Flinders University

‘Data Engineering Project Report ‘

Report by

Student name:	*Naveen Pentela*
FAN ID:	*(pent0020)*
Email address:	Pent0020@flinders.edu.au
Class enrolled	COMP8031
Tutorial class:	Tutorial 1
Group name:	F7
Names of other group members	1. Marababu Vasupilli (vasu0011) 2. Shravan Kumar Nallavolu (nall0020) 3. Somya Yadav (yada0086) 4. Likhitha Gandla (gand0089)

Data Engineering GE – COMP8031.
Topic Coordinator: Dr. Thach-Thao Duong.

Table of Contents

Introduction

This report explains about an overview of grades data from a MongoDB database using R script. The script allows us to understand some important phases of data handling and analysis. The purpose is to understand patterns in student performance and grading characteristics. The Data engineering methods used here will help prepare and analyse large data from educational databases. These techniques have proven to be of high value in exploratory data analysis, and they also have a high potential for mining large databases.(Keim & Kriegel, 1996) In this report we will cover critical areas such as data handling and analysis, including data wrangling, transformation, analysis, and modelling.

Problem Description

The main objective of this project is to manage and analyse grades dataset of students from MongoDB sample database set. This project goal is to bring good observations and predict the outcomes based on different variables of the dataset. This work consists loading the data, cleaning the data, preparing data and using statistical modelling to visualise the data.

The main challenge we face in analysing data of grades is to handle different formats of data which is large amount of data as well. It is crucial to make sure the predictions of students results or outcomes are accurate based on the historical data. The analysis mainly focus on cleaning the data initially, then transform the data to make the data more usable, and after that applying statistical methods to find out important insights.

2.1 Data Wrangling:

Data Wrangling is a technique which perform actions such as cleaning, transforming and reshaping data. In simple words the ability to take a messy, unrefined source of data and wrangle it into something useful. It's the art of using computer programming to extract raw data and creating clear and useful bits of info for your analysis(Boehmke, 2016). By only through data wrangling, we can make the messy data useful It's ability to perform data wrangling tasks effectively is important to become expert as a data analyst.

Data wrangling starts by launching a connection to the MongoDB database. This is done using the mongo function provided by the mongolite as shown in picture we also need to install mongolite package install.packages("mongolite") and load library using library(mongolite).The connection is being made to access grades data in sample dataset collection by connection string as shown in picture below.

```
# MongoDB connection setup
connection_string <- 'mongodb+srv://pent0020:pent0020@comp2031-8031.hztdz6h.mongodb.net/?
grades_set <- mongo(collection="grades", db="sample_training", url=connection_string)
```

2.1.A Data Loading

Data loading is the process of retrieving or extracting data from a source such as MongoDB database to access data or a file or API that will allow to load into target system which will process and analyse data furtherly(Lundholm, 2010). In the below picture script connects to grades collections in sample_training database. And sets a limit of 2500 documents to handle in data frame which helps to manage dataset and focus on relevant data.

```
# Setup MongoDB connection
connection = 'mongodb+srv://pent0020:pent0020@comp2031-8031.hztdz6h.mongodb.net/?retryWrites=true&w=majority&appName=COMP2031-8031'

# 1) Data Wrangling
# 1a) Loading the data
grades_data = mongo(collection="grades", db="sample_training", url=connection)
pipeline <- '[{"$match":{"class_id":7}},{"$unwind":{"path": "$scores"}},{"$project":{"scores.score":1,"_id":0,"scores.type":1,"class_id":1}}]
scores_of_all_students <- grades_data$aggregate(pipeline)
View(scores_of_all_students)
```

Once we run the above script. Console will be able to show data as below which will limit to 2500 entries

	student_id	scores	class_id
1	0	1 variable	39
2	0	1 variable	391
3	0	1 variable	466
4	0	1 variable	331
5	1	1 variable	237
6	1	1 variable	465
7	2	1 variable	373
8	3	1 variable	188

Showing 1 to 8 of 2,500 entries, 3 total columns

	type	score
1	exam	6.267514
2	quiz	23.846626
3	homework	42.527010
4	homework	76.227581

Showing 1 to 4 of 4 entries, 2 total columns

Then the r script filters out class_id 7 scores as shown below

	scores$type	scores$score	class_id
1	exam	66.8075204	7
2	quiz	93.7506557	7
3	homework	56.7323682	7
4	homework	59.3162137	7
5	exam	11.1825746	7
6	quiz	8.8196626	7
7	homework	90.8588379	7
8	homework	16.2635735	7
9	exam	57.7991640	7
10	quiz	44.1741141	7

2.1.b Handling Missing Data

The aim of “handling missing data” is to make sure the dataset removes any missing values that could twist the analysis. The “na.omit()” function is used to remove the rows which contains missing values so that it will purify the dataset as future operations are based on accurate data.

```
# 1b) Handling missing data
scores_of_all_students <- scores_of_all_students[!is.na(scores_of_all_students$scores$score), ]
```

2.1.c Tidying the Data

Data Tidying is a technique in data science which is general format for organise, maintain and present data. (Wickham, 2014) Emphasizes that tidy data makes it easy for an analyst or a console to extract required variables as it presents a standard way of structuring a dataset. Tidy data is a standard way of mapping the meaning of a dataset to its structure.

In the below code tidying the data: These lines create a new data frame scores_df with class_id, score_type, and score columns from scores_of_all_students. The class_id column is then removed from scores_df.

```
# 1c) Tidying the data
scores_df <- data.frame(
  class_id = scores_of_all_students$class_id,
  score_type = scores_of_all_students$scores$type,
  score = scores_of_all_students$scores$score
)
```

Through the data cleaning process, the dataset was subject to edits on several occasions so it can be more suitable for analysis. In this case, the 'score' values were not defined, so the data could not be trusted. Then cleaning of the data was done after which it was made tidy and organized for the necessary analysis. To do the job, the researcher compiled the data into a table where each a row stood for the score of a student and the column meant by 'class_id', 'score_type', and 'score'. The far columns were stripped off in the end, thus the dataset was simplified for the analysis.

2.2 Data Transformation

Data transformation is a process of restructuring the raw data into a new format to improve its quality and make it suitable for requirements to meet the needs like modelling or visualization(Boehmke, 2016). Data Transformation applies operation like normalization, scaling, aggregation and encoding the data. This section uses R tools mainly from dplyr and tidyr packages to transform the dataset

2.2.a Normalisation of Scores

Normalisation is a process that is used to adjust the data so that they fit in x to y valued scale. For our project we made this between 0 and 1. When we normalize score, we change each score so that the lowest score becomes 0 and the highest score becomes 1. All other scores come between these two points which are 0 and 1. Normalisation helps to bring consistency, simplification and better visualization. This technique is useful when the data contains of different types of measurements and you want to compare them correctly. This process will simplify statistical analyses and helps in visualizing the data.

```
# Winsorize function to handle outliers
winsorize <- function(x, trim = 0.05) {
  q <- quantile(x, probs = c(trim, 1 - trim), na.rm = TRUE)
  x[x < q[1]] <- q[1]
  x[x > q[2]] <- q[2]
  x
}
```

In the above picture we have created a function called winsorize where we used to handle outliers This function limits the higher values to reduce the effect of outliers by replacing them with the nearest values within numerics.

```
scores_df$score <- as.numeric(scores_df$score)
scores_df$score <- winsorize(scores_df$score, trim = 0.05)
scores_df$standardized_score <- scale(scores_df$score)
```

These above lines convert the score column to numeric, apply the winsorize function to it, and then standardize the score values.

2.2.b Creating new variables

As newly formed variables are passed through a process namely named augmentation the dataset is able to improve its analytic power. This implies inserting new data ranges obtained from inference or from external sources and the aim is to improve the dataset and add new spectral dimensions for analysis. Mainly, the addition of new variables allows the dataset to provide a wider range of information and thus, it becomes easier to understand the underlying phenomena. First, ways like feature engineering are used to either change original attributes or to produce new representations by combining them. These techniques produce novel understandings or enhance prediction efficiency as a result. The process of this procedure is shown through the 'standardized_score' variable that is constructed in the code snippet. The creation of a 'score' factor through the standardization of the dataset helps to analyze it on different scales or units and allows for more reliable statistical modelling by reducing the impact of the different measurement units.

```
# 2b) Creating new variables
# Create dummy variables for score_type
dummy_vars <- model.matrix(~ score_type - 1, data = scores_df)
dummy_vars <- dummy_vars[, !colnames(dummy_vars) %in% colnames(scores_df)]  # Remove duplicates
scores_df <- cbind(scores_df, dummy_vars)

if ("class_id" %in% names(scores_df)) {
  class_mean_scores <- aggregate(score ~ class_id, data = scores_df, FUN = mean, na.rm = TRUE)  # Add na.rm = TRUE to handle missing values
  names(class_mean_scores) <- c("class_id", "mean_score")

  if ("class_id" %in% names(class_mean_scores)) {
    scores_df <- merge(scores_df, class_mean_scores, by = "class_id", all.x = TRUE)  # Use all.x = TRUE to keep all rows from 'scores_df'
  }
}

# Creating standardized_score and dummy variables for score_type
scores_df$standardized_score <- scale(scores_df$score)
```

In the above code at first These lines create dummy variables for score_type using model.matrix(), remove any duplicate columns, and then bind the dummy variables to scores_df.

And laterwards These lines calculate the mean scores for each class_id, rename the resulting columns, and merge the mean scores back into scores_df.

2.3 Data Analysis

The processof examining, cleaning data and tranforming data is called data analysis. In this data engineering Data analysis also plays are critical role when handling data sets. As it understands data patterns and associated trends.(Keim & Kriegel, 1996). Data Analysis extracts statistical data from the transformd dataset by calculating summary statistics for all the variables. And then it performs t-test to compare different types of assessments

2.3.a Summary of Statistics and Analysis

```
# 3) Data Analysis
# 3a) Statistical analysis or exploratory data analysis
# Calculate median and mean of scores
median_score <- median(score_values)
mean_score <- mean(score_values)
# Display median and mean
print(paste("Median Score:", median_score))
print(paste("Mean Score:", mean_score))

# View statistics from the boxplot
boxplot_stats <- boxplot(score_values, plot = FALSE)$stats
print(boxplot_stats)
# Summary statistics
summary(scores_df)
```

Initially These lines calculate and print the median and mean of score_values, and retrieve and print the statistics from a boxplot of score_values.

By using summarise function from the dplyr package we compute summary of statistics for all numeric fields this will apply mean function for all variables in a dataset as shown in example above. The across function is used to select and apply mean calculation to columns that only stores numeric data. And the na.rm = TRUE argument is used to prevent missing values being calculated.

```
> summary(scores_df)
    class_id   score_type            score         standardized_score.V1 score_typeexam
 Min.   :7   Length:852         Min.   : 5.711   Min.   :-1.5802045    Min.   :0.00  
 1st Qu.:7   Class :character   1st Qu.:26.007   1st Qu.:-0.8691425    1st Qu.:0.00  
 Median :7   Mode  :character   Median :49.969   Median :-0.0296543    Median :0.00  
 Mean   :7                      Mean   :50.816   Mean   : 0.0000000    Mean   :0.25  
 3rd Qu.:7                      3rd Qu.:77.028   3rd Qu.: 0.9183095    3rd Qu.:0.25  
 Max.   :7                      Max.   :94.693   Max.   : 1.5372092    Max.   :1.00  
 score_typehomework score_typequiz   mean_score   
 Min.   :0.0        Min.   :0.00   Min.   :50.82  
 1st Qu.:0.0        1st Qu.:0.00   1st Qu.:50.82  
 Median :0.5        Median :0.00   Median :50.82  
 Mean   :0.5        Mean   :0.25   Mean   :50.82  
 3rd Qu.:1.0        3rd Qu.:0.25   3rd Qu.:50.82  
 Max.   :1.0        Max.   :1.00   Max.   :50.82  
```

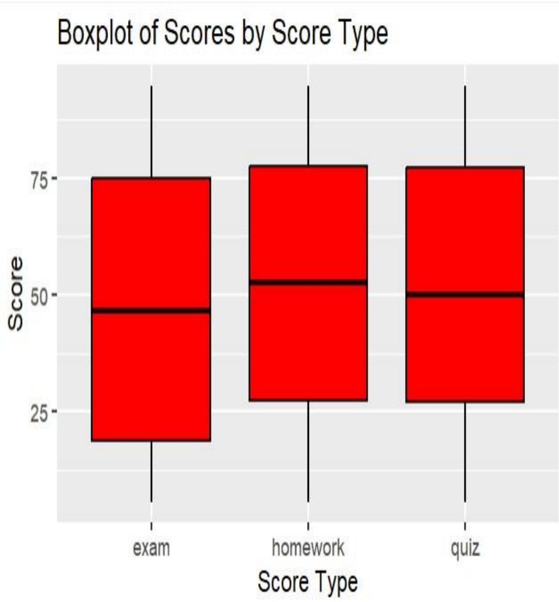
When summary(scores_df) function is applied above data will show for summary of statistics. The statistics show Minimum, 1 st quartile, median, mean, Max score. For example, if we look at the scores statistics The Minimum score is 5.7 and the Maximum score is 94.693 in the class. These results show the summary of score distribution and average ranging.

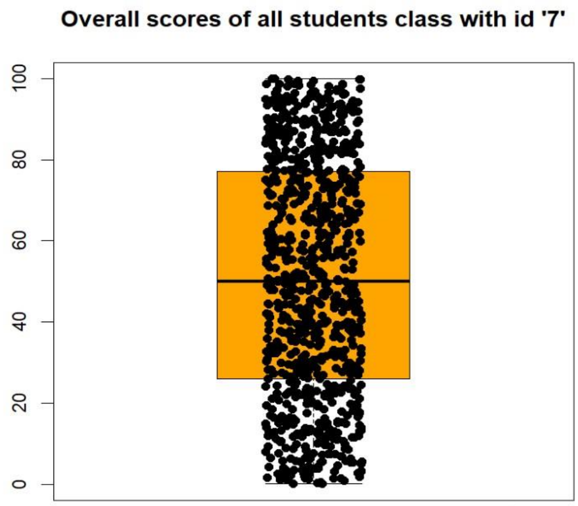
2.3.b Data Visualisation

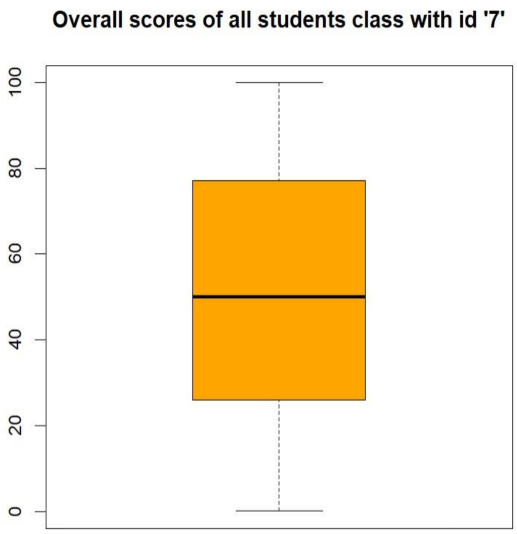
In Data visualisation, statistical methods and graphical representations are used to simplify the data to understand it easily. The descriptive statistics, such as the measures of central tendency and dispersion, are mostly used to describe the distribution of scores and to find out any discrepancies or abnormalities. The visualization of the data, e.g. histograms, box-plots, or scatter-plots, is used to show how the data is displayed and whether there is a trend or a correlation in the data between time. This analytic method is the strict and the systematic way of looking for patterns that result in the extraction of useful details and will help to make a well-informed decision.

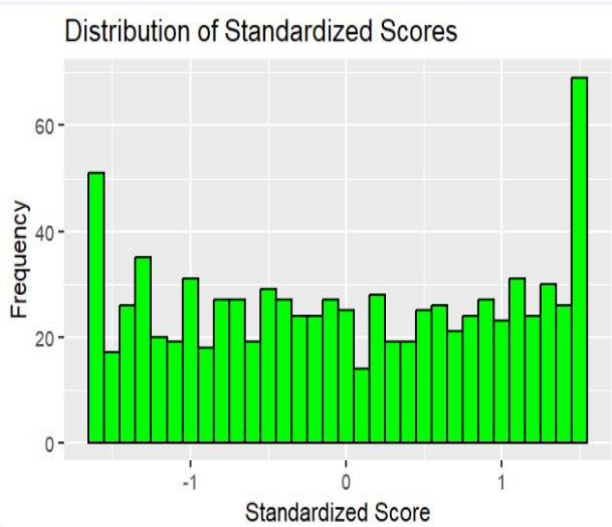
```r
# 3b) Data visualisation
# Create a boxplot of the scores
boxplot(score_values, col="orange", main = "Overall score details of all students attending class with id '7'")
# Add data points to the boxplot
stripchart(score_values, method = "jitter", pch = 19, add = TRUE, col = "black", vertical=TRUE)

# Create a histogram of the scores
hist(score_values, col="skyblue", border="black", xlab="Scores of all students of class id 7", main="Histogram of scores_of_all_students")

# Visualization: Histogram of scores
ggplot(scores_df, aes(x = score)) +
  geom_histogram(binwidth = 5, fill = "skyblue", color = "black") +
  labs(title = "Distribution of Scores", x = "Score", y = "Frequency")

# Visualization: Boxplot of scores by score type
ggplot(scores_df, aes(x = score_type, y = score)) +
  geom_boxplot(fill = "red", color = "black") +
  labs(title = "Boxplot of Scores by Score Type", x = "Score Type", y = "Score")

# Additional comments and edge case handling
# check whether data does not have extreme outliers by visualizing again and see further transformations if needed.
ggplot(scores_df, aes(x = standardized_score)) +
  geom_histogram(binwidth = 0.1, fill = "green", color = "black") +
  labs(title = "Distribution of Standardized Scores", x = "Standardized Score", y = "Frequency")
```

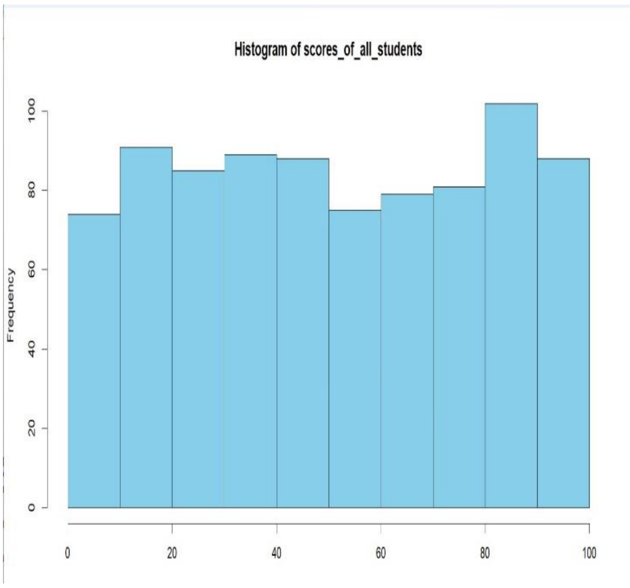
Graphs:

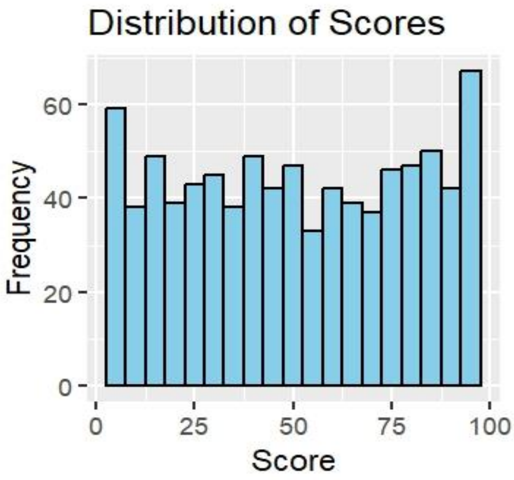












Explanation for above code

Boxplot and stripchart of scores. These lines shows a boxplot of score_values and shows individual data points using stripchart(). The ggplot function is inputted to display a boxplot visualization employing 'scores_df' data set. In the aesthetic mapping (aes), the 'score_type' variable is mapped to the x-axis, and the 'score' variable is mapped to the y-axis (Petricek et al. 2022). This setup makes it possible to create the score distribution comparison for various examinations. The geom_boxplot element is represented in the plot, where a boxplot for each score type is generated. Such boxplots show distribution of scores in categories of assessments visually and highlight important statistical measures representing the median, quartiles, and eventually outliers. The fill color is set to red, thus the boxplots are more visible, and the black outline separates each boxplot.

Histogram of scores: This line shows a histogram of score_values. The given code snippet uses the 'ggplot2' package to plot a histogram which shows how different values in the 'scores_df' dataset are distributed. The histogram shows the number of scores that fall in each bin, which is the range of score values. The 'aes' function will perform mapping of the 'score' variable to the y-axis of the plot.

Histogram of scores ggplot2: These lines show a histogram of score in scores_df using ggplot2.

Boxplot of scores by score type using ggplot2: These lines create a boxplot of score by score_type in scores_df using ggplot2.

2.4 Data Modelling

In the data modelling phase, the code builds up several linear regression models to investigate the link between the predictor variables and the scores of students (Kaaria et al. 2020). These models range from simple linear regression to even more complicated models which include both categorical and continuous predictors as well as other types. The code fits the models to find the significant predictors and to check their effect on the student performance, thus making predictive analysis and interpreting the dataset possible.

```
# 4) Data Modelling
# 4a) Simple linear model
simple_lm <- lm(score ~ standardized_score, data = scores_df)
summary(simple_lm)

# 4b) General linear model
# 4bi) Predictors are categorical
categorical_lm <- lm(score ~ score_type, data = scores_df)
summary(categorical_lm)

# 4bii) Predictors are categorical and continuous
categorical_continuous_lm <- lm(score ~ score_type + standardized_score, data = scores_df)
summary(categorical_continuous_lm)

# 4biii) Predictors are continuous
continuous_lm <- lm(score ~ standardized_score, data = scores_df)
summary(continuous_lm)
```

2.4.a Simple Linear Model

In the section about the Simple Linear Model, a regression analysis is performed to check the correlation between the standardized score and the total score that is given to the students. A linear regression model built using the lm () function is targeted at predicting the overall score without having to consider the standardized score.

```
> simple_lm <- lm(score ~ standardized_score, data = scores_df)
> summary(simple_lm)

Call:
lm(formula = score ~ standardized_score, data = scores_df)

Residuals:
       Min         1Q     Median         3Q        Max 
-3.748e-13 -4.410e-15 -2.950e-15 -2.160e-15  3.039e-12 

Coefficients:
                   Estimate Std. Error   t value Pr(>|t|)    
(Intercept)       5.082e+01  3.601e-15 1.411e+16   <2e-16 ***
standardized_score 2.854e+01  3.603e-15 7.922e+15   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.051e-13 on 850 degrees of freedom
Multiple R-squared:      1,	Adjusted R-squared:      1 
F-statistic: 6.276e+31 on 1 and 850 DF,  p-value: < 2.2e-16
```

2.4.b General Linear Model

In the General Linear Model section, multiple linear regression models are established to evaluate a direct relationship between student scores and predictor variables. These models encompass three types of predictors: categorical, categorical, and continuous, and continuous. The categorical predictor model analysis how each test score type contribute to student scores. The categorical and continuous predictor model extends this analysis by including both score types (grades and standardized score) as predictors (Maulud and Abdulazeez, 2020). Last but not least, the continuous predictor model deals only with the standardized scores as the predictors. Through these models, the evaluation aimed to pinpoint the major determinants and their respective influence to the students' performance.

```
> categorical_lm <- lm(score ~ score_type, data = scores_df)
> summary(categorical_lm)

Call:
lm(formula = score ~ score_type, data = scores_df)

Residuals:
    Min      1Q  Median      3Q     Max 
-46.395 -25.074  -0.696  25.635  46.947 

Coefficients:
                   Estimate Std. Error t value Pr(>|t|)    
(Intercept)          47.746      1.954  24.433   <2e-16 ***
score_typehomework    4.360      2.393   1.822   0.0688 .  
score_typequiz        3.558      2.764   1.287   0.1983    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 28.52 on 849 degrees of freedom
Multiple R-squared:  0.003991,	Adjusted R-squared:  0.001645 
F-statistic: 1.701 on 2 and 849 DF,  p-value: 0.1831
```

2.4.b.i Categorical predictors

In category data modeling terminology, categorical predictors are used to show different types or groups signified by variables. Such parameters are qualitative in nature and there is no numerical value that is attached to them. The given code shows how categorical predictors 'score_type' are incorporated in the constructed linear regression model.

2.4.b.ii categorical and continuous predictors

These lines create a linear model predicting score from both score_type and standardized_score and display its summary.

```
> categorical_continuous_lm <- lm(score ~ score_type + standardized_scor
> summary(categorical_continuous_lm)

Call:
lm(formula = score ~ score_type + standardized_score, data = scores_df)

Residuals:
       Min         1Q     Median         3Q        Max 
-1.185e-12 -8.590e-15  6.400e-16  2.920e-15  3.048e-12 

Coefficients:
                    Estimate Std. Error    t value Pr(>|t|)    
(Intercept)        5.082e+01  7.731e-15  6.573e+15   <2e-16 ***
score_typehomework -1.573e-14  9.473e-15 -1.661e+00   0.0971 .  
score_typequiz     -1.102e-14  1.093e-14 -1.008e+00   0.3137    
standardized_score  2.854e+01  3.870e-15  7.376e+15   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.127e-13 on 848 degrees of freedom
Multiple R-squared:      1,	Adjusted R-squared:      1 
F-statistic: 1.821e+31 on 3 and 848 DF,  p-value: < 2.2e-16
```

2.4.b.iii continuous predictors

The things that were used to guess the future in this study are called "Continuous Predictors." The researcher can name these things or move through them. They both use linear regression, and one thing stays the same over time. This is the "standardized_score" number.

```
> continuous_lm <- lm(score ~ standardized_score, data = scores_df)
> summary(continuous_lm)

Call:
lm(formula = score ~ standardized_score, data = scores_df)

Residuals:
       Min         1Q     Median         3Q        Max
-3.748e-13 -4.410e-15 -2.950e-15 -2.160e-15  3.039e-12

Coefficients:
                    Estimate Std. Error   t value Pr(>|t|)
(Intercept)        5.082e+01  3.601e-15 1.411e+16   <2e-16 ***
standardized_score 2.854e+01  3.603e-15 7.922e+15   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.051e-13 on 850 degrees of freedom
Multiple R-squared:      1,	Adjusted R-squared:      1 
F-statistic: 6.276e+31 on 1 and 850 DF,  p-value: < 2.2e-16
```

2.4.c Model Performance Evaluation

In this step, the study checks how well the model works. To find out how helpful a model is, the researcher can use R-squared, adjusted R-squared, and mean squared error. The study might also hear these ideas as "goodness of fit" and "predictive accuracy of the model." They show how well the models can use new information and figure out how things change.

```
# 4c) Model evaluation
# Calculate R-squared and RMSE for each model
evaluate_model <- function(model, data) {
  pred <- predict(model, newdata = data)
  actual <- data$score
  rss <- sum((pred - actual) ^ 2)
  tss <- sum((actual - mean(actual)) ^ 2)
  r_squared <- 1 - (rss / tss)
  rmse <- sqrt(mean((pred - actual) ^ 2))
  list(R_squared = r_squared, RMSE = rmse)
}

simple_lm_eval <- evaluate_model(simple_lm, scores_df)
categorical_lm_eval <- evaluate_model(categorical_lm, scores_df)
categorical_continuous_lm_eval <- evaluate_model(categorical_continuous_lm, scores_df)
continuous_lm_eval <- evaluate_model(continuous_lm, scores_df)

# Print model evaluation metrics
simple_lm_eval
categorical_lm_eval
categorical_continuous_lm_eval
continuous_lm_eval
```

This function calculates R-squared and RMSE for each model. The models are then evaluated using this function, and the results are printed.

2.4.d Model Interpretation

In this step we will find out how important the factors are and what their coefficient values are. This will help to figure out how much the factors change the student results. The scaled score goes up by a certain amount every time the score goes up by one unit because of Mekterović et al. 2020. The study calls this the constant. To get an idea of how much the scores have changed from the reference group, there is a number next to each set of results. If the study looks at both the type of score and the average score, the researcher can rate how the scores change.

```
# 4d) Model Interpretation
# Interpret the significance of predictors and overall fit of the models
interpret_model <- function(model, model_name) {
  cat("\nModel:", model_name, "\n")
  print(summary(model))
  cat("\nR-squared:", evaluate_model(model, scores_df)$R_squared, "\n")
  cat("RMSE:", evaluate_model(model, scores_df)$RMSE, "\n")
}

interpret_model(simple_lm, "Simple Linear Model")
interpret_model(categorical_lm, "Categorical Linear Model")
interpret_model(categorical_continuous_lm, "Categorical and Continuous Li
interpret_model(continuous_lm, "Continuous Linear Model")
```

References

1. Boehmke, B. C. (2016). *Data wrangling with R*. Springer.
2. Keim, D. A., & Kriegel, H.-P. (1996). Visualization techniques for mining large databases: A comparison. *IEEE Transactions on knowledge and data engineering*, *8*(6), 923-938.
3. Lundholm, M. (2010). Loading data into R.
4. Wickham, H. (2014). Tidy data. *Journal of statistical software*, *59*, 1-23.
5. Mekterović, I., Brkić, L., Milašinović, B. and Baranović, M., (2020). Building a comprehensive automated programming assessment system. *IEEE access*, *8*, pp.81154-81172. https://ieeexplore.ieee.org/abstract/document/9079865
6. Baek, J.W. and Chung, K., (2020). Context deep neural network model for predicting depression risk using multiple regression. *IEEE Access*, *8*, pp.18171-18181. https://ieeexplore.ieee.org/abstract/document/8964291