```
# Assignment 4

    # 1. Measure of Central Tendency

        # 1 Program to calculate the Mean of data

        data <- c(1, 2, 3, 4, 5)
        mean_manual <- sum(data) / length(data)
        mean_builtin <- mean(data)
        cat("Mean (Manual Calculation):", mean_manual, "\n")
        cat("Mean (Built-in Function):", mean_builtin, "\n")

        # Output
        "Mean (Manual Calculation): 3
        Mean (Built-in Function): 3"

        # 2 Program to calculate the Median of data

        data <- c(1, 3, 3, 6, 7, 8, 9)
        n <- length(data)
        data_sorted <- sort(data)
        if (n %% 2 == 1) {
          median_manual <- data_sorted[(n + 1) / 2]
        } else {
          median_manual <- (data_sorted[n / 2] + data_sorted[n / 2 + 1]) / 2
        }

        median_builtin <- median(data)
        cat("Median (Manual Calculation):", median_manual, "\n")
        cat("Median (Built-in Function):", median_builtin, "\n")

        #Output
        "Median (Manual Calculation): 6
        Median (Built-in Function): 6 "

        # 3 Program to calculate the Mode of data

        data <- c(1, 2, 2, 3, 3, 3, 4, 4, 5)
        get_mode <- function(x) {
          uniq_vals <- unique(x)
          freq <- tabulate(match(x, uniq_vals))
          mode_val <- uniq_vals[which.max(freq)]
          return(mode_val)
        }

        mode_manual <- get_mode(data)
        cat("Mode (Manual Calculation):", mode_manual, "\n")

        # Output
        "Mode (Manual Calculation): 3"

    # 2. Measure of Dispersion

        # 1 Program to calculate the Standard Deviation

        data <- c(1, 2, 3, 4, 5)
        std_manual <- sqrt(sum((data - mean(data))^2) / (length(data) - 1))
        std_builtin <- sd(data)
        cat("Standard Deviation (Manual Calculation):", std_manual, "\n")
        cat("Standard Deviation (Built-in Function):", std_builtin, "\n")

        # Output
        "Standard Deviation (Manual Calculation): 1.581139
        Standard Deviation (Built-in Function): 1.581139 "
```

```
    # 2 Program to calculate the Variance

    variance_manual <- sum((data - mean(data))^2) / (length(data) - 1)
    variance_builtin <- var(data)
    cat("Variance (Manual Calculation):", variance_manual, "\n")
    cat("Variance (Built-in Function):", variance_builtin, "\n")

    # Output
    "Variance (Manual Calculation): 2.5
    Variance (Built-in Function): 2.5 "

    # 3 Program to calculate the Coefficient of Variation (CV)

    cv_manual <- (std_manual / mean(data)) * 100
    cat("Coefficient of Variation (Manual Calculation):", cv_manual, "%\n")

    # Output
    "Coefficient of Variation (Manual Calculation): 52.70463 %"

# 3. Correlation Analysis

    # 1 Program to generate a Scatter Plot

    x <- c(1, 2, 3, 4, 5)
    y <- c(2, 4, 5, 4, 5)
    plot(x, y, main = "Scatter Plot", xlab = "X Values", ylab = "Y Values")

    # 2 Program to calculate Pearson's Correlation Coefficient

    pearson_manual <- cor(x, y, method = "pearson")
    cat("Pearson's Correlation Coefficient:", pearson_manual, "\n")

    # Output
    "Pearson's Correlation Coefficient: 0.7745967"

    # 3 Program to calculate Spearman's Rank Correlation Coefficient

    spearman_manual <- cor(x, y, method = "spearman")
    cat("Spearman's Rank Correlation Coefficient:", spearman_manual, "\n")

    # Output
    "Spearman's Rank Correlation Coefficient: 0.7378648"

# 4. Regression Analysis

    # 1 Program to calculate regression line y on x

    lm_model <- lm(y ~ x)
    cat("Regression Equation: y =", coef(lm_model)[1], "+", coef(lm_model)[2], "*
x\n")

    # Output
    "Regression Equation: y = 2.2 + 0.6 * x"

    # 2. Regression of x on y (x = a + by)

    lm_x_on_y <- lm(x ~ y)
    cat("Regression Equation (x on y): x =", coef(lm_x_on_y)[1], "+", coef(lm_x_on_y)
[2], "* y\n")

    # Output
    "Regression Equation (x on y): x = 0.7 + 0.4 * y"

# 5. Linear Curve Fitting
```

```
        # Given a set of data points, fit a linear curve of the form y = a + bx
        # using the least squares method. Provide the equation and plot the fitted line.

        x <- c(1, 2, 3, 4, 5)
        y <- c(2, 4, 5, 4, 5)

        linear_model <- lm(y ~ x)

        a_linear <- coef(linear_model)[1]
        b_linear <- coef(linear_model)[2]

        cat("Equation of fitted line: y =", round(a_linear, 2), "+", round(b_linear, 2),
"* x\n")
        plot(x, y, main = "Linear Curve Fitting", xlab = "X Values", ylab = "Y Values",
col = "blue", pch = 19)
        abline(linear_model, col = "red", lwd = 2)

        # Output
        "Equation of fitted line: y = 2.2 + 0.6 * x"


    # 6. Quadratic Curve Fitting

        # Given a set of data points, fit a quadratic curve of the form y = a + bx + cx^2
        # using the least squares method. Provide the equation and plot the fitted curve.

        x_quad <- c(1, 2, 3, 4, 5)
        y_quad <- c(3, 6, 11, 18, 27)

        quadratic_model <- lm(y_quad ~ poly(x_quad, 2, raw = TRUE))

        a_quad <- coef(quadratic_model)[1]
        b_quad <- coef(quadratic_model)[2]
        c_quad <- coef(quadratic_model)[3]

        cat("Equation of fitted quadratic curve: y =", round(a_quad, 2), "+",
round(b_quad, 2), "* x +", round(c_quad, 2), "* x^2\n")
        plot(x_quad, y_quad, main = "Quadratic Curve Fitting", xlab = "X Values", ylab =
"Y Values", col = "blue", pch = 19)
        curve(a_quad + b_quad * x + c_quad * x^2, add = TRUE, col = "green", lwd = 2)

        # Output
        "Equation of fitted quadratic curve: y = 2 + 0 * x + 1 * x^2"

    # 7. Exponential Curve Fitting

        # Given a set of data points, fit an exponential curve of the form y = a * e^(bx)
        # using the least squares method. Provide the equation and plot the fitted curve.

        x_exp <- c(1, 2, 3, 4, 5)
        y_exp <- c(2.7, 7.3, 20.1, 54.6, 148.4)

        log_y_exp <- log(y_exp)
        exp_model <- lm(log_y_exp ~ x_exp)
        log_a_exp <- coef(exp_model)[1]
        b_exp <- coef(exp_model)[2]
        a_exp <- exp(log_a_exp)

        cat("Equation of fitted exponential curve: y =", round(a_exp, 2), "* e^(",
round(b_exp, 2), " * x)\n")
        plot(x_exp, y_exp, main = "Exponential Curve Fitting", xlab = "X Values", ylab =
"Y Values", col = "blue", pch = 19)
        curve(a_exp * exp(b_exp * x), add = TRUE, col = "red", lwd = 2)
```

```
# Output
"Equation of fitted exponential curve: y = 0.99 * e^( 1  * x)"


# 8. Power Law Curve Fitting

    # Given a set of data points, fit a power law curve of the form y = a * x^b
    # using the least squares method. Provide the equation and plot the fitted curve.

    x_pow <- c(1, 2, 3, 4, 5)
    y_pow <- c(2, 4, 8, 16, 32)

    log_x_pow <- log(x_pow)
    log_y_pow <- log(y_pow)

    pow_model <- lm(log_y_pow ~ log_x_pow)
    log_a_pow <- coef(pow_model)[1]
    b_pow <- coef(pow_model)[2]
    a_pow <- exp(log_a_pow)

    cat("Equation of fitted power law curve: y =", round(a_pow, 2), "* x^",
round(b_pow, 2), "\n")
    plot(x_pow, y_pow, main = "Power Law Curve Fitting", xlab = "X Values", ylab = "Y
Values", col = "blue", pch = 19)
    curve(a_pow * x^b_pow, add = TRUE, col = "green", lwd = 2)

    # Output
    "Equation of fitted power law curve: y = 1.6 * x^ 1.68"


# 9. Logarithmic Curve Fitting

    # Given a set of data points, fit a logarithmic curve of the form y = a * x^b
    # using the least squares method. Provide the equation and plot the fitted curve.

    x_log <- c(1, 2, 3, 4, 5)
    y_log <- c(1, 4, 9, 16, 25)

    log_x_log <- log(x_log)
    log_y_log <- log(y_log)
    log_model <- lm(log_y_log ~ log_x_log)
    log_a_log <- coef(log_model)[1]

    b_log <- coef(log_model)[2]
    a_log <- exp(log_a_log)

    cat("Equation of fitted logarithmic curve: y =", round(a_log, 2), "* x^",
round(b_log, 2), "\n")
    plot(x_log, y_log, main = "Logarithmic Curve Fitting", xlab = "X Values", ylab =
"Y Values", col = "blue", pch = 19)
    curve(a_log * x^b_log, add = TRUE, col = "purple", lwd = 2)

    # Output
    "Equation of fitted logarithmic curve: y = 1 * x^ 2"
```