Machine Learning Final

Tony Daza

2023-12-08

Data Source & Overview

In this final project I am examining student mental health rating data along with student alcohol use. The data includes several predictors ranging from student stress to parent education and income. The data was simulated from kaggle from the following link: https://www.kaggle.com/datasets/rkiattisak/student-performance-in-mathematics

The actual data generator is from this website: http://roycekimmons.com/tools/generated_data/exams

The original data generator had the following variables:

- Gender: The gender of the student (male/female)
- Race/ethnicity: The student's racial or ethnic background (Asian, African-American, Hispanic, etc.)
- Parental level of education: The highest level of education attained by the student's parent(s) or guardian(s)
- Lunch: Whether the student receives free or reduced-price lunch (yes/no)
- Test preparation course: Whether the student completed a test preparation course (yes/no)
- Math score: The student's score on a standardized mathematics test
- Reading score: The student's score on a standardized reading test
- Writing score: The student's score on a standardized writing test

The remaining predictors were simulated and assigned in R:

- Parent Income: The student's family income in dollars, based on US averages (Taking into account parent level of education)
- School location: The student's school's location (Urban, suburban, or rural)
- School type: The student's school's type (Charter or Public)
- Race: The student's race, recoded from the original data for ease of reading
- Lunch: Recoded from the original data to take into account national income cutoff scores and split the data into (free, reduced, and standard) lunch. Using data from: https://www.federalregister.gov/documents/2020/03/20/2020-05982/child-nutrition-programs-income-eligibility-guidelines
- EL status: The student's EL status (EL or Non-EL)
- Home Language: The student's home language
- Grade: The student's grade level (6th - 8th grade)
- Age: The student's age (11-14 years old)
- Number of Close friends: The student's self reported number of close friends (Based on: DeLay D, Ha T, Van Ryzin M, Winter C, Dishion TJ. Changing Friend Selection in Middle School: A Social Network

Analysis of a Randomized Intervention Study Designed to Prevent Adolescent Problem Behavior. Prev Sci. 2016 Apr;17(3):285-94. doi: 10.1007/s11121-015-0605-4. PMID: 26377235; PMCID: PMC4791197.)

- Presence of a Trusted adult: The student's self-reported indicator of the presence of a trust adult at school.

Student Mental Health constructs based on a student Health & Wellness Survey and modeled off the findings of the Health and Wellness Survey results from the Lab School in Chicago:

- Mental Health Rating: The student's self-reported mental health rating on a 5 point scale

- Stress Rating: The student's self-reported school stress level on a 10 point scale

- Belonging rating: The student's belonging rating at school on a 7 point scale

- SES Scaled Score: The student's socio-economic status based on parent education, parent income, & FRP lunch status, but scaled to be on a 10 point scale.

Alcohol and Drug use based on: https://www.niaaa.nih.gov/publications/brochures-and-fact-sheets/underage-drinking • Student Alcohol Use: The student's self-report of ever using alcohol

- Marijuana Use: The student's self-report of ever using marijuana

- Number of Siblings: The student's self-reported number of siblings

- Number of Pets: The student's self-reported number of pets

Admittedly this is simulated data, but seeing as I could not find a suitable dataset and my projects do not currently have data for me to use, I found a data generator online that has simulated data for student scores.

The sample size is 5000 students from across public and charter schools from varying SES backgrounds and school settings such as suburban, urban, and rural.

Research Questions

The purpose of the assignment is to examine potential predictors of student mental health rating

Research Questions:

What factors are the most important for predicting student mental health rating?

Are there any variables that negatively impact student mental health ratings? What about positively impact mental health?

What factors are most important for predicting student use of alcohol?

What factors predict alcohol use positively and negatively?

##Data Loading

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.3     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.3     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.0
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(readr)
library(data.table)
```

```
##
## Attaching package: 'data.table'
##
## The following objects are masked from 'package:lubridate':
##
##     hour, isoweek, mday, minute, month, quarter, second, wday, week,
##     yday, year
##
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
##
## The following object is masked from 'package:purrr':
##
##     transpose
```

```
exams_1 <- read_csv("~/Desktop/Machine Learning Final/exams (1).csv")
```

```
## Rows: 1000 Columns: 8
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (5): gender, race/ethnicity, parental level of education, lunch, test pr...
## dbl (3): math score, reading score, writing score
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
exams_2 <- read_csv("~/Desktop/Machine Learning Final/exams.csv")
```

```
## Rows: 1000 Columns: 8
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (5): gender, race/ethnicity, parental level of education, lunch, test pr...
## dbl (3): math score, reading score, writing score
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
exams_3 <- read_csv("~/Desktop/Machine Learning Final/exams (2).csv")
```

```
## Rows: 1000 Columns: 8
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (5): gender, race/ethnicity, parental level of education, lunch, test pr...
## dbl (3): math score, reading score, writing score
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
exams_4 <- read_csv("~/Desktop/Machine Learning Final/exams (3).csv")
```

```
## Rows: 1000 Columns: 8
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (5): gender, race/ethnicity, parental level of education, lunch, test pr...
## dbl (3): math score, reading score, writing score
##
```

```
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
exams_5 <- read_csv("~/Desktop/Machine Learning Final/exams (4).csv")
```

```
## Rows: 1000 Columns: 8
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (5): gender, race/ethnicity, parental level of education, lunch, test pr...
## dbl (3): math score, reading score, writing score
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
exams_1$id <- seq(1, 1000)
exams_2$id <- seq(1001,2000)
exams_3$id <- seq(2001,3000)
exams_4$id <- seq(3001,4000)
exams_5$id <- seq(4001,5000)

exams <- rbind(exams_1, exams_2, exams_3, exams_4, exams_5)
```

I have been having trouble further down, so I am going to fix some of the column names. Currently some have an underscore and others use spaces, but I do not like the spaces so I am going to sub in underscores.

```
names(exams) <- gsub(" ", "_", names(exams))

# Display updated column names
names(exams)
```

```
## [1] "gender"                      "race/ethnicity"
## [3] "parental_level_of_education" "lunch"
## [5] "test_preparation_course"     "math_score"
## [7] "reading_score"               "writing_score"
## [9] "id"
```

Data Generation

Next I added in some hypothetical variables.

Parent Income

```
set.seed(121619)


# Education levels and associated weights (hypothetical)
education_weights <- c(
  "some high school" = 28000,
  "high school" = 34000,
  "some college" = 37000,
  "associate's degree" = 42000,
  "bachelor's degree" = 58000,
  "master's degree" = 80000
)
```

```r
# Creating a new column 'simulated_income' based on education weights
exams$parent_income <- education_weights[exams$parental_level_of_education]

# Adding randomness (variation) to the simulated income
exams$parent_income <- exams$parent_income + rnorm(nrow(exams), mean = 0, sd = 10000)

# Setting a minimum income value of 0
exams$parent_income <- pmax(exams$parent_income, 0)
```

School location

```r
set.seed(121691)

weight_urban <- 0.30
weight_suburban <- 0.57
weight_rural <- 0.13

# Create a vector representing school types (suburban, urban, rural)
school_location <- c("Suburban", "Urban", "Rural")

# Generate random school type assignments for each student
exams$school_location <- sample(school_location, nrow(exams), replace = TRUE, prob = c(weight_urban, wei
```

School type

```r
set.seed(121691)

# Weighted distribution percentages (estimated)
weight_public <- 0.90
weight_charter <- 0.1

# Create a vector representing school types (public, private, charter) based on weights
school_types <- c("Public", "Charter")

# Generate random school type assignments for each student based on weighted probabilities
exams$school_type <- sample(school_types, nrow(exams), replace = TRUE, prob = c(weight_public, weight_c
```

Free & reduced price lunch

```r
# Set the cutoff values
cutoff_free_lunch <- 34000  # Cutoff for free lunch
cutoff_reduced_lunch <- 49000  # Cutoff for reduced-price lunch

# Create a new column 'lunch_status' with default as 'Standard'
exams$lunch <- "Standard"

# Assign lunch status based on family income
exams$lunch[exams$parent_income <= cutoff_free_lunch] <- "Free"
exams$lunch[exams$parent_income > cutoff_free_lunch & exams$parent_income <= cutoff_reduced_lunch] <- "
```

```r
# Using summary function to get an overview of lunch status distribution
summary(exams$lunch)
```

```
##    Length     Class      Mode 
##      5000 character character
```

```r
# If you want counts of each lunch status category
table(exams$lunch)
```

```
## 
##     Free  Reduced Standard 
##     1858     1901     1241
```

```r
# If you want proportions/percentages of each lunch status category
prop.table(table(exams$lunch)) * 100
```

```
## 
##     Free  Reduced Standard 
##    37.16    38.02    24.82
```

Checks

```r
unique_levels <- unique(exams$parental_level_of_education)
print(unique_levels)
```

```
## [1] "master's degree"    "high school"        "associate's degree"
## [4] "some high school"   "bachelor's degree"  "some college"
```

```r
# Assuming 'exams' is your dataset containing the 'race' column
unique_levels_race <- unique(exams$`race/ethnicity`)
print(unique_levels_race)
```

```
## [1] "group E" "group D" "group A" "group B" "group C"
```

```r
# Using summary function to get an overview of lunch status distribution
summary(exams$`race/ethnicity`)
```

```
##    Length     Class      Mode 
##      5000 character character
```

```r
# If you want counts of each lunch status category
table(exams$`race/ethnicity`)
```

```
## 
## group A group B group C group D group E 
##     416    1047    1548    1275     714
```

```r
# If you want proportions/percentages of each lunch status category
prop.table(table(exams$`race/ethnicity`)) * 100
```

```
## 
## group A group B group C group D group E 
##    8.32   20.94   30.96   25.50   14.28
```

Race recode

```r
# Group C = White
# Group D = Latino
# Group B = Black
# Group E = 2 or more
# Group A = Asian
```

```r
tag_to_race <- c(
  "group C" = "White",
  "group D" = "Latine",
  "group B" = "Black",
  "group E" = "2 or more",
  "group A" = "Asian"
)

# Create a new column 'race_category' based on the mapping
exams$race <- tag_to_race[exams$`race/ethnicity`]

exams <- subset(exams, select = -`race/ethnicity`)
```

Language

```r
set.seed(121691)

exams$el_status <- NA
exams$home_language <- NA

# Hypothetical prevalence of languages other than English spoken at home in the US
non_english_prevalence <- c(
  "White" = 0.10,    # 10% for White group
  "Latine" = 0.45,   # 45% for Latine group
  "Black" = 0.11,    # 20% for Black group
  "2 or more" = 0.35,# 35% for 2 or more group
  "Asian" = 0.12     # 10% for Asian group
)

# Updated hypothetical weights for language other than English based on race categories
language_weights <- list(
  "White" = c("English", "Spanish", "French", "Other"),
  "Latine" = c("Spanish", "English", "Other"),
  "Black" = c("English", "French", "Other"),
  "2 or more" = c("English", "Spanish", "Other"),
  "Asian" = c("Chinese", "English", "Korean", "Other")
)

# Function to randomly assign language status and home language based on weights for individual student
assign_language_status <- function(student_id, weights, prevalence) {
  student_race <- exams$race[exams$id == student_id]  # Get the race for the given student ID
  el_status <- ifelse(runif(1) <= prevalence[student_race], "EL", "not EL")

  if (el_status == "EL") {
    # Selecting a home language if the student is an English Learner
    non_english_options <- weights[[student_race]]
    home_language <- if (length(non_english_options) > 0) {
      sample(non_english_options, 1)
    } else {
      "English"  # No non-English options available
    }
  } else {
```

```
    # For students not classified as English Learners, assign English as the home language
    home_language <- "English"
  }

  return(list(el_status = el_status, home_language = home_language))
}

# Generate language status and home language for each student based on their race
for (student_id in unique(exams$id)) {
  result <- assign_language_status(student_id, language_weights, non_english_prevalence)
  exams$el_status[exams$id == student_id] <- result$el_status
  exams$home_language[exams$id == student_id] <- result$home_language
}


table(exams$el_status)
```

```
##
##     EL not EL
##   1112   3888
```

```
table(exams$home_language)
```

```
##
## Chinese English  French  Korean   Other Spanish
##      12    4240      87      12     353     296
```

Grade level

```
set.seed(121619)  # For reproducibility

# Assuming 'exams' is your dataset and 'id' is the student ID column
exams$grade <- sample(6:8, nrow(exams), replace = TRUE)

table(exams$grade)
```

```
##
##    6    7    8
## 1659 1714 1627
```

Age

```
set.seed(121619)  # For reproducibility

# Assuming 'exams' is your dataset and 'grade' is the column representing the student's grade
exams$age <- ifelse(exams$grade == 6, sample(11:12, nrow(exams), replace = TRUE),
                    ifelse(exams$grade == 7, sample(12:13, nrow(exams), replace = TRUE),
                           ifelse(exams$grade == 8, sample(13:14, nrow(exams), replace = TRUE), NA)))

table(exams$age)
```

```
##
##   11   12   13   14
##  862 1645 1679  814
```

Number of Friends

```r
set.seed(121691)  # For reproducibility

exams$close_friends <- NA

n <- nrow(exams)  # Number of rows in the dataset
average_friends <- 3  # Desired average number of close friends

# Generate close_friends column with values between 0 and 7
exams$close_friends <- pmin(pmax(round(rnorm(n, mean = average_friends, sd = 1)), 0), 7)

# Check summary statistics of the close_friends column
summary(exams$close_friends)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   2.000   3.000   2.988   4.000   6.000
```

Presence of a Trusted adults

```r
set.seed(121691)  # For reproducibility

n <- nrow(exams)  # Number of rows in the dataset
exams$trusted_adult <- NA

# 0.70 is the hypothetical proportion indicating the presence of a trusted adult at school
exams$trusted_adult <- ifelse(runif(n) <= 0.72, 1, 0)

# Check the distribution of trusted_adult_at_school column
table(exams$trusted_adult)
```

```
##
##    0    1
## 1380 3620
```

Mental Health self-report

```r
set.seed(121619)  # For reproducibility

# Assuming 'exams' is your dataset
# Generate self-rated mental and emotional health
exams$mental_health <- sample(c("poor", "fair", "good", "very good", "excellent"),
                              nrow(exams), replace = TRUE,
                              prob = c(0.05, 0.1, 0.30, 0.40, 0.15))

# Adjust the distribution for mental health rating
good_verygood_excellent <- c("good", "very good", "excellent")
exams$mental_health <- ifelse(exams$mental_health %in% good_verygood_excellent,
                              exams$mental_health,
                              sample(c("fair", "poor"), sum(!exams$mental_health %in% good_verygood_exc

# Checking the distribution of self-rated mental and emotional health
mental_tab <- as.data.frame(table(exams$mental_health))
mental_tab
```

```
##        Var1 Freq
## 1 excellent  736
## 2      fair  351
## 3      good 1514
## 4      poor  395
## 5 very good 2004
```

Student stress rating

```r
set.seed(121619)  # For reproducibility

# Assuming 'exams' is your dataset
# Create a column for stress rating and initialize with NA values
exams$stress_rating <- NA

# Define the stress factors and their probabilities for Middle School students
middle_school_stress_factors <- c("school_work", "grades", "family_expectations")

# Assign stress ratings for Middle School students
middle_school_students <- exams$grade %in% 6:8  # Assuming grade 6, 7, and 8 are Middle School

# Generate stress ratings based on the stress factors for Middle School students
exams$stress_rating[middle_school_students] <- sample(c(1:10), sum(middle_school_students), replace = T
                                                      prob = c(0.01, 0.02, 0.08, 0.1, 0.27, 0.23, 0.1, 

# Checking the distribution of stress ratings for Middle School students
summary(exams$stress_rating[middle_school_students])
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   5.000   6.000   5.733   7.000  10.000
```

Belonging Rating

```r
set.seed(121619)  # For reproducibility

# Assuming 'exams' is your dataset
# Create a column for belonging scale and initialize with NA values
exams$belonging <- NA

# Assign belonging scale for all students
exams$belonging <- sample(c("very unwelcome", "mostly unwelcome", "welcome half the time", "mostly welc
                                nrow(exams), replace = TRUE,
                                prob = c(0.05, 0.05, 0.1, 0.2, 0.6))  # Adjust probabilities

# Adjust belonging scale based on demographic factors
# Students based on gender identity
male_students <- exams$gender == "male"

exams$belonging[male_students] <- sample(c("very unwelcome", "mostly unwelcome", "welcome half the time
                                                sum(male_students), replace = TRUE,
                                                prob = c(0.05, 0.05, 0.1, 0.2, 0.7))  # Adjust probabil

# Students based on self-identified race or ethnicity
```

```
white_students <- exams$race == "White"

exams$belonging[white_students] <- sample(c("very unwelcome", "mostly unwelcome", "welcome half the tim
                                          "mostly welcome", "very welcome"),
                                        sum(white_students), replace = TRUE,
                                        prob = c(0.1, 0.1, 0.2, 0.3, 0.3))  # Adjust probabilit

# Checking the distribution of belonging scale
table(exams$belonging)
```

```
##
##      mostly unwelcome        mostly welcome        very unwelcome
##                   327                  1076                   297
##          very welcome welcome half the time
##                  2662                   638
```

SES Scaled score

```
set.seed(121619)  # For reproducibility

# Create a column for SES score and initialize with NA values
exams$ses_score <- NA

# Assign weights to parent education level, parent income, and lunch status
weight_education <- c("some high school" = 1, "high school" = 3, "some college" = 4, "associate's degre


weight_income <- ifelse(exams$parent_income <= cutoff_free_lunch, 1,
                        ifelse(exams$parent_income <= cutoff_reduced_lunch, 3, 5))

# Assign SES score for each student
for (i in 1:nrow(exams)) {
  education_weight <- weight_education[exams$parental_level_of_education[i]]
  income_weight <- weight_income[i]
  lunch_weight <- ifelse(exams$lunch[i] == "free", 1, ifelse(exams$lunch[i] == "reduced", 2, 3))

  # Calculate SES score based on weighted factors
  exams$ses_score[i] <- education_weight + income_weight + lunch_weight
}

# Checking the distribution of SES scores
summary(exams$ses_score)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   5.000   7.000   9.000   9.797  12.000  16.000
```

```
# Find the minimum and maximum SES scores
min_ses <- min(exams$ses_score)
max_ses <- max(exams$ses_score)

# Perform min-max scaling to rescale SES scores to a range from 1 to 10
scaled_ses <- ((exams$ses_score - min_ses) / (max_ses - min_ses)) * 9 + 1

scaled_ses <- round(scaled_ses, 2)
```

```r
# Update the SES scores in the dataset with the scaled values
exams$ses_score <- scaled_ses

# Check the distribution of rescaled SES scores
summary(exams$ses_score)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   2.640   4.270   4.925   6.730  10.000
```

Alcohol Use Self-report

```r
set.seed(120623)  # For reproducibility

# Create a column for alcohol use and initialize with 0s (indicating 'No' or 'Not used alcohol')
exams$alcohol_use <- 0

# Function to randomly assign alcohol use based on age and gender rates
assign_alcohol_use <- function(age, gender) {
  if (age >= 14) {
    if (gender == "male") {
      return(runif(1) <= 0.20)  # 20% alcohol use rate for male students age 14-15
    } else {
      return(runif(1) <= 0.22)  # 22% alcohol use rate for female students age 14-15
    }
  } else if (age >= 12) {
    if (gender == "male") {
      return(runif(1) <= 0.17)  # 17% alcohol use rate for male students age 12-13
    } else {
      return(runif(1) <= 0.18)  # 18% alcohol use rate for female students age 12-13
    }
  } else {
    if (gender == "male") {
      return(runif(1) <= 0.05)  # 5% alcohol use rate for male students age 11
    } else {
      return(runif(1) <= 0.08)  # 8% alcohol use rate for female students age 11
    }
  }
}


# Generate alcohol use for each student based on age and gender
for (i in 1:nrow(exams)) {
  age_of_student <- exams$age[i]  # Assuming you have an 'age' column in your dataset
  gender_of_student <- exams$gender[i]  # Assuming you have a 'gender' column

  # Assign alcohol use based on age and gender rates
  exams$alcohol_use[i] <- ifelse(assign_alcohol_use(age_of_student, gender_of_student), 1, 0)
}

# Check the distribution of alcohol_use column
table(exams$alcohol_use)
```

```
##
```

```
##    0    1
## 4170  830
```

Marijuana Use

```
set.seed(121619)  # For reproducibility

exams$marijuana_use <- 0

# Function to randomly assign marijuana use based on age rates
assign_marijuana_use <- function(age) {
  if (age == 14) {
    return(runif(1) <= 0.08)  # 8% marijuana use rate for students age 14
  } else if (age == 12 | age == 13) {
    return(runif(1) <= 0.025)  # 2.5% marijuana use rate for students age 12-13
  } else {
    return(runif(1) <= 0.005)  # No marijuana use for other ages
  }
}

# Generate marijuana use for each student based on age
for (i in 1:nrow(exams)) {
  age_of_student <- exams$age[i]  # Assuming you have an 'age' column in your dataset

  # Assign marijuana use based on age rates
  exams$marijuana_use[i] <- ifelse(assign_marijuana_use(age_of_student), 1, 0)
}

# Check the distribution of marijuana_use column
table(exams$marijuana_use)
```

```
##
##    0    1
## 4862  138
```

Number of siblings

```
set.seed(121619)  # For reproducibility


# Generate number of siblings (0 to 4) randomly assigned with an average of 1
exams$siblings <- sample(0:4, nrow(exams), replace = TRUE, prob = c(0.2, 0.25, 0.25, 0.2, 0.1))


# Check the distribution of number_of_siblings and number_of_pets columns
summary(exams$siblings)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00    1.00    2.00    1.75    3.00    4.00
```

Number of pets

```r
set.seed(121619)  # For reproducibility

# Create 'number_of_pets' column in the dataset
exams$pets <- NA

# Generate number of pets (0 to 5) randomly assigned with a normal distribution around an average of 2
average_pets <- 0  # Average number of pets
std_dev_pets <- 2  # Standard deviation for number of pets

# Generate pets column with normal distribution
exams$pets <- round(rnorm(nrow(exams), mean = average_pets, sd = std_dev_pets))
exams$pets <- pmin(pmax(exams$pets, 0), 5)  # Ensure the values stay within 0 to 5 range

# Check the distribution of number_of_siblings and number_of_pets columns
summary(exams$pets)
```

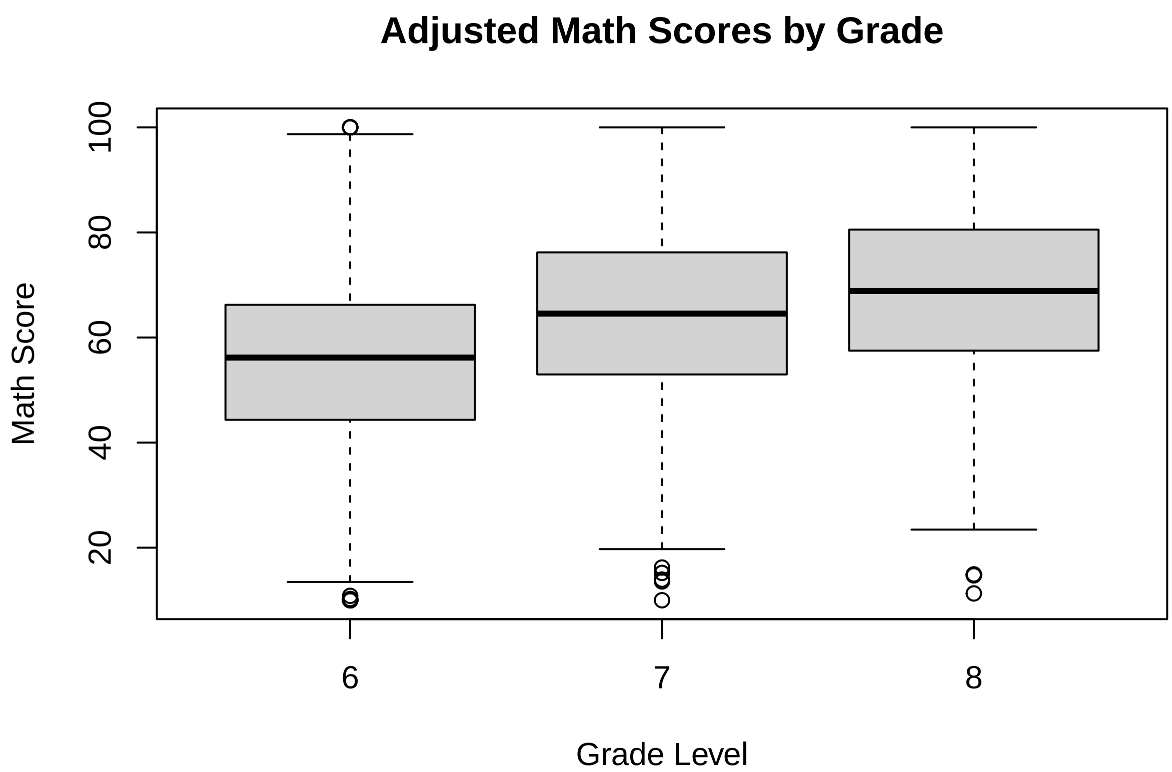
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.0000  0.0000  0.7612  1.0000  5.0000
```

Adjusting score distribution by grade

Math scores

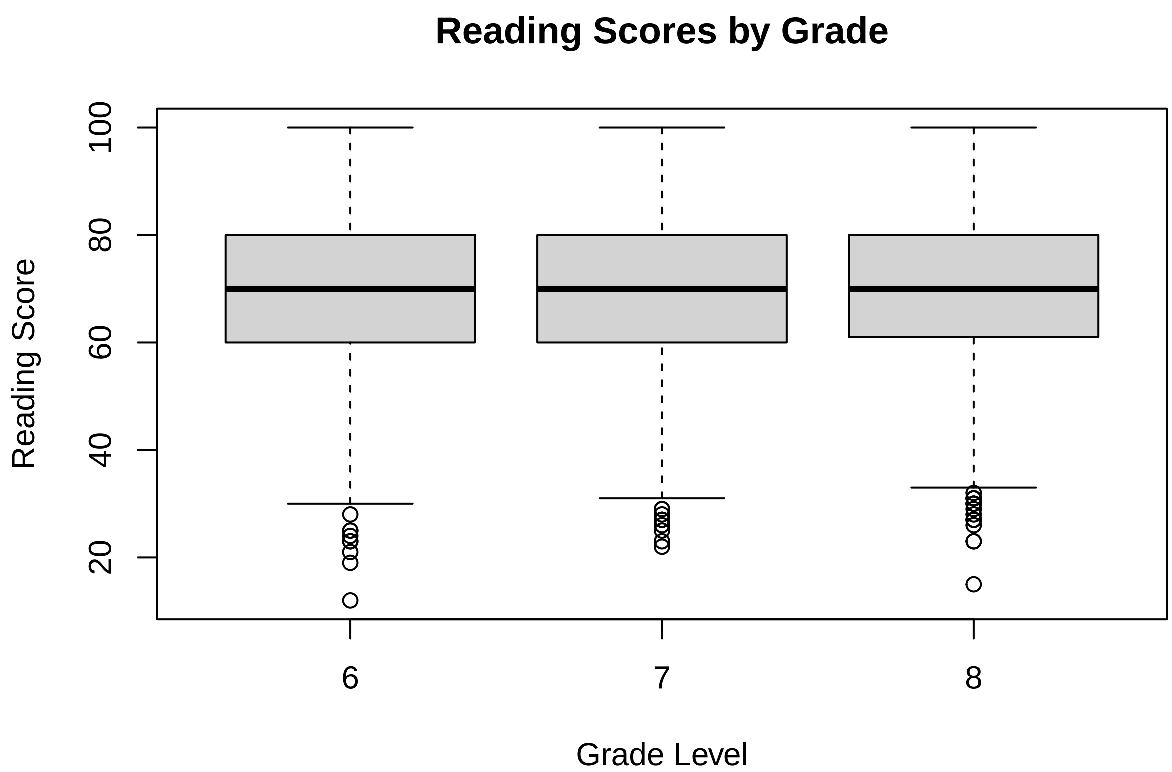
We would expect grades to be somewhat different by grade and to add that variability to the data we are going to adjust the distribution of the math, reading, and writing scores by grade level with a minimum of 10 for the scores.

```r
set.seed(121619)  # For reproducibility

# Function to generate math scores for each grade with desired averages and a minimum score of 10
generate_math_scores <- function(grade, n) {
  avg_score <- ifelse(grade == 8, 69, ifelse(grade == 7, 64, 56))
  min_score <- 10
  max_score <- 100

  # Generate math scores based on desired average and minimum score
  scores <- rnorm(n, mean = avg_score, sd = 17)
  scores <- pmax(pmin(scores, max_score), min_score)  # Ensure no score is below the minimum

  return(scores)
}

# Replace math scores by grade level
exams$math_score <- ifelse(exams$grade == 8,
                           generate_math_scores(8, sum(exams$grade == 8)),
                           ifelse(exams$grade == 7,
                                  generate_math_scores(7, sum(exams$grade == 7)),
                                  generate_math_scores(6, sum(exams$grade == 6))
                           )
)

# Check the updated distribution of math scores by grade
boxplot(math_score ~ grade, data = exams,
        main = "Adjusted Math Scores by Grade", ylab = "Math Score", xlab = "Grade Level")
```

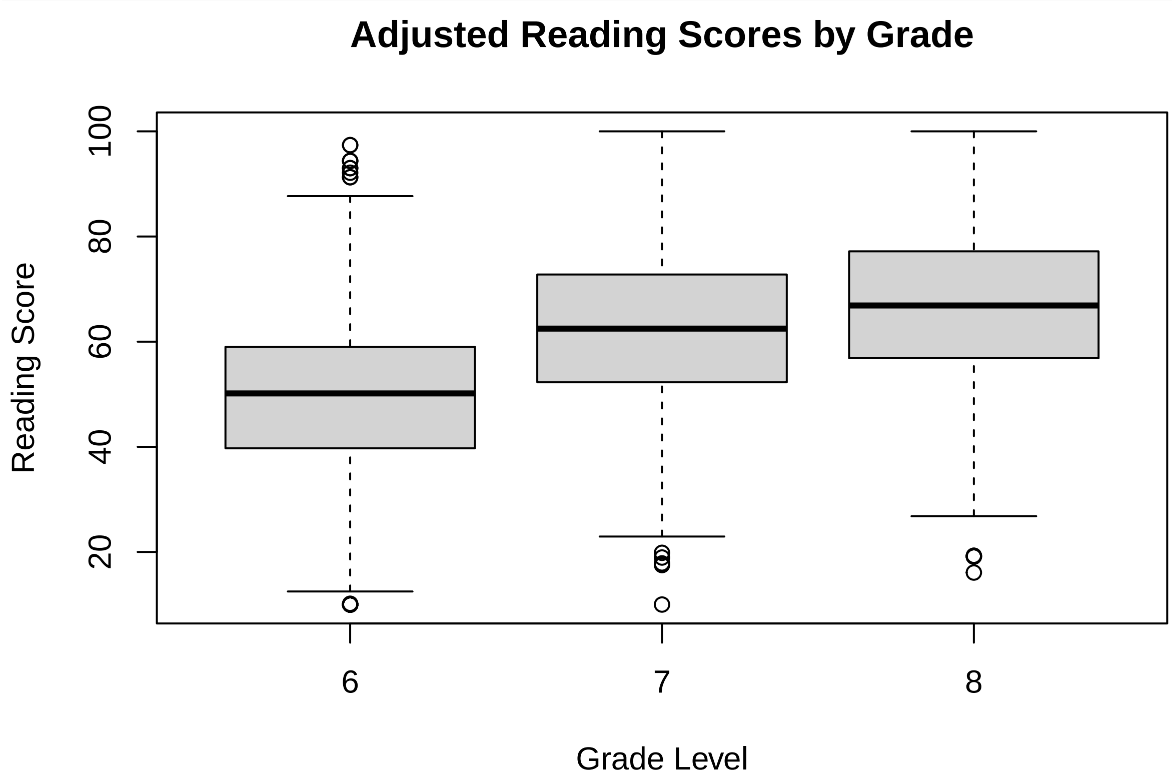


Reading scores

```r
boxplot(reading_score ~ grade, data = exams,
        main = "Reading Scores by Grade", ylab = "Reading Score", xlab = "Grade Level")
```



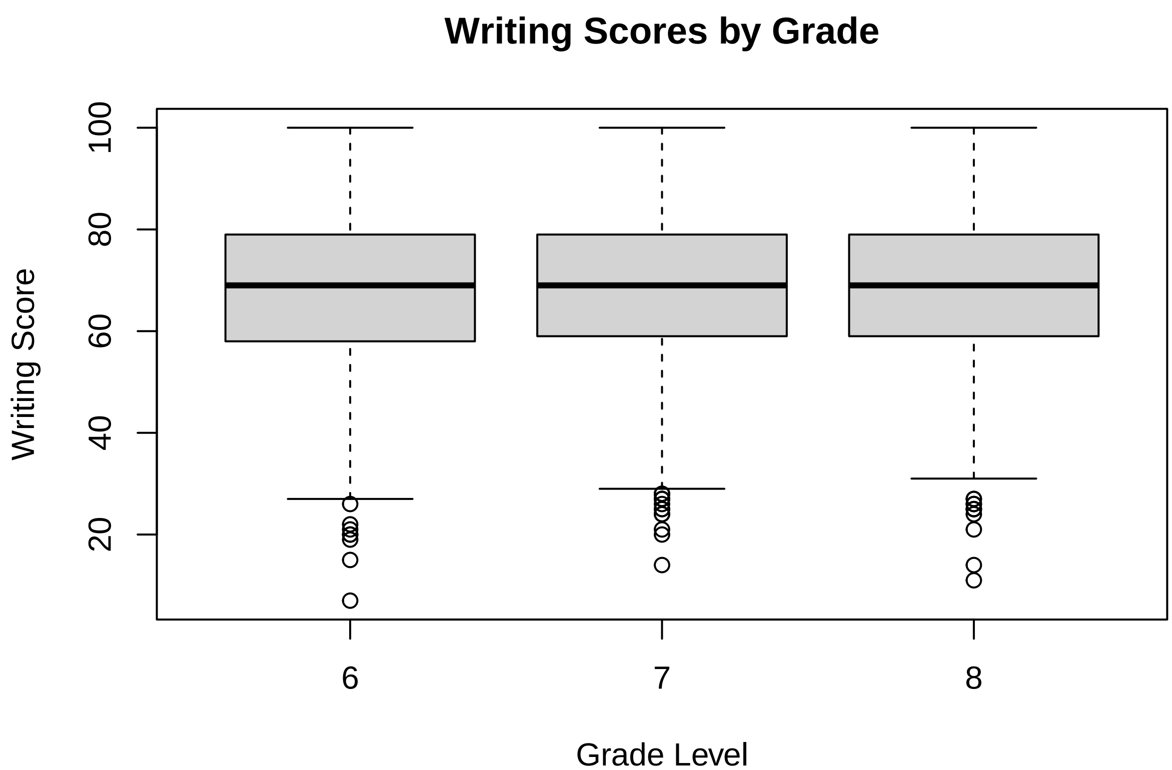
```r
set.seed(121619)  # For reproducibility

# Function to generate reading scores for each grade with desired averages and a minimum score of 10
generate_reading_scores <- function(grade, n) {
  avg_score <- ifelse(grade == 8, 67, ifelse(grade == 7, 62, 50))
  min_score <- 10
  max_score <- 100

  # Generate reading scores based on desired average and minimum score
  scores <- rnorm(n, mean = avg_score, sd = 15)
  scores <- pmax(pmin(scores, max_score), min_score)  # Ensure no score is below the minimum

  return(scores)
}

# Replace reading scores by grade level
exams$reading_score <- ifelse(exams$grade == 8,
                              generate_reading_scores(8, sum(exams$grade == 8)),
                              ifelse(exams$grade == 7,
                                     generate_reading_scores(7, sum(exams$grade == 7)),
                                     generate_reading_scores(6, sum(exams$grade == 6))
                              )
)

# Check the updated distribution of reading scores by grade
boxplot(reading_score ~ grade, data = exams,
        main = "Adjusted Reading Scores by Grade", ylab = "Reading Score", xlab = "Grade Level")
```



###

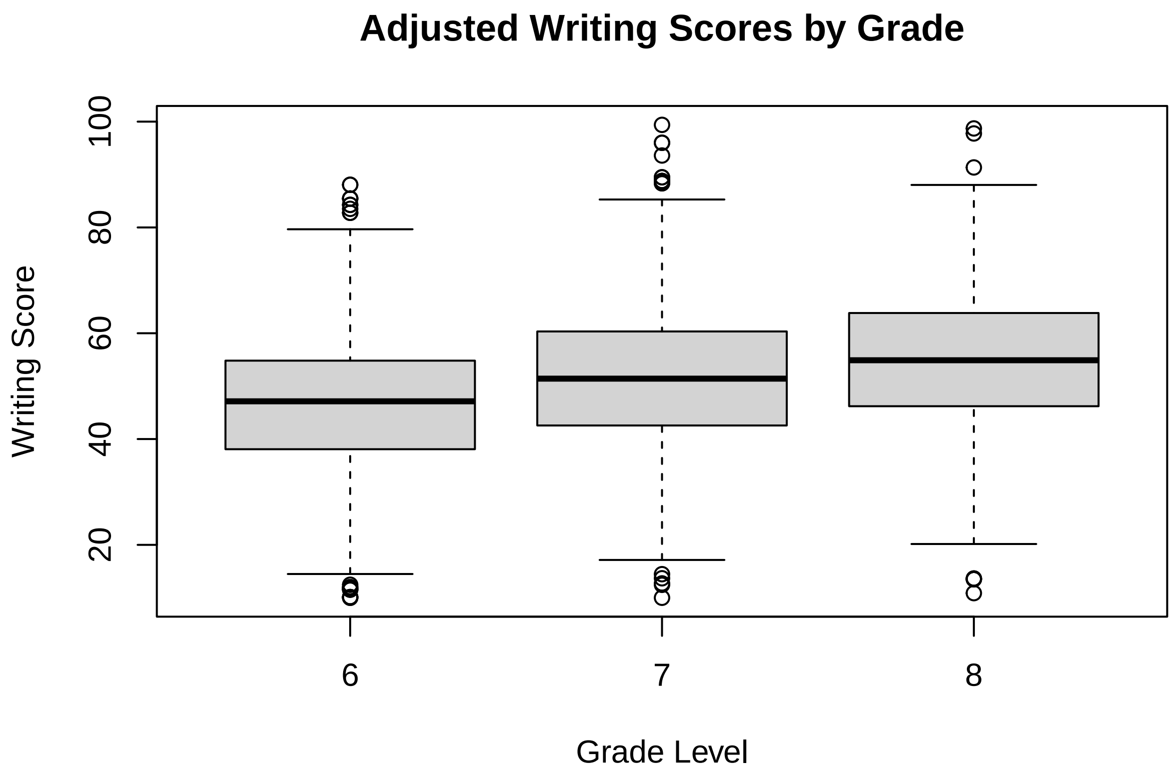
Writing scores

```
boxplot(writing_score ~ grade, data = exams,
        main = "Writing Scores by Grade", ylab = "Writing Score", xlab = "Grade Level")
```




```
set.seed(121619)  # For reproducibility

# Function to generate writing scores for each grade with desired averages and a minimum score of 10
generate_writing_scores <- function(grade, n) {
  avg_score <- ifelse(grade == 8, 55, ifelse(grade == 7, 51, 47))
  min_score <- 10
  max_score <- 100

  # Generate writing scores based on desired average and minimum score
  scores <- rnorm(n, mean = avg_score, sd = 13)
  scores <- pmax(pmin(scores, max_score), min_score)  # Ensure no score is below the minimum or maximum

  return(scores)
}

# Replace writing scores by grade level
exams$writing_score <- ifelse(exams$grade == 8,
                              generate_writing_scores(8, sum(exams$grade == 8)),
                              ifelse(exams$grade == 7,
                                     generate_writing_scores(7, sum(exams$grade == 7)),
                                     generate_writing_scores(6, sum(exams$grade == 6))
                              )
)

# Check the updated distribution of writing scores by grade
boxplot(writing_score ~ grade, data = exams,
```

```
        main = "Adjusted Writing Scores by Grade", ylab = "Writing Score", xlab = "Grade Level")
```



Renaming gender column

```
exams <- exams %>%
  rename(sex = gender)
```

Save the final dataset

```
write.csv(exams, "final_exams.csv", row.names = FALSE)
```

Mental Health Data

Research Questions

Categorical ordinal variable: mental health
What factors are the most important for predicting student mental health rating?

Are there any variables that negatively impact student mental health ratings? What about positively impact mental health?

Data processing:

Factoring and Reordering

```
require(recipes)
```

```
## Loading required package: recipes
```

```
##
## Attaching package: 'recipes'

## The following object is masked from 'package:stringr':
##
##     fixed

## The following object is masked from 'package:stats':
##
##     step
```

```r
# Check levels of the 'mental_health' column
levels(as.factor(exams$mental_health))
```

```
## [1] "excellent" "fair"      "good"      "poor"      "very good"
```

```r
# Define the order of the levels
new_order <- rev(c("excellent", "very good", "good", "fair", "poor"))

# Reorder the levels of the 'mental_health' factor variable
exams$mental_health <- factor(exams$mental_health, levels = new_order)

# Check the updated levels
levels(exams$mental_health)
```

```
## [1] "poor"      "fair"      "good"      "very good" "excellent"
```

```r
#########################################

# Parent level of education
levels(as.factor(exams$parental_level_of_education))
```

```
## [1] "associate's degree" "bachelor's degree"  "high school"
## [4] "master's degree"    "some college"       "some high school"
```

```r
new_order <- c("some high school", "high school", "associate's degree", "some college",
                   "bachelor's degree", "master's degree")

exams$parental_level_of_education <- factor(exams$parental_level_of_education, levels = new_order)

levels(exams$parental_level_of_education)
```

```
## [1] "some high school"   "high school"        "associate's degree"
## [4] "some college"       "bachelor's degree"  "master's degree"
```

```r
##########################################


levels(as.factor(exams$belonging))
```

```
## [1] "mostly unwelcome"      "mostly welcome"        "very unwelcome"
## [4] "very welcome"          "welcome half the time"
```

```r
new_order <- c("very unwelcome", "mostly unwelcome", "welcome half the time",
               "mostly welcome", "very welcome")

exams$belonging <- factor(exams$belonging, levels = new_order)

levels(exams$belonging)
```

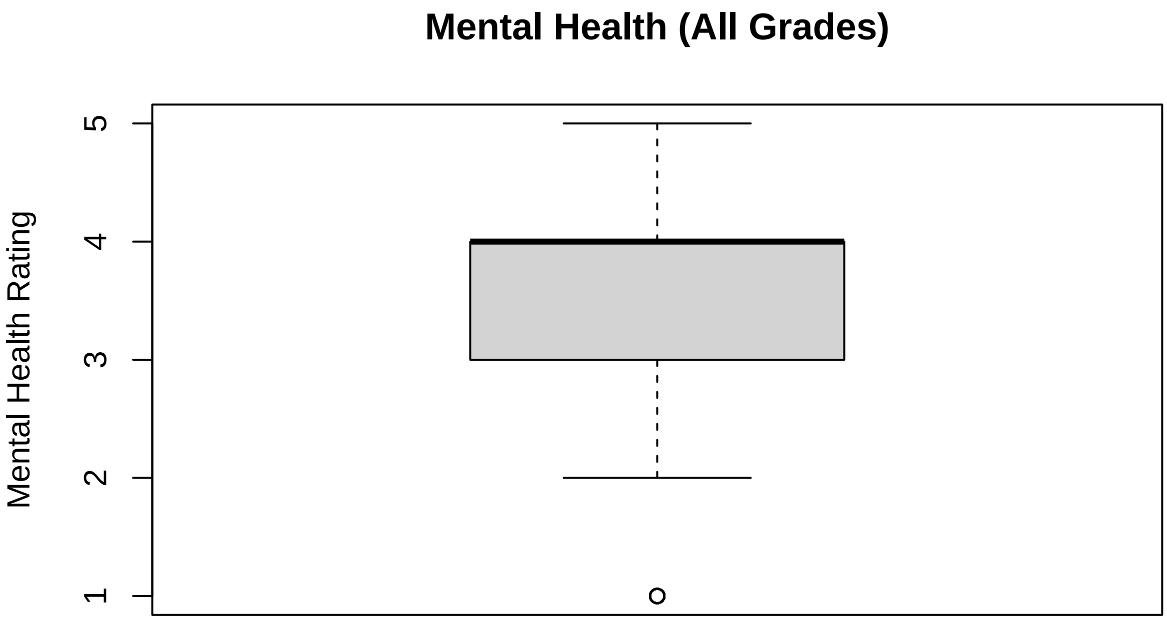
```
## [1] "very unwelcome"        "mostly unwelcome"      "welcome half the time"
## [4] "mostly welcome"        "very welcome"
```

Here is the distribution of the outcome variable of mental health rating

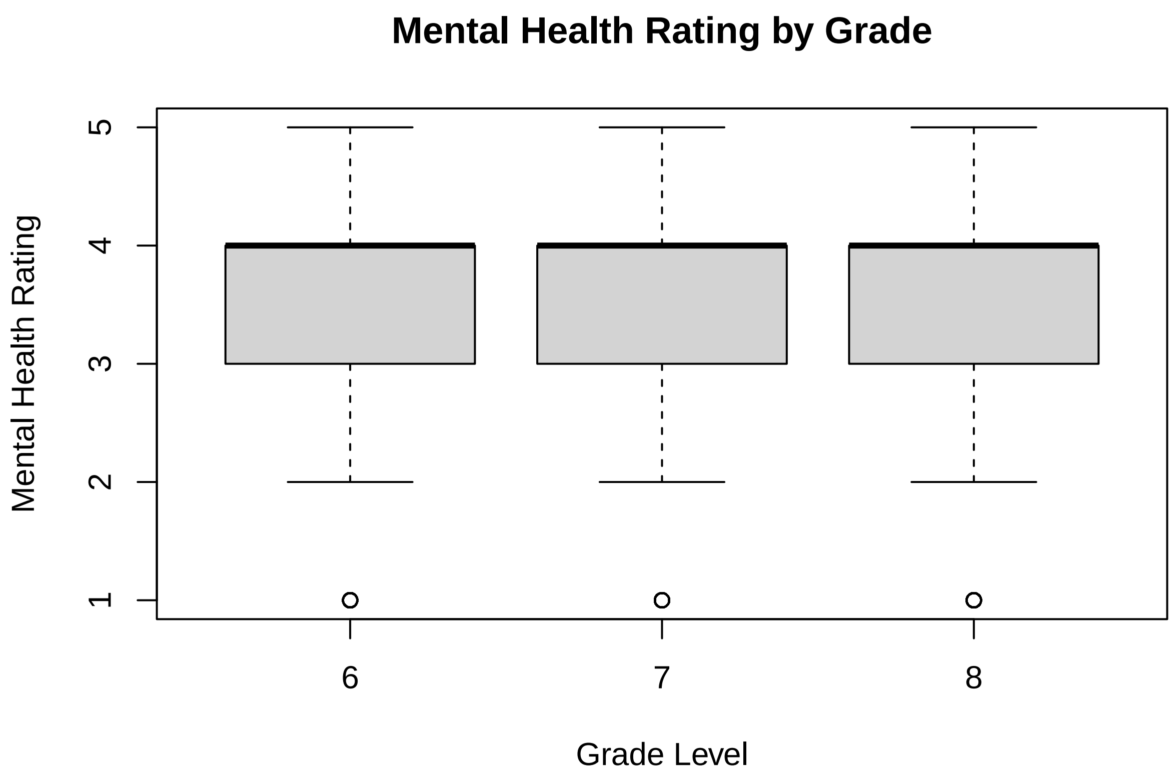
```r
summary(exams$mental_health)
```

```
##      poor      fair      good very good excellent
##       395       351      1514      2004       736
```

```r
# General boxplot for Mental Health Ratings
boxplot(exams$mental_health, main = "Mental Health (All Grades)", ylab = "Mental Health Rating")
```




```r
# Boxplot for Mental Health ratings by grade
boxplot(mental_health ~ grade, data = exams,
        main = "Mental Health Rating by Grade", ylab = "Mental Health Rating", xlab = "Grade Level")
```



Model Variables

```
outcome <- 'mental_health'
id <- 'id'

categorical <- c('sex', 'lunch', 'test_preparation_course',
                 'school_location', 'school_type', 'race', 'el_status',
                 'home_language', 'grade', 'trusted_adult', 'alcohol_use', 'marijuana_use',
                'parental_level_of_education', 'belonging')

numeric <- c('math_score', 'reading_score', 'writing_score', 'parent_income', 'age', 'stress_rating', '
```

```
exams <- exams %>%
  mutate(across(c('sex', 'lunch', 'test_preparation_course',
                 'school_location', 'school_type', 'race', 'el_status',
                 'home_language', 'grade', 'trusted_adult', 'alcohol_use', 'marijuana_use'), as.factor))

# Check complete
# str(exams)
```

Preparing the data

Time for the recipe

```
all_exam_pred <- c(categorical, numeric)


blueprint_exams <- recipe(x = exams) %>%
  update_role(id, new_role = "id") %>%
  update_role(outcome, new_role = "outcome") %>%
```

```r
  update_role(all_exam_pred, new_role = "predictor") %>%
  step_indicate_na(all_of(categorical),all_of(numeric)) %>%
  step_zv(all_numeric()) %>%
  step_impute_mean(all_of(numeric)) %>%
  step_impute_mode(all_of(categorical)) %>%
  step_poly(all_of(numeric),degree=2) %>%
  step_normalize(paste0(numeric,'_poly_1'),
                 paste0(numeric,'_poly_2')) %>%
  step_dummy(all_of(categorical),one_hot=TRUE)
```

```
## Warning: Using an external vector in selections was deprecated in tidyselect 1.1.0.
## i Please use `all_of()` or `any_of()` instead.
##   # Was:
##   data %>% select(outcome)
##
##   # Now:
##   data %>% select(all_of(outcome))
##
## See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
## Warning: Using an external vector in selections was deprecated in tidyselect 1.1.0.
## i Please use `all_of()` or `any_of()` instead.
##   # Was:
##   data %>% select(all_exam_pred)
##
##   # Now:
##   data %>% select(all_of(all_exam_pred))
##
## See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```r
# Summary of the blueprint
summary(blueprint_exams)
```

```
## # A tibble: 26 x 4
##    variable                    type      role      source
##    <chr>                       <list>    <chr>     <chr>
##  1 sex                         <chr [3]> predictor original
##  2 parental_level_of_education <chr [3]> predictor original
##  3 lunch                       <chr [3]> predictor original
##  4 test_preparation_course     <chr [3]> predictor original
##  5 math_score                  <chr [2]> predictor original
##  6 reading_score               <chr [2]> predictor original
##  7 writing_score               <chr [2]> predictor original
##  8 id                          <chr [2]> id        original
##  9 parent_income               <chr [2]> predictor original
## 10 school_location             <chr [3]> predictor original
## # i 16 more rows
```

Split data for testing

```
set.seed(121619)

loc       <- sample(1:nrow(exams), round(nrow(exams) * 0.9))
exams_train  <- exams[loc, ]
exams_test  <- exams[-loc, ]

dim(exams_train)
```

```
## [1] 4500   26
```

```
dim(exams_test)
```

```
## [1] 500  26
```

Cross Folding

```
#install.packages("caret")
require(caret)
```

```
## Loading required package: caret
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
exams_train = exams_train[sample(nrow(exams_train)),]

# Create 10 folds with equal size

    folds = cut(seq(1,nrow(exams_train)),breaks=10,labels=FALSE)

# Create the list for each fold

    my.indices <- vector('list',10)

    for(i in 1:10){
        my.indices[[i]] <- which(folds!=i)
    }
```

Prepared Data

```
prepare_exams <- prep(blueprint_exams,
                training = exams_train)
prepare_exams
```

```
##
```

```
## -- Recipe ----------------------------------------------------------------------
```

```
##
```

```
## -- Inputs

## Number of variables by role

## outcome:    1
## predictor: 24
## id:         1

##

## -- Training information

## Training data contained 4500 data points and no incomplete rows.

##

## -- Operations

## * Creating missing data variable indicators for: sex, lunch, ... | Trained

## * Zero variance filter removed: na_ind_sex, na_ind_lunch, ... | Trained

## * Mean imputation for: math_score, reading_score, writing_score, ... | Trained

## * Mode imputation for: sex, lunch, test_preparation_course, ... | Trained

## * Orthogonal polynomials on: math_score, reading_score, ... | Trained

## * Centering and scaling for: math_score_poly_1, ... | Trained

## * Dummy variables from: sex, lunch, test_preparation_course, ... | Trained
```

Baking data

```r
baked_train <- bake(prepare_exams, new_data = exams_train)

baked_test <- bake(prepare_exams, new_data = exams_test)

dim(baked_train)
```

```
## [1] 4500   67
```

```r
dim(baked_test)
```

```
## [1] 500  67
```

Data Analysis

Polynomial Logit Modeling

Finally done preparing the data

Tried to use a linear model without realizing that it is an ordinal categorical variable. Instead I am using logit modeling while taking into account the ordinal categorical nature of the outcome variable.

I attempted to use a ordinal model with caret to use cross validation but I struggled to get the model to converge. Instead I am going to use the polr function, though the results will likely not be as accurate.

```r
library(MASS)
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

```
library(DT)
#install.packages("ordinalNet")
#require(ordinalNet)

pol_mod <- polr(mental_health ~ ., data = exams_train)

### I wanted to use the caret model but I am not having much success.
# pol_log <- caret::train(
#   mental_health ~ .,
#   data = exams_train,
#   method = "polr",
#   trControl = cv)

### Then I tried to use a penalized ordinal model but I could not get the model to converge.
# grid <- expand.grid(
#   alpha = seq(0, 1, by = 0.1),
#   lambda = seq(0,1, by = 0.1),
#   criteria = c("AUC"),
#   link = c("probit"),
#   modeltype = "ordinalNet",
#   family = "acat")
#
# pol_mod <- caret::train(
#   mental_health ~ .,
#   data = exams_train,
#   method = "ordinalNet",
#   tuneGrid = grid,
#   trControl = cv,
#   parallelTerms = T)

#pol_mod$coefficients


# Here is a look at the coefficients
options(scipen=99)
coefficients <- coef(pol_mod)
zero_coef_vars <- names(coefficients[coefficients == 0])
cat(length(zero_coef_vars), "predictors had a value of zero for the coefficient with non-penalty regres
```

```
## 0 predictors had a value of zero for the coefficient with non-penalty regression
```

```
# Here we can see the ordered coefficients
coef_names <- names(coefficients)
coef_df <- data.frame(variable = coef_names, coefficient = coefficients)
sorted_coefs <- coef_df[order(coef_df$coefficient, decreasing = TRUE), ]
DT::datatable(sorted_coefs, rownames = FALSE)
```

```
## PhantomJS not found. You can install it with webshot::install_phantomjs(). If it is installed, pleas
```

```
# Cutoffs for Mental health
print(pol_mod$zeta)
```

```
##           poor|fair           fair|good      good|very good very good|excellent
##           -5.868167           -4.725601           -1.689457            2.224431
```

```r
#While not quite the same as in a binary model, here is manually calculated accuracy.

predicted_values <- predict(pol_mod, exams_test, type = "class")

# Calculate accuracy by comparing predicted vs. actual categories
accuracy <- mean(predicted_values == exams_test$mental_health)
cat("The Polr model accuracy is:", accuracy, "\n")
```

```
## The Polr model accuracy is: 0.774
```

```r
print(pol_mod)
```

```
## Call:
## polr(formula = mental_health ~ ., data = exams_train)
##
## Coefficients:
##                                      sexmale
##                              -0.0051592031182
##        parental_level_of_educationhigh school
##                              -4.6498963646748
## parental_level_of_educationassociate's degree
##                              -9.1175102193962
##       parental_level_of_educationsome college
##                              -6.8053607584981
##  parental_level_of_educationbachelor's degree
##                             -13.5954785736434
##    parental_level_of_educationmaster's degree
##                             -15.9343070388436
##                                 lunchReduced
##                              -4.5780232006240
##                                lunchStandard
##                              -9.2811562223370
##                   test_preparation_coursenone
##                              -0.0609344689102
##                                   math_score
##                              -0.0775762043523
##                                reading_score
##                               0.2049344381751
##                                writing_score
##                              -0.1356503350875
##                                           id
##                              -0.0000279689479
##                                parent_income
##                              -0.0000002242681
##                      school_locationSuburban
##                              -0.0574664200986
##                         school_locationUrban
##                              -0.1685996379063
##                            school_typePublic
##                              -0.0922024505574
##                                    raceAsian
##                               0.0734642778069
```

```
##                                    raceBlack
##                              -0.0514462913091
##                                   raceLatine
##                              -0.0414537863883
##                                    raceWhite
##                               0.0223160763400
##                              el_statusnot EL
##                              -0.0093652073936
##                          home_languageEnglish
##                               0.5641251575563
##                           home_languageFrench
##                               0.4507011279708
##                           home_languageKorean
##                               0.5301432036847
##                            home_languageOther
##                               0.5832007427960
##                          home_languageSpanish
##                               0.5749052806357
##                                       grade7
##                              -1.2270163448626
##                                       grade8
##                              -1.2829259626896
##                                          age
##                              -0.0164939141965
##                                close_friends
##                               0.0142143126011
##                                trusted_adult1
##                               0.1831257083494
##                                stress_rating
##                               0.0059069696587
##                     belongingmostly unwelcome
##                              -0.0660085330433
##                belongingwelcome half the time
##                              -0.2647291332250
##                       belongingmostly welcome
##                               0.4367579297134
##                         belongingvery welcome
##                               0.1615045460194
##                                    ses_score
##                               2.8000859976164
##                                 alcohol_use1
##                               0.0263960006384
##                                marijuana_use1
##                              -0.3666715908016
##                                     siblings
##                              -2.1220142604356
##                                         pets
##                              -0.0169855025047
##
## Intercepts:
##          poor|fair           fair|good      good|very good very good|excellent
##          -5.868167           -4.725601           -1.689457            2.224431
##
## Residual Deviance: 8449.392
```

```
## AIC: 8541.392
```

Tree Modeling

```r
#install.packages("rpart")
require(rpart)
```

```
## Loading required package: rpart
```

```r
grid <- data.frame(cp=seq(0,0.05,.001))

# Specify the trainControl for cross-validation
cv <- trainControl(method = "cv", number = 10)

# Train the model
mod <- caret::train(blueprint_exams,
             data = exams_train,
             method = "rpart",
             tuneGrid = grid,
             trControl = cv,
             control = list(minsplit = 10,
                            minbucket = 2,
                            maxdepth = 20))


# Get predictions
predictions <- predict(mod, newdata = exams_test)

# Evaluate the model
results <- confusionMatrix(predictions, exams_test$mental_health)
print(results)
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  poor fair good very good excellent
##   poor        32   32    0         0         0
##   fair         0    0    0         0         0
##   good         0    0   97         0         0
##   very good    0    0   53       192         0
##   excellent    8    9    0         0        77
##
## Overall Statistics
##
##                Accuracy : 0.796
##                  95% CI : (0.758, 0.8305)
##     No Information Rate : 0.384
##     P-Value [Acc > NIR] : < 0.00000000000000022
##
##                   Kappa : 0.7145
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
```

```
##                      Class: poor Class: fair Class: good Class: very good
## Sensitivity               0.8000       0.000      0.6467           1.0000
## Specificity               0.9304       1.000      1.0000           0.8279
## Pos Pred Value            0.5000         NaN      1.0000           0.7837
## Neg Pred Value            0.9817       0.918      0.8685           1.0000
## Prevalence                0.0800       0.082      0.3000           0.3840
## Detection Rate            0.0640       0.000      0.1940           0.3840
## Detection Prevalence      0.1280       0.000      0.1940           0.4900
## Balanced Accuracy         0.8652       0.500      0.8233           0.9140
##                      Class: excellent
## Sensitivity                    1.0000
## Specificity                    0.9598
## Pos Pred Value                 0.8191
## Neg Pred Value                 1.0000
## Prevalence                     0.1540
## Detection Rate                 0.1540
## Detection Prevalence           0.1880
## Balanced Accuracy              0.9799
```

Accuracy peaks at 0.8055619 with a cp of 0.025

```
mod$results
```

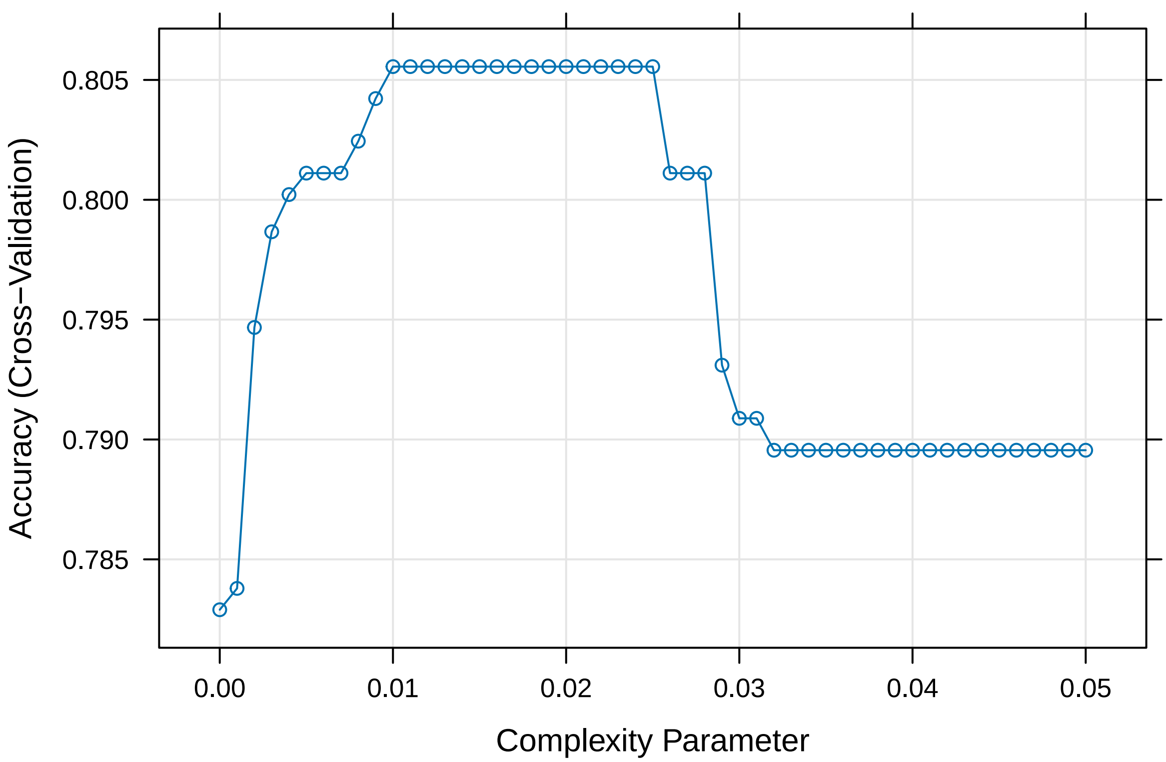
```
##       cp  Accuracy     Kappa  AccuracySD    KappaSD
## 1  0.000 0.7828977 0.6954526 0.014220108 0.01997193
## 2  0.001 0.7837876 0.6955342 0.013522413 0.01874037
## 3  0.002 0.7946731 0.7082311 0.013252021 0.01818453
## 4  0.003 0.7986647 0.7124884 0.009294851 0.01366837
## 5  0.004 0.8002178 0.7146055 0.011276654 0.01645524
## 6  0.005 0.8011106 0.7158796 0.011138831 0.01625712
## 7  0.006 0.8011106 0.7158796 0.011138831 0.01625712
## 8  0.007 0.8011106 0.7158796 0.011138831 0.01625712
## 9  0.008 0.8024449 0.7177541 0.011129723 0.01624274
## 10 0.009 0.8042227 0.7202639 0.010307472 0.01508293
## 11 0.010 0.8055561 0.7221494 0.009578980 0.01405186
## 12 0.011 0.8055561 0.7221494 0.009578980 0.01405186
## 13 0.012 0.8055561 0.7221494 0.009578980 0.01405186
## 14 0.013 0.8055561 0.7221494 0.009578980 0.01405186
## 15 0.014 0.8055561 0.7221494 0.009578980 0.01405186
## 16 0.015 0.8055561 0.7221494 0.009578980 0.01405186
## 17 0.016 0.8055561 0.7221494 0.009578980 0.01405186
## 18 0.017 0.8055561 0.7221494 0.009578980 0.01405186
## 19 0.018 0.8055561 0.7221494 0.009578980 0.01405186
## 20 0.019 0.8055561 0.7221494 0.009578980 0.01405186
## 21 0.020 0.8055561 0.7221494 0.009578980 0.01405186
## 22 0.021 0.8055561 0.7221494 0.009578980 0.01405186
## 23 0.022 0.8055561 0.7221494 0.009578980 0.01405186
## 24 0.023 0.8055561 0.7221494 0.009578980 0.01405186
## 25 0.024 0.8055561 0.7221494 0.009578980 0.01405186
## 26 0.025 0.8055561 0.7221494 0.009578980 0.01405186
## 27 0.026 0.8011116 0.7160656 0.016682891 0.02344794
## 28 0.027 0.8011116 0.7160656 0.016682891 0.02344794
## 29 0.028 0.8011116 0.7160656 0.016682891 0.02344794
## 30 0.029 0.7930997 0.7054386 0.018115025 0.02517685
## 31 0.030 0.7908825 0.7025065 0.017560104 0.02435993
```

```
## 32 0.031 0.7908825 0.7025065 0.017560104 0.02435993
## 33 0.032 0.7895521 0.7008303 0.018181479 0.02523394
## 34 0.033 0.7895521 0.7008303 0.018181479 0.02523394
## 35 0.034 0.7895521 0.7008303 0.018181479 0.02523394
## 36 0.035 0.7895521 0.7008303 0.018181479 0.02523394
## 37 0.036 0.7895521 0.7008303 0.018181479 0.02523394
## 38 0.037 0.7895521 0.7008303 0.018181479 0.02523394
## 39 0.038 0.7895521 0.7008303 0.018181479 0.02523394
## 40 0.039 0.7895521 0.7008303 0.018181479 0.02523394
## 41 0.040 0.7895521 0.7008303 0.018181479 0.02523394
## 42 0.041 0.7895521 0.7008303 0.018181479 0.02523394
## 43 0.042 0.7895521 0.7008303 0.018181479 0.02523394
## 44 0.043 0.7895521 0.7008303 0.018181479 0.02523394
## 45 0.044 0.7895521 0.7008303 0.018181479 0.02523394
## 46 0.045 0.7895521 0.7008303 0.018181479 0.02523394
## 47 0.046 0.7895521 0.7008303 0.018181479 0.02523394
## 48 0.047 0.7895521 0.7008303 0.018181479 0.02523394
## 49 0.048 0.7895521 0.7008303 0.018181479 0.02523394
## 50 0.049 0.7895521 0.7008303 0.018181479 0.02523394
## 51 0.050 0.7895521 0.7008303 0.018181479 0.02523394
```

```r
mod$bestTune # 0.025
```

```
##       cp
## 26 0.025
```

```r
plot(mod)
```



Tree Plot

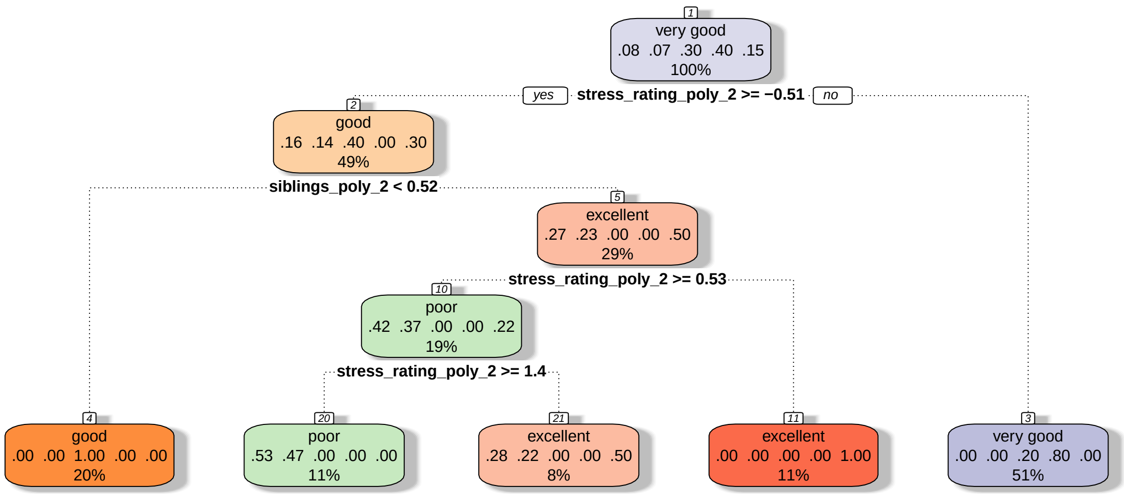
```r
#install.packages("rattle")
require(rattle)
```

```
## Loading required package: rattle
```

```
## Loading required package: bitops
```

```
## Rattle: A free graphical interface for data science with R.
## Version 5.5.1 Copyright (c) 2006-2021 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```r
fancyRpartPlot(mod$finalModel,type=2,sub='')
```



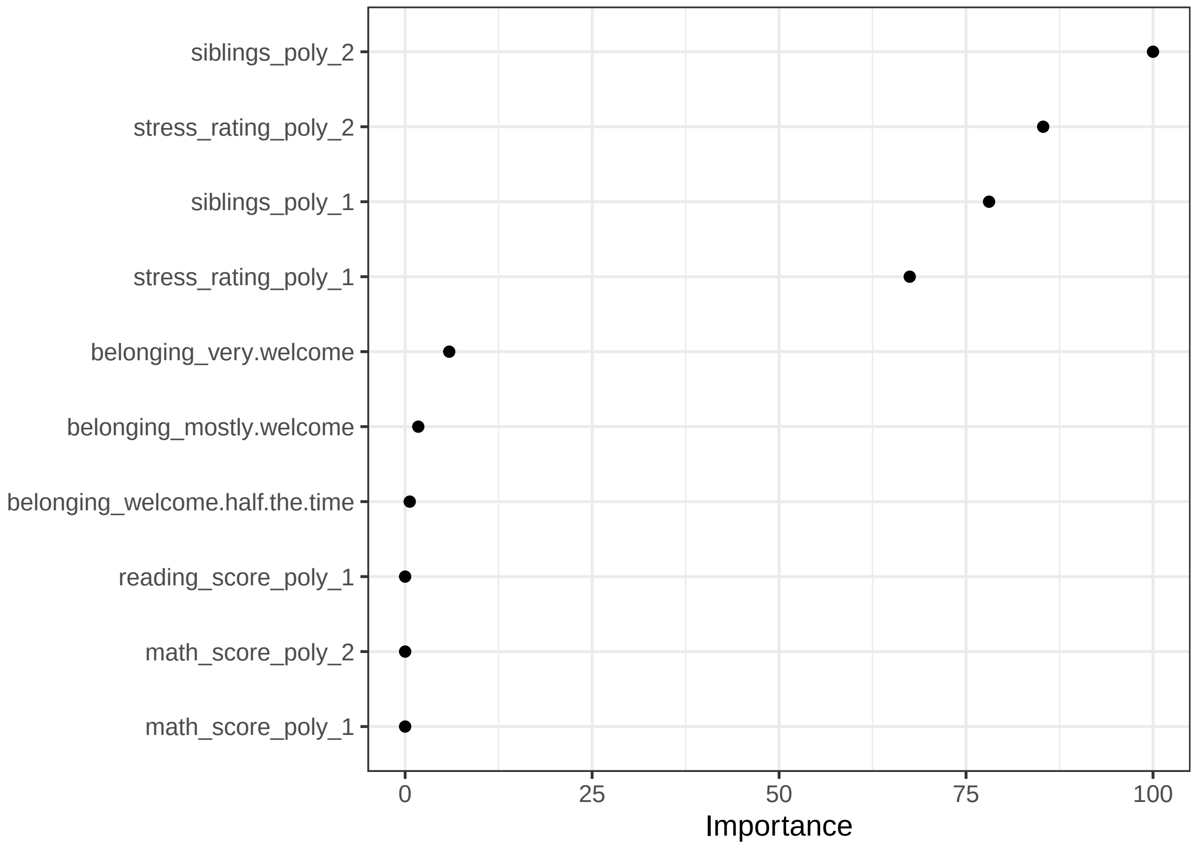

```r
# install.packages("vip")
require(vip)
```

```
## Loading required package: vip
```

```
##
## Attaching package: 'vip'
```

```
## The following object is masked from 'package:utils':
##
##     vi
```

```r
vip(mod,
    num_features = 10,
    geom = "point") +
  theme_bw()
```



Alcohol Use Data

Research Questions

Binary variable: Alcohol Use
What factors are most important for predicting student use of alcohol?

What factors predict alcohol use positively and negatively?

Data Preparation:

Data Descriptives:

First here is the distribution of the outcome variable of Alcohol use

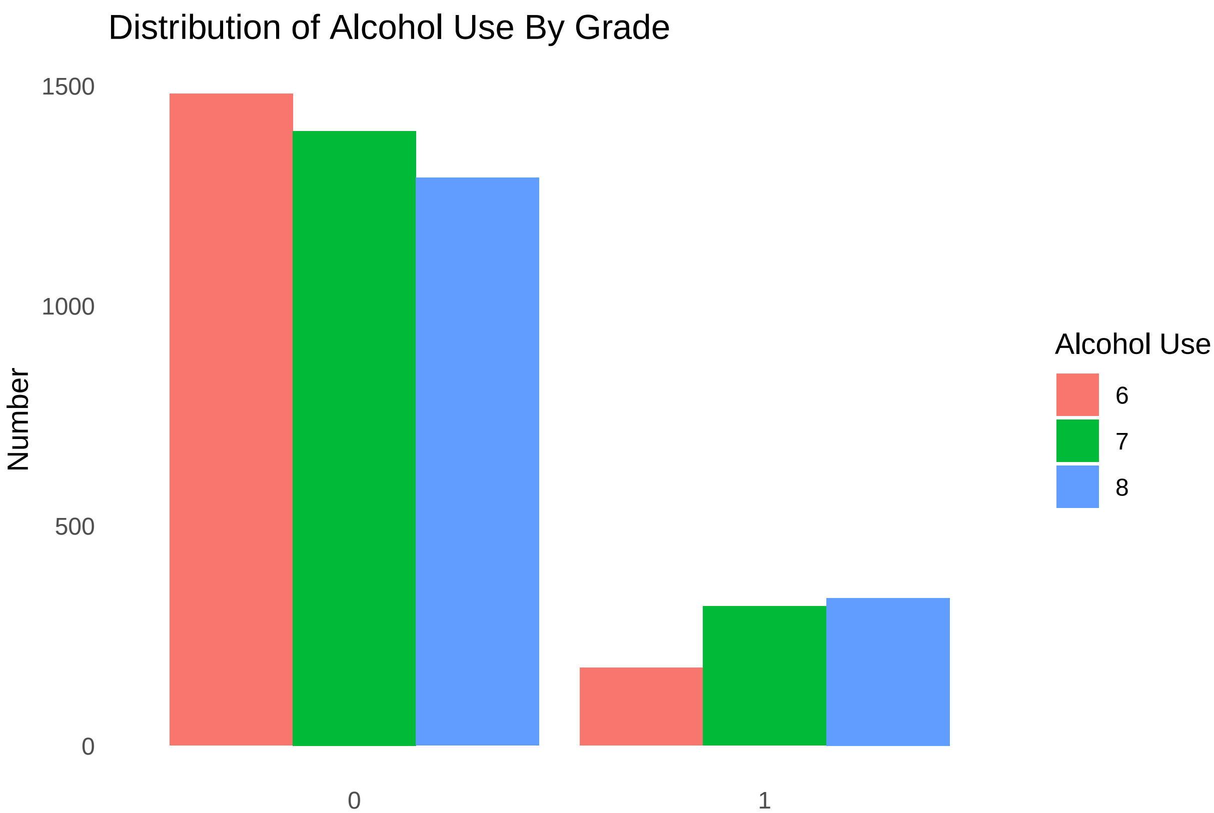
```
exams <- exams %>%
  mutate(across(c('sex', 'lunch', 'test_preparation_course',
                'school_location', 'school_type', 'race', 'el_status',
                'home_language', 'grade', 'trusted_adult', 'alcohol_use', 'marijuana_use'), as.factor))


summary(exams$alcohol_use)
```

```
##    0    1
## 4170  830
```

```
al_tab_gr <- as.data.table(table(exams$alcohol_use, exams$grade))
al_tab_gr
```

```
##    V1 V2    N
## 1:  0  6 1482
## 2:  1  6  177
## 3:  0  7 1397
## 4:  1  7  317
## 5:  0  8 1291
## 6:  1  8  336
```

```
al_tab_gr %>%
ggplot(aes(x = V1, y = N, fill = V2)) +
  geom_col(position = "dodge") +
  labs(title = "Distribution of Alcohol Use By Grade",
       x = "",
       y = "Number") +
  scale_fill_discrete(name = "Alcohol Use") +
  theme_minimal() +
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.background = element_blank())
```



Luckily not many students reported using alcohol, but I am still curious if we can predict who used alcohol.

Model Variables

```
exams <- exams %>%
  mutate(alcohol_use = recode_factor(alcohol_use,
      '0' = 'never used',
      '1' = 'used'))
```

```r
exams <- exams %>%
  mutate(marijuana_use = recode_factor(marijuana_use,
      '0' = 'never used',
      '1' = 'used'))

outcome <- 'alcohol_use'
id <- 'id'

categorical <- c('sex', 'lunch', 'test_preparation_course',
                 'school_location', 'school_type', 'race', 'el_status',
                 'home_language', 'grade', 'trusted_adult', 'marijuana_use', 'parental_level_of_educati

numeric <- c('math_score', 'reading_score', 'writing_score', 'parent_income', 'age', 'stress_rating', '
```

```r
# Check complete
#str(exams)
```

Preparing the data

Time for the recipe

```r
all_exam_pred <- c(categorical, numeric)


blueprint_exams <- recipe(x = exams) %>%
  update_role(id, new_role = "id") %>%
  update_role(outcome, new_role = "outcome") %>%
  update_role(all_exam_pred, new_role = "predictor") %>%
  step_indicate_na(all_of(categorical),all_of(numeric)) %>%
  step_zv(all_numeric()) %>%
  step_impute_mean(all_of(numeric)) %>%
  step_impute_mode(all_of(categorical)) %>%
  step_poly(all_of(numeric),degree=2) %>%
  step_normalize(paste0(numeric,'_poly_1'),
                 paste0(numeric,'_poly_2')) %>%
  step_dummy(all_of(categorical),one_hot=TRUE)

# Summary of the blueprint
summary(blueprint_exams)
```

```
## # A tibble: 26 x 4
##    variable                    type      role      source
##    <chr>                       <list>    <chr>     <chr>
##  1 sex                         <chr [3]> predictor original
##  2 parental_level_of_education <chr [3]> predictor original
##  3 lunch                       <chr [3]> predictor original
##  4 test_preparation_course     <chr [3]> predictor original
##  5 math_score                  <chr [2]> predictor original
##  6 reading_score               <chr [2]> predictor original
##  7 writing_score               <chr [2]> predictor original
##  8 id                          <chr [2]> id        original
##  9 parent_income               <chr [2]> predictor original
## 10 school_location             <chr [3]> predictor original
## # i 16 more rows
```

Split data for testing

```r
set.seed(121619)

loc       <- sample(1:nrow(exams), round(nrow(exams) * 0.9))
exams_train  <- exams[loc, ]
exams_test  <- exams[-loc, ]

dim(exams_train)
```

```
## [1] 4500   26
```

```r
dim(exams_test)
```

```
## [1] 500  26
```

It might make more sense to try this with cross folding.

```r
exams_train = exams_train[sample(nrow(exams_train)),]

# Create 10 folds with equal size

    folds = cut(seq(1,nrow(exams_train)),breaks=10,labels=FALSE)

# Create the list for each fold

    my.indices <- vector('list',10)

    for(i in 1:10){
        my.indices[[i]] <- which(folds!=i)
    }
```

```r
prepare <- prep(blueprint_exams,
                training = exams_train)
prepare
```

```
##
## -- Recipe ----------------------------------------------------------------------
##
## -- Inputs
## Number of variables by role
## outcome:    1
## predictor: 24
## id:         1
##
## -- Training information
## Training data contained 4500 data points and no incomplete rows.
##
## -- Operations
## * Creating missing data variable indicators for: sex, lunch, ... | Trained
```

```
## * Zero variance filter removed: na_ind_sex, na_ind_lunch, ... | Trained
## * Mean imputation for: math_score, reading_score, writing_score, ... | Trained
## * Mode imputation for: sex, lunch, test_preparation_course, ... | Trained
## * Orthogonal polynomials on: math_score, reading_score, ... | Trained
## * Centering and scaling for: math_score_poly_1, ... | Trained
## * Dummy variables from: sex, lunch, test_preparation_course, ... | Trained
```

Data Analysis Alcohol

Logit Modeling

There were a lot of warnings, I think I will have to examine the data using a penalty term.

```
cv <- trainControl(method = "repeatedcv", number = 10, repeats = 3)

mod_log <- caret::train(
  alcohol_use ~ .,
  data = exams_train,
  method = "glm",
  trControl = cv,
  family = binomial(link = "logit"),
  metric = "Accuracy")

print(mod_log)
```

```
## Generalized Linear Model
##
## 4500 samples
##   25 predictor
##    2 classes: 'never used', 'used'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 4050, 4050, 4050, 4050, 4050, 4050, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.8341486  -0.0005821386
```

```
predicted_test <- predict(mod_log, exams_test, type='prob')

dim(predicted_test)
```

```
## [1] 500   2
```

Evaluation of Logit Model Alcohol Non-Penalized

```
require(cutpointr)
```

```
## Loading required package: cutpointr
```

```
##
## Attaching package: 'cutpointr'
```

```
## The following objects are masked from 'package:caret':
##
##     precision, recall, sensitivity, specificity
```

```
cut.obj <- cutpointr(x     = predicted_test$used,
                     class = exams_test$alcohol_use)
```

```
## Assuming the positive class is used
## Assuming the positive class has higher x values
```

```
cat(auc(cut.obj), "AUC")
```

```
## 0.638129 AUC
```

```
cat("\n")
```

```
pred_class <- ifelse(predicted_test$used>.25,1,0)

confusion <- table(exams_test$alcohol_use,pred_class)

confusion
```

```
##             pred_class
##                0   1
##   never used 396  19
##   used        80   5
```

```
cat("Evaluation Metrics - Non-Penalized Logistic Regression Model")
```

```
## Evaluation Metrics - Non-Penalized Logistic Regression Model
```

```
cat("\n")
```

```
# True Negative Rate
TNR <- confusion[1,1]/(confusion[1,1]+confusion[1,2])
cat(TNR, "True Negative Rate")
```

```
## 0.9542169 True Negative Rate
```

```
cat("\n")
```

```
# False Positive Rate
FPR <- confusion[1,2]/(confusion[1,1]+confusion[1,2])
cat(FPR, "False Positive Rate")
```

```
## 0.04578313 False Positive Rate
```

```
cat("\n")
```

```
# True Positive Rate
TPR <- confusion[2,2]/(confusion[2,1]+confusion[2,2])
cat(TPR, "True Positive Rate")
```

```
## 0.05882353 True Positive Rate
```

```
cat("\n")
```

```
# Precision
PRE <- confusion[2,2]/(confusion[1,2]+confusion[2,2])
cat(PRE, "Precision")
```

```
## 0.2083333 Precision
```

```
cat("\n")
```

```
# Accuracy
ACC <- (confusion[1,1] + confusion[2,2])/(confusion[1,1]+confusion[1,2]+confusion[2,1]+confusion[2,2])
cat(ACC, "Accuracy")
```

```
## 0.802 Accuracy
```

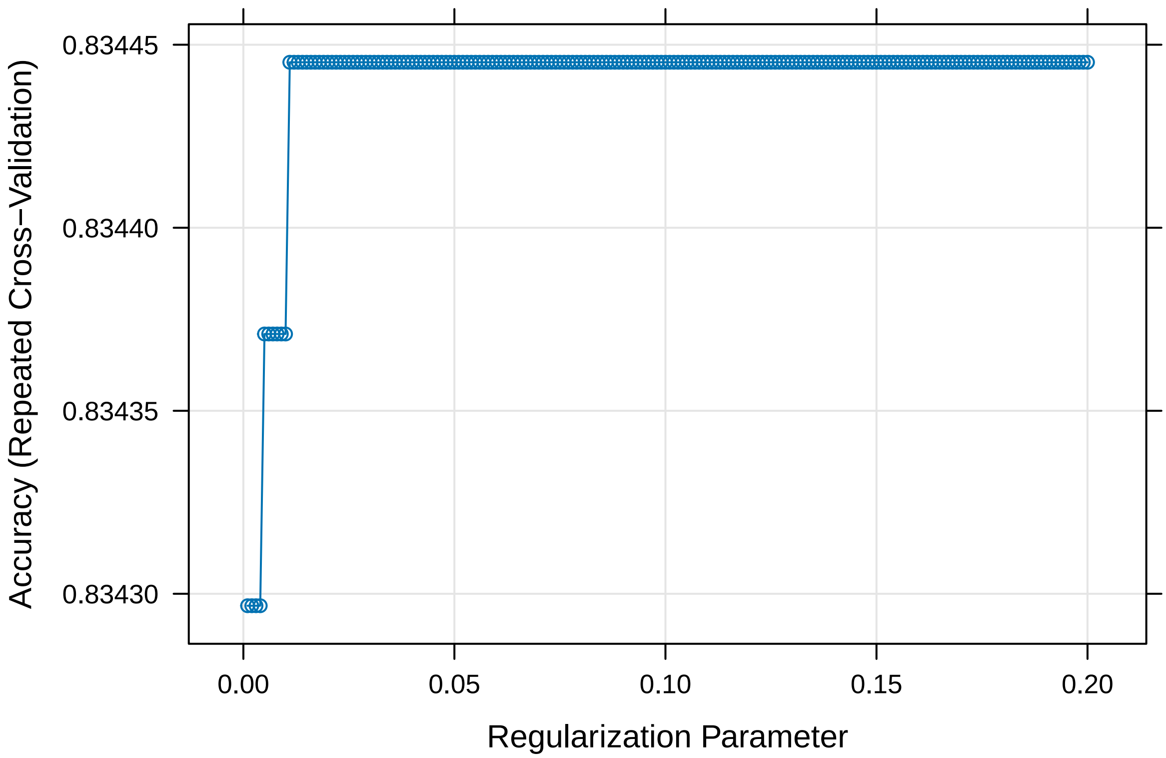
```
cat("\n")
```

Logit Ridge Model

```
grid_ridge <- data.frame(alpha = 0, lambda = seq(0.001,.2,.001))

ridge_mod <- caret::train(blueprint_exams,
                          data      = exams_train,
                          method    = "glmnet",
                          tuneGrid  = grid_ridge,
                          trControl = cv)
```

```
## Loading required namespace: glmnet

## Loading required package: Matrix

##
## Attaching package: 'Matrix'

## The following object is masked from 'package:bitops':
##
##     %&%

## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack

## Loaded glmnet 4.1-8
```

```
plot(ridge_mod)
```



Ridge Regression Evaluation

```
best_lambda <- ridge_mod$bestTune$lambda
best_lambda
```

```
## [1] 0.2
```

```
ridge_mod$results[200,]
```

```
##     alpha lambda  Accuracy Kappa   AccuracySD KappaSD
## 200     0    0.2 0.8344452     0 0.0009735508       0
```

```
predicted_test <- predict(ridge_mod, exams_test, type='prob')

summary(predicted_test$used)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.09443 0.13905 0.16941 0.16486 0.18764 0.28174
```

```
cut.obj_ridge <- cutpointr(x    = predicted_test$used,
                           class = exams_test$alcohol_use)
```

```
## Assuming the positive class is used
```

```
## Assuming the positive class has higher x values
```

```
cat(auc(cut.obj_ridge), "AUC")
```

```
## 0.63073 AUC
```

```
cat("\n")
```

```
pred_class <- ifelse(predicted_test$used>.25,1,0)
```

```r
confusion <- table(exams_test$alcohol_use,pred_class)

confusion
```

```
##             pred_class
##                0   1
##   never used 412   3
##   used        84   1
```

```r
cat("Evaluation Metrics - Ridge Penalty Model")
```

```
## Evaluation Metrics - Ridge Penalty Model
```

```r
cat("\n")
```

```r
# True Negative Rate
TNR <- confusion[1,1]/(confusion[1,1]+confusion[1,2])
cat(TNR, "True Negative Rate")
```

```
## 0.9927711 True Negative Rate
```

```r
cat("\n")
```

```r
# False Positive Rate
FPR <- confusion[1,2]/(confusion[1,1]+confusion[1,2])
cat(FPR, "False Positive Rate")
```

```
## 0.007228916 False Positive Rate
```

```r
cat("\n")
```

```r
# True Positive Rate
TPR <- confusion[2,2]/(confusion[2,1]+confusion[2,2])
cat(TPR, "True Positive Rate")
```

```
## 0.01176471 True Positive Rate
```

```r
cat("\n")
```

```r
# Precision
PRE <- confusion[2,2]/(confusion[1,2]+confusion[2,2])
cat(PRE, "Precision")
```

```
## 0.25 Precision
```

```r
cat("\n")
```

```r
# Accuracy
ACC <- (confusion[1,1] + confusion[2,2])/(confusion[1,1]+confusion[1,2]+confusion[2,1]+confusion[2,2])
cat(ACC, "Accuracy")
```

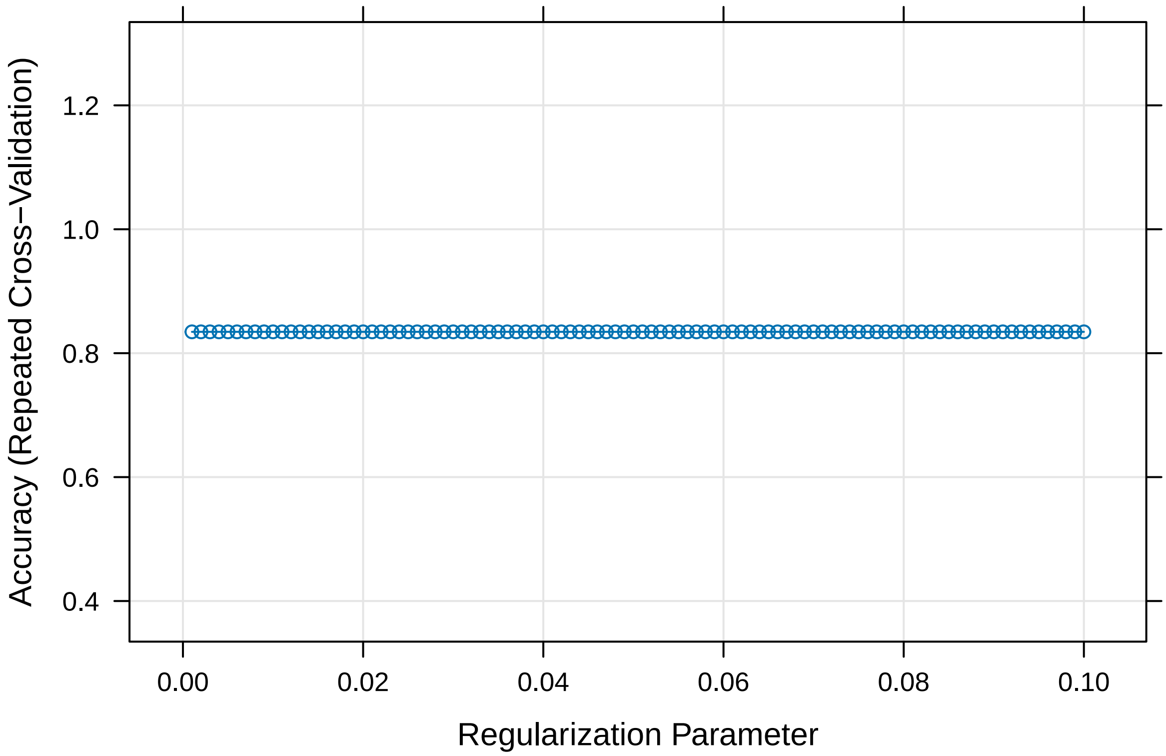
```
## 0.826 Accuracy
```

```r
cat("\n")
```

Logit Lasso Model

```r
grid_lasso <- data.frame(alpha = 1, lambda = seq(0.001,0.1,.001))

lasso_mod <- caret::train(blueprint_exams,
                          data      = exams_train,
```

```r
                                  method     = "glmnet",
                                  tuneGrid   = grid_lasso,
                                  trControl  = cv)
plot(lasso_mod)
```




```r
lasso_mod$bestTune$lambda
```

```
## [1] 0.1
```

```r
lasso_mod$results[100,]
```

```
##     alpha lambda  Accuracy Kappa   AccuracySD KappaSD
## 100     1    0.1 0.8344451     0 0.0009974215       0
```

```r
predicted_test <- predict(lasso_mod, exams_test, type='prob')
summary(predicted_test$used)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.1656  0.1656  0.1656  0.1656  0.1656  0.1656
```

```r
cut.obj_lasso <- cutpointr(x  = predicted_test$used,
                           class = exams_test$alcohol_use)
```

```
## Assuming the positive class is used
## Assuming the positive class has higher x values
## Multiple optimal cutpoints found, applying break_ties.
```

```r
cat(auc(cut.obj_lasso), "AUC")
```

```
## 0.5 AUC
```

```r
cat("\n")
```

```r
pred_class <- ifelse(predicted_test$used>.25,1,0)

summary(predicted_test$used)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.1656  0.1656  0.1656  0.1656  0.1656  0.1656
```

```r
#
# cat("Evaluation Metrics - Lasso Penalty Model")
# cat("\n")
#
# # True Negative Rate
# TNR <- confusion[1,1]/(confusion[1,1]+confusion[1,2])
# cat(TNR, "True Negative Rate")
# cat("\n")
#
# # False Positive Rate
# FPR <- confusion[1,2]/(confusion[1,1]+confusion[1,2])
# cat(FPR, "False Positive Rate")
# cat("\n")
#
# # True Positive Rate
# TPR <- confusion[2,2]/(confusion[2,1]+confusion[2,2])
# cat(TPR, "True Positive Rate")
# cat("\n")
#
# # Precision
# PRE <- confusion[2,2]/(confusion[1,2]+confusion[2,2])
# cat(PRE, "Precision")
# cat("\n")
#
# # Accuracy
# ACC <- (confusion[1,1] + confusion[2,2])/(confusion[1,1]+confusion[1,2]+confusion[2,1]+confusion[2,2]
# cat(ACC, "Accuracy")
# cat("\n")
```

For some reason the Lasso and Ridge Regression Models seem to be performing worse than the non-penalized model.

Alcohol Variable Coefficients

Coefficients for the ridge model.

```r
options(scipen=99)
coefs <- coef(ridge_mod$finalModel,ridge_mod$bestTune$lambda)
coefs.zero <- coefs[which(coefs[,1]==0),]
cat(length(coefs.zero), "predictors had a value of zero for the coefficient with Ridge penalty regressi
```

```
## 0 predictors had a value of zero for the coefficient with Ridge penalty regression
```

```r
ind   <- order(coefs,decreasing=T)
```

```
## <sparse>[ <logic> ]: .M.sub.i.logical() maybe inefficient
```

```
DT::datatable(as.matrix(coefs[ind[-1],]))
```

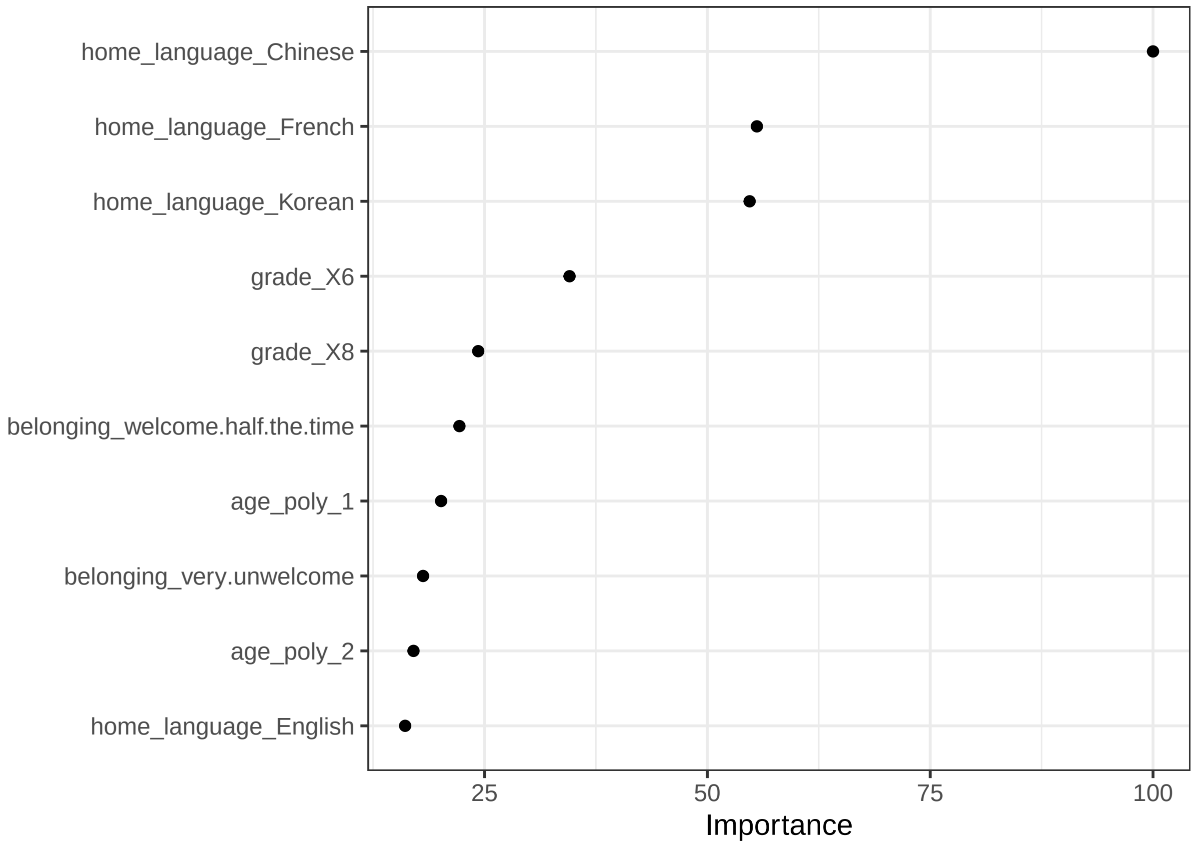
Coefficients for the non-penalized model.

```
coefficients <- coef(mod_log$finalModel)
coef_names <- names(coefficients)
coef_df <- data.frame(variable = coef_names, coefficient = coefficients)
sorted_coefs <- coef_df[order(coef_df$coefficient, decreasing = TRUE), ]
DT::datatable(sorted_coefs, rownames = FALSE)
```

Examining Variables of importance

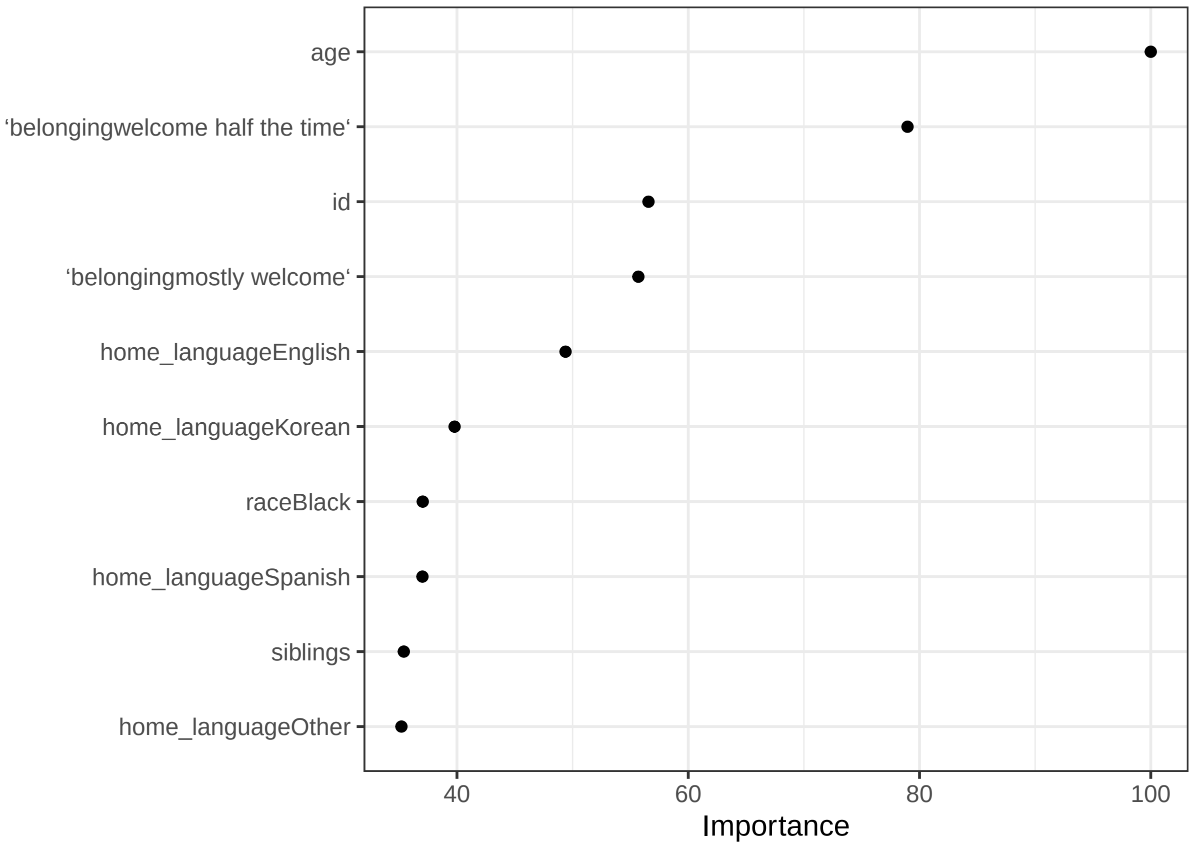
Importance of variables based on ridge model.

```
vip(ridge_mod,
    num_features = 10,
    geom = "point") +
  theme_bw()
```



Importance of variables based on non-penalized model.

```
vip(mod_log,
    num_features = 10,
    geom = "point") +
  theme_bw()
```



Tree Modeling

The tree model does not predict the alcohol use well. It predicts no students use alcohol. I tried using a bagged tree model but was running into errors after it initially ran.

```
# exams <- exams %>%
#   mutate(alcohol_use = recode_factor(alcohol_use,
#       'never used' = '0',
#       'used' = '1'))
#
# exams <- exams %>%
#   mutate(marijuana_use = recode_factor(marijuana_use,
#       'never used' = '0',
#       'used' = '1'))
#
# all_exam_pred <- c(categorical, numeric)
#
#
# blueprint_exams <- recipe(x = exams) %>%
#   update_role(id, new_role = "id") %>%
#   update_role(outcome, new_role = "outcome") %>%
#   update_role(all_exam_pred, new_role = "predictor") %>%
#   step_indicate_na(all_of(categorical),all_of(numeric)) %>%
#   step_zv(all_numeric()) %>%
#   step_impute_mean(all_of(numeric)) %>%
#   step_impute_mode(all_of(categorical)) %>%
#   step_poly(all_of(numeric),degree=2) %>%
```

```
#   step_normalize(paste0(numeric,'_poly_1'),
#                  paste0(numeric,'_poly_2')) %>%
#   step_dummy(all_of(categorical),one_hot=TRUE)
#
# set.seed(121619)
#
# loc      <- sample(1:nrow(exams), round(nrow(exams) * 0.9))
# exams_train  <- exams[loc, ]
# exams_test  <- exams[-loc, ]
#
# exams_train = exams_train[sample(nrow(exams_train)),]
#
#   # Create 10 folds with equal size
#
#     folds = cut(seq(1,nrow(exams_train)),breaks=10,labels=FALSE)
#
#   # Create the list for each fold
#
#     my.indices <- vector('list',10)
#     for(i in 1:10){
#       my.indices[[i]] <- which(folds!=i)
#     }
#
# prepare <- prep(blueprint_exams,
#                 training = exams_train)
# prepare
#
# cv <- trainControl(method = "cv",
#                      index  = my.indices,
#                      classProbs = TRUE,
#                      summaryFunction = mnLogLoss)
```

```
#install.packages("ranger")
#require(ranger)

# grid <- expand.grid(mtry = 24,splitrule='gini',min.node.size=2)


grid <- data.frame(cp=seq(0,5,.01))

cv <- trainControl(method = "cv", number = 10)

# Train the model
tree_mod <- caret::train(blueprint_exams,
             data = exams_train,
             method = "rpart",
             tuneGrid = grid,
             trControl = cv,
             control = list(minsplit = 1,
                            minbucket = 2,
                            maxdepth = 10))


# Get predictions
```

```r
pred <- predict(tree_mod, newdata = exams_test)
table(pred)
```

```
## pred
## never used       used
##        500          0
```

```r
# Evaluate the model
results <- confusionMatrix(pred, exams_test$alcohol_use)
print(results)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   never used used
##   never used        415   85
##   used                0    0
##
##                Accuracy : 0.83
##                  95% CI : (0.7941, 0.8619)
##     No Information Rate : 0.83
##     P-Value [Acc > NIR] : 0.5289
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : <0.0000000000000002
##
##             Sensitivity : 1.00
##             Specificity : 0.00
##          Pos Pred Value : 0.83
##          Neg Pred Value :  NaN
##              Prevalence : 0.83
##          Detection Rate : 0.83
##    Detection Prevalence : 1.00
##       Balanced Accuracy : 0.50
##
##        'Positive' Class : never used
##
```

```r
#print(data.frame(Predicted = predictions, Actual = exams_test$alcohol_use))
```

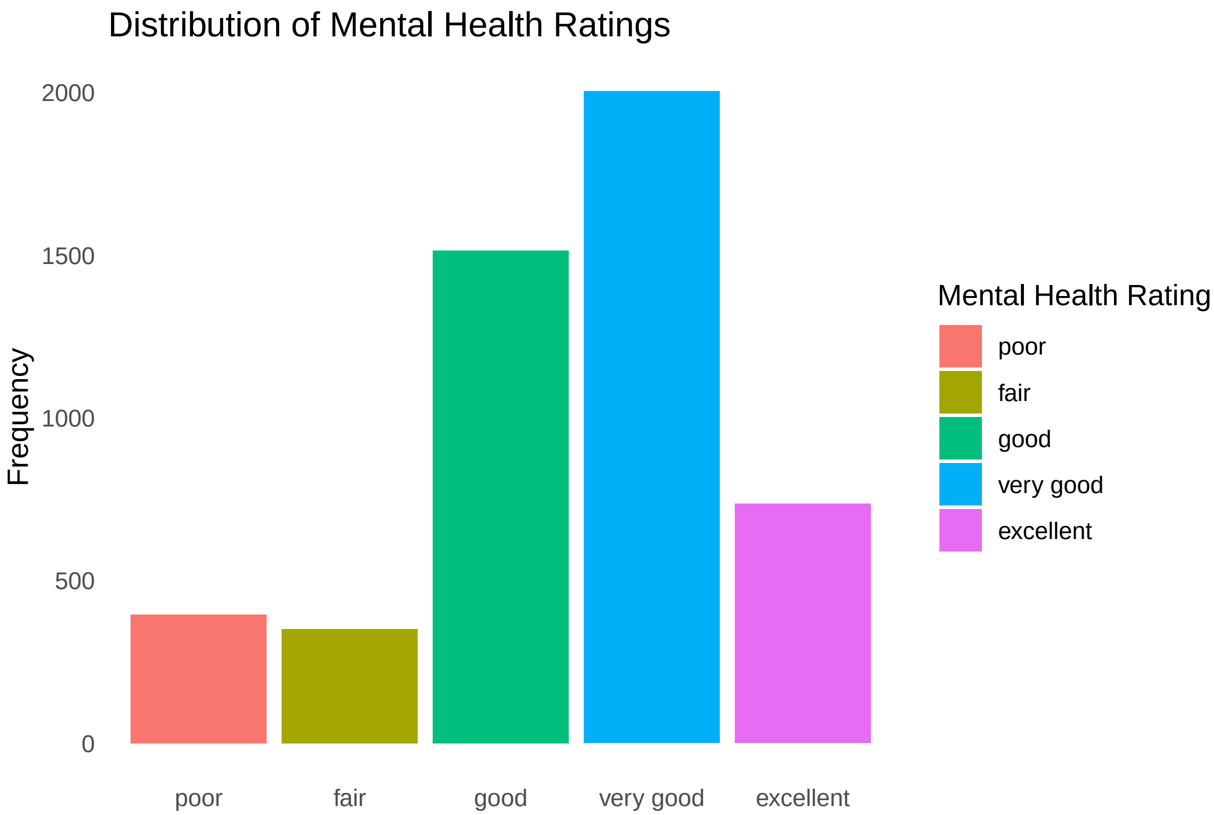
```r
# vip(tree_mod,
#     num_features = 10,
#     geom = "point") +
#   theme_bw()
```

Graphs

Mental Health

```r
mental_tab <- as.data.frame(table(exams$mental_health))

mental_tab %>%
ggplot(aes(x = Var1, y = Freq, fill = Var1)) +
  geom_bar(stat = "identity", position = "dodge") +
```

```r
  labs(title = "Distribution of Mental Health Ratings",
       x = "",
       y = "Frequency") +
  scale_fill_discrete(name = "Mental Health Rating") +
  theme_minimal() +
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.background = element_blank())
```



Mental Health x Alcohol

```r
############################################################
# Mental Health Data
############################################################

alcohol_mental_table <- as.data.frame(table(exams$alcohol_use, exams$mental_health))
alcohol_mental_table
```

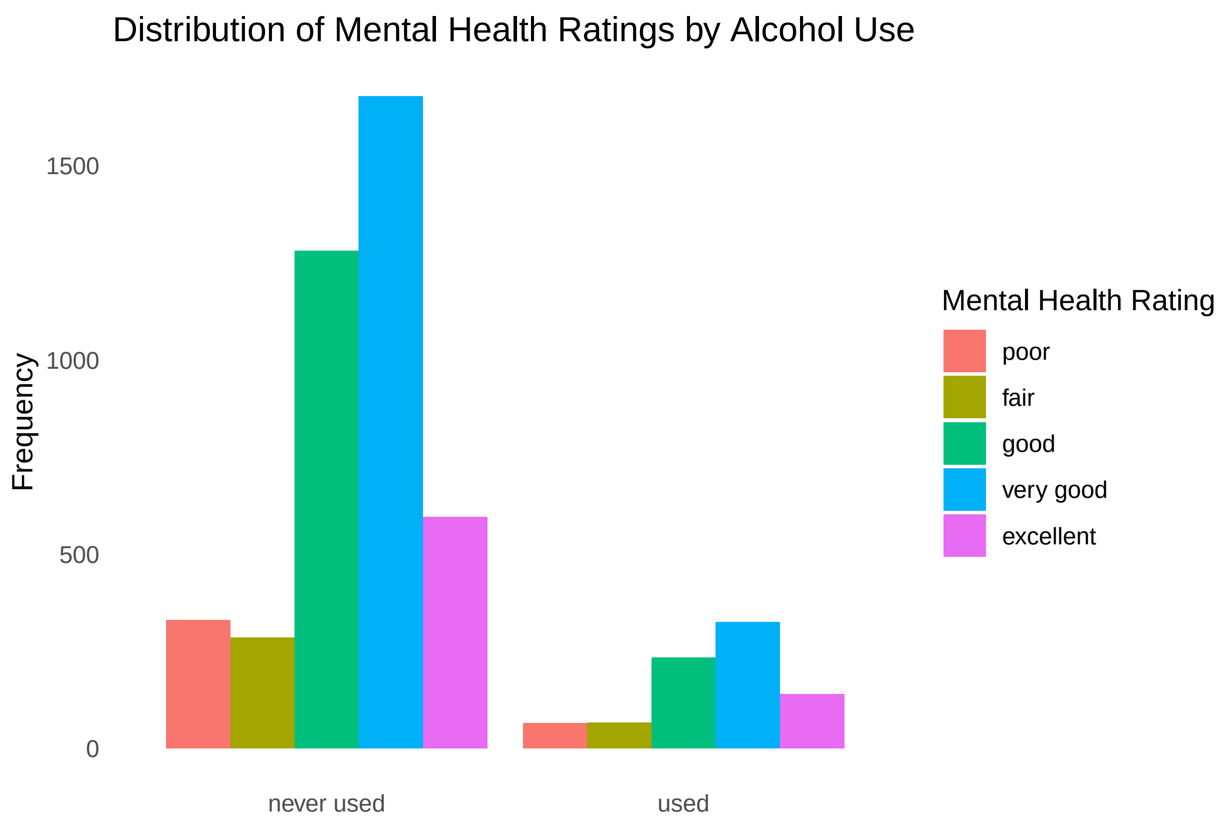
```
##          Var1      Var2 Freq
## 1  never used      poor  330
## 2        used      poor   65
## 3  never used      fair  285
## 4        used      fair   66
## 5  never used      good 1280
## 6        used      good  234
## 7  never used very good 1679
## 8        used very good  325
## 9  never used excellent  596
```

```
## 10       used excellent  140
```

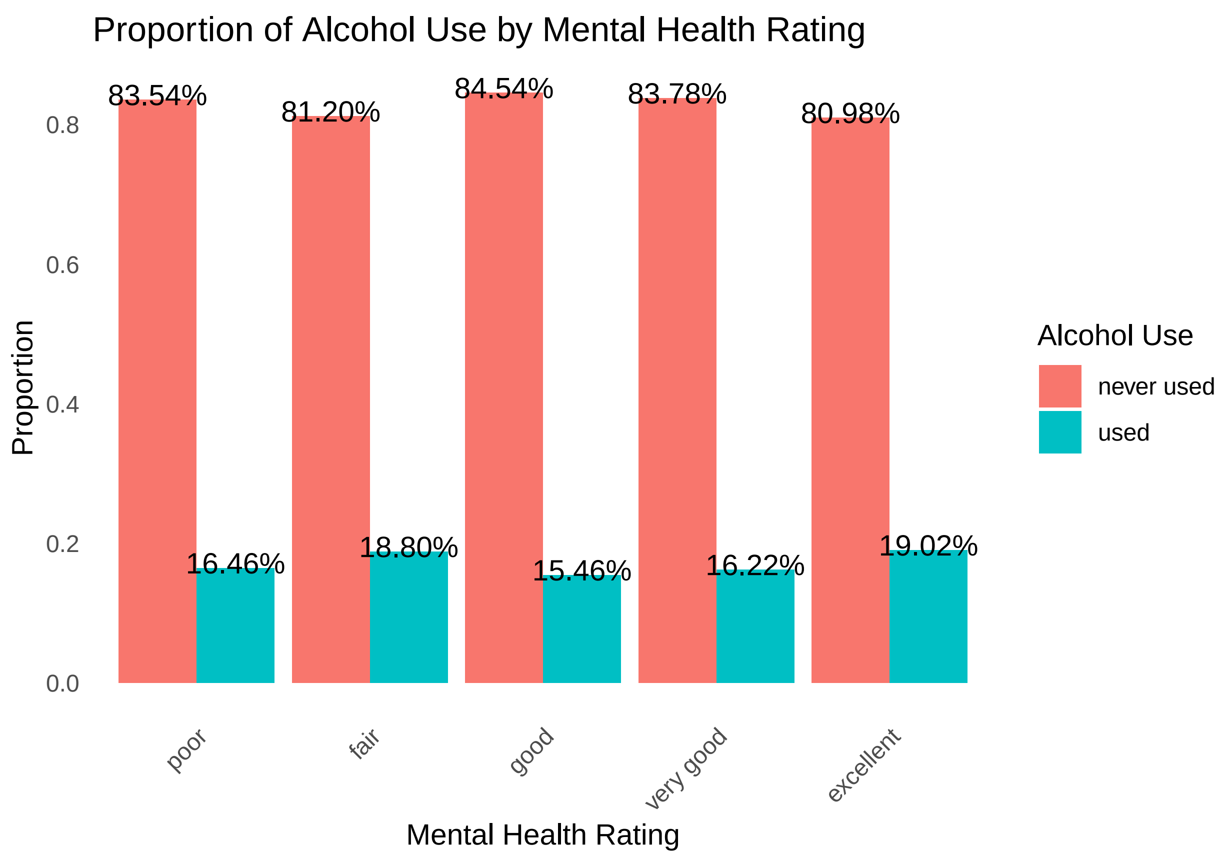
```r
proportions <- prop.table(table(exams$alcohol_use, exams$mental_health), margin = 2)
alcohol_mental_table_proportions <- as.data.frame(proportions)
colnames(alcohol_mental_table_proportions) <- c("Alcohol_Use", "Mental Health", "Proportion")
alcohol_mental_table_proportions
```

```
##    Alcohol_Use Mental Health Proportion
## 1   never used          poor  0.8354430
## 2         used          poor  0.1645570
## 3   never used          fair  0.8119658
## 4         used          fair  0.1880342
## 5   never used          good  0.8454425
## 6         used          good  0.1545575
## 7   never used     very good  0.8378244
## 8         used     very good  0.1621756
## 9   never used     excellent  0.8097826
## 10        used     excellent  0.1902174
```

```r
alcohol_mental_table %>%
ggplot(aes(x = Var1, y = Freq, fill = Var2)) +
  geom_col(position = "dodge") +
  labs(title = "Distribution of Mental Health Ratings by Alcohol Use",
       x = "",
       y = "Frequency") +
  scale_fill_discrete(name = "Mental Health Rating") +
  theme_minimal() +
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.background = element_blank())
```




```
alcohol_mental_table_proportions %>%
ggplot(aes(x = `Mental Health`, y = Proportion, fill = Alcohol_Use)) +
  geom_col(position = "dodge") +
  labs(title = "Proportion of Alcohol Use by Mental Health Rating",
       x = "Mental Health Rating",
       y = "Proportion") +
  scale_fill_discrete(name = "Alcohol Use") +
  theme_minimal() +
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.background = element_blank(),
        axis.text.x = element_text(angle = 45, hjust = 1)) +
  geom_text(aes(label = scales::percent(Proportion)),
            position = position_dodge(width = 0.9),
            vjust = 0.25)
```



Marijuana Graph x Alcohol

```
########################################################################
# Marijuana Data
#######################################################################

alcohol_mari_table <- as.data.frame(table(exams$alcohol_use, exams$marijuana_use))
alcohol_mari_table
```

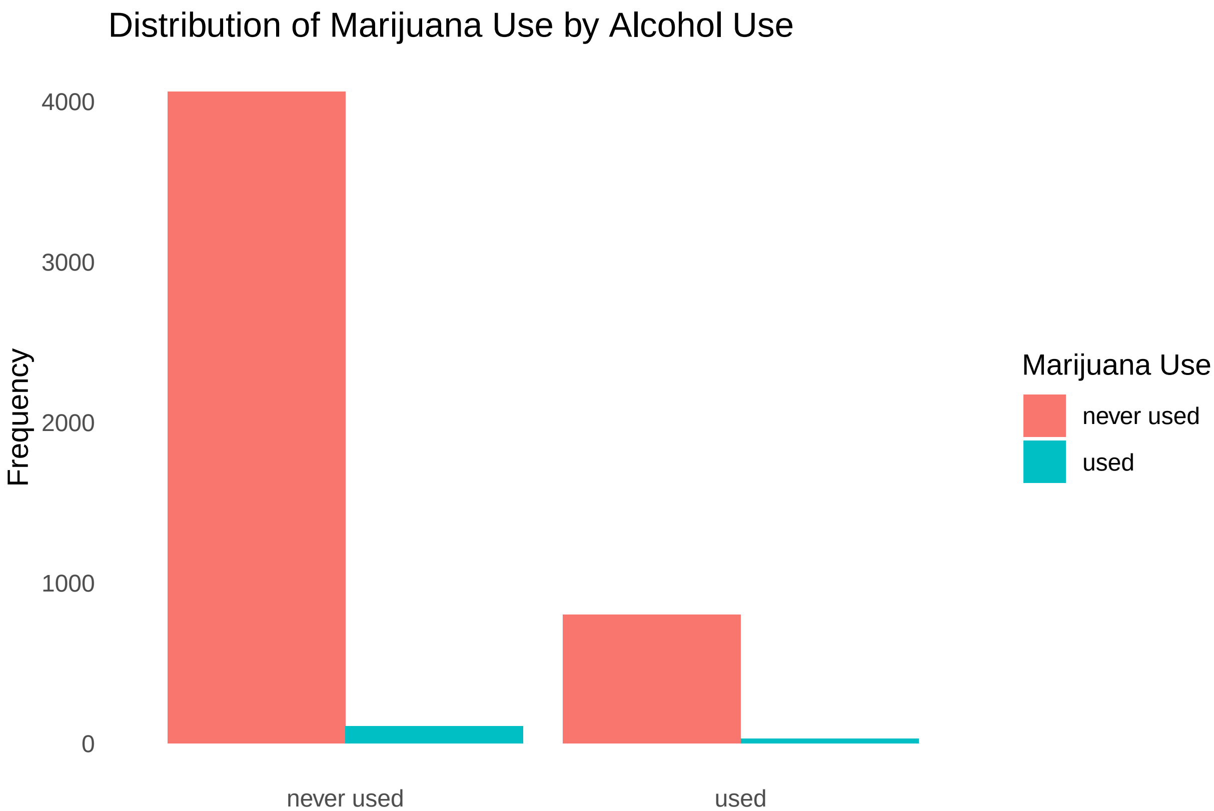
```
##          Var1       Var2 Freq
## 1 never used never used 4061
## 2       used never used  801
## 3 never used       used  109
## 4       used       used   29
```

```
proportions <- prop.table(table(exams$alcohol_use, exams$marijuana_use), margin = 2)
alcohol_mari_table_proportions <- as.data.frame(proportions)
colnames(alcohol_mari_table_proportions) <- c("Alcohol_Use", "Marijuana_Use", "Proportion")
alcohol_mari_table_proportions
```

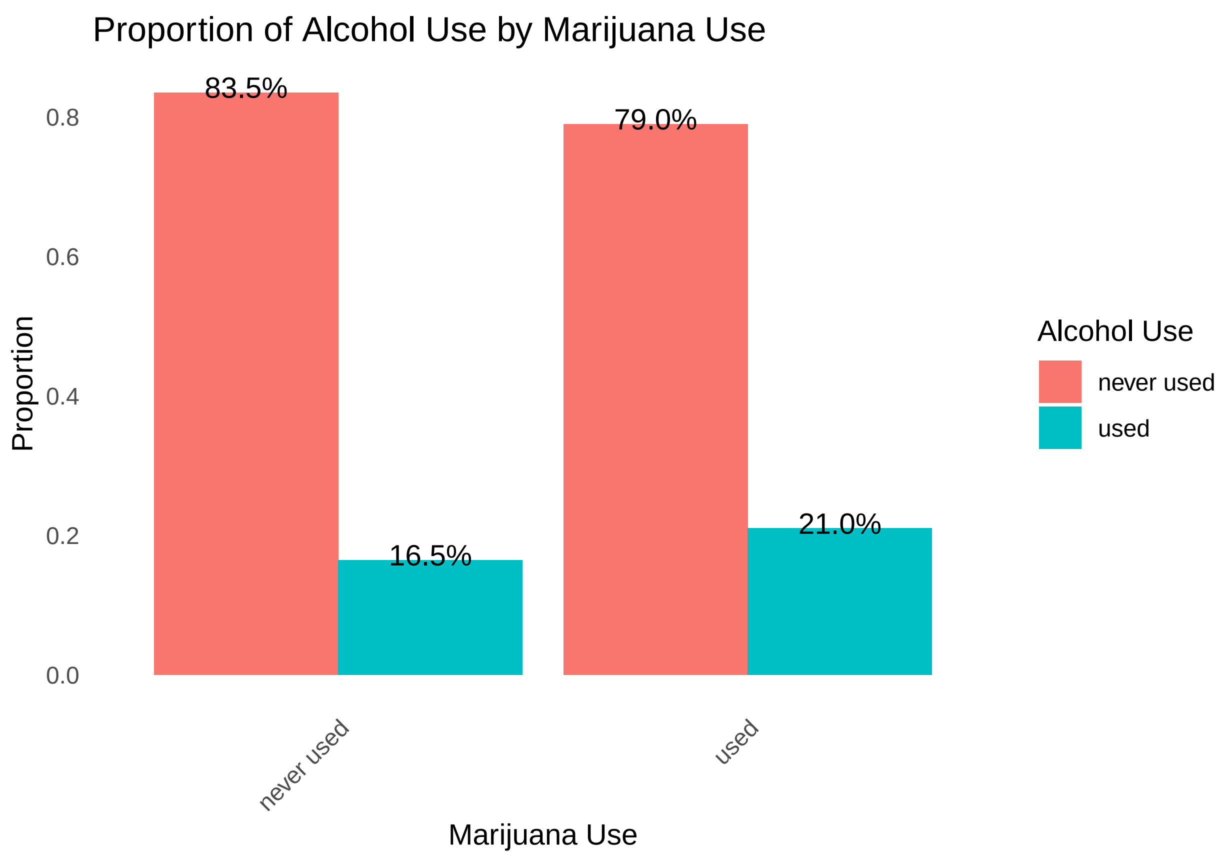
```
##   Alcohol_Use Marijuana_Use Proportion
## 1  never used    never used  0.8352530
## 2        used    never used  0.1647470
## 3  never used          used  0.7898551
## 4        used          used  0.2101449
```

```
alcohol_mari_table %>%
ggplot(aes(x = Var1, y = Freq, fill = Var2)) +
```

```r
  geom_col(position = "dodge") +
  labs(title = "Distribution of Marijuana Use by Alcohol Use",
       x = "",
       y = "Frequency") +
  scale_fill_discrete(name = "Marijuana Use") +
  theme_minimal() +
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.background = element_blank())
```




```r
alcohol_mari_table_proportions %>%
ggplot(aes(x = Marijuana_Use, y = Proportion, fill = Alcohol_Use)) +
  geom_col(position = "dodge") +
  labs(title = "Proportion of Alcohol Use by Marijuana Use",
       x = "Marijuana Use",
       y = "Proportion") +
  scale_fill_discrete(name = "Alcohol Use") +
  theme_minimal() +
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.background = element_blank(),
        axis.text.x = element_text(angle = 45, hjust = 1)) +
  geom_text(aes(label = scales::percent(Proportion)),
            position = position_dodge(width = 0.9),
            vjust = 0.25)
```



Home language x Alcohol

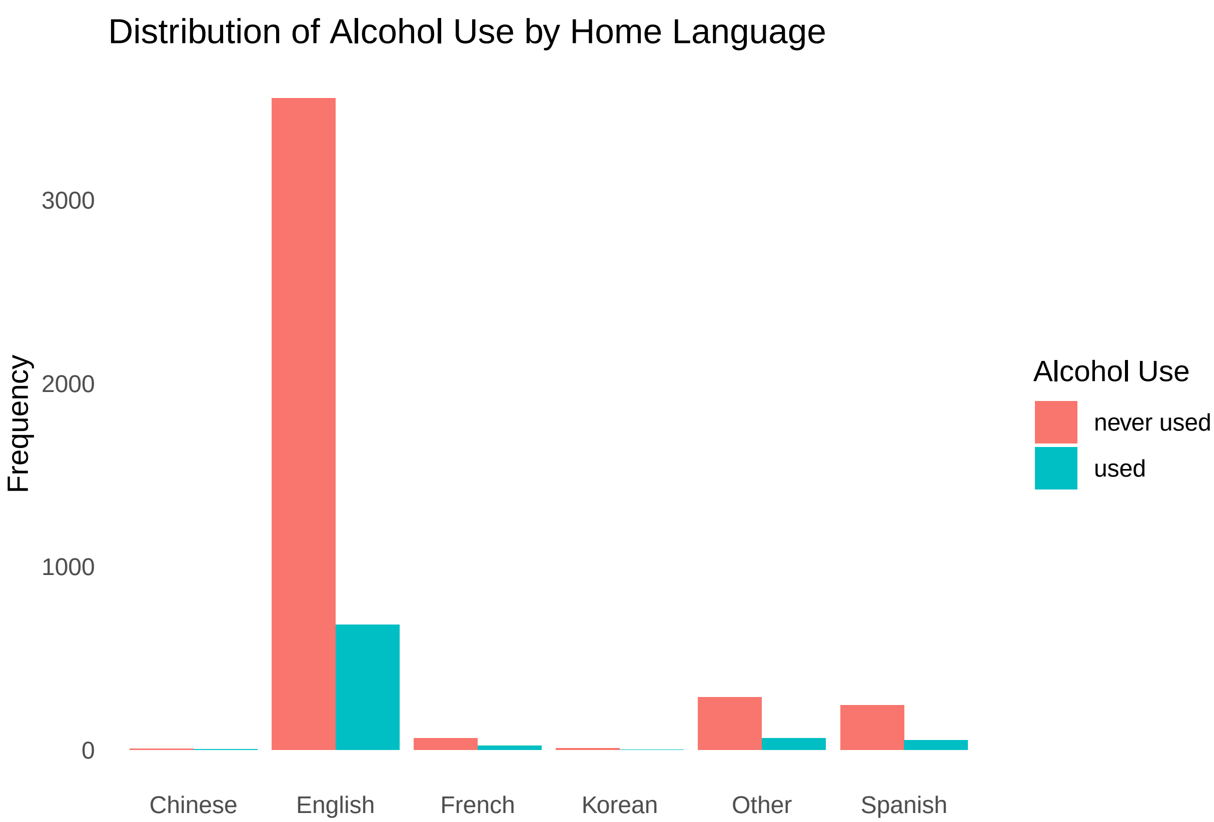
```
########################################################################
# Home Language Data
########################################################################
alcohol_lang_table <- as.data.frame(table(exams$alcohol_use, exams$home_language))
alcohol_lang_table
```

```
##            Var1    Var2 Freq
## 1  never used Chinese    8
## 2        used Chinese    4
## 3  never used English 3556
## 4        used English  684
## 5  never used  French   64
## 6        used  French   23
## 7  never used  Korean   10
## 8        used  Korean    2
## 9  never used   Other  288
## 10       used   Other   65
## 11 never used Spanish  244
## 12       used Spanish   52
```

```
proportions <- prop.table(table(exams$alcohol_use, exams$home_language), margin = 2)
alcohol_lang_table_proportions <- as.data.frame(proportions)
colnames(alcohol_lang_table_proportions) <- c("Alcohol_Use", "Home_Language", "Proportion")
alcohol_lang_table_proportions
```

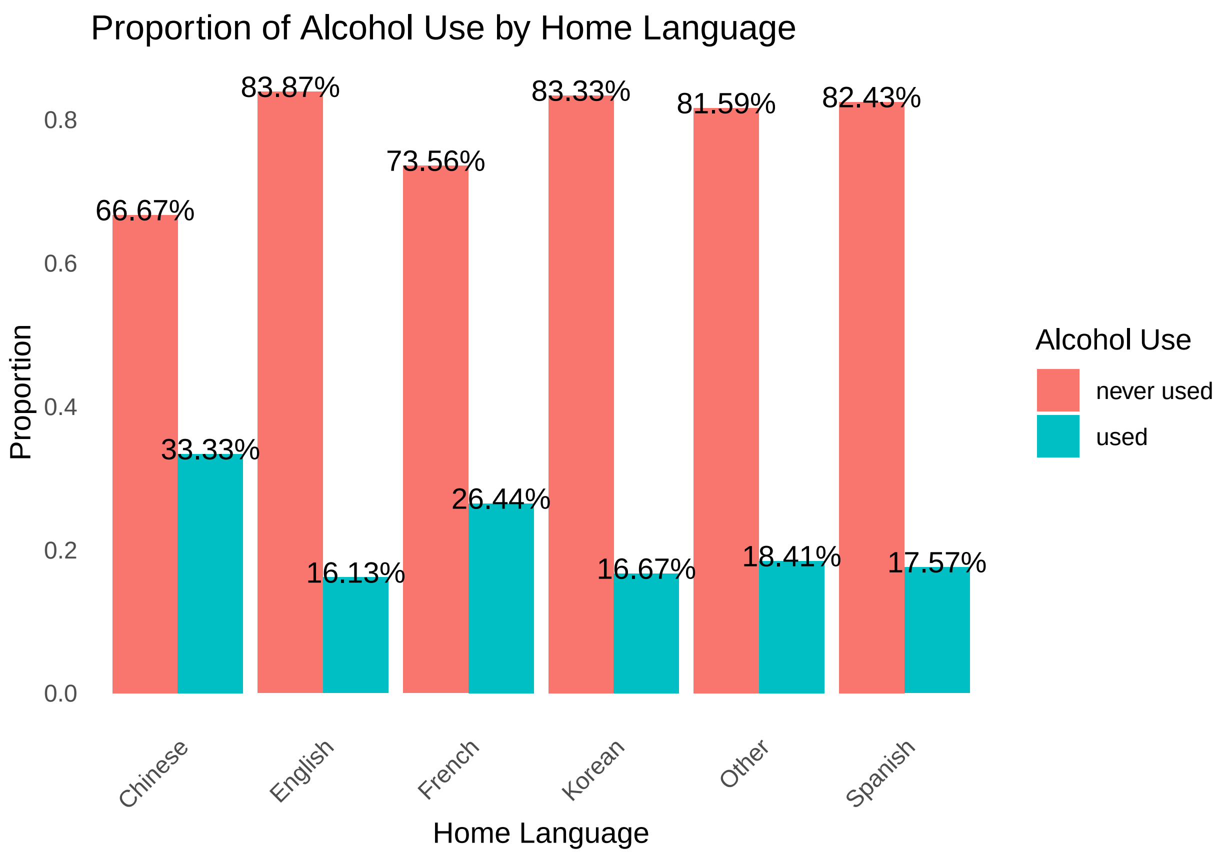
```
##    Alcohol_Use Home_Language Proportion
## 1   never used       Chinese  0.6666667
## 2         used       Chinese  0.3333333
## 3   never used       English  0.8386792
## 4         used       English  0.1613208
## 5   never used        French  0.7356322
## 6         used        French  0.2643678
## 7   never used        Korean  0.8333333
## 8         used        Korean  0.1666667
## 9   never used         Other  0.8158640
## 10        used         Other  0.1841360
## 11  never used       Spanish  0.8243243
## 12        used       Spanish  0.1756757
```

```r
alcohol_lang_table %>%
ggplot(aes(x = Var2, y = Freq, fill = Var1)) +
  geom_col(position = "dodge") +
  labs(title = "Distribution of Alcohol Use by Home Language",
       x = "",
       y = "Frequency") +
  scale_fill_discrete(name = "Alcohol Use") +
  theme_minimal() +
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.background = element_blank())
```




```r
alcohol_lang_table_proportions %>%
ggplot(aes(x = Home_Language, y = Proportion, fill = Alcohol_Use)) +
  geom_col(position = "dodge") +
```

```r
  labs(title = "Proportion of Alcohol Use by Home Language",
       x = "Home Language",
       y = "Proportion") +
  scale_fill_discrete(name = "Alcohol Use") +
  theme_minimal() +
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.background = element_blank(),
        axis.text.x = element_text(angle = 45, hjust = 1)) +
  geom_text(aes(label = scales::percent(Proportion)),
            position = position_dodge(width = 0.9),
            vjust = 0.25)
```



Belonging x Alcohol

```r
########################################################################
# Belonging Data
########################################################################
alcohol_belong_table <- as.data.frame(table(exams$alcohol_use, exams$belonging))
alcohol_belong_table
```

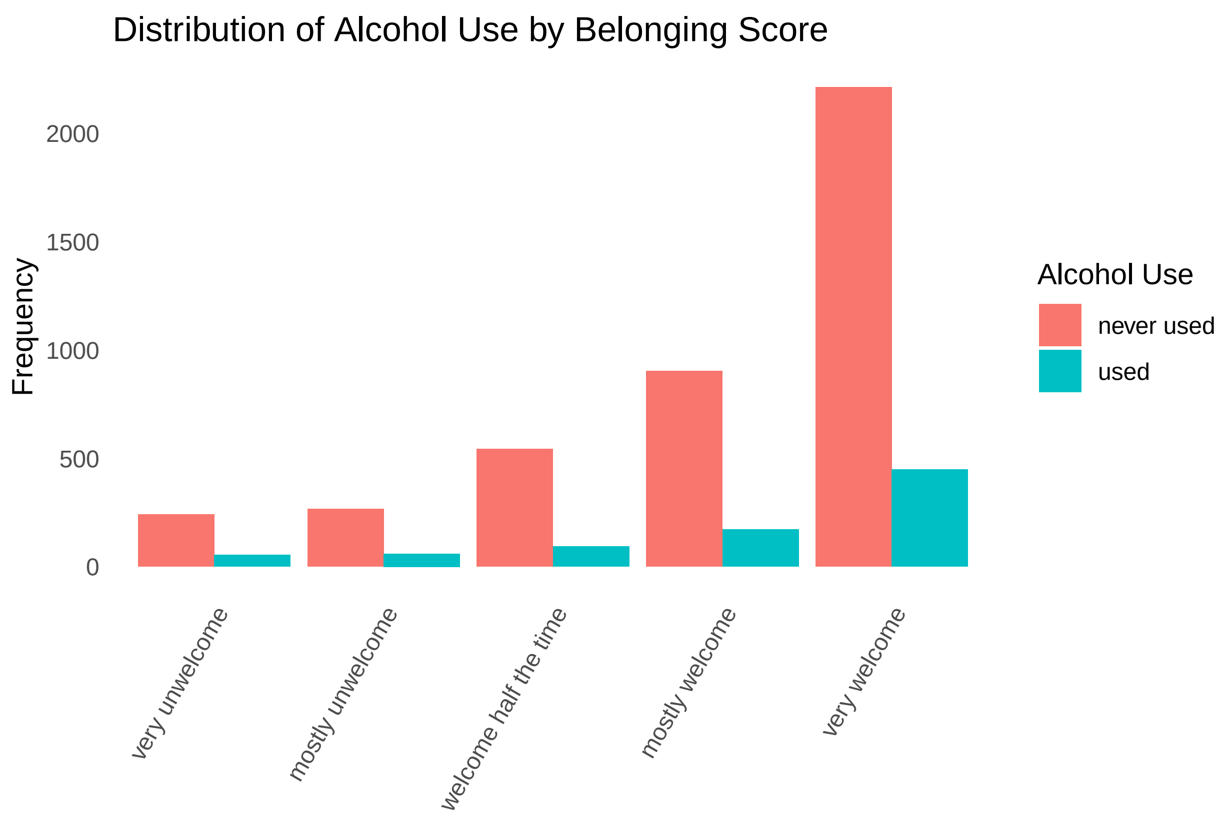
```
##          Var1                 Var2 Freq
## 1  never used       very unwelcome  242
## 2        used       very unwelcome   55
## 3  never used     mostly unwelcome  267
## 4        used     mostly unwelcome   60
```

```
## 5  never used welcome half the time  544
## 6        used welcome half the time   94
## 7  never used        mostly welcome  904
## 8        used        mostly welcome  172
## 9  never used          very welcome 2213
## 10       used          very welcome  449
```

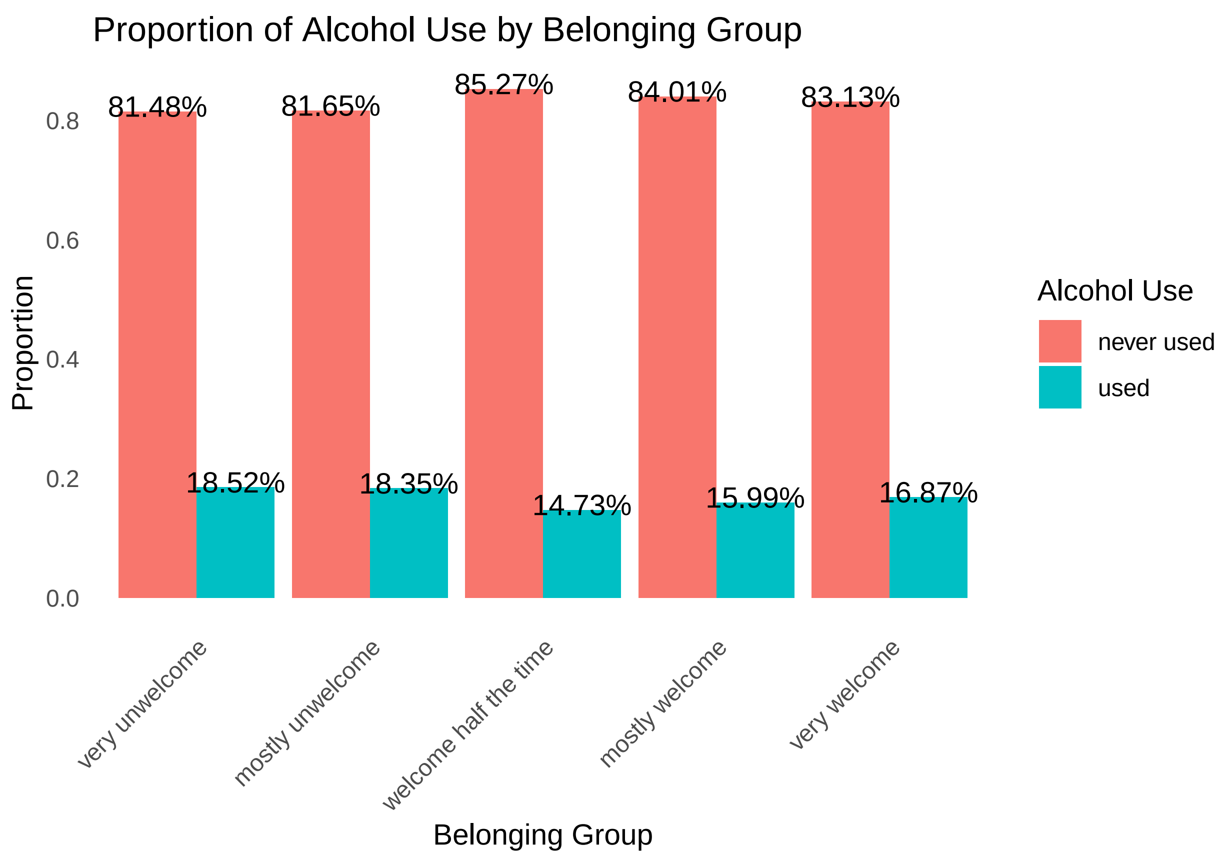
```r
proportions <- prop.table(table(exams$alcohol_use, exams$belonging), margin = 2)
alcohol_belong_table_proportions <- as.data.frame(proportions)
colnames(alcohol_belong_table_proportions) <- c("Alcohol_Use", "Belonging", "Proportion")
alcohol_belong_table_proportions
```

```
##    Alcohol_Use             Belonging Proportion
## 1   never used        very unwelcome  0.8148148
## 2         used        very unwelcome  0.1851852
## 3   never used      mostly unwelcome  0.8165138
## 4         used      mostly unwelcome  0.1834862
## 5   never used welcome half the time  0.8526646
## 6         used welcome half the time  0.1473354
## 7   never used        mostly welcome  0.8401487
## 8         used        mostly welcome  0.1598513
## 9   never used          very welcome  0.8313298
## 10        used          very welcome  0.1686702
```

```r
alcohol_belong_table %>%
ggplot(aes(x = Var2, y = Freq, fill = Var1)) +
  geom_col(position = "dodge") +
  labs(title = "Distribution of Alcohol Use by Belonging Score",
       x = "",
       y = "Frequency") +
  scale_fill_discrete(name = "Alcohol Use") +
  theme_minimal() +
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.background = element_blank(),
        axis.text.x = element_text(angle = 60, hjust = 1))
```




```
alcohol_belong_table_proportions %>%
ggplot(aes(x = Belonging, y = Proportion, fill = Alcohol_Use)) +
  geom_col(position = "dodge") +
  labs(title = "Proportion of Alcohol Use by Belonging Group",
       x = "Belonging Group",
       y = "Proportion") +
  scale_fill_discrete(name = "Alcohol Use") +
  theme_minimal() +
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.background = element_blank(),
        axis.text.x = element_text(angle = 45, hjust = 1)) +
  geom_text(aes(label = scales::percent(Proportion)),
            position = position_dodge(width = 0.9),
            vjust = 0.25)
```


Proportion of Alcohol Use by Belonging Group
81.48%
81.65%
85.27%
84.01%
83.13%
18.52%
18.35%
14.73%
15.99%
16.87%
0.8
0.6
0.4
0.2
0.0
Proportion
Alcohol Use
never used
used
very unwelcome
mostly unwelcome
welcome half the time
mostly welcome
very welcome
Belonging Group