors $x1, x2, x3$:

```
x=x1+x2+x3;
```

(Do this.) Compute

```
mean(x==0)
mean(x==1)
mean(x==2)
mean(x==3)
```

What is your conclusion?

Here is a quicker way to obtain the empirical frequencies. Execute the commands

```
f=hist(x,0:3);
f/10000
```

Do the four components of the normalized frequency vector $f/10000$ on your screen coincide with your previous computations of the empirical frequencies? Now do the bar plot

```
bar(0:3,f/10000)
```

The plot you see on the screen should be an approximation to the plot of the probability mass function (PMF) of the random variable X.

1.8 Exp 8: Simulating Nonequiprobable Experiment

In this experiment, you learn how to simulate observations of a random variable whose values are not equally likely. Let X be a random variable taking values -1.42, 0.53, 1.25, and 2.31 with probabilities 0.1, 0.2, 0.3, 0.4, respectively. Since the values of X are not equally likely, we have to simulate X by a different method than used previously. Conceptually, we can partition the unit interval $[0, 1]$ into the four subintervals

$$[0, 0.1], \quad [0.1, 0.3], \quad [0.3, 0.6], \quad [0.6, 1]$$

of lengths $0.1, 0.2, 0.3, 0.4$, respectively. We then generate a pseudorandom number U in $[0, 1]$ using `rand`. Finally, we declare that $X = -1.42$, $X = 0.53$, $X = 1.25$, or $X = 2.31$ depending upon whether U is in the first, second, third, or fourth subinterval, respectively. The following Matlab script creates 10000 pseudorandom simulated observations of X:

```
for i=1:10000
u=rand(1,1);
if u<0.1
x(i)=-1.42;
elseif u<0.3
x(i)=0.53;
elseif u<0.6
x(i)=1.25;
else
x(i)=2.31;
end
end
```

Do the computations

```
mean(x==-1.42)
mean(x==0.53)
mean(x==1.25)
mean(x==2.31)
```

These empirical frequencies will probably yield accurate approximations of the respective probabilities $0.1, 0.2, 0.3, 0.4$ to the nearest one-tenth. Do they? If you have time before the final 10 minutes, re-compute each of the four empirical frequencies over 10 runs and average to see if you get approximations to the respective probabilities $0.1, 0.2, 0.3, 0.4$ accurate to two decimal places.

1.9 Exercises

Write programs to simulate (a) the total on toss of three fair dies, and (b) the total number of heads on toss of four fair coins. Use the function `hist` to interpret the results, as we did in the last part of Experiment 7.

EE 3025 S2007 Recitation 1 Lab Form

Name and Student Number of Team Member 1:

Name and Student Number of Team Member 2:

Name and Student Number of Team Member 3:

****

Your lab form will look like this. Your proctor will give you hard copy of the lab form which tells you what you are to do. What you are to do will vary slightly from section to section. However, I can tell you in general terms what you will be doing: You will be asked to use Matlab code that will simulate a certain number of independent trials of the experiment "Flip k fair coins and count the total number of heads." (You will be told how many trials and what k is.) Then you will be asked to use Matlab to compute histogram estimates for the probabilities of various outcomes. Before coming to class, it would be helpful for you to study Experiment 7, particularly the bottom half of page 8.