Computing for Scientists - Lab 2

CS 1340 — Dr. Mihail
Department of Computer Science
Valdosta State University

February 23, 2018

1 Introduction

In this lab, you will learn about random numbers. Many engineering problems require random numbers as part of their *solutions* or in *simulations*. Simulations are simply realizations (or instantiations) of stochastic (random) processes, that would otherwise be impossible to conduct with simple data collection. Random numbers can also be used to *model* natural processes, such as noise. For example, the static you hear on your analog radio is a noise sequence.

The key aspect of a random value is that individual outcomes are impossible to predict. However, the **distribution** of the outcomes that are the result of a random process can be characterized by mathematical equations. We begin with the most simple distribution, the **uniform distribution**

Uniform distribution Imagine rolling a six-sided die 1000 times. Each time you roll the die, you record the number of dots that is on the top side (e.g., in the image below, the top side indicates a 2).

The die has six sides, so the only possible random values that you can get with this stochastic process (throwing the die) are the numbers 1 through 6. The set of possible outcomes is called the **sample space**. The resulting distribution (how many times each number occurred) is called a **uniform distribution**, because each side is **equally likely** to appear.

Binomial distribution Now imagine rolling two six-sided dice at the same time, 1000 times. Each time you roll the dice, you compute the **sum** of their top faces, and record this number. In the example below, we rolled a 6 and a 5, so we record 11.

The sample space of this experiment are the numbers 2 through 12. These numbers are said to follow a "binomial distribution", which is the discrete version of the **normal** or **Gaussian** distribution. It has the shape of a bell.

2 Random numbers in computers

Computers are deterministic machines, there is **nothing** random about their operation, so truly random number generation is impossible. Instead, computer scientists developed algorithms whose output *simulate the* ***properties*** *of random numbers* and are called **pseudo-random number generators** (PRNGs).

Simple PRNG Below is a sketch of a very simple pseudo-random number generator algorithm. Here is the algorithm design:

1. Take a number to start with (referred to as the **seed**)
2. Square it
3. Take the middle of it - trim some digits in the front and end
4. Output that as your random number
5. Repeat the process by using the previously generated random number as the seed next time

Notice we had to start with a **seed** to initialize the sequence of random numbers.

3 Random numbers in MATLAB

MATLAB PRNGs also use a seed and the command to set the seed is:

```
rng(seed)
```

Here is an example command to set the seed value to 10:

```
>> rng(10)
>>
```

For a fixed seed, **the sequence of random numbers will always be the same**. This is useful when repeatability is needed.

Uniform random numbers in MATLAB MATLAB provides a uniform PRNG, accessible using the function: `rand`. When used with one argument, it produces a square matrix of size n with floating point, uniformly distributed random numbers in the interval $(0, 1)$. For example:

```
>> X = rand(5)

X =

    0.0208    0.1981    0.9534    0.7218    0.1422
    0.6336    0.7605    0.0039    0.2919    0.3733
    0.7488    0.1691    0.5122    0.9178    0.6741
    0.4985    0.0883    0.8126    0.7146    0.4418
    0.2248    0.6854    0.6125    0.5425    0.4340
```

When used with two arguments, such as: `rand(rows, cols)`, it produces a matrix of size *rows* by *columns*.

```
 >> X = rand(1, 5)

X =

    0.7666    0.3211    0.6380    0.0092    0.5435
```

Numbers in this interval $(0, 1)$ are very useful because their range can easily be manipulated. For example, say we wanted to change the range of the above column vector from $(0, 1)$ to $[10, 20]$. We would map $X \in (0, 1)$ to $X \in [10, 20]$ as follows:

- Figure out the range: $range = (20 - 10) + 1 = 11$
- Multiply by that range: $X = X * 10 = [7.6656, 3.2114, 6.3804, 0.0922, 5.4347]$
- Add the lower bound of the range: $X = X + 11 = [18.6656, 14.2114, 17.3804, 11.0922, 16.4347]$
- "fix" the numbers, or truncate the decimals: $X = fix(X) = [18, 14, 17, 11, 16]$.

Or in one line of MATLAB code: `X = fix(X*11 + 10)`

Problem 1 (40 points) Write a MATLAB script to simulate throwing a single six-sided die by using the `rand` function to generate a 100 column vector. Transform the resulting vector from it's original range of $(0, 1) \in \mathcal{R}$ to the set $\{1, 2, 3, 4, 5, 6\} \in \mathcal{N}$. You may include all operations in one line, e.g.: `throws = fix(samples*range + lowerbound)`. `samples` is a column vector, `range` and `lowerboud` are scalar. The result of the operation is a vector.

In the script, you will set the seed to 10. You will also create a histogram using MATLAB's `hist` command to visualize the distribution. Submit this script before lab period ends, using the electronic assignment submission page. Make sure you include a prolog at the beginning of the script. Below is an example of prolog:

```
% John Doe
% CS1340 Lab 3, Problem 1
% Purpose: simulation of repeatedly throwing a single six-sided die
```

Problem 2 (30 points) Write a MATLAB script to simulate throwing two dice multiple times and recording the sum of the resulting numbers. Make the number of throws a variable. You will have to map the results of the two dice from the interval $(0, 1)$ into the set $1, 2, 3, 4, 5, 6$, then add them. You have to work with column vectors. Create a histogram using MATLABs `hist` command to visualize the distribution.

Run this script several times for multiple values of throws. For example, run the script 5 times with 10 number of throws, and observe what the distribution (histogram) looks like. Try 10, 100, 1000, 10000, 100000 and 10000000. At the end of the script, as comments, explain what the effects of increasing the number of throws has on the resulting distribution. Submit this script using the electronic assignment submission page.

Problem 3 (30 points)

Each chemical substance has a density ρ (rho), that can be used to predict the mass given a volume. There is a linear relationship between mass and volume:

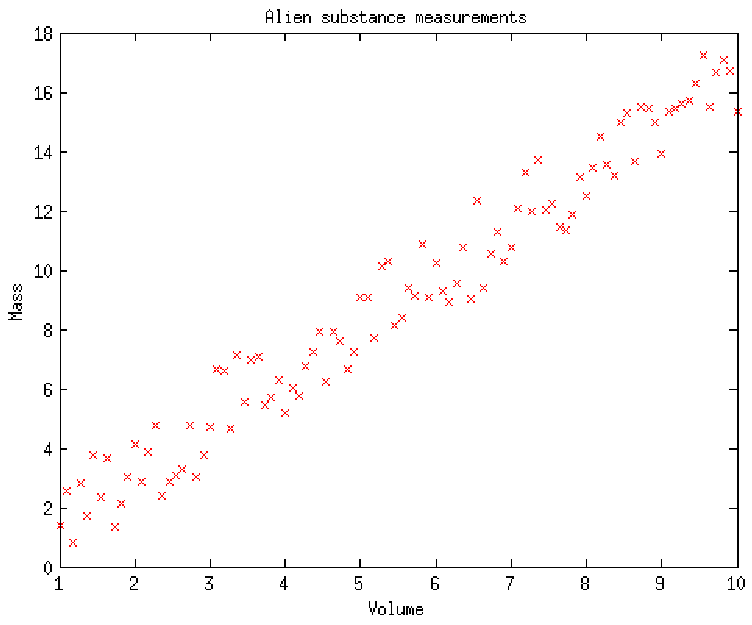
$$m = \rho * V \tag{1}$$

where m is the mass (in grams) and V is the volume in cubic meters. Agent Dana Scully from the X-Files division of the FBI is interested in determining the density, ρ, on Earth, of an alien substance (pictured above). She has a very accurate way to control the volume of a container of that substance, so she records the weight m for 100 volumes, with constant increments from 1 to 10. She notes that her scale is old, with a digital display that she has to lightly tap (and sometimes hit hard) often before it works.

Problem 3 part 1 (20 points) You are collaborating with Agent Scully, and are tasked to write a MATLAB script that loads the measurement data she sends you in an Excel spreadsheet (done using MATLAB *xlsread* function, and calculate ρ. This spreadsheet is available here: `http://mypages.valdosta.edu/rpmihail/teaching/S16/CS1340/datasets/lab3/data_from_scully.xlsx`. Agent Scully has MATLAB on her machine, but she is not a very good programmer. She expects the script to do everything, including to output the density that Agent Fox Mulder knows is accurate, up to two significant digits:

```
The density of the alien substance is 1.64
```

Agent Scully also makes a plot, to help you better understand the data:



Problem 3 part 2 (10 points) You notice that your MATLAB script produced a result slightly different than the one Agent Mulder believes is accurate. Provide two paragraphs (as comments at the end of your .m MATLAB script) that explain **why** your answer is different than Agent Mulder's knowledge of ρ. Submit this script using the electronic assignment submission link on the course web page.

Due dates

1. Lab Problems due at the end of the day, Sunday March 4th. Submissions for problems 1, 2 and 3 in this lab will be MATLAB script files (.m).