# MODULE 5

Randomness

# WHY RANDOMNESS?

- Data scientists should be able to understand **randomness**.
- For example, they should be able to assign individuals to **treatment** and **control** groups **at random**, and then try to say whether any observed differences in the outcomes of the two groups are simply due to the random assignment or genuinely due to the treatment.

# COMPARISON AND BOOLEANS

# RANDOMNESS AND BOOLEANS

- A fundamental question about random events is **whether or not** they occur. For example:
  - Did an individual get assigned to the treatment group, or not?
- Once the event has occurred, you can answer "yes" or "no" to these questions.
- In programming, it is conventional to do this by labeling statements as True or False.
- In Python, Boolean values, named for the logician George Boole, represent **truth** and take only two possible values: True and False.

# COMPARISON OPERATORS

The result of a comparison expression is a **bool** value

```
x = 2                y = 3
```

Assignment statements

```
x > 1                x > y                y >= 3

x == y               x != 2               2 < x < 5
```

Comparison expressions

# AGGREGATING COMPARISONS

Summing an array or list of bool values will count the True values only.

```
1    + 0     + 1            == 2
True + False + True         == 2
sum([1   , 0    , 1   ])   == 2
sum([True, False, True])   == 2
```

(Demo – notebook 5.1,
Aggregating comparisons)

# NP.COUNT_NONZERO()

- The ***numpy*** method ***count_nonzero*** evaluates to the number of non-zero (that is, True) elements of the array.

```
tosses = make_array('Tails', 'Heads', 'Tails', 'Heads', 'Heads')
tosses == 'Heads'
```

```
array([False,  True, False,  True,  True], dtype=bool)
```

```
np.count_nonzero(tosses == 'Heads')
```

```
3
```

# CONDITIONAL STATEMENTS

# RANDOMNESS AND CONDITIONAL STATEMENTS

- In many situations, actions and results depend on a specific set of conditions being satisfied.
  - For example, individuals in randomized controlled trials receive the treatment if they have been assigned to the treatment group.
- A *conditional statement* is a multi-line statement that allows Python to choose among different alternatives **based on the truth value** of an expression.

# CONDITIONAL STATEMENTS

- A conditional statement always begins with an if header, which is a single line followed by an indented body.
  - The purpose of **if** is to define functions that choose different behavior based on their arguments
- The body is only executed if the expression directly following if (called the *if expression*) evaluates to a true value.
- If the *if expression* evaluates to a false value, then the body of the if is skipped.

# GENERAL FORM OF CONDITIONAL STATEMENTS

- A conditional statement can have **multiple clauses** with **multiple bodies**, and only one of those bodies can ever be executed.

```
if <if expression>:
    <if body>
elif <elif expression 0>:
    <elif body 0>
elif <elif expression 1>:
    <elif body 1>
...
else:
    <else body>
```

(Demo – notebook 5.1, Conditional statements)

# RANDOM SELECTION

# RANDOM SELECTION IN PYTHON

**np.random.choice**

- Selects uniformly at random
- with replacement
- from an array
- a specified number of times

**np.random.choice(some_array, sample_size)**

Demo – notebook 5.1, Random selection

# ITERATION

# RANDOMNESS AND ITERATION

- In programming – especially when dealing with randomness – we often want to repeat a process multiple times.
- For example, we might want to assign each person in a study to the treatment group or to control, based on tossing a coin.
- We could run `np.random.choice(make_array('Heads', 'Tails'))`
  - However, we would need to copy and paste and run it for each of the participants in the study, even if they are 1000!
- Better strategy? Iteration

# ITERATION

- A more automated solution is to use a for statement to loop over the contents of a sequence. This is called *iteration*.
- A for statement:
  - begins with the word for,
  - followed by a name we want to give each item in the sequence,
  - followed by the word in,
  - and ending with an expression that evaluates to a sequence.
- The indented body of the for statement is executed once *for each item in that sequence*.

# FOR STATEMENTS

- **for** is a keyword that begins a control statement
- The purpose of **for** is to perform a computation for every element in a list or array
- Example:

```
for i in np.arange(3):
    print(i)
```

```
0
1
2
```

(Demo – notebook 5.1,
For statements)

# APPENDING ARRAYS

# STORING THE RESULTS OF AN ITERATION

- The *for statement* in the previous slide simply prints the output.
- This output is NOT in a form that we can use for computation.
- A typical use of a *for statement* is to **create an array of results**, ***by augmenting it each time***.

- The append method in numpy helps us do this.

# APPENDING TO AN ARRAY

- **np.append(array_1, value)**
  - new array with value appended to array_1
  - value has to be of the same type as elements of array_1
- **np.append(array_1, array_2)**
  - new array with array_2 appended to array_1
  - array_2 elements must have the same type as array_1 elements

(Demo – notebook 5.1,
Appending arrays)

# SIMULATION

# DEFINITION AND STEPS

Simulation is the process of using a computer to mimic a physical experiment.

- **Step 1: What to Simulate**
    - For example, you might decide that you want to simulate the outcomes of tosses of a coin.
- **Step 2: Simulating One Value**
    - In our example, figure out how to simulate the outcome of *one* toss of a coin.
- **Step 3: Number of Repetitions**
    - Decide how many times you want to simulate the quantity, then repeat Step 2 that many times. E.g., 1000
- **Step 4: Coding the Simulation**
    - Put it all together in code.

# SIMULATION STEP 4

**Step 4: Coding the Simulation**

1. Create an empty array in which to collect all the simulated values. We will call this the **collection array.**

2. Create a "repetitions sequence," - a sequence whose length is the number of repetitions you specified in Step 3.
   1. For n repetitions we will often use the sequence np.arange(n).

3. Create a for loop. For each element of the repetitions sequence:
   1. Simulate *one* value based on the code you developed in Step 2.
   2. Augment the collection array with this simulated value.

# EX 1: BIGGER NUMBER = $1

- Let's play a game: we each roll a die.
  - If my number is bigger: you pay me a dollar.
  - If they're the same: we do nothing.
  - If your number is bigger: I pay you a dollar.
- Steps:
  1. **What to simulate**: two dice rolls.
  2. **Simulate one value**: compute how much money we win/lose based on the result of the roll
  3. **Number of repetitions**: Do steps 1 and 2 10,000 times.
  4. **Put it all in code**

(Demo – notebook 5.1, Simulation)

# EX 2: NUMBER OF HEADS IN 100 TOSSES

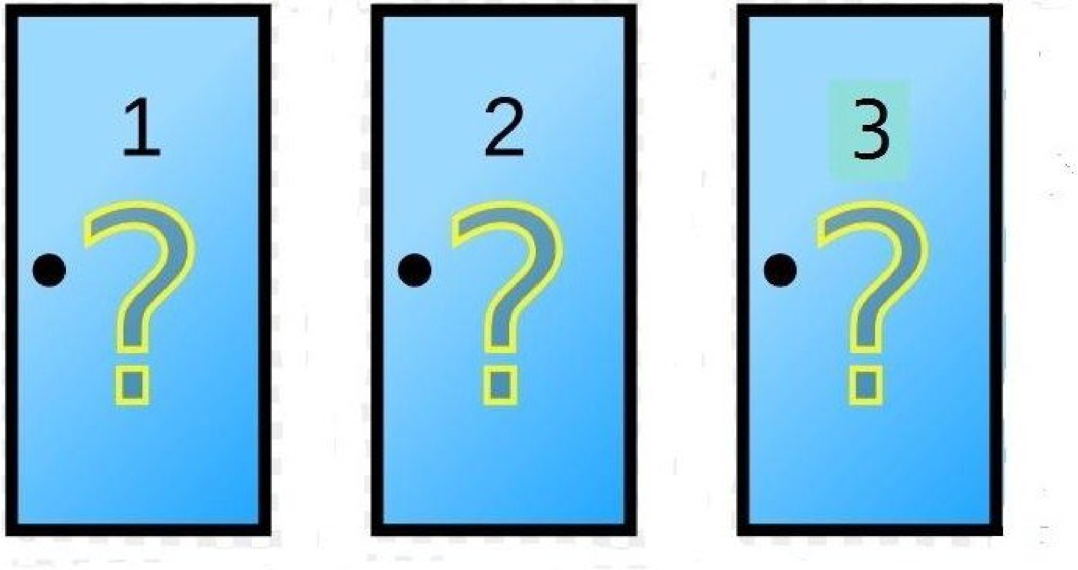
- In this example we will simulate the number of heads in 100 tosses of a coin.
- Steps:
  1. **What to simulate**: outcomes of tosses of a coin.
  2. **Simulate one value**: make ***one* set of 100 tosses** and **count the number of heads**
  3. **Number of repetitions**: Do steps 1 and 2 10,000 times.
  4. **Put it all in code**

(Demo – notebook 5.1, Simulation)

# CHANCE AND PROBABILITY

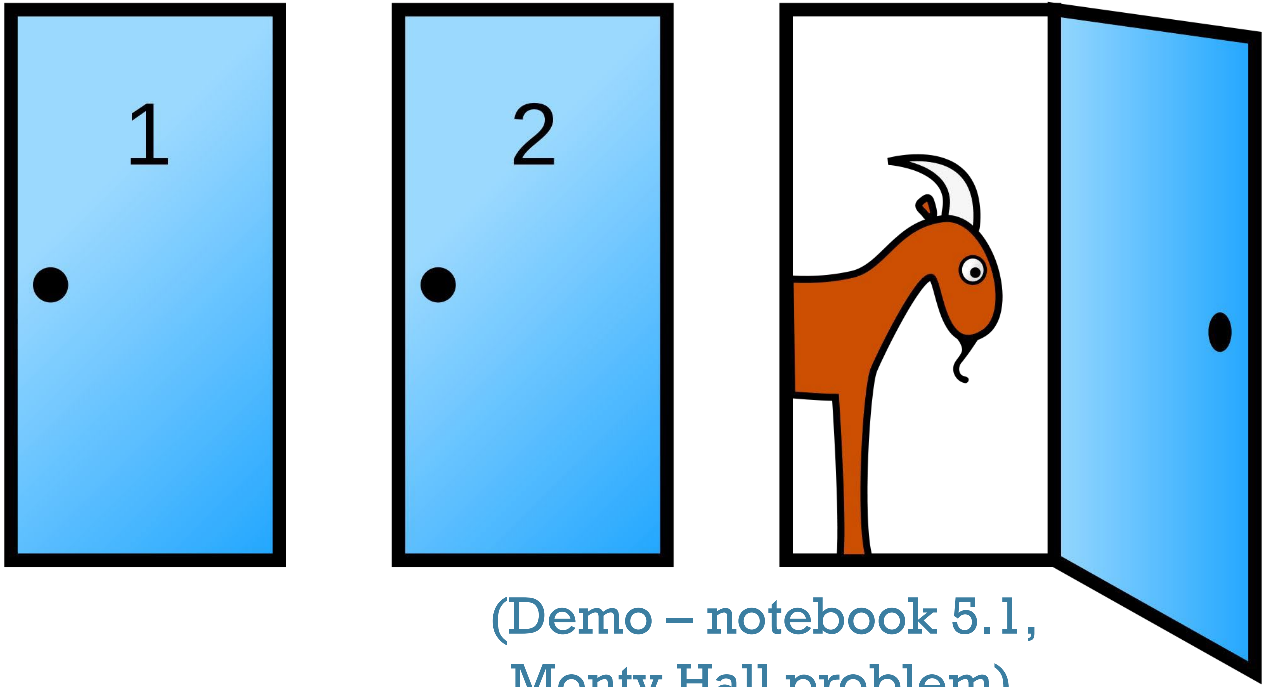
# THE MONTY HALL PROBLEM

# MONTY HALL PROBLEM




https://probabilityandstats.files.wordpress.com/2017/05/monty-hall-pic-1.jpg

# THE FINAL CHOICE




(Demo – notebook 5.1, Monty Hall problem)

https://en.wikipedia.org/wiki/Monty_Hall_problem

Still can’t wrap your head around it? Here’s another intuitive way to think about it

# PROBABILITY

# DEFINITIONS AND NOTATIONS

- Most probabilities will be **relative frequencies**
- By convention, probabilities are numbers between 0 and 1, or, equivalently, 0% and 100%.
- Standard notation: **P(event)** denotes the **probability that "event" happens**

# BASICS

- **Lowest value**: 0
  - Chance of event that is impossible
- **Highest value**: 1 (or 100%)
  - Chance of event that is certain
- **Complement**: P(an event doesn't happen)
  - = 1–P(the event happens)
  - If an event has chance 70%, then the chance that it doesn't happen is
    - 100% - 70% = 30%, or, 1 - 0.7 = 0.3

# EQUALLY LIKELY OUTCOMES

**Assuming** all outcomes are equally likely, the chance of an event A is:

$$P(A) = \frac{\text{number of outcomes that make A happen}}{\text{total number of outcomes}}$$

# A QUESTION

- I have three cards: **ace of hearts**, **king of diamonds**, and **queen of spades**.

- I shuffle them and draw two cards *at random without replacement.*

- What is the chance that I get the Queen followed by the King?

# MULTIPLICATION RULE

Chance that two events *A* and *B* both happen
= P(*A* happens) x P(*B* happens given that *A* has happened)

- The answer is *less than or equal to* each of the two chances being multiplied
- The more conditions you have to satisfy, the less likely you are to satisfy them all

# ANOTHER QUESTION

- I have three cards: **ace of hearts**, **king of diamonds**, and **queen of spades**.
- I shuffle them and draw two cards *at random without replacement.*
- What is the chance that one of the cards I draw is a King and the other is Queen?

AK AQ KA KQ QA QK

# ADDITION RULE

If event *A* can happen in *exactly one* of two ways, then

$$P(A) = P(\text{first way}) + P(\text{second way})$$

- The answer is *greater than or equal to* the chance of each individual way

# COMPLEMENT: E.G., AT LEAST ONE HEAD

- In 3 tosses:
  - Any outcome *except* TTT
  - P(T) = ½
  - P(TTT) = (1/2) x (1/2) x (1/2) = (1/2)**3 = 1/8
  - P(at least one head) = 1 - P(TTT) = 1 - (1/8) = 87.5%
- In 10 tosses:
  - 1 - (1/2)**10 ≅ 99.9%

# DISCUSSION QUESTION

A population has 100 people, including Rick and Morty. We sample two people at random **without replacement.**

(a) P(both Rick and Morty are in the sample)

= P(first Rick, then Morty) + P(first Morty, then Rick)

= (1/100) * (1/99) + (1/100) * (1/99) = 0.0002

(b) P(neither Rick nor Morty is in the sample)

= (98/100) * (97/99) = 0.9602

# SAMPLING

# RANDOM SAMPLES

- Deterministic sample:
  - Sampling scheme doesn't involve chance

- Random sample:
  - Before the sample is drawn, you have to know the selection probability of every group of people in the population
  - NOTE: Not all individuals / groups have to have equal chance of being selected

# PROBABILITY SAMPLES

- A Random sample is a probability sample, because we know the probability of every individual in the sample
- Recap of terminology:
  - ***Individual***: study subjects, typically, what a row of data will contain
- New terminology
  - A ***population*** is the set of all elements from whom a sample will be drawn.
  - A ***probability sample*** is one for which it is possible to calculate, before the sample is drawn, the chance with which any subset of elements will enter the sample.
- In a probability sample, all elements don't need to have the same chance of being chosen.

# SYSTEMATIC SAMPLE
# AN EXAMPLE OF PROBABILISTIC SAMPLE

- Suppose we wanted to sample elements from a sequence
- One method of sampling is to start by choosing a random position early in the list, and then evenly space positions after that.
- The sample consists of the elements in those positions.
- Such a sample is called a systematic sample.

(Demo – notebook 5.2,
Random sampling)

# SAMPLING WITH OR WITHOUT REPLACEMENT

- Random sampling **with** replacement, same individual/element **can** be sampled multiple times.
  - This is the default behavior of *np.random.choice* when it samples from an array.
- Random sampling **without** replacement, AKA, "simple random sample", same individual/element **cannot** be sampled multiple times
  - because it (individual/element) is replaced into the population after it has been sampled.

# SAMPLE OF CONVENIENCE

- Example: sample consists of whoever walks by
- Just because you think you're sampling "randomly", doesn't mean you have a random sample.
- If you can't figure out ahead of time
  - what's the population
  - what's the chance of selection, for each group in the population

then you don't have a random sample

# DISTRIBUTIONS

# PROBABILITY DISTRIBUTION

- Random quantity with various possible values
- "Probability distribution":
  - All the possible values of the quantity
  - The probability of each of those values
- If you can do the math, you can work out the probability distribution without ever simulating it
- But… simulation is often easier!

(Demo – notebook 5.2, Distributions)

# EMPIRICAL DISTRIBUTION

- "Empirical": based on observations
- Observations can be from repetitions of an experiment
- "Empirical Distribution"
  - All observed values
  - The proportion of times each value appears

(Demo – notebook 5.2, Distributions)

# LARGE RANDOM SAMPLES

# LAW OF AVERAGES / LAW OF LARGE NUMBERS

- If a **chance** experiment is **repeated many times**, ***independently and under the same conditions***, then the proportion of times that an event occurs gets closer to the theoretical probability of the event

- As you increase the number of rolls of a die, the proportion of times you see the face with five spots gets closer to 1/6

# EMPIRICAL DISTRIBUTION OF A SAMPLE

- If the **sample size is large**, then the ***empirical distribution of a uniform random sample*** **resembles the distribution of the population**, with high probability

(Demo – notebook 5.2,
Large Random Samples)

# A STATISTIC

# INFERENCE

- **Statistical Inference:**

  Making conclusions based on data in random samples
- **Example**:

fixed

Use the data to guess the value of an unknown number

depends on the random sample

Create an estimate of the unknown quantity

# TERMINOLOGY

- Parameter
  - A number associated with the population
- Statistic
  - A number calculated from the sample

A statistic can be used as an estimate of a parameter

# PROBABILITY DISTRIBUTION OF A STATISTIC

- Values of a statistic vary because random samples vary
- "Sampling distribution" or "probability distribution" of the statistic:
  - All possible values of the statistic,
  - and all the corresponding probabilities
- Can be hard to calculate
  - Either have to do the math
  - Or have to generate all possible samples and calculate the statistic based on each sample

(Demo – notebook 5.2, Simulating statistics)

# EMPIRICAL DISTRIBUTION OF A STATISTIC

- Empirical distribution of the statistic:
  - Based on simulated values of the statistic
  - Consists of all the observed values of the statistic
  - and the proportion of times each value appeared

- Good approximation to the probability distribution of the statistic
  - if the number of repetitions in the simulation is large

(Demo – notebook 5.2,
Simulating statistics )

# QUESTIONS?