A Computational Review - Stats 100 A

Andrew Lizarraga

Department of Statistics and Data Science

July 9, 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

```
# 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

```
# 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

```
# 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

```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


# 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

```
# 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

```
# 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

```
# 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

```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


```

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
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

```
# 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

```
# 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
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)
```