Introduction to Randomized Algorithms I

Joaquim Madeira

Version 0.4 – October 2024

Overview

- Deterministic vs Non-Deterministic Algorithms
- Randomized Algorithms
- Randomness as a Source of Efficiency – Example
- Simulation of Random Events
- Examples of Statistical Experiments
- Examples of Simple Games

DETERMINISTIC VS NON-DETERMINISTIC ALGS

Algorithms

- Algorithm
  - Sequence of non-ambiguous instructions
  - Finite amount of time

- Input to an algorithm
  - An instance of the problem the algorithm solves

- How to classify / group algorithms ?
  - Type of problems solved
  - Design techniques
  - Deterministic vs non-deterministic

Deterministic Algorithms

- A deterministic algorithm
  - Returns the same answer no matter how many times it is called on the same data.
  - Always takes the same steps to complete the task when applied to the same data.

- The most familiar kind of algorithm !

- There is a more formal definition in terms of state machines…

Non-Deterministic Algorithms

- A non-deterministic algorithm
  - Can exhibit different behavior, for the same input data, on different runs.
  - As opposed to a deterministic algorithm !
- Often used to obtain approximate solutions to given problem instances
  - When it is too costly to find exact solutions using a deterministic algorithm

Non-Deterministic Algorithms

- How to behave differently from run to run ?
- Factors of non-deterministic behavior
  - External state other than the input data
    - User input / timer values / random values
  - Timing-sensitive operation on multiple processor machines
  - Hardware errors might force state to change in unexpected ways

RANDOMIZED ALGORITHMS

Randomized Algorithms

- Use a degree of randomness as part of an algorithm's logic

- Algorithm behavior can be guided by random bits as an auxiliary input
  - Take decisions by tossing coins !

- Aiming at good performance on average !

Randomized Algorithms

- What is the effect of randomness ?
- Algorithm running time and / or algorithm output are random variables
  - Determined by the random bits / by the coin tossing results

APPLICATION EXAMPLE

Randomness as a source of efficiency

- Computers C_1 and C_2 at separate locations
  - Connected via a network
- Initial copies of the same DB: DB_1 and DB_2
- BUT, contents evolve over time !
- DB changes have been done simultaneously
- Do DB_1 and DB_2 contain the same data ?

Deterministic approach

- DBs of size n bits (e.g., $n = 10^{16}$)
- Is the data on both computers the same ?
  - Yes / No – Decision Problem

- What is the number of bits that have to be exchanged, between C_1 and C_2, to solve the problem ?
- At least n bits !!
  - Send the entire DB, without communication errors

Randomized approach

mod SAME AS %

- Contents of DB_1 are a string X of n bits
- Contents of DB_2 are a string Y of n bits

- C_1 makes a uniform random choice of a prime number p from $[2, n^2]$
- Computes s = Number(X) mod p
  - String X is the binary rep. of natural Number(X)
- And sends (s, p) to C_2

Randomized approach

- Size of the message (s, p) ?

- At most,

$$4 \times \text{ceil}(\log_2 n) \text{ bits}$$

- Given that $s \leq p < n^2$

- $n = 10^{16}$ implies a message of, at most, 256 bits

Randomized approach

- C_2 reads (s, p)
- Computes r = Number(Y) mod p

- If $s \neq r$, then C_2 outputs "X and Y are different"
- If $s = r$, then C_2 outputs "X and Y are equal"

maybe are equal

- Reliable answers ? Not Always
If x=15, y=22 and p=7

Randomized approach

- Reliability of the final answer ?

- If $X = Y$, then the answer is always correct !!
- If $X \neq Y$, then the answer might be wrong !!

- For $X \neq Y$, the output might be "X and Y are EQUAL", if the chosen prime was a "bad" prime for (X, Y)

  - **Number(X) mod p = Number(Y) mod p, with $X \neq Y$**

Exemple:
If x=15, y=22 and p=7

Randomized approach

- Choose p from {2, 3, 5, 7, 11, 13, 17, 19, 23}
- $p = 7$

- $X = 01111 \quad \rightarrow \quad \text{Number}(X) = 15$
- $Y = 10110 \quad \rightarrow \quad \text{Number}(Y) = 22$

- $\text{Number}(X) \bmod p = \text{Number}(Y) \bmod p$
- BUT, $X \neq Y$

Randomized approach

- Error probability ?

- At most, $(\ln n^2) / n$, which presents no real risk…

- For n = 10^{16} the error probability is, at most, 0.36892×10^{-14}

Randomized approach

- If we want to be safer, we can use 10 rand. chosen primes
  - 10 independent repetitions
  - Message will be 10 times larger ! – OK !

- Error probability ?
  - Are all 10 primes “bad” primes ?

- For n = 10^{16} the error probability is smaller than 0.4717×10^{-141}

RANDOM NUMBER GENERATORS

Random Number Generators

- The source of randomness is usually a random number generator
  - Repeated calls return a stream of numbers
  - That appear to be randomly chosen
  - From some range / interval
- In reality, they are pseudo-random numbers !
  - Generated by particular recurrence relations
  - It is possible to calculate each value from a sequence of preceeding values !!

Dependant from an initial seed. If we always run with same seed, we always get the same result

Random Number Generators

- Check the story of Daniel Corriveau at
  - http://www.americancasinoguide.com/gambling-stories/costly-casino-mistakes-the-keno-mix-up
- What happened ?

Python – The `random` Module

- For integers
  - `randint(…)`
  - `randrange(…)`

- For sequences
  - `choice(…)`
  - `sample(…)`
  - …

```python
import random

mylist = ["apple", "banana", "cherry"]

print(random.choice(mylist))
```

```
banana
```

```python
from random import sample

# Prints list of random items of given length
list1 = [1, 2, 3, 4, 5]

print(sample(list1,3))
```

Output:

```
[2, 3, 5]
```

Code #2:

Python – The `random` Module

- For generating real-valued distributions
  - `random() # next random float in [0,1)`
  - `uniform(…)`
  - `gauss(…)`
  - …

Python – The `random` Module

- Reproducibility
  - It might be useful to reproduce the sequences given by a pseudo random number generator
- Re-using of seed values
  - Same sequence should be reproducible from run to run, as long as multiple threads are not running
  - `seed(…)`

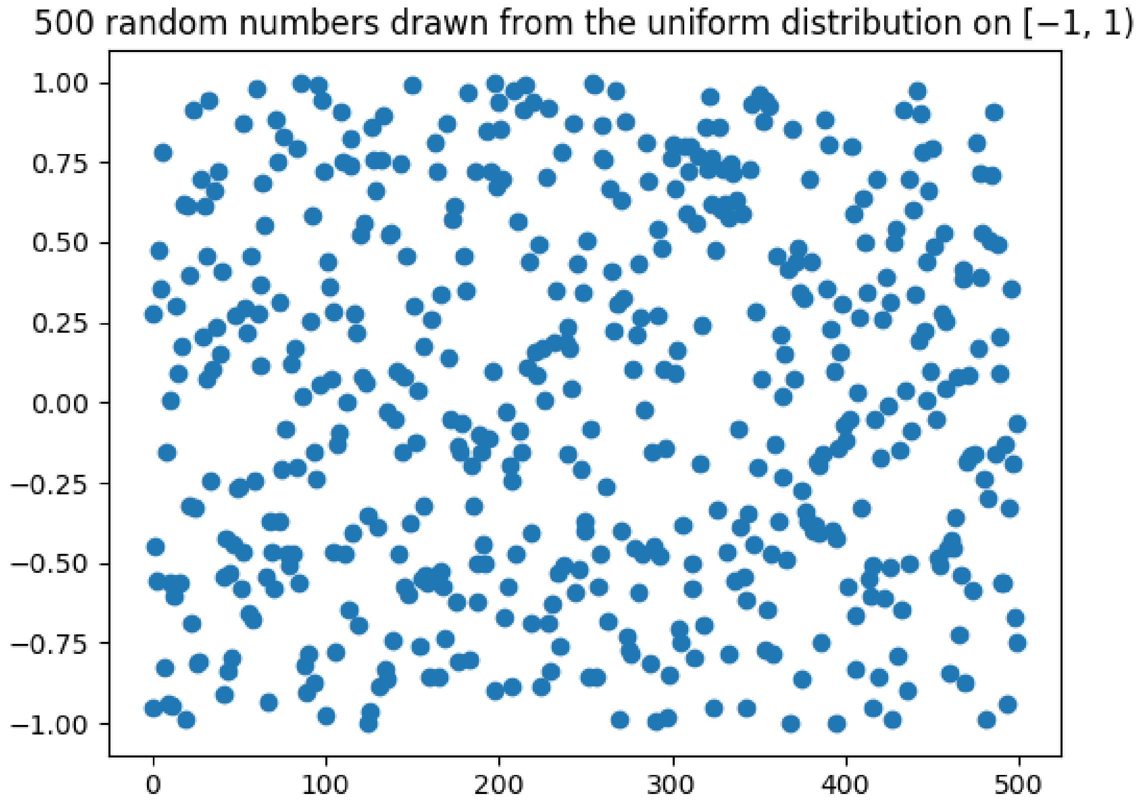
Python – The `secrets` Module

- The pseudo-random generators of the `random` module should not be used for security purposes !

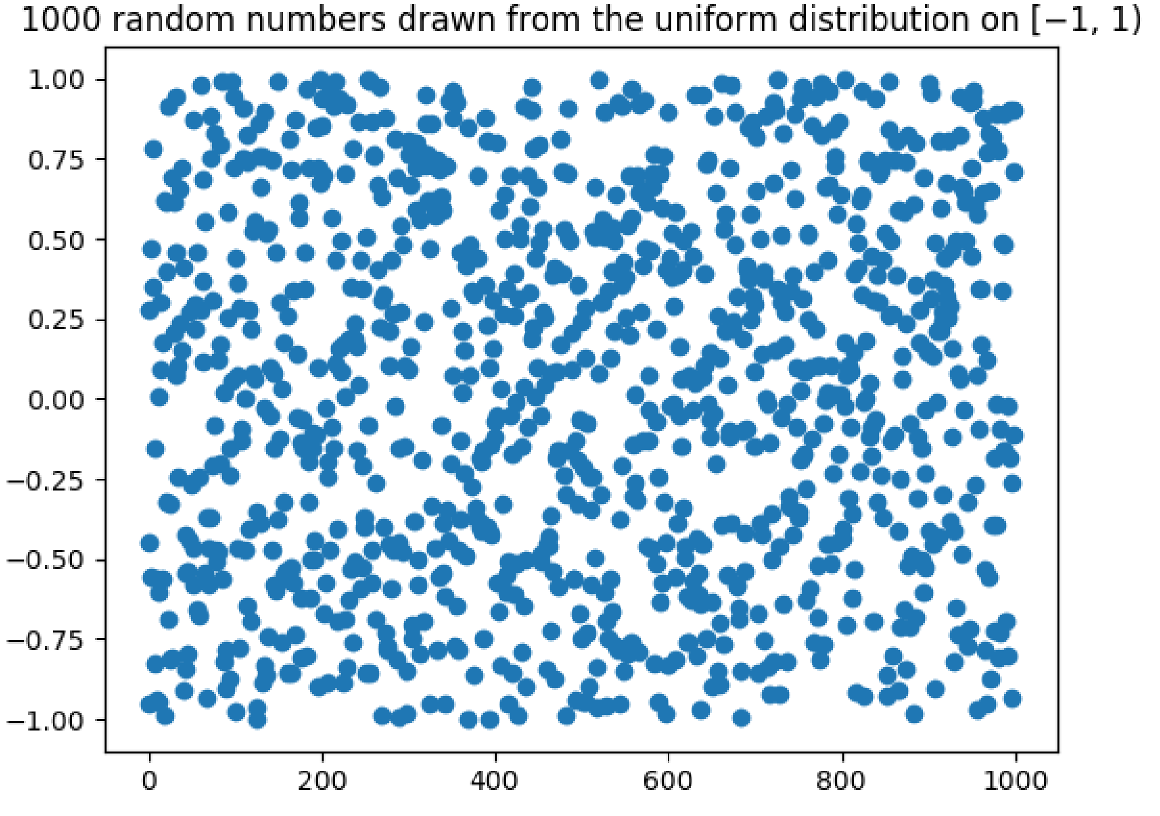
- For security or cryptographic uses, use the `secrets` module instead !
  - Generation of secure random numbers

RANDOM NUMBER DISTRIBUTIONS

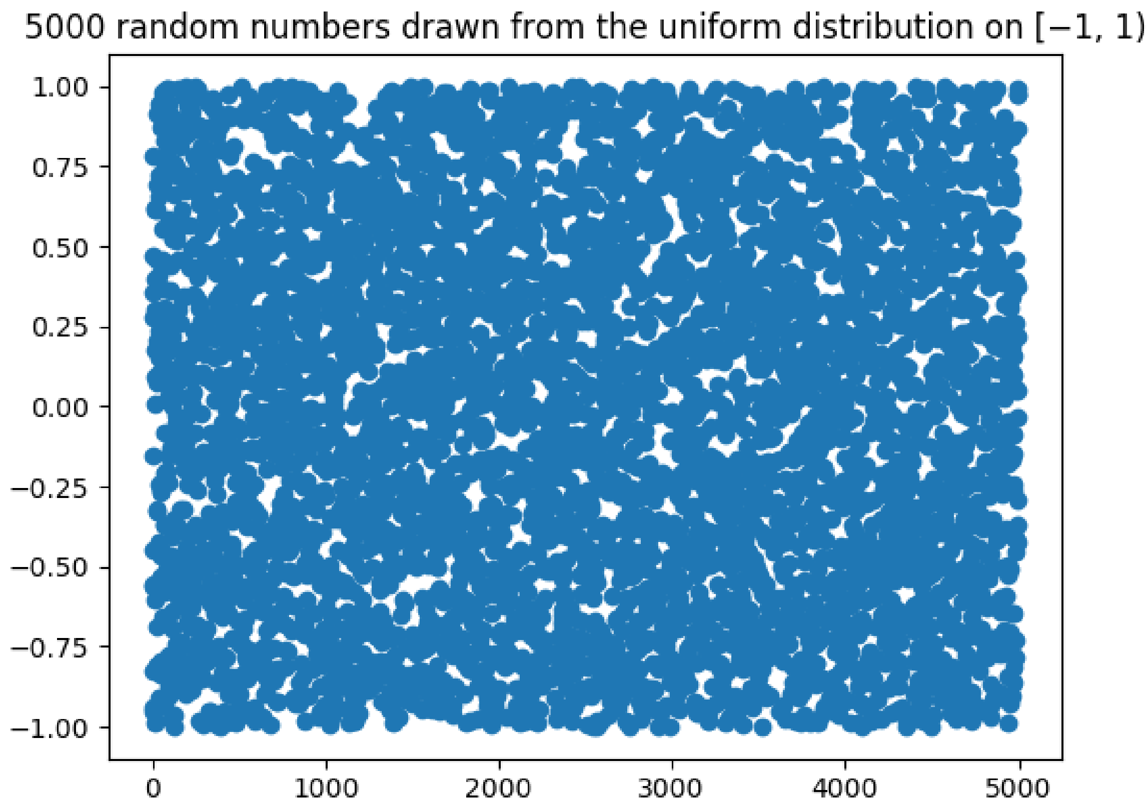
Uniform Distribution



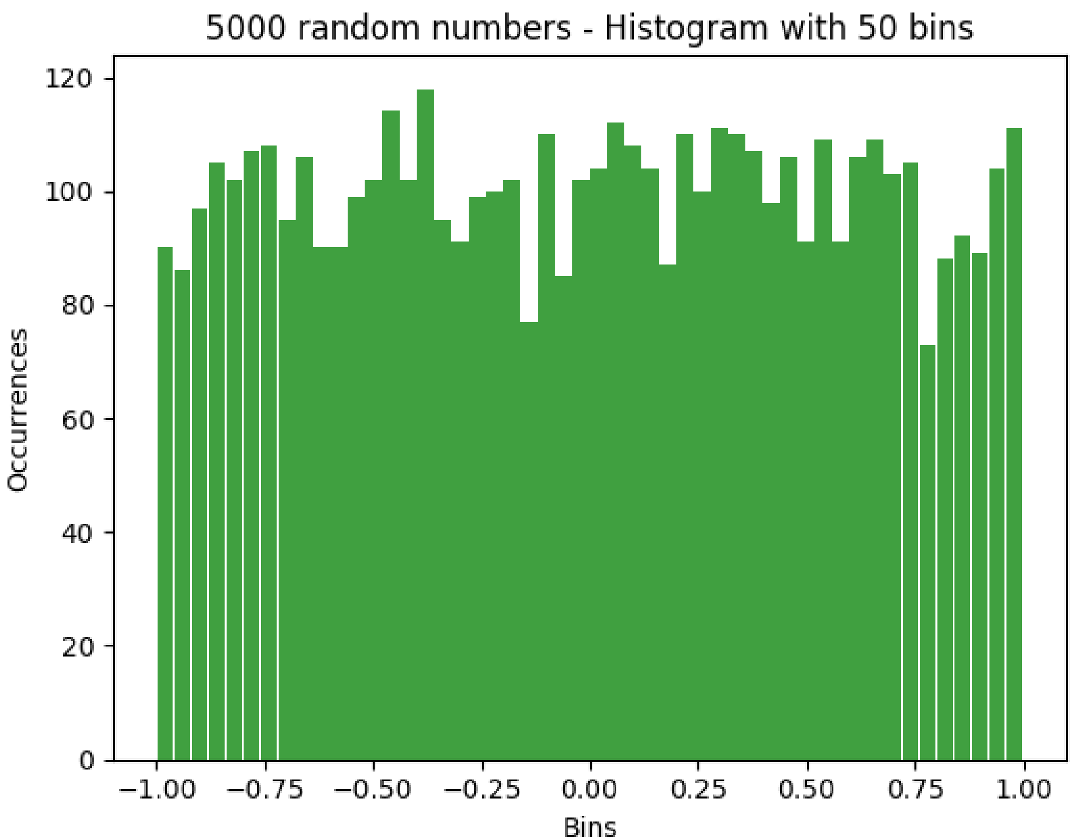
Uniform Distribution



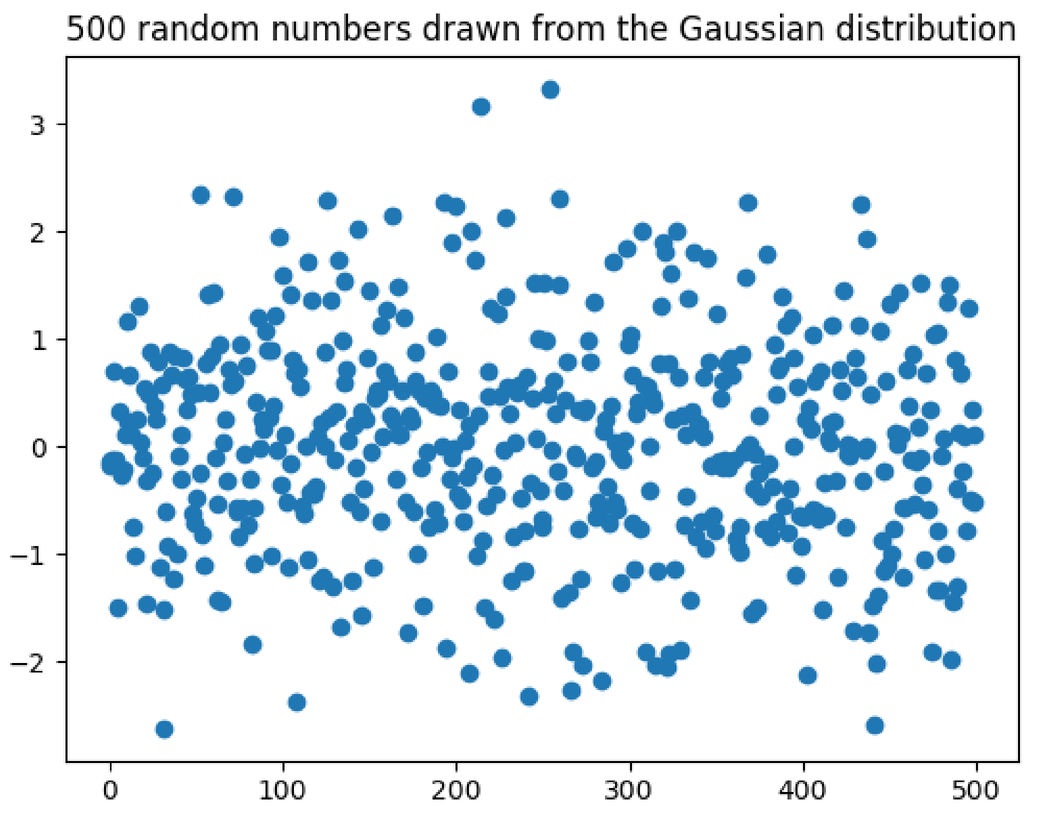
Uniform Distribution



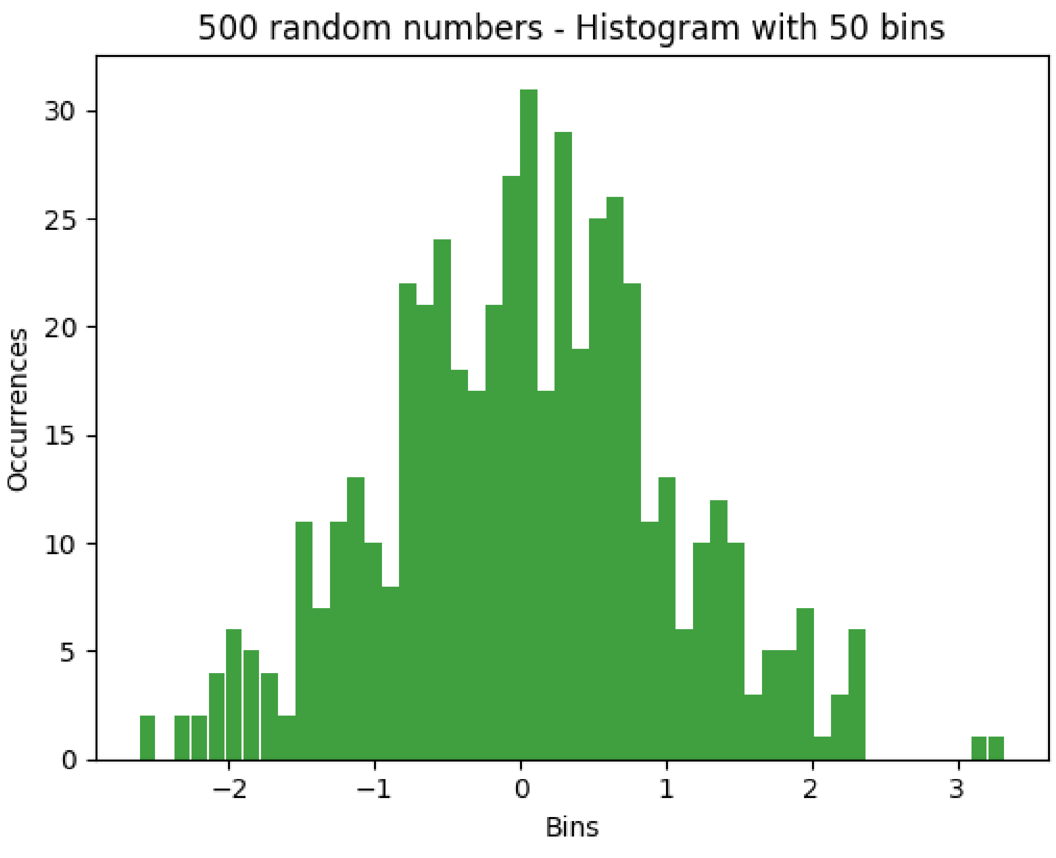
Uniform Distribution



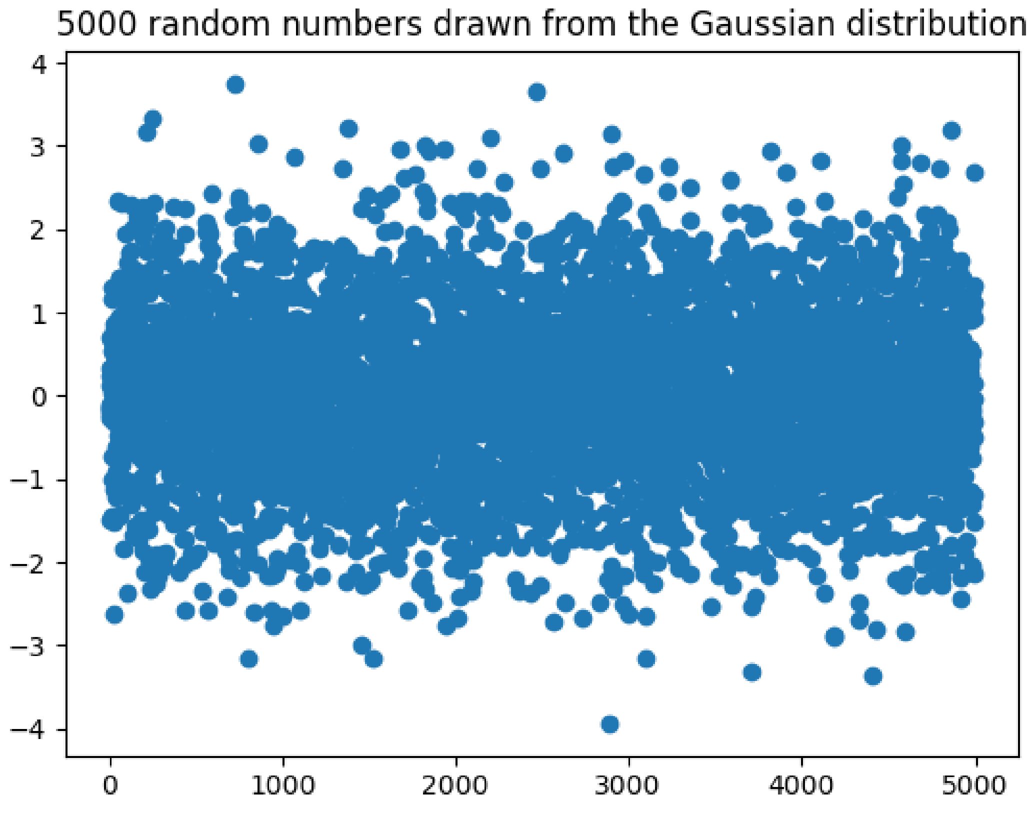
Gaussian Distribution



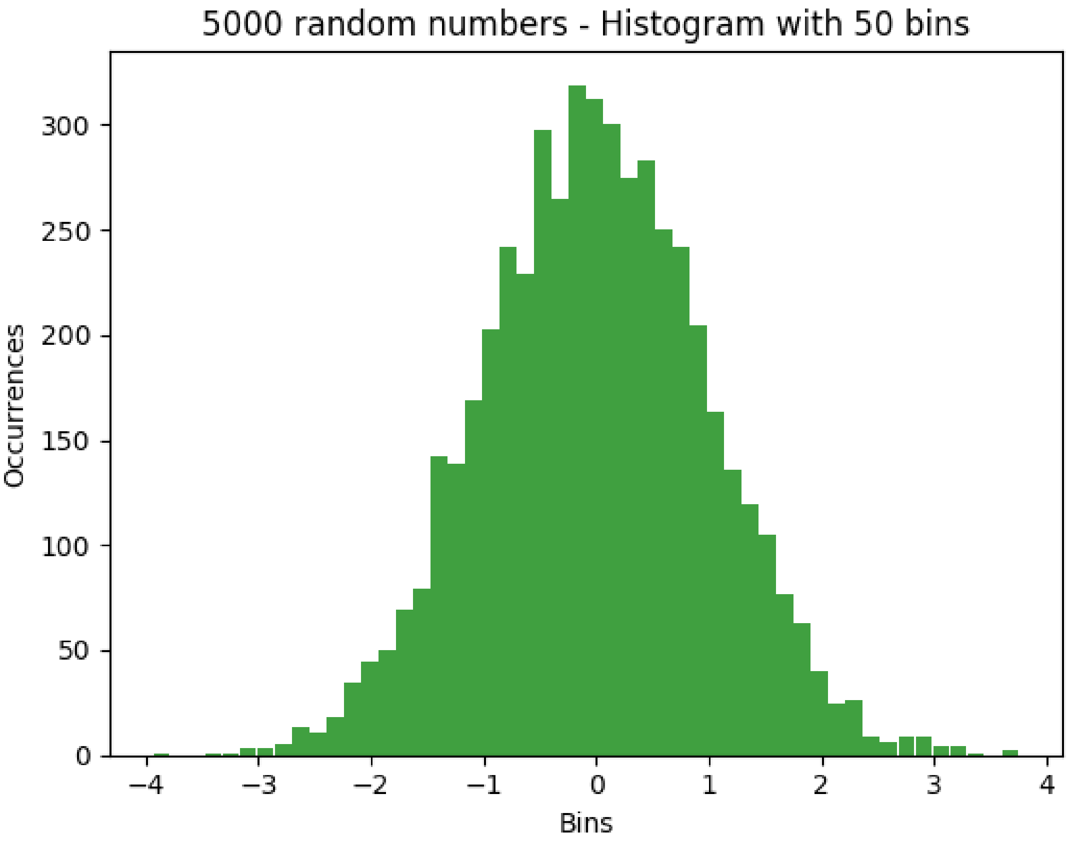
Gaussian Distribution (or normal distribution)



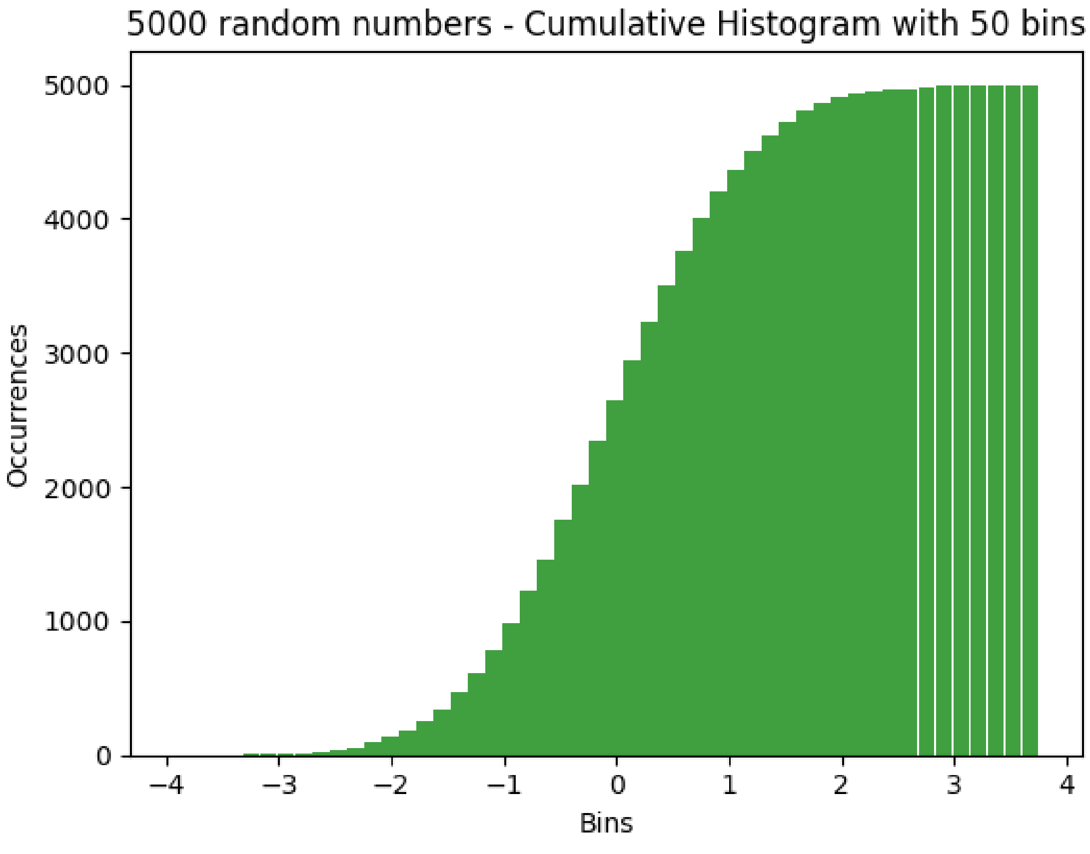
Gaussian Distribution



Gaussian Distribution



Gaussian Distribution



SIMULATION OF RANDOM EVENTS

Simulation of Random Events

- Model random events, such that simulated outcomes closely match real-world outcomes

- Analyze simulated outcomes to gain insight !

- Why approximate the real-world ?
  - No precise mathematical description…
  - OR
  - Less time / effort / cost than other approaches

Simulations have to be useful

- How to mirror the real-world ?
- 1st – Prepare the experiment !
- Identify the possible outcomes
- Link each outcome to one (or more) random number(s)
- Choose a source of random numbers

Simulations have to be useful

- 2nd – Run the experiment loop !

- Choose one (or more) random number(s)
- Record the simulated outcome

- 3rd – Analyze the data and report results !
  - Histogram
  - …

Applications

- Simulation of real-world systems for which the input is random in some way
  - Queueing in check-out lines
  - …
- Simulation of statistical experiments
  - Tossing balanced / biased coins
  - Throwing fair / unfair dice
  - …

STATISTICAL EXPERIMENTS

A Coin Experiment – V1

n is not the number of experiments

n is the number of coins tossed in each experiment

- Toss a balanced coin n times
  - n >= 1 – parameter of the experiment
  - n independent replications of the simplest exp.
- Record the total score of the experiment
  - 1 for heads or 0 for tails
- What do you expect ?

Task – Toss a balanced coin 3 times

- How to represent ? Let's do it !

- Binary table
- Binary tree
  - Order is important ! (order of the coins)
- Directed Graph
  - Order is not important ! (order of the coins) - only care about number of heads share the ones with same number of heads
  - Check the paths !
- Counting Heads – Table of probability distribution

A Coin Experiment – V2

- The coin is now biased !!
- It turns up heads only 45% of the time
- Again, toss the biased coin n times
- And record the total score
- Now, what do you expect ?

Task – Toss a biased coin 3 times

- Representation ?
- What are the needed changes ??

- Binary tree
- Directed Graph

- Counting Heads – Table of probability distribution

Tasks – Simulations

- Simulate both coin experiments
- For $n = 1, 3, 5$, and 7
- Run the simulations 10, 100 and 1000 times
- Observe the outcomes
  - Histograms

A Die Experiment – V1

- Throw a standard 6-sided die n times
- Record the total score of the experiment
- What do you expect ?

Task – Throw a balanced die 2 times

- Count the number of “eyes”
- How to represent ? Let’s do it !

- 6-ary tree
- Directed Graph
- Table of probability distribution

A Die Experiment – V2

- The die is now an unfair die !!

ace -> one dot

- For which an ace is twice as likely to turn up as any other face 2/6

- Again throw the unfair die n times
- And record the total score

- What do you expect ?

Task – Throw a biased die 2 times

- Count the number of "eyes"
- Representation ?
- What are the needed changes ??

- 6-ary tree
- Directed Graph
- Table of probability distribution

Tasks – Simulations

- Simulate both die experiments
- For n = 1, 3, 5, and 7
- Run the simulations 10, 100 and 1000 times
- Observe the outcomes
  - Histograms

Another experiment with coins

- Toss two balanced coins n times !!
- Record the total score of the experiment
  - 1 for heads or 0 for tails
- What do you expect ?

Another experiment with dice

- Throw a pair of fair dice n times !!
- Record the sum of the faces that turn up
- What do you expect ?

Tasks – Simulations

- Simulate both experiments
- For n = 1, 3, 5, and 7
- Run the simulations 10, 100 and 1000 times
- Observe the outcomes
  - Histograms

A Die-Coin Experiment

- A standard die is thrown and then a coin is tossed the number of times shown on the die
  - Compound experiment
  - Second, dependent stage
- Record the total coin score

- Randomization of the first coin experiment !

Task – A Die-Coin Experiment

- Draw the tree representing the experiment

- What is the probability of getting 6 heads ?
- …
- What is the probability of getting 0 heads ?

- Table of probability distribution

A Coin-Dice Experiment

- A coin is tossed
- If the coin lands heads, a red die is thrown
- If the coin lands tails, a green die is thrown
  - Again, a compound experiment

- Record the die color and score

Task – A Coin-Dice Experiment

- Draw the tree representing the experiment

- What is the probability of getting 6 heads ?
- What is the probability of getting 6 green heads ?

- …

- Table of probability distribution

Tasks – Simulations

- Simulate both experiments
- Run the simulations 10, 100 and 1000 times
- Observe the outcomes
  - Histograms

Extra Tasks

- Simulate experiments using k-sided dice

[Wikipedia]

SIMPLE GAMES

Task – A Simple Game

- You pay 1 euro to roll two dice
  - Red + Green
- You win 2 euros, if there are more eyes on red than on the green die
- Should you play this game ?
- Run a few simulations and decide !!

Task – A Simple Game

- You roll two dice and, beforehand, guess the sum of the eyes: n eyes
- If the guess turns out to be right, you earn n euros; otherwise, you pay 1 euro

- Should you play this game ?

- Run a few simulations and decide !!

AN INTERESTING READING – IN PORTUGUESE

Uma leitura interessante

- Persi Diaconis: "Atirar um moeda ao ar é física, não é aleatório"
  - https://sol.sapo.pt/artigo/776994/persi-diaconis-atirar-uma-moeda-ao-ar-e-fisica-nao-e-aleatorio

REFERENCES

References

- D. Vrajitoru and W. Knight, *Practical Analysis of Algorithms*, Springer, 2014
  - Chapter 6
- J. Hromkovic, *Design and Analysis of Randomized Algorithms*, Springer, 2005
  - Chapter 1
- H. P. Langtangen, *A Primer on Scientific Programming with Python*, 4th Ed., Springer, 2014
  - Chapter 8