R Notebook: SOC 5800 Team Project

Malcolm S. Townes, Jacob Eikenberry, Daniel Ferris, Nicholas Sokolis

(November 30, 2019)

## Introduction

This is an R Notebook for a study that examines how formerly incarcerated individuals who successfully reintegrated cope with barriers to re-entry. This study was conducted as a team project for the SOC 5800 Survey Design and Sampling course during the Fall 2019 semester at Saint Louis University.

## Project Set Up

The following code chunk enables the R Notebook to integrate seamlessly with the project organization format. This is normally included in the R Notebook to simplify file calls and enable file portability but it has been causing an error. To work around this problem, I’ve embedded the here() function where I enter a file path when necessary.

knitr::opts\_knit$set(root.dir = here::here())

## Load Dependencies

The following code chunk loads package dependencies required to perform the necessary tasks. Basic tasks include importing, reading, wrangling, and cleaning data; selecting a subset of the data; checking for unique observations; analyzing missing data; and performing various types of regression analyses.

library(tidyverse) # loads the basic R packages  
library(here) # enables file portability  
library(readr) # functions for reading data  
library(dplyr) # functions for data wrangling  
library(janitor) # functions for data cleaning  
library(naniar) # functions for analyzing missing data  
library(expss) # functions for calculating on values  
library(ggplot2) # functions for data visualizations  
library(boot) # functions for regression analysis  
library(generalhoslem) # Hosmer-Lemeshow test for binary and multinomial logistic models  
library(ordinal) # functions for regression models for ordinal data  
library(MASS) # functions for ordered logistic or probit regression  
library(broom) # functions for tidying ordinal logistic regression models  
library(gvlma) # functions for global validation of linear model assumptions  
library(lmtest) # functions for testing linear regression models  
library(ltm) # functions for latent trait models under Item Response Theory  
library(leaps) # functions for regression subset selection  
library(car) # companion to applied regression  
library(aod) # functions to analyze overdispersed data counts and proportions  
library(pscl) # contains function for pseudo R2 measures for logistic regression  
library(ResourceSelection) # contains function for Hosmer-Lemeshow goodness of fit test  
library(psy) # functions for various procedures used in psychometry  
library(psych) # Procedures for Psychological, Psychometric, and Personality Research

## Load Raw Data

The following code chunk imports the raw data from the csv file.

dataRaw <- read.csv(here("Data","DataRaw","SOC5800\_Data\_NumericValues\_Raw\_CSV.csv"),   
 sep = ",", header = TRUE, fill = TRUE, dec = ".")

## Rename Variables

The following code chunk performs several actions to clean the raw data. It first renames the variables with descriptive camel case names. It then removes unused variables.

dataRaw %>%  
 dplyr::rename(startDate = StartDate,  
 endDate = EndDate,  
 status = Status,  
 IPaddress = IPAddress,  
 progress = Progress,  
 sessionDuration = Duration..in.seconds.,  
 surveyCompleted = Finished,  
 surveyDate = RecordedDate,  
 responseID = ResponseId,  
 locationLat = LocationLatitude,  
 locationLong = LocationLongitude,  
 consent = Q43,  
 browserName = Q10\_Browser,  
 browserVersion = Q10\_Version,  
 opSyst = Q10\_Operating.System,  
 screenRes = Q10\_Resolution,  
 currentlyIncarcerated = Q1,  
 currentlyDetained = Q2,  
 residentialTreatment = Q3,  
 involCommitment = Q4,  
 gender = Q5,  
 genderSelfDescribe = Q5\_3\_TEXT,  
 transgender = Q7,  
 ethnicityRace = Q6,  
 ethnicityRaceOther = Q6\_10\_TEXT,  
 typeHometown = Q8,  
 religiousAffiliation = Q9,  
 ReligiousOther = Q9\_12\_TEXT,  
 ageNow = Q11,  
 ageRelease = Q13,  
 incarcerationYears = Q14.1\_1,  
 incarcerationMonths = Q14.2\_1,  
 educationLevels = Q15,  
 whenHighestEd = Q16,  
 relationshipStatus = Q17,  
 householdSize = Q18,  
 financialSupportInitial = Q19,  
 financialSupportGovt = Q19\_4\_TEXT,  
 financialSupportNonGovt = Q19\_5\_TEXT,  
 finanicalSupportOther = Q19\_6\_TEXT,  
 selfEmployment = Q20,  
 incomeInitial = Q21,  
 incomeLastYr = Q22,  
 managingFinancially = Q23,  
 savings = Q24,  
 pssQ1 = Q41\_1,  
 pssQ2 = Q41\_2,  
 pssQ3 = Q41\_3,  
 pssQ4 = Q41\_4,  
 pssQ5 = Q41\_5,  
 pssQ6 = Q41\_6,  
 pssQ7 = Q41\_7,  
 pssQ8 = Q41\_8,  
 pssQ9 = Q41\_9,  
 pssQ10 = Q41\_10,  
 pssQ11 = Q41\_11,  
 pssQ12 = Q41\_12,  
 issFamilyQ1 = Q30\_1,  
 issFamilyQ2 = Q30\_2,  
 issFamilyQ3 = Q30\_3,  
 issFamilyQ4 = Q30\_4,  
 issFamilyQ5 = Q30\_5,  
 issFriendQ1 = Q31\_1,  
 issFriendQ2 = Q31\_2,  
 issFriendQ3 = Q31\_3,  
 issFriendQ4 = Q31\_4,  
 issFriendQ5 = Q31\_5,  
 programsUsed = Q32,  
 importancePublicTrans = Q33\_1,  
 importanceHousing = Q33\_2,  
 importanceSNAP = Q33\_3,  
 importanceWIC = Q33\_4,  
 importanceTANF = Q33\_5,  
 importanceEITC = Q33\_6,  
 importanceJobTraining = Q33\_7,  
 importanceMedicare = Q33\_8,  
 importanceEducAid = Q33\_9,  
 aceQ1 = Q34\_1,  
 aceQ2 = Q34\_2,  
 aceQ3 = Q34\_3,  
 aceQ4 = Q34\_4,  
 aceQ5 = Q34\_5,  
 aceQ6 = Q34\_6,  
 aceQ7 = Q34\_7,  
 aceQ8 = Q34\_8,  
 aceQ9 = Q34\_9,  
 aceQ10 = Q34\_10,  
 helpDuring = Q35,  
 helpDuringType = Q33,  
 helpDuringOther = Q33\_8\_TEXT,  
 helpAfter = Q36,  
 helpAfterType = Q44,  
 helpAfterOther = Q44\_8\_TEXT,  
 gritQ1 = Q35\_1,  
 gritQ2 = Q35\_2,  
 gritQ3 = Q35\_3,  
 gritQ4 = Q35\_4,  
 gritQ5 = Q35\_5,  
 gritQ6 = Q35\_6,  
 gritQ7 = Q35\_7,  
 gritQ8 = Q35\_8,  
 stayInitial = Q36.1,  
 stayInitialOther = Q36\_6\_TEXT,  
 foodSecurityQ1 = Q37\_1,  
 foodSecurityQ2 = Q37\_2,  
 mostHelpful = Q38,  
 greatestObstacle = Q39,  
 helpKind = Q40,  
 justiceInteraction = Q41  
 ) -> dataRenamed  
  
dataRenamed %>%  
 dplyr::select (-c(status, progress, RecipientLastName, RecipientFirstName, RecipientEmail, ExternalReference)) -> dataAugmented

## Change Data Type

The following code chunk changes the data type for certain variables.

dataAugmented$pssQ1 <- as.numeric(dataAugmented$pssQ1)  
dataAugmented$pssQ2 <- as.numeric(dataAugmented$pssQ2)  
dataAugmented$pssQ3 <- as.numeric(dataAugmented$pssQ3)  
dataAugmented$pssQ4 <- as.numeric(dataAugmented$pssQ4)  
dataAugmented$pssQ5 <- as.numeric(dataAugmented$pssQ5)  
dataAugmented$pssQ6 <- as.numeric(dataAugmented$pssQ6)  
dataAugmented$pssQ7 <- as.numeric(dataAugmented$pssQ7)  
dataAugmented$pssQ8 <- as.numeric(dataAugmented$pssQ8)  
dataAugmented$pssQ9 <- as.numeric(dataAugmented$pssQ9)  
dataAugmented$pssQ10 <- as.numeric(dataAugmented$pssQ10)  
dataAugmented$pssQ11 <- as.numeric(dataAugmented$pssQ11)  
dataAugmented$pssQ12 <- as.numeric(dataAugmented$pssQ12)  
dataAugmented$issFamilyQ1 <- as.numeric(dataAugmented$issFamilyQ1)  
dataAugmented$issFamilyQ2 <- as.numeric(dataAugmented$issFamilyQ2)  
dataAugmented$issFamilyQ3 <- as.numeric(dataAugmented$issFamilyQ3)  
dataAugmented$issFamilyQ4 <- as.numeric(dataAugmented$issFamilyQ4)  
dataAugmented$issFamilyQ5 <- as.numeric(dataAugmented$issFamilyQ5)  
dataAugmented$issFriendQ1 <- as.numeric(dataAugmented$issFriendQ1)  
dataAugmented$issFriendQ2 <- as.numeric(dataAugmented$issFriendQ2)  
dataAugmented$issFriendQ3 <- as.numeric(dataAugmented$issFriendQ3)  
dataAugmented$issFriendQ4 <- as.numeric(dataAugmented$issFriendQ4)  
dataAugmented$issFriendQ5 <- as.numeric(dataAugmented$issFriendQ5)  
dataAugmented$aceQ1 <- as.numeric(dataAugmented$aceQ1)  
dataAugmented$aceQ2 <- as.numeric(dataAugmented$aceQ2)  
dataAugmented$aceQ3 <- as.numeric(dataAugmented$aceQ3)  
dataAugmented$aceQ4 <- as.numeric(dataAugmented$aceQ4)  
dataAugmented$aceQ5 <- as.numeric(dataAugmented$aceQ5)  
dataAugmented$aceQ6 <- as.numeric(dataAugmented$aceQ6)  
dataAugmented$aceQ7 <- as.numeric(dataAugmented$aceQ7)  
dataAugmented$aceQ8 <- as.numeric(dataAugmented$aceQ8)  
dataAugmented$aceQ9 <- as.numeric(dataAugmented$aceQ9)  
dataAugmented$aceQ10 <- as.numeric(dataAugmented$aceQ10)  
dataAugmented$gritQ1 <- as.numeric(dataAugmented$gritQ1)  
dataAugmented$gritQ2 <- as.numeric(dataAugmented$gritQ2)  
dataAugmented$gritQ3 <- as.numeric(dataAugmented$gritQ3)  
dataAugmented$gritQ4 <- as.numeric(dataAugmented$gritQ4)  
dataAugmented$gritQ5 <- as.numeric(dataAugmented$gritQ5)  
dataAugmented$gritQ6 <- as.numeric(dataAugmented$gritQ6)  
dataAugmented$gritQ7 <- as.numeric(dataAugmented$gritQ7)  
dataAugmented$gritQ8 <- as.numeric(dataAugmented$gritQ8)

## Adjust Value Assignments

The following code chunk adjusts the values assigned to fixed pre-determined response choices for select variables and recodes blanks with NA in the dataset.

dataAugmented$ageNow <- (as.numeric(dataAugmented$ageNow)+11)  
dataAugmented$ageRelease <- (as.numeric(dataAugmented$ageRelease)+11)  
  
# Recode variables with `yes` and `no` response options as 1 and 0 respectively  
dataAugmented %>%  
 mutate(savings = ifelse (savings == 1,1,0)) %>%  
 mutate(helpDuring = ifelse(helpDuring == 1,1,0)) %>%  
 mutate(helpAfter = ifelse(helpAfter == 1,1,0)) -> dataAugmented  
# mutate(savings = case\_when(  
# as.numeric(savings) == 1 ~ 1,  
# as.numeric(savings) == 2 ~ 0,  
# TRUE ~ savings)  
# )%>%  
# mutate(helpDuring = case\_when(  
# as.numeric(helpDuring) == 1 ~ 1,  
# as.numeric(helpDuring) == 2 ~ 0,  
# TRUE ~ helpDuring)  
# )%>%  
# mutate(helpAfter = case\_when(  
# as.numeric(helpDuring) == 1 ~ 1,  
# as.numeric(helpDuring) == 2 ~ 0,  
# TRUE ~ helpAfter)  
# )-> dataAugmented  
  
# Recode instances of missing data with `NA`  
dataAugmented[dataAugmented==""] <- NA

## Create Variables

The following code chunk creates additional variables needed for the study.

dataAugmented %>%  
 # Calculate score for multidimensional scale for percieved social support  
 mutate (mspssScore = case\_when(  
 !is.na(pssQ1)|!is.na(pssQ2)|!is.na(pssQ3)|!is.na(pssQ4)|!is.na(pssQ5)|!is.na(pssQ6)|!is.na(pssQ7)|  
 !is.na(pssQ8)|!is.na(pssQ9)|!is.na(pssQ10)|!is.na(pssQ11)|!is.na(pssQ12) ~  
 (pssQ1+pssQ2+pssQ3+pssQ4+pssQ5+pssQ6+pssQ7+pssQ8+pssQ9+pssQ10+pssQ11+pssQ12)/12)  
 )%>%  
   
 # Calculate score for family instrumental social support  
 mutate (issFamilyScore = case\_when(  
 !is.na(issFamilyQ1)|!is.na(issFamilyQ2)|!is.na(issFamilyQ3)|!is.na(issFamilyQ4)|!is.na(issFamilyQ5)  
 ~ (issFamilyQ1+issFamilyQ2+issFamilyQ3+issFamilyQ4+issFamilyQ5)/5)  
 )%>%  
   
 # Calculate score for friend instrumental social support  
 mutate (issFriendScore = case\_when(  
 !is.na(issFriendQ1)|!is.na(issFriendQ2)|!is.na(issFriendQ3)|!is.na(issFriendQ4)|!is.na(issFriendQ5)  
 ~ (issFriendQ1+issFriendQ2+issFriendQ3+issFriendQ4+issFriendQ5)/5)  
 )%>%  
   
 # Calculate score for total instrumental social support  
 mutate(issTotalScore = case\_when(  
 !is.na(issFriendScore)|!is.na(issFamilyScore) ~ (issFamilyScore+issFriendScore)/2)  
 )%>%  
   
 # Calculate score for adverse childhood experience  
 mutate (aceScore = case\_when (  
 !is.na(aceQ1)|!is.na(aceQ2)|!is.na(aceQ3)|!is.na(aceQ4)|!is.na(aceQ5)|!is.na(aceQ6)|!is.na(aceQ7)|  
 !is.na(aceQ8)|!is.na(aceQ9)|!is.na(aceQ10) ~   
 (aceQ1+aceQ1+aceQ2+aceQ3+aceQ4+aceQ5+aceQ6+aceQ7+aceQ8+aceQ9+aceQ10))  
 ) %>%  
   
 # Calculate socre for grit  
 mutate (gritQ1R = 6-gritQ1, gritQ3R = 6-gritQ3, gritQ5R = 6-gritQ5, gritQ6R = 6-gritQ6) %>%  
 mutate (gritScore = case\_when(  
 !is.na(gritQ1R)|!is.na(gritQ2)|!is.na(gritQ2)|!is.na(gritQ2)|!is.na(gritQ2)|  
 !is.na(gritQ2)|!is.na(gritQ2)|!is.na(gritQ2) ~   
 ((gritQ1R+gritQ2+gritQ3R+gritQ4+gritQ5R+gritQ6R+gritQ7+gritQ8)/8)  
 ))%>%  
   
 # Calculate total incarceration time in months  
 mutate (incarcerationTime = ((incarcerationYears-1)\*12)+(incarcerationMonths-1)) %>%  
   
 # Calculate total time released  
 mutate (timeReleased = ageNow - ageRelease) %>%  
   
 # Calculate interval variable for poverty ratio  
 mutate (povertyRatio = if(as.numeric(incomeLastYr) <= 6) {  
 (12490+((as.numeric(incomeLastYr)-1))\*4420)/(8070+(as.numeric(householdSize)\*4420))  
 } else {  
 if(as.numeric(incomeLastYr) == 7) {  
 (59590/(8070+(as.numeric(householdSize)\*4420)))  
 } else {  
 if(as.numeric(incomeLastYr) == 8) {  
 (62500/(8070+(as.numeric(householdSize)\*4420)))  
 } else {  
 if(as.numeric(incomeLastYr) == 9) {  
 (87500/(8070+(as.numeric(householdSize)\*4420)))  
 } else {  
 if(as.numeric(incomeLastYr) == 10) {  
 (125000/(8070+(as.numeric(householdSize\*4420))))  
 } else {  
 (150000/(8070+(as.numeric(householdSize)\*4420)))  
 }  
 }  
 }  
 }  
 }  
 ) %>%  
   
 # Calculate change in income  
 mutate(incomeChange = incomeLastYr - incomeInitial) %>%  
   
 # Determine highest education level  
 mutate(educationHighest = sapply(strsplit(as.character(educationLevels), split = ","),   
 function(x) max(as.numeric(x)))) %>%  
   
 # Calculate binary variable indicating food insecurity  
 mutate(foodInsecurity = case\_when(  
 foodSecurityQ1 == 2|foodSecurityQ1 == 3|foodSecurityQ2 == 2|foodSecurityQ2 == 3 ~ 1,  
 foodSecurityQ1 == 1 & foodSecurityQ2 == 1 ~ 0)  
 )%>%  
   
 # Calculate binary variable indicating use of any social services  
 mutate (programsUsedBinary = case\_when(  
 is.na(programsUsed) ~ 0,  
 !is.na(programsUsed) ~ 1)  
 ) %>%  
   
 # Determine number of public and social services used after release  
 mutate(countServicesAfter = count.fields(textConnection(as.character(programsUsed)), sep = ",")) %>%  
   
 # Determine number of social services used during incarceration  
 mutate(countHelpDuring = count.fields(textConnection(as.character(helpDuringType)), sep = ",")) %>%  
   
 # Determine number of social services used after release  
 mutate(countHelpAfter = count.fields(textConnection(as.character(helpAfterType)), sep = ",")) -> dataAugmented

## Select Cases

The following code chunk removes cases where the respondent was part of a protected population or did not provide informed consent. The original data for the variables was not recoded. It then removes duplicate cases.

dataAugmented %>%  
 subset(currentlyIncarcerated=2) %>%  
 subset(currentlyDetained=2) %>%  
 subset(residentialTreatment=2) %>%  
 subset(involCommitment=2) %>%  
 dplyr::distinct(.,.keep\_all = TRUE) -> dataClean

## Evaluate Missing Data

The following code chunk evaluates missing data.

miss\_var\_summary(dataClean, order = TRUE)

## # A tibble: 129 x 3  
## variable n\_miss pct\_miss  
## <chr> <int> <dbl>  
## 1 genderSelfDescribe 59 100   
## 2 ethnicityRaceOther 56 94.9  
## 3 financialSupportNonGovt 54 91.5  
## 4 selfEmployment 54 91.5  
## 5 helpAfterOther 53 89.8  
## 6 finanicalSupportOther 51 86.4  
## 7 helpDuringOther 51 86.4  
## 8 ReligiousOther 49 83.1  
## 9 stayInitialOther 47 79.7  
## 10 financialSupportGovt 39 66.1  
## # ... with 119 more rows

miss\_case\_summary(dataClean, order = TRUE)

## # A tibble: 59 x 3  
## case n\_miss pct\_miss  
## <int> <int> <dbl>  
## 1 49 107 82.9  
## 2 50 107 82.9  
## 3 53 90 69.8  
## 4 56 89 69.0  
## 5 51 84 65.1  
## 6 58 36 27.9  
## 7 57 23 17.8  
## 8 16 22 17.1  
## 9 52 20 15.5  
## 10 55 18 14.0  
## # ... with 49 more rows

## Calculate Descriptive Statistics

The following code calculates descriptive statistics for select variables of interest.

print("Number of cases")

## [1] "Number of cases"

nrow(dataClean)

## [1] 59

print("gender; 1 - male, 2 - female, 3 - other, 4 - undisclosed")

## [1] "gender; 1 - male, 2 - female, 3 - other, 4 - undisclosed"

print("values")

## [1] "values"

table(as.numeric(dataClean$gender))

##   
## 1 2   
## 31 26

print("proportions")

## [1] "proportions"

prop.table(table(as.numeric(dataClean$gender)), margin = NULL)

##   
## 1 2   
## 0.5438596 0.4561404

print("ethnicityRace; 1 - East Asian, 2 - Central Asian, 3 - Southern Asian, 4 - Black or African-American,  
 5 - Hispanic or Latino, 6 - Middle Eastern or North African, 7 - Native American or Alaska,  
 8 - Native Hawaiian or Pacific Islander, 9 - White or European, 10 - Other")

## [1] "ethnicityRace; 1 - East Asian, 2 - Central Asian, 3 - Southern Asian, 4 - Black or African-American,\n 5 - Hispanic or Latino, 6 - Middle Eastern or North African, 7 - Native American or Alaska,\n 8 - Native Hawaiian or Pacific Islander, 9 - White or European, 10 - Other"

print("values")

## [1] "values"

table(as.numeric(dataClean$ethnicityRace))

##   
## 3 4 5 7 9 10   
## 1 13 4 2 34 3

print("proportions")

## [1] "proportions"

prop.table(table(as.numeric(dataClean$ethnicityRace)), margin = NULL)

##   
## 3 4 5 7 9 10   
## 0.01754386 0.22807018 0.07017544 0.03508772 0.59649123 0.05263158

print("religiousAffiliation; 1 - Atheism, 2 - Agnoticism, 3 - Buddhist, 4 - Eastern Orthodox, 5 - Hindu,  
 6 - Islam, 7 - Jewish, 8 - Latter Day Saints, 9 - Non-denominational, 10 - Protestant,  
 11 - Roman Catholic, 12 - Other, 13 - prefer not to answer")

## [1] "religiousAffiliation; 1 - Atheism, 2 - Agnoticism, 3 - Buddhist, 4 - Eastern Orthodox, 5 - Hindu,\n 6 - Islam, 7 - Jewish, 8 - Latter Day Saints, 9 - Non-denominational, 10 - Protestant,\n 11 - Roman Catholic, 12 - Other, 13 - prefer not to answer"

print("values")

## [1] "values"

table(as.numeric(dataClean$religousAffiliation))

## < table of extent 0 >

print("proportions")

## [1] "proportions"

prop.table(table(as.numeric(dataClean$religiousAffiliation)), margin = NULL)

##   
## 1 2 7 8 9 10   
## 0.03508772 0.12280702 0.03508772 0.01754386 0.26315789 0.08771930   
## 11 12 13   
## 0.10526316 0.22807018 0.10526316

print("educationLevels; 1 - high schoor or GED, 2 - trade school, 3 - some college, 4 - associate,  
 5 - bachelor, 6 - master, 7 - doctorate")

## [1] "educationLevels; 1 - high schoor or GED, 2 - trade school, 3 - some college, 4 - associate,\n 5 - bachelor, 6 - master, 7 - doctorate"

print("values")

## [1] "values"

table(as.numeric(dataClean$educationLevels))

##   
## 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17   
## 1 1 1 2 2 2 1 2 1 1 12 2 1 12 9 7

print("proportions")

## [1] "proportions"

prop.table(table(as.numeric(dataClean$educationLevels)), margin = NULL)

##   
## 2 3 4 5 6 7   
## 0.01754386 0.01754386 0.01754386 0.03508772 0.03508772 0.03508772   
## 8 9 10 11 12 13   
## 0.01754386 0.03508772 0.01754386 0.01754386 0.21052632 0.03508772   
## 14 15 16 17   
## 0.01754386 0.21052632 0.15789474 0.12280702

print("whenHighestEd; 1 - before, 2 - during, 3 - after")

## [1] "whenHighestEd; 1 - before, 2 - during, 3 - after"

print("values")

## [1] "values"

table(as.numeric(dataClean$whenHighestEd))

##   
## 1 2 3   
## 10 8 39

print("proportions")

## [1] "proportions"

prop.table(table(as.numeric(dataClean$whenHighestEd)), margin = NULL)

##   
## 1 2 3   
## 0.1754386 0.1403509 0.6842105

print("typeHometown; 1 - rural, 2 - suburban, 3 - urban")

## [1] "typeHometown; 1 - rural, 2 - suburban, 3 - urban"

print("values")

## [1] "values"

table(as.numeric(dataClean$typeHometown))

##   
## 1 2 3   
## 1 21 34

print("proportions")

## [1] "proportions"

prop.table(table(as.numeric(dataClean$typeHometown)), margin = NULL)

##   
## 1 2 3   
## 0.01785714 0.37500000 0.60714286

print("ageNow; ratio data (years)")

## [1] "ageNow; ratio data (years)"

describe(as.numeric(dataClean$ageNow), na.rm = TRUE)

## vars n mean sd median trimmed mad min max range skew kurtosis se  
## X1 1 57 43.65 9.08 44 43.19 8.9 25 70 45 0.64 0.69 1.2

print("ageRelease; ratio data (years)")

## [1] "ageRelease; ratio data (years)"

describe(as.numeric(dataClean$ageRelease), na.rm = TRUE)

## vars n mean sd median trimmed mad min max range skew kurtosis  
## X1 1 54 33.54 10.23 33 32.95 9.64 15 63 48 0.44 -0.15  
## se  
## X1 1.39

print("incarcerationTime; ratio data (months)")

## [1] "incarcerationTime; ratio data (months)"

summary(as.numeric(dataClean$incarcerationTime))

## Min. 1st Qu. Median Mean 3rd Qu. Max. NA's   
## 10.00 27.00 59.50 90.79 111.00 367.00 25

print("incomeInitial; ordinal data")

## [1] "incomeInitial; ordinal data"

print("values")

## [1] "values"

table(as.numeric(dataClean$incomeInitial))

##   
## 1 2 3 4 5 6 7 8 10 11 12   
## 23 8 7 2 3 3 3 2 1 1 2

print("proportions")

## [1] "proportions"

prop.table(table(as.numeric(dataClean$incomeInitial)), margin = NULL)

##   
## 1 2 3 4 5 6   
## 0.41818182 0.14545455 0.12727273 0.03636364 0.05454545 0.05454545   
## 7 8 10 11 12   
## 0.05454545 0.03636364 0.01818182 0.01818182 0.03636364

print("incomeLastYr; ordinal data")

## [1] "incomeLastYr; ordinal data"

print("values")

## [1] "values"

table(as.numeric(dataClean$incomeLastYr))

##   
## 1 2 3 4 5 6 7 8 9 10 11   
## 13 1 3 7 3 4 8 7 3 4 2

print("proportions")

## [1] "proportions"

prop.table(table(as.numeric(dataClean$incomeLastYr)), margin = NULL)

##   
## 1 2 3 4 5 6   
## 0.23636364 0.01818182 0.05454545 0.12727273 0.05454545 0.07272727   
## 7 8 9 10 11   
## 0.14545455 0.12727273 0.05454545 0.07272727 0.03636364

print("selfEmployment; 1 - 0%, 2 - 1-25%, 3 - 26-50%, 4 - 51-75%, 5 - 75-99%, 6 - 100%")

## [1] "selfEmployment; 1 - 0%, 2 - 1-25%, 3 - 26-50%, 4 - 51-75%, 5 - 75-99%, 6 - 100%"

print("values")

## [1] "values"

table(as.numeric(dataClean$selfEmployment))

##   
## 1 3 4 6   
## 1 1 1 2

print("proportions")

## [1] "proportions"

prop.table(table(as.numeric(dataClean$selfEmployment)), margin = NULL)

##   
## 1 3 4 6   
## 0.2 0.2 0.2 0.4

print("foodInsecurity; 1 - yes, 0 - no")

## [1] "foodInsecurity; 1 - yes, 0 - no"

print("values")

## [1] "values"

table(as.numeric(dataClean$foodInsecurity))

##   
## 0 1   
## 31 21

print("proportions")

## [1] "proportions"

prop.table(table(as.numeric(dataClean$foodInsecurity)), margin = NULL)

##   
## 0 1   
## 0.5961538 0.4038462

## Compute Scale Reliabilities

The following code chunk evaluates the reliability for the MSPSS, ACE, and grit scales by calculating the Cronbach’s alpha for each variable.

reliabilityMSPSS <- cronbach (subset(dataClean, select=c(pssQ1, pssQ2, pssQ3, pssQ4, pssQ5, pssQ6, pssQ7, pssQ8, pssQ9, pssQ10, pssQ11, pssQ12)))  
print("Reliability for MSPSS")

## [1] "Reliability for MSPSS"

reliabilityMSPSS

## $sample.size  
## [1] 52  
##   
## $number.of.items  
## [1] 12  
##   
## $alpha  
## [1] 0.9231374

reliabilityISSfamily <- cronbach (subset(dataClean, select=c(issFamilyQ1, issFamilyQ2, issFamilyQ3, issFamilyQ4, issFamilyQ5)))  
print("Reliability for Family Instrumental Social Support")

## [1] "Reliability for Family Instrumental Social Support"

reliabilityISSfamily

## $sample.size  
## [1] 53  
##   
## $number.of.items  
## [1] 5  
##   
## $alpha  
## [1] 0.8265399

reliabilityISSfriend <- cronbach (subset(dataClean, select=c(issFriendQ1, issFriendQ2, issFriendQ3, issFriendQ4, issFriendQ5)))  
print("Reliability for Family Instrumental Social Support")

## [1] "Reliability for Family Instrumental Social Support"

reliabilityISSfriend

## $sample.size  
## [1] 51  
##   
## $number.of.items  
## [1] 5  
##   
## $alpha  
## [1] 0.9320187

reliabilityACE <- cronbach (subset(dataClean, select=c(aceQ1, aceQ2, aceQ3, aceQ4, aceQ5, aceQ6, aceQ7, aceQ8, aceQ9, aceQ10)))  
print("Reliability for ACE Scale")

## [1] "Reliability for ACE Scale"

reliabilityACE

## $sample.size  
## [1] 51  
##   
## $number.of.items  
## [1] 10  
##   
## $alpha  
## [1] 0.6815232

reliabilityGrit <- cronbach (subset(dataClean, select=c(gritQ1R, gritQ2, gritQ3R, gritQ4, gritQ5R, gritQ6R, gritQ7, gritQ8)))  
print("Reliability for Short Grit Scale")

## [1] "Reliability for Short Grit Scale"

reliabilityGrit

## $sample.size  
## [1] 53  
##   
## $number.of.items  
## [1] 8  
##   
## $alpha  
## [1] 0.7431265

reliabilityHungerVitalSigns <- cronbach (subset(dataClean, select=c(foodSecurityQ1,foodSecurityQ2)))  
print("Reliability for Hunger Vital Signs")

## [1] "Reliability for Hunger Vital Signs"

reliabilityHungerVitalSigns

## $sample.size  
## [1] 51  
##   
## $number.of.items  
## [1] 2  
##   
## $alpha  
## [1] 0.9463585

## Perform Comparison of Means of Grit

The following code chunk performs calculations for comparison of means of the primary variables of interest.

# Comparison of means for `gritScore` grouped by various operationalizations of success using ANOVA  
# Coding for `gritscore`: values range from 1 (low grit) to 8 (high grit)  
# Null hypothesis: the means of the different groups are the same  
# Alternative hypothesis: the sample mean of at least one group is not equal to the others  
  
  
# Comparison of `gritScore` for cases grouped by `incomeLastYr` using ANOVA  
# Coding for `incomeLastYr` (Qualitrics Q22): ordinal ranging 1 - 10  
groupIncome <- group\_by(dataClean, incomeLastYr)  
  
groupIncome %>%  
 summarise(  
 count = n(),  
 mean = mean(gritScore, na.rm = TRUE),  
 sd = sd(gritScore, na.rm = TRUE)  
 )

## # A tibble: 12 x 4  
## incomeLastYr count mean sd  
## <int> <int> <dbl> <dbl>  
## 1 1 13 3.79 0.596  
## 2 2 1 3.62 NaN   
## 3 3 3 4 0.760  
## 4 4 7 3.30 0.710  
## 5 5 3 3.38 0.331  
## 6 6 4 4.41 0.413  
## 7 7 8 3.46 0.425  
## 8 8 7 3.90 0.713  
## 9 9 3 4.29 0.191  
## 10 10 4 4 0.941  
## 11 11 2 2.75 0.884  
## 12 NA 4 NaN NaN

aovGroupIncome <- aov(gritScore ~ as.numeric(incomeLastYr), data = groupIncome)  
summary(aovGroupIncome)

## Df Sum Sq Mean Sq F value Pr(>F)  
## as.numeric(incomeLastYr) 1 0.002 0.0018 0.004 0.951  
## Residuals 51 23.886 0.4683   
## 6 observations deleted due to missingness

# Comparison of `gritScore` for cases grouped by `povertyLevel` using ANOVA  
# Coding for `povertyLevel` (created variable): 1 - above, 0 - at or below  
dataClean %>%  
 mutate(povertyLevel = if (povertyRatio >1) {1} else {0})-> dataClean  
  
groupPoverty <- group\_by(dataClean, povertyLevel)  
  
groupPoverty %>%  
 summarise(  
 count = n(),  
 mean = mean(gritScore, na.rm = TRUE),  
 sd = sd(gritScore, na.rm = TRUE)  
 )

## # A tibble: 1 x 4  
## povertyLevel count mean sd  
## <dbl> <int> <dbl> <dbl>  
## 1 0 59 3.73 0.678

aovGroupPoverty <- aov(gritScore ~ as.numeric(povertyLevel), data = groupPoverty)  
summary(aovGroupPoverty)

## Df Sum Sq Mean Sq F value Pr(>F)  
## Residuals 52 23.89 0.4594   
## 6 observations deleted due to missingness

# Comparison of `gritScore` for cases grouped by `managingFinancially` using ANOVA  
# Coding for `managingFinancially` (Qualitrics Q23): 1 - not able, 2 - just able, 3 - comfortable no savings, 4 - comfortable and saving  
groupFinStat <- group\_by(dataClean, managingFinancially)  
  
groupFinStat %>%  
 summarise(  
 count = n(),  
 mean = mean(gritScore, na.rm = TRUE),  
 sd = sd(gritScore, na.rm = TRUE)  
 )

## # A tibble: 5 x 4  
## managingFinancially count mean sd  
## <int> <int> <dbl> <dbl>  
## 1 1 12 3.70 0.631   
## 2 2 19 3.54 0.723   
## 3 3 15 3.62 0.604   
## 4 4 7 4.34 0.519   
## 5 NA 6 4.31 0.0884

aovGroupFinStat <- aov(gritScore ~ as.numeric(managingFinancially), data = groupFinStat)  
summary(aovGroupFinStat)

## Df Sum Sq Mean Sq F value Pr(>F)  
## as.numeric(managingFinancially) 1 1.255 1.2553 2.806 0.1  
## Residuals 49 21.922 0.4474   
## 8 observations deleted due to missingness

# Comparison of `gritScore` for cases grouped by `savings` using ANOVA  
# Coding for `savings` (Qualitrics Q24): 1 - yes, 2 - no  
groupSavings <- group\_by(dataClean, savings)  
  
groupSavings %>%  
 summarise(  
 count = n(),  
 mean = mean(gritScore, na.rm = TRUE),  
 sd = sd(gritScore, na.rm = TRUE)  
 )

## # A tibble: 3 x 4  
## savings count mean sd  
## <dbl> <int> <dbl> <dbl>  
## 1 0 40 3.56 0.648  
## 2 1 15 4.18 0.551  
## 3 NA 4 NaN NaN

aovGroupSavings <- aov(gritScore ~ as.numeric(savings), data = groupSavings)  
summary(aovGroupSavings)

## Df Sum Sq Mean Sq F value Pr(>F)   
## as.numeric(savings) 1 4.122 4.122 10.63 0.00198 \*\*  
## Residuals 51 19.766 0.388   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
## 6 observations deleted due to missingness

## Comparison of Means for Categorical Variables

The following code chunk performs comparison of means for select categorical variables of interest.

# Null hypothesis: The two variables are independent  
# Alternative hypothesis: The two variables relate to each other  
  
print("`typeHometown` and `foodInsecurity`")

## [1] "`typeHometown` and `foodInsecurity`"

ChiSqr01 <- chisq.test(dataClean$typeHometown, dataClean$foodInsecurity)  
ChiSqr01

##   
## Pearson's Chi-squared test  
##   
## data: dataClean$typeHometown and dataClean$foodInsecurity  
## X-squared = 0.71504, df = 2, p-value = 0.6994

print("`whenHighestEd` and `incomeLastYr`")

## [1] "`whenHighestEd` and `incomeLastYr`"

ChiSqr02 <- chisq.test(dataClean$whenHighestEd, dataClean$incomeLastYr)  
ChiSqr02

##   
## Pearson's Chi-squared test  
##   
## data: dataClean$whenHighestEd and dataClean$incomeLastYr  
## X-squared = 10.882, df = 20, p-value = 0.9492

## Perform Correlational Analysis

The following code chunk performs correlational analysis for the primary variables of interest.

dataClean %>%  
 dplyr::select(incomeLastYr, incomeChange, povertyRatio, managingFinancially, savings, programsUsedBinary,  
 countServicesAfter, countHelpDuring, countHelpAfter, educationHighest, ageNow, ageRelease,  
 incarcerationTime, timeReleased, mspssScore, issTotalScore, aceScore, gritScore   
 ) -> correlationDataset   
col\_names <- names(correlationDataset)  
correlationDataset[,col\_names] <- lapply(correlationDataset[,col\_names], as.numeric)  
correlation <- cor(correlationDataset, use="pairwise.complete.obs")  
correlation

## incomeLastYr incomeChange povertyRatio  
## incomeLastYr 1.00000000 0.62482535 0.90578130  
## incomeChange 0.62482535 1.00000000 0.57088437  
## povertyRatio 0.90578130 0.57088437 1.00000000  
## managingFinancially 0.40586231 0.22977264 0.24798815  
## savings 0.17256374 0.03563944 0.10726547  
## programsUsedBinary -0.22678323 0.05087505 -0.12165382  
## countServicesAfter -0.19191908 0.10645947 -0.22408511  
## countHelpDuring -0.11962310 0.05955237 -0.04728375  
## countHelpAfter -0.01285084 0.14400773 -0.03241994  
## educationHighest 0.33318683 0.30792185 0.44530936  
## ageNow -0.11081109 -0.06807876 0.03311116  
## ageRelease -0.24684348 -0.35578660 -0.08304453  
## incarcerationTime -0.09992073 -0.12658057 -0.06426462  
## timeReleased 0.25612365 0.53116534 0.19455359  
## mspssScore 0.08568878 0.04048865 0.06090829  
## issTotalScore -0.14899162 -0.27682815 -0.15198710  
## aceScore 0.12372821 -0.14093115 0.18268279  
## gritScore 0.00860284 0.03274548 0.01599544  
## managingFinancially savings programsUsedBinary  
## incomeLastYr 0.40586231 0.172563740 -0.226783228  
## incomeChange 0.22977264 0.035639442 0.050875049  
## povertyRatio 0.24798815 0.107265475 -0.121653817  
## managingFinancially 1.00000000 0.465169802 0.168101584  
## savings 0.46516980 1.000000000 0.118957738  
## programsUsedBinary 0.16810158 0.118957738 1.000000000  
## countServicesAfter -0.04546393 -0.139129636 0.395744156  
## countHelpDuring -0.04737473 0.089722875 0.186725995  
## countHelpAfter 0.03477889 -0.004776222 0.125038596  
## educationHighest -0.16314221 0.023136735 0.007530313  
## ageNow -0.24607676 -0.079523693 0.065615387  
## ageRelease -0.28127030 -0.053947668 -0.005941853  
## incarcerationTime -0.12000998 0.274925636 0.019202122  
## timeReleased 0.10309456 -0.038318772 0.116111981  
## mspssScore -0.05715269 0.078874552 0.099528730  
## issTotalScore -0.01516596 0.012049376 0.079196581  
## aceScore 0.05294731 0.115835570 0.048407327  
## gritScore 0.23272742 0.415394986 0.331767042  
## countServicesAfter countHelpDuring countHelpAfter  
## incomeLastYr -0.19191908 -0.119623101 -0.012850836  
## incomeChange 0.10645947 0.059552366 0.144007728  
## povertyRatio -0.22408511 -0.047283753 -0.032419943  
## managingFinancially -0.04546393 -0.047374729 0.034778891  
## savings -0.13912964 0.089722875 -0.004776222  
## programsUsedBinary 0.39574416 0.186725995 0.125038596  
## countServicesAfter 1.00000000 0.056534457 0.246311145  
## countHelpDuring 0.05653446 1.000000000 0.320884949  
## countHelpAfter 0.24631115 0.320884949 1.000000000  
## educationHighest -0.06901919 0.080244605 0.135340750  
## ageNow 0.04969350 -0.076346329 0.111470037  
## ageRelease -0.07133912 0.002144288 0.058158604  
## incarcerationTime -0.15511305 0.317788277 0.323490954  
## timeReleased 0.17946294 -0.119890611 0.062933009  
## mspssScore -0.24586681 -0.165462468 -0.058601491  
## issTotalScore -0.37394779 -0.110734804 -0.039540032  
## aceScore -0.08959358 -0.066758693 -0.280020058  
## gritScore -0.02884087 -0.132117961 -0.108723218  
## educationHighest ageNow ageRelease  
## incomeLastYr 0.333186828 -0.11081109 -0.246843481  
## incomeChange 0.307921855 -0.06807876 -0.355786599  
## povertyRatio 0.445309360 0.03311116 -0.083044529  
## managingFinancially -0.163142209 -0.24607676 -0.281270301  
## savings 0.023136735 -0.07952369 -0.053947668  
## programsUsedBinary 0.007530313 0.06561539 -0.005941853  
## countServicesAfter -0.069019190 0.04969350 -0.071339117  
## countHelpDuring 0.080244605 -0.07634633 0.002144288  
## countHelpAfter 0.135340750 0.11147004 0.058158604  
## educationHighest 1.000000000 0.26043998 0.050811510  
## ageNow 0.260439977 1.00000000 0.820731148  
## ageRelease 0.050811510 0.82073115 1.000000000  
## incarcerationTime -0.092704582 0.16821392 0.381277251  
## timeReleased 0.335603087 0.13070043 -0.459143908  
## mspssScore 0.086080378 0.11755203 0.259760622  
## issTotalScore -0.163983291 0.23195723 0.474736479  
## aceScore 0.033904457 0.21280707 0.276536370  
## gritScore 0.120523856 0.09270469 -0.020837932  
## incarcerationTime timeReleased mspssScore  
## incomeLastYr -0.09992073 0.25612365 0.08568878  
## incomeChange -0.12658057 0.53116534 0.04048865  
## povertyRatio -0.06426462 0.19455359 0.06090829  
## managingFinancially -0.12000998 0.10309456 -0.05715269  
## savings 0.27492564 -0.03831877 0.07887455  
## programsUsedBinary 0.01920212 0.11611198 0.09952873  
## countServicesAfter -0.15511305 0.17946294 -0.24586681  
## countHelpDuring 0.31778828 -0.11989061 -0.16546247  
## countHelpAfter 0.32349095 0.06293301 -0.05860149  
## educationHighest -0.09270458 0.33560309 0.08608038  
## ageNow 0.16821392 0.13070043 0.11755203  
## ageRelease 0.38127725 -0.45914391 0.25976062  
## incarcerationTime 1.00000000 -0.42259121 -0.05857119  
## timeReleased -0.42259121 1.00000000 -0.25545170  
## mspssScore -0.05857119 -0.25545170 1.00000000  
## issTotalScore 0.26972259 -0.44993734 0.76878108  
## aceScore 0.09365148 -0.11031789 0.14027692  
## gritScore 0.18657837 0.19928977 0.07119170  
## issTotalScore aceScore gritScore  
## incomeLastYr -0.14899162 0.12372821 0.00860284  
## incomeChange -0.27682815 -0.14093115 0.03274548  
## povertyRatio -0.15198710 0.18268279 0.01599544  
## managingFinancially -0.01516596 0.05294731 0.23272742  
## savings 0.01204938 0.11583557 0.41539499  
## programsUsedBinary 0.07919658 0.04840733 0.33176704  
## countServicesAfter -0.37394779 -0.08959358 -0.02884087  
## countHelpDuring -0.11073480 -0.06675869 -0.13211796  
## countHelpAfter -0.03954003 -0.28002006 -0.10872322  
## educationHighest -0.16398329 0.03390446 0.12052386  
## ageNow 0.23195723 0.21280707 0.09270469  
## ageRelease 0.47473648 0.27653637 -0.02083793  
## incarcerationTime 0.26972259 0.09365148 0.18657837  
## timeReleased -0.44993734 -0.11031789 0.19928977  
## mspssScore 0.76878108 0.14027692 0.07119170  
## issTotalScore 1.00000000 0.25948482 0.17331852  
## aceScore 0.25948482 1.00000000 0.11068394  
## gritScore 0.17331852 0.11068394 1.00000000

## Perform Regression Analysis

The following code chunk performs a multiple regression using select variables of interest.

# Multiple regression using `incomeLastYr` as dependent variable  
regressionIncome <- lm(as.numeric(incomeLastYr) ~ as.numeric(programsUsedBinary) + as.numeric(ageNow) +  
 as.numeric(ageRelease) + as.numeric(timeReleased) + mspssScore + issTotalScore +  
 aceScore + gritScore + gender + as.factor(ethnicityRace) +   
 as.factor(typeHometown),   
 data = dataClean, na.action = na.omit)  
summary(regressionIncome)

##   
## Call:  
## lm(formula = as.numeric(incomeLastYr) ~ as.numeric(programsUsedBinary) +   
## as.numeric(ageNow) + as.numeric(ageRelease) + as.numeric(timeReleased) +   
## mspssScore + issTotalScore + aceScore + gritScore + gender +   
## as.factor(ethnicityRace) + as.factor(typeHometown), data = dataClean,   
## na.action = na.omit)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -4.4183 -1.2365 -0.0149 1.1679 4.3719   
##   
## Coefficients: (1 not defined because of singularities)  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 12.49743 5.46175 2.288 0.03018 \*   
## as.numeric(programsUsedBinary) -10.15515 3.09860 -3.277 0.00288 \*\*  
## as.numeric(ageNow) 0.10554 0.10844 0.973 0.33905   
## as.numeric(ageRelease) -0.12300 0.09009 -1.365 0.18343   
## as.numeric(timeReleased) NA NA NA NA   
## mspssScore 0.75676 0.48483 1.561 0.13020   
## issTotalScore -1.62809 1.21211 -1.343 0.19040   
## aceScore 0.12033 0.19062 0.631 0.53320   
## gritScore 1.00796 0.70487 1.430 0.16419   
## gender -1.64582 1.05240 -1.564 0.12949   
## as.factor(ethnicityRace)4 -0.86711 3.09493 -0.280 0.78148   
## as.factor(ethnicityRace)5 1.11991 3.28097 0.341 0.73549   
## as.factor(ethnicityRace)7 3.32720 3.53304 0.942 0.35468   
## as.factor(ethnicityRace)9 -0.25210 3.05620 -0.082 0.93487   
## as.factor(ethnicityRace)10 1.74369 3.55354 0.491 0.62761   
## as.factor(typeHometown)3 -1.03418 1.08560 -0.953 0.34923   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 2.67 on 27 degrees of freedom  
## (17 observations deleted due to missingness)  
## Multiple R-squared: 0.5111, Adjusted R-squared: 0.2576   
## F-statistic: 2.016 on 14 and 27 DF, p-value: 0.05729

# Multiple regressiong using `povertyRatio` as dependent variable  
regressionPoverty <- lm(as.numeric(povertyRatio) ~ as.numeric(programsUsedBinary) + as.numeric(ageNow) +  
 as.numeric(ageRelease) + as.numeric(timeReleased) + mspssScore + issTotalScore +  
 aceScore + gritScore + gender + as.factor(ethnicityRace) +  
 as.factor(typeHometown),   
 data = dataClean, na.action = na.omit)  
summary(regressionPoverty)

##   
## Call:  
## lm(formula = as.numeric(povertyRatio) ~ as.numeric(programsUsedBinary) +   
## as.numeric(ageNow) + as.numeric(ageRelease) + as.numeric(timeReleased) +   
## mspssScore + issTotalScore + aceScore + gritScore + gender +   
## as.factor(ethnicityRace) + as.factor(typeHometown), data = dataClean,   
## na.action = na.omit)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -0.56012 -0.19372 -0.05032 0.07849 0.95970   
##   
## Coefficients: (1 not defined because of singularities)  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 0.944571 0.850680 1.110 0.2794   
## as.numeric(programsUsedBinary) -0.974995 0.493850 -1.974 0.0616 .  
## as.numeric(ageNow) 0.003337 0.017187 0.194 0.8479   
## as.numeric(ageRelease) 0.005200 0.014683 0.354 0.7268   
## as.numeric(timeReleased) NA NA NA NA   
## mspssScore 0.194719 0.086430 2.253 0.0351 \*  
## issTotalScore -0.478596 0.216048 -2.215 0.0379 \*  
## aceScore 0.013328 0.032891 0.405 0.6894   
## gritScore 0.279716 0.130570 2.142 0.0440 \*  
## gender -0.329910 0.179168 -1.841 0.0797 .  
## as.factor(ethnicityRace)4 -0.215592 0.480905 -0.448 0.6585   
## as.factor(ethnicityRace)5 0.100089 0.548937 0.182 0.8571   
## as.factor(ethnicityRace)7 0.904504 0.590565 1.532 0.1406   
## as.factor(ethnicityRace)9 0.043267 0.470499 0.092 0.9276   
## as.factor(ethnicityRace)10 -0.122368 0.551673 -0.222 0.8266   
## as.factor(typeHometown)3 0.159810 0.189103 0.845 0.4076   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.4014 on 21 degrees of freedom  
## (23 observations deleted due to missingness)  
## Multiple R-squared: 0.5009, Adjusted R-squared: 0.1682   
## F-statistic: 1.506 on 14 and 21 DF, p-value: 0.1927

# Multiple regression using `managingFinancially` as dependent variable  
regressionFinStat <- lm(as.numeric(managingFinancially) ~ as.numeric(programsUsedBinary) +   
 as.numeric(ageNow) + as.numeric(ageRelease) + as.numeric(timeReleased) +   
 mspssScore + issTotalScore + aceScore + gritScore + gender +   
 as.factor(ethnicityRace) + as.factor(typeHometown),   
 data = dataClean, na.action = na.omit)  
summary(regressionFinStat)

##   
## Call:  
## lm(formula = as.numeric(managingFinancially) ~ as.numeric(programsUsedBinary) +   
## as.numeric(ageNow) + as.numeric(ageRelease) + as.numeric(timeReleased) +   
## mspssScore + issTotalScore + aceScore + gritScore + gender +   
## as.factor(ethnicityRace) + as.factor(typeHometown), data = dataClean,   
## na.action = na.omit)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -1.2805 -0.6750 0.0000 0.6085 1.6177   
##   
## Coefficients: (1 not defined because of singularities)  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 1.70971 2.01940 0.847 0.4049   
## as.numeric(programsUsedBinary) -0.77886 1.14573 -0.680 0.5026   
## as.numeric(ageNow) -0.03838 0.04024 -0.954 0.3490   
## as.numeric(ageRelease) -0.01255 0.03398 -0.369 0.7148   
## as.numeric(timeReleased) NA NA NA NA   
## mspssScore -0.13181 0.17944 -0.735 0.4692   
## issTotalScore 0.18293 0.44950 0.407 0.6874   
## aceScore 0.08079 0.07260 1.113 0.2760   
## gritScore 0.46217 0.26164 1.766 0.0891 .  
## gender -0.30535 0.39015 -0.783 0.4409   
## as.factor(ethnicityRace)4 0.73249 1.14434 0.640 0.5277   
## as.factor(ethnicityRace)5 1.31207 1.21304 1.082 0.2893   
## as.factor(ethnicityRace)7 1.65569 1.30618 1.268 0.2162   
## as.factor(ethnicityRace)9 1.07772 1.13056 0.953 0.3492   
## as.factor(ethnicityRace)10 2.39367 1.31622 1.819 0.0805 .  
## as.factor(typeHometown)3 -0.20699 0.41052 -0.504 0.6184   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.987 on 26 degrees of freedom  
## (18 observations deleted due to missingness)  
## Multiple R-squared: 0.4093, Adjusted R-squared: 0.09124   
## F-statistic: 1.287 on 14 and 26 DF, p-value: 0.2795

# Binary logistic regression using `savings` as dependent variable  
# Prepare data  
dataClean %>%  
 dplyr::select(incomeLastYr, povertyRatio, povertyLevel, managingFinancially, savings, programsUsedBinary,   
 ageNow, ageRelease, mspssScore, issTotalScore, aceScore, gritScore) -> dataBinary  
  
# Remove cases with missing data in the `savings` variable  
dataBinary %>%  
 drop\_na() -> dataBinary  
  
logitSavings <- glm(savings ~ programsUsedBinary + ageNow + ageRelease + mspssScore + issTotalScore +   
 aceScore + gritScore,   
 data = dataBinary, family = binomial, na.action = na.omit)  
summary(logitSavings)

##   
## Call:  
## glm(formula = savings ~ programsUsedBinary + ageNow + ageRelease +   
## mspssScore + issTotalScore + aceScore + gritScore, family = binomial,   
## data = dataBinary, na.action = na.omit)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.3239 -0.5582 -0.3746 0.1367 2.0327   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -20.33319 2399.54780 -0.008 0.9932   
## programsUsedBinary 10.64810 2399.54596 0.004 0.9965   
## ageNow -0.12970 0.09792 -1.325 0.1853   
## ageRelease 0.02510 0.08704 0.288 0.7731   
## mspssScore 0.26811 0.48162 0.557 0.5777   
## issTotalScore -0.01106 1.18305 -0.009 0.9925   
## aceScore 0.14609 0.19706 0.741 0.4585   
## gritScore 2.47753 1.06846 2.319 0.0204 \*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 40.488 on 35 degrees of freedom  
## Residual deviance: 28.670 on 28 degrees of freedom  
## AIC: 44.67  
##   
## Number of Fisher Scoring iterations: 15

# Raise e to the coefficients  
print(exp(coef(logitSavings)))

## (Intercept) programsUsedBinary ageNow   
## 1.477095e-09 4.211268e+04 8.783575e-01   
## ageRelease mspssScore issTotalScore   
## 1.025420e+00 1.307491e+00 9.890011e-01   
## aceScore gritScore   
## 1.157300e+00 1.191185e+01

# Obtain various pseudo R-squared measures  
print(pR2(logitSavings))

## llh llhNull G2 McFadden r2ML r2CU   
## -14.3351916 -20.2440652 11.8177472 0.2918818 0.2798320 0.4144183

# Confidence intervals for the coefficients  
print(exp(confint(logitSavings, level = 0.95)))

## 2.5 % 97.5 %  
## (Intercept) NA 7.819156e+178  
## programsUsedBinary 2.317899e-86 NA  
## ageNow 7.018995e-01 1.043372e+00  
## ageRelease 8.668258e-01 1.237051e+00  
## mspssScore 5.117305e-01 3.628320e+00  
## issTotalScore 8.078762e-02 1.036382e+01  
## aceScore 7.963728e-01 1.777218e+00  
## gritScore 2.020598e+00 1.507872e+02

# Hosmer-Lemeshow Goodness of Fit Test  
# Null hypothesis: the model is a good fit for the data  
# Alternative hypothesis: the model is NOT a good fit for the data  
HosLemLogitSavings <- hoslem.test(dataBinary$savings,   
 fitted(logitSavings), g=10)  
print(HosLemLogitSavings)

##   
## Hosmer and Lemeshow goodness of fit (GOF) test  
##   
## data: dataBinary$savings, fitted(logitSavings)  
## X-squared = 10.718, df = 8, p-value = 0.2182

print(cbind(HosLemLogitSavings$expected, HosLemLogitSavings$observed))

## yhat0 yhat1 y0 y1  
## [6.39e-08,0.0118] 3.975808 0.02419172 4 0  
## (0.0118,0.0413] 3.914357 0.08564276 4 0  
## (0.0413,0.0902] 2.778554 0.22144642 3 0  
## (0.0902,0.11] 3.589933 0.41006705 4 0  
## (0.11,0.138] 2.610528 0.38947236 2 1  
## (0.138,0.235] 3.313454 0.68654628 3 1  
## (0.235,0.328] 2.193337 0.80666328 1 2  
## (0.328,0.53] 2.268095 1.73190515 3 1  
## (0.53,0.603] 1.256724 1.74327587 3 0  
## (0.603,0.861] 1.099211 2.90078918 0 4

## Save Data

The following code chunk saves the cleaned data used for the analysis.

write.csv(dataClean, here("Data","DataClean","SOC5800\_Data\_NumericValues\_Clean\_CSV.csv"), append = FALSE)