EE 303 Lab 1: Counting, Sampling, and Games in MATLAB

1 Introduction

In this lab, we will study counting experiments and demonstrate how they relate to random sampling from a set. These ideas will be used to examine some of the games that people play. As with most problems in engineering, you will be required to do some mathematical reasoning and then verify your results using a software tool. For this class, that software tool will be MATLAB.

2 Counting and Sampling

The purpose of counting in probability theory is to determine the numbers of ways an experiment can turn out. For example, in a 42-number lottery, there are 5,245,768 ways for the State to pick the 6 winning numbers from a hat containing 42. We shall prove this.

If you think about it, the State draws little plastic markers *labelled* with numbers, not numbers. That is, the State uses integers to code the objects of the experiment. This is a matter of convenience. Even football teams use numbers on jerseys to code their human participants. As we shall see, the integer coding of objects in an experiment simplifies the description. It will not matter whether the objects are people or playing cards.

Because there is a game to illustrate every counting principle, we can study the classical counting formulas by studying games.

2.1 Sampling With and Without Replacement

Consider the drawing of numbers from a hat containing the numbers 1 through n. The first draw produces a number of which there are n possible. If the drawn number is placed back into the hat and a second number is drawn (out of a possible n numbers), then there are n^2 possible ways that the experiment can turn out. If this number is placed back into the hat and the experiment continues until r numbers have been drawn, then the number of possible outcomes for the experiment is

$$N = n^r. \tag{1}$$

We call this experiment "sampling with replacement."

Now suppose r numbers are drawn from a hat without replacement. The first draw produces one of n possible outcomes, the second produces one of $n-1$ possible outcomes, and so on. The number of possible outcomes for r such draws is

$$N = (n)(n-1)\cdots(n-r+1). \tag{2}$$

We call this experiment "sampling without replacement," and we call the number of possible outcomes "r permutations of n." For convenience, we will use the following notation for this number

$$(n)_r = (n)(n-1)\cdots(n-r+1) = \frac{n!}{(n-r)!}. \tag{3}$$

Note that "n permutations of n" is "n factorial," namely $(n)_n = n!$.

If we again consider the experiment where we draw numbers from a hat containing n numbers, then permutations depend on the order of the numbers drawn. For example, if there were numbers 1 through 8 in the hat and we drew 3 numbers (without replacement), then the sequence of draws that produces (3,6,4) is different than the sequence of draws that produces (6,4,3), even though they share the same set of numbers. For this example, there are $(8)_3 = 336$ ways to pick 3 numbers. For each set of 3 numbers, there are 3! permutations of each set. In the example above, the $3! = 6$ permutations of (3,6,4) are

```
3 4 6
3 6 4
4 3 6
4 6 3
6 3 4
6 4 3.
```

If we are interested in the number of 3 number combinations that may be drawn from the 8 number hat, then we need to remove the multiplicity of each combination, namely $\frac{(8)_3}{3!} = 56$. We call this the "number of combinations of eight things taken three at a time" or "the number of ways to draw three objects from eight" and denote it $\binom{8}{3}$. The general formula is

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} = \frac{(n)_k}{k!}, \tag{4}$$

which is read "n choose k". It is the number of subsets of size k within a set of size n.

3 MATLAB Essentials

This section discusses both the basic MATLAB commands that are needed for this lab and the set of functions that will be required for this lab. If you are already familiar with MATLAB, you may skip the second subsection.

3.1 Basic MATLAB Operations

For this lab, you will be working primarily with vectors. For example, if you think that the top three finishers in a horse race are going to be horses 5, 2, and 7, then you could create the vector

```
>> my_picks = [5 2 7];
```

in MATLAB, where the semicolon at the end is to suppress the output from being echoed to the screen. If you want to create a vector that has evenly spaced elements, you can use the colon operator. For example, if you want numbers starting at 0 that are spaced 0.4 apart and are less than or equal to 1.5, you would type and see the following

```
>> 0:0.4:1.5

ans =

         0    0.4000    0.8000    1.2000
```

Notice that the upper bound of 1.5 does not need to appear in the answer. If the increment number (0.4 in the above example) is not included, then it is implied to be 1. You will use this feature to generate a sample index for plotting.

There are two ways to enter numbers in MATLAB. The standard way is to enter the number (e.g. 298.24). The other way is to use scientific notation. Here, the MATLAB number `2.9824e2` is $2.9824 \times 10^2 = 294.24$. Note that this number may also be entered as `.29824e3`, `29.824e1`, `298.24e0`, `2982.4e-1`, etc.

Logical operators and functions will be used frequently in this lab. With all functions and operators, you can use the MATLAB commands `help` and `type` to learn more about various commands (e.g., try typing `help stem` at the MATLAB command prompt `>>`).

One of the most fundamental operators is the "equal to" operator, namely `==`. This operator may be used to either check the element-by-element equivalence of two matrices, or to check every element of a matrix for a single value. For an example of the first case, consider comparing the vector `my_picks` to the vector `[4 2 7]`. The first element is different (i.e. $4 \neq 5$), but the last two elements are the same. In MATLAB, this can be coded as

```
>> act_top3 = [4 2 7];
>> my_matches = (act_top3 == my_picks)

my_matches =

     0     1     1
```

Notice that MATLAB uses a numeric zero to represent a Boolean FALSE and a numeric one to represent a Boolean TRUE. In this example, the variable `act_top3` could represent the actual top 3 finishing horses in a horse race, which means that `my_picks` matched only two out of the top 3 horses.

For an example of the second case of the "equal to" operator, consider trying to find which elements of `my_picks` are equal to `7`. In MATLAB, this can be coded as

```
>> is_this_a_7 = (my_picks == 7)

is_this_a_7 =

     0     0     1
```

This result is interpreted as the first two elements of `my_picks` are not equal to `7` and the last element is equal to `7`.

The dual the "equal to" operator is the "not equal to" operator, namely `~=`. Try replacing the "equal to" operator with the "not equal to" operator in the examples above. As we will show later, the "not" operator, namely `~`, may also be used to negate logical functions.

The "equal to" and "not equal to" operators are often used in conjunction with other MATLAB functions such as `find`, `any` or `all`. The function `find` will return the indices of the nonzero numbers in the matrix passed to it. An example of this is

```
>> which_match = find(act_top3 == my_picks)

which_match =

     2     3
```

which returns the indices of the matching values. Recall that the command `act_top3 == my_picks` returns the vector `[0 1 1]`. If all of the input numbers are zero, the result is an empty matrix. An example of this is

```
find(my_picks == 4)

ans =

   Empty matrix: 1-by-0
```

Empty matrices can not be added to other numbers or matrices. For example try typing

```
find(my_picks == 4) + 2
```

in MATLAB. To deal with these, we can use the built-in MATLAB function `isempty`, which will return a 1 if the matrix is empty or 0 if the matrix is not empty. To get the opposite result you can use the "not" operator, namely `~isempty`. You will see examples of this in the programs that you write for this lab.

The functions `any` and `all` return either a `0` or `1` if any or all numbers in a matrix are nonzero, respectively. Try applying these functions to the MATLAB expressions given above.

Logical operators are commonly used to direct `if` statements. In MATLAB, the usage is

```
if expression
        block of code
end
```

When the result of the expression is 1 (which is supposed to represent a Boolean TRUE), or if every number in the expression is non-zero, then the "block of code" will run. There are also `if-else` statements that can execute a block of code when the expression is FALSE. To learn more about this, type `help if` at the MATLAB prompt.

The last topic in this section is on structured variables, which are often referred to as structs in programming. Structure variables allow multiple variables of various types to be stored within one struct. An example of this is

```
student.name = 'Mike';
student.age = 29;
student.school = 'Colorado State University';
```

Here, the fields `name` and `school` are character strings and the field `age` is a numeric entry. To see all of the data fields, type `student` at the MATLAB command prompt. To access a specific variable, like the name of the school, type `student.school`. For more information on struct variables, type `help struct` at the MATLAB command prompt.

3.2 Counting and Sampling in MATLAB

The are four main functions that you will need for this lab. The first two are built-in MATLAB functions for doing permutations and combinations, and the last two are homebrewed functions that you can use to perform sampling without replacement and to count the number of element matches between two vectors.

The permutation function in MATLAB is `perms`. You pass it a vector and it returns all of the permutations of the vector. To see this try typing `perms(1:4)` or even `perms('abcd')`. If you do not like the order of the results, you can try either `perms(4:-1:1)` or `sortrows(perms(1:4))`.

There is one function that both computes the value of $\binom{n}{k}$ and returns all of the combinations of length k, namely `nchoosek(n,k)`. This function takes two inputs. When the first input is a single number, the output is the numerical value of $\binom{n}{k}$. When the first input is a vector of length n, then `nchoosek` returns the $\binom{n}{k}$ combinations of the n numbers 1 through n, taken k at a time . To see this, type `nchoosek(5,3)` to see that there are 10 combinations of 5 elements taken 3 at a time. To see the actual combinations, type `nchoosek(1:5,3)`.

3.2.1 Sampling without Replacement

For sampling without replacement, download the function `pick_nums.m` from the class webpage. This function randomly draws without replacement k numbers from a set of n numbers. To see this, try typing `pick_nums(1:12,3)` or `pick_nums(1:42,6)` many times over. NB: In MATLAB, you can press the up arrow to scroll through previously entered commands.

The last function is `count_matches.m`, which takes in two vectors and counts the number of elements they have in common without regard for the position of the elements in their respective vectors. For example, the vector `[1 4 7]` has 2 matches with the vector `[7 3 4]`. The two vectors do not need to be the same length. An example of this that uses `count_matches` is

```
% Keno example
State = pick_nums(1:60,20);   % pick 20 numbers
player = pick_nums(1:60,10);  % pick 10 numbers
num_matches = count_matches(State,player)
```

When this function is called with only one return variable (`num_matches` above), it only returns the number of matches. If you want to see which values are actually matched, you can replace the last line with

```
[num_matches values_matched] = count_matches(State,player)
```

You are encouraged to use `help` and `type` on all these functions to learn more about them.

4 The Games People Play

In this section, we will use games to illustrate some counting principles.

4.1 Lotto

Lotto is a game where each player chooses K unique numbers out of a possible N numbers until the State closes the game. At that point, the State picks its own K unique numbers and then pays each player based on the number of correctly matched numbers.

In a hypothetical Lotto game, let there be $N = 42$ possible numbers. Each player (and hence the State) chooses $K = 6$ numbers. This means that there are $\binom{42}{6} = 5,245,786$ possible ways to choose 6 numbers from 42 possible numbers. Therefore, the odds of matching all 6 numbers is 5,245,786 to 1. But what are the odds of matching, say, only 4 numbers. To see this, break the 42 possible numbers into a desired set of 6 numbers and the unwanted set of 36 numbers. If a player matches 4 of the desired numbers, then the player also matches 2 numbers from the unwanted set. Therefore, the total number of combinations of 4 desired numbers and 2 unwanted numbers is $\binom{6}{4}\binom{36}{2} = 9450$. This means that the odds of matching exactly 4 numbers is 5,245,786 to 9450, which is approximately 555 to 1. In general, there are

$$\binom{6}{k}\binom{42-6}{6-k} \tag{5}$$

ways to match $k = 0, 1, \ldots, 6$ numbers in a (42,6) Lotto. The odds of matching k numbers is just the number of possible outcomes vs. the number of outcomes that produce k matches, namely $\binom{42}{6}$ to $\binom{6}{k}\binom{42-6}{6-k}$. The "inverse of odds" is the *probability* of k matches

$$P(k; 42, 6) = \frac{\binom{6}{k}\binom{42-6}{6-k}}{\binom{42}{6}}; \quad k = 0, 1 \ldots, 6. \tag{6}$$

There are two functions available on the class webpage for simulating a Lotto game. To utilize these functions, download all of the MATLAB files for this lab from the class webpage.

The first one, namely `lotto_game.m`, plays a Lotto game with M players and returns the draw that the state made along with player draws and their respective matches. To use this function, you have define three values: N =total possible numbers, k =number of draws for each player, and M =number of players. For example, if you type

```
>> lotto_game(42,6,10)
```

the result is

```
State =

    35     2    25    41    12    37

player_data =

     5     3    12    40     4    14     1
    42     9    21    13    29    40     0
    33    28     6     4     1    14     0
    35    42     1    33    25    22     2
    11    35    12    32    28     8     2
    26    25    29     8    28     7     1
    23    27    30    29    38     1     0
    14    33    13    40    28     3     0
     3     1    12    24    21    32     1
    25    21     3     9    17    13     1
```

NB: In the `player_data` matrix, the last column contains the number of matches that the player got. To see a plot for a game like above, you can type:

```
>> plotstats(42,6,10)
```

The second function is `lotto_histo.m`, which plays a Lotto game with M players and plots the measured and theoretical statistics for the game. To see this, type the following:

```
>> [est_pmf act_pmf] = lotto_histo(42,6,10)
```

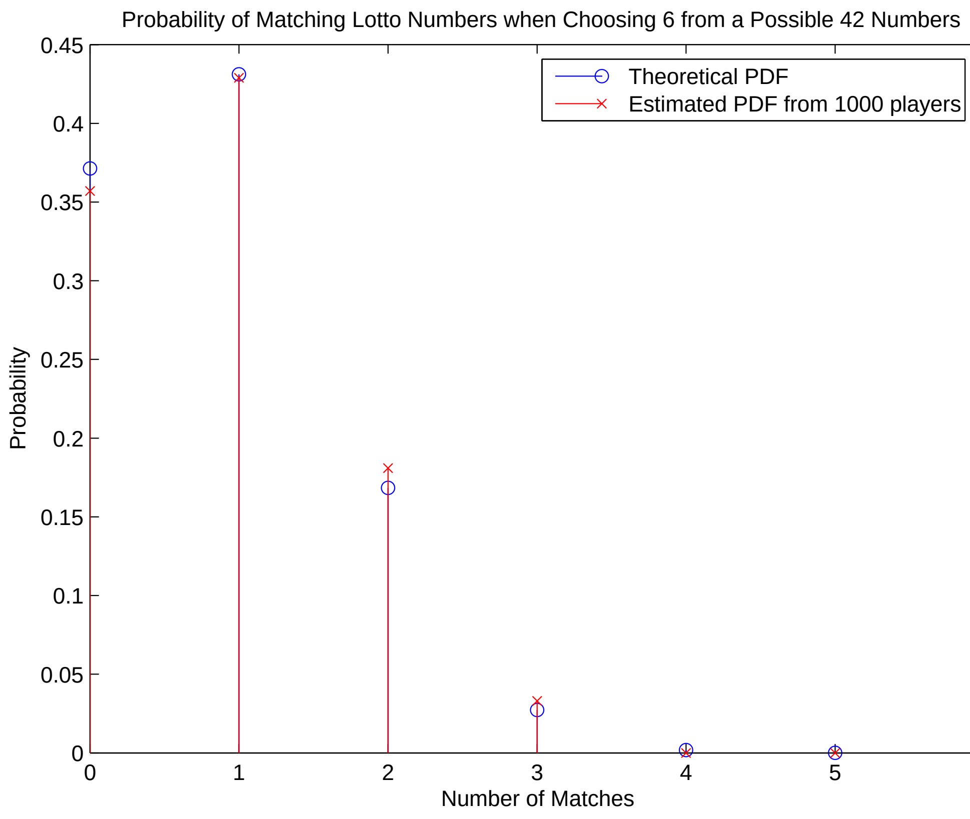

Figure 1: Output of `lotto_histo.m`

An example output from `lotto_histo.m` is shown in Figure 1.

Assignment

1. Go through `lotto_histo.m`. Explain what each piece of code does in this file.

4.2 Keno

Keno is similar to Lotto in that the players choose K numbers out a possible N and the players are paid based on matching k numbers. However in Keno, the State draws $n \geq K$ numbers. Such a game is referred to as an (N, n, K) Keno game. The probability of getting k matches in an (N, n, K) Keno game is

$$P(k; N, n, K) = \frac{\binom{n}{k}\binom{N-n}{K-k}}{\binom{N}{K}}; \quad k = 0, 1 \ldots, K, \tag{7}$$

where $\binom{n}{k}$ is the number of ways a player can match k desired numbers, $\binom{N-n}{K-k}$ is the number of ways a player can match $K-k$ unwanted numbers, and $\binom{N}{K}$ is the total number of ways that a player can choose K numbers from a possible N numbers.

Assignment

2. In a hypothetical scenario, consider playing a (60,20,10) Keno game. What is the probability of matching $0 \leq k \leq 10$ numbers? Write MATLAB functions named `keno_game.m` and `keno_histo.m` that provide the same information as `lotto_game.m` and `lotto_histo.m`, respectively. Use these programs to list and plot example plays and interesting graphs. Include code, graphs, and example plays in your lab write-up.

3. The Lotto program given earlier is a special case of the Keno program. How would you use the Keno program to simulate the Lotto experiment?

4.3 Horse Racing

In a 12 horse race, there are $12! = 479,001,600$ possible ways for the horses to finish, so choosing the order of all 12 is a long shot. However, there are only $(12)(11)(10) = (12)_9 = 1320$ ways for the order of finish for the first, second, and third horse to finish. Matching the top three horses in order is known as a "trifecta."

Assignment

4. Write a MATLAB program that simulates many horse races and keep track of the number of trifecta's that are hit. Have the program display (or echo) the estimated probability of hitting a trifecta. What is the theoretical probability of hitting a trifecta? How does the estimated probability compare to the actual probability. Use many trials.

5. A "trifecta box" bet allows a player to pick the top three horses without specifying the order. These bets cost 6 times as much a regular trifecta bet. Why? How many possible ways can a trifecta box? (HINT: Think about going from permutations to combinations.) Repeat Question 4 for a trifecta box.

Assignment - Advanced Lotto

6. Powerball is a Lotto game where each player picks 5 unique numbers between 1-55 and then one Powerball number between 1-42. The State does the same. How many possible ways are there to create a six digit Powerball number? How many possible ways are there to match $0 \leq k \leq 5$ of the first five numbers correctly and match the Powerball number correctly? How many possible ways are there to get $0 \leq k \leq 5$ of the first five numbers correctly and match the Powerball number incorrectly? Create versions of `lotto_game.m` and `lotto_histo.m` for Powerball. For the `lotto_game.m` program, create two Probability graphs: one for matching the first 5 numbers when the Powerball matches and one for matching the first 5 numbers when the Powerball does not match. Then, create a third graph that has the probability of matching $0, \ldots, 6$ numbers (i.e., do not distinguish between matching one of the five numbers and matching one of the Powerball numbers. NB: Matching the Powerball counts as matching one of the six numbers. How do the odds of matching numbers in Powerball compare to matching numbers in (42,6) Lotto? That is, compare the theoretical probabilities used in the third plot from this problem with the theoretical probabilities for a (42,6) Lotto.

   (HINT: When you are counting the number of matches for the first two graphs, you want to make sure that you are only counting matches when the Powerball is correct/incorrect. One way to do this is to set up a vector of length M that is one when the Powerball matches and zero otherwise. Then, you can use `find` on this vector to select the correct indices for the various `matches` vectors. Also, the first two plots are of conditional probabilities, so their probabilities may not sum up to one. However, the final plot is a PMF and the sum of its probabilities should be one.)

5 Conclusion

The counting ideas that were explored in this lab extend into many other instances. Some of these include (but are not limited to) committee building, polling, birthday matching, card games, and virtually every game that you can place bets on. These methods for calculating odds can be used to explain why Casinos get rich and gamblers get poor. For example, the odds of matching 8 numbers in a (60,20,10) Keno game are roughly 768 to 1, which means that a \$1 Keno ticket with 8 matches should pay

$768. In reality, it may only pay $100, which means that even though you won $99, you were not properly paid for your risk!