## SIMOA FINAL

\*\*\* Planned Analyses \*\*\*

All statistical analyses will be conducted using multiply imputed data to account for missingness across variables. Multiple imputation is used to reduce bias and improve statistical power by generating plausible values based on the observed data distribution. The imputation model includes all variables of interest, consistent with current best practices for handling missing data in psychological and health research.

Group Comparisons

To examine whether older adults who discontinued their use of benzodiazepine receptor agonists (BZRA) differ from those who continued use, a series of independent samples t-tests will be conducted. These analyses will compare the two groups on a range of psychological and personality variables, including:

BFI-10 (Big Five Inventory - 10-item short form),

SURPS (Substance Use Risk Profile Scale),

PHQ-2 (Patient Health Questionnaire – depression screener),

OSSS-3 (Oslo Social Support Scale),

DBAS-16 (Dysfunctional Beliefs and Attitudes about Sleep),

CISS-21 (Coping Inventory for Stressful Situations).

Additional between-group comparisons will be conducted on demographic and health-related variables (e.g., age, gender, sleep quality, comorbidities) to identify any systematic differences that may be associated with BZRA cessation status.

\*\*\* Predictive Modelling \*\*\*

To identify the most important variables associated with successful BZRA cessation, a Random Forest classification model will be employed. Random Forest is a non-parametric ensemble machine learning method that handles complex interactions and non-linear relationships, and is robust to multicollinearity and overfitting.

The Random Forest model will be trained using all personality, psychological, demographic, and health-related variables as predictors, with BZRA cessation (yes/no) as the outcome. Variable importance scores will be used to rank predictors based on their contribution to classification accuracy.

\*\*\* Model Confirmation \*\*\*

To validate the findings from the Random Forest model, a logistic regression analysis will be conducted using the top predictors identified by the Random Forest. This traditional regression model will allow for the estimation of effect sizes (odds ratios) and the statistical significance of each variable’s unique contribution to BZRA cessation. Confidence intervals and p-values will be reported, and model diagnostics will be used to assess model fit.

Together, this multi-step analytic strategy aims to both explore and confirm key predictors of BZRA discontinuation among older adults, leveraging the strengths of both machine learning and traditional inferential statistics.

## Data Loading, Screening and Packages

In this section I am loading all the necessary packages for my analysis and loading in the data.

```{r}

#| label: Data Loading and Packages

# Installing and Loading Packages

install.packages("MissMech")

install.packages("BaylorEdPsych")

library(dplyr)

library(ggplot2)

library(mice)

library(MissMech)

library(naniar)

library(readr)

library(stringr)

library(tidyverse)

library(VIM)

# Data Loading

SIMOA\_Report <- read\_csv("SIMOA Report.csv")

View(SIMOA\_Report)

```

Below I am creating a new object containing only the participants we can confirm are >= 65 and are using a BZRA by answering which specific BZRA they are using. This is to ensure they are not using other medications that may have sedative effects such as antihistamines or SSRI's.

```{r}

#| label: Creating new object

Dataset <- SIMOA\_Report %>%

# Apply filtering based on age\_cat and age

filter(

age\_cat == 1 | (age\_cat == 0 & age >= 65)

) %>%

# Apply filtering to keep only rows where any of c\_sp\_\_\_1 to c\_sp\_\_\_14 == 1

filter(

rowSums(select(., starts\_with("c\_sp\_\_\_"))[, 1:14] == 1, na.rm = TRUE) > 0

)

```

## Data Cleaning

Below is the section where I will separate the questions into the categories the participants truly saw to avoid higher values for missingness which would be incorrect.

```{r}

#| label: Making Sata Questions Into Factors

# I will begin changing all the 'Select all that apply" questions to factors with 1 indicating an answer was provided and 0 meaning that none of the options were selected by the participant (likely not shown the questions)

# Define all variable prefixes

prefixes <- c(

"liv\_sit\_\_\_", "sleep\_health\_con\_\_\_", "def\_insomnia\_\_\_", "phys\_health\_con\_\_\_",

"ment\_health\_con\_\_\_", "alt\_nic\_use\_\_\_", "can\_use\_rsn\_\_\_", "c\_sp\_\_\_",

"tablet\_split\_\_\_", "tablet\_split\_2\_\_\_", "stop\_wdl\_exp\_2\_\_\_", "stopping\_aids\_2\_\_\_", "other\_sub\_\_\_"

)

# Flatten into a single list of all relevant column names

all\_factor\_columns <- names(Dataset)[

sapply(names(Dataset), function(col) any(startsWith(col, prefixes)))

]

# Convert to factors: 1 = "Selected", 0 = "Not selected"

Dataset <- Dataset %>%

mutate(across(all\_of(all\_factor\_columns), ~ factor(., levels = c(0, 1), labels = c("Not selected", "Selected"))))

# Check that it worked

str(Dataset[all\_factor\_columns])

saveRDS(Dataset, "cleaned\_dataset.rds")

Dataset <- readRDS("cleaned\_dataset.rds")

attr(Dataset, "spec") <- NULL

################################################################################

################################################################################

#| label: Removing Columns With Only N/A

# This next section will remove the columns with only N/A values showing that they were not answered by anyone regardless of group and are not needed for our analysis.

# Identify columns with all NA

cols\_all\_na <- names(Dataset)[colSums(!is.na(Dataset)) == 0]

print(cols\_all\_na)

# Then remove them

Dataset <- Dataset[, colSums(!is.na(Dataset)) > 0]

################################################################################

################################################################################

#| label: Branching Logic Seperation for the Questions Based on Frequency of Use

# This section will group the questions into which ones were seen by those who answered "2" or using their BZRA more than 1 time per month to the "freq\_dich" question.

# === Load REDCap data dictionary ===

dict <- read.csv("Data\_Dict.csv", stringsAsFactors = FALSE)

# Rename the branching logic column for easier access

names(dict)[names(dict) == "Branching.Logic..Show.field.only.if...."] <- "branching\_logic"

# Normalize branching logic text to lowercase for consistent pattern matching

dict$branching\_logic <- tolower(dict$branching\_logic)

# === Extract questions based on branching logic ===

# 1. Questions shown only if freq\_dich = '2'

freq\_use\_questions <- dict$Variable...Field.Name[

grepl("\\[freq\_dich\\]\\s\*=\\s\*'2'", dict$branching\_logic)

]

# 2. Questions shown only if freq\_dich = '0' OR freq\_dich = '1'

not\_freq\_use\_questions <- dict$Variable...Field.Name[

grepl("\\[freq\_dich\\]\\s\*=\\s\*'0'|\\[freq\_dich\\]\\s\*=\\s\*'1'", dict$branching\_logic)

]

# 3. Questions shown to everyone (no branching logic or no freq\_dich condition)

common\_questions <- dict$Variable...Field.Name[

is.na(dict$branching\_logic) | dict$branching\_logic == "" | !grepl("\\[freq\_dich\\]", dict$branching\_logic)

]

# === OPTIONAL: Filter to variables actually present in your dataset ===

vars\_in\_dataset <- names(Dataset)

freq\_use\_questions <- freq\_use\_questions[freq\_use\_questions %in% vars\_in\_dataset]

not\_freq\_use\_questions <- not\_freq\_use\_questions[not\_freq\_use\_questions %in% vars\_in\_dataset]

common\_questions <- common\_questions[common\_questions %in% vars\_in\_dataset]

# === Check results ===

cat("Number of freq\_use\_questions:", length(freq\_use\_questions), "\n")

cat("Number of not\_freq\_use\_questions:", length(not\_freq\_use\_questions), "\n")

cat("Number of common\_questions:", length(common\_questions), "\n")

################################################################################

################################################################################

#| label: Branching Logic For All Other Questions

# I need to further the branching logic because men and women were shown separate questions so below is how I have gone about ensuring that missingness is only calculated for the questions participants could have been shown

# Number of participants

n <- nrow(Dataset)

# --- Create logical indicator for other\_sub\_use upfront ---

# Checkbox columns for other\_sub

checkbox\_cols <- paste0("other\_sub\_\_\_", 0:8)

checkbox\_cols <- checkbox\_cols[checkbox\_cols %in% colnames(Dataset)] # ensure columns exist

# Logical vector if "Other" checkbox (\_\_\_8) is selected

Dataset$other\_sub\_use <- FALSE

if ("other\_sub\_\_\_8" %in% colnames(Dataset)) {

Dataset$other\_sub\_use <- Dataset[["other\_sub\_\_\_8"]] == 1

Dataset$other\_sub\_use[is.na(Dataset$other\_sub\_use)] <- FALSE

}

# --- Function to convert REDCap branching logic to R expression string ---

convert\_branching\_logic <- function(expr) {

if (is.na(expr) || expr == "") return(NULL)

expr\_r <- expr %>%

tolower() %>%

# Explicitly replace [other\_sub\_use] first

str\_replace\_all("\\[other\_sub\_use\\]", "Dataset[['other\_sub\_use']]") %>%

# Replace [var(num)] with triple underscores

str\_replace\_all("\\[([a-z0-9\_]+)\\((\\d+)\\)\\]", "Dataset[['\\1\_\_\_\\2']]") %>%

# Replace [var]

str\_replace\_all("\\[([a-z0-9\_]+)\\]", "Dataset[['\\1']]") %>%

# Fix operators

str\_replace\_all("(?<![!<>=])=", "==") %>%

str\_replace\_all("!====", "!=") %>%

str\_replace\_all("'([0-9]+)'", "\\1") %>%

str\_replace\_all("\\band\\b", "&") %>%

str\_replace\_all("\\bor\\b", "|")

expr\_r

}

# --- Initialize list to store applicability vectors ---

applicability\_list <- vector("list", length = nrow(dict))

names(applicability\_list) <- dict$Variable...Field.Name

# --- Loop over all questions and evaluate branching logic ---

for (i in seq\_len(nrow(dict))) {

varname <- dict$Variable...Field.Name[i]

logic <- dict$branching\_logic[i]

if (is.na(logic) || logic == "") {

# No branching logic → all TRUE

applicability\_list[[varname]] <- rep(TRUE, n)

} else {

# Convert to R logical expression string

expr\_r <- convert\_branching\_logic(logic)

if (is.null(expr\_r)) {

applicability\_list[[varname]] <- rep(TRUE, n)

} else {

# Evaluate expression safely with improved handling

applicable\_vec <- tryCatch(

{

res <- eval(parse(text = expr\_r))

# Convert to logical vector if not already logical

if (!is.logical(res)) {

res <- !is.na(res) & res != 0 & res != ""

}

# Ensure length matches n

if (length(res) != n) {

warning(sprintf("Branching logic for %s returned vector length %d != %d", varname, length(res), n))

res <- rep(FALSE, n)

}

res

},

error = function(e) {

warning(sprintf("Failed to evaluate branching logic for %s: %s", varname, e$message))

rep(FALSE, n)

}

)

# Diagnostic output

cat(sprintf("Variable: %s, length: %d, class: %s\n", varname, length(applicable\_vec), class(applicable\_vec)))

# Final check for logical vector of correct length

if (!is.logical(applicable\_vec) || length(applicable\_vec) != n) {

warning(sprintf("Branching logic for %s did not return logical vector of length %d", varname, n))

applicable\_vec <- rep(FALSE, n)

}

applicability\_list[[varname]] <- applicable\_vec

}

}

}

# Convert list to data.frame or matrix for easy indexing

applicability\_matrix <- do.call(cbind, applicability\_list)

# Fix colnames for clarity

colnames(applicability\_matrix) <- names(applicability\_list)

## Description of the applicability matrix

# applicability\_matrix[i, j] = TRUE/FALSE if participant i should see question j

##### TESTING TO SEE IF THE CODE WORKED #####

#participants\_zero <- which(Dataset$scrn\_stopped\_bzra == "0")

## Subset responses for confirm\_stop\_attempt among these participants

#responses\_confirm <- Dataset$confirm\_stop\_attempt[participants\_zero]

## Check how many responded (not NA)

#num\_answered <- sum(!is.na(responses\_confirm))

#num\_missing <- sum(is.na(responses\_confirm))

#cat("Number of participants with scrn\_stopped\_bzra == '0':", length(participants\_zero), "\n")

#cat("Among them, how many answered confirm\_stop\_attempt:", num\_answered, "\n")

#cat("And how many did NOT answer (missing):", num\_missing, "\n")

##### EXTRA CODE THAT MAY BE USEFUL #####

## Example: Check how many participants should see each question

#applicability\_counts <- colSums(applicability\_matrix)

## Example: Filter questions that apply to at least some participants

#questions\_shown <- names(applicability\_counts)[applicability\_counts > 0]

```

## Missingness

Now that the branching logic has been successfully followed, I can carry on and check missingness and determine if my data is MCAR, MAR, or MNAR which will impact if Multiple Implication is a valuable tool or not. On top of this, if my missingness value is below 5%, it is generally accepted that MI is not needed.

```{r}

#| label: Missingness

# Initialize data frame to store results

missingness\_df <- data.frame(

question = colnames(applicability\_matrix),

n\_missing = integer(length(colnames(applicability\_matrix))),

percent\_missing = numeric(length(colnames(applicability\_matrix))),

n\_applicable = integer(length(colnames(applicability\_matrix))),

stringsAsFactors = FALSE

)

# Loop over each question

for (question in colnames(applicability\_matrix)) {

applicable <- applicability\_matrix[, question] # TRUE/FALSE per respondent

responses <- Dataset[[question]] # Actual values in dataset

n\_app <- sum(applicable, na.rm = TRUE)

if (!is.null(applicable) && n\_app > 0) {

responses\_applicable <- responses[applicable]

if (length(responses\_applicable) > 0) {

is\_missing <- is.na(responses\_applicable)

n\_miss <- sum(is\_missing)

pct\_miss <- if (n\_app > 0) (n\_miss / n\_app) \* 100 else 0

missingness\_df$n\_missing[missingness\_df$question == question] <- n\_miss

missingness\_df$n\_applicable[missingness\_df$question == question] <- n\_app

missingness\_df$percent\_missing[missingness\_df$question == question] <- pct\_miss

} else {

# No responses for applicable rows

missingness\_df$n\_missing[missingness\_df$question == question] <- 0

missingness\_df$n\_applicable[missingness\_df$question == question] <- 0

missingness\_df$percent\_missing[missingness\_df$question == question] <- 0

}

} else {

# Question never applied to anyone

missingness\_df$n\_missing[missingness\_df$question == question] <- 0

missingness\_df$n\_applicable[missingness\_df$question == question] <- 0

missingness\_df$percent\_missing[missingness\_df$question == question] <- 0

}

}

# Sort by percentage missing in descending order

missingness\_df <- missingness\_df[order(-missingness\_df$percent\_missing), ]

print(missingness\_df)

################################################################################

################################################################################

#| label: MCAR Analysis

# Here is the code I will use to examine if my data is MCAR, MAR, or MNAR to help me determine if MI is useful.