```
---
title: "MPxMA profiles initial"
author: "Alena"
date: "8/29/2023"
output:
  html_document:
    df_print: paged
  pdf_document: default
---

```{r setup, include = FALSE, cache = TRUE}
knitr::opts_chunk$set(echo = TRUE)
```

```{r libraries, include = FALSE}
install.packages(c("readxl", "tidyverse", "formattable", "lme4", "lmerTest","writexl",
"irr", "sjPlot", "sjstats", "apaTables", "Hmisc", "dplyr", "DemographicTable",
"dunn.test","rstatix","gtsummary", "interactions" ), dependencies = TRUE,
repos="http://cran.us.r-project.org")

require(readxl)
require(tidyverse)
require(formattable)
require(lme4) # for mixed model
require(lmerTest) # for p-values in mixed model
require(writexl) # for exporting final excel
require(irr)
require(sjstats) # for tau 11
require(dplyr)
require(apaTables)
require(ggplot2)
require(plotly)
require(dunn.test)
require(rstatix) # to present dunn.test results in rows
library(DemographicTable) # for demographics table presented by group
require(gtsummary) # for shorter demographics table presented by group without Skewdness,
ect
require(sjPlot) # for comparison of lmer models
library(multcomp) # for Tukey test
library(tidyr) # for combined vizualizations
library(cluster) # for Silhouette Score
library(car)
library(interactions)



rm(list = ls())
setwd("~/Documents/Data_Analysis/MPxMA_Replication_poster")

```

```{r vis settings, include = FALSE}

# Define the Wes Anderson colors
#install.packages('wesanderson', repos="http://cran.us.r-project.org")
library(wesanderson)
wes_colors <- wes_palette(n = 5, name = "AsteroidCity1")

```

```{r assessment data, include = FALSE}

### Loading assessment data
```
```

```
## Loading assessment data
assess_student <- read.csv("assess_student.csv")

# Preparing Math performance var names
names(assess_student)[names(assess_student) == 'pre_total_math_score'] <- 'PreMP'
names(assess_student)[names(assess_student) == 'post_total_math_score'] <- 'PostMP'

# Preparing Math anxiety var names
names(assess_student)[names(assess_student) == 'pre_MA_total_score'] <- 'PreMA'
names(assess_student)[names(assess_student) == 'post_MA_total_score'] <- 'PostMA'

# Preparing MSE var names
names(assess_student)[names(assess_student) == 'pre_MSE_avg_score'] <- 'PreMSE'
names(assess_student)[names(assess_student) == 'post_MSE_avg_score'] <- 'PostMSE'

# Create difference in MP variable
assess_student$MPdif <- assess_student$PostMP - assess_student$PreMP

# Create difference in MA variable
assess_student$MAdif <- assess_student$PostMA - assess_student$PreMA

# Choosing columns
# Note: Scale.ScoreX = state assessment, math.grade7 - final course grade
assess_student <- assess_student [, (colnames(assess_student) %in%
  c('StuID',
    'PreMA', 'PostMA', 'MAdif',
    'pre_negative_reaction_score', 'pre_numerical_confindence_score', 'pre_worry_score',
    'PreMP', 'PostMP', 'MPdif',
    'delayed_total_math_score', 'Scale_Score7', 'Scale_Score5', 'math_grade7',
    'pre_sub_P_score', 'pre_sub_C_score', 'pre_sub_F_score',
    'PreMSE'))]

```

```{r demo data, include = FALSE}

### Loading demographic data
student_demo <- read.csv("student_demo.csv")

# Preparing gender var
student_demo$Gender <- dplyr::recode(dplyr::na_if(student_demo$Gender,""),
  'F' = 'Female',
  'M' = 'Male',
  .missing = 'Unknown')
student_demo$Gender <- as.factor(student_demo$Gender)

# Preparing ethnicities var
student_demo$race_ethnicity <- dplyr::recode(student_demo$race_ethnicity,
  '1' = 'Hispanic/Latino',
  '2' = 'American Indian/Alaska Native',
  '3' = 'Asian',
  '4' = 'Black/African American',
  '5' = 'Native Hawaiian or Other Pacific Islander',
  '6' = 'White',
  '7' = 'Two or more races')
student_demo$race_ethnicity <- as.factor(student_demo$race_ethnicity)

# Choosing columns
student_demo <- student_demo [, (colnames(student_demo) %in%
  c('StuID', 'Gender', 'race_ethnicity', 'IEP', 'EIP', 'GIFTED', 'ESOL'))]

```

```{r roster data, echo=FALSE}
```

```
## Loading roster data
student_roster <- read.csv("student_roster.csv")

## Create variable that shows if student stayed in the same class until the end of the
intervention
student_roster$Remained_in_same_class <- ifelse(student_roster$ClaIDPre ==
student_roster$ClaIDEnd, TRUE, FALSE)

## Modify movement variable
student_roster$movement <-ifelse(student_roster$movement %in% c('', "INPERSON_VIRTUAL" ,
"VIRTUAL_INPERSON"), "OTHER", student_roster$movement)
student_roster$movement <- as.factor(student_roster$movement)

# Choosing columns
student_roster <- student_roster [, (colnames(student_roster) %in%
  c('StuID',
    'ClaIDPre', 'TeaIDPre',
    'condition_assignment',
    'movement'))]

```

```{r merging data, include = FALSE}

#put all data frames into list
Assess_demo_roster <- list(assess_student, student_demo, student_roster)

#merge all data frames together
Assess_demo_roster <- Assess_demo_roster %>% reduce(full_join, by='StuID')
```

```{r cutting-dataset, include = FALSE}

# Create dataset with FH2T only
FH2T <- Assess_demo_roster %>% filter (condition_assignment == "FH2T", na.rm = TRUE)

```

Sample desctiptives

Correlation matrix

```{r cor matrix, echo=FALSE}

## Quantitative variables
FH2T_quant_with_action <-
  FH2T [, (colnames(FH2T) %in%
             c('PreMA', 'PreMP', 'PreMSE'
               ))]

# Creating matrix
apa.cor.table(
  data = FH2T_quant_with_action,
  filename = "Descriptives.doc",
  table.number = 1,
  show.conf.interval = TRUE,
  show.sig.stars = TRUE,
  landscape = TRUE
)

```

Sample demographics

```{r sample demographics, echo=FALSE}
```

```
FH2T$IEP <- as.factor(FH2T$IEP)
FH2T$EIP <- as.factor(FH2T$EIP)
FH2T$GIFTED <- as.factor(FH2T$GIFTED)
FH2T$ESOL <- as.factor(FH2T$ESOL)

# Table for RMarkdown
FH2T %>%
  dplyr::select(c('Gender', 'race_ethnicity', 'IEP', 'EIP', 'GIFTED', 'ESOL', 'PreMP',
'PreMA','PreMSE')) %>%
  tbl_summary()
```

Analysis

Clustering

```{r kmean packages, include = FALSE}

install.packages(c("factoextra", "cluster", "NbClust", "ggfortify", "RColorBrewer",
"corrplot"), repos="http://cran.us.r-project.org")

library(factoextra)
library(cluster)  # clustering algorithms and Silhouette Score
library(NbClust)  # clustering algorithms & visualization
library(ggfortify)
```

Choosing number of clusters

Dropping NAs and Z-scoring

```{r RQ1 z-scores, echo=TRUE}

### --- Delete cases with NAs in Pre Math anxiety or Pre Math performance
FH2T <- FH2T %>% drop_na(PreMA)
FH2T <- FH2T %>% drop_na(PreMP)

# Z-scoring MP and MA
FH2T$PreMP_z <-
  (FH2T$PreMP - mean(FH2T$PreMP))/sd(FH2T$PreMP)
FH2T$PreMA_z <-
  (FH2T$PreMA - mean(FH2T$PreMA))/sd(FH2T$PreMA)

# Creating new dataframes for PRE-levels clustering based on scaled variables
PRE_z <- FH2T %>% as.data.frame() %>%
  dplyr::select(PreMA_z, PreMP_z)
```

Elbow method

```{r RQ1 elbow method, echo=TRUE}

fviz_nbclust(PRE_z, kmeans, method = "wss") +
  geom_vline(xintercept = 4, linetype = 2)+
  labs(subtitle = "Elbow method")
```

Silhouette scores

```{r RQ1 choosing clusters with Silhouette score, echo=TRUE}
# Range of cluster numbers to test
silhouette_scores <- numeric(10)

# Loop through different numbers of clusters
for (k in 2:10) {
  set.seed(123)  # For reproducibility
  kmeans_result <- kmeans(PRE_z, centers = k)
  sil <- silhouette(kmeans_result$cluster, dist(PRE_z))
  silhouette_scores[k] <- mean(sil[, 3])  # Average Silhouette score for this k
}

# Find the number of clusters with the highest average Silhouette score
best_k <- which.max(silhouette_scores)
cat("The optimal number of clusters is", best_k, "with an average Silhouette score of",
silhouette_scores[best_k], "\n")

# Plot the Silhouette scores for each number of clusters
plot(2:10, silhouette_scores[2:10], type = "b",
     xlab = "Number of Clusters", ylab = "Average Silhouette Score",
     main = "Silhouette Score for Different Numbers of Clusters")

```

Clustering with 4 centers

```{r RQ1_standartization, include = TRUE}

### --- Applying k-means clustering
set.seed(20)
pre_cluster <- kmeans(PRE_z, centers = 4, nstart = 25) # put the optimal number of
clusters in "centers"
print(pre_cluster)

# Save the cluster number in the dataset as column 'cluster_results'
FH2T$pre_cluster_results <- as.factor(pre_cluster$cluster)

```

```{r RQ1 name clusters, include = FALSE}

## Saving clusters mean MP and MA values
FH2T <- FH2T %>%
 group_by(pre_cluster_results) %>%
  mutate(PreMP_mean = mean(PreMP),
            PreMP_sd = sd(PreMP),
            PreMA_mean = mean(PreMA),
            PreMA_sd = sd(PreMA)) %>%
  ungroup()

## Saving clusters names based on mean MP and MA values
# Put in MP levels
FH2T$pre_MP_group <-
  ifelse(FH2T$PreMP_mean<mean(c(FH2T$PreMP,FH2T$PostMP), na.rm=TRUE),
         "lMP", "hMP")
# Put in MA levels
FH2T$pre_MA_group <-
  ifelse(FH2T$PreMA_mean<mean(c(FH2T$PreMA,FH2T$PostMA), na.rm=TRUE),
         "lMA", "hMA")
# Combining MP and MA levels into one var
FH2T$pre_cluster_groups <-
  paste(FH2T$pre_MP_group, FH2T$pre_MA_group, sep="_")

## Saving clusters as factors with appropriate levels
FH2T$pre_cluster_groups <-
```

```
  factor(FH2T$pre_cluster_groups,
         levels = c("lMP_hMA", "lMP_lMA", "hMP_lMA", "hMP_hMA"))

## Calculating means in clusters to check if they are correct
FH2T %>%
 group_by(pre_cluster_groups) %>%
  summarise(PreMP_mean = mean(PreMP),
            PreMP_sd = sd(PreMP),
            PreMA_mean = mean(PreMA),
            PreMA_sd = sd(PreMA))

# To compare to the best group
FH2T$pre_cluster_groups_best <-
  factor(FH2T$pre_cluster_groups,
         levels = c("hMP_lMA", "hMP_hMA", "lMP_lMA", "lMP_hMA"))

```

Visualizing clusters

```{r RQ1 vis with centroids, echo=TRUE}
# Calculate centroids from your K-means result
centroids <- as.data.frame(pre_cluster$centers)

cluster_colors <- c("#E69F00", "#56B4E9", "#009E73", "#F0E442")

# Visualize the data with ggplot
library(ggplot2)
ggplot(FH2T, aes(PreMA_z, PreMP_z)) +
  geom_jitter(aes(color = factor(pre_cluster_groups))) +
  geom_point(data = centroids, aes(x = PreMA_z, y = PreMP_z),
             color = "black", size = 4, shape = 8) +  # Red stars for centroids
  scale_color_manual(values = cluster_colors) +
  labs(color = "Cluster", x = "Math Anxiety Score", y = "Math Test Score") +
  theme_minimal()

```

Clusters' demographics

```{r RQ1 clusters demographics, echo=FALSE}

FH2T$IEP <- as.factor(FH2T$IEP)
FH2T$EIP <- as.factor(FH2T$EIP)
FH2T$GIFTED <- as.factor(FH2T$GIFTED)
FH2T$ESOL <- as.factor(FH2T$ESOL)

# Table with all stats (does not knitted in RMarkdown)
descriptives <- DemographicTable(data=FH2T, groups = 'pre_cluster_groups', include =
c('Gender', 'race_ethnicity', 'IEP', 'EIP', 'GIFTED', 'ESOL','PreMP', 'PreMA','PreMSE'))

# Table for RMarkdown
FH2T %>%
  dplyr::select(c('Gender', 'race_ethnicity', 'IEP', 'EIP', 'GIFTED', 'ESOL', 'PreMP',
'PreMA','PreMSE', 'pre_cluster_groups')) %>%
  tbl_summary(by='pre_cluster_groups')

# Table with PreMP and PreMA means and sds
FH2T %>%
    group_by(pre_cluster_groups) %>%
    summarise_at( c('PreMP','PreMA'), c(mean = mean, sd = sd))

```

Comparison by MP

```{r RQ1 comparison of MP, echo=TRUE}

# Checking normality - normally distributed
FH2T %>%
  group_by(pre_cluster_groups) %>%
  summarise(shapiro_statistic = shapiro.test(PreMP)$statistic,
            p.value = shapiro.test(PreMP)$p.value)

# Checking homogeneity of variance - not normally distributed
leveneTest(PreMP ~ pre_cluster_groups, data = FH2T)
bartlett.test(PreMP ~ pre_cluster_groups, data = FH2T)

## MP comparison via Dunn test, as variances not normally distributed
dunn.test(FH2T$PreMP, g=FH2T$pre_cluster_groups, method='bonferroni')

```

Comparison by MA

```{r RQ1 comparison of MA, echo=TRUE}

# Checking normality - not normally distributed
FH2T %>%
  group_by(pre_cluster_groups) %>%
  summarise(shapiro_statistic = shapiro.test(PreMA)$statistic,
            p.value = shapiro.test(PreMA)$p.value)

# Checking homogeneity of variance - normally distributed
leveneTest(PreMA ~ pre_cluster_groups, data = FH2T)
bartlett.test(PreMA ~ pre_cluster_groups, data = FH2T)

## MA comparison via Dunn test, as data is not normally distributed
dunn.test(FH2T$PreMA, g=FH2T$pre_cluster_groups, method='bonferroni')

```

Vizualization of comparison by MP and MA (z-scored)

```{r RQ1 vis comparison of MP and MA, echo=TRUE}

## Visualization for both

# Creating long format table
FH2T_data_long <- pivot_longer(FH2T,
                               cols = c('PreMP_z', 'PreMA_z'),
                               names_to = 'Variable',
                               values_to = 'Value')

# Specify levels for factor "Variable" (so MP goes first on the visualization)
FH2T_data_long$Variable <- factor(FH2T_data_long$Variable , levels=c("PreMP_z",
"PreMA_z"))


# Create a boxplot for each variable with facets for clusters
ggplot(FH2T_data_long, aes(x = Variable , y = Value, fill = Variable)) +
  geom_boxplot() +
  labs(x = "Cluster", y = "Value") +
  facet_wrap(~ pre_cluster_groups_best, scales = "fixed") +
  scale_fill_manual(values = wes_colors) +
  theme_minimal()
```

MP distribution

```{r RQ1 MP distribution comparison, echo=FALSE}

# Create the ggplot2 density plot
p_PreMP <- ggplot(FH2T,
                  aes(x = PreMP, fill = pre_cluster_groups)) +
        geom_density(alpha = 0.8) +
        scale_fill_manual(values = cluster_colors) +
        theme_minimal()

# Convert the ggplot object to a plotly object
p_plotly_PreMP <- ggplotly(p_PreMP, tooltip = "fill")

# Make the plotly plot interactive such that hovering over the legend highlights the
specific category
p_plotly_PreMP %>%
  style(hoverinfo = "none", hoveron = "points", traces = c(1,2)) %>%
  layout(showlegend = TRUE)

```

MA distribution

```{r RQ1 MA distridution comparison, echo=FALSE}

# Create the ggplot2 density plot
p_PreMA <- ggplot(FH2T,
                  aes(x = PreMA, fill = pre_cluster_groups)) +
        geom_density(alpha = 0.8) +
        scale_fill_manual(values = cluster_colors) +
        theme_minimal()

# Convert the ggplot object to a plotly object
p_plotly_PreMA <- ggplotly(p_PreMA, tooltip = "fill")

# Make the plotly plot interactive such that hovering over the legend highlights the
specific category
p_plotly_PreMA %>%
  style(hoverinfo = "none", hoveron = "points", traces = c(1,2)) %>%
  layout(showlegend = TRUE)

```


Compare by MSE

Z-scoring, checking normality and comparing MSE

```{r RQ2 comparing MSE, echo=TRUE}

# Checking normality - not normally distributed
FH2T %>%
  group_by(pre_cluster_groups) %>%
  summarise(shapiro_statistic = shapiro.test(PreMSE)$statistic,
            p.value = shapiro.test(PreMSE)$p.value)

# Checking homogeneity of variance - not normally distributed
leveneTest(PreMSE ~ pre_cluster_groups, data = FH2T)
bartlett.test(PreMSE ~ pre_cluster_groups, data = FH2T)

## MA comparison via Dunn test, as data is not normally distributed
dunn.test(FH2T$PreMSE, g=FH2T$pre_cluster_groups, method='bonferroni')

```
Visualizing MSE distribution

```{r RQ2 MSE distribution comparison, echo=FALSE}

# Create the ggplot2 density plot
p_PreMSE <- ggplot(FH2T,
                aes(x = PreMSE, fill = pre_cluster_groups)) +
        geom_density(alpha = 0.8) +
        scale_fill_manual(values = cluster_colors) +
        theme_minimal()

# Convert the ggplot object to a plotly object
p_plotly_PreMSE <- ggplotly(p_PreMSE, tooltip = "fill")

# Make the plotly plot interactive such that hovering over the legend highlights the
specific category
p_plotly_PreMSE %>%
  style(hoverinfo = "none", hoveron = "points", traces = c(1,2)) %>%
  layout(showlegend = TRUE)

```
Visualizinf with MP and MA within groups

```{r RQ2 visualizing MSE, MP and MA in groups, echo=FALSE}

# Z-scoring MSE
FH2T$PreMSE_z <-
  (FH2T$PreMSE - mean(FH2T$PreMSE, na.rm=TRUE))/sd(FH2T$PreMSE, na.rm=TRUE)

# Creating long format table
FH2T_data_long <- pivot_longer(FH2T,
                               cols = c('PreMP_z', 'PreMA_z', 'PreMSE_z'),
                               names_to = 'Variable',
                               values_to = 'Value')

# Specify levels for factor "Variable" (so MP goes first on the viz)
FH2T_data_long$Variable <- factor(FH2T_data_long$Variable , levels=c("PreMP_z","PreMA_z",
"PreMSE_z"))

# Create a boxplot for each variable with facets for clusters
ggplot(FH2T_data_long, aes(x = Variable , y = Value, fill = Variable)) +
  geom_boxplot() +
  labs(x = "Cluster", y = "Value") +
  facet_wrap(~ pre_cluster_groups_best, scales = "fixed") +
  scale_fill_manual(values = wes_colors) +
  theme_minimal()
```
```{r RQ2 predicting MP with moderator of MSE, echo=FALSE}

# Rank the variables
PreMP_rank <- rank(FH2T$PreMP)
PreMA_rank <- rank(FH2T$PreMA)
PreMSE_rank <- rank(FH2T$PreMSE)

# Z-scoring ranked MP
PreMP_rank_z <-
  (PreMP_rank - mean(PreMP_rank))/sd(PreMP_rank)
# Z-scoring ranked MS
PreMA_rank_z <-
  (PreMA_rank - mean(PreMA_rank))/sd(PreMA_rank)
# Z-scoring ranked MSE
PreMSE_rank_z <-
  (PreMSE_rank - mean(PreMSE_rank))/sd(PreMSE_rank)

# Combine ranked variables into a data frame
ranked_data <- data.frame(PreMP_rank_z, PreMA_rank_z, PreMSE_rank_z)
```

```
## Predicting MP
# Fit the linear model with interaction
model_MA_ranked <- lm(PreMP_rank_z ~ PreMA_rank_z * PreMSE_rank_z, data = ranked_data)

# Display the summary of the model
summary(model_MA_ranked)

# Plot the interaction effect using the 'interactions' package
interact_plot(model_MA_ranked,
              pred = PreMA_rank_z,
              modx = PreMSE_rank_z,
              x.label = "Ranked MA",
              y.label = "Ranked MP",
              plot.gitter = TRUE,
              interval = TRUE)
```

```{r RQ2 predicting MA with moderator of MSE, echo=FALSE}

## Predicting MA
# Fit the linear model with interaction
model_MA_ranked <- lm(PreMA_rank_z ~ PreMP_rank_z * PreMSE_rank_z, data = ranked_data)

# Display the summary of the model
summary(model_MA_ranked)

# Plot the interaction effect using the 'interactions' package
interact_plot(model_MA_ranked,
              pred = PreMP_rank_z,
              modx = PreMSE_rank_z,
              x.label = "Ranked MP",
              y.label = "Ranked MA",
              plot.gitter = TRUE,
              interval = TRUE)

```