A Computational Review - Stats 100 A

Andrew Lizarraga

Department of Statistics and Data Science

April 16, 2024

A Quick Note

Note:

This lecture will review the computation aspects of what you've previously have learned in this course.

Now is the time to open up your text editor. We will be reviewing very basic **R**.

The supplemental **R** files should be available here:
`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/`

Install R and RStudio

macOS & Windows Install:

https://posit.co/download/rstudio-desktop/.

If you have a package manager:

- **Windows:**
- winget install -e –id RProject.R
- winget install -e –id RStudio.RStudio.OpenSource
- **macOS:**
- brew install r
- brew install –cask rstudio

Generate Uniform Random Numbers

```
# Generate 1000 uniform random numbers between 0 and 1


# Plot the histogram of the random numbers


```

Generate Uniform Random Numbers

```r
# Generate 1000 uniform random numbers between 0 and 1
random_numbers <- runif(1000, min = 0, max = 1)

# Plot the histogram of the random numbers


```

Generate Uniform Random Numbers

```r
# Generate 1000 uniform random numbers between 0 and 1
random_numbers <- runif(1000, min = 0, max = 1)

# Plot the histogram of the random numbers
hist(random_numbers, main = "Histogram of Uniform Random Numbers", xlab = "Value",
ylab = "Frequency",col = "lightblue", border = "blue")
```

Uniformly Random Scatter Plot

```
# Number of times to repeat generating the two numbers


# Generate n uniform random numbers for X and Y



# Plot the scatterplot of X vs Y


```

Uniformly Random Scatter Plot

```r
# Number of times to repeat generating the two numbers
n <- 1000

# Generate n uniform random numbers for X and Y



# Plot the scatterplot of X vs Y


```

Uniformly Random Scatter Plot

```
# Number of times to repeat generating the two numbers
n <- 1000

# Generate n uniform random numbers for X and Y
X <- runif(n, min = 0, max = 1)
Y <- runif(n, min = 0, max = 1)

# Plot the scatterplot of X vs Y


```

Uniformly Random Scatter Plot

```r
# Number of times to repeat generating the two numbers
n <- 1000

# Generate n uniform random numbers for X and Y
X <- runif(n, min = 0, max = 1)
Y <- runif(n, min = 0, max = 1)

# Plot the scatterplot of X vs Y
plot(X, Y, main = "Scatterplot of Independent Uniform Random Numbers",
     xlab = "X Values", ylab = "Y Values", col = "blue", pch = 19)
```

Estimating Pi

```
# Number of points to generate


# Generate n uniform random numbers for X and Y in the range [-1, 1]



# Count how many points fall inside the unit circle


# Estimate Pi


# Print the estimate

```

Estimating Pi

```r
# Number of points to generate
n <- 10000

# Generate n uniform random numbers for X and Y in the range [-1, 1]



# Count how many points fall inside the unit circle


# Estimate Pi


# Print the estimate

```

Estimating Pi

```r
# Number of points to generate
n <- 10000

# Generate n uniform random numbers for X and Y in the range [-1, 1]
X <- runif(n, min = -1, max = 1)
Y <- runif(n, min = -1, max = 1)

# Count how many points fall inside the unit circle


# Estimate Pi


# Print the estimate

```

Estimating Pi

```r
# Number of points to generate
n <- 10000

# Generate n uniform random numbers for X and Y in the range [-1, 1]
X <- runif(n, min = -1, max = 1)
Y <- runif(n, min = -1, max = 1)

# Count how many points fall inside the unit circle
points_inside <- sum(X^2 + Y^2 < 1)

# Estimate Pi


# Print the estimate

```

Estimating Pi

```r
# Number of points to generate
n <- 10000

# Generate n uniform random numbers for X and Y in the range [-1, 1]
X <- runif(n, min = -1, max = 1)
Y <- runif(n, min = -1, max = 1)

# Count how many points fall inside the unit circle
points_inside <- sum(X^2 + Y^2 < 1)

# Estimate Pi
pi_estimate <- (points_inside / n) * 4

# Print the estimate

```

Estimating Pi

```r
# Number of points to generate
n <- 10000

# Generate n uniform random numbers for X and Y in the range [-1, 1]
X <- runif(n, min = -1, max = 1)
Y <- runif(n, min = -1, max = 1)

# Count how many points fall inside the unit circle
points_inside <- sum(X^2 + Y^2 < 1)

# Estimate Pi
pi_estimate <- (points_inside / n) * 4

# Print the estimate
print(pi_estimate)
```

Flipping A Fair Coin

```
# Generate a uniform random number U between 0 and 1


# Determine the value of Z based on U


# Print the result

```

Flipping A Fair Coin

```r
# Generate a uniform random number U between 0 and 1
U <- runif(1, min = 0, max = 1)

# Determine the value of Z based on U


# Print the result

```

Flipping A Fair Coin

```r
# Generate a uniform random number U between 0 and 1
U <- runif(1, min = 0, max = 1)

# Determine the value of Z based on U
Z <- ifelse(U < 0.5, 0, 1)

# Print the result

```

Flipping A Fair Coin

```r
# Generate a uniform random number U between 0 and 1
U <- runif(1, min = 0, max = 1)

# Determine the value of Z based on U
Z <- ifelse(U < 0.5, 0, 1)

# Print the result
print(Z)
```

Flipping Many Coins

```
# Number of coin flips


# Generate n uniform random numbers U between 0 and 1


# Determine the value of Z for each U


# Print the results

```

Flipping Many Coins

```r
# Number of coin flips
n <- 10

# Generate n uniform random numbers U between 0 and 1


# Determine the value of Z for each U


# Print the results

```

Flipping Many Coins

```
# Number of coin flips
n <- 10

# Generate n uniform random numbers U between 0 and 1
U <- runif(n, min = 0, max = 1)

# Determine the value of Z for each U


# Print the results

```

Flipping Many Coins

```r
# Number of coin flips
n <- 10

# Generate n uniform random numbers U between 0 and 1
U <- runif(n, min = 0, max = 1)

# Determine the value of Z for each U
Z <- ifelse(U < 0.5, 0, 1)

# Print the results

```

Flipping Many Coins

```r
# Number of coin flips
n <- 10

# Generate n uniform random numbers U between 0 and 1
U <- runif(n, min = 0, max = 1)

# Determine the value of Z for each U
Z <- ifelse(U < 0.5, 0, 1)

# Print the results
print(Z)
```

Averaging Coin Flips

```python
# Number of coins to flip in each experiment


# Number of experiments


# Generate m experiments of n coin flips, where each flip is represented by a uniform
↪   random number < 0.5 (head) or >= 0.5 (tail)


# Sum the number of heads (1s) in each experiment to get X


# Plot the histogram of X


# Plot the histogram of X/n


```

Averaging Coin Flips

```r
# Number of coins to flip in each experiment
n <- 10

# Number of experiments


# Generate m experiments of n coin flips, where each flip is represented by a uniform
    random number < 0.5 (head) or >= 0.5 (tail)


# Sum the number of heads (1s) in each experiment to get X


# Plot the histogram of X



# Plot the histogram of X/n


```

Averaging Coin Flips

```r
# Number of coins to flip in each experiment
n <- 10

# Number of experiments
m <- 1000

# Generate m experiments of n coin flips, where each flip is represented by a uniform random number < 0.5 (head) or >= 0.5 (tail)


# Sum the number of heads (1s) in each experiment to get X


# Plot the histogram of X



# Plot the histogram of X/n


```

Averaging Coin Flips

```r
# Number of coins to flip in each experiment
n <- 10

# Number of experiments
m <- 1000

# Generate m experiments of n coin flips, where each flip is represented by a uniform
↪  random number < 0.5 (head) or >= 0.5 (tail)
coin_flips <- matrix(runif(n * m, min = 0, max = 1) < 0.5, nrow = m, ncol = n)

# Sum the number of heads (1s) in each experiment to get X


# Plot the histogram of X



# Plot the histogram of X/n


```

Averaging Coin Flips

```r
# Number of coins to flip in each experiment
n <- 10

# Number of experiments
m <- 1000

# Generate m experiments of n coin flips, where each flip is represented by a uniform random number < 0.5 (head) or >= 0.5 (tail)
coin_flips <- matrix(runif(n * m, min = 0, max = 1) < 0.5, nrow = m, ncol = n)

# Sum the number of heads (1s) in each experiment to get X
X <- rowSums(coin_flips)

# Plot the histogram of X



# Plot the histogram of X/n


```

Averaging Coin Flips

```
# Number of coins to flip in each experiment
n <- 10

# Number of experiments
m <- 1000

# Generate m experiments of n coin flips, where each flip is represented by a uniform
↪  random number < 0.5 (head) or >= 0.5 (tail)
coin_flips <- matrix(runif(n * m, min = 0, max = 1) < 0.5, nrow = m, ncol = n)

# Sum the number of heads (1s) in each experiment to get X
X <- rowSums(coin_flips)

# Plot the histogram of X
hist(X, main = "Histogram of Number of Heads (X)", xlab = "Number of Heads",
ylab = "Frequency", col = "lightblue", border = "blue")

# Plot the histogram of X/n
```

Averaging Coin Flips

```r
# Number of coins to flip in each experiment
n <- 10

# Number of experiments
m <- 1000

# Generate m experiments of n coin flips, where each flip is represented by a uniform random number < 0.5 (head) or >= 0.5 (tail)
coin_flips <- matrix(runif(n * m, min = 0, max = 1) < 0.5, nrow = m, ncol = n)

# Sum the number of heads (1s) in each experiment to get X
X <- rowSums(coin_flips)

# Plot the histogram of X
hist(X, main = "Histogram of Number of Heads (X)", xlab = "Number of Heads",
ylab = "Frequency", col = "lightblue", border = "blue")

# Plot the histogram of X/n
hist(X/n, main = "Histogram of Proportion of Heads (X/n)", xlab = "Proportion of Heads",
ylab = "Frequency", col = "lightgreen", border = "darkgreen")
```

A Random Walk

```
# Total number of steps


# Generate uniform random numbers


# Generate Z: -1 if U < 0.5, 1 otherwise


# Initialize X


# Compute X_t for each step






# Plot the trajectory of the random walk


# Plot a histogram of the final positions


```

A Random Walk

```r
# Total number of steps
t <- 100

# Generate uniform random numbers


# Generate Z: -1 if U < 0.5, 1 otherwise


# Initialize X


# Compute X_t for each step




# Plot the trajectory of the random walk



# Plot a histogram of the final positions


```

A Random Walk

```r
# Total number of steps
t <- 100

# Generate uniform random numbers
U <- runif(t, min = 0, max = 1)

# Generate Z: -1 if U < 0.5, 1 otherwise


# Initialize X


# Compute X_t for each step




# Plot the trajectory of the random walk


# Plot a histogram of the final positions



```

A Random Walk

```r
# Total number of steps
t <- 100

# Generate uniform random numbers
U <- runif(t, min = 0, max = 1)

# Generate Z: -1 if U < 0.5, 1 otherwise
Z <- ifelse(U < 0.5, -1, 1)

# Initialize X


# Compute X_t for each step




# Plot the trajectory of the random walk



# Plot a histogram of the final positions


```

A Random Walk

```r
# Total number of steps
t <- 100

# Generate uniform random numbers
U <- runif(t, min = 0, max = 1)

# Generate Z: -1 if U < 0.5, 1 otherwise
Z <- ifelse(U < 0.5, -1, 1)

# Initialize X
X <- rep(0, t + 1)

# Compute X_t for each step




# Plot the trajectory of the random walk



# Plot a histogram of the final positions


```

A Random Walk

```r
# Total number of steps
t <- 100

# Generate uniform random numbers
U <- runif(t, min = 0, max = 1)

# Generate Z: -1 if U < 0.5, 1 otherwise
Z <- ifelse(U < 0.5, -1, 1)

# Initialize X
X <- rep(0, t + 1)

# Compute X_t for each step
for (i in 1:t) {
  X[i + 1] <- X[i] + Z[i]
}

# Plot the trajectory of the random walk



# Plot a histogram of the final positions


```

A Random Walk

```r
# Total number of steps
t <- 100

# Generate uniform random numbers
U <- runif(t, min = 0, max = 1)

# Generate Z: -1 if U < 0.5, 1 otherwise
Z <- ifelse(U < 0.5, -1, 1)

# Initialize X
X <- rep(0, t + 1)

# Compute X_t for each step
for (i in 1:t) {
  X[i + 1] <- X[i] + Z[i]
}

# Plot the trajectory of the random walk
plot(0:t, X, type = "l", main = "Trajectory of the Random Walk", xlab = "Time t",
ylab = "Position X", col = "blue")

# Plot a histogram of the final positions
```

A Random Walk

```r
# Total number of steps
t <- 100

# Generate uniform random numbers
U <- runif(t, min = 0, max = 1)

# Generate Z: -1 if U < 0.5, 1 otherwise
Z <- ifelse(U < 0.5, -1, 1)

# Initialize X
X <- rep(0, t + 1)

# Compute X_t for each step
for (i in 1:t) {
  X[i + 1] <- X[i] + Z[i]
}

# Plot the trajectory of the random walk
plot(0:t, X, type = "l", main = "Trajectory of the Random Walk", xlab = "Time t",
ylab = "Position X", col = "blue")

# Plot a histogram of the final positions
hist(X, main = "Histogram of Positions at Final Time Step", xlab = "Position X",
ylab = "Frequency", col = "lightgreen", border = "darkgreen")
```

Transformation Of Random Variable

```
# Number of random variables to generate


# Generate uniform random variables U


# Transform U to get exponential random variables X


# Plot histogram of X to visualize the exponential distribution


```

Transformation Of Random Variable

```r
# Number of random variables to generate
n <- 1000

# Generate uniform random variables U


# Transform U to get exponential random variables X


# Plot histogram of X to visualize the exponential distribution


```

Transformation Of Random Variable

```r
# Number of random variables to generate
n <- 1000

# Generate uniform random variables U
U <- runif(n, min = 0, max = 1)

# Transform U to get exponential random variables X


# Plot histogram of X to visualize the exponential distribution


```

Transformation Of Random Variable

```r
# Number of random variables to generate
n <- 1000

# Generate uniform random variables U
U <- runif(n, min = 0, max = 1)

# Transform U to get exponential random variables X


# Plot histogram of X to visualize the exponential distribution


```

Transformation Of Random Variable

```r
# Number of random variables to generate
n <- 1000

# Generate uniform random variables U
U <- runif(n, min = 0, max = 1)

# Transform U to get exponential random variables X
X <- -log(U)

# Plot histogram of X to visualize the exponential distribution


```

Transformation Of Random Variable

```r
# Number of random variables to generate
n <- 1000

# Generate uniform random variables U
U <- runif(n, min = 0, max = 1)

# Transform U to get exponential random variables X
X <- -log(U)

# Plot histogram of X to visualize the exponential distribution
hist(X, main = "Histogram of Exponential Random Variables", xlab = "X",
ylab = "Frequency", col = "lightblue", border = "blue", breaks = 50)
```

Central Limit Theorem & Law of Large Numbers

```r
# Number of trials
trials <- 10000

# Initialize vectors to store the results
mean_u <- numeric(trials)
clt_u <- numeric(trials)

# Number of observations (change this to see different effects)
n <- 30

# Simulation
for (i in 1:trials) {
  # Generate n uniform random numbers
  U <- runif(n, min = 0, max = 1)

  # Calculate the mean
  mean_u[i] <- mean(U)

  # Calculate for CLT
  clt_u[i] <- sqrt(n) * (mean_u[i] - 1/2)
}

# Plot the histogram of mean_u to demonstrate LLN
hist(mean_u, main = "LLN: Histogram of U-bar", xlab = "U-bar",
ylab = "Frequency", col = "lightblue", border = "blue", breaks = 30)

# Plot the histogram of clt_u to demonstrate CLT
hist(clt_u, main = "CLT: Histogram of sqrt(n) (U-bar - 1/2)",
xlab = "sqrt(n) (U-bar - 1/2)", ylab = "Frequency", col = "lightgreen",
border = "darkgreen", breaks = 30)
```

Appendix

Note:

The following Appendix is meant as a supplemental resource.

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?

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:

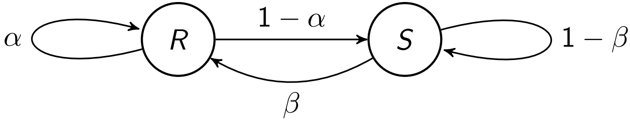
- $R \rightarrow R \rightarrow R$
- $R \xrightarrow{\alpha} R \xrightarrow{\alpha} R$
- Prob $= \alpha^2$

Case 2:

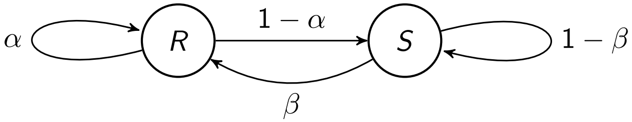
- $R \longrightarrow S \rightarrow R$
- $R \xrightarrow{1-\alpha} S \xrightarrow{\beta} R$
- Prob $= (1-\alpha)\beta$

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?

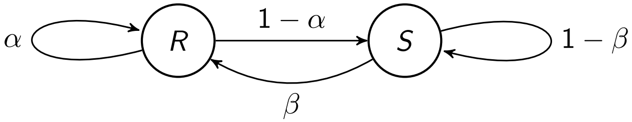


Rainy or Sunny?



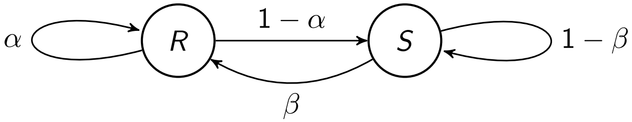
- Recall that we make express the transitions between R and S as conditional probabilities.

Rainy or Sunny?



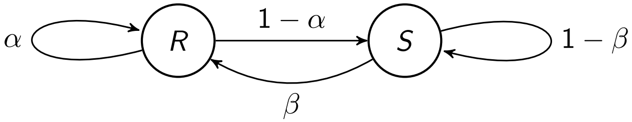
- Recall that we make express the transitions between R and S as conditional probabilities.
- $P(X_1 = R | X_0 = R) = \alpha$

Rainy or Sunny?



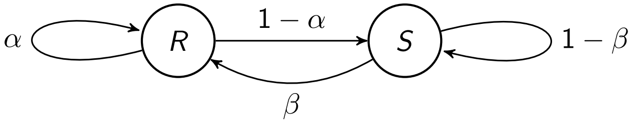
- Recall that we make express the transitions between R and S as conditional probabilities.
- $P(X_1 = R | X_0 = R) = \alpha$
- $P(X_1 = R | X_0 = S) = \beta$

Rainy or Sunny?



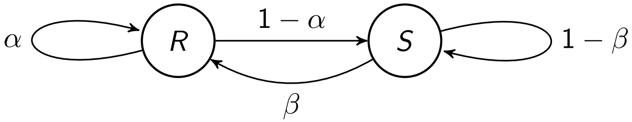
- Recall that we make express the transitions between R and S as conditional probabilities.
- $P(X_1 = R|X_0 = R) = \alpha$
- $P(X_1 = R|X_0 = S) = \beta$
- We discovered that: $P(X_2 = R|X_0 = R) = \alpha^2 + (1-\alpha)\beta$

Rainy or Sunny?



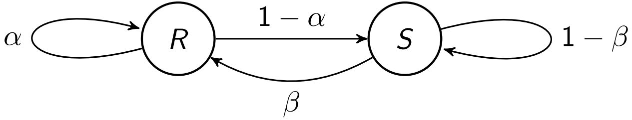
- Let's think about the problem in reverse.

Rainy or Sunny?



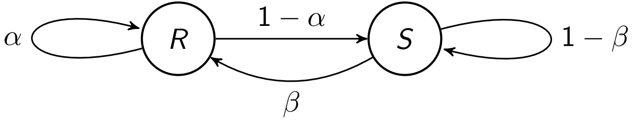
- Let's think about the problem in reverse.
- $P(X_2 = R | X_1 = R) = \alpha$ and $P(X_1 = R) = \alpha$

Rainy or Sunny?



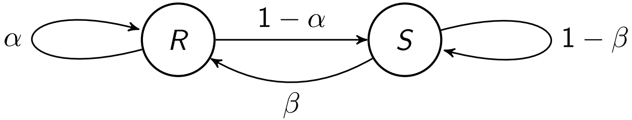
- Let's think about the problem in reverse.
- $P(X_2 = R|X_1 = R) = \alpha$ and $P(X_1 = R) = \alpha$
- $P(X_2 = R|X_1 = S) = \beta$ and $P(X_1 = S) = 1 - \alpha$

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:

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

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

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()
```

Recall!

- 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

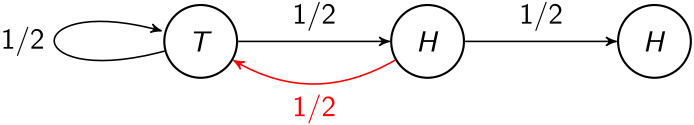
Problem 7:

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

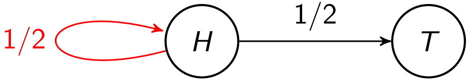
Case HH:

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.

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.

```python
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.

```
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.

```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
```

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
            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.

```
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/`