SIMULATIONS

ESTIMATE PI MONTE CARLO

OUR TEAM

Ali Mohamed Sayed Ahmed Ali	20221449583
Yassen Ahmed SaadAllah shahin	20221310260
Ziad Asharf Ibrahim Taher	20221369225
Rana Alaa Ahmed Ahmed	20221456493
Moaz Mamdouh Fawzy	20221454751

INTRODUCTION

- One popular method for estimating pi is the Monte Carlo method, a computational technique that uses random numbers to simulate complex systems.

- The Monte Carlo method for estimating pi is based on the idea that if you randomly scatter points inside a square and count the number of points that fall inside a circle inscribed in the square, you can estimate the value of pi

STEPS OF ALGORITHM

- Generate a random point (x, y) inside a square of side 2 centred at the origin.
- Determine whether the point falls inside the unit circle inscribed in the square by checking whether $x^2 + y^2 <= 1$.
- Repeat steps 1 and 2 for a large number of points
- Calculate the ratio of the number of points that fell inside the circle to the total number of points generated.
- Multiply the ratio by 4 to estimate the value of pi.

SCREEN OF CODE

```r
# Load the readxl package
library(readxl)
# Read the Excel file containing the x and y sequences
# Replace "input.xlsx" with the path to your Excel file
data <- read_excel("input.xlsx")
# Extract x and y sequences from the data frame
x_sequence <- data[["x_sequence"]]
y_sequence <- data[["y_sequence"]]

# Check if the lengths of x and y sequences match
if (length(x_sequence) != length(y_sequence)) {
  stop("Length of x sequence must match length of y sequence.")
}

```

- this code reads data from an Excel file, extracts two sequences from it, and ensures that the lengths of these sequences match.
- If they don't match, it stops execution and displays an error message.

SCREEN OF CODE

- computes the Euclidean distance from the origin to each point.
- Points with distances greater than the corresponding values in y_sequence are assigned the color "red", otherwise "black".
- Points with distances greater than the corresponding values in **y_sequence** are labeled as "IN", otherwise "OUT".

```
# Calculate distances from the origin for all points
distances <- sqrt(1 - x_sequence^2)
# Initialize an empty vector to store colors
colors <- ifelse(distances > y_sequence, "red", "black")
# Calculate the decision for each point
decisions <- ifelse(distances > y_sequence, "IN", "OUT")
# Create a data frame with the columns
data <- data.frame(
  "First Random Number (X)" = x_sequence,
  "Second Random Number (Y)" = y_sequence,
  "Distances" = distances,
  "Decision" = decisions
)
```

SCREEN OF CODE

```
# Save the data frame to a CSV file
write.csv(data, "output.csv", row.names = TRUE)
# Display the data frame
print(data)
# Save the plot as a PNG image
png("output.png", width = 1600, height = 1400, units = "px", res = 300)
plot(x_sequence, y_sequence, pch = 20, col = colors)
dev.off()
```

- Saving the data data frame to a CSV file named "output.csv".
- It saves a plot of the x_sequence against y_sequence with points colored based on the colors vector to a PNG file named "output.png"

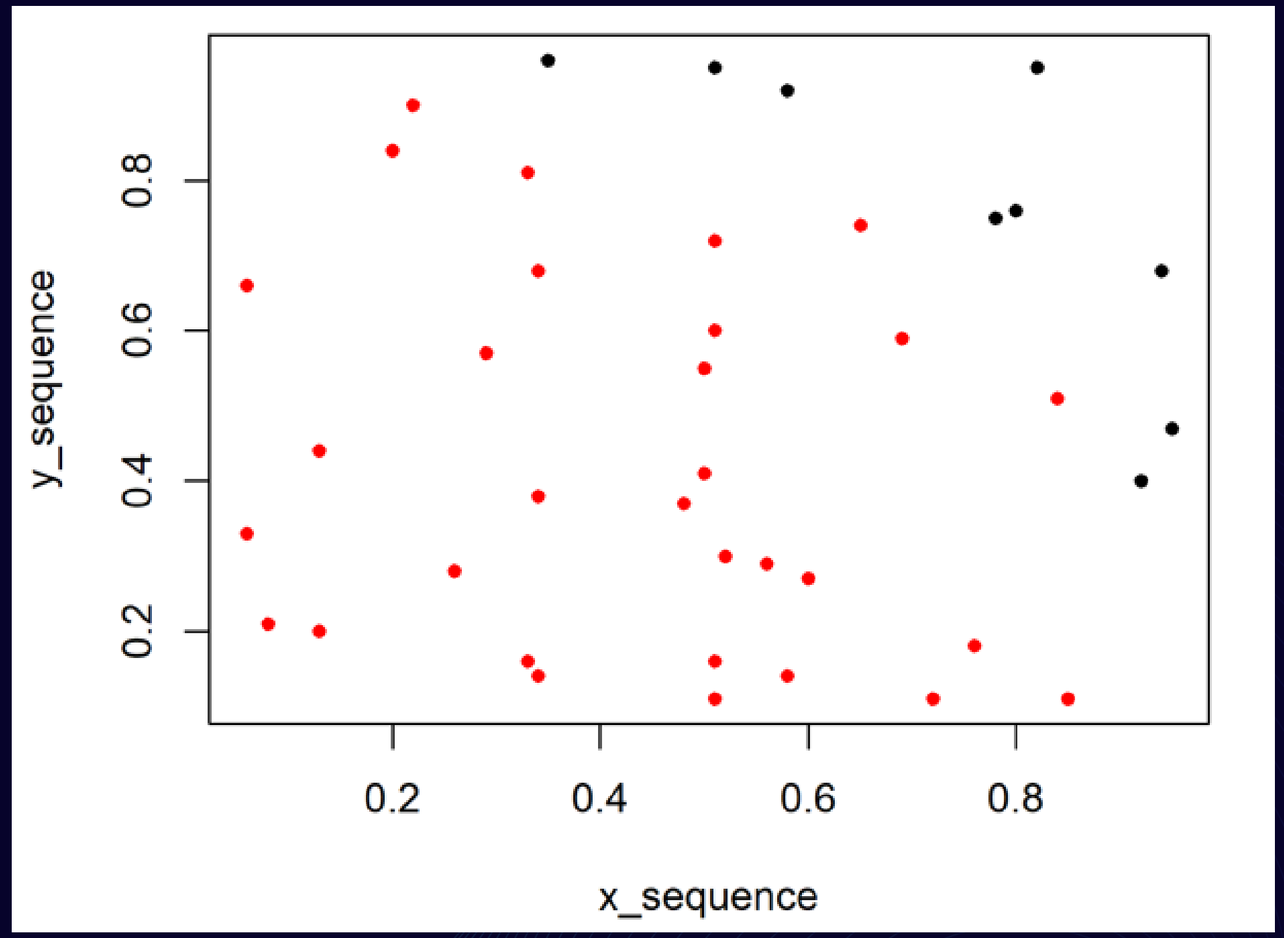
SCREEN OF CODE

- Count the number of "IN" decisions and stored it in M and calculate the numbers of iterations then stores it in N
- It estimates the value of Pi using the formula: estimated_pi=$(M/N)\times4$

```
# Count the number of "OUT" decisions
M <- sum(decisions == "IN")
# Calculate the count of iterations
N <- length(x_sequence)
# Calculate the estimated value of Pi
estimated_pi <- (M/N)*4
# Print the estimated value of Pi
cat("The value of Pi is estimated to be:", estimated_pi, "\n")
```

OUTPUT GRAPH:

- From this graph the points have colour red:
  - when √(1 - x^2) larger than Y
  - the points are inside the circle.
- From this graph the points have colour black:
  - when √(1 - x^2) Small than Y
  - the points are outside the circle.

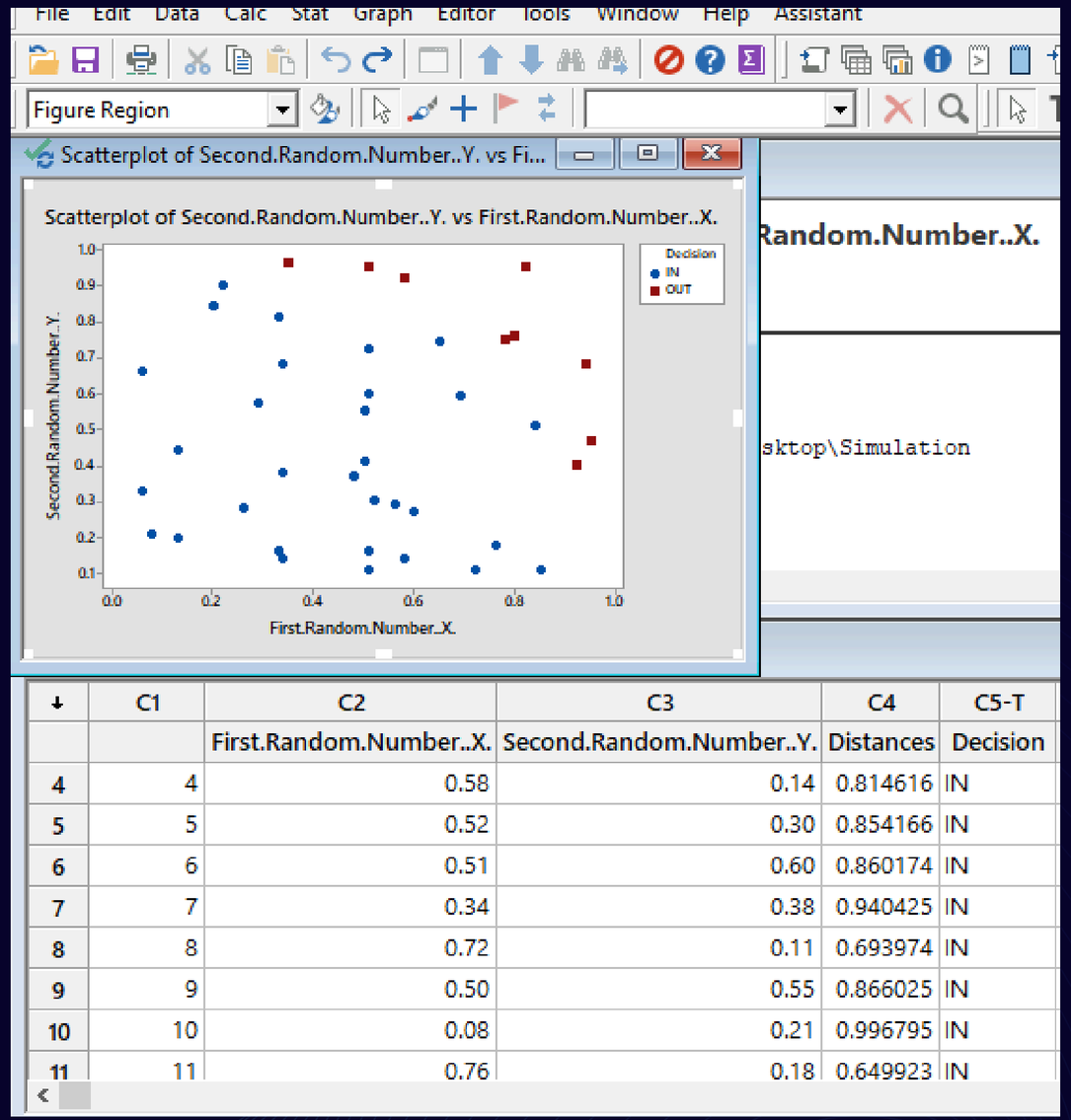


BY MINITAB:

- after we run simulations by R then we write output into file (excel).
  then take this file as input in MINITAB to visualize it by scatterplots

- From this graph the points have colour red:
  - when √(1 - x^2) large than Y
  - the points are inside the circle.

- From this graph the points have colour black:
  - when √(1 - x^2) Small than Y
  - the points are outside the circle.



	C1	C2	C3	C4	C5-T
		First.Random.Number..X.	Second.Random.Number..Y.	Distances	Decision
4	4	0.58	0.14	0.814616	IN
5	5	0.52	0.30	0.854166	IN
6	6	0.51	0.60	0.860174	IN
7	7	0.34	0.38	0.940425	IN
8	8	0.72	0.11	0.693974	IN
9	9	0.50	0.55	0.866025	IN
10	10	0.08	0.21	0.996795	IN
11	11	0.76	0.18	0.649923	IN