PSYCH308D - Data Analysis (DA03)

Brady C. Jackson

2025/04/19

Table of Contents

# 1 Libraries

Load all requisite libraries here.

# Load packages. Set messages and warnings to FALSE so I don't have to see the  
# masking messages in the output.  
library(jmv) # for descriptive  
library(ggplot2)  
library(dplyr)  
library(corrplot) # For fancy covariance matrix plots  
library(apaTables) # For Word formatted tables  
library(car) # for ncvTest (Breusch Pagan)  
library(tidyverse)  
library(jmv) # for descriptives  
library(ggplot2)  
library(dplyr)  
library(psych)  
library(corrplot) # For fancy covariance matrix plots  
library(car) # for ncvTest (Breusch Pagan)  
library(stringr) # for sub\_str operations  
library(Hmisc) # for fun.dat substitution  
library(see) # for outliers analysis   
library(magrittr)  
library(foreign)  
library(broom)  
library(robmed)  
library(mediation) # For mediation analysis  
library(multilevel)  
library(GGally)  
library(lsr)  
library(car)  
library(mvnTest) # Multivariate Normality  
library(lm.beta)  
library(lavaan) # Structural Equation Modeling  
library(haven)  
library(foreign)  
library(parallel)  
# library(AER)  
library(janitor) # Data cleaning  
library(naniar) # Data cleaning  
library(performance) # Data cleaning  
library(mice) # Data cleaning

# 2 Metadata

This section of code is to setup some general variables that we’ll use throughout the code (e.g. figure colors, etc)

# First we'll defines some meta-data to use in all of our plots so they're nice and clean  
font\_color = "#4F81BD"  
grid\_color\_major = "#9BB7D9"  
grid\_color\_minor = "#C8D7EA"  
back\_color = "gray95"  
rb\_colmap = colorRampPalette( c("firebrick", "grey86", "dodgerblue3") )(200)  
  
# I'm going to try to save off my preferred ggplot theme combinations as a unqiue theme object that I can just reference  
# later in the code....totally unclear if ggplot works this way....  
my\_gg\_theme = theme\_minimal() +  
 theme( plot.title = element\_text(size = 12, face = "italic", color = font\_color),  
 axis.title.x = element\_text(color = font\_color),  
 axis.title.y = element\_text(color = font\_color),  
 axis.text.x = element\_text(color = font\_color),  
 axis.text.y = element\_text(color = font\_color),  
 legend.title = element\_text(color = font\_color),  
 legend.text = element\_text(color = font\_color),  
 panel.grid.minor = element\_line(color = grid\_color\_minor),  
 panel.grid.major = element\_line(color = grid\_color\_major),  
 panel.background = element\_rect(fill = back\_color, color = font\_color)  
 )

# 3 Part 0: Data Cleaning Prep

We’re going to do some basic work here so we can get into the line-by-line cleaning tasks in the assignment a bit  
smarter

## 3.1 Load the Data

# Load the assignment data from CSV  
raw\_dat = read.csv("./308D.DA3.Data.csv", na = c("", "NA", "-999", "na", "n/a", "N/A"))  
  
# Rename columns to lower because why not  
colnames(raw\_dat) <- tolower( colnames(raw\_dat) )  
  
# Ensure that the numbers of each subject in the study are unique to prevent any duplicate data  
# if the size of the unique-entries only is the same as the whole vector then there are no duplicate subjects  
# NOTE: This fails if the colname of the subject ID is input wrong. So make sure you UPDATE the "test\_col"   
# entry below  
test\_colname = "x"  
  
test\_unique = ( length( unique( raw\_dat[test\_colname] ) ) == length(raw\_dat[test\_colname]))  
if(!test\_unique){  
 print("WARNING: There are duplicate data entries in the raw data")  
}else{  
 print("No duplicate entries detected in raw data")  
}

## [1] "No duplicate entries detected in raw data"

## 3.2 Name-Mapping

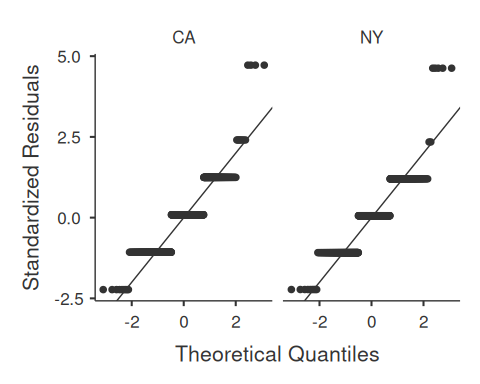
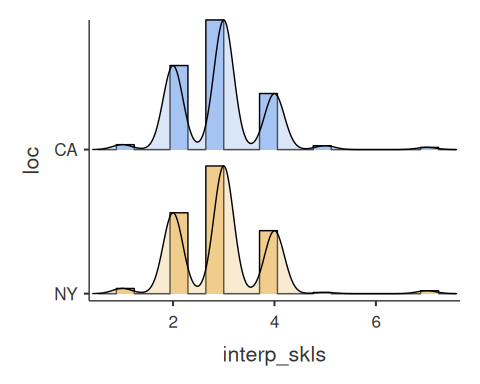
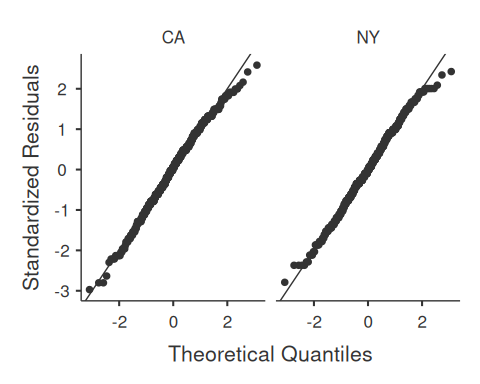
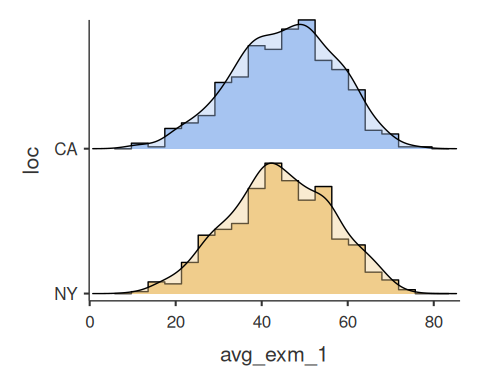
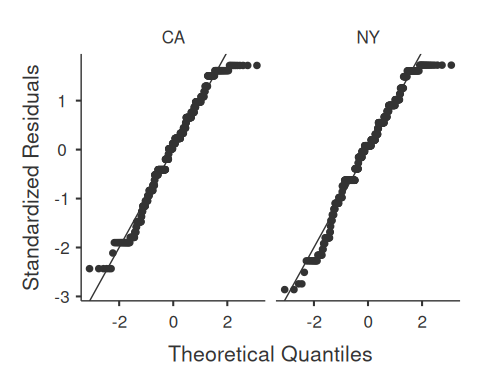
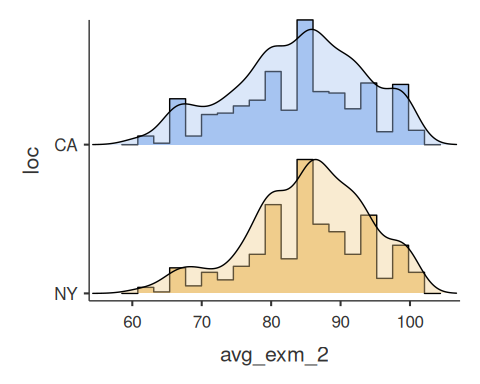
The names of the vars as given suck. We’re going to remap them all.

# We're going to map the column names as given in the dataframe to a set we prefer.  
raw\_names <- c("x", "school", "subject", "average.exam.2.grade", "average.exam.1.grade", "school.year",  
 "location", "interpersonal")  
map\_names <- c("idx", "schl\_lvl", "sub", "avg\_exm\_2", "avg\_exm\_1", "schl\_yr",  
 "loc", "interp\_skls")  
  
# Loop through each raw name and apply the corresponding map\_name  
dummy\_list <- list()  
my\_dat <- data.frame()  
for(iii in 1:length(raw\_names) ){  
  
 # Extract paired names  
 this\_map <- map\_names[iii]  
 this\_raw <- raw\_names[iii]  
   
 # Create a column in the new dataframe list named the name from map\_names  
 # We use a list because R sucks at dynamic binding  
 dummy\_list[[this\_map]] <- raw\_dat[[this\_raw]]  
   
}  
  
# Convert the list to a dataframe  
my\_dat <- as.data.frame(dummy\_list, stringsAsFactors = FALSE)

## 3.3 Descriptives

# Names of numeric vars. There are only 3 of them.  
cont\_names = c("avg\_exm\_2", "avg\_exm\_1", "interp\_skls")  
  
# We're going to use some split descriptives to help us understand mising values quantities  
loc\_descr = jmv::descriptives( my\_dat,  
 vars = cont\_names[],  
 splitBy = "loc",  
 hist = TRUE,  
 dens = TRUE,  
 qq = TRUE,  
 sd = TRUE,  
 variance = TRUE,  
 se = TRUE,  
 missing = TRUE  
 )  
print(loc\_descr)

##   
## DESCRIPTIVES  
##   
## Descriptives   
## ──────────────────────────────────────────────────────────────────────   
## loc avg\_exm\_2 avg\_exm\_1 interp\_skls   
## ──────────────────────────────────────────────────────────────────────   
## N CA 510 510 511   
## NY 486 482 486   
## Missing CA 1 1 0   
## NY 0 4 0   
## Mean CA 83.85098 45.30392 2.923679   
## NY 85.32716 44.17427 2.948560   
## Std. error mean CA 0.4158753 0.5259984 0.03818648   
## NY 0.3856874 0.5413730 0.03971142   
## Median CA 85.00000 46.00000 3   
## NY 86.00000 44.00000 3.000000   
## Standard deviation CA 9.391786 11.87872 0.8632173   
## NY 8.502635 11.88557 0.8754545   
## Variance CA 88.20565 141.1039 0.7451441   
## NY 72.29481 141.2669 0.7664206   
## Minimum CA 61 10 1   
## NY 61 11 1   
## Maximum CA 100 76 7   
## NY 100 73 7   
## ──────────────────────────────────────────────────────────────────────



print(" ")

## [1] " "

print("------")

## [1] "------"

print(" ")

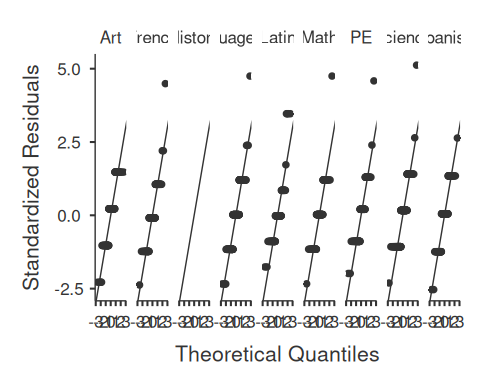
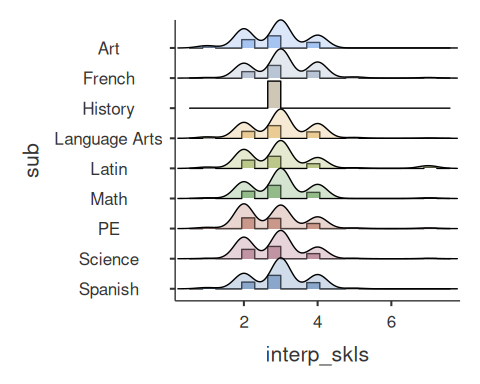
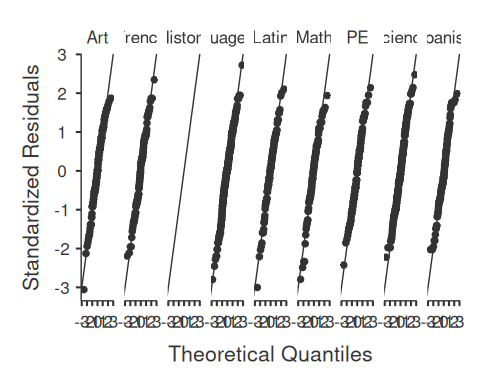
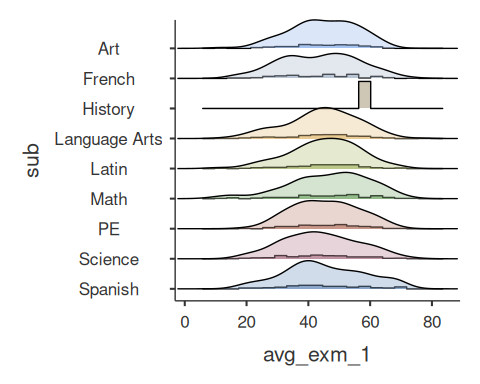
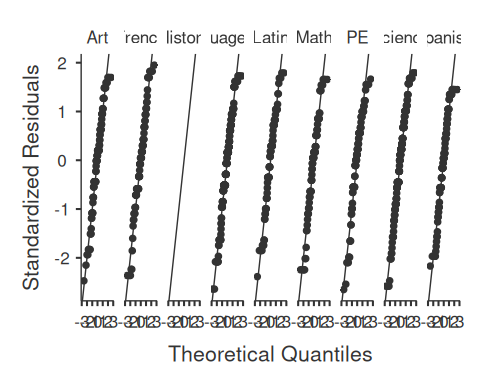
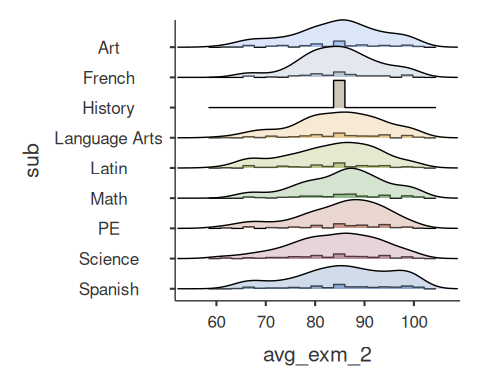
## [1] " "

sub\_descr = jmv::descriptives( my\_dat,  
 vars = cont\_names[],  
 splitBy = "sub",  
 hist = TRUE,  
 dens = TRUE,  
 qq = TRUE,  
 sd = TRUE,  
 variance = TRUE,  
 se = TRUE,  
 missing = TRUE  
 )

## Warning in qt(tCriticalValue, df = stats[["n"]] - 1): NaNs produced  
## Warning in qt(tCriticalValue, df = stats[["n"]] - 1): NaNs produced  
## Warning in qt(tCriticalValue, df = stats[["n"]] - 1): NaNs produced

print(sub\_descr)

##   
## DESCRIPTIVES  
##   
## Descriptives   
## ────────────────────────────────────────────────────────────────────────────────   
## sub avg\_exm\_2 avg\_exm\_1 interp\_skls   
## ────────────────────────────────────────────────────────────────────────────────   
## N Art 107 106 107   
## French 105 105 105   
## History 1 1 1   
## Language Arts 215 215 216   
## Latin 92 92 92   
## Math 93 93 93   
## PE 101 99 101   
## Science 177 177 177   
## Spanish 101 100 101   
## Missing Art 0 1 0   
## French 0 0 0   
## History 0 0 0   
## Language Arts 1 1 0   
## Latin 0 0 0   
## Math 0 0 0   
## PE 0 2 0   
## Science 0 0 0   
## Spanish 0 1 0   
## Mean Art 84.12150 44.86792 2.822430   
## French 84.62857 43.53333 3.076190   
## History 86.00000 60.00000 3.000000   
## Language Arts 84.57674 44.44651 2.981481   
## Latin 83.28261 45.09783 3.021739   
## Math 85.56989 46.60215 2.978495   
## PE 84.97030 45.49495 2.811881   
## Science 84.01695 43.96610 2.864407   
## Spanish 85.58416 45.25000 2.960396   
## Std. error mean Art 0.9042222 1.044222 0.07721694   
## French 0.7694079 1.225497 0.08526541   
## History   
## Language Arts 0.6087903 0.7906091 0.05757180   
## Latin 0.9734073 1.184292 0.1197009   
## Math 0.9035277 1.359102 0.08780010   
## PE 0.8993348 1.056766 0.09088440   
## Science 0.6697800 0.9116710 0.06069115   
## Spanish 0.9900762 1.295164 0.07697343   
## Median Art 85 45.00000 3   
## French 85 45 3   
## History 86 60 3   
## Language Arts 86 45 3.000000   
## Latin 85.00000 46.00000 3.000000   
## Math 87 48 3   
## PE 86 45 3   
## Science 85 43 3   
## Spanish 86 43.00000 3   
## Standard deviation Art 9.353347 10.75093 0.7987382   
## French 7.884085 12.55761 0.8737105   
## History   
## Language Arts 8.926618 11.59260 0.8461292   
## Latin 9.336595 11.35933 1.148130   
## Math 8.713305 13.10670 0.8467135   
## PE 9.038203 10.51469 0.9133769   
## Science 8.910843 12.12899 0.8074432   
## Spanish 9.950143 12.95164 0.7735734   
## Variance Art 87.48510 115.5824 0.6379827   
## French 62.15879 157.6936 0.7633700   
## History   
## Language Arts 79.68450 134.3885 0.7159345   
## Latin 87.17200 129.0343 1.318204   
## Math 75.92169 171.7856 0.7169238   
## PE 81.68911 110.5586 0.8342574   
## Science 79.40312 147.1125 0.6519646   
## Spanish 99.00535 167.7449 0.5984158   
## Minimum Art 61 12 1   
## French 66 16 1   
## History 86 60 3   
## Language Arts 61 12 1   
## Latin 61 11 1   
## Math 66 10 1   
## PE 61 20 1   
## Science 61 17 1   
## Spanish 64 19 1   
## Maximum Art 100 65 4   
## French 100 73 7   
## History 86 60 3   
## Language Arts 100 76 7   
## Latin 100 69 7   
## Math 100 72 7   
## PE 100 68 7   
## Science 100 74 7   
## Spanish 100 71 5   
## ────────────────────────────────────────────────────────────────────────────────



print(" ")

## [1] " "

print("------")

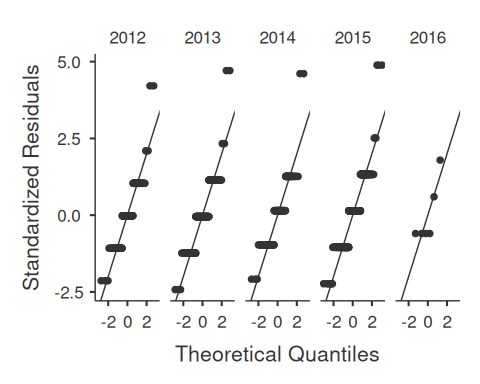
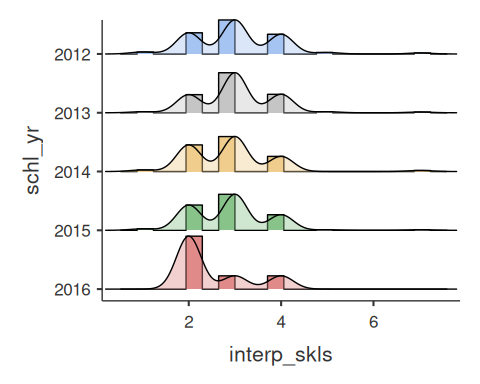
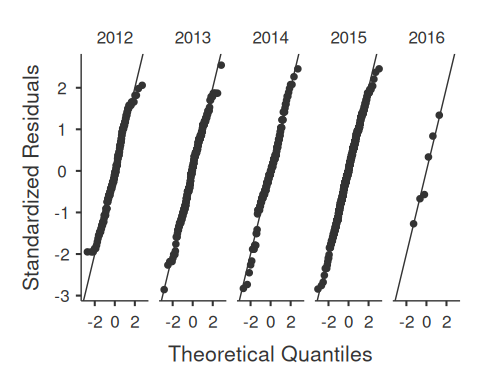
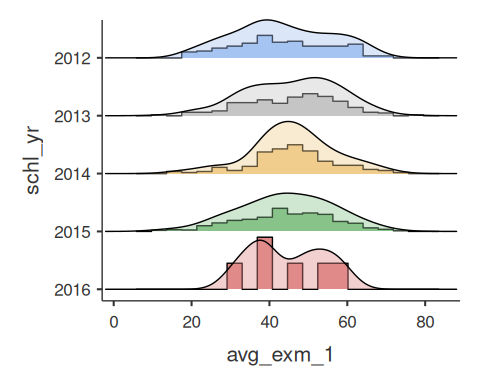
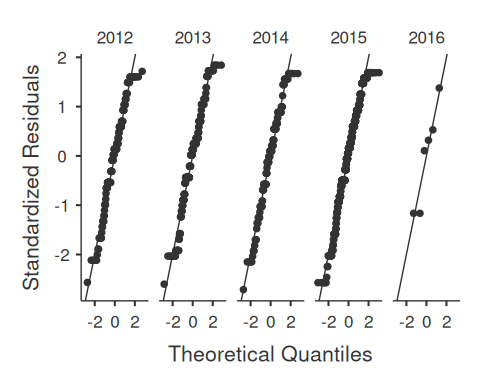
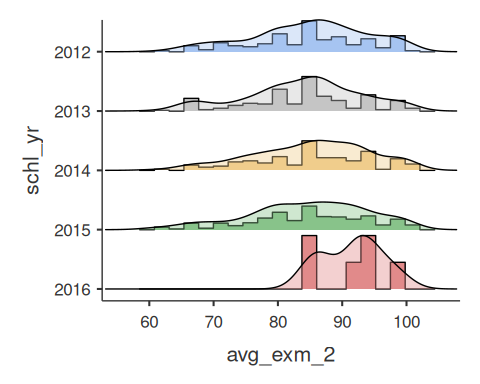
## [1] "------"

print(" ")

## [1] " "

yr\_descr = jmv::descriptives( my\_dat,  
 vars = cont\_names[],  
 splitBy = "schl\_yr",  
 hist = TRUE,  
 dens = TRUE,  
 qq = TRUE,  
 sd = TRUE,  
 variance = TRUE,  
 se = TRUE,  
 missing = TRUE  
 )  
print(yr\_descr)

##   
## DESCRIPTIVES  
##   
## Descriptives   
## ──────────────────────────────────────────────────────────────────────────   
## schl\_yr avg\_exm\_2 avg\_exm\_1 interp\_skls   
## ──────────────────────────────────────────────────────────────────────────   
## N 2012 161 160 161   
## 2013 228 227 228   
## 2014 152 152 152   
## 2015 442 440 443   
## 2016 6 6 6   
## Missing 2012 0 1 0   
## 2013 0 1 0   
## 2014 0 0 0   
## 2015 1 3 0   
## 2016 0 0 0   
## Mean 2012 84.78261 43.31250 3.018634   
## 2013 83.84211 45.85022 3.039474   
## 2014 85.13816 45.98026 2.868421   
## 2015 84.54525 44.33864 2.882619   
## 2016 91.50000 44.66667 2.500000   
## Std. error mean 2012 0.6993062 0.9863928 0.07449209   
## 2013 0.5812578 0.7866784 0.05571253   
## 2014 0.7217462 0.8599403 0.07271811   
## 2015 0.4353860 0.5759222 0.04003870   
## 2016 1.927866 4.063387 0.3415650   
## Median 2012 86 42.00000 3   
## 2013 85.00000 47 3.000000   
## 2014 86.00000 46.00000 3.000000   
## 2015 86.00000 44.00000 3   
## 2016 92.50000 43.50000 2.000000   
## Standard deviation 2012 8.873201 12.47699 0.9451987   
## 2013 8.776801 11.85251 0.8412408   
## 2014 8.898284 10.60206 0.8965291   
## 2015 9.153465 12.08065 0.8427172   
## 2016 4.722288 9.953224 0.8366600   
## Variance 2012 78.73370 155.6753 0.8934006   
## 2013 77.03223 140.4819 0.7076861   
## 2014 79.17946 112.4036 0.8037644   
## 2015 83.78593 145.9420 0.7101723   
## 2016 22.30000 99.06667 0.7000000   
## Minimum 2012 62 19 1   
## 2013 61 12 1   
## 2014 61 16 1   
## 2015 61 10 1   
## 2016 86 32 2   
## Maximum 2012 100 69 7   
## 2013 100 76 7   
## 2014 100 72 7   
## 2015 100 74 7   
## 2016 98 58 4   
## ──────────────────────────────────────────────────────────────────────────



print(" ")

## [1] " "

print("------")

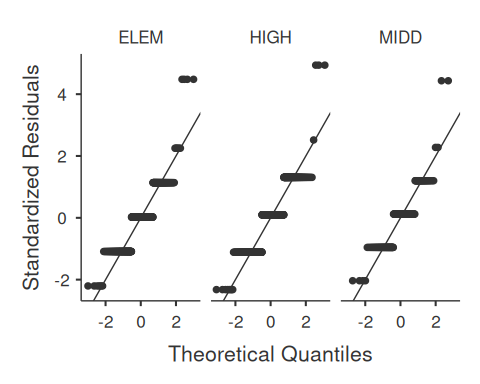
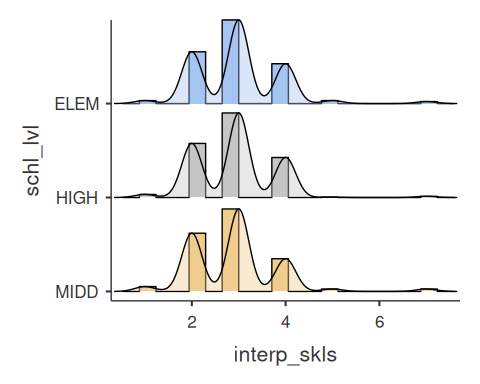
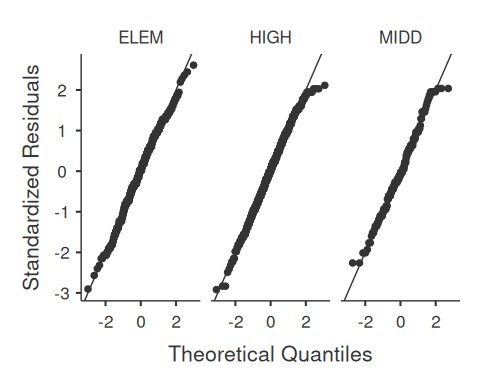
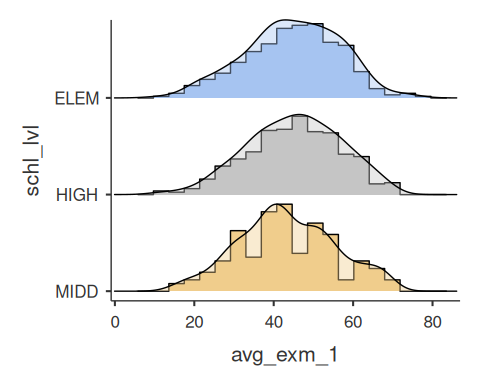
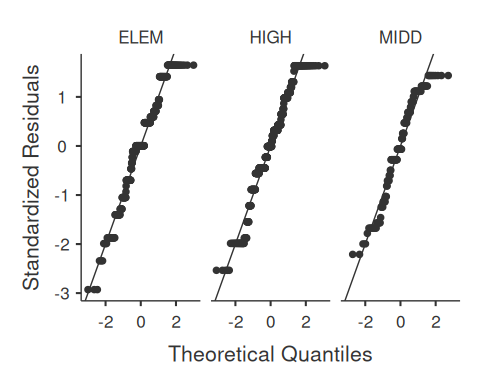
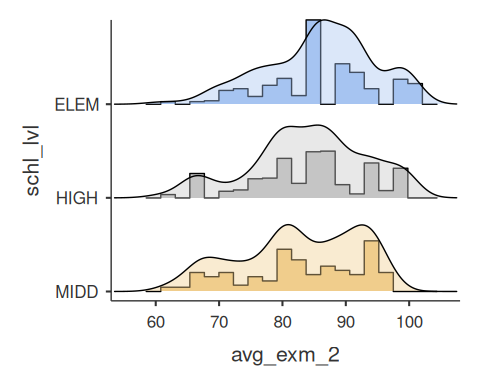
## [1] "------"

print(" ")

## [1] " "

lvl\_descr = jmv::descriptives( my\_dat,  
 vars = cont\_names[],  
 splitBy = "schl\_lvl",  
 hist = TRUE,  
 dens = TRUE,  
 qq = TRUE,  
 sd = TRUE,  
 variance = TRUE,  
 se = TRUE,  
 missing = TRUE  
 )  
print(lvl\_descr)

##   
## DESCRIPTIVES  
##   
## Descriptives   
## ───────────────────────────────────────────────────────────────────────────   
## schl\_lvl avg\_exm\_2 avg\_exm\_1 interp\_skls   
## ───────────────────────────────────────────────────────────────────────────   
## N ELEM 368 367 368   
## HIGH 481 478 481   
## MIDD 150 150 151   
## Missing ELEM 0 1 0   
## HIGH 0 3 0   
## MIDD 1 1 0   
## Mean ELEM 85.95924 44.74387 2.978261   
## HIGH 84.09979 45.20921 2.918919   
## MIDD 82.61333 43.35333 2.887417   
## Std. error mean ELEM 0.4442833 0.6245972 0.04680166   
## HIGH 0.4154585 0.5361304 0.03768221   
## MIDD 0.7613591 0.9873081 0.07549282   
## Median ELEM 86.00000 45 3.000000   
## HIGH 84 45.50000 3   
## MIDD 82.00000 42.50000 3   
## Standard deviation ELEM 8.522832 11.96556 0.8978116   
## HIGH 9.111715 11.72153 0.8264354   
## MIDD 9.324707 12.09201 0.9276713   
## Variance ELEM 72.63866 143.1747 0.8060656   
## HIGH 83.02335 137.3943 0.6829955   
## MIDD 86.95016 146.2166 0.8605740   
## Minimum ELEM 61 10 1   
## HIGH 61 11 1   
## MIDD 62 16 1   
## Maximum ELEM 100 76 7   
## HIGH 99 70 7   
## MIDD 96 68 7   
## ───────────────────────────────────────────────────────────────────────────



print(" ")

## [1] " "

print("------")

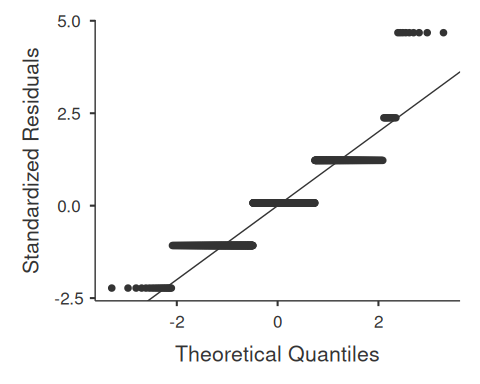
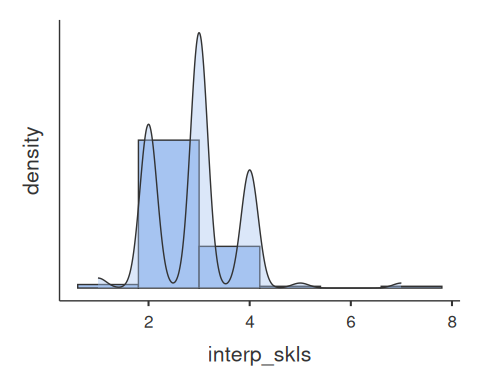
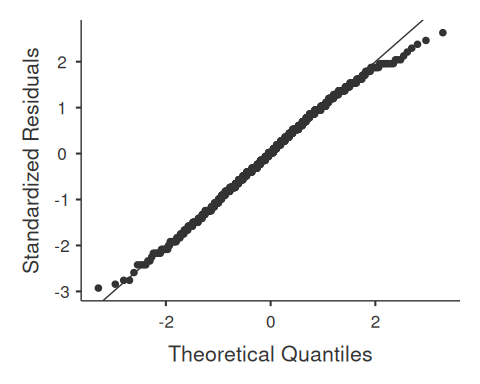
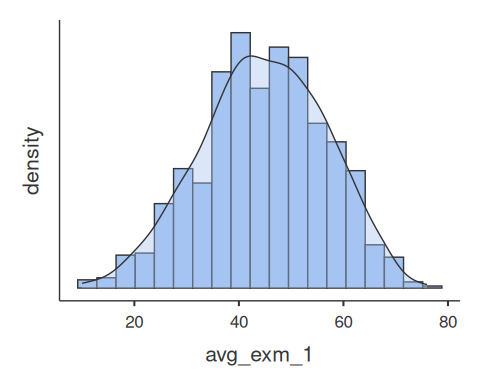
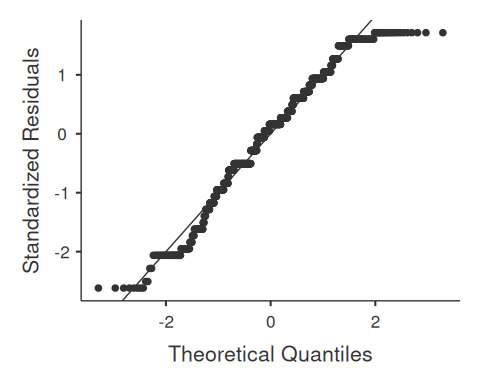
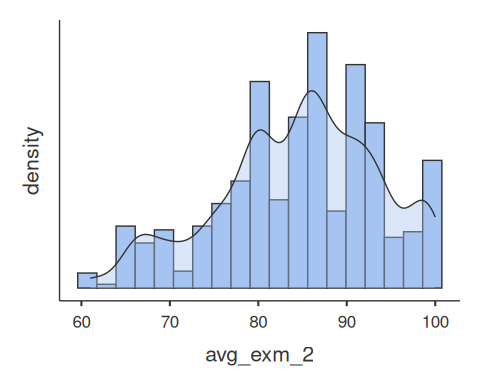
## [1] "------"

print(" ")

## [1] " "

lvl\_descr = jmv::descriptives( my\_dat,  
 vars = cont\_names[],  
 hist = TRUE,  
 dens = TRUE,  
 qq = TRUE,  
 sd = TRUE,  
 variance = TRUE,  
 se = TRUE,  
 missing = TRUE  
 )  
print(lvl\_descr)

##   
## DESCRIPTIVES  
##   
## Descriptives   
## ───────────────────────────────────────────────────────────────   
## avg\_exm\_2 avg\_exm\_1 interp\_skls   
## ───────────────────────────────────────────────────────────────   
## N 999 995 1000   
## Missing 1 5 0   
## Mean 84.56156 44.75779 2.936000   
## Std. error mean 0.2847790 0.3763944 0.02747105   
## Median 86 45 3.000000   
## Standard deviation 9.001000 11.87284 0.8687109   
## Variance 81.01800 140.9644 0.7546587   
## Minimum 61 10 1   
## Maximum 100 76 7   
## ───────────────────────────────────────────────────────────────



# 4 Part 1: Data Cleaning Questions / Tasks

Download the dataset posted on canvas called “308A.DA3.Data.csv” and create an RMarkdown file.  
This DA consists of three categories of tasks for you to complete – data cleaning (complete in RStudio),  
data querying (complete in RStudio and respond to questions below), and a code investigation (respond below).  
Upload both a word document with your completed questions and your knitted RMarkdown file in either word or pdf format.

The dataset contains data regarding average grades for Exam 1 and 2 for various classes,  
each case is classified by school level (elem, midd, high), subject, year, and location.

## 4.1 Handle Missing Data for all Variables

Question #1 of Data Cleaning Section. NOTE: We loaded the data with the following code snippet (above):  
raw\_dat = read.csv("./308D.DA3.Data.csv", na = c("", "NA", "-999", "na", "n/a, "N/A")) …. so all values in raw\_dat (and therefore my\_dat) should be na if they were missing.

We look for unique values in all non-numeric variables in our dataset, and we check for non-numeric values in our  
numeric variables to confirm this worked as expected.

# We already confirmed that "x" as loaded had 1000 unique entries in it in our load data section, so we don't need to   
# handle that column  
  
# Look for unique entries in each of the categorical columns. This will help us understand how creatively formatted   
# data in each column is. We can spot weird missing data or poorly formatted entries this way.  
uni\_lvl <- unique(my\_dat$schl\_lvl)  
uni\_sub <- unique(my\_dat$sub)  
uni\_yr <- unique(my\_dat$schl\_yr)  
uni\_loc <- unique(my\_dat$loc)  
  
# Print unique values to see if we have any permutations like CA, ca, and CA in loc  
print(uni\_lvl)

## [1] "ELEM" "MIDD" "HIGH"

print(uni\_sub)

## [1] "Math" "History" "Science" "French"   
## [5] "Language Arts" "Art" "Spanish" "PE"   
## [9] "Latin" NA

print(uni\_yr)

## [1] 2015 2013 2012 2014 NA 2016

print(uni\_loc)

## [1] "CA" "NY" NA

# Output indicates no weird duplicate formats so we don't need to mess with that. Yay!  
  
## EXPLANATION - WHY  
# Per our descriptives we have:  
# 1x datapoint in the "history" subject, so that should be removed (it's a non-useful level in subject)  
# 5 missing exam 1 scores and 1 missing exam 2 scores and no missing interpersonal skills data.  
# The largest missing:sample\_size ratio we have, if we slice the data by categories, is 2 missing scores  
# for 99 PE samples  
#  
# So given all that, our power shouldn't be severely damaged if we just drop all rows missing data in the dataset.  
# (99 isn't a big sample but difference between 99 and 97 isn't much)  
my\_dat\_with\_missing <- my\_dat  
my\_dat <- na.omit(my\_dat)  
  
# Toss our one dumb history datapoint  
my\_dat <- my\_dat[my\_dat$sub != "History", ]  
  
# Print data loss metric  
data\_loss = 1 - ( length(my\_dat$idx) / length(my\_dat\_with\_missing$idx) )  
cat( paste("Total Percentage of data lost due to dropping missing: ", 100\*data\_loss, "%\n\n", sep="") )

## Total Percentage of data lost due to dropping missing: 2.6%

## 4.2 Convert Categorical Variables

Convert School Level, Subject, Year, and Location to categorical variables

# We convert all categorical variables to factors and then dummy code each  
  
# We convert all 4x to factors  
my\_dat$schl\_lvl <- as.factor(my\_dat$schl\_lvl)  
my\_dat$sub <- as.factor(my\_dat$sub)  
my\_dat$schl\_yr <- as.factor(my\_dat$schl\_yr)  
my\_dat$loc <- as.factor(my\_dat$loc)  
  
# ---  
# Dummy Code School Level with Elem as the reference  
my\_dat$lvl\_mid\_refd\_elem <- as.numeric(my\_dat$schl\_lvl)  
my\_dat$lvl\_high\_refd\_elem <- as.numeric(my\_dat$schl\_lvl)  
  
# Midd  
my\_dat$lvl\_mid\_refd\_elem[my\_dat$schl\_lvl == "ELEM"] <- 0  
my\_dat$lvl\_mid\_refd\_elem[my\_dat$schl\_lvl == "MIDD"] <- 1  
my\_dat$lvl\_mid\_refd\_elem[my\_dat$schl\_lvl == "HIGH"] <- 0  
  
# High  
my\_dat$lvl\_high\_refd\_elem[my\_dat$schl\_lvl == "ELEM"] <- 0  
my\_dat$lvl\_high\_refd\_elem[my\_dat$schl\_lvl == "MIDD"] <- 0  
my\_dat$lvl\_high\_refd\_elem[my\_dat$schl\_lvl == "HIGH"] <- 1  
  
# Move Category to front of DCs  
my\_dat <- my\_dat %>% relocate(schl\_lvl, .before=lvl\_mid\_refd\_elem)  
  
# ---  
# Dummy Code subject with math as the reference - Leave history out since we dropped it  
# my\_dat$sub\_hist\_refd\_math <- as.numeric(my\_dat$sub)  
my\_dat$sub\_sci\_refd\_math <- as.numeric(my\_dat$sub)  
my\_dat$sub\_frnch\_refd\_math <- as.numeric(my\_dat$sub)  
my\_dat$sub\_lang\_refd\_math <- as.numeric(my\_dat$sub)  
my\_dat$sub\_art\_refd\_math <- as.numeric(my\_dat$sub)  
my\_dat$sub\_span\_refd\_math <- as.numeric(my\_dat$sub)  
my\_dat$sub\_pe\_refd\_math <- as.numeric(my\_dat$sub)  
my\_dat$sub\_latn\_refd\_math <- as.numeric(my\_dat$sub)  
  
# # History  
# my\_dat$sub\_hist\_refd\_math[my\_dat$sub == "Math"] <- 0  
# my\_dat$sub\_hist\_refd\_math[my\_dat$sub == "History"] <- 1  
# my\_dat$sub\_hist\_refd\_math[my\_dat$sub == "Science"] <- 0  
# my\_dat$sub\_hist\_refd\_math[my\_dat$sub == "French"] <- 0  
# my\_dat$sub\_hist\_refd\_math[my\_dat$sub == "Language Arts"] <- 0  
# my\_dat$sub\_hist\_refd\_math[my\_dat$sub == "Art"] <- 0  
# my\_dat$sub\_hist\_refd\_math[my\_dat$sub == "Spanish"] <- 0  
# my\_dat$sub\_hist\_refd\_math[my\_dat$sub == "PE"] <- 0  
# my\_dat$sub\_hist\_refd\_math[my\_dat$sub == "Latin"] <- 0  
# my\_dat$sub\_hist\_refd\_math[is.na(my\_dat$sub)] <- NA  
  
# Science  
my\_dat$sub\_sci\_refd\_math[my\_dat$sub == "Math"] <- 0  
my\_dat$sub\_sci\_refd\_math[my\_dat$sub == "History"] <- 0  
my\_dat$sub\_sci\_refd\_math[my\_dat$sub == "Science"] <- 1  
my\_dat$sub\_sci\_refd\_math[my\_dat$sub == "French"] <- 0  
my\_dat$sub\_sci\_refd\_math[my\_dat$sub == "Language Arts"] <- 0  
my\_dat$sub\_sci\_refd\_math[my\_dat$sub == "Art"] <- 0  
my\_dat$sub\_sci\_refd\_math[my\_dat$sub == "Spanish"] <- 0  
my\_dat$sub\_sci\_refd\_math[my\_dat$sub == "PE"] <- 0  
my\_dat$sub\_sci\_refd\_math[my\_dat$sub == "Latin"] <- 0  
my\_dat$sub\_sci\_refd\_math[is.na(my\_dat$sub)] <- NA  
  
# French  
my\_dat$sub\_frnch\_refd\_math[my\_dat$sub == "Math"] <- 0  
my\_dat$sub\_frnch\_refd\_math[my\_dat$sub == "History"] <- 0  
my\_dat$sub\_frnch\_refd\_math[my\_dat$sub == "Science"] <- 0  
my\_dat$sub\_frnch\_refd\_math[my\_dat$sub == "French"] <- 1  
my\_dat$sub\_frnch\_refd\_math[my\_dat$sub == "Language Arts"] <- 0  
my\_dat$sub\_frnch\_refd\_math[my\_dat$sub == "Art"] <- 0  
my\_dat$sub\_frnch\_refd\_math[my\_dat$sub == "Spanish"] <- 0  
my\_dat$sub\_frnch\_refd\_math[my\_dat$sub == "PE"] <- 0  
my\_dat$sub\_frnch\_refd\_math[my\_dat$sub == "Latin"] <- 0  
my\_dat$sub\_frnch\_refd\_math[is.na(my\_dat$sub)] <- NA  
  
# Language Arts  
my\_dat$sub\_lang\_refd\_math[my\_dat$sub == "Math"] <- 0  
my\_dat$sub\_lang\_refd\_math[my\_dat$sub == "History"] <- 0  
my\_dat$sub\_lang\_refd\_math[my\_dat$sub == "Science"] <- 0  
my\_dat$sub\_lang\_refd\_math[my\_dat$sub == "French"] <- 0  
my\_dat$sub\_lang\_refd\_math[my\_dat$sub == "Language Arts"] <- 1  
my\_dat$sub\_lang\_refd\_math[my\_dat$sub == "Art"] <- 0  
my\_dat$sub\_lang\_refd\_math[my\_dat$sub == "Spanish"] <- 0  
my\_dat$sub\_lang\_refd\_math[my\_dat$sub == "PE"] <- 0  
my\_dat$sub\_lang\_refd\_math[my\_dat$sub == "Latin"] <- 0  
my\_dat$sub\_lang\_refd\_math[is.na(my\_dat$sub)] <- NA  
  
# Art  
my\_dat$sub\_art\_refd\_math[my\_dat$sub == "Math"] <- 0  
my\_dat$sub\_art\_refd\_math[my\_dat$sub == "History"] <- 0  
my\_dat$sub\_art\_refd\_math[my\_dat$sub == "Science"] <- 0  
my\_dat$sub\_art\_refd\_math[my\_dat$sub == "French"] <- 0  
my\_dat$sub\_art\_refd\_math[my\_dat$sub == "Language Arts"] <- 0  
my\_dat$sub\_art\_refd\_math[my\_dat$sub == "Art"] <- 1  
my\_dat$sub\_art\_refd\_math[my\_dat$sub == "Spanish"] <- 0  
my\_dat$sub\_art\_refd\_math[my\_dat$sub == "PE"] <- 0  
my\_dat$sub\_art\_refd\_math[my\_dat$sub == "Latin"] <- 0  
my\_dat$sub\_art\_refd\_math[is.na(my\_dat$sub)] <- NA  
  
# Spanish  
my\_dat$sub\_span\_refd\_math[my\_dat$sub == "Math"] <- 0  
my\_dat$sub\_span\_refd\_math[my\_dat$sub == "History"] <- 0  
my\_dat$sub\_span\_refd\_math[my\_dat$sub == "Science"] <- 0  
my\_dat$sub\_span\_refd\_math[my\_dat$sub == "French"] <- 0  
my\_dat$sub\_span\_refd\_math[my\_dat$sub == "Language Arts"] <- 0  
my\_dat$sub\_span\_refd\_math[my\_dat$sub == "Art"] <- 0  
my\_dat$sub\_span\_refd\_math[my\_dat$sub == "Spanish"] <- 1  
my\_dat$sub\_span\_refd\_math[my\_dat$sub == "PE"] <- 0  
my\_dat$sub\_span\_refd\_math[my\_dat$sub == "Latin"] <- 0  
my\_dat$sub\_span\_refd\_math[is.na(my\_dat$sub)] <- NA  
  
# PE  
my\_dat$sub\_pe\_refd\_math[my\_dat$sub == "Math"] <- 0  
my\_dat$sub\_pe\_refd\_math[my\_dat$sub == "History"] <- 0  
my\_dat$sub\_pe\_refd\_math[my\_dat$sub == "Science"] <- 0  
my\_dat$sub\_pe\_refd\_math[my\_dat$sub == "French"] <- 0  
my\_dat$sub\_pe\_refd\_math[my\_dat$sub == "Language Arts"] <- 0  
my\_dat$sub\_pe\_refd\_math[my\_dat$sub == "Art"] <- 0  
my\_dat$sub\_pe\_refd\_math[my\_dat$sub == "Spanish"] <- 0  
my\_dat$sub\_pe\_refd\_math[my\_dat$sub == "PE"] <- 1  
my\_dat$sub\_pe\_refd\_math[my\_dat$sub == "Latin"] <- 0  
my\_dat$sub\_pe\_refd\_math[is.na(my\_dat$sub)] <- NA  
  
# Latin  
my\_dat$sub\_latn\_refd\_math[my\_dat$sub == "Math"] <- 0  
my\_dat$sub\_latn\_refd\_math[my\_dat$sub == "History"] <- 1  
my\_dat$sub\_latn\_refd\_math[my\_dat$sub == "Science"] <- 0  
my\_dat$sub\_latn\_refd\_math[my\_dat$sub == "French"] <- 0  
my\_dat$sub\_latn\_refd\_math[my\_dat$sub == "Language Arts"] <- 0  
my\_dat$sub\_latn\_refd\_math[my\_dat$sub == "Art"] <- 0  
my\_dat$sub\_latn\_refd\_math[my\_dat$sub == "Spanish"] <- 0  
my\_dat$sub\_latn\_refd\_math[my\_dat$sub == "PE"] <- 0  
my\_dat$sub\_latn\_refd\_math[my\_dat$sub == "Latin"] <- 1  
my\_dat$sub\_latn\_refd\_math[is.na(my\_dat$sub)] <- NA  
  
# Move Category to front of DCs  
# my\_dat <- my\_dat %>% relocate(sub, .before=sub\_hist\_refd\_math)  
my\_dat <- my\_dat %>% relocate(sub, .before=sub\_sci\_refd\_math )  
  
# ---  
# Dummy Code yr with (shouldn't be a categorical) with 2012 as the reference  
my\_dat$yr\_2013\_refd\_2012 <- as.numeric(my\_dat$schl\_yr)  
my\_dat$yr\_2014\_refd\_2012 <- as.numeric(my\_dat$schl\_yr)  
my\_dat$yr\_2015\_refd\_2012 <- as.numeric(my\_dat$schl\_yr)  
my\_dat$yr\_2016\_refd\_2012 <- as.numeric(my\_dat$schl\_yr)  
  
# 2013  
my\_dat$yr\_2013\_refd\_2012[my\_dat$schl\_yr == "2012"] <- 0  
my\_dat$yr\_2013\_refd\_2012[my\_dat$schl\_yr == "2013"] <- 1  
my\_dat$yr\_2013\_refd\_2012[my\_dat$schl\_yr == "2014"] <- 0  
my\_dat$yr\_2013\_refd\_2012[my\_dat$schl\_yr == "2015"] <- 0  
my\_dat$yr\_2013\_refd\_2012[my\_dat$schl\_yr == "2016"] <- 0  
my\_dat$yr\_2013\_refd\_2012[is.na(my\_dat$schl\_yr)] <- NA  
  
# 2014  
my\_dat$yr\_2014\_refd\_2012[my\_dat$schl\_yr == "2012"] <- 0  
my\_dat$yr\_2014\_refd\_2012[my\_dat$schl\_yr == "2013"] <- 0  
my\_dat$yr\_2014\_refd\_2012[my\_dat$schl\_yr == "2014"] <- 1  
my\_dat$yr\_2014\_refd\_2012[my\_dat$schl\_yr == "2015"] <- 0  
my\_dat$yr\_2014\_refd\_2012[my\_dat$schl\_yr == "2016"] <- 0  
my\_dat$yr\_2014\_refd\_2012[is.na(my\_dat$schl\_yr)] <- NA  
  
# 2015   
my\_dat$yr\_2015\_refd\_2012[my\_dat$schl\_yr == "2012"] <- 0  
my\_dat$yr\_2015\_refd\_2012[my\_dat$schl\_yr == "2013"] <- 0  
my\_dat$yr\_2015\_refd\_2012[my\_dat$schl\_yr == "2014"] <- 0  
my\_dat$yr\_2015\_refd\_2012[my\_dat$schl\_yr == "2015"] <- 1  
my\_dat$yr\_2015\_refd\_2012[my\_dat$schl\_yr == "2016"] <- 0  
my\_dat$yr\_2015\_refd\_2012[is.na(my\_dat$schl\_yr)] <- NA  
  
# 2016  
my\_dat$yr\_2016\_refd\_2012[my\_dat$schl\_yr == "2012"] <- 0  
my\_dat$yr\_2016\_refd\_2012[my\_dat$schl\_yr == "2013"] <- 0  
my\_dat$yr\_2016\_refd\_2012[my\_dat$schl\_yr == "2014"] <- 0  
my\_dat$yr\_2016\_refd\_2012[my\_dat$schl\_yr == "2015"] <- 0  
my\_dat$yr\_2016\_refd\_2012[my\_dat$schl\_yr == "2016"] <- 1  
my\_dat$yr\_2016\_refd\_2012[is.na(my\_dat$schl\_yr)] <- NA  
  
# Move Category to front of DCs  
my\_dat <- my\_dat %>% relocate(schl\_yr, .before=yr\_2013\_refd\_2012)  
  
# ---  
# Dummy Code Location with California as the reference.  
my\_dat$loc\_ny\_refd\_ca <- as.numeric(my\_dat$loc)  
  
# NY - Handle NA cases  
my\_dat$loc\_ny\_refd\_ca[my\_dat$loc == "CA"] <- 0  
my\_dat$loc\_ny\_refd\_ca[my\_dat$loc == "NY"] <- 1  
my\_dat$loc\_ny\_refd\_ca[is.na(my\_dat$loc)] <- NA  
  
# Move Category to front of DCs  
my\_dat <- my\_dat %>% relocate(loc, .before=loc\_ny\_refd\_ca)

## 4.3 Rename the Variables “exam1” and “exam2”

# I like my names better  
my\_dat <- my\_dat %>% dplyr::rename(exam1 = avg\_exm\_1)  
my\_dat <- my\_dat %>% dplyr::rename(exam2 = avg\_exm\_2)

## 4.4 Check the Alpha for “exam1” and “exam2”

Check the Alpha for “exam1” and “exam2” to see if we can make a composite score.

# Using column names slice so I don't have to pay attention to what order columns are in  
cronbachs\_alpha( my\_dat[c("exam1", "exam2")] )

## [1] 0.06939527

## 4.5 Combine Exam Grades for Each Classes

Create 1 variable for exam grade for each class (average of the two)

# I'm assuming this means to create a column, exam\_mean, which just row-wise means exam1 and exam2. (i.e. each row has  
# a unique exam\_mean value). But this could also be interpreted as meaning across both exams for each category of class   
# (i.e. every "math" exam\_mean would be equal). I don't see how the latter is helpful so I'm going to do the former.  
my\_dat$exam\_mean <- rowMeans(my\_dat[ , c("exam1", "exam2")])

## 4.6 Reorder the Columns

Reorder the Columns so all categories (level, subject, year, location) are listed first,  
followed by Interpersonal, Exam 1, Exam 2, and average Exam

# First we move Interpersonal to the back, then all the exams  
my\_dat <- my\_dat %>% relocate(interp\_skls, .after=last\_col())  
my\_dat <- my\_dat %>% relocate(exam1, .after=last\_col())  
my\_dat <- my\_dat %>% relocate(exam2, .after=last\_col())  
my\_dat <- my\_dat %>% relocate(exam\_mean, .after=last\_col())

## 4.7 Construct Reverse Codes

There was an error in qualtrics and the scores for Interpersonal skills were not set up with  
reverse coding. Reverse code the Interpersonal scores using R.

# Per our descriptives, interp skills runs 1 to 7, so we need to map 1 to 7, 7 to 1, and etc. in the appropriate order  
my\_dat\_rev\_code <- my\_dat  
my\_dat$interp\_skls <- dplyr::recode(my\_dat$interp\_skls, '1'=7, '2'=6, '3'=5, '4'=4, '5'=3, '6'=2, '7'=1)

## 4.8 Standardize the Exam and Interpersonal Scores

Standardize the Exam and Interpersonal Scores for ease of comparison.

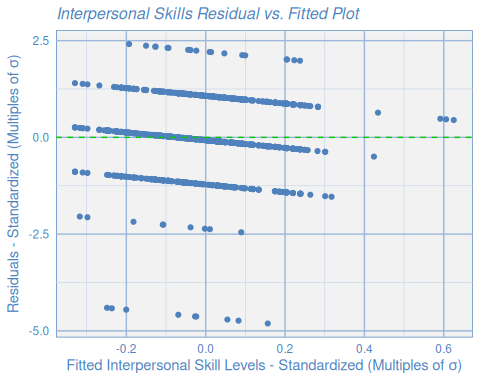
# Save unstandardized data for reference  
my\_dat\_unstd <- my\_dat  
  
# Scale the 4x numeric vars we have  
my\_dat$interp\_skls <- scale( my\_dat$interp\_skls, center = TRUE, scale = TRUE )[,1]  
my\_dat$exam1 <- scale( my\_dat$exam1, center = TRUE, scale = TRUE )[,1]  
my\_dat$exam2 <- scale( my\_dat$exam2, center = TRUE, scale = TRUE )[,1]  
my\_dat$exam\_mean <- scale( my\_dat$exam\_mean, center = TRUE, scale = TRUE )[,1]

## 4.9 Dummy Code Location

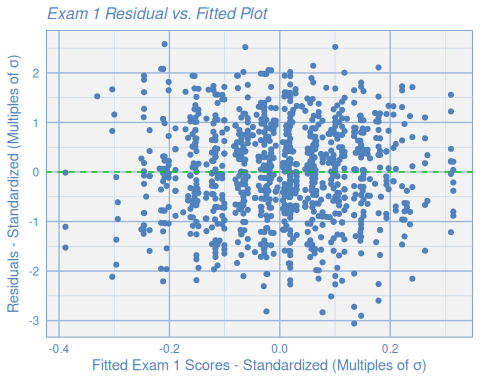
I already did this in the “create categorical variables” section above.

## 4.10 Detect Outliers and Handle Accordingly.

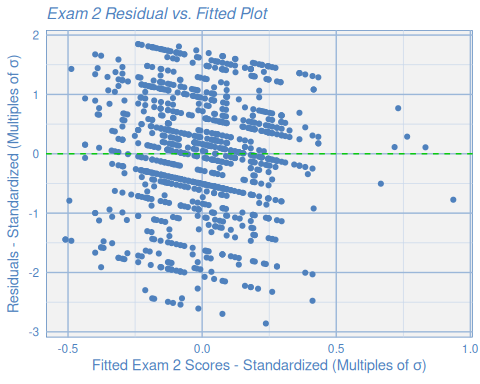
# Specify the fully saturated model for each of the 4x numeric outcome variables (exam scores and interpersonal skills)  
mod\_interp <- lm( interp\_skls ~ lvl\_mid\_refd\_elem +   
 lvl\_high\_refd\_elem +  
 sub\_sci\_refd\_math +  
 sub\_frnch\_refd\_math +  
 sub\_lang\_refd\_math +  
 sub\_art\_refd\_math +  
 sub\_span\_refd\_math +  
 sub\_pe\_refd\_math +  
 sub\_latn\_refd\_math +   
 yr\_2013\_refd\_2012 +  
 yr\_2014\_refd\_2012 +  
 yr\_2015\_refd\_2012 +  
 yr\_2016\_refd\_2012 +   
 loc\_ny\_refd\_ca,   
 data = my\_dat   
 )  
mod\_exam1 <- lm( exam1 ~ lvl\_mid\_refd\_elem +   
 lvl\_high\_refd\_elem +  
 sub\_sci\_refd\_math +  
 sub\_frnch\_refd\_math +  
 sub\_lang\_refd\_math +  
 sub\_art\_refd\_math +  
 sub\_span\_refd\_math +  
 sub\_pe\_refd\_math +  
 sub\_latn\_refd\_math +   
 yr\_2013\_refd\_2012 +  
 yr\_2014\_refd\_2012 +  
 yr\_2015\_refd\_2012 +  
 yr\_2016\_refd\_2012 +   
 loc\_ny\_refd\_ca,   
 data = my\_dat   
 )  
mod\_exam2 <- lm( exam2 ~ lvl\_mid\_refd\_elem +   
 lvl\_high\_refd\_elem +  
 sub\_sci\_refd\_math +  
 sub\_frnch\_refd\_math +  
 sub\_lang\_refd\_math +  
 sub\_art\_refd\_math +  
 sub\_span\_refd\_math +  
 sub\_pe\_refd\_math +  
 sub\_latn\_refd\_math +   
 yr\_2013\_refd\_2012 +  
 yr\_2014\_refd\_2012 +  
 yr\_2015\_refd\_2012 +  
 yr\_2016\_refd\_2012 +   
 loc\_ny\_refd\_ca,   
 data = my\_dat   
 )  
mod\_exam\_mean <- lm( exam\_mean ~ lvl\_mid\_refd\_elem +   
 lvl\_high\_refd\_elem +  
 sub\_sci\_refd\_math +  
 sub\_frnch\_refd\_math +  
 sub\_lang\_refd\_math +  
 sub\_art\_refd\_math +  
 sub\_span\_refd\_math +  
 sub\_pe\_refd\_math +  
 sub\_latn\_refd\_math +   
 yr\_2013\_refd\_2012 +  
 yr\_2014\_refd\_2012 +  
 yr\_2015\_refd\_2012 +  
 yr\_2016\_refd\_2012 +   
 loc\_ny\_refd\_ca,   
 data = my\_dat   
 )  
#----  
# Brady way of Plotting outliers because I'm a bit extra  
  
# Interp Skills per Category  
# Since we're using standardized data units will be in Standard Deviations  
interp\_skls\_sd\_fig = ggplot( mod\_interp,  
 aes(.fitted , .resid )  
 )  
  
interp\_skls\_sd\_fig +  
 geom\_point(col = font\_color) +  
 geom\_hline(yintercept=0, col="green3", linetype="dashed") +  
 xlab(expression( "Fitted Interpersonal Skill Levels - Standardized (Multiples of " \* sigma \* ")" ) ) +  
 ylab(expression( "Residuals - Standardized (Multiples of " \* sigma \* ")" ) ) +  
 ggtitle("Interpersonal Skills Residual vs. Fitted Plot") +  
 my\_gg\_theme



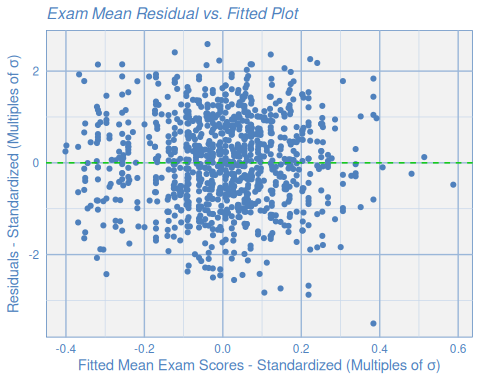
# Exam 1 Skills per Category  
# Since we're using standardized data units will be in Standard Deviations  
exam1\_sd\_fig = ggplot( mod\_exam1,  
 aes(.fitted , .resid )  
 )  
  
exam1\_sd\_fig +  
 geom\_point(col = font\_color) +  
 geom\_hline(yintercept=0, col="green3", linetype="dashed") +  
 xlab(expression( "Fitted Exam 1 Scores - Standardized (Multiples of " \* sigma \* ")" ) ) +  
 ylab(expression( "Residuals - Standardized (Multiples of " \* sigma \* ")" ) ) +  
 ggtitle("Exam 1 Residual vs. Fitted Plot") +  
 my\_gg\_theme



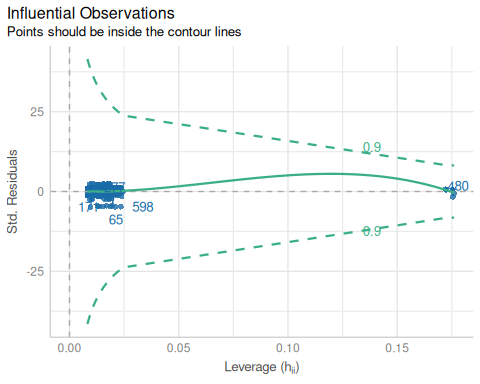
# Exam 2 Skills per Category  
# Since we're using standardized data units will be in Standard Deviations  
exam2\_sd\_fig = ggplot( mod\_exam2,  
 aes(.fitted , .resid )  
 )  
  
exam2\_sd\_fig +  
 geom\_point(col = font\_color) +  
 geom\_hline(yintercept=0, col="green3", linetype="dashed") +  
 xlab(expression( "Fitted Exam 2 Scores - Standardized (Multiples of " \* sigma \* ")" ) ) +  
 ylab(expression( "Residuals - Standardized (Multiples of " \* sigma \* ")" ) ) +  
 ggtitle("Exam 2 Residual vs. Fitted Plot") +  
 my\_gg\_theme



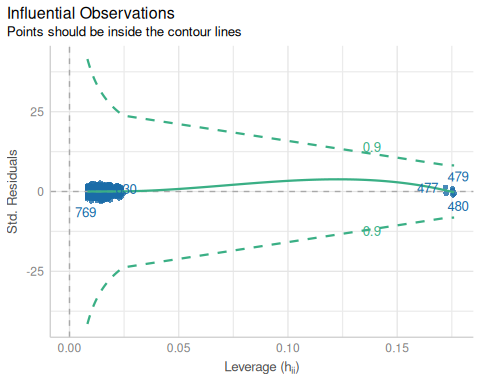
# Exam Mean Skills per Category  
# Since we're using standardized data units will be in Standard Deviations  
exam\_m\_sd\_fig = ggplot( mod\_exam\_mean,  
 aes(.fitted , .resid )  
 )  
  
exam\_m\_sd\_fig +  
 geom\_point(col = font\_color) +  
 geom\_hline(yintercept=0, col="green3", linetype="dashed") +  
 xlab(expression( "Fitted Mean Exam Scores - Standardized (Multiples of " \* sigma \* ")" ) ) +  
 ylab(expression( "Residuals - Standardized (Multiples of " \* sigma \* ")" ) ) +  
 ggtitle("Exam Mean Residual vs. Fitted Plot") +  
 my\_gg\_theme



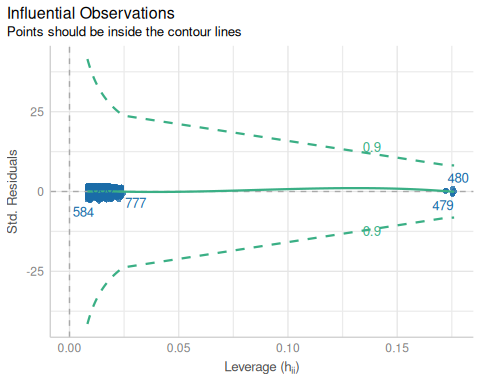
# ----  
# Dr. Diaz method of plotting outliers:  
  
# Outlier plots for all 4x vars  
interp\_outliers <- check\_outliers(mod\_interp)  
plot(interp\_outliers, type = "dots")



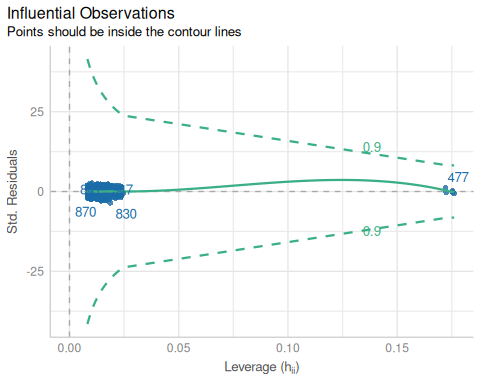
exam1\_outliers <- check\_outliers(mod\_exam1)  
plot(exam1\_outliers, type = "dots")



exam2\_outliers <- check\_outliers(mod\_exam2)  
plot(exam2\_outliers, type = "dots")



exam\_mean\_outliers <- check\_outliers(mod\_exam\_mean)  
plot(exam\_mean\_outliers, type = "dots")



#----  
#identify multivariate outliers (alpha = 0.001)  
# library(performance)  
  
# BJ\_Note:Looks above 99% malaanobis distance (1 - .01). Can change to 95% w/ (1 - .05), etc.  
# Increased percentile (e.g. 99%) finds fewer outliers. Decreased percentile (e.g. 95%) finds more  
# We look for outliers in all 3 unadjusted numeric variables space (exam 1, exam 2, and interp). We don't include  
# exam\_mean as that's a composite of 1 and 2.  
# Used 95% threshold  
cont\_names <- c("interp\_skls", "exam1", "exam2")  
out\_multi.05 <- check\_outliers( my\_dat[cont\_names],   
 method = "mahalanobis",   
 threshold = stats::qchisq( p = 1 - 0.05, df = ncol( my\_dat[cont\_names] ) )  
   
   
 )  
out\_multi.05

## 36 outliers detected: cases 65, 81, 171, 193, 251, 277, 308, 321, 384,  
## 407, 437, 445, 492, 584, 598, 659, 673, 704, 735, 753, 769, 808, 827,  
## 830, 839, 841, 846, 859, 866, 870, 883, 901, 916, 942, 947, 958.  
## - Based on the following method and threshold: mahalanobis (8).  
## - For variables: interp\_skls, exam1, exam2.

# Unstandardized outliers, just to see if its different at all  
out\_multi.05\_unst <- check\_outliers( my\_dat\_unstd[cont\_names],   
 method = "mahalanobis",   
 threshold = stats::qchisq( p = 1 - 0.05, df = ncol( my\_dat\_unstd[cont\_names] ) )  
   
   
 )  
out\_multi.05\_unst

## 36 outliers detected: cases 65, 81, 171, 193, 251, 277, 308, 321, 384,  
## 407, 437, 445, 492, 584, 598, 659, 673, 704, 735, 753, 769, 808, 827,  
## 830, 839, 841, 846, 859, 866, 870, 883, 901, 916, 942, 947, 958.  
## - Based on the following method and threshold: mahalanobis (8).  
## - For variables: interp\_skls, exam1, exam2.

# remove outliers  
my\_dat\_with\_outliers <- my\_dat  
my\_clean\_dat <- my\_dat[!out\_multi.05,]  
  
# Remove outliers from unstandardized data too  
my\_dat\_unstd\_with\_outliers <- my\_dat\_unstd  
my\_clean\_dat\_unstd <- my\_dat\_unstd[!out\_multi.05,]  
  
# my\_dat\_unstd\_2 <- my\_dat\_unstd[!out\_multi.05\_unst,]  
  
# Sample size = 938 (removed 36 outliers whose mahalanobis exceeded the 95% percentile)

## 4.11 Check the Alpha for “exam1” and “exam2” Without Outliers

Check the Alpha for “exam1” and “exam2” a second time with the outliers removed to see if they’re any better…

# Using column names slice so I don't have to pay attention to what order columns are in  
cronbachs\_alpha( my\_clean\_dat[c("exam1", "exam2")] )

## [1] 0.04778261

cronbachs\_alpha( my\_clean\_dat\_unstd[c("exam1", "exam2")] )

## [1] 0.0459635

# cronbachs\_alpha( my\_dat\_unstd\_2[c("exam1", "exam2")] )

# 5 Part 2: Queries

## 5.1 What is the average overall grade for each level of school?

# NOTE: I'm assuming I'm supposed to use my cleaned data for these  
  
# For Elem  
elem\_avg\_grade <- mean( my\_clean\_dat\_unstd$exam\_mean[my\_clean\_dat\_unstd$schl\_lvl == "ELEM"] )  
cat( paste("Average Elementary School Grade: ", elem\_avg\_grade, "\n\n", sep="") )

## Average Elementary School Grade: 65.7827988338192

# For Middle  
mid\_avg\_grade <- mean( my\_clean\_dat\_unstd$exam\_mean[my\_clean\_dat\_unstd$schl\_lvl == "MIDD"] )  
cat( paste("Average Middle School Grade: ", mid\_avg\_grade, "\n\n", sep="") )

## Average Middle School Grade: 63.1443661971831

# For High  
high\_avg\_grade <- mean( my\_clean\_dat\_unstd$exam\_mean[my\_clean\_dat\_unstd$schl\_lvl == "HIGH"] )  
cat( paste("Average High School Grade: ", high\_avg\_grade, "\n\n", sep="") )

## Average High School Grade: 64.8885209713024

## 5.2 What is the average exam 2 grade for math classes?

# NOTE: I'm assuming I'm supposed to use my cleaned data for these  
  
math\_avg\_exam2 <- mean( my\_clean\_dat\_unstd$exam2[my\_clean\_dat\_unstd$sub == "Math"] )  
cat( paste("Average Math Exam 2 Grade: ", math\_avg\_exam2, "\n\n", sep="") )

## Average Math Exam 2 Grade: 85.7865168539326

## 5.3 Calculate the overall average exam grade for all classes.

# NOTE: I'm assuming I'm supposed to use my cleaned data for these  
  
overall\_avg\_exam <- mean( my\_clean\_dat\_unstd$exam\_mean )  
cat( paste("Average Overall Exam Grade: ", overall\_avg\_exam, "\n\n", sep="") )

## Average Overall Exam Grade: 64.9514925373134

## 5.4 Create a new data frame with only classes from CA.

What is the average exam 1 score?

# NOTE: I'm assuming I'm supposed to use my cleaned data for these  
cali\_clean\_da\_unstd <- my\_clean\_dat\_unstd[my\_clean\_dat\_unstd$loc == "CA", ]  
cali\_avg\_exam1 <- mean( cali\_clean\_da\_unstd$exam1 )  
cat( paste("Average Cali Exam 1: ", cali\_avg\_exam1, "\n\n", sep="") )

## Average Cali Exam 1: 45.6979166666667