L27: Random Numbers

Or rather: not-so-random numbers

Lucas A. J. Bastien

E7 Spring 2017, University of California at Berkeley

March 24, 2017

Version: release

Announcements

Lab 10 is due on April 7 at 12 pm (noon)

Today:

- Pseudo-random numbers

Next Week: Spring break!

- Get some rest, have some fun, see friends and family
- Get a head start on lab 10 and on your E7 project

After Spring break:

- Numerical differentiation (Chapter 17)
- Numerical integration (Chapter 18)

Programming project:

- Instructions and support code will be released during Spring break
- See slides of lecture L26 (March 22^{nd}) to get started on your project now
- "Random" components will be removed from the project

Pseudo-random numbers: introduction

Matlab can generate pseudo-random numbers. The numbers thus generated are not actually random, but are generated using an algorithm that produces predictable numbers with random-like properties

```
>> % Create a 3 by 4 array of pseudo-random
>> % integers ranging from 1 to 100
>> randi([1, 100], [3, 4])
ans =
    82    92    28    97
    91    64    55    16
    13    10    96    98

>> % Try it again, and get different integers
>> randi([1, 100], [3, 4])
ans =
    96    15    80     4
    49    43    96    85
    81    92    66    94
```

Pseudo-random numbers: the seed

The starting point used by the pseudo-random number generator when generating random-like numbers is controlled by the seed

Use `rng(seed)` to set the seed (seed: 0 or positive integer)

```
>> % Use seed = 0
>> rng(0)
>> randi([1, 100], [2, 5])
ans =
    82    13    64    28    96
    91    92    10    55    97

>> % Use seed = 1
>> rng(1)
>> randi([1, 100], [2, 5])
ans =
    42     1    15    19    40
    73    31    10    35    54

>> % Use seed = 0 again
>> rng(0)
>> randi([1, 100], [2, 5])
ans =
    82    13    64    28    96
    91    92    10    55    97
```

Pseudo-random numbers: the seed

Another example:

```
>> rng(25)
>> randi([1, 1000000], [1, 1])
ans =
      870125

>> randi([1, 1000000], [1, 1])
ans =
      582277

>> rng(25)
>> randi([1, 1000000], [1, 1])
ans =
      870125

>> randi([1, 1000000], [1, 1])
ans =
      582277
```

Pseudo-random numbers: practice question

```
>> rng(10)
>> randi([1, 20], [3, 7])
ans =
    16    15     4     2     1    13    19
     1    10    16    14    11    15    15
    13     5     4    20    17     6    11
```

What will the value of variable "v" be after executing the following code?

```
>> rng(10)
>> v = randi([1, 20], [1, 7]);
```

(A) `[16, 15, 4, 2, 1, 13, 19]`

(B) `[16, 1, 13, 15, 10, 5, 4]`

(C) It is impossible to know

Matlab creates arrays of pseudo-random numbers column-by-column

Matlab built-in functions for pseudo-random numbers

- `randi([p, q], [m, n])`:
  Generate a `m` × `n` array of pseudo-random integers between `p` and `q`*
  Numbers are drawn from the corresponding uniform distribution
- `rand([m, n])`:
  Generate a `m` × `n` array of pseudo-random floating-point numbers between `0` and `1`**
  Numbers are drawn from the corresponding uniform distribution
- `randn([m, n])`:
  Generate a `m` × `n` array of pseudo-random floating-point numbers
  Numbers are drawn from the standard normal distribution
- `rng(seed)`:
  Set the seed of the random number generator to seed (*e.g.*, zero; a positive integer; `'shuffle'` to set the seed to a "random" value)

* both boundaries included
** both boundaries excluded
randi, rand, and randn all return variables of class double

Uniform distribution: when generating pseudo-random numbers, all numbers have the same probability of being generated

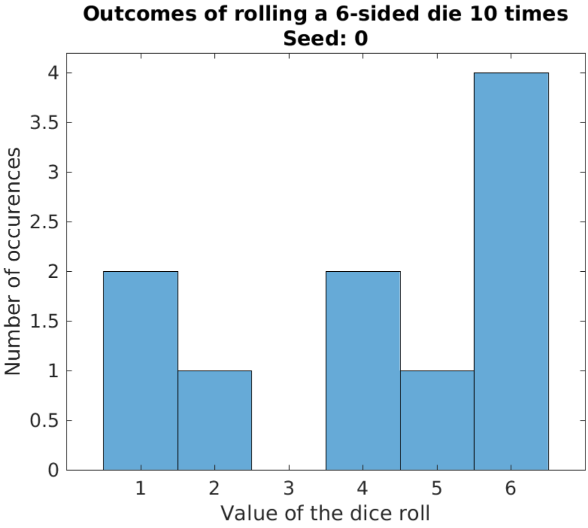
Examples of application:

- Simulate flipping a well-balanced coin
- Simulate rolling a well-balanced die

For example: see the function `my_dice_roller`, which simulates n rolls of a p-sided die, and creates the histogram of outcomes

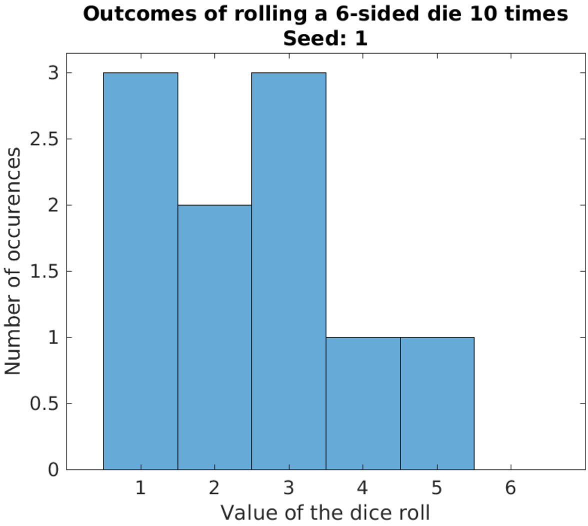
Uniform distribution

For a small number of dice rolls, the distribution of outcomes might not appear to be uniform



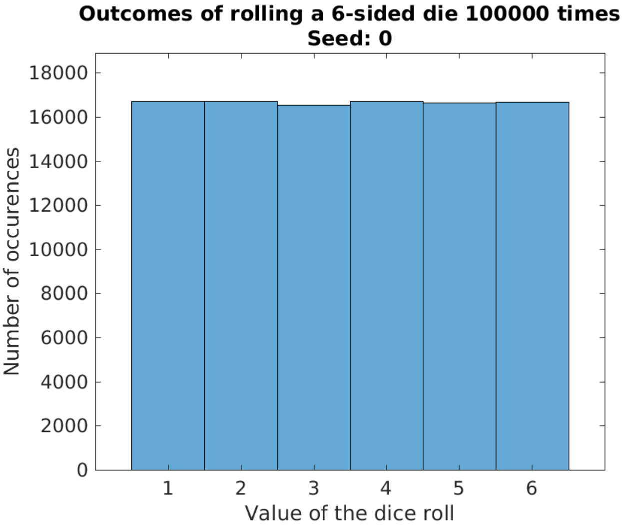
Uniform distribution

For a small number of dice rolls, the distribution of outcomes might not appear to be uniform



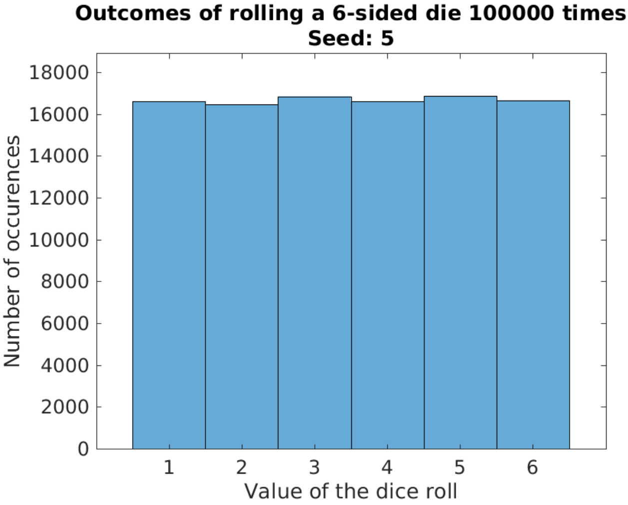
Uniform distribution

For a large number of dice rolls, the histogram of outcomes illustrates the uniformity of the distribution



Uniform distribution

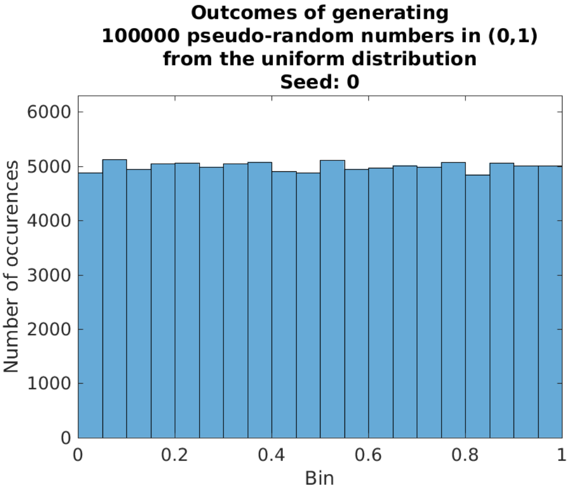
For a large number of dice rolls, the histogram of outcomes illustrates the uniformity of the distribution



Uniform distribution

Generate 100000 pseudo-random floating-point numbers between 0 and 1 drawn from the uniform distribution:

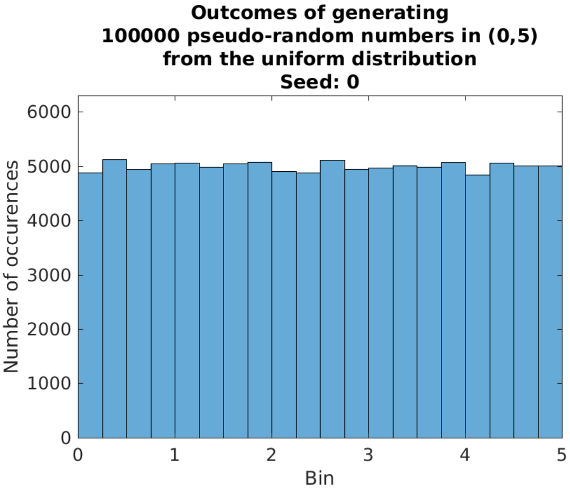
```
rand([1, 100000]);
```



Uniform distribution

Generate 100000 pseudo-random floating-point numbers between 0 and 5 drawn from the uniform distribution:

```
rand([1, 100000]) * 5;
```



Normal distribution

Standard normal distribution: when generating a pseudo-random number using `randn`, the probability φ of a particular `double` x to be generated is proportional to:

$$\varphi(x) = \frac{1}{\sqrt{2\pi}} e^{-(x^2)/2} \tag{1}$$

The mean μ of this distribution is 0 and its standard deviation σ is 1

The **standard** normal distribution φ is a special case of the generic normal distribution f:

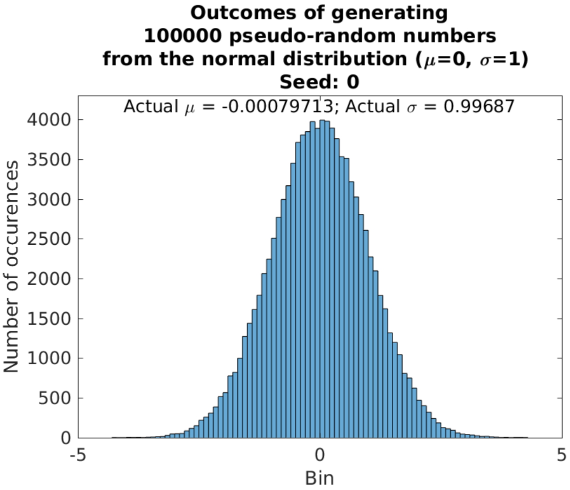
$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x-\mu)^2/(2\sigma^2)} \tag{2}$$

Note: the number of different numbers that Matlab can generate is finite. The concept of probability distributions is slightly different for continuous variables

Normal distribution

Generate 100000 pseudo-random numbers drawn from the standard normal distribution:

```
randn([1, 100000]);
```



Normal distribution

Generate 100000 pseudo-random numbers drawn from the normal distribution with mean $\mu = 10$ and standard deviation $\sigma = 5$:

```
10 + randn([1, 100000])*5;
```