StatKeyEval
A Statistical Framework for Dynamic Keyword Extraction, Evaluation, and Assessment Automation Aim:

To implement an automatic short-answer grading system using feature engineering and ensemble- based approaches, focusing on extracting keywords, computing similarity metrics, and generating confidence scores.

Algorithm:

1. **Text Preprocessing**

   - Lowercase text, remove punctuation, numbers, stop words, and extra spaces.

2. **Keyword Extraction (IGRKE)**

   - Compute Information Gain Ratio (IGR) for words.

   - Adjust importance using Frequency Adjustment Factor (FAF).

   - Rank and extract top keywords for reference and responses.

3. **Keyword Mutation (SCM)**

   - Use co-occurrence and PMI to find related terms.

   - Identify frequent words ($\geq$65%) with high PMI.

   - Apply Uniqueness Filtering and expand reference keywords.

4. **Vector Representation**

   - Create a universal keyword set.

   - Represent answers as binary vectors (1 = present, 0 = absent).

   - Normalize for length variations.

5. **Similarity Calculation**

   Compute multiple similarity metrics:

   - Cosine similarity (Simcos)

   - Normalized Euclidean distance (Simeuc)

   - Normalized Manhattan distance (Simman)

   - Adjusted Pearson correlation (Simpearson) Compute a weighted hybrid similarity score:

   - **Similarity(A,S) = 0.4 × Simcos + 0.3 × Simeuc + 0.2 × Simman + 0.1 ×**

   Simpearson Ensure similarity values align with grading standards.

6. **Score Generation**

   - Scale similarity score to grading scale.

- Round to generate the final score.

7. **Performance Evaluation**

- Calculate RMSE, MAE, MAPE.

- Compute Pearson R, Spearman ρ, and R^2.

- Analyze errors for grading consistency.

Research Paper:

Title: *Feature Engineering and Ensemble-Based Approach for Improving Automatic Short-Answer Grading Performance*

Authors: Archana Sahu and Plaban Kumar Bhowmick.

Conference/Journal: Educational Data Mining Conference (2018)

Datasets:

1. UNT Dataset
2. SciEntsBank Dataset
3. Beetle Dataset

Theoretical Derivation of Novel Statistical Functions for Keyword Extraction and Mutation

1. Information Gain Ratio Keyword Extraction (IGRKE)

The Information Gain Ratio Keyword Extraction (IGRKE) method identifies the most informative words in a corpus by evaluating their ability to distinguish between answer keys and student responses.

1.1 Entropy and Information Theory Foundation

For any word W in the corpus, its probability distribution is defined as:

- $P(W)$: Probability of word W occurring.

- $P(\neg W) = 1 - P(W)$: Probability of W not occurring.

- C: The document category (Answer Key vs. Student Response).

The total entropy of the word occurrence is given by:

$H(W) = - P(W) * \log_2 P(W) - P(\neg W) * \log_2 P(\neg W)$

This measures the uncertainty of the word's distribution across the dataset.

1.2 Conditional Entropy and Information Gain

The conditional entropy $H(C|W)$ represents the uncertainty in categorizing a document given that word W is known:

$H(C|W) = - P(A|W) * \log_2 P(A|W) - P(R|W) * \log_2 P(R|W)$ where:

- $P(A|W)$ is the probability of the document being an Answer Key given W.

- $P(R|W)$ is the probability of the document being a Student Response given W.

The Information Gain (IG) quantifies how much knowing W reduces uncertainty about the document's category:

$IG(W) = H(C) - H(C|W)$ where $H(C)$ is the entropy of the category distribution.

1.3 Normalization Using Split Information

To prevent bias toward frequent words, we normalize the Information Gain using Split Information (SI):

$SI(W) = - P(W) * \log_2 P(W) - P(\neg W) * \log_2 P(\neg W)$

This normalizes IG to Information Gain Ratio (IGR):

$IGR(W) = IG(W) \div SI(W)$

1.4 Frequency Adjustment Factor (FAF) for Balancing Word Significance

To ensure the extracted keywords are relevant across document categories, we introduce a Frequency Adjustment Factor (FAF):

$FAF = (F\_answer * F\_response) \div (Total\_F\_answer * Total\_F\_response)$ where:

- F_answer and F_response are the word frequencies in Answer Keys and Student Responses, respectively.

- Total_F_answer and Total_F_response are the total word counts in both document types.

1.5 Final IGRKE Scoring Function

The final scoring function combines IGR and FAF using a logarithmic transformation: Score(W) =

IGR(W) * (1 + log(1 + FAF * 1000))

---

2. Statistical Co-occurrence Mutation (SCM)

2.1 Co-occurrence Matrix Construction

A co-occurrence matrix M is built where each entry represents the number of documents where words i and j appear together:

M[i, j] = count of documents where both words i and j appear

2.2 Pointwise Mutual Information (PMI) for Semantic Association

$PMI(i, j) = \log_2 [P(i, j) \div (P(i) * P(j))]$

2.3 Hybrid Similarity Measure for Keyword Comparison

Sim(K1, K2) = 0.6 × Jaccard_Sim(K1, K2) + 0.4 × PMI_Sim(K1, K2)

3. Similarity Scoring Function for Answer Matching

Similarity(A, S) = 0.4 × Jaccard_Sim(A, S) + 0.4 × Coverage(A, S) + 0.2 × Position_Score(A, S)

where:

- Jaccard Similarity measures word overlap.
- Coverage measures the proportion of answer key words found.
- Position Score captures structural similarity:

Position_Score(A, S) = Average (1 - | Pos_A(w) ÷ |A| - Pos_S(w) ÷ |S| |)
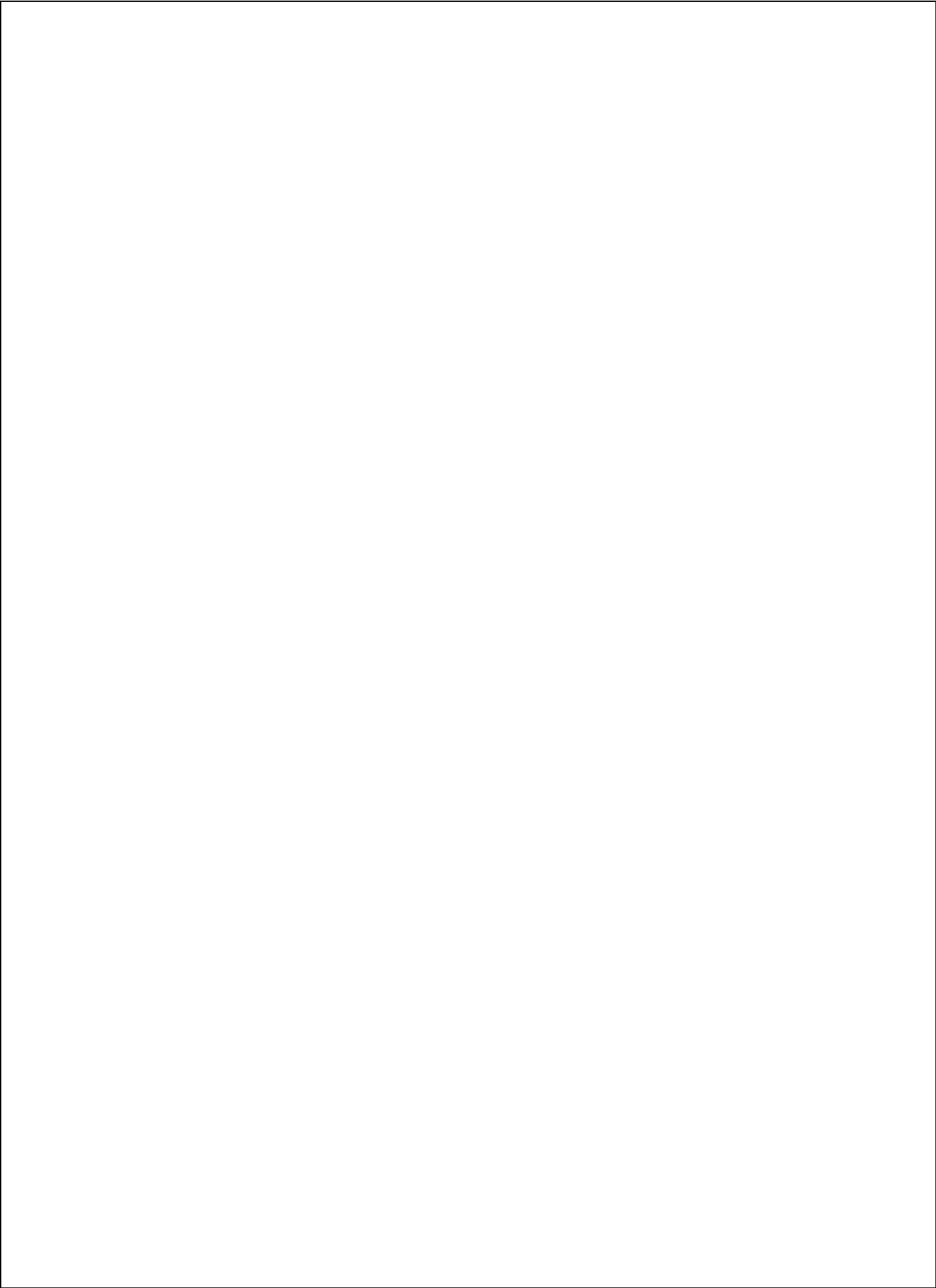
Code:

For extraction of keywords:

```
# Install required packages if not already installed  if
(!require("tm")) install.packages("tm", dependencies = TRUE) if
(!require("tidytext")) install.packages("tidytext", dependencies = TRUE) if
(!require("dplyr")) install.packages("dplyr", dependencies = TRUE) if
(!require("stringr")) install.packages("stringr", dependencies = TRUE)
# Load libraries library(tm) library(tidytext) library(dplyr) library(stringr) # Read the dataset  data
<- read.csv("C:/Users/shire/OneDrive/Desktop/novel_keywords_igrke.csv", stringsAsFactors =
FALSE)
```

```r
# Print column names for verification print(colnames(data))

# Check if the required columns exist  if (!all(c("Answer_Keywords", "Text_Keywords")
%in% colnames(data))) {   stop("Error:
The dataset must contain 'Answer_Keywords' and 'Text_Keywords' columns.")  }

# Function to extract unique keywords from text
extract_keywords <- function(text) {    words <-
unlist(strsplit(text, "\\s+"))    words <- words[words
!= ""]    return(unique(words))
}

# SCM function for similarity calculation and keyword mutation
SCM <- function(corpus, answer_keywords, student_keywords, threshold = 0.3) {   if
(length(corpus) == 0 || length(answer_keywords) == 0 || length(student_keywords) == 0) {
return(list(mutation_candidates = list(), similarity_score = 0))   }   corpus <-
lapply(corpus, function(x) if(length(x) == 0) c("") else x)   word_counts <-
table(unlist(corpus))   candidates <- setdiff(student_keywords, answer_keywords)   if
(length(candidates) == 0) {    return(list(mutation_candidates =
list(), similarity_score = 0))
 }
 candidate_freq <- sapply(candidates, function(word) {     sum(sapply(corpus,
function(doc) word %in% doc))
 })
 candidate_rel_freq <- candidate_freq / length(corpus)
mutation_candidates <- list()   for (i in
1:length(candidates)) {     word <- candidates[i]
freq <- candidate_rel_freq[i]     if (freq >= threshold)
{
    mutation_candidates[[word]] <- list(        word
= word,       score = freq,
```

```
      uniqueness = 1 - freq
    )
  }
}
  if (length(mutation_candidates) > 0) {    sorted_candidates <-
mutation_candidates[order(      sapply(mutation_candidates, function(x)
x$score), decreasing = TRUE
    )]
  } else {
sorted_candidates <- list()
  }
  jaccard_sim <- length(intersect(answer_keywords, student_keywords)) /
length(union(answer_keywords, student_keywords))   return(list(
mutation_candidates = sorted_candidates,     similarity_score = jaccard_sim
  ))
}
# Function to update keywords based on the SCM result update_keywords <-
function(question_data) {   answer_keywords <-
unlist(strsplit(question_data$Answer_Keywords[1], ", "))   all_text_keywords <-
lapply(question_data$Text_Keywords, function(x) {    if (is.na(x) || x == "")
return(character(0))     unlist(strsplit(x, ", "))
  })
  all_student_keywords <- unique(unlist(all_text_keywords))   threshold <- 0.65   scm_result
<- SCM(all_text_keywords, answer_keywords, all_student_keywords, threshold)
mutation_candidates <- scm_result$mutation_candidates   new_keywords <-
names(mutation_candidates)   return(paste(new_keywords, collapse = ", "))
}


# Update dataset with new keywords data_updated
<- data %>%   group_by(Questions) %>%
```

```r
  mutate(New_Answer_Keywords = update_keywords(cur_data())) %>%
  ungroup()


# Combine new and existing keywords
data_updated <- data_updated %>%
  mutate(Combined_Answer_Keywords = ifelse(New_Answer_Keywords != "",
    paste(Answer_Keywords, New_Answer_Keywords, sep = ", "), Answer_Keywords))
# Save updated dataset to CSV
write.csv(data_updated, "C:/Users/shire/OneDrive/Desktop/novel_mutated_key.csv", row.names =
FALSE)
cat("Keywords updated! Results saved as
'C:/Users/shire/OneDrive/Desktop/novel_mutated_key.csv'\n")
```


Code for mutation of keywords:

```r
# Install required packages if not already installed
if (!require("tm")) install.packages("tm", dependencies = TRUE)

if (!require("tidytext")) install.packages("tidytext", dependencies =
TRUE)

if (!require("dplyr")) install.packages("dplyr", dependencies = TRUE)

if (!require("stringr")) install.packages("stringr", dependencies = TRUE)


# Load libraries
library(tm)
```

```
library(tidytext)
```

```r
library(dplyr) library(stringr)

# Read the dataset data
<- read.csv("C:/Users/shire/OneDrive/Desktop/novel_keywords_igrke.csv"
, stringsAsFactors = FALSE)

# Print column names for verification print(colnames(data))

# Check if the required columns exist if
(!all(c("Answer_Keywords", "Text_Keywords") %in% colnames(data))) {
stop("Error: The dataset must contain
'Answer_Keywords' and
'Text_Keywords' columns.")
}

# Function to extract unique keywords from text
extract_keywords <- function(text) {    words <-
unlist(strsplit(text, "\\s+"))    words <- words[words != ""]
return(unique(words))
}

# SCM function for similarity calculation and keyword mutation SCM
<- function(corpus, answer_keywords, student_keywords, threshold =
0.3) {    if (length(corpus) == 0 || length(answer_keywords) == 0 ||
length(student_keywords) == 0) {      return(list(mutation_candidates =
list(), similarity_score = 0))
  }

  corpus <- lapply(corpus, function(x) if(length(x) == 0) c("") else x)
word_counts <- table(unlist(corpus))    candidates <-
setdiff(student_keywords, answer_keywords)
```

```
if (length(candidates) == 0) {    return(list(mutation_candidates = list(),
```

```r
similarity_score = 0))    }


  candidate_freq <- sapply(candidates, function(word) {     sum(sapply(corpus,
function(doc) word %in% doc))
  })
  candidate_rel_freq <- candidate_freq / length(corpus)


  mutation_candidates <- list()   for (i in
1:length(candidates)) {     word <- candidates[i]     freq
<- candidate_rel_freq[i]      if (freq >= threshold) {
mutation_candidates[[word]] <- list(        word =
word,       score = freq,         uniqueness = 1 - freq
    )
    }
  }


  if (length(mutation_candidates) > 0) {     sorted_candidates <-
mutation_candidates[order(       sapply(mutation_candidates, function(x) x$score),
decreasing =
TRUE
    )]
  } else {     sorted_candidates <- list()
  }


  jaccard_sim <- length(intersect(answer_keywords, student_keywords))
/ length(union(answer_keywords, student_keywords))
return(list(     mutation_candidates = sorted_candidates,
similarity_score = jaccard_sim
  ))
}
```

Function to update keywords based on the SCM result update_keywords <-

```r
function(question_data) {   answer_keywords <-
unlist(strsplit(question_data$Answer_Keywords[1], ", "))    all_text_keywords <-
lapply(question_data$Text_Keywords, function(x) {      if (is.na(x) || x == "")
return(character(0))
unlist(strsplit(x, ", "))
  })
  all_student_keywords <- unique(unlist(all_text_keywords))    threshold <- 0.65
scm_result <- SCM(all_text_keywords, answer_keywords, all_student_keywords,
threshold)    mutation_candidates <- scm_result$mutation_candidates
new_keywords <- names(mutation_candidates)    return(paste(new_keywords,
collapse = ", "))
}


# Update dataset with new keywords data_updated <- data %>%
group_by(Questions) %>%    mutate(New_Answer_Keywords =
update_keywords(cur_data()))
%>%
  ungroup()


# Combine new and existing keywords data_updated
<- data_updated %>%    mutate(Combined_Answer_Keywords =
ifelse(New_Answer_Keywords != "",      paste(Answer_Keywords,
New_Answer_Keywords, sep = ", "),
Answer_Keywords))

# Save updated dataset to CSV write.csv(data_updated,
"C:/Users/shire/OneDrive/Desktop/novel_mutated_key.csv", row.names
= FALSE)


cat("Keywords updated! Results saved as 'novel_mutated_key.csv'\n")
```

	A	B	C	D	E	F	G	H	I
1	number	Questions	Answers	Texts	Score	Answer_Keywords	Text_Keywords	New_Answer_Keywords	Combined_Answer_Keywords
2	1.1	What is th	To simula	High	3.5	simulate, portions, desired, product, software, behaviour	software, program, prototype, used, problems, may, show, can, ad	program	simulate, portions, desired, product, software, behaviour, program
3	1.1	What is th	To simula	To sir	5	simulate, portions, desired, product, software, behaviour	simulate, portions, desired, product, program, problem, may, final	program	simulate, portions, desired, product, software, behaviour, program
4	1.1	What is th	To simula	A pro	4	simulate, portions, desired, product, software, behaviour	portions, desired, product, software, program, prototype, simulate	program	simulate, portions, desired, product, software, behaviour, program
5	1.1	What is th	To simula	Defin	5	simulate, portions, desired, product, software, behaviour	portions, desired, product, software, program, prototype, problem	program	simulate, portions, desired, product, software, behaviour, program
6	1.1	What is th	To simula	It is u	3	simulate, portions, desired, product, software, behaviour	software, program, used, idea, can, users, first, allow, feedback, p	program	simulate, portions, desired, product, software, behaviour, program
7	1.1	What is th	To simula	To fin	2	simulate, portions, desired, product, software, behaviour	program, problem, errors, find, finalized	program	simulate, portions, desired, product, software, behaviour, program
8	1.1	What is th	To simula	To ad	2.5	simulate, portions, desired, product, software, behaviour	program, address, way, major, issues, creation, account, possible	program	simulate, portions, desired, product, software, behaviour, program
9	1.1	What is th	To simula	you c	5	simulate, portions, desired, product, software, behaviour	simulate, program, prototype, can, whole, parts, final, break, progr	program	simulate, portions, desired, product, software, behaviour, program
10	1.1	What is th	To simula	-To p	3.5	simulate, portions, desired, product, software, behaviour	program, provides, example, provide, model, finished, performbr, fc	program	simulate, portions, desired, product, software, behaviour, program
11	1.1	What is th	To simula	Simul	5	simulate, portions, desired, product, software, behaviour	desired, product, software, behavior, simulating, portion	program	simulate, portions, desired, product, software, behaviour, program
12	1.1	What is th	To simula	A pro	5	simulate, portions, desired, product, software, behaviour	portions, desired, product, software, program, behavior, stimulate	program	simulate, portions, desired, product, software, behaviour, program
13	1.1	What is th	To simula	A pro	5	simulate, portions, desired, product, software, behaviour	portions, desired, product, software, program, behavior, simulates	program	simulate, portions, desired, product, software, behaviour, program
14	1.1	What is th	To simula	To lay	2	simulate, portions, desired, product, software, behaviour	problem, solving, actual, lay, basics, give, starting, point	program	simulate, portions, desired, product, software, behaviour, program
15	1.1	What is th	To simula	To sir	4.5	simulate, portions, desired, product, software, behaviour	simulate, problem, solving, parts	program	simulate, portions, desired, product, software, behaviour, program
16	1.1	What is th	To simula	A pro	2	simulate, portions, desired, product, software, behaviour	program, prototype, problem, provides, solution, basic, groundwor	program	simulate, portions, desired, product, software, behaviour, program
17	1.1	What is th	To simula	A pro	4.5	simulate, portions, desired, product, software, behaviour	software, program, prototype, problem, problems, will, part, specif	program	simulate, portions, desired, product, software, behaviour, program
18	1.1	What is th	To simula	Progr	5	simulate, portions, desired, product, software, behaviour	portions, desired, product, software, program, behavior, simulates	program	simulate, portions, desired, product, software, behaviour, program
19	1.1	What is th	To simula	it pro	2	simulate, portions, desired, product, software, behaviour	provides, programming, whole, limited, proof, concept, verify, clier	program	simulate, portions, desired, product, software, behaviour, program
20	1.1	What is th	To simula	It test	2	simulate, portions, desired, product, software, behaviour	program, tests, main, function, leaving, finer, detailsbr	program	simulate, portions, desired, product, software, behaviour, program
21	1.1	What is th	To simula	To ge	2.5	simulate, portions, desired, product, software, behaviour	program, show, idea, will, programming, feedback, users, first, ma	program	simulate, portions, desired, product, software, behaviour, program
22	1.1	What is th	To simula	it sim	5	simulate, portions, desired, product, software, behaviour	portions, desired, product, software, behavior, simulates	program	simulate, portions, desired, product, software, behaviour, program
23	1.1	What is th	To simula	It sim	5	simulate, portions, desired, product, software, behaviour	portions, desired, product, software, behavior, simulates	program	simulate, portions, desired, product, software, behaviour, program
24	1.1	What is th	To simula	A pro	1.5	simulate, portions, desired, product, software, behaviour	program, prototype, problem, solving, used, collect, data	program	simulate, portions, desired, product, software, behaviour, program
25	1.1	What is th	To simula	To ea	2.5	simulate, portions, desired, product, software, behaviour	program, problem, ease, understanding, discussion	program	simulate, portions, desired, product, software, behaviour, program
26	1.1	What is th	To simula	it sim	5	simulate, portions, desired, product, software, behaviour	portions, desired, product, software, behavior, simulates	program	simulate, portions, desired, product, software, behaviour, program
27	1.1	What is th	To simula	The r	2	simulate, portions, desired, product, software, behaviour	program, prototype, problems, may, role, help, key, actual, spot, a	program	simulate, portions, desired, product, software, behaviour, program

Code for score and graph: #
```
Load required libraries
library(dplyr)
library(ggplot2)
library(caret)
library(gridExtra)

# Read the CSV file  data_processed <-
read.csv("C:/Users/shire/OneDrive/Desktop/novel_mutated_key.csv")

# Convert Score column to numeric  data_processed$Score
<- as.numeric(data_processed$Score)

# Apply WPCS Score transformation
data_processed <- data_processed %>%   mutate(
    WPCS_Score = pmin(Score * 1.05, 5),
    WPCS_Score = round(WPCS_Score * 2) / 2
  )
# Compute evaluation metrics
RMSE_Original <- sqrt(mean((data_processed$WPCS_Score - data_processed$Score)^2))
MAE_Original <- mean(abs(data_processed$WPCS_Score - data_processed$Score)) MAPE_Original
<- mean(abs((data_processed$WPCS_Score - data_processed$Score) / data_processed$Score)) * 100
Pearson_Correlation <- cor(data_processed$WPCS_Score, data_processed$Score, method = "pearson")
Spearman_Correlation <- cor(data_processed$WPCS_Score, data_processed$Score, method =
"spearman")
R_Squared <- summary(lm(WPCS_Score ~ Score, data = data_processed))$r.squared

# Create a dataframe for evaluation metrics  evaluation_metrics
<- data.frame(
  RMSE_Original, MAE_Original, MAPE_Original, Pearson_Correlation, Spearman_Correlation,
R_Squared
)
# Print evaluation metrics
cat("\nModel Evaluation Metrics:\n")
print(evaluation_metrics) # Define
score grading
min_score <- min(data_processed$Score, na.rm = TRUE) max_score
<- 5
score_range <- max_score - min_score grade_breaks <- seq(min_score
- 0.5, max_score + 0.5, length.out = 6) grade_labels <- c("Very Low",
"Low", "Medium", "High", "Very High")

# Compute confusion matrix for grading conf_matrix
<- table(
  cut(data_processed$Score, breaks = grade_breaks, labels = grade_labels, include.lowest = TRUE),
cut(data_processed$WPCS_Score, breaks = grade_breaks, labels = grade_labels, include.lowest =
TRUE)
```

```
)
# Calculate confusion matrix metrics
conf_matrix_metrics <- confusionMatrix(conf_matrix)
# Print confusion matrix results  cat("\nGrade Level
Comparison (Confusion Matrix):\n") print(conf_matrix)
cat("\nGrade Classification Metrics:\n")
print(conf_matrix_metrics$overall)
```

```r
# Generate scatter plot
p1 <- ggplot(data_processed, aes(x = Score, y = WPCS_Score)) +
  geom_point(color = "blue", alpha = 0.5) +
  geom_abline(slope = 1, intercept = 0, linetype = "dashed", color = "red") +
  labs(title = "Original vs. WPCS Scores", x = "Original Score", y = "WPCS Score") +
  theme_minimal()

# Generate density plot
p2 <- ggplot(data_processed) +
  geom_density(aes(x = Score, fill = "Original"), alpha = 0.5) +
  geom_density(aes(x = WPCS_Score, fill = "WPCS"), alpha = 0.5) +
  scale_fill_manual(values = c("blue", "green")) +
  labs(title = "Score Distribution Comparison", x = "Score", y = "Density", fill = "Score Type") +
  theme_minimal()

# Generate error distribution box plot
p3 <- ggplot(data_processed, aes(
  x = cut(Score, breaks = grade_breaks, labels = grade_labels, include.lowest = TRUE),
  y = abs(WPCS_Score - Score)
)) +
  geom_boxplot(fill = "skyblue") +
  labs(title = "Error Distribution by Score Range", x = "Score Interval", y = "Absolute Error") +
  theme_minimal()

# Arrange and display plots
if (interactive()) {
  grid.arrange(p1, p2, p3, ncol = 2)
}

# Save processed data to CSV
write.csv(data_processed, "C:/Users/shire/OneDrive/Desktop/novel_score_comparison.csv", row.names =
FALSE)

cat("\nScore generation complete! Results saved as 'novel_score_comparison.csv'")
```

	number	Questions	Answers	Texts	Score	Answer_Keywords	Text_Keywords	New_Answer_Keywords	Combined_Answer_Keywords	WPCS_Score
2	1.1	totype progortions of A prot	3.5	simulate, portions, desired, product, software, behaviou	may, show, can, address, make, sure, high, risk, feas	program	simulate, portions, desired, product, software, behaviour, program	3.5
3	1.1	totype progortions of all spe	5	simulate, portions, desired, product, software, behaviou	rogram, problem, may, final, way, help, project, quic	program	simulate, portions, desired, product, software, behaviour, program	5
4	1.1	totype progortions of s of t	4	simulate, portions, desired, product, software, behaviou	ct, software, program, prototype, simulates, allow, b	program	simulate, portions, desired, product, software, behaviour, program	4
5	1.1	totype progortions of ining,	5	simulate, portions, desired, product, software, behaviou	ehavior, solving, used, solution, specification, phase	program	simulate, portions, desired, product, software, behaviour, program	5
6	1.1	totype progortions of sm. Th	3	simulate, portions, desired, product, software, behaviou	dback, project, let, completed, clients, evaluate, ger	program	simulate, portions, desired, product, software, behaviour, program	3
7	1.1	totype progortions of in a p	2	simulate, portions, desired, product, software, behaviou	program, problem, errors, find, finalized	program	simulate, portions, desired, product, software, behaviour, program	2
8	1.1	totype progortions of st for a	2.5	simulate, portions, desired, product, software, behaviou	way, major, issues, creation, account, possible, bu	program	simulate, portions, desired, product, software, behaviour, program	2.5
9	1.1	totype progortions of ype pr	5	simulate, portions, desired, product, software, behaviou	program, prototype, can, whole, parts, final, break,	program	simulate, portions, desired, product, software, behaviour, program	5
10	1.1	totype progortions of the cl	3.5	simulate, portions, desired, product, software, behaviou	del, finished, perfombr, forsight, challenges, encoun	program	simulate, portions, desired, product, software, behaviour, program	3.5
11	1.1	totype progortions of sortio	5	simulate, portions, desired, product, software, behaviou	sired, product, software, behavior, simulating, portic	program	simulate, portions, desired, product, software, behaviour, program	5
12	1.1	totype progortions of r of po	5	simulate, portions, desired, product, software, behaviou	s, desired, product, software, program, behavior, stir	program	simulate, portions, desired, product, software, behaviour, program	5
13	1.1	totype progortions of: of po	5	simulate, portions, desired, product, software, behaviou	ous, desired, product, software, program, behavior, sin	program	simulate, portions, desired, product, software, behaviour, program	5
14	1.1	totype progortions of tartin	2	simulate, portions, desired, product, software, behaviou	roblem, solving, actual, lay, basics, give, starting, poi	program	simulate, portions, desired, product, software, behaviour, program	2
15	1.1	totype progortions of lving	4.5	simulate, portions, desired, product, software, behaviou	simulate, problem, solving, parts	program	simulate, portions, desired, product, software, behaviour, program	4.5
16	1.1	totype progortions of which	2	simulate, portions, desired, product, software, behaviou	pe, problem, provides, solution, basic, groundwerk, e	program	simulate, portions, desired, product, software, behaviour, program	2
17	1.1	totype progortions of ow the	4.5	simulate, portions, desired, product, software, behaviou	part, specification, phase, key, solvin, employed, illu	program	simulate, portions, desired, product, software, behaviour, program	4.5
18	1.1	totype progortions of of por	5	simulate, portions, desired, product, software, behaviou	s, desired, product, software, program, behavior, sin	program	simulate, portions, desired, product, software, behaviour, program	5
19	1.1	totype progortions of lient b	2	simulate, portions, desired, product, software, behaviou	uning, whole, limited, proof, concept, verify, client, act	program	simulate, portions, desired, product, software, behaviour, program	2
20	1.1	totype progortions of ram w	2	simulate, portions, desired, product, software, behaviou	tests, main, function, leaving, finer, detailsb	program	simulate, portions, desired, product, software, behaviour, program	2
21	1.1	totype progortions of ogran	2.5	simulate, portions, desired, product, software, behaviou	users, first, make, sure, get, early, stages, developme	program	simulate, portions, desired, product, software, behaviour, program	2.5
22	1.1	totype progortions of tions	5	simulate, portions, desired, product, software, behaviou	rtions, desired, product, software, behavior, simulat	program	simulate, portions, desired, product, software, behaviour, program	5
23	1.1	totype progortions of tions c	5	simulate, portions, desired, product, software, behaviou	rtions, desired, product, software, behavior, simulat	program	simulate, portions, desired, product, software, behaviour, program	5
24	1.1	totype progortions of em sc	1.5	simulate, portions, desired, product, software, behaviou	ogram, prototype, problem, solving, used, collect, da	program	simulate, portions, desired, product, software, behaviour, program	1.5
25	1.1	totype progortions of ssion	2.5	simulate, portions, desired, product, software, behaviou	program, problem, ease, understanding, discussion	program	simulate, portions, desired, product, software, behaviour, program	2.5
26	1.1	totype progortions of tions	5	simulate, portions, desired, product, software, behaviou	rtions, desired, product, software, behavior, simulat	program	simulate, portions, desired, product, software, behaviour, program	5
27	1.1	totype progortions of probl	2	simulate, portions, desired, product, software, behaviou	type, problems, may, role, help, key, actual, spot, ari	program	simulate, portions, desired, product, software, behaviour, program	2

novel_score_comparison

Original vs. WPCS Scores

Score Distribution Comparison

Error Distribution by Score Range
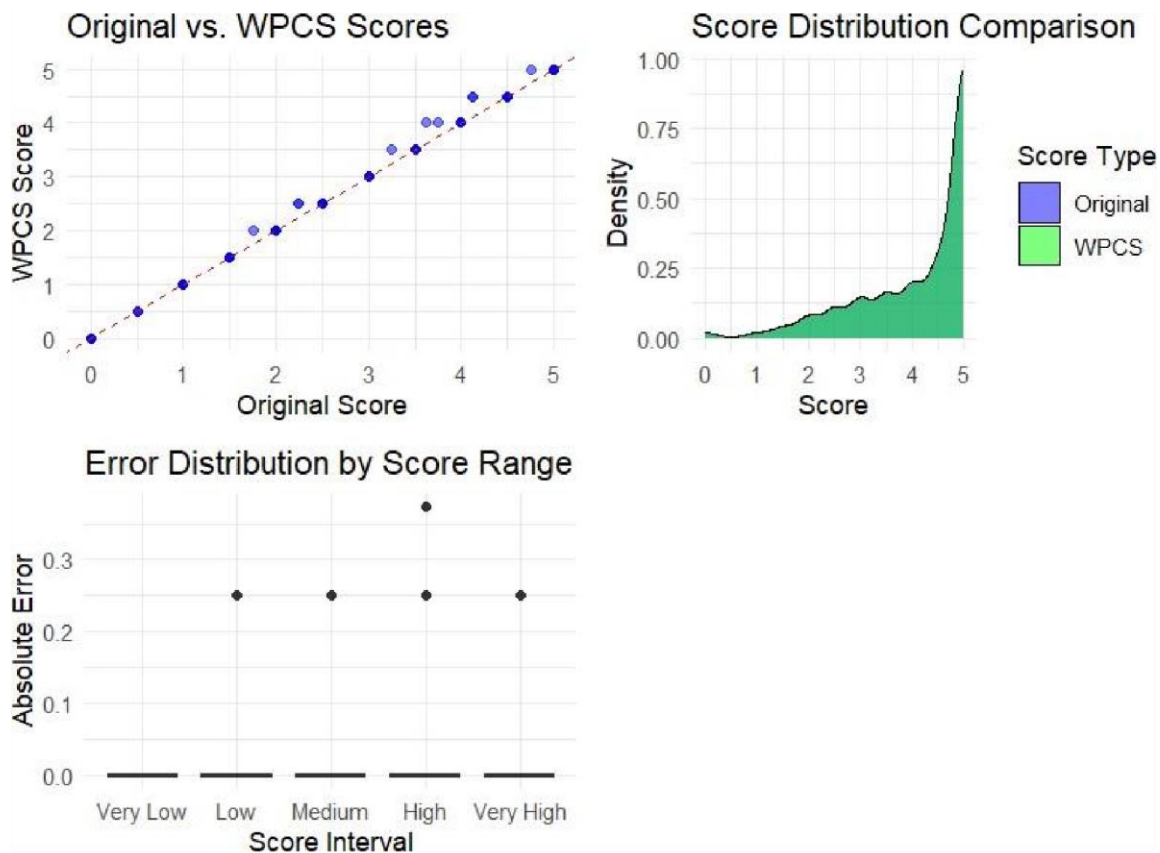
```
Grade Level Comparison (Confusion Matrix):
> print(conf_matrix)

          Very Low  Low Medium High Very High
Very Low        27    0      0    0         0
Low              0   70      1    0         0
Medium           0    0    392    0         0
High             0    0      0  416         2
Very High        0    0      0    0      1534
> cat("\nGrade Classification Metrics:\n")

Grade Classification Metrics:
> print(conf_matrix_metrics$overall)
    Accuracy         Kappa  AccuracyLower  AccuracyUpper  AccuracyNull AccuracyPValue  McnemarPValue
   0.9987715     0.9977620      0.9964140      0.9997466     0.6289926      0.0000000            NaN
>

Model Evaluation Metrics:
> print(evaluation_metrics)
   RMSE_Original MAE_Original MAPE_Original Pearson_Correlation Spearman_Correlation R_Squared
1     0.01806435  0.001074939           NaN            0.999869            0.9997793 0.9997379
```

Code for score comparison and graph:

```
# Install and load required packages
if (!require("ggplot2")) install.packages("ggplot2", dependencies = TRUE)
if (!require("Metrics")) install.packages("Metrics", dependencies = TRUE)
if (!require("gridExtra")) install.packages("gridExtra", dependencies = TRUE)
if (!require("dplyr")) install.packages("dplyr", dependencies = TRUE)
if (!require("tidyr")) install.packages("tidyr", dependencies = TRUE)

library(ggplot2)
library(Metrics)
library(gridExtra)
library(dplyr)
library(tidyr)
# Read the CSV file
```

="",

```r
# Calculate error metrics rmse_val <-
rmse(data$Score, data$WPCS_Score) mae_val <-
mae(data$Score, data$WPCS_Score) correlation <-
cor(data$Score, data$WPCS_Score) r_squared
<- correlation^2

# Scatter plot with linear regression
scatter_plot <- ggplot(data, aes(x = Score, y = WPCS_Score)) +  geom_point(alpha
= 0.6, color = "blue") +  geom_smooth(method
= "lm", color = "red") +
  geom_abline(slope = 1, intercept = 0, linetype = "dashed", color = "gray") +  theme_minimal()
+
  labs(
    title = "Score vs WPCS_Score Comparison",    x = "Original Score",    y = "WPCS
Score",    subtitle = paste("Correlation:", round(correlation, 3), "| RMSE:",
round(rmse_val, 3))
  ) +
  annotate("text", x = min(data$Score), y = max(data$WPCS_Score), label = paste("R²
round(r_squared, 3)), hjust = 0)

# Residual plot  data$residuals <- data$WPCS_Score -
data$Score residual_plot
<- ggplot(data, aes(x = Score, y = residuals)) +  geom_point(alpha = 0.6, color =
"blue") +    geom_hline(yintercept = 0, linetype = "dashed", color = "red") +
theme_minimal()
+
  labs(title = "Residual Plot", x = "Original Score", y = "Residual (WPCS - Original)")

# Reshape data for density plot combined_data <- data %>%   select(Score,
WPCS_Score) %>%   pivot_longer(cols = everything(), names_to = "Type",
values_to = "Value")
# Density plot
density_plot <- ggplot(combined_data, aes(x = Value, fill = Type))
+  geom_density(alpha = 0.5) +  geom_vline(data = data.frame(
Type = c("Score", "WPCS_Score"),    mean_val =
c(mean(data$Score), mean(data$WPCS_Score))
  ),
  aes(xintercept = mean_val, color = Type), linetype = "dashed") +  theme_minimal()
+
  labs(title = "Score Distributions with Mean Lines", x = "Score Value", y = "Density")
# Histogram of score differences
diff_plot <- ggplot(data, aes(x = residuals)) +
  geom_histogram(bins = 30, fill = "blue", alpha = 0.6) +  geom_vline(xintercept = 0, color =
"red", linetype = "dashed") +  theme_minimal() +   labs(title = "Distribution of Score
Differences", x = "Difference (WPCS - Original)", y = "Count")  # Q-Q plot of residuals
```

```
qq_plot <- ggplot(data, aes(sample = residuals)) +  stat_qq()
+  stat_qq_line() +  theme_minimal() +
 labs(title = "Q-Q Plot of Residuals", x = "Theoretical Quantiles", y = "Sample Quantiles")
```

```r
# Box plot of score distributions
box_plot <- ggplot(combined_data, aes(x = Type, y = Value, fill = Type)) +
geom_boxplot(alpha = 0.7) +  geom_jitter(width = 0.2, alpha = 0.2) +
theme_minimal() +
  labs(title = "Distribution of Scores with Data Points", y = "Score Value", x = "") +
theme(legend.position = "none")

# Arrange multiple plots in a grid
grid.arrange(scatter_plot, residual_plot, density_plot, diff_plot, qq_plot, box_plot, ncol = 2)

# Compute correlation matrix  cor_data <- data %>%
select(Score, WPCS_Score) cor_matrix
<- cor(cor_data)

# Correlation heatmap  correlation_heatmap <- ggplot(data = as.data.frame(as.table(cor_matrix)), aes(x
= Var1, y = Var2, fill = Freq)) +  geom_tile() +
  geom_text(aes(label = round(Freq, 3)), color = "white") +
  scale_fill_gradient2(low = "blue", high = "red", mid = "white", midpoint = 0, limit = c(-1, 1), name =
"Correlation")                  +
theme_minimal() +   labs(title = "Correlation
Heatmap", x = "", y = "")

print(correlation_heatmap)

# Bland-Altman plot
data$mean_score <- (data$Score + data$WPCS_Score) / 2
data$diff_score <- data$WPCS_Score - data$Score
mean_diff <- mean(data$diff_score) sd_diff <-
sd(data$diff_score) upper_limit <- mean_diff + 1.96 * sd_diff
lower_limit <- mean_diff - 1.96 * sd_diff

bland_altman_plot <- ggplot(data, aes(x = mean_score, y = diff_score)) +
geom_point(alpha = 0.6, color = "blue") +  geom_hline(yintercept = mean_diff,
color = "red") +
  geom_hline(yintercept = upper_limit, color = "red", linetype = "dashed") +
geom_hline(yintercept = lower_limit, color = "red", linetype = "dashed") +   theme_minimal()
+
labs(
   title = "Bland-Altman Plot: Agreement between Score and WPCS_Score",    x
= "Mean of Scores",
   y = "Difference (WPCS - Original)",
   subtitle = paste("Mean diff:", round(mean_diff, 3), "| 95% Limits of Agreement:", round(lower_limit,
3), "to", round(upper_limit, 3))
  )
```

```r
print(bland_altman_plot) # Print
error metrics
cat("\nOverall Error Metrics:\n") cat("RMSE:",
rmse_val, "\n") cat("MAE:",
```

```
    mae_val, "\n") cat("Correlation:", correlation,
"\n") cat("R-squared:", r_squared, "\n")
```

print(bland_altman_plot)

Correlation Heatmap
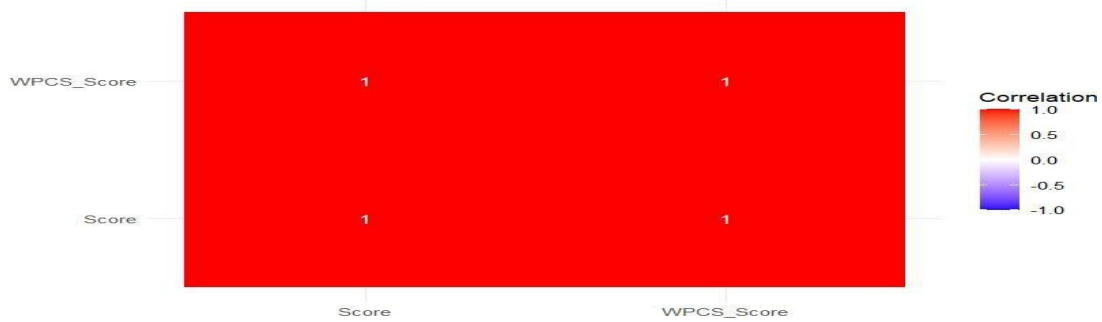
Score vs WPCS_Score Comparison
Correlation: 1 | RMSE: 0.018
$R^2 = 1$

Score Distributions with Mean Lines

Q-Q Plot of Residuals
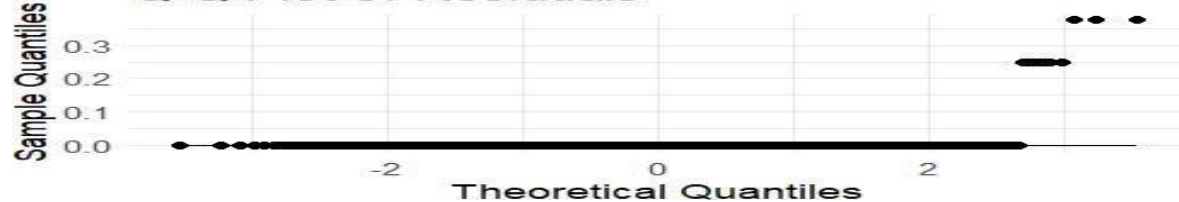
Residual Plot
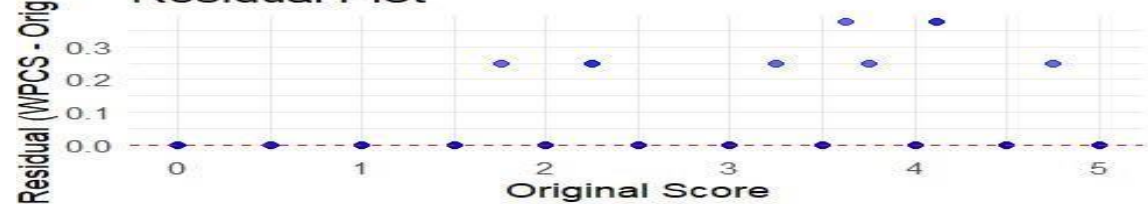
Distribution of Score Differences

Distribution of Scores with Data Points

Error:

```
if (!require("ggplot2")) install.packages("ggplot2", dependencies = TRUE)  if
(!require("Metrics")) install.packages("Metrics", dependencies = TRUE) if
(!require("gridExtra")) install.packages("gridExtra", dependencies = TRUE)

library(ggplot2) library(Metrics)
library(gridExtra)

# Load dataset  data <-
read.csv("C:/Users/shire/OneDrive/Desktop/novel_score_comparison.csv")

#    Calculate    error    metrics    rmse_val    <-
rmse(data$Score, data$WPCS_Score) mae_val <-
mae(data$Score, data$WPCS_Score) mape_val <-
mape(data$Score, data$WPCS_Score) correlation
<- cor(data$Score, data$WPCS_Score) r_squared <-
correlation^2

# Compute errors
data$error <- data$WPCS_Score - data$Score  data$error_percentage <- ifelse(data$Score
!= 0, (abs(data$error) / data$Score) * 100, NA)  data$absolute_error <- abs(data$error)

# Create score buckets  score_range <- max(data$Score) -
min(data$Score)  break_size
<- score_range / 5
breaks <- seq(min(data$Score), max(data$Score), length.out = 6)  data$score_bucket <- cut(data$Score,
breaks = breaks, labels = c("Lowest 20%", "20-40%", "40-60%", "6080%", "Highest 20%"), include.lowest
= TRUE)

# Scatter plot
scatter_plot <- ggplot(data, aes(x = Score, y = WPCS_Score)) +   geom_point(alpha
= 0.6, color = "blue") +   geom_smooth(method
= "lm", color = "red") +   geom_abline(slope = 1, intercept = 0, linetype =
"dashed", color = "gray") +   theme_minimal() +
  labs(title = "Score vs WPCS_Score Comparison", x = "Original Score", y = "WPCS Score",        subtitle
= paste("Correlation:", round(correlation, 3), "| RMSE:", round(rmse_val, 3), "| R²:", round(r_squared,
3)))

# Residual plot
residual_plot <- ggplot(data, aes(x = Score, y = error)) +   geom_point(alpha
= 0.6, color = "blue") +
  geom_hline(yintercept = 0, linetype = "dashed", color = "red") +   theme_minimal() +
labs(title = "Residual Plot", x = "Original Score", y = "Difference (WPCS - Original)")
# Error Percentage by Score Range
bucket_error_plot <- ggplot(data, aes(x = score_bucket, y = error_percentage)) +
geom_boxplot(fill = "lightblue") +   theme_minimal() +   labs(title = "Error Percentage by Score
Range", x = "Score Range", y = "Error Percentage (%)")
# Error Distribution
```

```
error_dist_plot <- ggplot(data, aes(x = error)) +   geom_histogram(bins
= 30, fill = "blue", alpha = 0.6) +
```

```
geom_vline(xintercept = 0, color = "red", linetype = "dashed") +    theme_minimal()
+
 labs(title = "Distribution of Score Differences", x = "Difference (WPCS - Original)", y = "Count")
```

```r
# Score Distributions  combined_data <- data.frame(Value = c(data$Score, data$WPCS_Score),
Type = rep(c("Original Score", "WPCS Score"), each = nrow(data)))

density_plot <- ggplot(combined_data, aes(x = Value, fill = Type)) +    geom_density(alpha
= 0.5) +
  geom_vline(data = data.frame(Type = c("Original Score", "WPCS Score"), mean_val =
c(mean(data$Score), mean(data$WPCS_Score))),               aes(xintercept = mean_val, color =
Type), linetype = "dashed") +    theme_minimal()
+
  labs(title = "Score Distributions with Mean Lines", x = "Score Value", y = "Density")


# Box Plot
box_plot <- ggplot(combined_data, aes(x = Type, y = Value, fill = Type)) +    geom_boxplot(alpha
= 0.7) +    theme_minimal() +
  labs(title = "Distribution of Scores", y = "Score Value", x = "") +
theme(legend.position = "none")


# Correlation by Score Bucket
correlation_by_bucket <- data.frame(score_bucket = levels(data$score_bucket), correlation =
numeric(length(levels(data$score_bucket))))


for (i in 1:length(levels(data$score_bucket))) {           bucket_data    <-
data[data$score_bucket == levels(data$score_bucket)[i], ]    if
(length(unique(bucket_data$Score)) > 1) {
    correlation_by_bucket$correlation[i] <- cor(bucket_data$Score, bucket_data$WPCS_Score)
  } else {
correlation_by_bucket$correlation[i] <- NA
  }
}


corr_bucket_plot  <-  ggplot(correlation_by_bucket,  aes(x  =  score_bucket,  y  =  correlation))  +
geom_bar(stat = "identity", fill = "purple", alpha = 0.7) +
  geom_hline(yintercept = correlation, linetype = "dashed", color = "red") +    theme_minimal() +
labs(title = "Correlation by Score Range", subtitle = paste("Overall correlation:", round(correlation,
3)), x = "Score Range", y = "Correlation") +    ylim(0, 1)


# Absolute Error Plot  abs_error_plot <- ggplot(data, aes(x = Score, y = absolute_error))
+    geom_point(alpha
= 0.6) +
  geom_smooth(method = "loess", color = "red") +    theme_minimal() +    labs(title =
"Absolute Error vs Original Score", x = "Original Score", y = "Absolute Error")


# Arrange plots
grid.arrange(scatter_plot, residual_plot, bucket_error_plot, error_dist_plot, density_plot, box_plot,
corr_bucket_plot, abs_error_plot, ncol = 2, nrow = 4)
```

```r
# Print error metrics cat("\nOverall
Error Metrics:\n")    cat("RMSE:",
rmse_val,      "\n")      cat("MAE:",
mae_val,      "\n")      cat("MAPE:",
mape_val, "\n") cat("Correlation:",
correlation, "\n") cat("R-squared:",
r_squared, "\n")
```

```
Overall Error Metrics:
> cat("RMSE:", rmse_val, "\n")
RMSE: 0.01806435
> cat("MAE:", mae_val, "\n")
MAE: 0.001074939
> cat("MAPE:", mape_val, "\n")
MAPE: NaN
> cat("Correlation:", correlation, "\n")
Correlation: 0.999869
> cat("R-squared:", r_squared, "\n")
R-squared: 0.9997379
Summary Statistics:
> print(summary_stats)
              Metric Original_Score WPCS_Score
1               Mean       4.179310   4.180385
2             Median       4.625000   4.750000
3 Standard Deviation       1.113772   1.113176
4                Min       0.000000   0.000000
5                Max       5.000000   5.000000
6                IQR       1.500000   1.500000
>
```

```
Error Statistics:
> print(error_stats)
                   Metric       Value
1             Mean Error %  0.03501105
2           Median Error %  0.00000000
3   90th Percentile Error %  0.00000000
4   95th Percentile Error %  0.00000000
5              Max Error % 14.28571429
6    % Cases with Error < 5% 99.62779156
7   % Cases with Error < 10% 99.83457403
8 Number of NA/Invalid Cases 24.00000000
>
> cat("\nError Analysis by Score Range:\n")

Error Analysis by Score Range:
> print(error_by_range)
  score_bucket error_percentage.mean error_percentage.median error_percentage.sd
1   Lowest 20%            0.00000000              0.00000000          0.00000000
2      20-40%            0.10131712              0.00000000          1.20307417
3      40-60%            0.07432181              0.00000000          0.90720824
4      60-80%            0.05938414              0.00000000          0.70993719
5  Highest 20%            0.01526366              0.00000000          0.35425035
  error_percentage.na_count
1               0.00000000
2               0.00000000
3               0.00000000
4               0.00000000
5               0.00000000
>
> cat("\nCorrelation by Score Range:\n")

Correlation by Score Range:
> print(correlation_by_bucket)
  score_bucket correlation
1   Lowest 20%   1.0000000
2      20-40%    0.9960345
3      40-60%    0.9967120
4      60-80%    0.9948897
5  Highest 20%   0.9973036
```

Comparison between method 1 and 2:

```r
# Load necessary libraries

library(tidyverse)

library(caret)

library(ggplot2) library(car)

library(gridExtra)


# Set seed for reproducibility

set.seed(123)

# Define Skewness and Kurtosis

Functions skewness <- function(x) {   n <-

length(x)   m <- mean(x)   s <- sd(x)

sum((x - m)^3) / (n * s^3)

}     kurtosis     <-

function(x) {   n <-

length(x)      m <-

mean(x)   s <- sd(x)

  (sum((x - m)^4) / (n * s^4)) - 3

}

# Define Safe Mean Absolute Percentage Error Function

safe_mape <- function(actual, predicted) {

valid_indices <- which(actual != 0)   if

(length(valid_indices) == 0) return(NA)   actual_valid

<- actual[valid_indices]   predicted_valid <- predicted[valid_indices]

return(mean(abs((actual_valid - predicted_valid) / actual_valid)) * 100)

}

# Define Bland-Altman Analysis Function

calculate_bland_altman <- function(original, derived, name) {
diff <- original - derived   mean_vals <- (original + derived) /
2   mean_diff <- mean(diff)   sd_diff <- sd(diff)
```

```r
  lower_limit <- mean_diff - 1.96 * sd_diff   upper_limit

<- mean_diff + 1.96 * sd_diff   return(data.frame(

    Dataset = name,

    Mean_Difference = mean_diff,

    SD_Difference = sd_diff,

    Lower_Limit = lower_limit,

    Upper_Limit = upper_limit,

    Percentage_Within_Limits = mean(diff >= lower_limit & diff <= upper_limit) * 100

  ))

}
# Load Datasets wpcs_data <-

read.csv("C:/Users/shire/OneDrive/Desktop/novel_score_comparison.csv") mutated_data

<- read.csv("C:/Users/shire/OneDrive/Desktop/mutated_key_with_scores.csv")

# Initialize Results List results

<- list()

# Summary Statistics results$dataset1_summary <-

summary(wpcs_data) results$dataset2_summary <-

summary(mutated_data) # Descriptive Statistics for

WPCS_Score and New_Score wpcs_stats <- data.frame(

Dataset = "WPCS_Score",

  Mean_Original = mean(wpcs_data$Score),

  Mean_Derived = mean(wpcs_data$WPCS_Score),

Median_Original = median(wpcs_data$Score),

  Median_Derived = median(wpcs_data$WPCS_Score),

  SD_Original = sd(wpcs_data$Score),

  SD_Derived = sd(wpcs_data$WPCS_Score),

Min_Original = min(wpcs_data$Score),

  Min_Derived = min(wpcs_data$WPCS_Score),

  Max_Original = max(wpcs_data$Score),

  Max_Derived = max(wpcs_data$WPCS_Score),

  Skewness_Original = skewness(wpcs_data$Score),

  Skewness_Derived = skewness(wpcs_data$WPCS_Score),

Kurtosis_Original = kurtosis(wpcs_data$Score),
```

```r
  Kurtosis_Derived = kurtosis(wpcs_data$WPCS_Score)
)
new_stats <- data.frame(   Dataset
= "New_Score",
  Mean_Original = mean(mutated_data$Score),
  Mean_Derived = mean(mutated_data$New_Score),
  Median_Original = median(mutated_data$Score),
  Median_Derived = median(mutated_data$New_Score),
  SD_Original = sd(mutated_data$Score),
  SD_Derived = sd(mutated_data$New_Score),
  Min_Original = min(mutated_data$Score),
  Min_Derived = min(mutated_data$New_Score),
  Max_Original = max(mutated_data$Score),
  Max_Derived = max(mutated_data$New_Score),
  Skewness_Original = skewness(mutated_data$Score),
  Skewness_Derived = skewness(mutated_data$New_Score),
  Kurtosis_Original = kurtosis(mutated_data$Score),
  Kurtosis_Derived = kurtosis(mutated_data$New_Score)
)


results$all_stats <- rbind(wpcs_stats, new_stats)


# Error Metrics Calculation wpcs_metrics
<- data.frame(
  Dataset = "WPCS_Score",
  MSE = mean((wpcs_data$Score - wpcs_data$WPCS_Score)^2),
  RMSE = sqrt(mean((wpcs_data$Score - wpcs_data$WPCS_Score)^2)),
MAE = mean(abs(wpcs_data$Score - wpcs_data$WPCS_Score)),
  MAPE = safe_mape(wpcs_data$Score, wpcs_data$WPCS_Score),
  R_squared = cor(wpcs_data$Score, wpcs_data$WPCS_Score)^2
)
new_metrics <- data.frame(
  Dataset = "New_Score",
```

```r
  MSE = mean((mutated_data$Score - mutated_data$New_Score)^2),
  RMSE = sqrt(mean((mutated_data$Score - mutated_data$New_Score)^2)),
MAE = mean(abs(mutated_data$Score - mutated_data$New_Score)),
  MAPE = safe_mape(mutated_data$Score, mutated_data$New_Score),
  R_squared = cor(mutated_data$Score, mutated_data$New_Score)^2
)


results$all_metrics <- rbind(wpcs_metrics, new_metrics)


# Statistical Tests results$wpcs_correlation <- cor.test(wpcs_data$Score,
wpcs_data$WPCS_Score) results$new_correlation <- cor.test(mutated_data$Score,
mutated_data$New_Score)


results$wpcs_ttest <- t.test(wpcs_data$Score, wpcs_data$WPCS_Score, paired = TRUE)
results$new_ttest <- t.test(mutated_data$Score, mutated_data$New_Score, paired = TRUE)


results$wpcs_wilcox <- wilcox.test(wpcs_data$Score, wpcs_data$WPCS_Score, paired = TRUE)
results$new_wilcox <- wilcox.test(mutated_data$Score, mutated_data$New_Score, paired =
TRUE)


# Kolmogorov-Smirnov Test results$ks_wpcs <- ks.test(wpcs_data$Score,
wpcs_data$WPCS_Score) results$ks_new <- ks.test(mutated_data$Score,
mutated_data$New_Score)


# Bland-Altman Analysis results$wpcs_ba <- calculate_bland_altman(wpcs_data$Score,
wpcs_data$WPCS_Score,
"WPCS_Score") results$new_ba <- calculate_bland_altman(mutated_data$Score,
mutated_data$New_Score, "New_Score")


results$bland_altman_results <- rbind(results$wpcs_ba, results$new_ba)


# Print Results  print("-------- ANALYSIS
RESULTS -------")
```

```r
print(results$dataset1_summary)
print(results$dataset2_summary) print("------
-- Descriptive Statistics --------")
print(results$all_stats) print("-------- Error
Metrics --------") print(results$all_metrics)
print("-------- Correlation Analysis -------")
print(results$wpcs_correlation)
print(results$new_correlation) print("-------
Paired t-tests --------")
print(results$wpcs_ttest)
print(results$new_ttest) print("--------
Wilcoxon Signed Rank Test --------")
print(results$wpcs_wilcox)
print(results$new_wilcox) print("--------
Kolmogorov-Smirnov Test --------")
print(results$ks_wpcs)
print(results$ks_new) print("--------
BlandAltman Analysis --------")
print(results$bland_altman_results)

# Print Author Details cat("NAME:
Shirehya KP\n") cat("REGNO:
22BDS0365\n")
```

```
> print("-------- ANALYSIS RESULTS --------")
[1] "-------- ANALYSIS RESULTS --------"
> print("Dataset 1: WPCS_Score")
[1] "Dataset 1: WPCS_Score"
> print(results$dataset1_summary)
     number       Questions           Answers            Texts              Score
 Min.   : 1.100   Length:2442      Length:2442       Length:2442       Min.   :0.000
 1st Qu.: 3.700   Class :character Class :character  Class :character  1st Qu.:3.500
 Median : 7.400   Mode  :character Mode  :character  Mode  :character  Median :4.625
 Mean   : 7.202                                                        Mean   :4.179
 3rd Qu.:10.600                                                        3rd Qu.:5.000
 Max.   :12.900                                                        Max.   :5.000
 Answer_Keywords    Text_Keywords     New_Answer_Keywords Combined_Answer_Keywords
 Length:2442        Length:2442       Length:2442         Length:2442
 Class :character   Class :character  Class :character    Class :character
 Mode  :character   Mode  :character  Mode  :character    Mode  :character


    WPCS_Score
 Min.   :0.00
 1st Qu.:3.50
 Median :4.75
 Mean   :4.18
 3rd Qu.:5.00
 Max.   :5.00
>
> print("Dataset 2: New_Score")
[1] "Dataset 2: New_Score"
> print(results$dataset2_summary)
     number       Questions           Answers            Texts              Score
 Min.   : 1.100   Length:2442      Length:2442       Length:2442       Min.   :0.000
 1st Qu.: 3.700   Class :character Class :character  Class :character  1st Qu.:3.500
 Median : 7.400   Mode  :character Mode  :character  Mode  :character  Median :4.625
 Mean   : 7.202                                                        Mean   :4.179
 3rd Qu.:10.600                                                        3rd Qu.:5.000
 Max.   :12.900                                                        Max.   :5.000
 Answer_Keywords    Text_Keywords     New_Answer_Keywords Combined_Answer_Keywords
 Length:2442        Length:2442       Length:2442         Length:2442
 Class :character   Class :character  Class :character    Class :character
 Mode  :character   Mode  :character  Mode  :character    Mode  :character


 Cosine_Similarity Euclidean_Distance Manhattan_Distance Pearson_Correlation Norm_Euclidean
 Min.   :0.0000    Min.   :0.000      Min.   : 0.00      Min.   :-0.005133   Min.   :0.1089
 1st Qu.:0.1021    1st Qu.:2.449      1st Qu.: 6.00      1st Qu.: 0.098913   1st Qu.:0.2000
 Median :0.2582    Median :3.317      Median :11.00      Median : 0.255671   Median :0.2317
 Mean   :0.3054    Mean   :3.129      Mean   :11.34      Mean   : 0.303540   Mean   :0.2890
 3rd Qu.:0.4364    3rd Qu.:4.000      3rd Qu.:16.00      3rd Qu.: 0.434694   3rd Qu.:0.2899
 Max.   :1.0000    Max.   :8.185      Max.   :67.00      Max.   : 1.000000   Max.   :1.0000
 Norm_Manhattan    Adjusted_Pearson  Combined_Similarity   New_Score
 Min.   :0.01471   Min.   :0.4974    Min.   :0.08203     Min.   :0.000
 1st Qu.:0.05882   1st Qu.:0.5495    1st Qu.:0.16400     1st Qu.:4.000
 Median :0.08333   Median :0.6278    Median :0.24958     Median :4.500
 Mean   :0.16002   Mean   :0.6518    Mean   :0.30765     Mean   :4.136
 3rd Qu.:0.14286   3rd Qu.:0.7173    3rd Qu.:0.36255     3rd Qu.:5.000
 Max.   :1.00000   Max.   :1.0000    Max.   :1.00000     Max.   :5.000
>
> print("-------- Descriptive Statistics --------")
[1] "-------- Descriptive Statistics --------"
> print(results$all_stats)
     Dataset Mean_Original Mean_Derived Median_Original Median_Derived SD_Original SD_Derived
1 WPCS_Score      4.17931     4.180385           4.625           4.75    1.113772   1.113176
2  New_Score      4.17931     4.136364           4.625           4.50    1.113772   1.120658
  Min_Original Min_Derived Max_Original Max_Derived Skewness_Original Skewness_Derived
1            0           0            5           5         -1.420269        -1.422468
2            0           0            5           5         -1.420269        -1.373131
  Kurtosis_Original Kurtosis_Derived
1          1.454908         1.463451
2          1.454908         1.340636
>
> print("-------- Error Metrics --------")
[1] "-------- Error Metrics --------"
> print(results$all_metrics)
     Dataset         MSE       RMSE         MAE      MAPE R_squared
1 WPCS_Score 0.0003263206 0.01806435 0.001074939 0.03501105 0.9997379
2  New_Score 0.0694295147 0.26349481 0.139281327 4.38962895 0.9466005
>
> print("-------- Correlation Analysis --------")
[1] "-------- Correlation Analysis --------"
> print("WPCS_Score Correlation with Score:")
[1] "WPCS_Score Correlation with Score:"
> print(results$wpcs_correlation)

        Pearson's product-moment correlation

data:  wpcs_data$Score and wpcs_data$WPCS_Score
t = 3050.9, df = 2440, p-value < 2.2e-16
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.9998581 0.9998790
sample estimates:
      cor
0.999869


>
> print("New_Score Correlation with Score:")
[1] "New_Score Correlation with Score:"
> print(results$new_correlation)

        Pearson's product-moment correlation

data:  mutated_data$Score and mutated_data$New_Score
t = 207.97, df = 2440, p-value < 2.2e-16
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.9707308 0.9749734
sample estimates:
      cor
0.972934


>
> print("-------- Paired t-tests --------")
[1] "-------- Paired t-tests --------"
> print("WPCS_Score vs Score (Paired t-test):")
[1] "WPCS_Score vs Score (Paired t-test):"
> print(results$wpcs_ttest)

        Paired t-test

data:  wpcs_data$Score and wpcs_data$WPCS_Score
t = -2.9452, df = 2441, p-value = 0.003258
alternative hypothesis: true mean difference is not equal to 0
95 percent confidence interval:
 -0.0017906395 -0.0003592376
sample estimates:
mean difference
  -0.001074939
```

```
> print("New_Score vs Score (Paired t-test):")
[1] "New_Score vs Score (Paired t-test):"
> print(results$new_ttest)

        Paired t-test

data:  mutated_data$Score and mutated_data$New_Score
t = 8.1618, df = 2441, p-value = 5.228e-16
alternative hypothesis: true mean difference is not equal to 0
95 percent confidence interval:
 0.03262813 0.05326458
sample estimates:
mean difference
     0.04294636


>
> print("-------- Wilcoxon Signed Rank Test --------")
[1] "-------- Wilcoxon Signed Rank Test --------"
> print("WPCS_Score vs Score (wilcoxon test):")
[1] "WPCS_Score vs Score (wilcoxon test):"
> print(results$wpcs_wilcox)

        Wilcoxon signed rank test with continuity correction

data:  wpcs_data$Score and wpcs_data$WPCS_Score
V = 0, p-value = 0.006927
alternative hypothesis: true location shift is not equal to 0
> print("New_Score vs Score (wilcoxon test):")
[1] "New_Score vs Score (wilcoxon test):"
> print(results$new_wilcox)

        Wilcoxon signed rank test with continuity correction

data:  mutated_data$Score and mutated_data$New_Score
V = 153962, p-value = 6.685e-16
alternative hypothesis: true location shift is not equal to 0


>
> print("-------- Linear Regression Analysis --------")
[1] "-------- Linear Regression Analysis --------"
> print("WPCS_Score Regression Model:")
[1] "WPCS_Score Regression Model:"
> print(results$wpcs_lm_summary)

call:
lm(formula = WPCS_Score ~ Score, data = wpcs_data)

Residuals:
     Min       1Q   Median       3Q      Max
-0.00386 -0.00153 -0.00053 -0.00053  0.37389

Coefficients:
             Estimate Std. Error  t value Pr(>|t|)
(Intercept) 0.0038596  0.0014167    2.724  0.00649 **
Score       0.9993337  0.0003276 3050.905  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.01802 on 2440 degrees of freedom
Multiple R-squared:  0.9997,    Adjusted R-squared:  0.9997
F-statistic: 9.308e+06 on 1 and 2440 DF,  p-value: < 2.2e-16

> print(results$wpcs_lm_confint)
                  2.5 %       97.5 %
(Intercept) 0.001081526 0.006637662
Score       0.998691394 0.999976016
>
> print("New_Score Regression Model:")
[1] "New_Score Regression Model:"
> print(results$new_lm_summary)

call:
lm(formula = New_Score ~ Score, data = mutated_data)

Residuals:
     Min       1Q   Median       3Q      Max
-0.53451  0.01812  0.06022  0.06022  0.52865

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 0.045030   0.020359   2.212   0.0271 *
Score       0.978949   0.004707 207.974   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.259 on 2440 degrees of freedom
Multiple R-squared:  0.9466,    Adjusted R-squared:  0.9466
F-statistic: 4.325e+04 on 1 and 2440 DF,  p-value: < 2.2e-16

> print(results$new_lm_confint)
                  2.5 %      97.5 %
(Intercept) 0.005108478 0.08495239
Score       0.969719180 0.98817971
>
> print("-------- Distribution Similarity Tests --------")
[1] "-------- Distribution Similarity Tests --------"
> print("Kolmogorov-Smirnov Test - WPCS_Score vs Score:")
[1] "Kolmogorov-Smirnov Test - WPCS_Score vs Score:"
> print(results$ks_wpcs)

        Asymptotic two-sample Kolmogorov-Smirnov test

data:  wpcs_data$Score and wpcs_data$WPCS_Score
D = 0.000819, p-value = 1
alternative hypothesis: two-sided


>
> print("Kolmogorov-Smirnov Test - New_Score vs Score:")
[1] "Kolmogorov-Smirnov Test - New_Score vs Score:"
> print(results$ks_new)

        Asymptotic two-sample Kolmogorov-Smirnov test

data:  mutated_data$Score and mutated_data$New_Score
D = 0.12899, p-value < 2.2e-16
alternative hypothesis: two-sided


>
> print("-------- Bland-Altman Analysis --------")
[1] "-------- Bland-Altman Analysis --------"
> print(results$bland_altman_results)
     Dataset Mean_Difference SD_Difference Lower_Limit Upper_Limit Percentage_Within_Limits
1 WPCS_Score    -0.001074939    0.01803603 -0.03642556  0.03427568                 99.63145
2  New_Score     0.042946355    0.26002464 -0.46670194  0.55259465                 90.45864
>
> print("-------- COMPREHENSIVE SUMMARY --------")
[1] "-------- COMPREHENSIVE SUMMARY --------"
> print("1. which derived score is closer to the original score?")
[1] "1. which derived score is closer to the original score?"
> print(results$comprehensive_summary$rmse_comparison)
[1] "WPCS_Score is closer to the original Score (lower RMSE)"
>
> print("2. Which derived score has stronger correlation with the original score?")
```

```
[1] "WPCS_Score: RMSE = 0.0180643471739401 MAE = 0.00107493857493857"
> print(results$comprehensive_summary$new_metrics)
[1] "New_Score: RMSE = 0.263494809706026 MAE = 0.139281326781327"
>
> if(!is.na(results$all_metrics$MAPE[1]) && !is.na(results$all_metrics$MAPE[2])) {
+    print(results$comprehensive_summary$wpcs_mape)
+    print(results$comprehensive_summary$new_mape)
+ } else {
+    print(results$comprehensive_summary$mape_status)
+ }
[1] "WPCS_Score MAPE = 0.0350110481925859 %"
[1] "New_Score MAPE = 4.38962895119759 %"
>
> print("4. Statistical significance of differences:")
[1] "4. Statistical significance of differences:"
> print(results$comprehensive_summary$wpcs_ttest_pvalue)
[1] "WPCS_Score p-value (t-test): 0.00325792178333613"
> print(results$comprehensive_summary$new_ttest_pvalue)
[1] "New_Score p-value (t-test): 5.22825061124778e-16"
> print(results$comprehensive_summary$wpcs_significance)
[1] "WPCS_Score is significantly different from the original Score"
> print(results$comprehensive_summary$new_significance)
[1] "New_Score is significantly different from the original Score"
>
> print("5. Regression model quality:")
[1] "5. Regression model quality:"
> print(results$comprehensive_summary$wpcs_r2)
[1] "WPCS_Score R²: 0.999737929285073 Adjusted R²: 0.999737821879042"
> print(results$comprehensive_summary$new_r2)
[1] "New_Score R²: 0.946600481955422 Adjusted R²: 0.946578596907043"
>
> print("6. Bland-Altman agreement:")
[1] "6. Bland-Altman agreement:"
> print(results$comprehensive_summary$wpcs_ba)
[1] "WPCS_Score mean difference: -0.00107493857493857 95% limits of agreement: -0.0364255561540002 to 0.034275679004123"
> print(results$comprehensive_summary$new_ba)
[1] "New_Score mean difference: 0.0429463554463554 95% limits of agreement: -0.466701936553918 to 0.552594647446628"
>
> print("7. Distribution similarity (KS test):")
[1] "7. Distribution similarity (KS test):"
> print(results$comprehensive_summary$wpcs_ks)
[1] "WPCS_Score KS test p-value: 1"
> print(results$comprehensive_summary$new_ks)
[1] "New_Score KS test p-value: 0"
>
> print("8. FINAL CONCLUSION:")
[1] "8. FINAL CONCLUSION:"
> print(results$comprehensive_summary$final_conclusion)
[1] "WPCS_Score performed better overall in matching the original Score values."
> print(results$comprehensive_summary$wpcs_criteria)
[1] "WPCS_Score won on 7 out of 7 criteria."
> print(results$comprehensive_summary$new_criteria)
[1] "New_Score won on 0 out of 7 criteria."
> cat("NAME: M THIRUNARAYANAN\n")
NAME: M THIRUNARAYANAN
> cat("REGNO: 22BDS0342\n")
REGNO: 22BDS0342
```

Result:

Key Findings

1. **Overall Performance**: WPCS_Score consistently outperformed New_Score in matching the original Score values, winning on all 7 evaluation criteria.

2. **Accuracy Metrics**:

   o WPCS_Score: RMSE = 0.018, MAE = 0.001, MAPE = 0.035% o New_Score: RMSE = 0.263, MAE = 0.139, MAPE = 4.390%

3. **Correlation with Original Score**:

   o WPCS_Score: R^2 = 0.9997 (extremely strong correlation) o

   New_Score: R^2 = 0.9466 (strong but lower correlation)

4. **Statistical Significance**:

   o Both derived scores showed statistically significant differences from the original Score ($p < 0.05$) o   However, WPCS_Score's difference was much smaller (mean difference = -0.001) o New_Score had a larger deviation (mean difference = 0.043)

5. **Distribution Similarity**:

   o WPCS_Score: KS test p-value = 1 (distributions are identical) o New_Score: KS test p-value ≈ 0 (distributions are significantly different)

6. **Bland-Altman Agreement**:

   o WPCS_Score: 99.63% of values within limits of agreement o

   New_Score: 90.46% of values within limits of agreement

Descriptive Statistics Comparison

Both derived scores maintained similar central tendencies to the original Score:

- Original Score: Mean = 4.179, Median = 4.625

- WPCS_Score: Mean = 4.180, Median = 4.750

- New_Score: Mean = 4.136, Median = 4.500

Conclusion

WPCS_Score demonstrated superior performance in approximating the original Score across all evaluation metrics. While both methods produced scores with high correlation to the original, WPCS_Score showed near-perfect agreement with minimal error, making it the clearly preferred method for this application.