DATA501-Assignment1

2024-07-26

Program Description

The objective of the project is to develop a function to perform Shapiro-Wilk test and derive W statistic. The R code for the function and code descriptions were given below.

1.) Implementation of git, GitHub and commit history

GitHub Repository: here

- The project (package) is developed in a local repository and connected to a GitHub repository named *Normality-Test-R* click here to the repository The repository has 3 branches
  - main - the initial implementation (success scenario) of the code is committed and pushed to the main branch
  - unit-testing - this branch is created from the main branch and the unit test cases for the initial development is added to the test directory
  - feature-branch - it is created from unit-testing branch; then exception handling and input validation is added with unit test cases for the exception part.

Development codes were regularly committed to the repo with relevant commit messages; and the branches were properly merged (creating pull request to the prior branch) when the development is successful.

2.) Functionality of the R function

```r
normality <- function(data, plot=FALSE) {

  # Exceptions: detect NULL / Inf / NA values
  if (all(is.null(data))) stop("Input need to be a non NULL value")
  if (all(is.infinite(data))) stop("Input need to be a non Inf value")
  if (all(is.na(data))) stop("Input need to be a non NA value")


  #Exceptions: Detect invalid data formats / Invalid Dimensions / Data catch
  stopifnot(is.numeric(data))
  if (length(dim(data)) > 1) stop("Input must be an vector/1-D array")
  if (length(unique(data)) == 1) stop("Input must not be identical")

  n <- length(data)
  if (n <=2 ) stop("Input must have atleast three or more non NA values")

  # Warnings: NA values
  if (any(is.na(data))) {
    warning("DATA has NA values: W-test statistics value is derived ignoring NAs")
    data <- data[complete.cases(data)]
    n <- length(data)
  }

  # Exceptions: Detect invalid value for argument-plot
  if (!(plot %in% c(TRUE, FALSE))) stop("Paramete plot must have Boolean values")

  if (plot == TRUE) {
    qqnorm(data, pch = 1, frame = FALSE)
    qqline(data, col = "blue", lwd = 1)
  }

  # result <- shapiro.test(data)

  sorted_data <- sort(data)
  a_numerator = (1:n - 3/8)
  a_denominator = (n + 1/4)
  a_value <- qnorm(a_numerator / a_denominator)
  W <- sum(a_value * sorted_data)^2 / sum((sorted_data - mean(data))^2)
  return(W)

}
```

3.) Code description

The function takes two input arguments named data and plot which is an optional parameter. data takes the input to test the normality and plot takes Boolean value for generating a QQplot.

- For the success scenario of the function follows the below steps,
  - it removes if any NA value is detected in the output
  - sorts the dataset in ascending order
  - finds the a value
  - calculates and return the W statistic value using shapiro wilk formula
  - Optional QQ-plot
    - the function generates a qqplot along with the W value when the plot parameter is TRUE
    - returns only the W statistics if the parameter is FALSE

Input/Output

- Input = continuous vector (numerical)
- Output = W statistic value (Integer) & QQ-plot (optional)

Validation

- On the exception handling scenario, testing input / output / functionality
  - The function raises custom exception for the below scenarios
    1. NULL / Inf / NA values as input

    2. Invalid data format - any values except numerical vector/1-D array
    3. Invalid dimensions
    4. If all the values in the vector are identical
    5. If the vector has less than or equal to two values
    6. any values except Boolean for plot parameter

  – The function also raises warnings if any NA values present in the input data

Unit Testing for the above both positive and exception scenarios 15 - different test cases were created and created into a test script. Please find the file inside **test/testthat**.

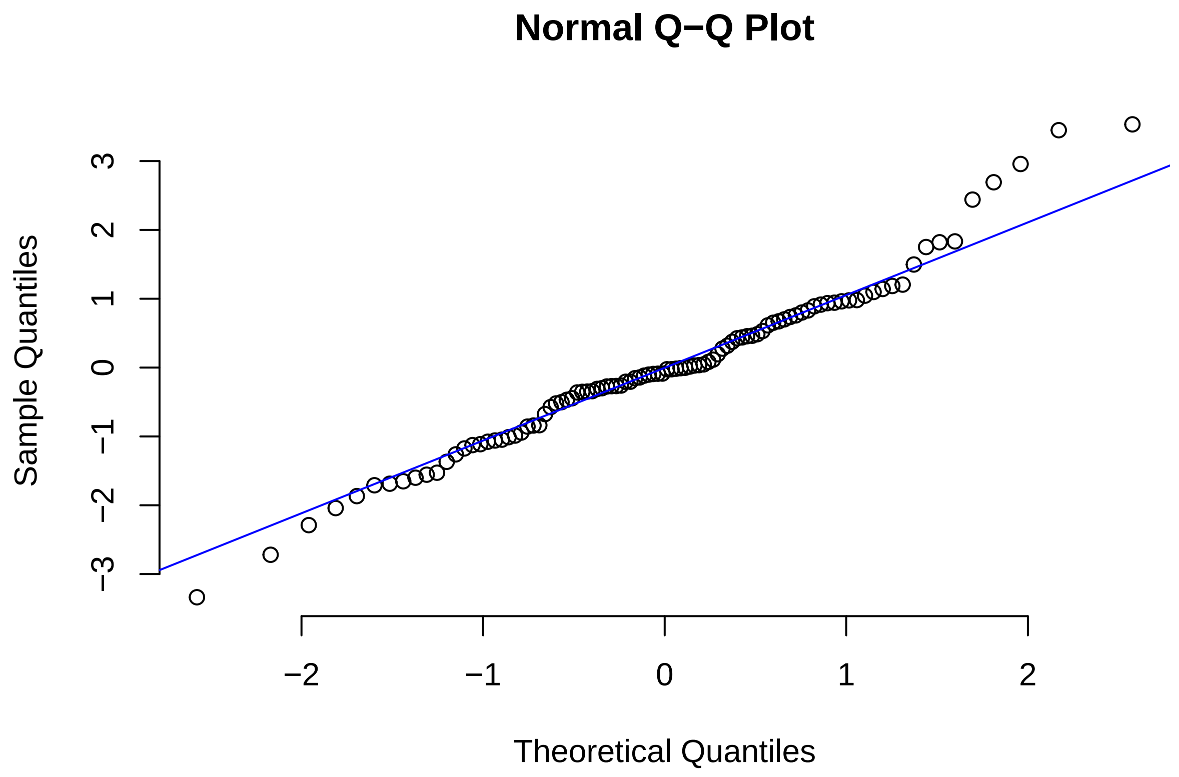
Error Handling Custom error message were written to capture the effect of invalid inputs or logic of the functionality.Additionally added Warning messages to the required cases.

4.) Examples

```
# positive scenario
normality(rnorm(15))
```

```
## [1] 12.20421
```

```
# positive scenario with Output plot
normality (rnorm(100), plot=TRUE)
```



```
## [1] 94.30996
```

```r
# Exception for the NULL Input
normality(NULL)
```

```
## Error in normality(NULL): Input need to be a non NULL value
```

```r
# Exception for the NA Input
normality(NA)
```

```
## Error in normality(NA): Input need to be a non NA value
```

```r
# Exception for invalid input format
normality(c("random","input","vector","in", "wrong","format"))
```

```
## Error in normality(c("random", "input", "vector", "in", "wrong", "format")): is.numeric(data) is not
```

```r
# Exception for Invalid dimension
normality(array(c(1,2,3,4),dim=c(2,2)))
```

```
## Error in normality(array(c(1, 2, 3, 4), dim = c(2, 2))): Input must be an vector/1-D array
```