**Mugisha Ivan Jalagatha**

**2024/HD05/22104U**

**MDSE**

**End Of Semester Project**

**Question 1**

1. linear-model

Purpose:

•Performs linear regression using matrix algebra.

•Returns model coefficients, including the intercept and slope(s), depending on the number of predictors.

Implementation Details:

•Constructs a design matrix X with a column of ones (for intercept) and predictor variables.

•Solves the linear equation , where is the coefficients vector.

Code:

(define (linear-model xs y)

(let ([X (list\*->matrix

(map (\x y (flatten (list x y)))

(build-list (length xs) (const 1)) xs))]

[Y (->col-matrix y)])

(matrix->list (matrix\*

(matrix-inverse (matrix\* (matrix-transpose X) X))

(matrix\* (matrix-transpose X) Y)))))

2. list->sentiment

Purpose:

•Converts a list of word-frequency pairs into sentiment scores using a specified sentiment lexicon.

Inputs:

•lst: A list of pairs where each pair contains a word and its frequency.

•#:lexicon: Specifies the lexicon to use for sentiment analysis (nrc, bing, or AFINN).

Code:

(define (list->sentiment lst #:lexicon [lexicon 'nrc])

(define (pack-sentiment lst lexicon)

(apply append (list '("word" "sentiment" "freq"))

(map (\x

(let ([result (token->sentiment (first x) #:lexicon lexicon)])

(map (\y (append y (list (second x)))) result)))

lst)))

(let ([sentiment (pack-sentiment lst lexicon)])

(if (> (length sentiment) 1)

sentiment

'())))

3. read-csv

Purpose:

•Reads a CSV file and optionally converts data to numeric format.

•Handles headers and commented lines.

Example Usage:

(define data (read-csv "data.csv" #:->number? #t))

(display data)

4. qq-plot\*

Purpose:

•Creates and displays a Q-Q plot, comparing sample quantiles to theoretical normal quantiles.

Inputs:

•lst: A list of numeric values to plot.

•#:scale?: Whether to z-transform the data (default: #t).

Implementation Details:

• Uses the qq-plot function to generate the plot and displays it using Racket's plotting capabilities.

Code:

(define (qq-plot\* lst #:scale? [scale? #t])

(plot (qq-plot lst #:scale? scale?)

#:x-label "Theoretical Normal Quantiles"

#:y-label "Sample Quantiles"))

Example Usage:

(define sample-data (sample (normal-dist 0 1) 100))

(qq-plot\* sample-data)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

5. hist

Purpose:

• Generates a discrete histogram for a list of values.

Inputs:

• lst: A list of discrete numeric values.

Output:

Example Usage:

(define values '(3 3 2 1 4 4 4))

(hist values)

**Summary of Abstraction Levels**

High-Level:

1. User-friendly interfaces for specific tasks (e.g., generating Q-Q plots, reading CSVs, performing regression).
2. Mid-Level: Encapsulation of core computational logic (e.g., matrix algebra, sentiment analysis, histogram calculation).
3. Low-Level: Detailed algorithmic implementation, including matrix operations and data parsing.
4. Implementation-Level: Use of functional constructs, modularity, and library integrations.

User Interaction: Customizable parameters and simple usage patterns

**Challenges**

1. Difficulty in getting raw installing dependencies like json, etc
2. Setting up was challenging

Pictorials

Assignment 1.

Assignment 2.

The Iink to the my github Repository :

Conclusion

Data abstraction plays a vital role in the design of the functions outlined in this report. Each function abstracts away complexities, providing users with intuitive interfaces for high-level operations like regression, visualization, and text processing. By layering abstractions—from high-level usability to low-level algorithmic implementation—the toolkit effectively bridges simplicity and functionality.

At the highest level, these functions enable users to accomplish specific goals without needing to manage intricate details. For example, the linear-model function simplifies linear regression by abstracting matrix algebra operations. Similarly, read-csv hides the complexities of file parsing, while list->sentiment encapsulates sentiment analysis tasks.