Algebra I Podsie Study

Isha Patel ishap

August 2 2024

Contents


```
algebraData <- read.table("algebra.txt", header = TRUE, sep = "\t", quote = "\"", fill = TRUE, comment.
PostTest <- read.table("post-test.txt", header = TRUE, sep = "\t", quote = "\"", fill = TRUE, comment.cl
library("knitr")
PreTest <- read.table("pre-test.txt", header = TRUE, sep = "\t", quote = "\"", fill = TRUE, comment.cha
ExpTest <- read.table("pre-test.txt", header = TRUE, sep = "\t", quote = "\"", fill = TRUE, comment.cha
library("kableExtra")
library("pander")
library("readr")
library("magrittr")
library("car")
library("interactions")
library("leaps")
library("dplyr")
library("writexl")
library("ggplot2")
library("tinytex")
library("car")
```

Introduction

In this study, we look at how spaced retrieval practice for Algebra I concepts affects learning compared to no practice. This study aims to determine if spaced retrieval practice of Algebra I in a computer-based system can improve learning. 10 learning objectives from class will be identified, and two questions will be created. To prevent students from memorizing the answer, the pre-test will consist of 20 questions, each with 3 variations. For each student, 40% of the learning objectives will be chosen at random and withheld from the Personal Deck. The questions for the remaining 60% of the learning objectives will be inserted to the Personal Deck for practice over the next 4 weeks. Every Tuesday and Thursday, each student will have 15 minutes to practice their personal deck, which will use the SuperMemo2 algorithm to determine whether a student should practice each specific question. At the end of four weeks, students will take a Post-Test with 20 questions covering the 10 learning objectives.

Exploratory Data Analysis

DATA: This data was conducted by Ashley, an Algebra I teacher who will be using Podsie with her 80 students over the next 5 weeks of school. The study will run from February 26, 2024 to March 21, 2024. There are 28 variables in this study which are called Row, Sample, StudentID, ProblemHierachy, ProblemName, ProblemView, StepName, StepStartTime, FirstTransactionTime, CorrectTransactionTime, StepEndTime, StepDuration, CorrectStepDuration, ErrorStepDuration, FirstAttempt, Incorrect, Hint, Correct, Condition, KCLO, OppurtunityLO, PredictedErrorRateLO, KCSingleKC, OppurtunitySingleKC, PredictedErrorRateSingleKC, KCUniqueStep, OppurtunityUniqueStep and PredictedErrorRateUniqueStep.

This is the structure of the data that will be used.

```
str(algebraData)
```

```
## 'data.frame':    6028 obs. of  28 variables:
##  $ Row                          : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Sample                       : chr  "All Data" "All Data" "All Data" "All Data" ...
##  $ StudentID                    : int  25940 25940 25940 25940 25940 25940 25940 25940 25940 25940 ..
##  $ ProblemHierarchy             : chr  "LO LO 1" "LO LO 1" "LO LO 2" "LO LO 3" ...
##  $ ProblemName                  : chr  "Simplify the expression.  \\left(9n-16\\right)+\\left(-1+3n\\
##  $ ProblemView                  : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ StepName                     : int  97183 98210 98211 98217 98219 98220 98221 98222 98238 98239 ..
##  $ StepStartTime                : chr  "2024-03-05 15:37:10" "2024-03-05 15:38:12" "2024-03-05 15:39:
##  $ FirstTransactionTime         : chr  "2024-03-05 15:38:12" "2024-03-05 15:39:10" "2024-03-05 15:39:
##  $ CorrectTransactionTime       : chr  "2024-03-05 15:38:12" "" "2024-03-05 15:39:33" "2024-03-05 15:
##  $ StepEndTime                  : chr  "2024-03-05 15:38:12" "2024-03-05 15:39:10" "2024-03-05 15:39:
##  $ StepDuration                 : chr  "62" "58" "23" "82" ...
##  $ CorrectStepDuration          : chr  "62" "." "23" "82" ...
##  $ ErrorStepDuration            : chr  "." "58" "." "." ...
##  $ FirstAttempt                 : chr  "correct" "incorrect" "correct" "correct" ...
##  $ Incorrect                    : int  0 1 0 0 0 0 0 1 0 0 ...
##  $ Hint                         : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Correct                      : int  1 0 1 1 1 1 1 0 2 1 ...
##  $ Condition                    : chr  "without (podsie_personal_deck)" "without (podsie_personal_dec
##  $ KCLO                         : chr  "LO 1" "LO 1" "LO 2" "LO 3" ...
##  $ OpportunityLO                : int  1 2 1 1 1 2 1 2 1 2 ...
##  $ PredictedErrorRateLO         : num  0.193 0.193 0.161 0.25 0.21 ...
##  $ KCSingleKC                   : chr  "Single-KC" "Single-KC" "Single-KC" "Single-KC" ...
##  $ OpportunitySingleKC          : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ PredictedErrorRateSingleKC   : num  0.347 0.345 0.343 0.34 0.338 ...
##  $ KCUniqueStep                 : chr  "KC9" "KC36" "KC65" "KC67" ...
##  $ OpportunityUniqueStep        : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ PredictedErrorRateUniqueStep: num  0.108 0.372 0 0.437 0.425 ...
```

Now we look at a summary of the data.

```
summary(algebraData)
```

```
##       Row          Sample            StudentID     ProblemHierarchy
##  Min.   :   1   Length:6028        Min.   :25940   Length:6028
##  1st Qu.:1508   Class :character   1st Qu.:25960   Class :character
##  Median :3014   Mode  :character   Median :26046   Mode  :character
##  Mean   :3014                      Mean   :26038
##  3rd Qu.:4521                      3rd Qu.:26083
##  Max.   :6028                      Max.   :26209
##
```

```
##  ProblemName         ProblemView       StepName      StepStartTime
##  Length:6028        Min.   :1.000   Min.   : 97183   Length:6028
##  Class :character   1st Qu.:1.000   1st Qu.: 98245   Class :character
##  Mode  :character   Median :1.000   Median : 99020   Mode  :character
##                     Mean   :1.194   Mean   :101261
##                     3rd Qu.:1.000   3rd Qu.:106887
##                     Max.   :7.000   Max.   :106941
##
##  FirstTransactionTime CorrectTransactionTime StepEndTime
##  Length:6028          Length:6028            Length:6028
##  Class :character     Class :character       Class :character
##  Mode  :character     Mode  :character       Mode  :character
##
##
##
##
##  StepDuration       CorrectStepDuration ErrorStepDuration  FirstAttempt
##  Length:6028        Length:6028         Length:6028        Length:6028
##  Class :character   Class :character    Class :character   Class :character
##  Mode  :character   Mode  :character    Mode  :character   Mode  :character
##
##
##
##
##    Incorrect           Hint      Correct        Condition
##  Min.   :0.0000   Min.   :0   Min.   :0.0000   Length:6028
##  1st Qu.:0.0000   1st Qu.:0   1st Qu.:1.0000   Class :character
##  Median :0.0000   Median :0   Median :1.0000   Mode  :character
##  Mean   :0.2381   Mean   :0   Mean   :0.8369
##  3rd Qu.:0.0000   3rd Qu.:0   3rd Qu.:1.0000
##  Max.   :4.0000   Max.   :0   Max.   :2.0000
##
##      KCLO           OpportunityLO   PredictedErrorRateLO  KCSingleKC
##  Length:6028        Min.   : 1.00   Min.   :0.0126       Length:6028
##  Class :character   1st Qu.: 2.00   1st Qu.:0.1083       Class :character
##  Mode  :character   Median : 4.00   Median :0.1888       Mode  :character
##                     Mean   : 5.23   Mean   :0.2113
##                     3rd Qu.: 7.00   3rd Qu.:0.2947
##                     Max.   :28.00   Max.   :0.6903
##
##  OpportunitySingleKC PredictedErrorRateSingleKC KCUniqueStep
##  Min.   :  1.00      Min.   :0.0297             Length:6028
##  1st Qu.: 19.00      1st Qu.:0.1325             Class :character
##  Median : 38.00      Median :0.2029             Mode  :character
##  Mean   : 39.26      Mean   :0.2113
##  3rd Qu.: 57.00      3rd Qu.:0.2761
##  Max.   :125.00      Max.   :0.5827
##
##  OpportunityUniqueStep PredictedErrorRateUniqueStep
##  Min.   :1.000         Min.   :0.00000
##  1st Qu.:1.000         1st Qu.:0.04965
##  Median :1.000         Median :0.15370
##  Mean   :1.179         Mean   :0.21049
##  3rd Qu.:1.000         3rd Qu.:0.31877
```

```
##  Max.   :6.000          Max.   :0.92780
##  NA's   :42             NA's   :42
```

It is important to also check for duplicates.

```
duplicate_count <- sum(duplicated(algebraData$StudentID))
print(paste("Number of duplicate StudentID entries:", duplicate_count))
```

```
## [1] "Number of duplicate StudentID entries: 5948"
```

There are 5948 duplicates in StudentID, this can assist in figuring out how many students completed both the Pre-Test and Post-Test.

```
table(PostTest$Sample)
```

```
##
## Post-Test
##      1600
```

```
table(PreTest$Sample)
```

```
##
## Pre-Test Data
##          1600
```

```
table(PostTest$StudentId)
```

```
##
## 25940 25941 25942 25943 25944 25945 25946 25947 25948 25949 25950 25951 25952
##    20    20    20    20    20    20    20    20    20    20    20    20    20
## 25953 25954 25955 25956 25957 25958 25959 25960 25961 25962 25963 26030 26031
##    20    20    20    20    20    20    20    20    20    20    20    20    20
## 26032 26033 26034 26035 26036 26037 26038 26039 26040 26041 26042 26043 26044
##    20    20    20    20    20    20    20    20    20    20    20    20    20
## 26045 26046 26047 26048 26049 26050 26051 26052 26053 26054 26071 26072 26073
##    20    20    20    20    20    20    20    20    20    20    20    20    20
## 26074 26075 26076 26077 26078 26079 26080 26081 26082 26083 26084 26085 26086
##    20    20    20    20    20    20    20    20    20    20    20    20    20
## 26087 26088 26089 26090 26091 26092 26093 26094 26095 26096 26179 26185 26186
##    20    20    20    20    20    20    20    20    20    20    20    20    20
## 26208 26209
##    20    20
```

```
table(PreTest$StudentId)
```

```
##
## 25940 25941 25942 25943 25944 25945 25946 25947 25948 25949 25950 25951 25952
##    20    20    20    20    20    20    20    20    20    20    20    20    20
## 25953 25954 25955 25956 25957 25958 25959 25960 25961 25962 25963 26030 26031
##    20    20    20    20    20    20    20    20    20    20    20    20    20
## 26032 26033 26034 26035 26036 26037 26038 26039 26040 26041 26042 26043 26044
##    20    20    20    20    20    20    20    20    20    20    20    20    20
## 26045 26046 26047 26048 26049 26050 26051 26052 26053 26054 26071 26072 26073
##    20    20    20    20    20    20    20    20    20    20    20    20    20
## 26074 26075 26076 26077 26078 26079 26080 26081 26082 26083 26084 26085 26086
##    20    20    20    20    20    20    20    20    20    20    20    20    20
## 26087 26088 26089 26090 26091 26092 26093 26094 26095 26096 26179 26185 26186
##    20    20    20    20    20    20    20    20    20    20    20    20    20
## 26208 26209
```

```
##    20    20
```

The tables above show that each student completed the Pre-Test and Post-Test, for a total of 1600 tests completed divided by 20 tests per Student ID, resulting in 80 students.

Now it is important to look for missing values in the dataset.

```
MissingValues <- colSums(is.na(algebraData))
print(MissingValues)
```

```
##                          Row                       Sample
##                            0                            0
##                    StudentID             ProblemHierarchy
##                            0                            0
##                  ProblemName                  ProblemView
##                            0                            0
##                     StepName                StepStartTime
##                            0                            0
##         FirstTransactionTime       CorrectTransactionTime
##                            0                            0
##                  StepEndTime                 StepDuration
##                            0                            0
##          CorrectStepDuration            ErrorStepDuration
##                            0                            0
##                FirstAttempt                    Incorrect
##                            0                            0
##                         Hint                      Correct
##                            0                            0
##                    Condition                         KCLO
##                            0                            0
##                OpportunityLO         PredictedErrorRateLO
##                            0                            0
##                   KCSingleKC          OpportunitySingleKC
##                            0                            0
##   PredictedErrorRateSingleKC                 KCUniqueStep
##                            0                            0
##        OpportunityUniqueStep PredictedErrorRateUniqueStep
##                           42                           42
```

There are 42 missing values in the OpportunityUniqueStep and PredictedErrorRateUniqueStep.

We must now the see spacing per value.

```
table(algebraData$ProblemHierarchy)
```

```
##
##  LO LO 1 LO LO 10  LO LO 2  LO LO 3  LO LO 4  LO LO 5  LO LO 6  LO LO 7
##       583       653       549       606       551       678       663       620
##  LO LO 8  LO LO 9
##       543       582
```

The following are the only unique LO values: LO1, LO2, LO3, LO4, LO5, LO6, LO7, LO8, LO9, and LO10. In a different software called Excel, I was able to manually go through each StudentId, count all the unique LOs in the experiment, and divide by 10, yielding a result of 60%, which confirms what the pre-registration stated about the question selection.

Univariate Exploration

To begin the analysis, we look at each variable individually. To investigate the distribution of quantitative

variables, we use histograms and scatterplots, while for categorical variables we use bar charts and tables. When looking at both varibles boxplots will be used. The response variable is FirstAttempt, which includes correct if they got the question right and incorrect if they got it wrong. The variable Incorrect and Correct are also used in this dataset.

```
table(algebraData$FirstAttempt)
```

```
##
##   correct incorrect
##      4754      1274
```

This table shows that 4754 out of 6058 students got the question correct which is 78.87%

It is now important to learn how many students used Podsie.

```
table(algebraData$Condition)
```

```
##
##    with (podsie_personal_deck) without (podsie_personal_deck)
##                           4747                           1281
```

The table above shows how many students used Podise and how many didn't. 4747 students used Podsie out of 6028 which is 78.75% and 1281 did not which is 21.25% That number is very close to the number of students who got the question correct on the first attempt.

Let's see how many students got the questions correct on the PreTest and PostTest.

```
table(PreTest$FirstAttempt)
```

```
##
##   correct incorrect
##      1148       452
```

In the PreTest 71.75% of the students got the questions correct.

```
table(PostTest$FirstAttempt)
```

```
##
##   correct incorrect
##      1482       118
```

In the PostTest 92.63% of the students got the questions correct which is an increase by 20.85% from PreTest.

```
pretest_contingency_table <- table(PreTest$FirstAttempt, PreTest$Condition)

print(pretest_contingency_table)
```

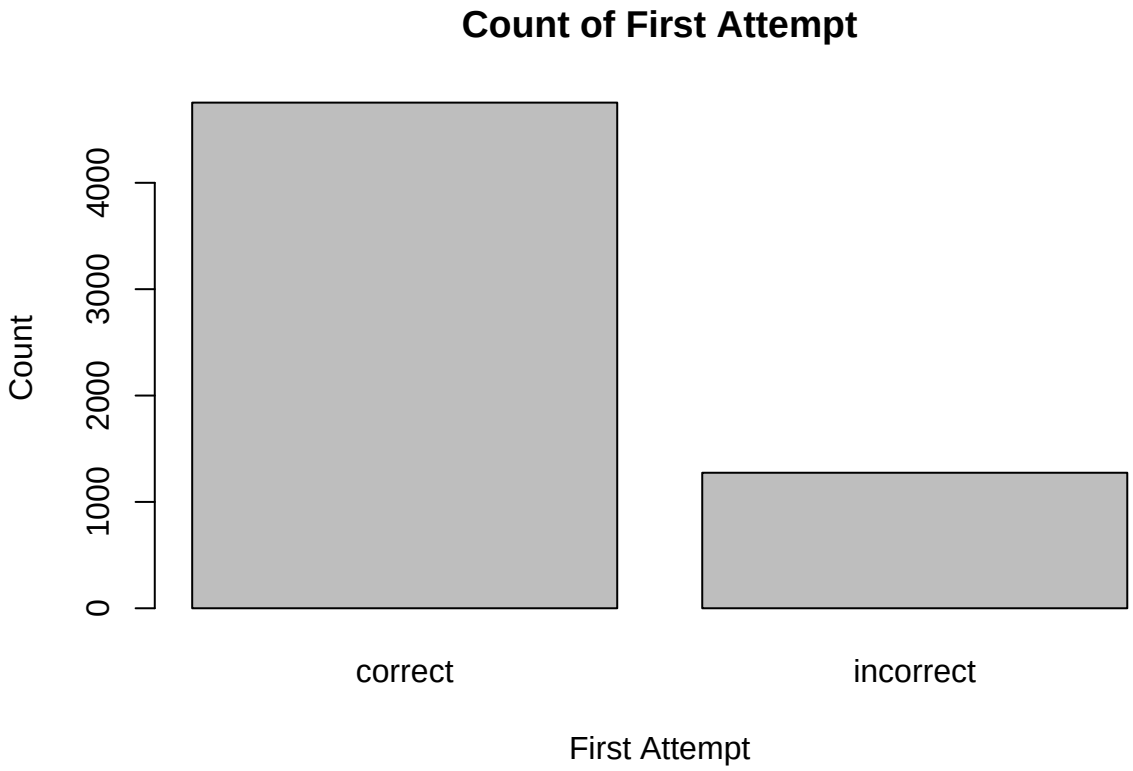
```
##
##             with (podsie_personal_deck) without (podsie_personal_deck)
##   correct                           683                            465
##   incorrect                         277                            175
```

```
posttest_contingency_table <- table(PostTest$FirstAttempt, PostTest$Condition)

print(posttest_contingency_table)
```

```
##
##             with (podsie_personal_deck) without (podsie_personal_deck)
##   correct                           895                            587
##   incorrect                          65                             53
```

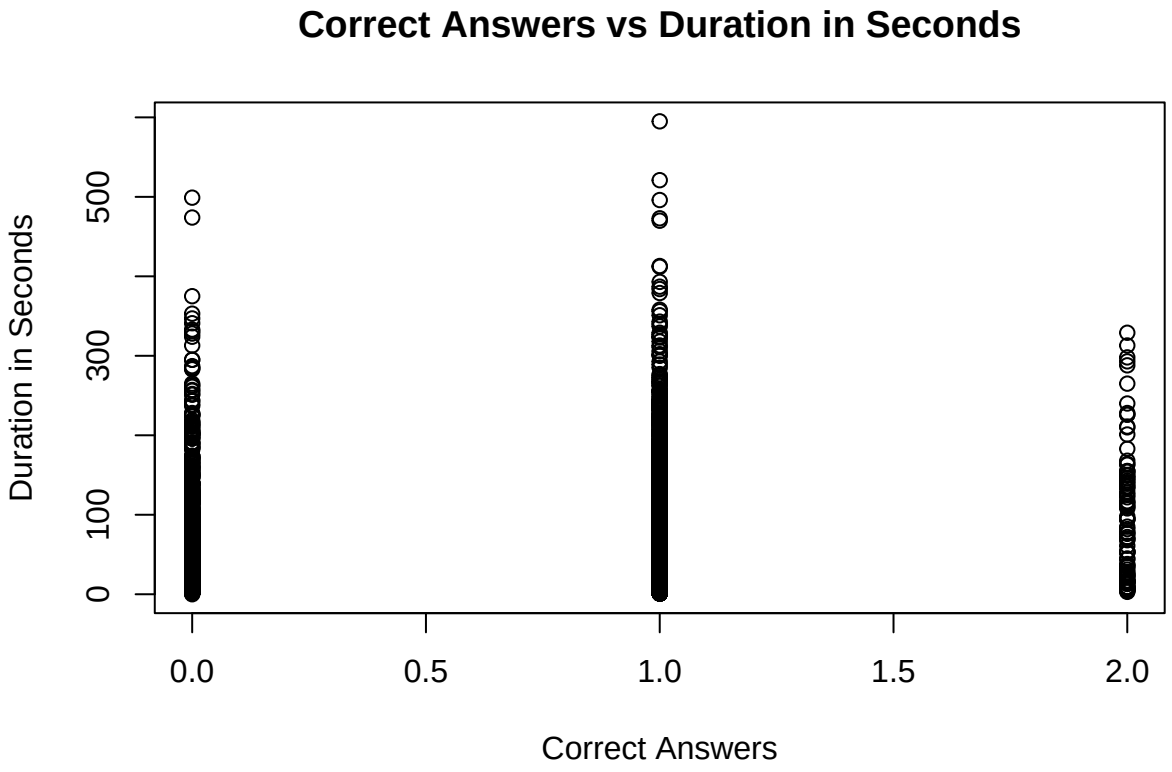
Bivariate Exploration It is important to visualize the data. Although we already know the number of incorrect and correct answers, it should be shown in a barplot.

```
barplot(table(algebraData$FirstAttempt),
        main = "Count of First Attempt",
        xlab = "First Attempt", ylab = "Count")
```

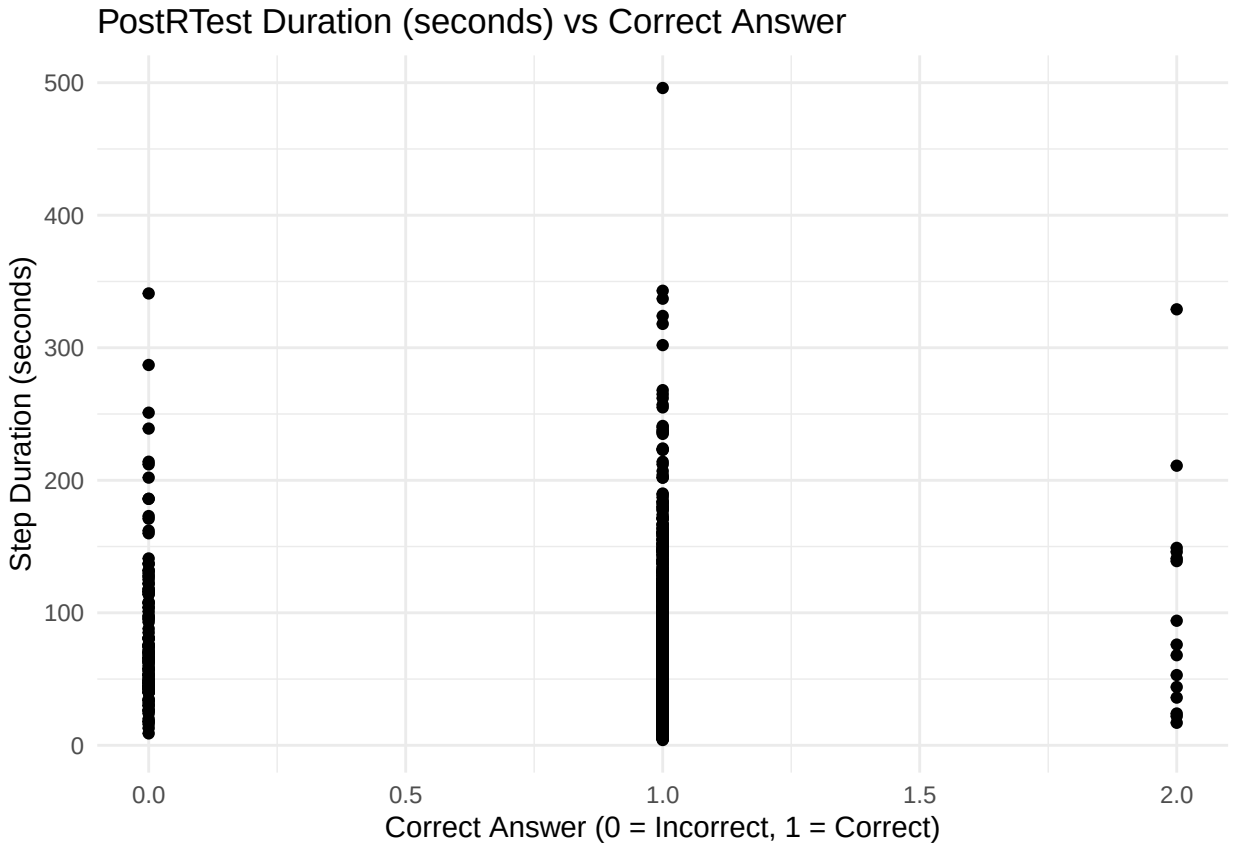


After examining the accuracy in answers, we can now try to see the correlation with the number of seconds it takes to answer the question.

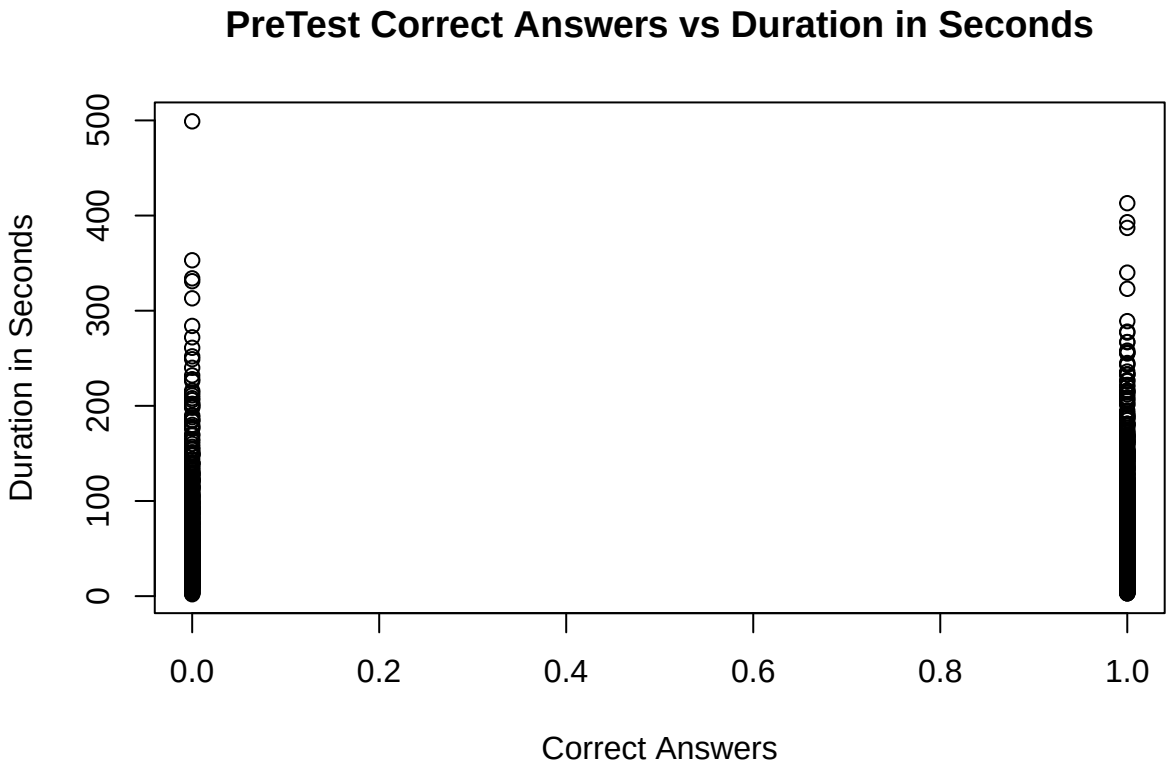
```
plot(StepDuration ~ Correct,
     data = algebraData,
     main = "Correct Answers vs Duration in Seconds",
     ylab = "Duration in Seconds",
     xlab = "Correct Answers")
```



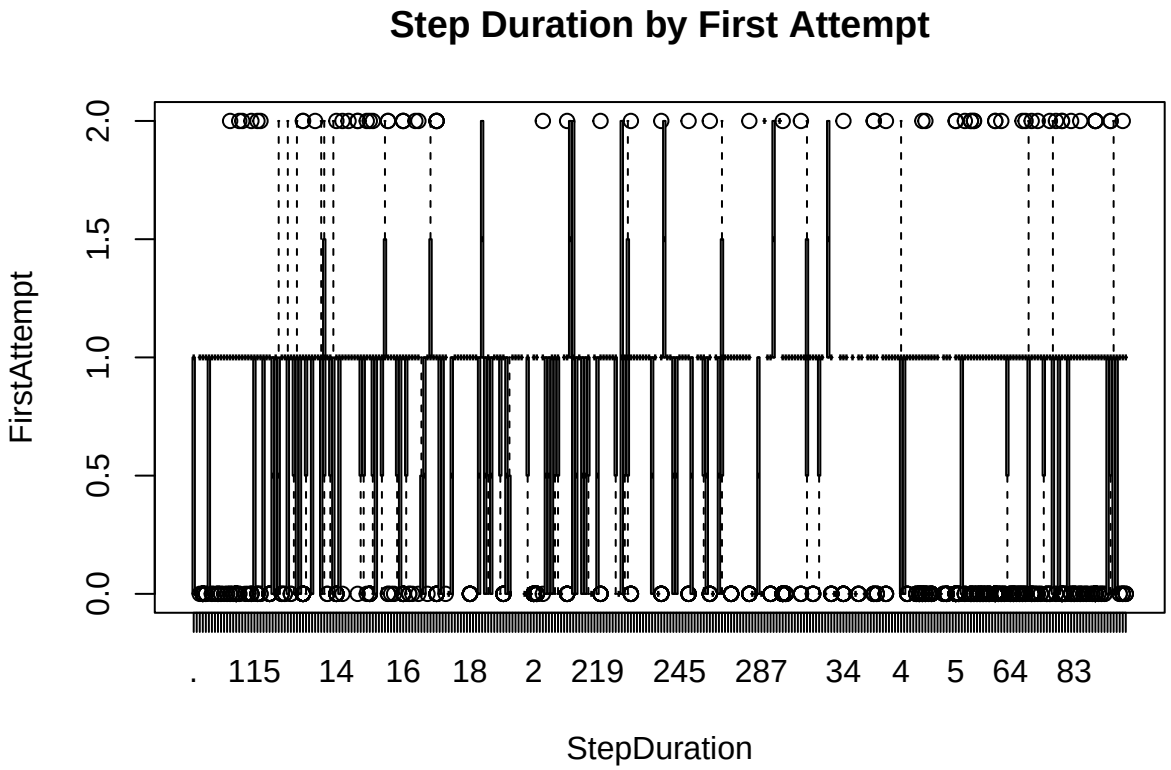

```
library(ggplot2)

ggplot(PostTest, aes(x = Correct, y = StepDuration)) +
  geom_point() +
  labs(title = "PostRTest Duration (seconds) vs Correct Answer",
       y = "Step Duration (seconds)",
       x = "Correct Answer (0 = Incorrect, 1 = Correct)") +
  theme_minimal()
```




```
plot(StepDuration ~ Correct,
     data = PreTest,
     main = "PreTest Correct Answers vs Duration in Seconds",
     ylab = "Duration in Seconds",
     xlab = "Correct Answers")
```




```
boxplot(Correct ~ StepDuration, data = algebraData,
        main = "Step Duration by First Attempt",
        xlab = "StepDuration",
        ylab = "FirstAttempt")
```



The scatterplot above shows the correlation between seconds and correct answers. 0 is incorrect, and 1 is

correct. More seconds spent answering the question increases the likelihood of receiving the correct answer. It was also graphed with the same variables on a boxplot which is a lot messier and difficult to read.

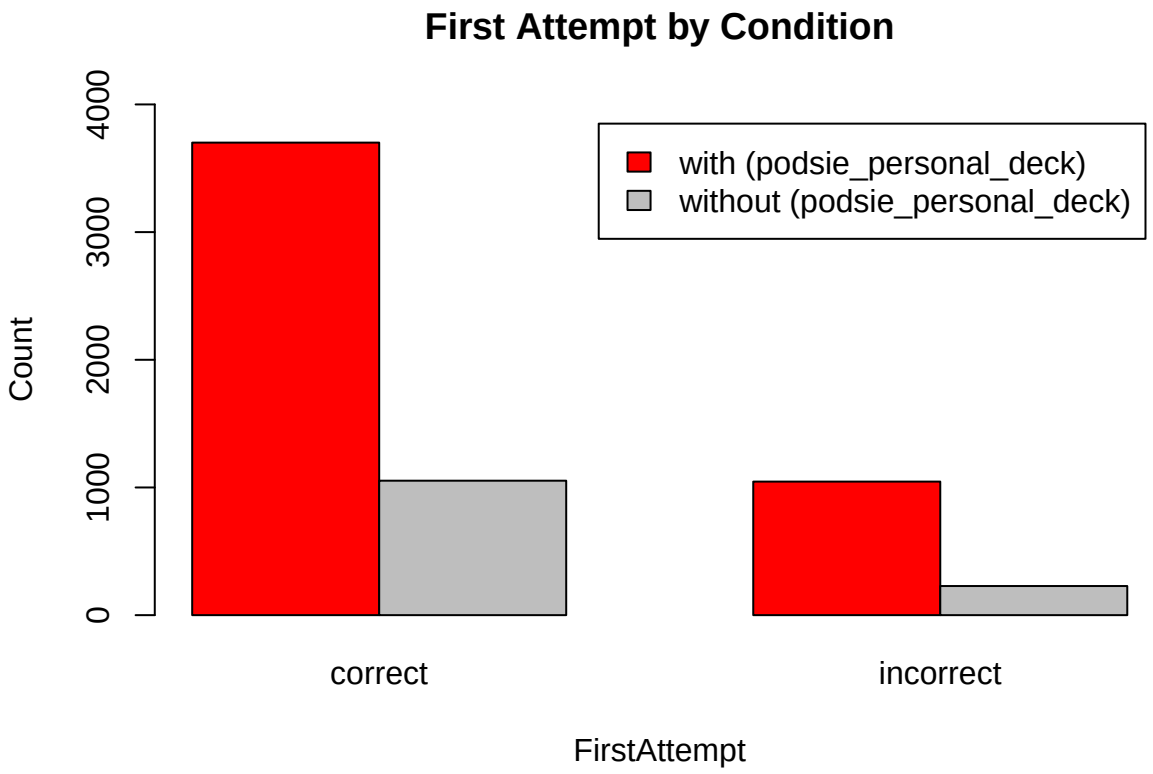
```
contingency_table1 <- table(algebraData$FirstAttempt, algebraData$Condition)
print(contingency_table1)
```

```
##
##             with (podsie_personal_deck) without (podsie_personal_deck)
##   correct                          3701                            1053
##   incorrect                        1046                             228
```

We can see that while Podsie produces more correct answers, it can also produce more incorrect answers than without the usage of it.

Here is a barplot to visually see the above data.

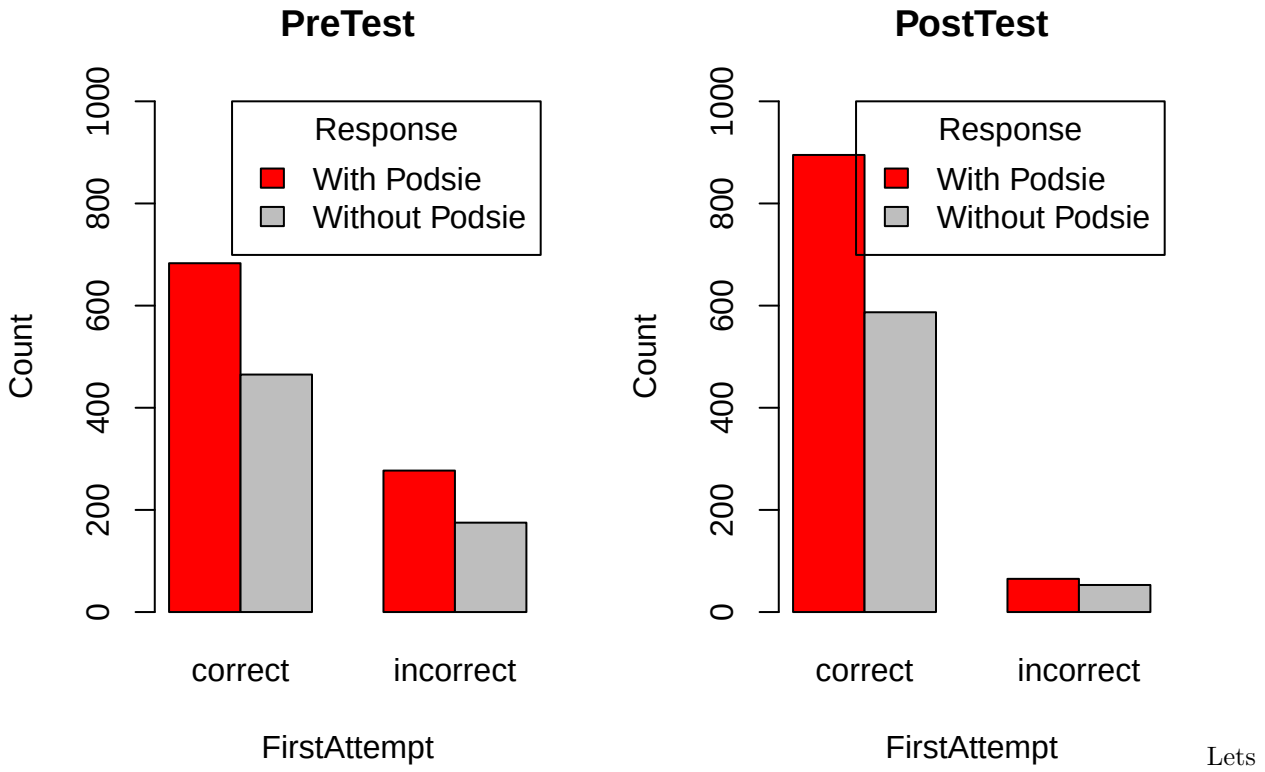
```
barplot(table(algebraData$Condition, algebraData$FirstAttempt), beside = TRUE,
        main = "First Attempt by Condition",
        xlab = "FirstAttempt",
        ylab = "Count",
        ylim = c(0,4000),
        legend = rownames(table(algebraData$Condition, algebraData$FirstAttempt)),
        col = c("red", "grey"))
```



Lets see that same plot but seperately for PreTest & PostTest.

```
pretest_counts <- table(PreTest$Condition, PreTest$FirstAttempt)
posttest_counts <- table(PostTest$Condition, PostTest$FirstAttempt)

par(mfrow = c(1, 2))

barplot(pretest_counts, beside = TRUE,
```

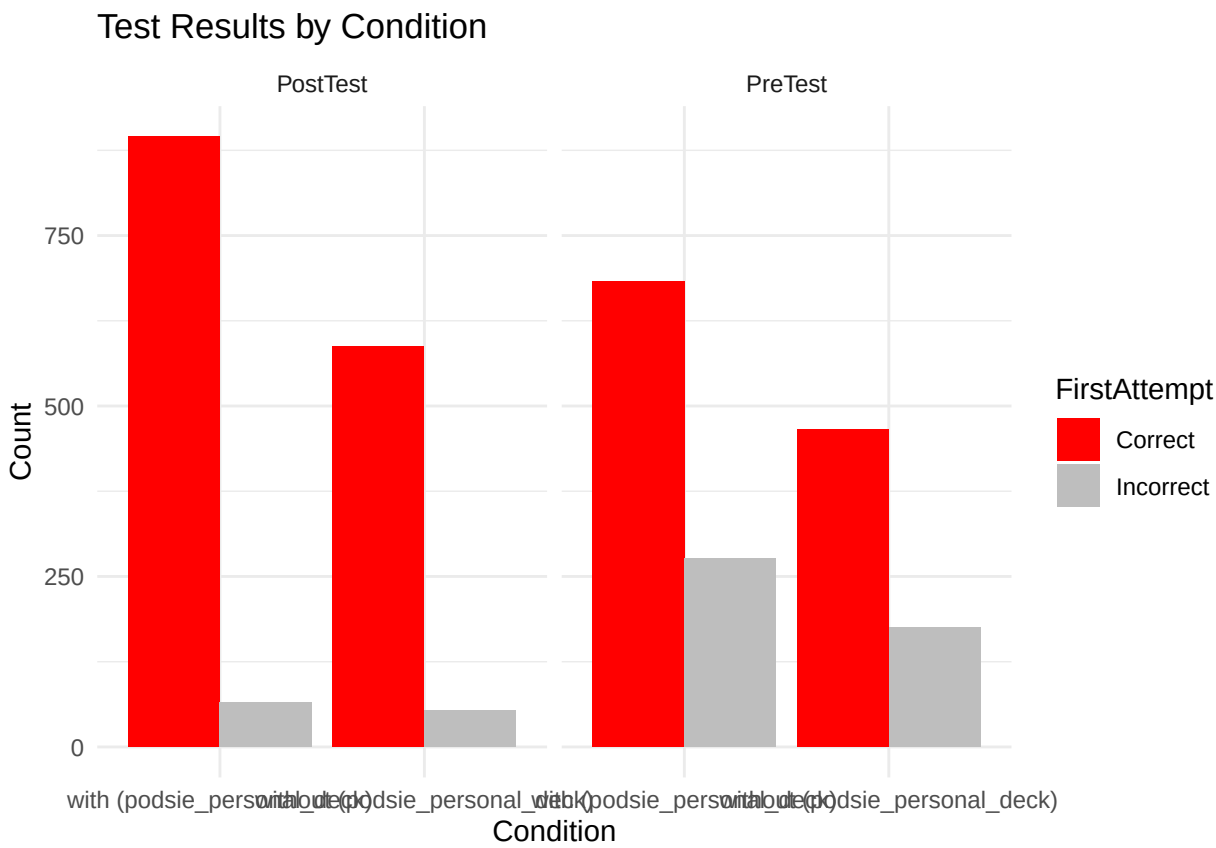
```
        col = c("red", "grey"),
        main = "PreTest",
        xlab = "FirstAttempt",
        ylab = "Count",
        ylim = c(0,1000),
        legend.text = c("With Podsie", "Without Podsie"),
        args.legend = list(title = "Response", x = "topright"))

barplot(posttest_counts, beside = TRUE,
        col = c("red", "grey"),
        main = "PostTest",
        xlab = "FirstAttempt",
        ylim = c(0,1000),
        ylab = "Count",
        legend.text = c("With Podsie", "Without Podsie"),
        args.legend = list(title = "Response", x = "topright"))
```



Lets now swap the variables and convert it into a ggplot.

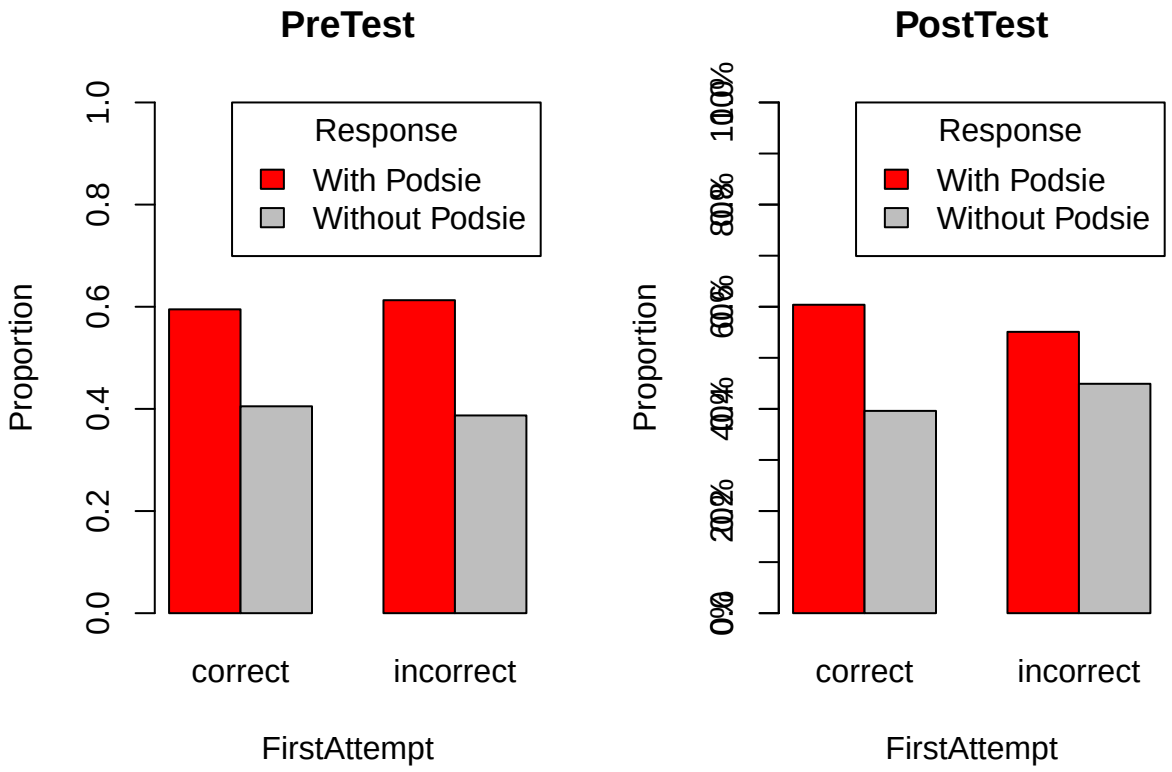
```
library(ggplot2)
library(dplyr)
library(tidyr)

# Assuming PreTest and PostTest are your dataframes
pretest_counts <- table(PreTest$Condition, PreTest$FirstAttempt)
posttest_counts <- table(PostTest$Condition, PostTest$FirstAttempt)

# Convert tables to data frames
pretest_df <- as.data.frame(pretest_counts)
posttest_df <- as.data.frame(posttest_counts)
```

```r
# Rename columns for clarity
names(pretest_df) <- c("Condition", "FirstAttempt", "Count")
names(posttest_df) <- c("Condition", "FirstAttempt", "Count")

# Add a column to distinguish between pretest and posttest
pretest_df$TestType <- "PreTest"
posttest_df$TestType <- "PostTest"

# Combine both data frames
combined_df <- rbind(pretest_df, posttest_df)

# Plot using ggplot2
ggplot(combined_df, aes(x = Condition, y = Count, fill = FirstAttempt)) +
  geom_bar(stat = "identity", position = "dodge") +
  facet_wrap(~ TestType) +
  scale_fill_manual(values = c("red", "grey"), labels = c("Correct", "Incorrect")) +
  labs(title = "Test Results by Condition",
       x = "Condition",
       y = "Count",
       fill = "FirstAttempt") +
  theme_minimal()
```




```r
pretest_counts <- table(Condition = PreTest$Condition, FirstAttempt = PreTest$FirstAttempt)
posttest_counts <- table(Condition = PostTest$Condition, FirstAttempt = PostTest$FirstAttempt)

pretest_proportions <- prop.table(pretest_counts, margin = 2)
```

```r
posttest_proportions <- prop.table(posttest_counts, margin = 2)

par(mfrow = c(1, 2))

barplot(pretest_proportions, beside = TRUE,
        col = c("red", "grey"),
        main = "PreTest",
        xlab = "FirstAttempt",
        ylab = "Proportion",
        ylim = c(0, 1),
        legend.text = c("With Podsie", "Without Podsie"),
        args.legend = list(title = "Response", x = "topright"))

barplot(posttest_proportions, beside = TRUE,
        col = c("red", "grey"),
        main = "PostTest",
        xlab = "FirstAttempt",
        ylab = "Proportion",
        ylim = c(0, 1),
        legend.text = c("With Podsie", "Without Podsie"),
        args.legend = list(title = "Response", x = "topright"))

axis(2, at = seq(0, 1, by = 0.1), labels = paste0(seq(0, 100, by = 10), "%"))
```




```r
library(ggplot2)
library(dplyr)

create_plot <- function(data, title) {
  data <- data %>%
```

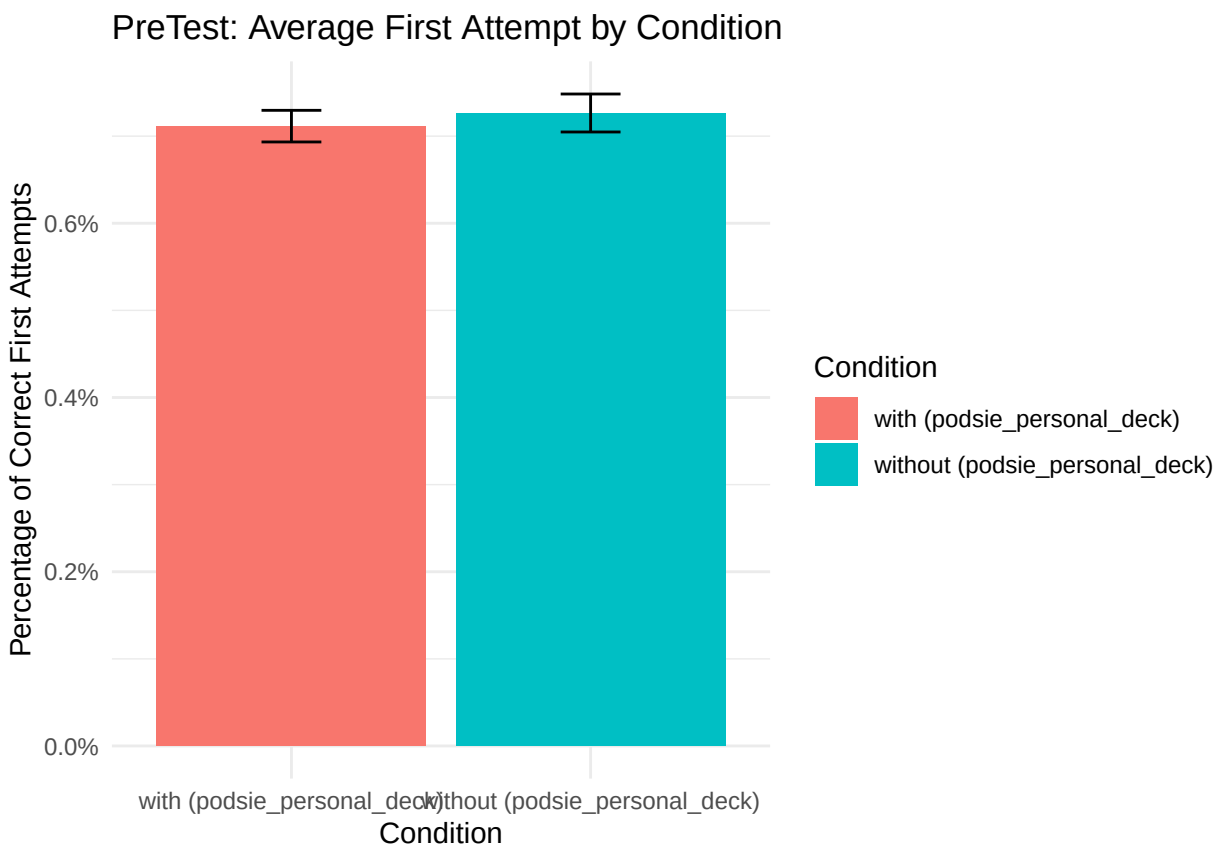
```
    mutate(FirstAttemptBinary = ifelse(FirstAttempt == "correct", 1, 0))

  student_averages <- data %>%
    group_by(StudentId, Condition) %>%
    summarise(Average = mean(FirstAttemptBinary))

  condition_averages <- student_averages %>%
    group_by(Condition) %>%
    summarise(
      Mean = mean(Average),
      SE = sd(Average) / sqrt(n())
    )

  print(condition_averages)

  ggplot(condition_averages, aes(x = Condition, y = Mean, fill = Condition)) +
    geom_bar(stat = "identity", position = "dodge") +
    geom_errorbar(aes(ymin = Mean - SE, ymax = Mean + SE), width = 0.2, position = position_dodge(0.9)) +
    scale_y_continuous(labels = scales::percent_format(scale = 1)) +
    labs(title = title,
         x = "Condition",
         y = "Percentage of Correct First Attempts",
         fill = "Condition") +
    theme_minimal()
}

pretest_plot <- create_plot(PreTest, "PreTest: Average First Attempt by Condition")
```

```
## # A tibble: 2 x 3
##   Condition                       Mean     SE
##   <chr>                          <dbl>  <dbl>
## 1 with (podsie_personal_deck)    0.711 0.0182
## 2 without (podsie_personal_deck) 0.727 0.0218
```

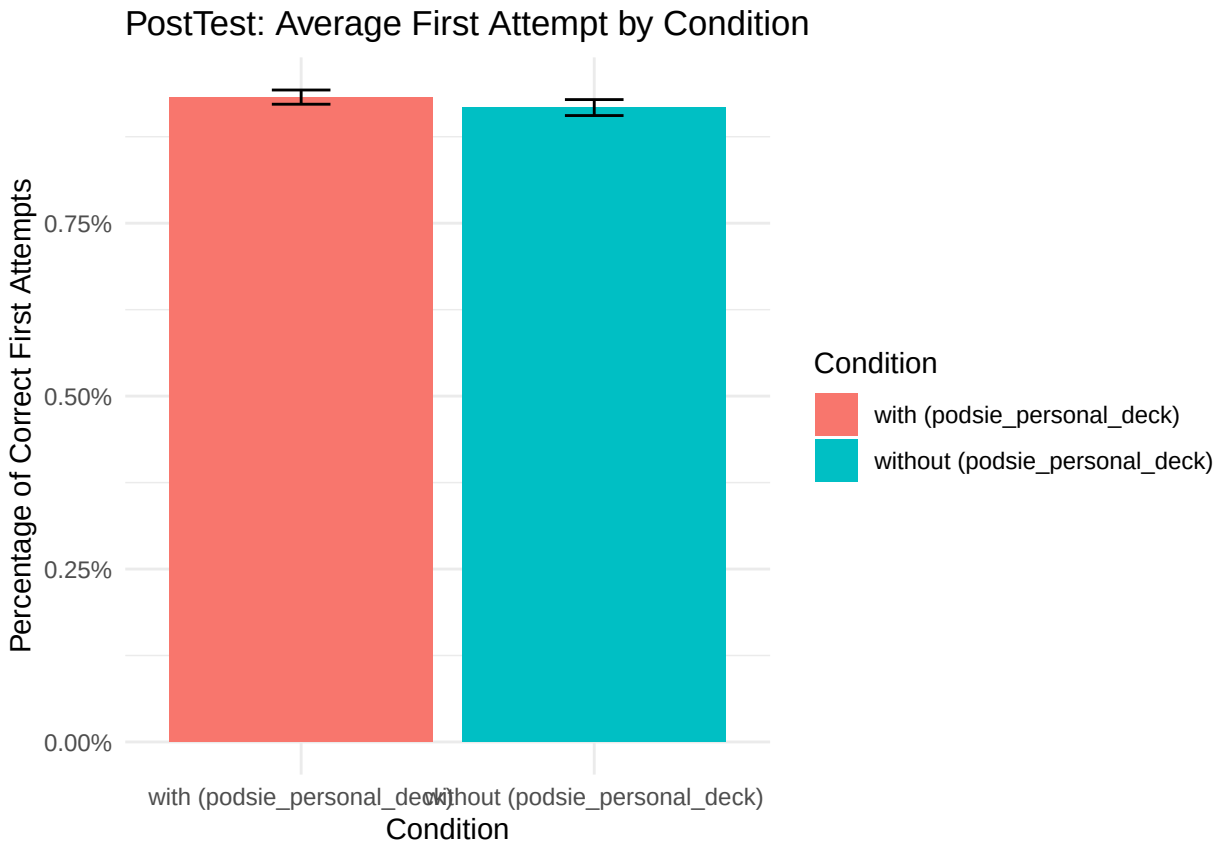
```
posttest_plot <- create_plot(PostTest, "PostTest: Average First Attempt by Condition")
```

```
## # A tibble: 2 x 3
##   Condition                       Mean     SE
##   <chr>                          <dbl>  <dbl>
## 1 with (podsie_personal_deck)    0.932 0.0103
## 2 without (podsie_personal_deck) 0.917 0.0115
```

```
print(pretest_plot)
```

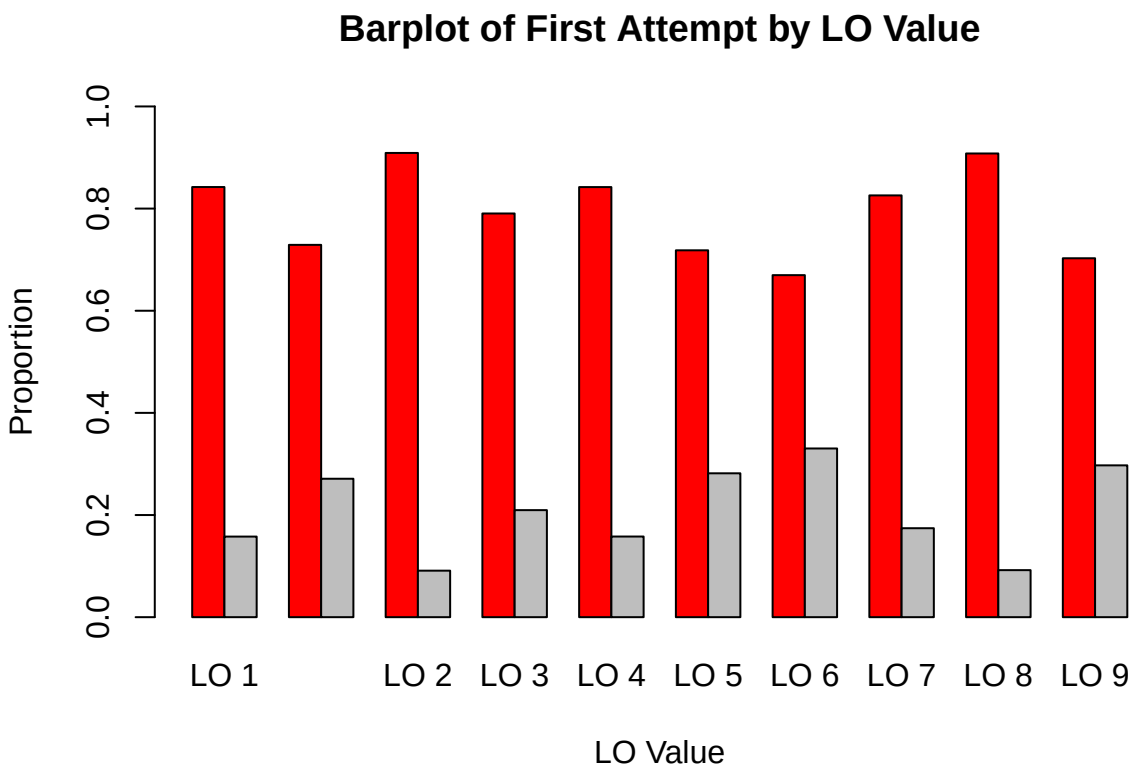



```
print(posttest_plot)
```



This makes one wonder if the LO Values can affect spacing.

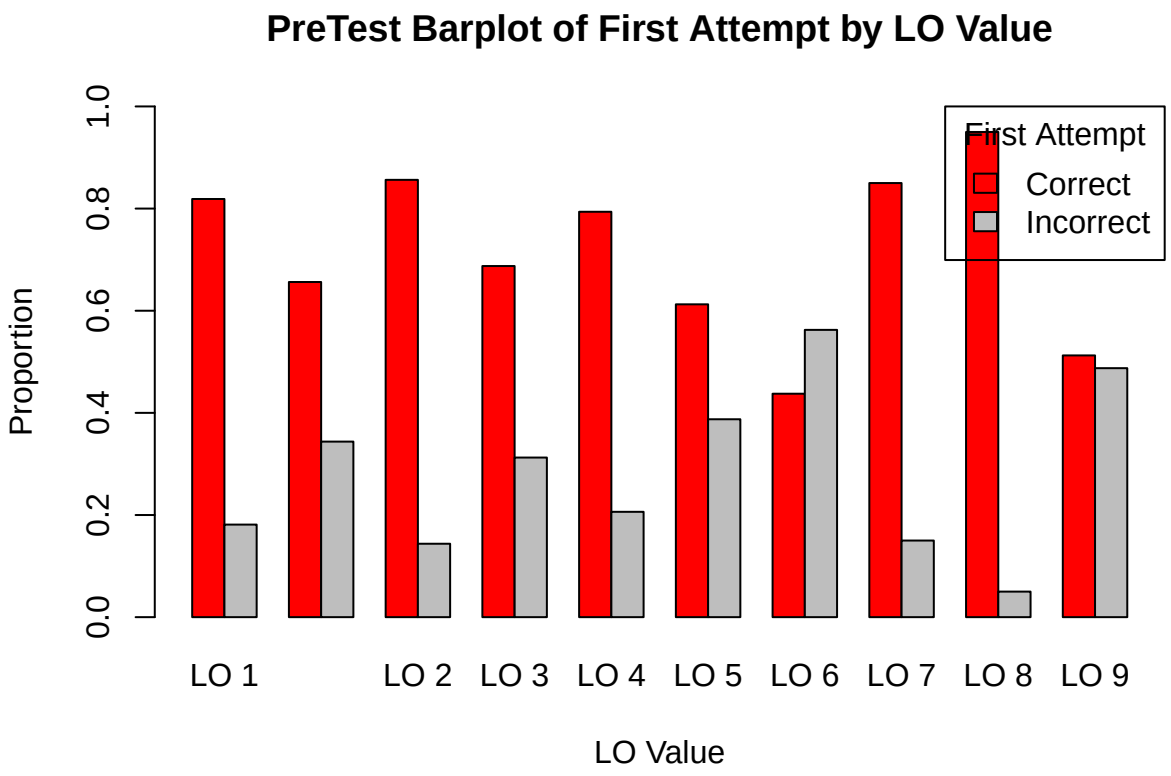
```
barplot(
  prop.table(
  table(algebraData$FirstAttempt, algebraData$KCLO),
  margin = 2),
  beside = TRUE,
  col = c("red", "grey"),
  main = "Barplot of First Attempt by LO Value",
  xlab = "LO Value",
  ylab = "Proportion",
  ylim = c(0,1)
)
```



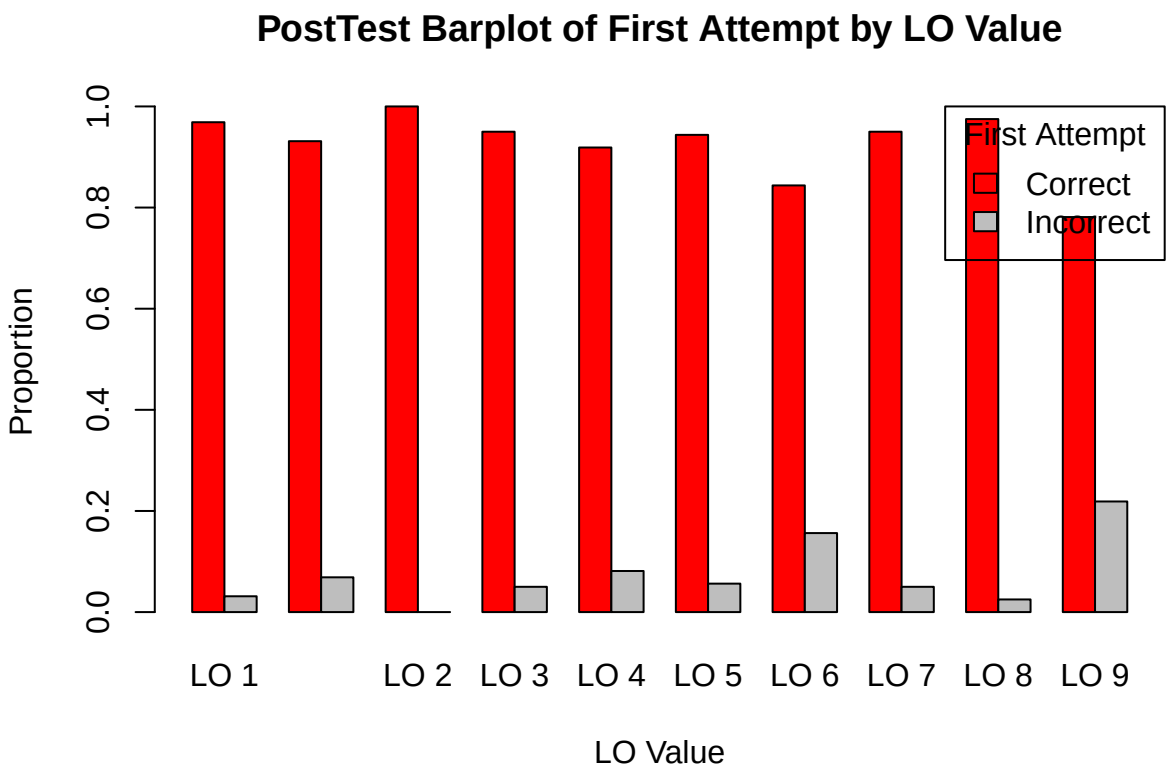
This bar graph shows that the LO Value of 3 and 9 had the fewest incorrect answers, implying that they also had the most correct answers.

Lets now see the LO value seperately for PreTest & PostTest instead of together.

```
colors <- c("red", "grey")

barplot(
  prop.table(table(PreTest$FirstAttempt, PreTest$KCLO), margin = 2),
  beside = TRUE,
  col = colors,
  main = "PreTest Barplot of First Attempt by LO Value",
  xlab = "LO Value",
  ylab = "Proportion",
  legend.text = TRUE,
  args.legend = list(title = "First Attempt", x = "topright", legend = c("Correct", "Incorrect")),
  names.arg = levels(PreTest$KCLO),
  ylim = c(0, 1)
)
```

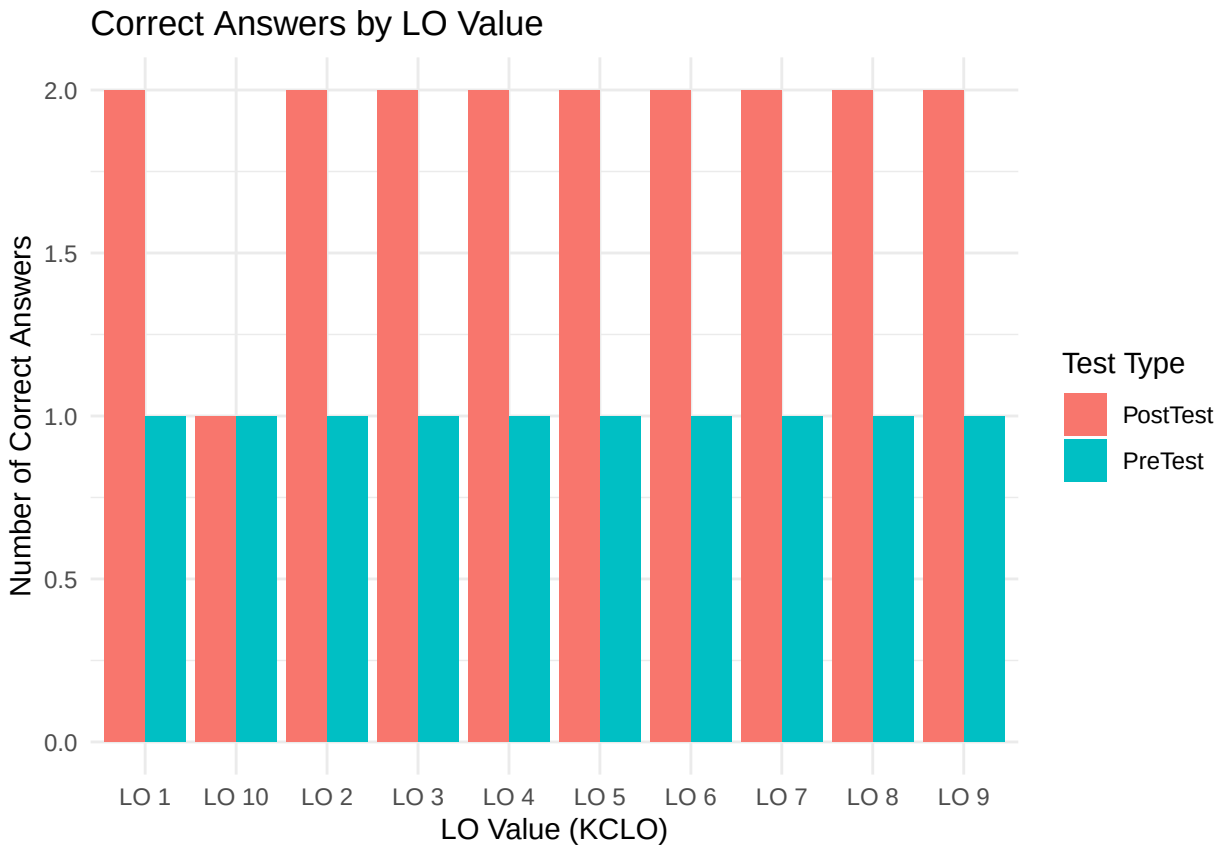



```r
colors <- c("red", "grey")

barplot(
  prop.table(table(PostTest$FirstAttempt, PostTest$KCLO), margin = 2),
  beside = TRUE,
  col = colors,
  main = "PostTest Barplot of First Attempt by LO Value",
  xlab = "LO Value",
  ylab = "Proportion",
  legend.text = TRUE,
  args.legend = list(title = "First Attempt", x = "topright", legend = c("Correct", "Incorrect")),
  names.arg = levels(PostTest$KCLO)
)
```



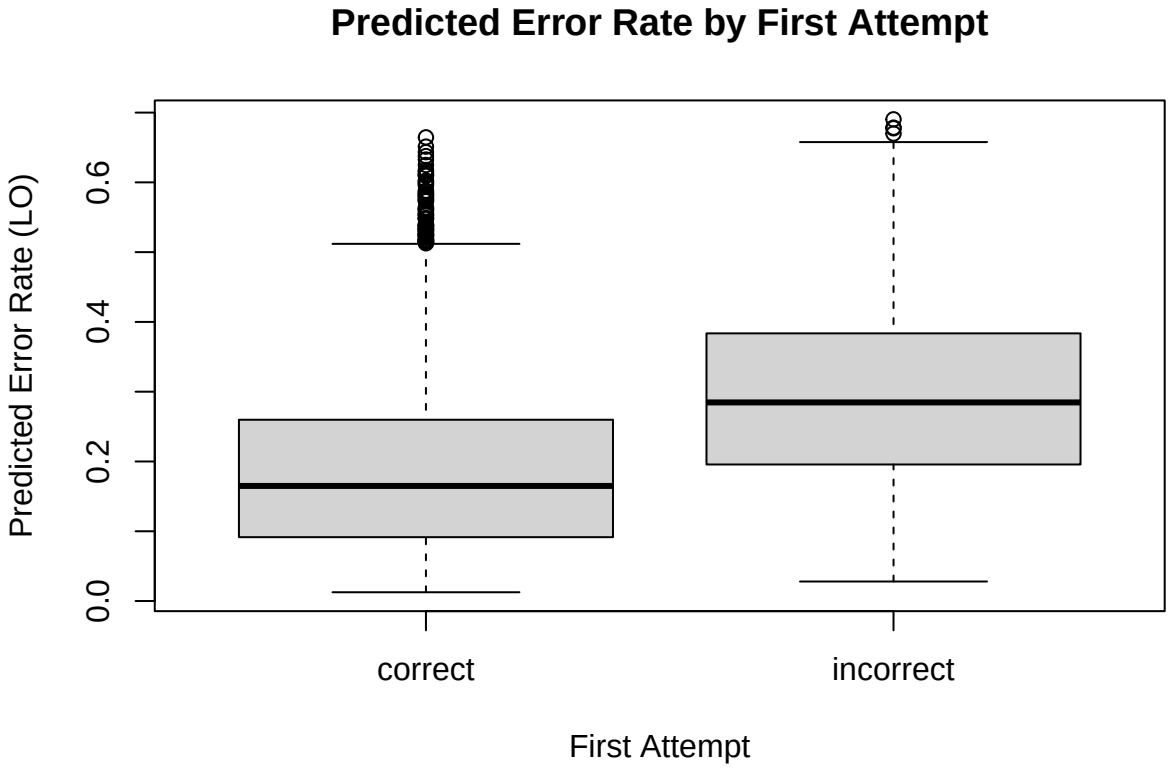
Now that we looked at Pre-Test & Post-Test seperately, lets see it together in a different plot.

```
PreTest$StepDuration <- as.numeric(as.character(PreTest$StepDuration))
PostTest$StepDuration <- as.numeric(as.character(PostTest$StepDuration))

PreTest$TestType <- 'PreTest'
PostTest$TestType <- 'PostTest'

combinedData <- bind_rows(PreTest, PostTest)

ggplot(combinedData, aes(x = factor(KCLO), y = Correct, fill = TestType)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Correct Answers by LO Value",
       x = "LO Value (KCLO)",
       y = "Number of Correct Answers",
       fill = "Test Type") +
  theme_minimal()
```



It is also important to look at the Predicted Error in the LO values.

```
boxplot(PredictedErrorRateLO ~ FirstAttempt, data = algebraData,
        main = "Predicted Error Rate by First Attempt",
        xlab = "First Attempt",
        ylab = "Predicted Error Rate (LO)")
```



Looking at the boxplot above for correct answer the predicted error rate for LO value ranges from 0.05 to 0.5 with several outliers above it, the median is around 0.15 ranging from 0.10 to 0.28. While for incorrect the boxplot ranges from 0.02 to 0.65 with only a few outliers above it, the median is around 0.28 which is higher than correct answer boxplot from 0.2 0.38.

Below is the summary for PredictedErrorRateLO not seperated by correct and incorrect.

```
summary(algebraData$PredictedErrorRateLO)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0126  0.1083  0.1888  0.2113  0.2947  0.6903
```

After looking at the LO value now we can sort the variable ProblemName which is the question by LO value from 1 to 10.

```
SortedProblemName <- algebraData[order(algebraData$KCLO, algebraData$ProblemName), c("ProblemName", "KC

print(head(SortedProblemName, 10))
```

```
##                                                          ProblemName KCLO
## 336  Simplify the expression.  \\left(5m-10\\right)+\\left(3-2m\\right) LO 1
## 412  Simplify the expression.  \\left(5m-10\\right)+\\left(3-2m\\right) LO 1
## 479  Simplify the expression.  \\left(5m-10\\right)+\\left(3-2m\\right) LO 1
## 549  Simplify the expression.  \\left(5m-10\\right)+\\left(3-2m\\right) LO 1
## 616  Simplify the expression.  \\left(5m-10\\right)+\\left(3-2m\\right) LO 1
## 700  Simplify the expression.  \\left(5m-10\\right)+\\left(3-2m\\right) LO 1
## 1030 Simplify the expression.  \\left(5m-10\\right)+\\left(3-2m\\right) LO 1
## 1104 Simplify the expression.  \\left(5m-10\\right)+\\left(3-2m\\right) LO 1
## 1178 Simplify the expression.  \\left(5m-10\\right)+\\left(3-2m\\right) LO 1
## 1261 Simplify the expression.  \\left(5m-10\\right)+\\left(3-2m\\right) LO 1
```

Above are the questions associated with each LO value. In excel I was able to sort the questions by LO value

as well making it easier to view.

Most of the questions associated with LO1 involve "Simplfy".

For LO2 it included "Which of the following expressions."

LO3 includes either "Which of the following relations represent a function?" or "Which ordered pair could be added to the set below to keep it a function?

L04 includes "The graph of a function" or "If a function" or "For the function."

L05 is mainly involving inequality or figuring out which is the correct solution.

LO6 is "Which of the following ordered pairs are solutions to the inequality" or "Which scenario could be modeled by this graph?" or "Which inequality could be modeled by this graph?" or "Consider the inequality 4x-2y<8 Determine all the ordered pairs that are solutions to this inequality."

LO7 includes "Which is the equation of the line that passes through the points" or "What is the slope of a line with an equation" or "Given the points(?1,4) and (3,?2), what are the slope (m) and y-intercept (b) of the line that passes through these points?."

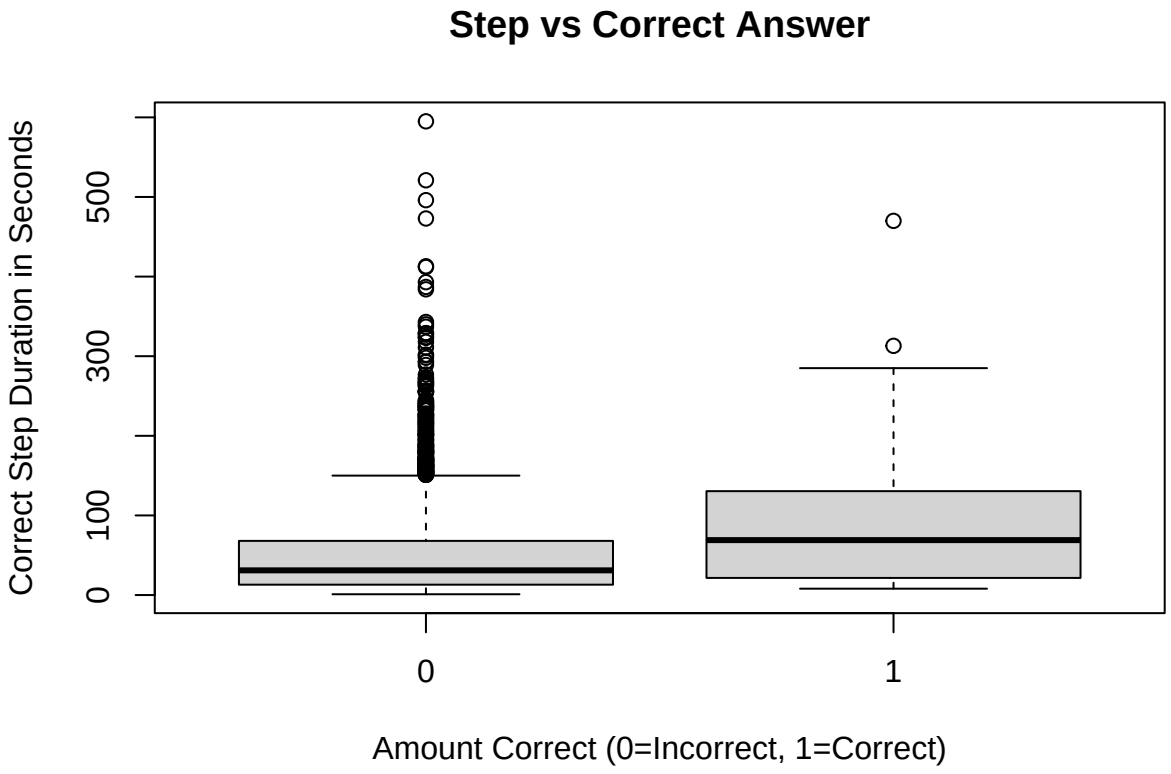
LO8 is involving different solutions.

LO9 includes "Simplfy"

L10 is "Integers are not closed under which operation?" or "Select ALL statements that are true." or "Select all statements below that are true." or "The set below only contains which types of numbers?" or "Which categories of numbers can be found in the set below? Select ALL that apply" or "Which of the following operations does NOT always result in an integer when performed"

We will now look at the variable CorrectStepDuration which is the amount of seconds it takes to answer a question.

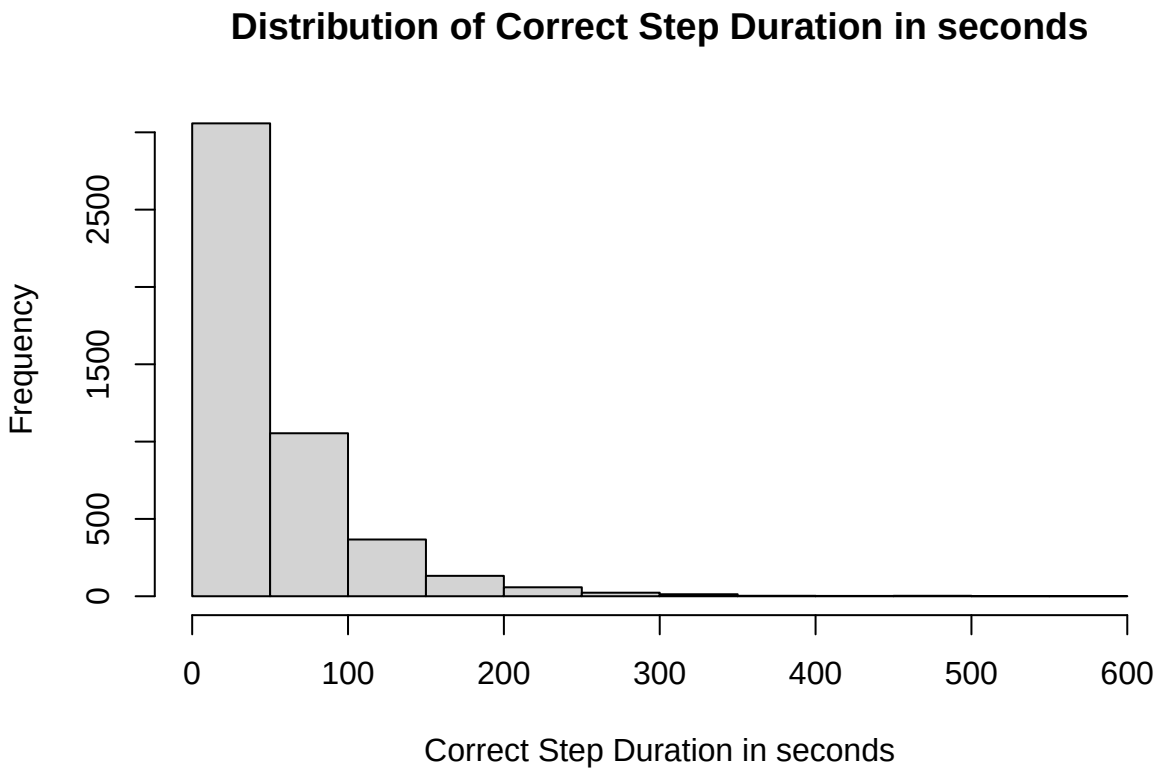
```
algebraData$CorrectStepDuration <- as.numeric(as.character(algebraData$CorrectStepDuration))

boxplot(CorrectStepDuration ~ Incorrect,
       data = algebraData,
       ylab = "Correct Step Duration in Seconds",
       xlab = "Amount Correct (0=Incorrect, 1=Correct)",
       main = "Step vs Correct Answer")
```



In this boxplot it shows that less seconds leads to an incorrect answer with several outliers while more seconds leads to a higher frequency in a correct answer with fewer outliers.

Looking at CorrectStepDuration individually:

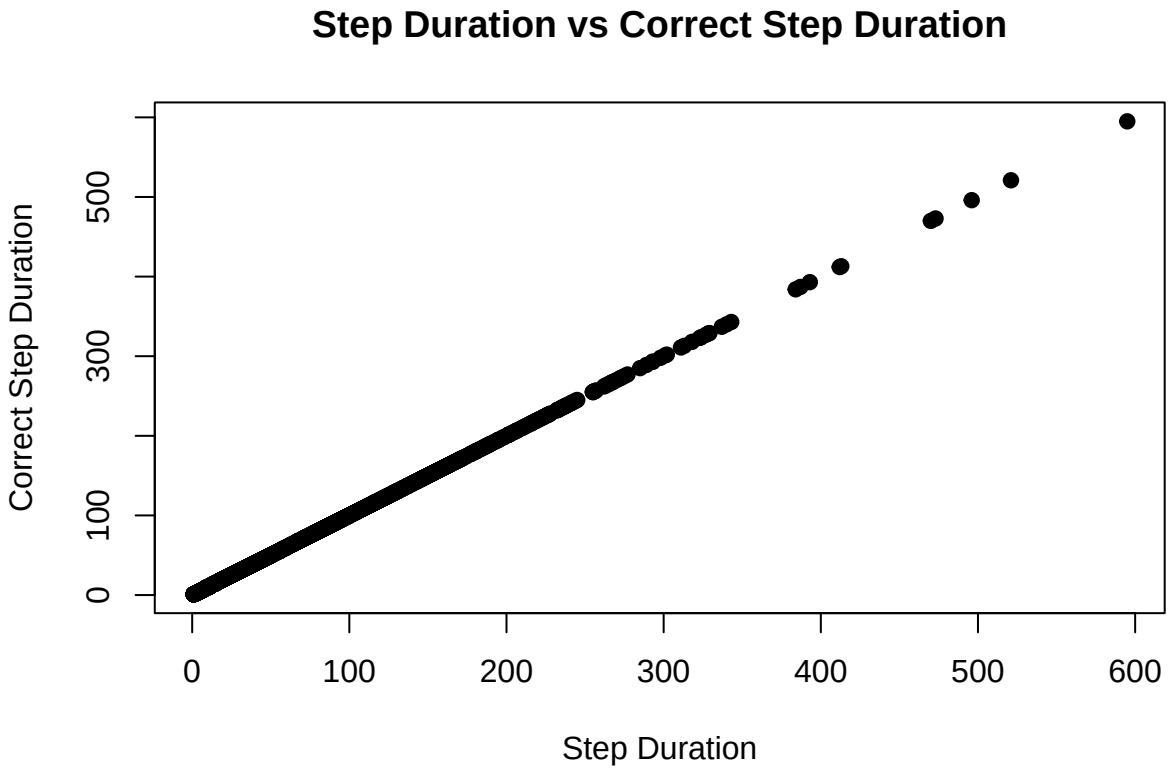
```
hist(algebraData$CorrectStepDuration,
     main = "Distribution of Correct Step Duration in seconds",
     xlab = "Correct Step Duration in seconds", breaks = 10)
```



The histogram above shows the CorrectStepDuration is strongly right skewed.

Lets now see the relationship between the variables StepDuration and CorrectStepDuration.

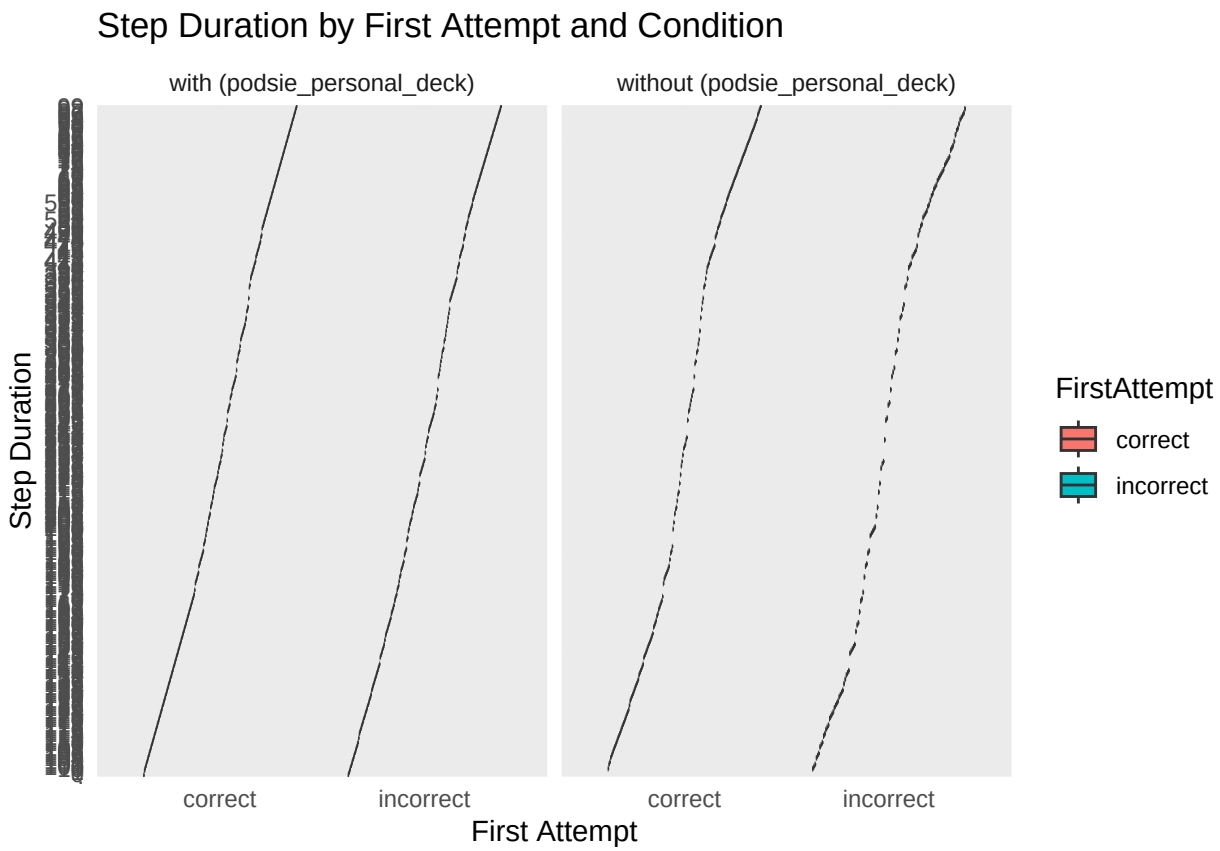
```
plot(algebraData$StepDuration, algebraData$CorrectStepDuration,
     main = "Step Duration vs Correct Step Duration",
     xlab = "Step Duration",
     ylab = "Correct Step Duration", pch = 19)
```



The straight scatterplot suggests that there is no significant difference between the two variables, StepDuration and CorrectStepDuration has a sharp straight line showing the relationship between both.

Now we can look at the relationship between CorrectStepDuration and FirstAttempt with a ggplot.

```
ggplot(algebraData, aes(x = FirstAttempt, y = StepDuration, fill = FirstAttempt)) +
  geom_boxplot() +
  facet_wrap(~ Condition) +
  labs(title = "Step Duration by First Attempt and Condition",
       x = "First Attempt", y = "Step Duration") +
  theme_minimal()
```



With the condition the plot is more linear which shows a consistent trend meaning it may imply that the condition has a more consistent impact on the length of the step. Without the condition the plot is more straight in the middle and less steep near the ends which can suggest a less clear relationship.

Now that we've identified some variables that correlate with our response variable, we can go into modeling.

Modeling

We have analyzed the distribution and relationships between spacing and variables. Now, to predict spacing, we will build an ANCOVA model after examining and visualizing the relationships between our variables.

```
relevant_vars <- c("Correct", "Incorrect", "ProblemName", "Condition", "StepDuration", "KCLO", "Problem
reduced_data <- algebraData[ , relevant_vars]

ancova_model <- aov(Correct ~ Incorrect + ProblemName + Condition + KCLO + StepDuration, data = reduced
summary(ancova_model)
```

```
##                Df Sum Sq Mean Sq   F value Pr(>F)
## Incorrect       1  639.6   639.6 12869.433 <2e-16 ***
## ProblemName    85   38.6     0.5     9.139 <2e-16 ***
## Condition       1   14.8    14.8   297.720 <2e-16 ***
## StepDuration  307   49.7     0.2     3.257 <2e-16 ***
## Residuals    5633  280.0     0.0
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Some of the variables in the model do not show up in the summary this can be due to multicolinearly and more.

```r
vif_model <- lm(Correct ~ Incorrect + ProblemName + Condition, data = reduced_data)
vif(vif_model)
```

```
##                 GVIF Df GVIF^(1/(2*Df))
## Incorrect   1.136662  1        1.066144
## ProblemName 1.374889 85        1.001875
## Condition   1.215935  1        1.102694
```

We will now create a new model with PostTest model.

```r
relevant_vars1 <- c("Correct", "Incorrect", "Condition", "FirstAttempt", "ProblemName", "StepDuration")

reduced_data1 <- PostTest[ , relevant_vars1]

ancova_model1 <- aov(Correct ~ Condition + StepDuration + FirstAttempt + ProblemName, data = reduced_da

summary(ancova_model1)
```

```
##                Df Sum Sq Mean Sq  F value Pr(>F)
## Condition       1   0.09    0.09    8.500 0.0036 **
## StepDuration    1   1.64    1.64  148.961 <2e-16 ***
## FirstAttempt    1 104.49  104.49 9474.025 <2e-16 ***
## ProblemName    19   0.13    0.01    0.633 0.8837
## Residuals    1577  17.39    0.01
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

This is our new model. The p-values Condition, StepDuration, FirstAttempt are significant and affect correct performance. The variable Correct was used as the reference since we are predicting correctness. The residuals had a sum of squares of 17.39 with 1577 degrees of freedom, indicating that the model fits the data reasonably well. FirstAttempt has the highest F value meaning it can be a critical factor in performance. Condition plays an important role in student outcomes since the purpose of this study was to see if Podsie can lead to higher performance. StepDuration informs us about the time it took to answer the question. FirstAttempt is important for overall performance. ProblemName can reflect the difficulty per question but it important to remember the 60% and 40% of what is kept in the Personal Deck.

The goal of this study was to investigate the impact of practice spacing (StepDuration), problem difficulty (KCLO), and experimental condition (Condition) on student performance (FirstAttempt) most importantly correctness (Correct). We predicted that students in the experimental condition would perform better on their FirstAttempt, after controlling for StepDuration and difficulty.

We used a dataset containing 6028 observations. The dependent variable was student performance (FirstAttempt), which had two levels: correct and incorrect. The independent variables included practice spacing (StepDuration), LO Values (KCLO), seconds needed for answering question (StepDuration) and experimental condition (Condition). An ANOVA test assess the impact of these independent variables on student performance.

Discussion

After looking at all of the variables, creating relevant graphs, and running ANCOVA models, it is safe to say that spacing can lead to better exam performance with Podsie. Specifically, the variable Condition, which differentiates between students who used Podsie spaced practice and those who did not, it has a statistically significant effect on performance (p-value $<$ 2e-16). It has an effect on the variable Correct. This highly significant p-value suggests that the difference in performance between the two conditions is not due to chance or randomization.

The results of this study indicate that the usage of spaced retrieval practice can greatly enhance student learning outcomes. To increase the success of their teaching methods, educators and teachers should consider utilizing programs like Podsie that incorporate spaced repetition algorithms, like SuperMemo2.

More research should be done to investigate the long-term effects of spaced retrieval practice, in addition to the four-week study's immediate post-test findings. Furthermore, investigating the effects of various spacing intervals and combining spaced practice with other learning techniques may provide additional insight into optimizing educational practices.

In conclusion, the findings of this study provide strong evidence that spaced retrieval practice, as enabled by Podsie, significantly improves students' Algebra I performance. Educators can use spacing to improve learning and retention, leading to higher student achievement.