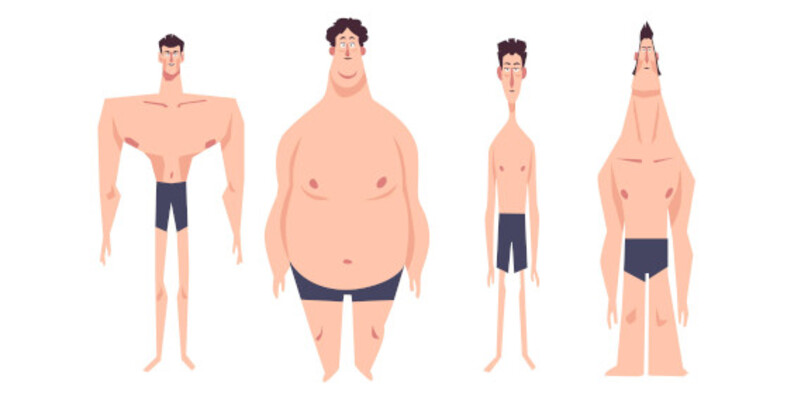
**"Bodyfat Explorer"**



|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 21F-9277 | Usman Saleem | BSCS |
| 2 | 21F-9158 | Mahnoor | BSCS |
| 3 | 21F-9197 | Sufyan Malik | BSCS |
| 4 | 22F-3165 | Muhammad Siaf | BSAI |
| 5 | 22F-3157 | Shaiq Irfan | BSAI |

**1. Problem Statement**

"Increasing awareness of health and fitness has led to a growing interest in understanding and analyzing body composition data. However, accessing and interpreting this data can be challenging for individuals. The project aims to address this challenge by developing an interactive and user-friendly Shiny app that allows users to explore, visualize, and gain insights from a comprehensive body fat dataset. The app includes features such as graphical representations, descriptive statistics, distribution analysis, linear regression modeling, and confidence interval calculations. By providing a centralized platform for health analytics, the project seeks to empower users to make informed decisions about their fitness goals and gain valuable insights into the factors influencing body fat composition."

**2. Objective**

* **Data Exploration and Visualization:**
  + Allow users to explore quantitative and qualitative aspects of body composition data through histograms, scatter plots, and bar charts.
  + Provide an intuitive interface for users to choose variables of interest for visualization.
* **Descriptive Statistics:**
  + Calculate and display summary statistics for selected variables to offer a comprehensive understanding of body composition metrics.
* **Distribution Analysis:**
  + Enable users to analyze the distribution of selected variables, including options for uniform and normal distributions.
  + Provide insights into the characteristics and patterns of the chosen variables.
* **Linear Regression Modeling:**
  + Implement a linear regression model to understand the relationship between dependent and independent variables.
  + Visualize the regression line along with scatter plots to illustrate trends and patterns.
* **Confidence Intervals:**
  + Calculate and display confidence intervals for both mean body fat percentage and regression coefficients.
  + Offer users a statistical perspective on the reliability and uncertainty associated with the estimates.
* **User Interaction and Customization:**
  + Design an interactive user interface allowing seamless input and exploration of various aspects of body composition data.
  + Facilitate user-driven analysis and decision-making by providing customization options for visualizations and model parameters.
* **Educational Value:**
  + Serve as an educational tool for users interested in understanding statistical concepts related to body composition analysis.
  + Provide informative outputs and explanations to enhance users' knowledge of health analytics.
* **Integration of Data Display:**
  + Display the underlying dataset, allowing users to view specific data points and gain a deeper understanding of the information being analyzed.
* **Overall User Empowerment:**
  + Empower users to make informed decisions about their health and fitness goals by providing a centralized platform for comprehensive body composition analysis.

**3. Data Description**

* **Dataset Origin:**
  + The dataset comprises body fat measurements for 436 individuals.
  + Initially, the dataset included measurements for 252 males. Subsequently, 184 samples for females were added, resulting in a combined dataset.
* **Measurement Method:**
  + Body density was measured using underwater weighing.
  + The Siri equation was applied to estimate body fat percentage based on the measured body density.
* **Data Source:**
  + The dataset was generously supplied by Dr. A. Garth Fisher.
  + Permission was granted for the free distribution and non-commercial use of the data.
* **Contributor Information:**
  + **Contributor:** Roger W. Johnson
  + **Affiliation:** Department of Mathematics & Computer Science, South Dakota School of Mines & Technology
  + **Address:** 501 East St. Joseph Street, Rapid City, SD 57701
  + **Email:** [rwjohnso@silver.sdsmt.edu](mailto:rwjohnso@silver.sdsmt.edu)
  + **Web Address:** <http://silver.sdsmt.edu/~rwjohnso>
* **Kaggle Dataset Link:**
  + [Body Fat Extended Dataset](https://www.kaggle.com/datasets/simonezappatini/body-fat-extended-dataset)
* **Usage Permission:**
  + Dr. A. Garth Fisher provided permission for the free distribution and non-commercial use of the dataset.
* **Acknowledgment:**
  + The dataset and associated information were made available by Dr. A. Garth Fisher, and the contribution is acknowledged.
* **Contact Information:**
  + For inquiries or further details, you may contact Roger W. Johnson via email at [rwjohnso@silver.sdsmt.edu](mailto:rwjohnso@silver.sdsmt.edu) or visit the web address <http://silver.sdsmt.edu/~rwjohnso>.

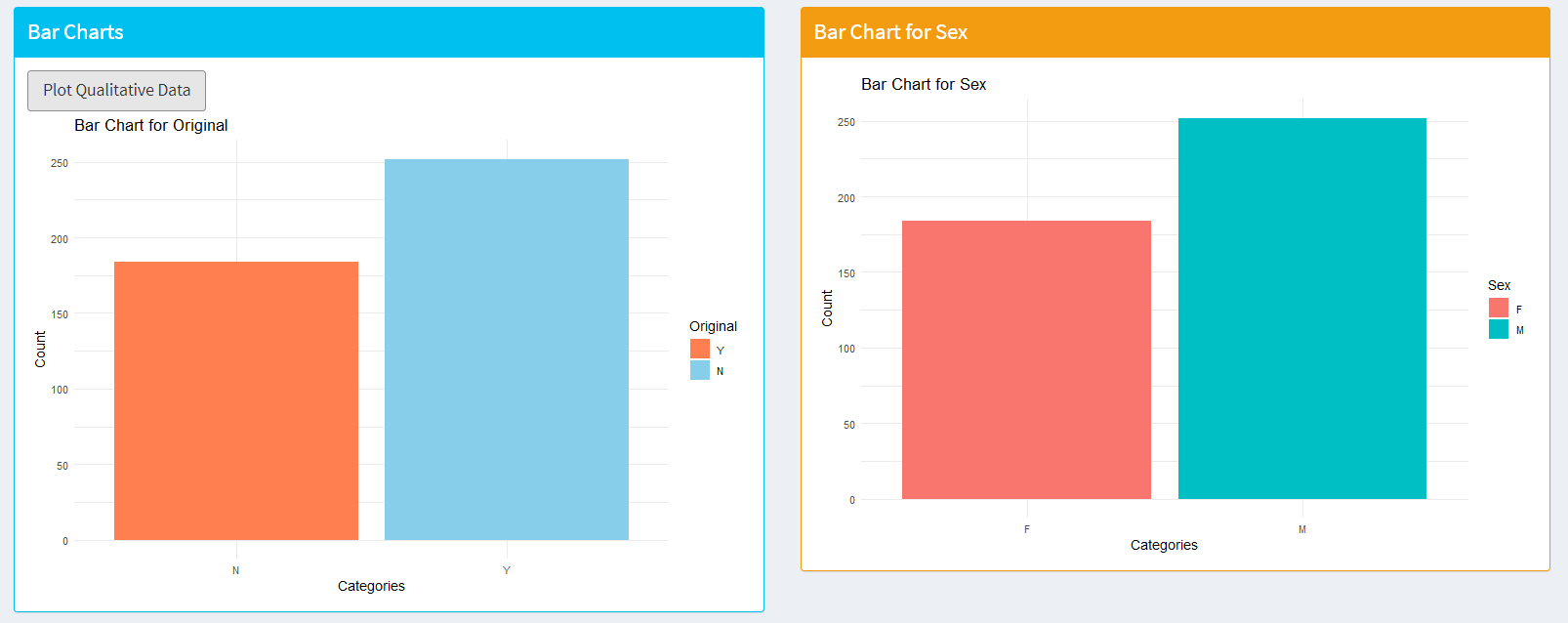
**4. Results**

**Graphs representing (Quantitative Data)**

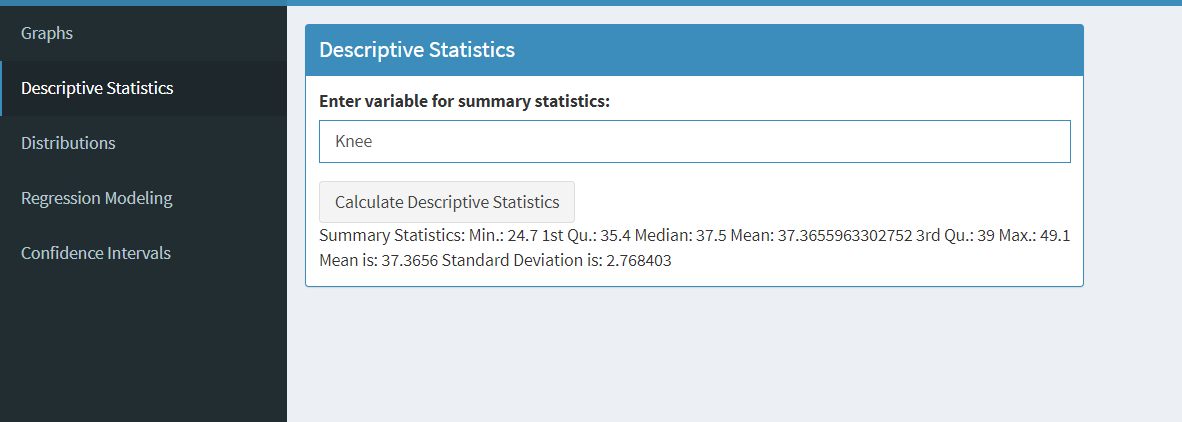
**A screenshot of a computer

Description automatically generated**

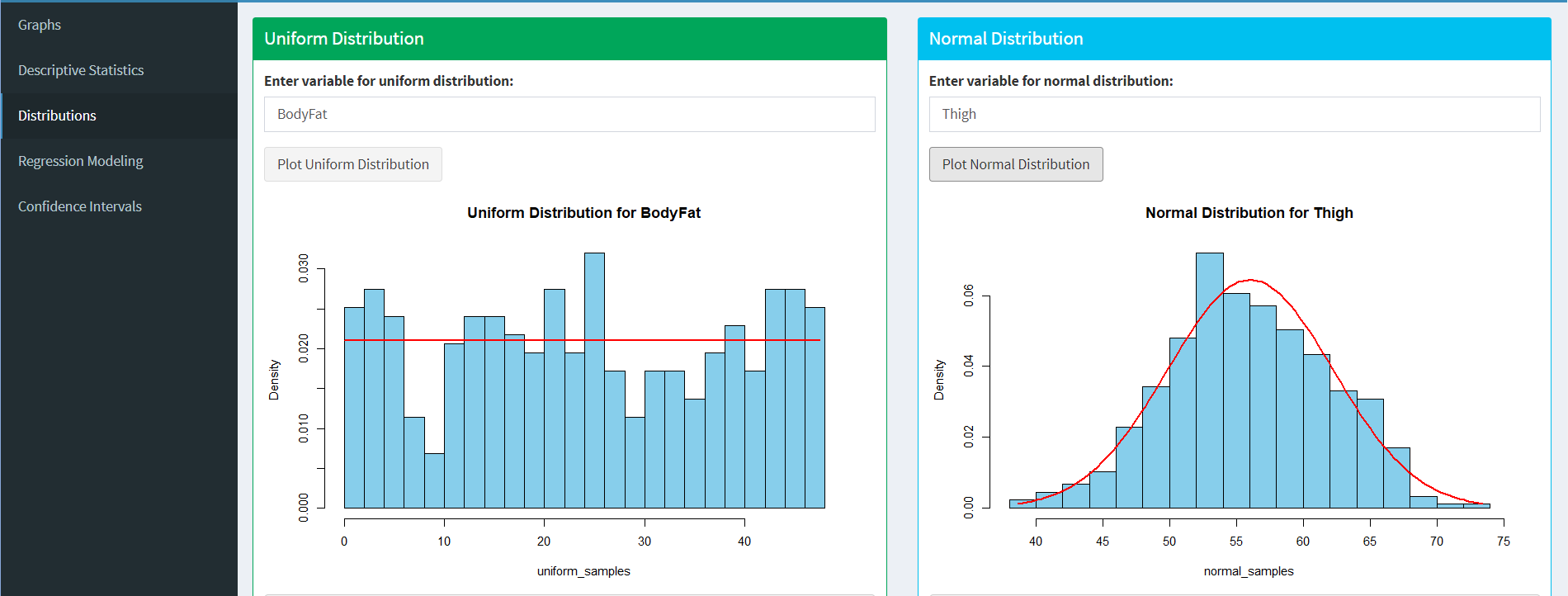
**Graphs representing (Qualitative Data)**

****

**Descriptive Statistics**

****

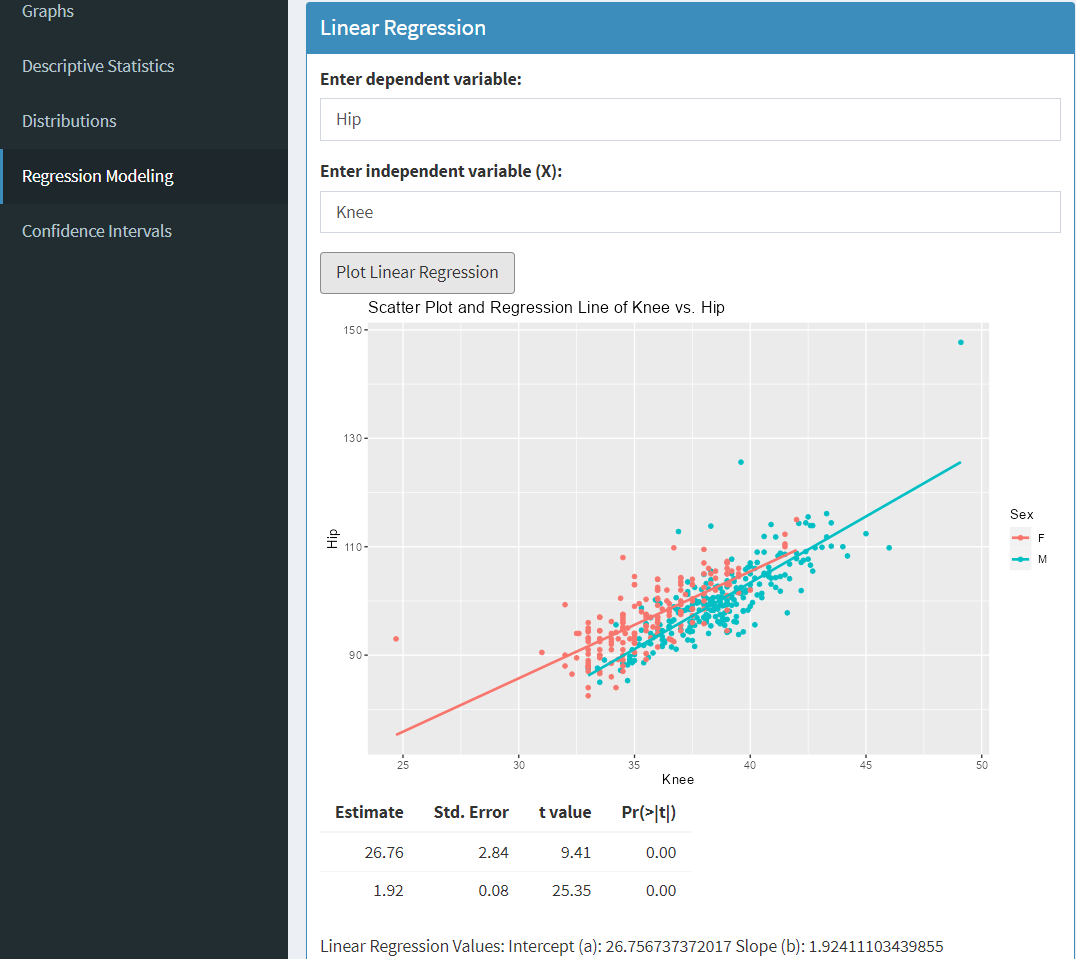
**Uniform and Normal Distribution**

****

**A white square with a black border

Description automatically generated with medium confidence**

**Regression Modeling:**

****

**Confidence Intervals**

**A screenshot of a computer

Description automatically generated**

**5. Codes**

**. R file**

read.csv("C:\\Users\\Wel\\Downloads\\BodyFat - Extended.csv")

# Load necessary libraries

# ggplot2: Used for creating and customizing complex plots and visualizations

library(ggplot2)

# dplyr: Provides a set of tools for efficiently manipulating datasets, including functions like filter(), select(), and summarise()

library(dplyr)

# MASS: Stands for Modern Applied Statistics with S. It provides various statistical functions and datasets.

library(MASS)

# Assuming your dataset is stored in a data frame called 'body\_data'

weight <- read.csv("C:\\Users\\Wel\\Downloads\\BodyFat - Extended.csv")

# Display the structure of the dataset

str(body\_data)

**# 1. Graphical and Tabular Data Representation**

# Quntitative Data

# Print the input

cat("Instructions 1.You cannot enter Orignal and Sex as a parameter beacuse they are Quantitative:", "\n")

# Prompt the user for input

user\_input <- readline("Enter the variable for X-axis: ")

# Histogram of Body Fat Percentage

# aes\_string(x = user\_input) specifies that the user-provided variable will be plotted on the x-axis

ggplot(weight, aes\_string(x = user\_input)) +

geom\_histogram(binwidth = 2, fill = "orange", color = "black") +

labs(title = user\_input ,

x = "X-axis",

y = "Frequency")

# Scatter plot of Weight vs. Body Fat Percentage for comparison

# Prompt the user for input

user\_input1 <- readline("Enter the first variable : ")

user\_input2 <- readline("Enter the second variable: ")

# Scatter plot using user inputs

ggplot(body\_data, aes\_string(x = user\_input1, y = user\_input2)) +

geom\_point(color = "darkgreen") +

labs(title = paste("Scatter plot of", user\_input2, "vs.", user\_input1),

x = user\_input1,

y = user\_input2)

# Qualitative Data

# Assuming 'Orignal' is a column in your data frame

# Create a bar chart

ggplot(weight, aes(x = Original, fill = Original)) +

geom\_bar() +

labs(title = "Bar Chart for Original", x = "Categories", y = "Count") +

scale\_fill\_manual(values = c("Y" = "skyblue", "N" = "coral"), labels = c("Y", "N")) +

theme\_minimal()

# Assuming 'Sex' is a column in your data frame

# Create a bar chart

ggplot(weight, aes(x = Sex, fill = Sex)) +

geom\_bar() +

labs(title = "Bar Chart for Sex", x = "Categories", y = "Count") +

theme\_minimal()

**# 2. Descriptive Statistical Measures**

# Prompt the user for input

user\_input3 <- readline("Enter the variable on which you want to apply summary statistics: ")

# Check if the entered variable exists in the dataset

if (!user\_input3 %in% names(body\_data)) {

cat("Error: The entered variable does not exist in the dataset.\n")

} else {

# Summary statistics

summary\_stats <- summary(body\_data[[user\_input3]])

print(summary\_stats)

# Mean and standard deviation

var1 <- mean(body\_data[[user\_input3]])

var2 <- sd(body\_data[[user\_input3]])

# Print the results

cat("Mean is:", var1, "\n")

cat("Standard Deviation is:", var2, "\n")

}

**# 3.Distributions**

# \*\*\*\*\* Uniform Distribution \*\*\*

#Chances of events are Equal

# Prompt the user for input

user\_variable <- readline("Enter the variable on which you want to apply the uniform distribution: ")

# Specify the range for the uniform distribution

min\_value <- min(weight[[user\_variable]])

max\_value <- max(weight[[user\_variable]])

# Calculate mean, variance, and standard deviation using the formula

mean\_value\_formula <- (min\_value + max\_value) / 2

variance\_value\_formula <- (max\_value - min\_value)^2 / 12

std\_deviation\_value\_formula <- sqrt(variance\_value\_formula)

# Probability Density Function (PDF) formula

pdf\_formula <- 1 / (max\_value - min\_value)

# Print the calculated values using the formula

cat("Mean (Formula):", mean\_value\_formula, "\n")

cat("Variance (Formula):", variance\_value\_formula, "\n")

cat("Standard Deviation (Formula):", std\_deviation\_value\_formula, "\n")

cat("PDF (Formula):", pdf\_formula, "\n")

# Generate random samples from a uniform distribution

n <- 437

uniform\_samples <- runif(n, min = min\_value, max = max\_value)

# Plot a histogram of the uniform distribution

hist(uniform\_samples, main = paste("Uniform Distribution for", user\_variable), col = "skyblue", breaks = 20)

# \*\*\*\*\*\*\*\*\* Normal Distribution \*\*\*\*\*\*\*\*\*

# Prompt the user for input

user\_variable <- readline("Enter the variable on which you want to apply the normal distribution: ")

# Extract the specified variable

selected\_variable <- weight[[user\_variable]]

# Calculate mean and standard deviation

mean\_value <- mean(selected\_variable)

std\_deviation\_value <- sd(selected\_variable)

# Calculate exact variance

variance\_value <- var(selected\_variable)

# Probability Density Function (PDF) formula for normal distribution

pdf\_normal <- function(x) {

exp(-((x - mean\_value)^2) / (2 \* std\_deviation\_value^2)) / (std\_deviation\_value \* sqrt(2 \* pi))

}

# Generate random samples from a normal distribution

n <- 437

normal\_samples <- rnorm(n, mean = mean\_value, sd = std\_deviation\_value)

# Plot a histogram of the normal distribution

hist(normal\_samples, main = paste("Normal Distribution for", user\_variable), col = "skyblue", breaks = 20, prob = TRUE)

# Plot the PDF curve

x\_values <- seq(min(normal\_samples), max(normal\_samples), length.out = 100)

lines(x\_values, pdf\_normal(x\_values), col = "red", lwd = 2)

# Print mean, variance, standard deviation, and PDF

cat("Mean:", mean\_value, "\n")

cat("Variance:", variance\_value, "\n")

cat("Standard Deviation:", std\_deviation\_value, "\n")

cat("PDF (Normal):", pdf\_normal(mean\_value), "\n")

**# 4. Regression Modeling and Predictions**

user\_variable1 <- readline("Enter the Dependent Var: ")

# Prompt user to enter the independent variable

independent\_var <- readline("Enter Independent Variable: ")

# Check if the entered variable names exist in the dataset

if (!(dependent\_var %in% colnames(weight))) {

stop("Entered dependent variable does not exist in the dataset.")

}

if (!(independent\_var %in% colnames(weight))) {

stop("Entered independent variable does not exist in the dataset.")

}

# Modify the formula with user input

formula <- formula(paste(dependent\_var, "~ Sex +", independent\_var))

# Perform linear regression

reg\_model <- lm(formula, data = weight)

# Display regression summary

summary(reg\_model)

# Scatter plot

library(ggplot2)

ggplot(weight, aes\_string(x = independent\_var, y = dependent\_var, color = "Sex")) +

geom\_point() +

geom\_smooth(method = "lm", se = FALSE) +

labs(title = "Scatter Plot and Regression Line",

x = independent\_var,

y = dependent\_var,

color = "Sex")

# Display regression coefficients

coefficients(reg\_model)

# Create a new data frame for prediction with user input

new\_data <- data.frame(Sex = c("Male", "Female"), Height = c(weight, weight))

# Display the predicted values

predicted\_values <- predict(reg\_model, newdata = new\_data)

cat("Predicted values of", dependent\_var, ":\n")

print(data.frame(Sex = new\_data$Sex, Predicted = predicted\_values))

**# 5. Confidence Interval of Descriptive Measures and Regression Estimates**

# Prompt user to enter the variable for which to calculate the confidence interval

variable\_name <- readline("Enter the variable name for which to calculate the confidence interval: ")

# Confidence interval for the mean of the specified variable

conf\_interval\_mean <- t.test(body\_data[[variable\_name]])$conf.int

cat(paste("95% Confidence Interval for Mean", variable\_name, ":", conf\_interval\_mean, "\n"))

# Confidence interval for the regression coefficients

# Assuming you have a linear regression model stored in 'model'

# You need to replace 'model' with the actual linear regression model object

# For example: model <- lm(Outcome ~ Predictor, data = your\_data)

conf\_interval\_regression <- confint(model)

cat("95% Confidence Intervals for Regression Coefficients:\n")

print(conf\_interval\_regression)

**. R (for shiny)**

# app.R

library(shiny)

library(ggplot2)

library(dplyr)

library(shinydashboard)

install.packages("shinydashboard")

# Read the data

weight <- read.csv("C:\\Users\\Wel\\Downloads\\BodyFat - Extended.csv")

ui <- dashboardPage(

dashboardHeader(title = "BodyFat Analysis Dashboard"),

dashboardSidebar(

sidebarMenu(

menuItem("Graphs", tabName = "graphs"),

menuItem("Descriptive Statistics", tabName = "desc\_stats"),

menuItem("Distributions", tabName = "distributions"),

menuItem("Regression Modeling", tabName = "regression"),

menuItem("Confidence Intervals", tabName = "conf\_intervals")

)

),

dashboardBody(

tabItems(

# Graphs tab

tabItem(

tabName = "graphs",

fluidRow(

box(

title = "Quantitative Data",

status = "primary",

solidHeader = TRUE,

width = 6,

textInput("quant\_variable", "Enter variable for X-axis:", ""),

actionButton("quant\_plotBtn", "Plot Quantitative Data"),

plotOutput("quant\_histogram")

),

box(

title = "Scatter Plot",

status = "success",

solidHeader = TRUE,

width = 6,

textInput("scatter\_x", "Enter X variable:", ""),

textInput("scatter\_y", "Enter Y variable:", ""),

actionButton("scatter\_plotBtn", "Plot Scatter Plot"),

plotOutput("scatter\_plot")

)

),

fluidRow(

box(

title = "Bar Charts",

status = "info",

solidHeader = TRUE,

width = 6,

actionButton("qual\_plotBtn", "Plot Qualitative Data"),

plotOutput("bar\_chart\_original")

),

box(

title = "Bar Chart for Sex",

status = "warning",

solidHeader = TRUE,

width = 6,

plotOutput("bar\_chart\_sex")

)

)

),

# Descriptive Statistics tab

tabItem(

tabName = "desc\_stats",

fluidRow(

box(

title = "Descriptive Statistics",

status = "primary",

solidHeader = TRUE,

width = 6,

textInput("desc\_variable", "Enter variable for summary statistics:", ""),

actionButton("desc\_statsBtn", "Calculate Descriptive Statistics"),

tableOutput("desc\_stats")

)

)

),

# Distributions tab

tabItem(

tabName = "distributions",

fluidRow(

box(

title = "Uniform Distribution",

status = "success",

solidHeader = TRUE,

width = 6,

textInput("uniform\_variable", "Enter variable for uniform distribution:", ""),

actionButton("uniform\_distributionBtn", "Plot Uniform Distribution"),

plotOutput("uniform\_distribution"),

verbatimTextOutput("uniform\_distribution\_values")

),

box(

title = "Normal Distribution",

status = "info",

solidHeader = TRUE,

width = 6,

textInput("normal\_variable", "Enter variable for normal distribution:", ""),

actionButton("normal\_distributionBtn", "Plot Normal Distribution"),

plotOutput("normal\_distribution"),

verbatimTextOutput("normal\_distribution\_values")

)

)

),

# Regression Modeling tab

tabItem(

tabName = "regression",

fluidRow(

box(

title = "Linear Regression",

status = "primary",

solidHeader = TRUE,

width = 6,

textInput("dependent\_variable", "Enter dependent variable:", ""),

textInput("independent\_variable", "Enter independent variable (X):", ""),

actionButton("regressionBtn", "Plot Linear Regression"),

plotOutput("regression\_plot"),

tableOutput("regression\_summary"),

tableOutput("regression\_output")

)

)

),

# Confidence Intervals tab

tabItem(

tabName = "conf\_intervals",

fluidRow(

box(

title = "Confidence Intervals",

status = "success",

solidHeader = TRUE,

width = 6,

textInput("variable\_name", "Enter variable name:", ""),

actionButton("conf\_intervalBtn", "Calculate Confidence Intervals"),

verbatimTextOutput("conf\_interval\_output")

)

)

)

)

)

)

# Define the server

server <- function(input, output) {

**# 1. Graphs**

output$quant\_histogram <- renderPlot({

ggplot(weight, aes\_string(x = input$quant\_variable)) +

geom\_histogram(binwidth = 2, fill = "orange", color = "black") +

labs(title = input$quant\_variable, x = "X-axis", y = "Frequency")

})

output$scatter\_plot <- renderPlot({

ggplot(weight, aes\_string(x = input$scatter\_x, y = input$scatter\_y)) +

geom\_point(color = "darkgreen") +

labs(title = paste("Scatter plot of", input$scatter\_y, "vs.", input$scatter\_x),

x = input$scatter\_x,

y = input$scatter\_y)

})

observeEvent(input$qual\_plotBtn, {

output$bar\_chart\_original <- renderPlot({

ggplot(weight, aes(x = Original, fill = Original)) +

geom\_bar() +

labs(title = "Bar Chart for Original", x = "Categories", y = "Count") +

scale\_fill\_manual(values = c("Y" = "skyblue", "N" = "coral"), labels = c("Y", "N")) +

theme\_minimal()

})

output$bar\_chart\_sex <- renderPlot({

ggplot(weight, aes(x = Sex, fill = Sex)) +

geom\_bar() +

labs(title = "Bar Chart for Sex", x = "Categories", y = "Count") +

theme\_minimal()

})

})

**# 2. Descriptive Statistics**

observeEvent(input$desc\_statsBtn, {

user\_input3 <- input$desc\_variable

if (!user\_input3 %in% names(weight)) {

output$desc\_stats <- renderPrint({

"Error: The entered variable does not exist in the dataset."

})

} else {

summary\_stats <- summary(weight[[user\_input3]])

mean\_value <- mean(weight[[user\_input3]])

sd\_value <- sd(weight[[user\_input3]])

output$desc\_stats <- renderPrint({

cat("Summary Statistics:\n", format\_summary(summary\_stats), "\n")

cat("Mean is:\n", mean\_value, "\n")

cat("Standard Deviation is:\n", sd\_value, "\n")

})

}

})

# Custom function to format summary statistics

format\_summary <- function(stats) {

formatted\_stats <- paste(names(stats), stats, sep = ": ", collapse = "\n")

return(formatted\_stats)

}

**# 3. Distributions**

observeEvent(input$uniform\_distributionBtn, {

user\_variable\_uniform <- isolate(input$uniform\_variable)

req(user\_variable\_uniform)

# Specify the range for the uniform distribution

min\_value\_uniform <- min(weight[[user\_variable\_uniform]])

max\_value\_uniform <- max(weight[[user\_variable\_uniform]])

# Calculate mean, variance, and standard deviation using the formula

mean\_value\_formula\_uniform <- (min\_value\_uniform + max\_value\_uniform) / 2

variance\_value\_formula\_uniform <- (max\_value\_uniform - min\_value\_uniform)^2 / 12

std\_deviation\_value\_formula\_uniform <- sqrt(variance\_value\_formula\_uniform)

# Probability Density Function (PDF) formula for uniform distribution

pdf\_uniform <- function(x) {

ifelse(x >= min\_value\_uniform & x <= max\_value\_uniform, 1 / (max\_value\_uniform - min\_value\_uniform), 0)

}

# Generate random samples from a uniform distribution

n\_uniform <- 437

uniform\_samples <- runif(n\_uniform, min = min\_value\_uniform, max = max\_value\_uniform)

# Plot a histogram of the uniform distribution with PDF

output$uniform\_distribution <- renderPlot({

hist(uniform\_samples, main = paste("Uniform Distribution for", user\_variable\_uniform), col = "skyblue", breaks = 20, probability = TRUE)

# Plot the PDF curve

x\_values\_uniform <- seq(min\_value\_uniform, max\_value\_uniform, length.out = 100)

lines(x\_values\_uniform, pdf\_uniform(x\_values\_uniform), col = "red", lwd = 2)

})

# Display the calculated values using verbatimTextOutput

output$uniform\_distribution\_values <- renderPrint({

list(

Mean = mean\_value\_formula\_uniform,

Variance = variance\_value\_formula\_uniform,

Standard\_Deviation = std\_deviation\_value\_formula\_uniform,

PDF\_at\_Mean = pdf\_uniform(mean\_value\_formula\_uniform) # Add the mean as x value to calculate PDF

)

})

})

# 3. Distributions (Normal)

observeEvent(input$normal\_distributionBtn, {

user\_variable\_normal <- isolate(input$normal\_variable)

req(user\_variable\_normal)

# Extract the specified variable

selected\_variable\_normal <- weight[[user\_variable\_normal]]

# Calculate mean and standard deviation

mean\_value\_normal <- mean(selected\_variable\_normal)

std\_deviation\_value\_normal <- sd(selected\_variable\_normal)

# Calculate exact variance

variance\_value\_normal <- var(selected\_variable\_normal)

# Probability Density Function (PDF) formula for normal distribution

pdf\_normal <- function(x) {

dnorm(x, mean = mean\_value\_normal, sd = std\_deviation\_value\_normal)

}

# Generate random samples from a normal distribution

n\_normal <- 437

normal\_samples <- rnorm(n\_normal, mean = mean\_value\_normal, sd = std\_deviation\_value\_normal)

# Plot a histogram of the normal distribution with PDF

output$normal\_distribution <- renderPlot({

hist(normal\_samples, main = paste("Normal Distribution for", user\_variable\_normal), col = "skyblue", breaks = 20, probability = TRUE)

# Plot the PDF curve

x\_values\_normal <- seq(min(normal\_samples), max(normal\_samples), length.out = 100)

lines(x\_values\_normal, pdf\_normal(x\_values\_normal), col = "red", lwd = 2)

})

# Display the calculated values using verbatimTextOutput

output$normal\_distribution\_values <- renderPrint({

list(

Mean = mean\_value\_normal,

Variance = variance\_value\_normal,

Standard\_Deviation = std\_deviation\_value\_normal,

PDF\_at\_Mean = pdf\_normal(mean\_value\_normal) # Add the mean as x value to calculate PDF

)

})

})

**# 4.Regression Modeling and Predictions**

observeEvent(input$regressionBtn, {

dependent\_var <- input$dependent\_variable

independent\_var <- input$independent\_variable

# Check if the entered variables exist in the dataset

if (!(dependent\_var %in% colnames(weight) && independent\_var %in% colnames(weight))) {

return() # Return if variables are not found

}

# Calculate means

mean\_X <- mean(weight[[independent\_var]])

mean\_Y <- mean(weight[[dependent\_var]])

# Calculate slope (b) and intercept (a) using least squares method

b <- sum((weight[[independent\_var]] - mean\_X) \* (weight[[dependent\_var]] - mean\_Y)) / sum((weight[[independent\_var]] - mean\_X)^2)

a <- mean\_Y - b \* mean\_X

# Predict Y values based on simple linear regression equation

predicted\_Y <- a + b \* weight[[independent\_var]]

output$regression\_plot <- renderPlot({

ggplot(weight, aes\_string(x = independent\_var, y = dependent\_var, color = "Sex")) +

geom\_point() +

geom\_smooth(method = "lm", se = FALSE) +

labs(

title = paste("Scatter Plot and Regression Line of", independent\_var, "vs.", dependent\_var),

x = independent\_var,

y = dependent\_var

)

})

output$regression\_output <- renderText({

paste("Linear Regression Values:\n",

"Intercept (a): ", a, "\n",

"Slope (b): ", b, "\n")

})

output$regression\_summary <- renderTable({

# Display other regression statistics if needed

summary(lm(weight[[dependent\_var]] ~ weight[[independent\_var]]))$coefficients

})

output$predicted\_values <- renderTable({

data.frame(Actual\_Y = weight[[dependent\_var]], Predicted\_Y = predicted\_Y)

})

})

**# 5.Confidence Intervals**

output$conf\_interval\_output <- renderPrint({

# Check if the button is clicked

req(input$conf\_intervalBtn)

# Prompt user to enter the variable name for which to calculate the confidence interval

variable\_name <- isolate(input$variable\_name)

# Check if the entered variable name exists in the dataset

if (!(variable\_name %in% colnames(weight))) {

return("Entered variable name does not exist in the dataset.")

}

# Confidence interval for the mean of the specified variable

conf\_interval\_mean <- t.test(weight[[variable\_name]])$conf.int

cat(paste("95% Confidence Interval for Mean", variable\_name, ":", conf\_interval\_mean, "\n\n"))

# Confidence interval for the regression coefficients

formula <- as.formula(paste(input$dependent\_variable, "~ Sex +", input$independent\_variable))

model <- lm(formula, data = weight)

conf\_interval\_regression <- confint(model)

cat("95% Confidence Intervals for Regression Coefficients:\n")

print(conf\_interval\_regression)

})

}

# Run the Shiny app

shinyApp(ui, server)

**6. Conclusion**

In conclusion, this project leverages the Body Fat Extended Dataset, consisting of measurements for 436 individuals, to offer a comprehensive analysis of body composition. Utilizing methods such as underwater weighing and the Siri equation, the dataset provides reliable estimates of body fat percentage. The project's user-centric approach empowers users to explore and analyze the dataset dynamically. Through a Shiny web application, users can input variables of interest, ranging from quantitative and qualitative data to conducting linear regression analyses and calculating confidence intervals. The interactive nature of the project facilitates a tailored exploration of the dataset, making it a versatile tool for researchers and enthusiasts interested in the intricacies of body fat measurement.

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*END\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***