A Computational Review - Stats 100 A

Andrew Lizarraga

Department of Statistics and Data Science

April 16 & 18, 2024

A Quick Note

- These lectures will review the computation aspects of what you've previously have learned in this course.
- This is not a computational course, so no need to worry if you don't understand all of this. **You will not be tested on this**.

Warm Up

Problem 1:

I flip a coin 2 times. What's the probability I get 50% H?

Warm Up

Problem 2:

I flip a coin 10 times. What's the probability I get 50% H?

Warm Up

Problem 3:

I flip a coin 100 times. What's the probability I get 50% H?

Warm Up

Problem 4:

- We simulate a random walk by flipping a fair coin.
- We start at $X_0 = 0$ on a number line.
- If we flip H, we move $+1$ to the right. If we flip T, we move -1 to the left.
- Suppose I take the cumulative sum of $+1$'s and -1's and I plot this graph for each time-step (i.e. each flip of the coin). What would you expect the graph to look like?

Simulating A Random Walk

- What is the distribution of the coin flip?
- What is the distribution of the random walk?

Simulating A Random Walk

```r
# Simulate coin flips
coin_flips <- 2 * rbinom(n = 100, size = 1, prob = 0.5) - 1

```

Simulating A Random Walk

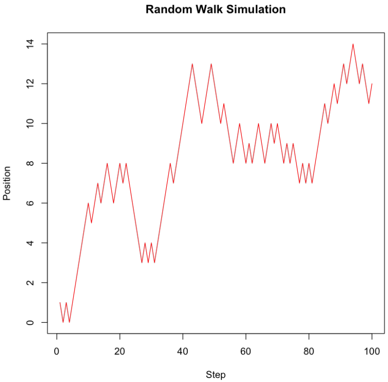
```r
# Simulate coin flips
coin_flips <- 2 * rbinom(n = 100, size = 1, prob = 0.5) - 1

# Calculate cumulative sum to simulate the random walk
summing <- cumsum(coin_flips)
```

Simulating A Random Walk

```r
# Simulate coin flips
coin_flips <- 2 * rbinom(n = 100, size = 1, prob = 0.5) - 1

# Calculate cumulative sum to simulate the random walk
summing <- cumsum(coin_flips)

# Plot the random walk
plot(summing, type = "l", col = 'red', xlab = "Step", ylab = "Position",
     main = "Random Walk Simulation")
```

Simulating A Random Walk

```r
# Simulate coin flips
coin_flips <- 2 * rbinom(n = 100, size = 1, prob = 0.5) - 1

# Calculate cumulative sum to simulate the random walk
summing <- cumsum(coin_flips)

# Plot the random walk
plot(summing, type = "l", col = 'red', xlab = "Step", ylab = "Position",
     main = "Random Walk Simulation")
```



Simulating A Random Walk

```r
simple_random_walk <- function(steps) {








}

# Generate a simple random walk of 100 steps
ls <- simple_random_walk(100)

# Plot the random walk
plot(ls, type = "l", col = 'red', xlab = "Step", ylab = "Position",
main = "Random Walk Simulation")
```

Simulating A Random Walk

```r
simple_random_walk <- function(steps) {
  s <- 0
  res <- numeric(steps)





}

# Generate a simple random walk of 100 steps
ls <- simple_random_walk(100)

# Plot the random walk
plot(ls, type = "l", col = 'red', xlab = "Step", ylab = "Position",
main = "Random Walk Simulation")
```

Simulating A Random Walk

```r
simple_random_walk <- function(steps) {
  s <- 0
  res <- numeric(steps)
  for (i in 1:steps) {
    s <- s + sample(c(-1, 1), size = 1)
    res[i] <- s
  }

}

# Generate a simple random walk of 100 steps
ls <- simple_random_walk(100)

# Plot the random walk
plot(ls, type = "l", col = 'red', xlab = "Step", ylab = "Position",
main = "Random Walk Simulation")
```

Simulating A Random Walk

```r
simple_random_walk <- function(steps) {
  s <- 0
  res <- numeric(steps)
  for (i in 1:steps) {
    s <- s + sample(c(-1, 1), size = 1)
    res[i] <- s
  }
  return(res)
}

# Generate a simple random walk of 100 steps
ls <- simple_random_walk(100)

# Plot the random walk
plot(ls, type = "l", col = 'red', xlab = "Step", ylab = "Position",
main = "Random Walk Simulation")
```

Short Break / Questions

Rainy or Sunny?

- If it rains today, then it rains tomorrow with probability α.
- If it is sunny today, then it rains tomorrow with probability β.
- If it is sunny today, (i.e. $X_0 = R$) then what is the probability it rains two days from now? (i.e. $P(X_2 = R) =?$)

Rainy or Sunny?

Consider the following two possibilities:

Case 1:

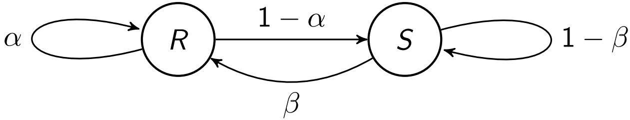
- $R \to R \to R$
- $R \xrightarrow{\alpha} R \xrightarrow{\alpha} R$
- Prob $= \alpha^2$

Case 2:

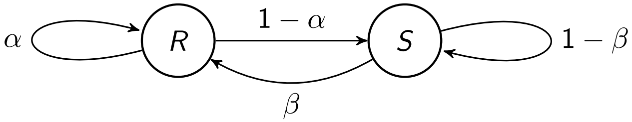
- $R \longrightarrow S \to R$
- $R \xrightarrow{1-\alpha} S \xrightarrow{\beta} R$
- Prob $= (1-\alpha)\beta$

Thus $P(X_2 = R) = P(\{(R, R, R)\} \cup \{(R, S, R)\})$
$= P((R, R, R)) + P((R, S, R)) = \alpha^2 + (1-\alpha)\beta$

Rainy or Sunny?

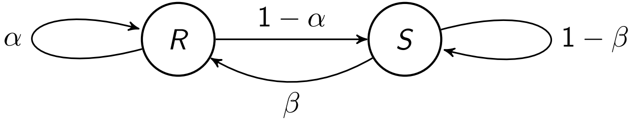


Rainy or Sunny?



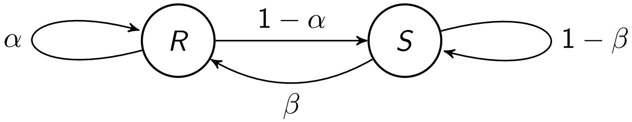
- Recall that we make express the transitions between R and S as conditional probabilities.

Rainy or Sunny?



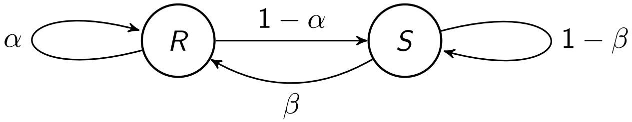
- Recall that we make express the transitions between R and S as conditional probabilities.
- $P(X_1 = R|X_0 = R) = \alpha$

Rainy or Sunny?



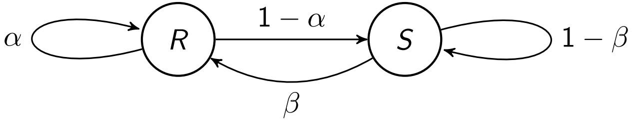
- Recall that we make express the transitions between R and S as conditional probabilities.
- $P(X_1 = R | X_0 = R) = \alpha$
- $P(X_1 = R | X_0 = S) = \beta$

Rainy or Sunny?



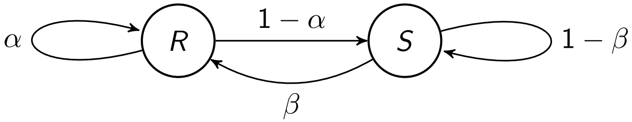
- Recall that we make express the transitions between R and S as conditional probabilities.
- $P(X_1 = R|X_0 = R) = \alpha$
- $P(X_1 = R|X_0 = S) = \beta$
- We discovered that: $P(X_2 = R|X_0 = R) = \alpha^2 + (1-\alpha)\beta$

Rainy or Sunny?



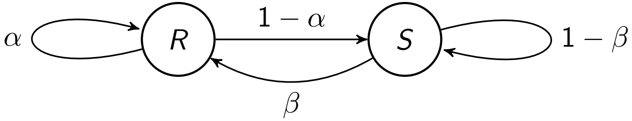
- Let's think about the problem in reverse.

Rainy or Sunny?



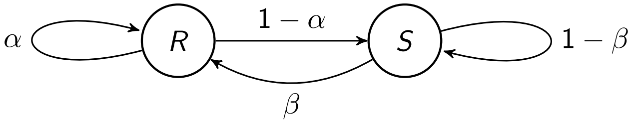
- Let's think about the problem in reverse.
- $P(X_2 = R | X_1 = R) = \alpha$ and $P(X_1 = R) = \alpha$

Rainy or Sunny?



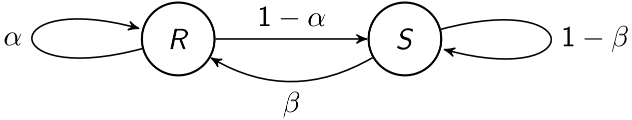
- Let's think about the problem in reverse.
- $P(X_2 = R | X_1 = R) = \alpha$ and $P(X_1 = R) = \alpha$
- $P(X_2 = R | X_1 = S) = \beta$ and $P(X_1 = S) = 1 - \alpha$

Rainy or Sunny?



- Let's think about the problem in reverse.
- $P(X_2 = R | X_1 = R) = \alpha$ and $P(X_1 = R) = \alpha$
- $P(X_2 = R | X_1 = S) = \beta$ and $P(X_1 = S) = 1 - \alpha$
- Now by the **law of total probability:**

Rainy or Sunny?



- Let's think about the problem in reverse.
- $P(X_2 = R|X_1 = R) = \alpha$ and $P(X_1 = R) = \alpha$
- $P(X_2 = R|X_1 = S) = \beta$ and $P(X_1 = S) = 1 - \alpha$
- Now by the **law of total probability:**

$$P(X_2 = R|X_0 = R) = P(X_2 = R|X_1 = R)P(X_1 = R) + P(X_2 = R|X_1 = S)P(X_1 = S)$$

Raining in the Future?

Problem 5:

Give it rains today ($X_0 = R$), we want to know the probability it rains 7 days from now $P(X_7 = R)$. How many different 7 day sequence are there, s.t. $X_0 = R$?

Raining in the Future?

Problem 5:

Give it rains today ($X_0 = R$), we want to know the probability it rains 7 days from now $P(X_7 = R)$. How many different 7 day sequence are there, s.t. $X_0 = R$?

- There are $2^7 = 128$ sequence that start with $X_0 = R$.

Raining in the Future?

Problem 5:

Give it rains today ($X_0 = R$), we want to know the probability it rains 7 days from now $P(X_7 = R)$. How many different 7 day sequence are there, s.t. $X_0 = R$?

- There are $2^7 = 128$ sequence that start with $X_0 = R$.
- This is too many to keep track of.

Raining in the Future?

Problem 5:

Give it rains today ($X_0 = R$), we want to know the probability it rains 7 days from now $P(X_7 = R)$. How many different 7 day sequence are there, s.t. $X_0 = R$?

- There are $2^7 = 128$ sequence that start with $X_0 = R$.
- This is too many to keep track of.
- We will review a better approach to this problem.

Raining in the Future?

- Denote the initial probability vector:
  $v_0 = [P(X_0 = R), P(X_0 = S)] = [1, 0]$

Raining in the Future?

- Denote the initial probability vector:
  $v_0 = [P(X_0 = R), P(X_0 = S)] = [1, 0]$
- $v_1 = [P(X_1 = R), P(X_1 = S)] = [\alpha, 1 - \alpha]$

Raining in the Future?

- Denote the initial probability vector: $v_0 = [P(X_0 = R), P(X_0 = S)] = [1, 0]$
- $v_1 = [P(X_1 = R), P(X_1 = S)] = [\alpha, 1 - \alpha]$
- $v_2 = [\alpha^2 + (1 - \alpha)\beta, 1 - (\alpha^2 + (1 - \alpha)\beta)]$

Raining in the Future?

- Denote the initial probability vector: $v_0 = [P(X_0 = R), P(X_0 = S)] = [1, 0]$
- $v_1 = [P(X_1 = R), P(X_1 = S)] = [\alpha, 1 - \alpha]$
- $v_2 = [\alpha^2 + (1 - \alpha)\beta, 1 - (\alpha^2 + (1 - \alpha)\beta)]$
- $\vdots$

Raining in the Future?

- Denote the initial probability vector:
  $v_0 = [P(X_0 = R), P(X_0 = S)] = [1, 0]$
- $v_1 = [P(X_1 = R), P(X_1 = S)] = [\alpha, 1 - \alpha]$
- $v_2 = [\alpha^2 + (1 - \alpha)\beta, 1 - (\alpha^2 + (1 - \alpha)\beta)]$
- $\vdots$
- $v_n = [P(X_n = R), P(X_n = S)]$

Raining in the Future?

The astute of you may recall that this update rule is due to **the law of total probability**. We may express the updates more generally as:

$$P(X_{n+1} = R) = P(X_{n+1} = R|X_n = R)P(X_n = R) + P(X_{n+1} = R|X_n = S)P(X_n = S)$$

$$P(X_{n+1} = S) = P(X_{n+1} = S|X_n = R)P(X_n = R) + P(X_{n+1} = S|X_n = S)P(X_n = S)$$

Raining in the Future?

The astute of you may recall that this update rule is due to **the law of total probability**. We may express the updates more generally as:

$$P(X_{n+1} = R) = P(X_{n+1} = R | X_n = R) P(X_n = R) + P(X_{n+1} = R | X_n = S) P(X_n = S)$$

$$P(X_{n+1} = S) = P(X_{n+1} = S | X_n = R) P(X_n = R) + P(X_{n+1} = S | X_n = S) P(X_n = S)$$

Note:

The probabilities on day $n+1$ are a **linear combination** of the probabilities on day n. What's so special about this?

Raining in the Future?

Since the next day probabilities are linear combinations of the previous day probabilities, we may represent the update rule via matrix multiplication:

$$
\begin{aligned}
&[P(X_{n+1} = R), P(X_{n+1} = S)] = \\
&\qquad [P(X_n = R), P(X_n = S)] \begin{bmatrix} P(X_{n+1} = R | X_n = R) & P(X_{n+1} = S | X_n = R) \\ P(X_{n+1} = R | X_n = S) & P(X_{n+1} = S | X_n = S) \end{bmatrix}
\end{aligned}
$$

Raining in the Future?

Since the next day probabilities are linear combinations of the previous day probabilities, we may represent the update rule via matrix multiplication:

$$\begin{aligned}[P(X_{n+1} = R), P(X_{n+1} = S)] = \\ [P(X_n = R), P(X_n = S)] \begin{bmatrix} P(X_{n+1} = R | X_n = R) & P(X_{n+1} = S | X_n = R) \\ P(X_{n+1} = R | X_n = S) & P(X_{n+1} = S | X_n = S) \end{bmatrix}\end{aligned}$$

$$v_{n+1} = v_n \begin{bmatrix} \alpha & 1 - \alpha \\ \beta & 1 - \beta \end{bmatrix}$$

Raining in the Future?

Since the next day probabilities are linear combinations of the previous day probabilities, we may represent the update rule via matrix multiplication:

$$[P(X_{n+1} = R), P(X_{n+1} = S)] = \\ [P(X_n = R), P(X_n = S)] \begin{bmatrix} P(X_{n+1} = R|X_n = R) & P(X_{n+1} = S|X_n = R) \\ P(X_{n+1} = R|X_n = S) & P(X_{n+1} = S|X_n = S) \end{bmatrix}$$

$$v_{n+1} = v_n \begin{bmatrix} \alpha & 1-\alpha \\ \beta & 1-\beta \end{bmatrix}$$

$$v_{n+1} = v_n K$$

Raining in the Future?

What Now?

How does knowing the matrix update rule for the probabilities help us determine $P(X_7 = R)$?

- $v_{n+1} = v_n K$

Raining in the Future?

What Now?

How does knowing the matrix update rule for the probabilities help us determine $P(X_7 = R)$?

- $v_{n+1} = v_n K$
- $v_7 = v_6 K$

Raining in the Future?

What Now?

How does knowing the matrix update rule for the probabilities help us determine $P(X_7 = R)$?

- $v_{n+1} = v_n K$
- $v_7 = v_6 K$
- $v_7 = (v_5 K)K = v_5 K^2$

Raining in the Future?

What Now?

How does knowing the matrix update rule for the probabilities help us determine $P(X_7 = R)$?

- $v_{n+1} = v_n K$
- $v_7 = v_6 K$
- $v_7 = (v_5 K)K = v_5 K^2$
- $\vdots$

Raining in the Future?

What Now?

How does knowing the matrix update rule for the probabilities help us determine $P(X_7 = R)$?

- $v_{n+1} = v_n K$
- $v_7 = v_6 K$
- $v_7 = (v_5 K)K = v_5 K^2$
- $\vdots$
- $v_7 = v_0 K^7 = [1, 0]K^7$

Simulating Rain Prediction

Let's say $\alpha = 2/3$, $\beta = 1/2$.

```python
import numpy as np
import matplotlib.pyplot as plt

# Parameters that effect chances of rain/sun
alpha = 2/3
beta = 1/2
v0 = np.array([1., 0. ])
```

Simulating Rain Prediction

Let's say $\alpha = 2/3$, $\beta = 1/2$.

```python
import numpy as np
import matplotlib.pyplot as plt

# Parameters that effect chances of rain/sun
alpha = 2/3
beta = 1/2
v0 = np.array([1., 0. ])

# The transition matrix
K = np.array([[alpha, 1 - alpha],
              [beta, 1 - beta]])
```

Simulating Rain Prediction

Let's say $\alpha = 2/3$, $\beta = 1/2$.

```python
import numpy as np
import matplotlib.pyplot as plt

# Parameters that effect chances of rain/sun
alpha = 2/3
beta = 1/2
v0 = np.array([1., 0. ])

# The transition matrix
K = np.array([[alpha, 1 - alpha],
              [beta, 1 - beta]])

# Loop for the first 8 days
v_list = []
for n in range(0,8):
    print("----")
    print("Day: ", n)
    v = v0 @ np.linalg.matrix_power(K, n)
    print("probabilities: ", v)
    v_list.append(v)

plt.plot(v_list)
plt.show()
```

Thank You / Questions

Important Note:

You definitely want to have your computers at hand next lecture. Make sure you have **Python** (or R) installed and use **pip** to install **numpy** for Python.

It will also help to have a **jupyter** compatible reader. Typically **jupyter notebook**, though this is not required.

Contact:

Prof. Ying Nian Wu may not be immediately available. You may contact at: **andrewlizarraga@g.ucla.edu** for the duration of this week.

Welcome Back!

- Last lecture we went over simulating a random walk and simulating simple weather model via a Markov Chain.
- In this lecture, we focus on more simulation and we involve more code.
- Before we do this, let's warm up.

Warm Up

Problem 1:

What is a Monte Carlo simulation?

Warm Up

Problem 2:
What is a Markov Chain?

Warm Up

Problem 3:

Let's compute 2^8. About how many multiplications is required for this computation?

Warm Up

Problem 3:

Let's compute 2^{100}. About how many multiplications is required for this computation?

Answer:

Naively you would think that 2^8 requires 8 multiplications. But you can actually perform it in far less:

$2 \times 2 = 4$, $4 \times 4 = 16$, $16 \times 16 = 256 = 2^8$. Only 3 multiplications.

Warm Up

Problem 4:

You flip a fair coin until a H comes up. About how many flips would you expect to make?

Warm Up

Problem 4:

You flip a fair coin until a H comes up. About how many flips would you expect to make?

Answer:

This is a geometric distribution. So $E[H] = \frac{1}{\frac{1}{2}} = 2$. Also it's intuitive that you would expect to see a H among 2 coin flips.

Warm Up

Problem 6:

You flip a fair coin until you see two H in a row. About how many flips would you expect to make?

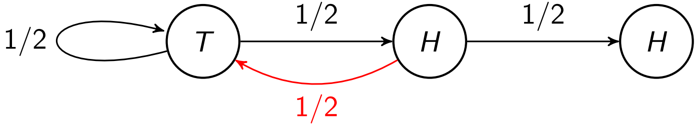
Warm Up

Problem 7:

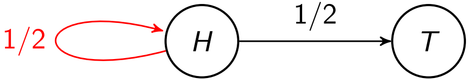
You flip a fair coin until you see a H followed by a T, i.e. the sequence HT. About how many flips would you expect to make? Is it a different number than the expected amount of flips for HH?

Warm Up

Case HH:
Suppose we just received a H. Now if we flip a coin and fail to get H (i.e. T), then at best we are 2 flips away from obtaining HH.



Case HT:
Suppose we just received a H. Now if we flip a coin and fail to get T (i.e. H), then at best we are 1 flips away from obtaining HT.



The Need To Simulate

As we saw with the some of the previous problems, seemingly simple problems that we feel certain about might be somewhat more complex than we would hope for.

To combat our easily tricked intuitions, it usually helps to run simulations in order to witness the qualitative and numerical behavior of a system.

Note:

Now is the time to open up **python files** or the **jupyter notebook**.

They should be on `https://bruinlearn.ucla.edu/`.

If you can't find them there. You can get them from my site:
`https://drewrl3v.github.io/teaching/spr24_stats100a/`

Short Break / Questions

Estimating Pi

Take a unit circle inscribed in a square. Uniformly sample a point in the square and take the number of points that land in the circle and divide it by the number of point that land outside the circle (and scale appropriately).

Estimating Pi

Take a unit circle inscribed in a square. Uniformly sample a point in the square and take the number of points that land in the circle and divide it by the number of point that land outside the circle (and scale appropriately).

```
import numpy as np
```

Estimating Pi

Take a unit circle inscribed in a square. Uniformly sample a point in the square and take the number of points that land in the circle and divide it by the number of point that land outside the circle (and scale appropriately).

```
import numpy as np
def est_pi(num_sims):
    count_in = 0
```

Estimating Pi

Take a unit circle inscribed in a square. Uniformly sample a point in the square and take the number of points that land in the circle and divide it by the number of point that land outside the circle (and scale appropriately).

```
import numpy as np
def est_pi(num_sims):
    count_in = 0
    for _ in range(num_sims):
        x, y = np.random.uniform(-1.0, 1.0, size=2)
```

Estimating Pi

Take a unit circle inscribed in a square. Uniformly sample a point in the square and take the number of points that land in the circle and divide it by the number of point that land outside the circle (and scale appropriately).

```
import numpy as np
def est_pi(num_sims):
    count_in = 0
    for _ in range(num_sims):
        x, y = np.random.uniform(-1.0, 1.0, size=2)
        if x**2 + y**2 < 1:
            count_in += 1


est_pi(10000)
```

Estimating Pi

Take a unit circle inscribed in a square. Uniformly sample a point in the square and take the number of points that land in the circle and divide it by the number of point that land outside the circle (and scale appropriately).

```
import numpy as np
def est_pi(num_sims):
    count_in = 0
    for _ in range(num_sims):
        x, y = np.random.uniform(-1.0, 1.0, size=2)
        if x**2 + y**2 < 1:
            count_in += 1
    return 4 * (count_in / num_sims)

est_pi(10000)
```

Number of Flips Until HH

Run a bunch of simulations and keep note of the number of flips until we see *HH* in each run. Sum all of these totals and divide by the number of simulations you ran.

```
import numpy as np
def simulate_HH(num_sims):
    number_of_flips_per_trial = []
```

Number of Flips Until HH

Run a bunch of simulations and keep note of the number of flips until we see *HH* in each run. Sum all of these totals and divide by the number of simulations you ran.

```
import numpy as np
def simulate_HH(num_sims):
    number_of_flips_per_trial = []
    for _ in range(num_sims):
        saw_HH = False
        trial = []
```

Number of Flips Until HH

Run a bunch of simulations and keep note of the number of flips until we see *HH* in each run. Sum all of these totals and divide by the number of simulations you ran.

```python
import numpy as np
def simulate_HH(num_sims):
    number_of_flips_per_trial = []
    for _ in range(num_sims):
        saw_HH = False
        trial = []
        while not saw_HH:
            # Lets say H = 1, T = 0
            flip = np.random.binomial(n=1, p=1/2)
```

Number of Flips Until HH

Run a bunch of simulations and keep note of the number of flips until we see *HH* in each run. Sum all of these totals and divide by the number of simulations you ran.

```
import numpy as np
def simulate_HH(num_sims):
    number_of_flips_per_trial = []
    for _ in range(num_sims):
        saw_HH = False
        trial = []
        while not saw_HH:
            # Lets say H = 1, T = 0
            flip = np.random.binomial(n=1, p=1/2)
            if len(trial) >= 1 and trial[-1] == 1 and flip == 1:
                saw_HH = True
```

Number of Flips Until HH

Run a bunch of simulations and keep note of the number of flips until we see *HH* in each run. Sum all of these totals and divide by the number of simulations you ran.

```python
import numpy as np
def simulate_HH(num_sims):
    number_of_flips_per_trial = []
    for _ in range(num_sims):
        saw_HH = False
        trial = []
        while not saw_HH:
            # Lets say H = 1, T = 0
            flip = np.random.binomial(n=1, p=1/2)
            if len(trial) >= 1 and trial[-1] == 1 and flip == 1:
                saw_HH = True
            trial.append(flip)

        number_of_flips_per_trial.append(len(trial))
    return sum(number_of_flips_per_trial) / num_sims

simulate_HH(50)
```

Number of Flips Until HT

Run a bunch of simulations and keep note of the number of flips until we see HT in each run. Sum all of these totals and divide by the number of simulations you ran.

Number of Flips Until HT

Run a bunch of simulations and keep note of the number of flips until we see HT in each run. Sum all of these totals and divide by the number of simulations you ran.

```python
import numpy as np
def simulate_HT(num_sims):
    number_of_flips_per_trial = []
    for _ in range(num_sims):
        saw_HT = False
        trial = []
        while not saw_HT:
            # Lets say H = 1, T = 0
            flip = np.random.binomial(n=1, p=1/2)
            if len(trial) >= 1 and trial[-1] == 1 and flip == 0:
                saw_HT = True
            trial.append(flip)

        number_of_flips_per_trial.append(len(trial))
    return sum(number_of_flips_per_trial) / num_sims

simulate_HT(1000)
```

Thank You / Questions

Contact:

Prof. Ying Nian Wu may not be immediately available. You may contact at: **andrewlizarraga@g.ucla.edu** for the duration of this week.

Note:

The lecture material for this week should be available on `https://bruinlearn.ucla.edu/`.

If you can't find them there. You can get them from my site: `https://drewrl3v.github.io/teaching/spr24_stats100a/`