Technical Appendix

Esther Kaydanovsky

12/2/2019

```
# load data
setwd("C:/Users/Esther/Desktop/Carnegie/Classes/Applied Linear Models/Homework/HW10")
ratings <- read.csv("ratings.csv")

#############################################################################
# PRELIMINARY DATA CLEANING
#############################################################################

# Remove unwanted columns:
# 1) X (observation number) because this is the same as row number
# 2) first12 because it is out of scope for the analysis
# 3) X1stInstr & X2ndInstr because there are too many missing values

ratings <- ratings[-c(24, 25, 26)]

# Next, each variable that contains missing or invalid entries are adjusted. If
# a variable does not have the correct class, convert the variable to its
# proper form (based on information from variable descriptions).

# Note: If a summary() and/or table() output of a variable is not shown, this
#       means there were no missing and/or invalid entries worth mentioning.

#################
### Classical ###
#################

# summary(ratings$Classical)

  #  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
  # 0.000   4.000   6.000   5.783   8.000  19.000      27

# table(ratings$Classical)

  # 0   1   2   3 3.5   4 4.2 4.6   5   6   7   8   9 9.5  10  19
  # 8 111 231 254   1 239   1   1 297 266 334 305 203   1 240   1

# Classsical can only taken on values from 1-10, so zero and 19 are invalid values.
# In addition, it can only take the form of an integer, so decimals are removed.
# There is a small portion of the data set like this, so the values are removed.

# clean the data set by removing missing or invalid values
ratings[which(ratings$Classical %in% c(0,3.5,4.2,4.6,6.8,9.5,19)),] <- NA
ratings <- ratings[!is.na(ratings$Classical),]

################
### Popular ###
################
```

```r
# summary(ratings$Popular)

  #   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
  #  0.000   4.000   5.000   5.381   7.000  19.000      27

# table(ratings$Popular)

  #  0   1   2   3 3.5   4 4.2 4.6   5   6 6.8   7   8   9  10  19
  # 25 171 197 218   1 282   1   1 371 314   1 350 312 136 112   1

# Popular can only taken on values from 1-10, so zero and 19 are invalid values.
# In addition, it can only take the form of an integer, so decimals are removed.
# There is a small portion of the data set like this, so the values are removed.

# clean the data set by removing missing or invalid values
ratings[which(ratings$Popular %in% c(0,3.5,19)),] <- NA
ratings <- ratings[!is.na(ratings$Popular),]

##################
### Muscician ###
##################

# Selfdeclare is dichotomized so that approximately half of the observations
# fall into a "Musician" or "Not Musician" category. This is necessary for
# later analysis

# create musician variable from Selfdeclare
ratings$Musician <- ifelse(as.numeric(ratings$Selfdeclare) > 2, 1, 0)
ratings$Musician <- as.factor(ratings$Musician)
levels(ratings$Musician) <- c("Yes","No")

##################
### ConsInstr ###
##################

# table(ratings$ConsInstr)

 #   0 0.67    1 1.33 1.67    2 2.33 2.67    3 3.33 3.67    4 4.33    5
 # 252   36  288   36  144   36  144   36  468   36  288   72  288  396

# ConsInstr can only taken on whole number values from 0-5, so decimals
# (40% of the data) are rounded to their neareast whole number.

# fix the values by combing them to their closest whole number
ratings$ConsInstr[which(ratings$ConsInstr %in% c(0.67,1.33))] <- 1
ratings$ConsInstr[which(ratings$ConsInstr %in% c(1.67,2.33))] <- 2
ratings$ConsInstr[which(ratings$ConsInstr %in% c(2.67,3.33))] <- 3
ratings$ConsInstr[which(ratings$ConsInstr %in% c(3.67,4.33))] <- 4
ratings$ConsInstr[which(ratings$ConsInstr == 4.67)] <- 5

##################
### ConsNotes ###
##################
```

```
# summary(ratings$ConsNotes)

  #  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
  # 0.000   0.750   3.000   2.533   5.000   5.000     360

# ConsNotes is missing 14.29% of the data. Median imputation used to replace NA's

# impute missing values for ConsNotes using median
ratings$ConsNotes[is.na(ratings$ConsNotes)] <- median(ratings$ConsNotes, na.rm = T)

#########################
### Instr.minus.Notes ###
#########################

# recompute Instr.minus.Notes using the previously adjusted values
ratings$Instr.minus.Notes <- ratings$ConsInstr - ratings$ConsNotes

##################
### PachListen ###
##################

# summary(ratings$PachListen)

  #    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
  #   0.000   5.000   5.000   4.515   5.000   5.000      72

# PachListen is missing 2.86% of the data. Median imputation used to replace NA's

# impute missing values using median
ratings$PachListen[is.na(ratings$PachListen)] <- median(ratings$PachListen, na.rm = T)

#################
### ClsListen ###
#################

# summary(ratings$ClsListen)

  #   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
  #  0.000   1.000   3.000   2.159   3.000   5.000      36

# ClsListen is missing 1.43% of the data. Median imputation used to replace NA's

# impute missing values using median
ratings$ClsListen[is.na(ratings$ClsListen)] <- median(ratings$ClsListen, na.rm = T)

################
### KnowRob ###
################

# summary(ratings$KnowRob)

  #    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
  #  0.0000  0.0000  0.0000  0.7692  0.0000  5.0000     180
```

```
# KnowRob is missing 7.14% of the data. Median imputation used to replace NA's

# impute missing values
ratings$KnowRob[is.na(ratings$KnowRob)] <- median(ratings$KnowRob, na.rm = T)

#################
### KnowAxis ###
#################

# summary(ratings$KnowAxis)

  #   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
  # 0.0000  0.0000  0.0000  0.9032  0.0000  5.0000     288

# KnowAxis is missing 11.43% of the data. Median imputation used to replace NA's

# impute missing values
ratings$KnowAxis[is.na(ratings$KnowAxis)] <- median(ratings$KnowAxis, na.rm = T)

####################
### X1990s2000s ###
####################

# summary(ratings$X1990s2000s)

 #   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
 #   0.000   3.000   5.000   4.061   5.000   5.000     144

# X1990s2000s is missing 5.71% of the data. Median imputation used to replace NA's

# impute missing values using median
ratings$X1990s2000s[is.na(ratings$X1990s2000s)] <- median(ratings$X1990s2000s, na.rm = T)

####################################
### X1990s2000s.minus.1960s1970s ###
####################################

# summary(ratings$X1990s2000s.minus.1960s1970s)

  #   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
  # -4.000   0.000   2.000   2.015   3.000   5.000     180

# X1990s2000s.minus.1960s1970s is missing 7.14% of the data.
# Median imputation used to replace NA's

# impute missing values using median
ratings$X1990s2000s.minus.1960s1970s[is.na(ratings$X1990s2000s.minus.1960s1970s)] <-
  median(ratings$X1990s2000s.minus.1960s1970s, na.rm = T)

##############################
### Music Class Variables ###
##############################
```

```r
# There are missing values for NoClass, but these will be addressed later
# because there is not a simple median imputation solution.

# summary(ratings$CollegeMusic)

  #   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
  #  0.000   1.000   1.000   0.791   1.000   1.000     108

# CollegeMusic is missing 4.29% of the data. Median imputation used to replace NA's

# impute missing values for CollegeMusic using median
ratings$CollegeMusic[is.na(ratings$CollegeMusic)] <- median(ratings$CollegeMusic, na.rm = T)

# summary(ratings$APTheory)

  #   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
  # 0.0000  0.0000  0.0000  0.2344  0.0000  1.0000     216

# APTheory is missing 8.57% of the data. Median imputation used to replace NA's

# impute missing values for APTheory using median
ratings$APTheory[is.na(ratings$APTheory)] <- median(ratings$APTheory, na.rm = T)

##################
### Composing ###
##################

# summary(ratings$Composing)

  #   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
  #     0       0       0       1       2       5      72

# Composing is missing 2.86% of the data. Median imputation used to replace NA's

# impute missing values using median
ratings$Composing[is.na(ratings$Composing)] <- median(as.numeric(ratings$Composing), na.rm = T)

###############
### FACTORS ###
###############

# 'data.frame': 2453 obs. of  24 variables:
#  $ Subject                    : Factor w/ 70 levels "15","16","17",..: 1 1 1 1 1 1 1 1 1 1 ...
#  $ Harmony                    : Factor w/ 4 levels "I-IV-V","I-V-IV",..: 1 1 1 1 1 1 1 1 1 2 ...
#  $ Instrument                 : Factor w/ 3 levels "guitar","piano",..: 1 1 1 2 2 2 3 3 3 1 ...
#  $ Voice                      : Factor w/ 3 levels "contrary","par3rd",..: 1 2 3 1 2 3 1 2 3 1 ...
#  $ Selfdeclare                : int  5 5 5 5 5 5 5 5 5 5 ...
#  $ OMSI                       : int  734 734 734 734 734 734 734 734 734 734 ...
#  $ X16.minus.17               : num  5 5 5 5 5 5 5 5 5 5 ...
#  $ ConsInstr                  : num  4 4 4 4 4 4 4 4 4 4 ...
#  $ ConsNotes                  : int  5 5 5 5 5 5 5 5 5 5 ...
#  $ Instr.minus.Notes          : num  -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 ...
#  $ PachListen                 : int  5 5 5 5 5 5 5 5 5 5 ...
```

```
#  $ ClsListen                   : int  4 4 4 4 4 4 4 4 4 4 ...
#  $ KnowRob                     : num  0 0 0 0 0 0 0 0 0 0 ...
#  $ KnowAxis                    : num  0 0 0 0 0 0 0 0 0 0 ...
#  $ X1990s2000s                 : int  5 5 5 5 5 5 5 5 5 5 ...
#  $ X1990s2000s.minus.1960s1970s: int  2 2 2 2 2 2 2 2 2 2 ...
#  $ CollegeMusic                : num  0 0 0 0 0 0 0 0 0 0 ...
#  $ NoClass                     : int  0 0 0 0 0 0 0 0 0 0 ...
#  $ APTheory                    : num  0 0 0 0 0 0 0 0 0 0 ...
#  $ Composing                   : num  4 4 4 4 4 4 4 4 4 4 ...
#  $ PianoPlay                   : int  1 1 1 1 1 1 1 1 1 1 ...
#  $ GuitarPlay                  : int  5 5 5 5 5 5 5 5 5 5 ...
#  $ Classical                   : num  3 3 1 3 2 8 10 6 5 1 ...
#  $ Popular                     : num  9 7 8 7 8 3 1 4 5 8 ...


# There are a lot of variables encoded as numeric or integers that need to be
# encoded as factors, so convert them to the intended class:

ratings$Selfdeclare <- as.factor(ratings$Selfdeclare)
ratings$ConsInstr <- as.factor(ratings$ConsInstr)
ratings$ConsNotes <- as.factor(ratings$ConsNotes)
ratings$Instr.minus.Notes <- as.factor(ratings$Instr.minus.Notes)
ratings$PachListen <- as.factor(ratings$PachListen)
ratings$ClsListen <- as.factor(ratings$ClsListen)
ratings$KnowRob <- as.factor(ratings$KnowRob)
ratings$KnowAxis <- as.factor(ratings$KnowAxis)
ratings$X1990s2000s <- as.factor(ratings$X1990s2000s)
ratings$X1990s2000s.minus.1960s1970s <- as.factor(ratings$X1990s2000s.minus.1960s1970s)
ratings$APTheory <- as.factor(ratings$APTheory)
ratings$CollegeMusic <- as.factor(ratings$CollegeMusic)
ratings$Composing <- as.factor(ratings$Composing)
ratings$PianoPlay <- as.factor(ratings$PianoPlay)
ratings$GuitarPlay <- as.factor(ratings$GuitarPlay)

#############################################################################
# DATA CLEANING FROM EDA
#############################################################################

# load packages for visualization
library(ggplot2)
library(gridExtra)

#################
### sqrt_OMSI ###
#################

# The histogram of OMSI is right skewed, so to make the data more normal, a
# square-root transformation is applied

par(mfrow = c(2,2))

# skewed
hist(ratings$OMSI, main = "Histogram of OMSI", xlab = "OMSI")
```

```r
# better (and no infinity problems)
hist((ratings$OMSI)^(1/2), main = "Histogram of OMSI^(1/2)", xlab = "OMSI")

# create new variable
ratings$sqrt_OMSI <- (ratings$OMSI)^(1/2)

##############################
### NoClass -> MusicClass ###
##############################

ggplot(ratings) + geom_bar(aes(as.factor(NoClass)), fill = "violetred4") + xlab("NoClass")

# It is seen from the barchart that the majority of people have either taken a
# music class or have not taken a music class. There are very few observations
# that fall into categories higher than 1. For this reason, a new variable,
# MusicClass, is made from dichotomizing NoClass so that 0 = no music class
# taken and 1 = at least one music class.

# transform NoClass variable to be binary
ratings$MusicClass <- ratings$NoClass
ratings$MusicClass[which(ratings$MusicClass > 1)] <- 1
ratings$MusicClass <- as.factor(ratings$MusicClass)
levels(ratings$MusicClass) <- c(0,1)

plot1 <- ggplot(ratings) + geom_bar(aes(MusicClass, fill = CollegeMusic)) +
  ggtitle("MusicClass by CollegeMusic")
plot2 <- ggplot(ratings) + geom_bar(aes(MusicClass, fill = APTheory)) +
  ggtitle("MusicClass by CollegeMusic")

# Additionally, there are some problems with the data where observations encoded
# as 1 for CollegeMusic and/or APTheory (meaning subject has taken a music class
# in college or AP Music theory in high school) are also encoded as 0 for NoClass
# (meaning they have never taken a music class). If the subject indicated that
# they took a music class in high school or college, then the values of 0 were
# changed to 1 for MusicClass.

ratings$MusicClass[which(ratings$CollegeMusic == 1 & ratings$MusicClass == 0)] <- 1
ratings$MusicClass[which(ratings$APTheory == 1 & ratings$MusicClass == 0)] <- 1

plot3 <- ggplot(ratings) + geom_bar(aes(MusicClass, fill = CollegeMusic)) +
  ggtitle("MusicClass by CollegeMusic")
plot4 <- ggplot(ratings) + geom_bar(aes(MusicClass, fill = APTheory)) +
  ggtitle("MusicClass by APTheory")

# Finally, there are missing values for MusicClass that have CollegeMusic
# encoded as 1. Since MusicClass is binary, these missing values can be
# filled because we know that the subject has taken at least one music class

grid.arrange(plot1, plot2, plot3, plot4, ncol = 2)

ratings$MusicClass[which(ratings$CollegeMusic == 1 & is.na(ratings$MusicClass))] <- 1
ratings$MusicClass[which(ratings$APTheory == 1 & is.na(ratings$MusicClass))] <- 1
```

```r
# summary(ratings$MusicClass)

#   0    1 NA's
# 323 2094 36

# For all other missing data (1.43% of the data), impute using the mode:

# function used to calculate mode
# source: https://www.tutorialspoint.com/r/r_mean_median_mode.htm
mode_calc <- function(x) {
   levels <- unique(x)
   levels[which.max(tabulate(match(x, levels)))]
}

ratings$MusicClass[is.na(ratings$MusicClass)] <- mode_calc(ratings$MusicClass)

######################
### Other Factors ###
######################

# Some of the ordinal variables are hard to think about because there
# are six levels. It would be easier to understand them if they were
# transformed to have less levels.

# To do this, we look at the break down of the variables of interest using barplots
# (variables of concern: PachList, ClsListen, KnowRob, KnowAxis, Composing,
# GuitarPlay, and PianoPlay).

plot1 <- ggplot(ratings) + geom_bar(aes(PachListen), fill = "violetred4")
plot2 <- ggplot(ratings) + geom_bar(aes(ClsListen), fill = "violetred4")
plot3 <- ggplot(ratings) + geom_bar(aes(KnowRob), fill = "violetred4")
plot4 <- ggplot(ratings) + geom_bar(aes(KnowAxis), fill = "violetred4")
plot5 <- ggplot(ratings) + geom_bar(aes(Composing), fill = "violetred4")
plot6 <- ggplot(ratings) + geom_bar(aes(PianoPlay), fill = "violetred4")
plot7 <- ggplot(ratings) + geom_bar(aes(GuitarPlay), fill = "violetred4")
grid.arrange(plot1, plot2, plot3, plot4, plot5, plot6, plot7, ncol = 4)

# The graphs confrim that certain variable's factor levels can be simplified:

# first we convert the variables to character for easy factor level conversion
ratings$KnowAxis <- as.character(ratings$KnowAxis)
ratings$KnowRob <- as.character(ratings$KnowRob)
ratings$PachListen <- as.character(ratings$PachListen)
ratings$ClsListen <- as.character(ratings$ClsListen)
ratings$Composing <- as.character(ratings$Composing)
ratings$PianoPlay <- as.character(ratings$PianoPlay)
ratings$GuitarPlay <- as.character(ratings$GuitarPlay)

# convert KnowAxis and KnowRob to a binary variable
ratings$KnowAxis[which(ratings$KnowAxis == 5)] <- 1
ratings$KnowAxis <- as.factor(ratings$KnowAxis)

ratings$KnowRob[which(ratings$KnowRob %in% c(1,5))] <- 1
```

```r
ratings$KnowRob <- as.factor(ratings$KnowRob)

# convert the level of ClsListen, PachListen, Composing, PianoPlay, and GuitarPlay
# so that there are three levels (0,1,2) where generally:
# 0 represents not at all
# 1 represents a little
# 2 represents a lot

ratings$ClsListen[which(ratings$ClsListen %in% c(3,4,5))] <- 2
ratings$ClsListen <- as.factor(ratings$ClsListen)

ratings$PachListen[which(ratings$PachListen %in% c(0,1))] <- 0
ratings$PachListen[which(ratings$PachListen %in% c(3,2))] <- 1
ratings$PachListen[which(ratings$PachListen %in% c(4,5))] <- 2
ratings$PachListen <- as.factor(ratings$PachListen)

ratings$Composing[which(ratings$Composing %in% c(1,2))] <- 1
ratings$Composing[which(ratings$Composing %in% c(3,4,5))] <- 2
ratings$Composing <- as.factor(ratings$Composing)

ratings$PianoPlay[which(ratings$PianoPlay %in% c(1,2))] <- 1
ratings$PianoPlay[which(ratings$PianoPlay %in% c(4,5))] <- 2
ratings$PianoPlay <- as.factor(ratings$PianoPlay)

ratings$GuitarPlay[which(ratings$GuitarPlay %in% c(1,2))] <- 1
ratings$GuitarPlay[which(ratings$GuitarPlay %in% c(4,5))] <- 2
ratings$GuitarPlay <- as.factor(ratings$GuitarPlay)

################################################################################
# EDA
################################################################################

#################
### Classical ###
#################

plot1 <- ggplot(ratings, aes(x = Harmony, y = Classical)) +
  geom_boxplot(fill = "powderblue")
plot2 <- ggplot(ratings, aes(x = Instrument, y = Classical)) +
  geom_boxplot(fill = "powderblue")
plot3 <- ggplot(ratings, aes(x = Voice, y = Classical)) +
  geom_boxplot(fill = "powderblue")
grid.arrange(plot1, plot2, plot3, ncol = 2)

cor(ratings$X16.minus.17, ratings$Classical)
cor(ratings$sqrt_OMSI, ratings$Classical)

plot1 <- ggplot(ratings, aes(x = ConsInstr, y = Classical)) +
  geom_boxplot(fill = "powderblue")
plot2 <- ggplot(ratings, aes(x = PachListen, y = Classical)) +
  geom_boxplot(fill = "powderblue")
plot3 <- ggplot(ratings, aes(x = X1990s2000s, y = Classical)) +
  geom_boxplot(fill = "powderblue")
```

```r
plot4 <- ggplot(ratings, aes(x = MusicClass, y = Classical)) +
  geom_boxplot(fill = "powderblue")
plot5 <- ggplot(ratings, aes(x = Composing, y = Classical)) +
  geom_boxplot(fill = "powderblue")
plot6 <- ggplot(ratings, aes(x = GuitarPlay, y = Classical)) +
  geom_boxplot(fill = "powderblue")
grid.arrange(plot1, plot2, plot3, plot4, plot5, plot6, ncol = 2)

###############
### Popular ###
###############

plot1 <- ggplot(ratings, aes(x = Harmony, y = Popular)) +
  geom_boxplot(fill = "powderblue")
plot2 <- ggplot(ratings, aes(x = Instrument, y = Popular)) +
  geom_boxplot(fill = "powderblue")
plot3 <- ggplot(ratings, aes(x = Voice, y = Popular)) +
  geom_boxplot(fill = "powderblue")
grid.arrange(plot1, plot2, plot3, ncol = 2)

cor(ratings$X16.minus.17, ratings$Popular)
cor(ratings$sqrt_OMSI, ratings$Popular)

plot1 <- ggplot(ratings, aes(x = ConsInstr, y = Popular)) +
  geom_boxplot(fill = "powderblue")
plot2 <- ggplot(ratings, aes(x = ClsListen, y = Popular)) +
  geom_boxplot(fill = "powderblue")
plot3 <- ggplot(ratings, aes(x = KnowRob, y = Popular)) +
  geom_boxplot(fill = "powderblue")
plot4 <- ggplot(ratings, aes(x = KnowAxis, y = Popular)) +
  geom_boxplot(fill = "powderblue")
plot5 <- ggplot(ratings, aes(x = X1990s2000s, y = Popular)) +
  geom_boxplot(fill = "powderblue")
plot6 <- ggplot(ratings, aes(x = APTheory, y = Popular)) +
  geom_boxplot(fill = "powderblue")
plot7 <- ggplot(ratings, aes(x = Composing, y = Popular)) +
  geom_boxplot(fill = "powderblue")
plot8 <- ggplot(ratings, aes(x = PianoPlay, y = Popular)) +
  geom_boxplot(fill = "powderblue")
plot9 <- ggplot(ratings, aes(x = GuitarPlay, y = Popular)) +
  geom_boxplot(fill = "powderblue")
grid.arrange(plot1, plot2, plot3, plot4, plot5, plot6,
             plot7, plot8, plot9, ncol = 3)


##############################################################################
# MODELING
##############################################################################

#############################################
#### Is a random effects model necessary? ###
#############################################
```

```r
# load library to use lmer() to fit model with random effects
library(lme4)

# Since the observations for the variables are subjective depending on the
# subject that answered them, a random effects model might be necessary to
# capture the variation in the three main factors. We use anova() to do this.

lm.1 <- lm(Classical ~ Instrument + Harmony + Voice, data = ratings)
lm.2 <- lmer(Classical ~ 1 + Instrument + Harmony + Voice + (1|Subject),
             data = ratings, REML = F)
compare <- as.data.frame(anova(lm.2, lm.1))
compare$Significant <- ifelse(compare$`Pr(>Chisq)` > 0.05, "No", "Yes")
knitr::kable(compare, caption = "ANOVA to Test Whether Random Effect Is Needed for Classical Ratings")

lm.1 <- lm(Popular ~ Instrument + Harmony + Voice, data = ratings)
lm.2 <- lmer(Popular ~ 1 + Instrument + Harmony + Voice + (1|Subject),
             data = ratings, REML = F)
compare <- as.data.frame(anova(lm.2, lm.1))
compare$Significant <- ifelse(compare$`Pr(>Chisq)` > 0.05, "No", "Yes")
knitr::kable(compare, caption = "ANOVA to Test Whether Random Effect Is Needed for Popular Ratings")

# The random effects model is necessary for both Classical and Popular Ratings.
# The anova() output indicates that The coefficient of the repeated measures model,
# found in the ANOVA, is significally different from the simpler model without the
# random effect. In addition, the AIC and BIC of the random effects model is lower
# than the simplier model. Therefore, the random intercept model is needed.

##########################################
#### Which Random Effect is Necessary? ###
##########################################

#################
### Classical ###
#################

# test all possible combination of random effects (just 1, 2, or all of them)
lmer.1 <- lmer(Classical ~ Instrument + Harmony + Voice + (Instrument|Subject),
               data = ratings, control = lmerControl(optimizer = 'bobyqa'), REML = F)
lmer.2 <- lmer(Classical ~ Instrument + Harmony + Voice + (Harmony|Subject),
               data = ratings, control = lmerControl(optimizer = 'bobyqa'), REML = F)
lmer.3 <- lmer(Classical ~ Instrument + Harmony + Voice + (Voice|Subject),
               data = ratings, control = lmerControl(optimizer = 'bobyqa'), REML = F)
lmer.4 <- lmer(Classical ~ Instrument + Harmony + Voice + (Instrument + Harmony|Subject),
               data = ratings, control = lmerControl(optimizer = 'bobyqa'), REML = F)
lmer.5 <- lmer(Classical ~ Instrument + Harmony + Voice + (Instrument + Voice|Subject),
               data = ratings, control = lmerControl(optimizer = 'bobyqa'), REML = F)
lmer.6 <- lmer(Classical ~ Instrument + Harmony + Voice + (Harmony + Voice|Subject),
               data = ratings, control = lmerControl(optimizer = 'bobyqa'), REML = F)
lmer.7 <- lmer(Classical ~ Instrument + Harmony + Voice +
                 (Instrument + Harmony + Voice|Subject),
               data = ratings, control = lmerControl(optimizer = 'bobyqa'), REML = F)

# use anova() to see which model was the best out of the seven
```

```
compare <- as.data.frame(anova(lm.2, lmer.1, lmer.2, lmer.3, lmer.4,
                               lmer.5, lmer.6, lmer.7))
compare$Significant <- ifelse(compare$`Pr(>Chisq)` > 0.05, "No", "Yes")
knitr::kable(compare, caption = "ANOVA to Test Different Random Effects for Classical Ratings")

###############
### Popular ###
###############

# test all possible combination of random effects (just 1, 2, or all of them)
lmer.1 <- lmer(Popular ~ Instrument + Harmony + Voice + (Instrument|Subject),
               data = ratings, control = lmerControl(optimizer = 'bobyqa'), REML = F)
lmer.2 <- lmer(Popular ~ Instrument + Harmony + Voice + (Harmony|Subject),
               data = ratings, control = lmerControl(optimizer = 'bobyqa'), REML = F)
lmer.3 <- lmer(Popular ~ Instrument + Harmony + Voice + (Voice|Subject),
               data = ratings, control = lmerControl(optimizer = 'bobyqa'), REML = F)
lmer.4 <- lmer(Popular ~ Instrument + Harmony + Voice + (Instrument + Harmony|Subject),
               data = ratings, control = lmerControl(optimizer = 'bobyqa'), REML = F)
lmer.5 <- lmer(Popular ~ Instrument + Harmony + Voice + (Instrument + Voice|Subject),
               data = ratings, control = lmerControl(optimizer = 'bobyqa'), REML = F)
lmer.6 <- lmer(Popular ~ Instrument + Harmony + Voice + (Harmony + Voice|Subject),
               data = ratings, control = lmerControl(optimizer = 'bobyqa'), REML = F)
lmer.7 <- lmer(Popular ~ Instrument + Harmony + Voice +
                 (Instrument + Harmony + Voice|Subject),
               data = ratings, control = lmerControl(optimizer = 'bobyqa'), REML = F)

# use anova() to see which model was the best out of the seven
compare <- as.data.frame(anova(lm.2, lmer.1, lmer.2, lmer.3, lmer.4,
                               lmer.5, lmer.6, lmer.7))
compare$Significant <- ifelse(compare$`Pr(>Chisq)` > 0.05, "No", "Yes")
knitr::kable(compare, caption = "ANOVA to Test Different Random Effects for Popular Ratings")

# The anova() shows that lmer.4 (random effects for instrument and harmoney) was
# significantly different from the random intercept model for both classical and
# popular ratings and had the lowest AIC and BIC compared to the other models.

########################################
### Backward Fitting of Fixed Effects ###
########################################

# load library for automatic variable selection
library(LMERConvenienceFunctions)

#################
### Classical ###
#################

# automatic method 1: fitLMER
model_cls1 <- lmer(Classical ~ Instrument + Harmony + Voice + Selfdeclare +
                     sqrt_OMSI + X16.minus.17 + ConsInstr + ConsNotes +
                     Instr.minus.Notes + PachListen + ClsListen + KnowAxis +
                     KnowRob + X1990s2000s + X1990s2000s.minus.1960s1970s +
                     CollegeMusic + MusicClass + APTheory + Composing +
```

```
                    PianoPlay + GuitarPlay + (Instrument|Subject) +
                    (0 + Harmony|Subject), data = ratings,
                  control = lmerControl(optimizer = 'bobyqa'), REML = F)

BIC_fitLMER_cls1 <- fitLMER.fnc(model_cls1, method = "BIC")
AIC_fitLMER_cls1 <- fitLMER.fnc(model_cls1, method = "AIC")

summary(BIC_fitLMER_cls1)
summary(AIC_fitLMER_cls1)

# thre results of the fitLMER automatic method do not yield any useful results,
# instead another "automatic method" is done: manual forward selection

# automatic method 2: manual forward selection
lm.base <- lmer(Classical ~ Instrument + Harmony + Voice + (Instrument + Harmony|Subject),
                data = ratings, control = lmerControl(optimizer = 'bobyqa'), REML = F)
lm.1 <- lmer(Classical ~ Instrument + Harmony + Voice + (Instrument + Harmony|Subject) +
               Selfdeclare, data = ratings, control = lmerControl(optimizer = 'bobyqa'), REML = F)
lm.2 <- lmer(Classical ~ Instrument + Harmony + Voice + (Instrument + Harmony|Subject) +
               sqrt_OMSI, data = ratings, control = lmerControl(optimizer = 'bobyqa'), REML = F)
lm.3 <- lmer(Classical ~ Instrument + Harmony + Voice + (Instrument + Harmony|Subject) +
               X16.minus.17, data = ratings, control = lmerControl(optimizer = 'bobyqa'), REML = F)
lm.4 <- lmer(Classical ~ Instrument + Harmony + Voice + (Instrument + Harmony|Subject) +
               ConsInstr, data = ratings, control = lmerControl(optimizer = 'bobyqa'), REML = F)
lm.5 <- lmer(Classical ~ Instrument + Harmony + Voice + (Instrument + Harmony|Subject) +
               ConsNotes, data = ratings, control = lmerControl(optimizer = 'bobyqa'), REML = F)
lm.6 <- lmer(Classical ~ Instrument + Harmony + Voice + (Instrument + Harmony|Subject) +
               Instr.minus.Notes, data = ratings, control = lmerControl(optimizer = 'bobyqa'), REML = F
lm.7 <- lmer(Classical ~ Instrument + Harmony + Voice + (Instrument + Harmony|Subject) +
               PachListen, data = ratings, control = lmerControl(optimizer = 'bobyqa'), REML = F)
lm.8 <- lmer(Classical ~ Instrument + Harmony + Voice + (Instrument + Harmony|Subject) +
               ClsListen, data = ratings, control = lmerControl(optimizer = 'bobyqa'), REML = F)
lm.9 <- lmer(Classical ~ Instrument + Harmony + Voice + (Instrument + Harmony|Subject) +
               KnowRob, data = ratings, control = lmerControl(optimizer = 'bobyqa'), REML = F)
lm.10 <- lmer(Classical ~ Instrument + Harmony + Voice + (Instrument + Harmony|Subject) +
                KnowAxis, data = ratings, control = lmerControl(optimizer = 'bobyqa'), REML = F)
lm.11 <- lmer(Classical ~ Instrument + Harmony + Voice + (Instrument + Harmony|Subject) +
                X1990s2000s, data = ratings, control = lmerControl(optimizer = 'bobyqa'), REML = F)
lm.12 <- lmer(Classical ~ Instrument + Harmony + Voice + (Instrument + Harmony|Subject) +
                X1990s2000s.minus.1960s1970s, data = ratings, control = lmerControl(optimizer = 'bobyqa
lm.13 <- lmer(Classical ~ Instrument + Harmony + Voice + (Instrument + Harmony|Subject) +
                CollegeMusic, data = ratings, control = lmerControl(optimizer = 'bobyqa'), REML = F)
lm.14 <- lmer(Classical ~ Instrument + Harmony + Voice + (Instrument + Harmony|Subject) +
                MusicClass, data = ratings, control = lmerControl(optimizer = 'bobyqa'), REML = F)
lm.15 <- lmer(Classical ~ Instrument + Harmony + Voice + (Instrument + Harmony|Subject) +
                APTheory, data = ratings, control = lmerControl(optimizer = 'bobyqa'), REML = F)
lm.16 <- lmer(Classical ~ Instrument + Harmony + Voice + (Instrument + Harmony|Subject) +
                Composing, data = ratings, control = lmerControl(optimizer = 'bobyqa'), REML = F)
lm.17 <- lmer(Classical ~ Instrument + Harmony + Voice + (Instrument + Harmony|Subject) +
                PianoPlay, data = ratings, control = lmerControl(optimizer = 'bobyqa'), REML = F)
lm.18 <- lmer(Classical ~ Instrument + Harmony + Voice + (Instrument + Harmony|Subject) +
                GuitarPlay, data = ratings, control = lmerControl(optimizer = 'bobyqa'), REML = F)
compare <- as.data.frame(anova(lm.base, lm.1, lm.2, lm.3, lm.4, lm.5, lm.6,
```

```
                                lm.7, lm.8, lm.9, lm.10, lm.11, lm.12, lm.13,
                                lm.14, lm.15, lm.16, lm.17, lm.18))
compare$Significant <- ifelse(compare$`Pr(>Chisq)` > 0.05, "No", "Yes")
knitr::kable(compare, caption = "1 ANOVA to Test Which Fixed Effects are Needed")

# the forward selection shows that X16.minus.17 are the most useful fixed effects
# in the model to reduce AIC and BIC

# these fixed effects are added to the model and compared to two other models
# base on intuition and the EDA done earlier:

# forward selection
cls1 <- lmer(Classical ~ Instrument + Harmony + Voice + X16.minus.17 +
               ClsListen + MusicClass + PianoPlay + (Instrument + Harmony|Subject),
             data = ratings,
             control = lmerControl(optimizer = 'bobyqa'), REML = F)

# intuition
cls2 <- lmer(Classical ~ Instrument + Harmony + Voice + ConsInstr +
               ClsListen + X1990s2000s + MusicClass + Composing +
                    PianoPlay + GuitarPlay + (Instrument + Harmony|Subject),
             data = ratings,
             control = lmerControl(optimizer = 'bobyqa'), REML = F)
# EDA
cls3 <- lmer(Classical ~ Instrument + Harmony + Voice + PachListen + ConsInstr +
               X1990s2000s + MusicClass + Composing + GuitarPlay + (Instrument + Harmony|Subject),
             data = ratings,
             control = lmerControl(optimizer = 'bobyqa'), REML = F)

compare <- as.data.frame(anova(lm.base, cls1, cls2, cls3))
compare$Significant <- ifelse(compare$`Pr(>Chisq)` > 0.05, "No", "Yes")
knitr::kable(compare, caption = "1 ANOVA to Test Which Fixed Effects are Needed")

# at the end, the model created from the forward selection was the best in
# reducing AIC and BIC

###############
### Popular ###
###############

# automatic method 1: fitLMER
model_pop <- lmer(Popular ~ Instrument + Harmony + Voice + Selfdeclare +
                    sqrt_OMSI + X16.minus.17 + ConsInstr + ConsNotes +
                    Instr.minus.Notes + PachListen + ClsListen + KnowAxis +
                    KnowRob + X1990s2000s + X1990s2000s.minus.1960s1970s +
                    CollegeMusic + MusicClass + APTheory + Composing +
                    PianoPlay + GuitarPlay + (Instrument|Subject) +
                    (0 + Harmony|Subject), data = ratings,
                  control = lmerControl(optimizer = 'bobyqa'), REML = F)

BIC_fitLMER_pop <- fitLMER.fnc(model_pop, method = "BIC")
AIC_fitLMER_pop <- fitLMER.fnc(model_pop, method = "AIC")
```

```r
summary(BIC_fitLMER_pop)
summary(AIC_fitLMER_pop)

# thre results of the fitLMER automatic method do not yield any useful results,
# instead another "automatic method" is done: manual forward selection

# automatic method 2: manual forward selection
lm.base <- lmer(Popular ~ Instrument + Harmony + Voice + (Instrument + Harmony|Subject),
                data = ratings, control = lmerControl(optimizer = 'bobyqa'), REML = F)
lm.1 <- lmer(Popular ~ Instrument + Harmony + Voice + (Instrument + Harmony|Subject) +
               Selfdeclare, data = ratings, control = lmerControl(optimizer = 'bobyqa'), REML = F)
lm.2 <- lmer(Popular ~ Instrument + Harmony + Voice + (Instrument + Harmony|Subject) +
               sqrt_OMSI, data = ratings, control = lmerControl(optimizer = 'bobyqa'), REML = F)
lm.3 <- lmer(Popular ~ Instrument + Harmony + Voice + (Instrument + Harmony|Subject) +
               X16.minus.17, data = ratings, control = lmerControl(optimizer = 'bobyqa'), REML = F)
lm.4 <- lmer(Popular ~ Instrument + Harmony + Voice + (Instrument + Harmony|Subject) +
               ConsInstr, data = ratings, control = lmerControl(optimizer = 'bobyqa'), REML = F)
lm.5 <- lmer(Popular ~ Instrument + Harmony + Voice + (Instrument + Harmony|Subject) +
               ConsNotes, data = ratings, control = lmerControl(optimizer = 'bobyqa'), REML = F)
lm.6 <- lmer(Popular ~ Instrument + Harmony + Voice + (Instrument + Harmony|Subject) +
               Instr.minus.Notes, data = ratings, control = lmerControl(optimizer = 'bobyqa'), REML = F
lm.7 <- lmer(Popular ~ Instrument + Harmony + Voice + (Instrument + Harmony|Subject) +
               PachListen, data = ratings, control = lmerControl(optimizer = 'bobyqa'), REML = F)
lm.8 <- lmer(Popular ~ Instrument + Harmony + Voice + (Instrument + Harmony|Subject) +
               ClsListen, data = ratings, control = lmerControl(optimizer = 'bobyqa'), REML = F)
lm.9 <- lmer(Popular ~ Instrument + Harmony + Voice + (Instrument + Harmony|Subject) +
               KnowRob, data = ratings, control = lmerControl(optimizer = 'bobyqa'), REML = F)
lm.10 <- lmer(Popular ~ Instrument + Harmony + Voice + (Instrument + Harmony|Subject) +
                KnowAxis, data = ratings, control = lmerControl(optimizer = 'bobyqa'), REML = F)
lm.11 <- lmer(Popular ~ Instrument + Harmony + Voice + (Instrument + Harmony|Subject) +
                X1990s2000s, data = ratings, control = lmerControl(optimizer = 'bobyqa'), REML = F)
lm.12 <- lmer(Popular ~ Instrument + Harmony + Voice + (Instrument + Harmony|Subject) +
               X1990s2000s.minus.1960s1970s, data = ratings, control = lmerControl(optimizer = 'bobyqa
lm.13 <- lmer(Popular ~ Instrument + Harmony + Voice + (Instrument + Harmony|Subject) +
               CollegeMusic, data = ratings, control = lmerControl(optimizer = 'bobyqa'), REML = F)
lm.14 <- lmer(Popular ~ Instrument + Harmony + Voice + (Instrument + Harmony|Subject) +
               MusicClass, data = ratings, control = lmerControl(optimizer = 'bobyqa'), REML = F)
lm.15 <- lmer(Popular ~ Instrument + Harmony + Voice + (Instrument + Harmony|Subject) +
               APTheory, data = ratings, control = lmerControl(optimizer = 'bobyqa'), REML = F)
lm.16 <- lmer(Popular ~ Instrument + Harmony + Voice + (Instrument + Harmony|Subject) +
               Composing, data = ratings, control = lmerControl(optimizer = 'bobyqa'), REML = F)
lm.17 <- lmer(Popular ~ Instrument + Harmony + Voice + (Instrument + Harmony|Subject) +
               PianoPlay, data = ratings, control = lmerControl(optimizer = 'bobyqa'), REML = F)
lm.18 <- lmer(Popular ~ Instrument + Harmony + Voice + (Instrument + Harmony|Subject) +
               GuitarPlay, data = ratings, control = lmerControl(optimizer = 'bobyqa'), REML = F)
compare <- as.data.frame(anova(lm.base, lm.1, lm.2, lm.3, lm.4, lm.5, lm.6,
                               lm.7, lm.8, lm.9, lm.10, lm.11, lm.12, lm.13,
                               lm.14, lm.15, lm.16, lm.17, lm.18))
compare$Significant <- ifelse(compare$`Pr(>Chisq)` > 0.05, "No", "Yes")
knitr::kable(compare, caption = "1 ANOVA to Test Which Fixed Effects are Needed")

# the forward selection shows that X16.minus.17 are the most useful fixed effects
# in the model to reduce AIC and BIC
```

```r
# these fixed effects are added to the model and compared to two other models
# base on intuition and the EDA done earlier:

# forward selection
pop1 <- lmer(Popular ~ Instrument + Harmony + Voice + X16.minus.17 +
               KnowRob + Composing + Selfdeclare + (Instrument + Harmony|Subject),
             data = ratings,
             control = lmerControl(optimizer = 'bobyqa'), REML = F)

# intuition
pop2 <- lmer(Popular ~ Instrument + Harmony + Voice + GuitarPlay + X1990s2000s
             + ClsListen + ConsInstr + (Instrument + Harmony|Subject),
             data = ratings,
             control = lmerControl(optimizer = 'bobyqa'), REML = F)

# EDA
pop3 <- lmer(Popular ~ Instrument + Harmony + Voice + APTheory +
               KnowRob + KnowAxis + X1990s2000s + Composing + PianoPlay +
               GuitarPlay + ClsListen + ConsInstr +(Instrument + Harmony|Subject),
             data = ratings,
             control = lmerControl(optimizer = 'bobyqa'), REML = F)

compare <- as.data.frame(anova(lm.base, pop1, pop2, pop3))
compare$Significant <- ifelse(compare$`Pr(>Chisq)` > 0.05, "No", "Yes")
knitr::kable(compare, caption = "1 ANOVA to Test Which Fixed Effects are Needed")

# at the end, the model created from the forward selection was the best in
# reducing AIC and BIC

########################################
### Forward Fitting of Random Effects ###
########################################

#################
### Classical ###
#################

# test all possible combination of random effects (just 1, 2, or all of them)
lmer.1 <- lmer(Classical ~ Instrument + Harmony + Voice + (Instrument|Subject) +
                 X16.minus.17 + ClsListen + MusicClass + PianoPlay,
               data = ratings, control = lmerControl(optimizer = 'bobyqa'), REML = F)
lmer.2 <- lmer(Classical ~ Instrument + Harmony + Voice + (Harmony|Subject) +
                 X16.minus.17 + ClsListen + MusicClass + PianoPlay,
               data = ratings, control = lmerControl(optimizer = 'bobyqa'), REML = F)
lmer.3 <- lmer(Classical ~ Instrument + Harmony + Voice + (Voice|Subject) +
                 X16.minus.17 + ClsListen + MusicClass + PianoPlay,
               data = ratings, control = lmerControl(optimizer = 'bobyqa'), REML = F)
lmer.4 <- lmer(Classical ~ Instrument + Harmony + Voice + (Instrument + Harmony|Subject) +
                 X16.minus.17 + ClsListen + MusicClass + PianoPlay,
               data = ratings, control = lmerControl(optimizer = 'bobyqa'), REML = F)
lmer.5 <- lmer(Classical ~ Instrument + Harmony + Voice + (Instrument + Voice|Subject) +
                 X16.minus.17 + ClsListen + MusicClass + PianoPlay,
               data = ratings, control = lmerControl(optimizer = 'bobyqa'), REML = F)
```

```
lmer.6 <- lmer(Classical ~ Instrument + Harmony + Voice + (Harmony + Voice|Subject) +
                 X16.minus.17 + ClsListen + MusicClass + PianoPlay,
               data = ratings, control = lmerControl(optimizer = 'bobyqa'), REML = F)
lmer.7 <- lmer(Classical ~ Instrument + Harmony + Voice +
                 (Instrument + Harmony + Voice|Subject) +
                 X16.minus.17 + ClsListen + MusicClass + PianoPlay,
               data = ratings, control = lmerControl(optimizer = 'bobyqa'), REML = F)

# use anova() to see which model was the best out of the seven
compare <- as.data.frame(anova(lm.2, lmer.1, lmer.2, lmer.3, lmer.4,
                               lmer.5, lmer.6, lmer.7))
compare$Significant <- ifelse(compare$`Pr(>Chisq)` > 0.05, "No", "Yes")
knitr::kable(compare, caption = "ANOVA to Test Different Random Effects for Classical Ratings")

# the random effects for instrument and harmony still reduced AIC & BIC the most

###############
### Popular ###
###############

# test all possible combination of random effects (just 1, 2, or all of them)
# with the added fixed effects

lmer.1 <- lmer(Popular ~ Instrument + Harmony + Voice + (Instrument|Subject) +
                 X16.minus.17 + KnowRob + Composing + Selfdeclare,
               data = ratings, control = lmerControl(optimizer = 'bobyqa'), REML = F)
lmer.2 <- lmer(Popular ~ Instrument + Harmony + Voice + (Harmony|Subject) +
                 X16.minus.17 + KnowRob + Composing + Selfdeclare,
               data = ratings, control = lmerControl(optimizer = 'bobyqa'), REML = F)
lmer.3 <- lmer(Popular ~ Instrument + Harmony + Voice + (Voice|Subject) +
                 X16.minus.17 + KnowRob + Composing + Selfdeclare,
               data = ratings, control = lmerControl(optimizer = 'bobyqa'), REML = F)
lmer.4 <- lmer(Popular ~ Instrument + Harmony + Voice + (Instrument + Harmony|Subject) +
                 X16.minus.17 + KnowRob + Composing + Selfdeclare,
               data = ratings, control = lmerControl(optimizer = 'bobyqa'), REML = F)
lmer.5 <- lmer(Popular ~ Instrument + Harmony + Voice + (Instrument + Voice|Subject) +
                 X16.minus.17 + KnowRob + Composing + Selfdeclare,
               data = ratings, control = lmerControl(optimizer = 'bobyqa'), REML = F)
lmer.6 <- lmer(Popular ~ Instrument + Harmony + Voice + (Harmony + Voice|Subject) +
                 X16.minus.17 + KnowRob + Composing + Selfdeclare,
               data = ratings, control = lmerControl(optimizer = 'bobyqa'), REML = F)
lmer.7 <- lmer(Popular ~ Instrument + Harmony + Voice +
                 (Instrument + Harmony + Voice|Subject) +
                 X16.minus.17 + KnowRob + Composing + Selfdeclare,
               data = ratings, control = lmerControl(optimizer = 'bobyqa'), REML = F)

# use anova() to see which model was the best out of the seven
compare <- as.data.frame(anova(lm.2, lmer.1, lmer.2, lmer.3, lmer.4,
                               lmer.5, lmer.6, lmer.7))
compare$Significant <- ifelse(compare$`Pr(>Chisq)` > 0.05, "No", "Yes")
knitr::kable(compare, caption = "ANOVA to Test Different Random Effects for Popular Ratings")

# the random effects for instrument and harmony still reduced AIC the most, but not BIC
```

```
#########################
### Residual Analysis ###
#########################

source("residual-functions.R") # load R file to calculate conditional and
                               # random effects residuals

#################
### Classical ###
#################

# reload the model
cls <- lmer(Classical ~ Instrument + Harmony + Voice + (Instrument + Harmony|Subject) +
                X16.minus.17 + ClsListen + MusicClass + PianoPlay,
              data = ratings, control = lmerControl(optimizer = 'bobyqa'), REML = F)

library(dplyr) # formatting data set for residual visualization

# get the residuals of interest using functions from residual-functions.R
marginal0 <- r.marg(cls)
conditional0 <- r.cond(cls)
rand_effects0 <- r.reff(cls)

# get fitted values
fit_marg0 <- yhat.marg(cls)
fit_cond0 <- yhat.cond(cls)
fit_reff0 <- yhat.reff(cls)

# plot overall residuals
plot(fit_marg0, marginal0)
plot(fit_cond0, conditional0)
plot(fit_reff0, rand_effects0)

# attach row number information that matches observation number
# in the original ratings data (to later merge them for graphing)

marginal <- as.data.frame(marginal0) %>% mutate(X = 1:n())
conditional <- as.data.frame(conditional0) %>% mutate(X = 1:n())
rand_effects <- as.data.frame(rand_effects0) %>% mutate(X = 1:n())

fit_marg <- as.data.frame(fit_marg0) %>% mutate(X = 1:n())
fit_cond <- as.data.frame(fit_cond0) %>% mutate(X = 1:n())
fit_reff <- as.data.frame(fit_reff0) %>% mutate(X = 1:n())

# join residual information to the data table
marg_cls <- ratings[c(1,2)] %>%
  left_join(., fit_marg, by = "X") %>%
  left_join(., marginal, by = "X")

cond_cls <- ratings[c(1,2)] %>%
  left_join(., fit_cond, by = "X") %>%
  left_join(., conditional, by = "X")
```

```r
reff_cls <- ratings[c(1,2)] %>%
  left_join(., fit_reff, by = "X") %>%
  left_join(., rand_effects, by = "X")

# getting ready to plot by subject (randomly sample 30 of them)
subject <- as.numeric(unique(ratings$Subject))
subset <- sort(sample(subject,30))
subset_marg <- marg_cls[marg_cls$Subject %in% subset,] %>% dplyr::select(-X)
subset_cond <- cond_cls[cond_cls$Subject %in% subset,] %>% dplyr::select(-X)
subset_reff <- reff_cls[reff_cls$Subject %in% subset,] %>% dplyr::select(-X)

# plot marginal residuals by subject
names(subset_marg) <- c("Subject","Fitted Values","Marginal Residuals")
ggplot(subset_marg) + geom_point(aes(x=`Fitted Values`, y=`Marginal Residuals`)) +
  facet_wrap( ~ Subject, as.table=F) +
  geom_hline(yintercept=0) +
  geom_hline(yintercept=mean(subset_marg$`Marginal Residuals`), color = "Red")

# plot conditional residuals by subject
names(subset_cond) <- c("Subject","Fitted Values","Conditional Residuals")
ggplot(subset_cond) + geom_point(aes(x=`Fitted Values`, y=`Conditional Residuals`)) +
  facet_wrap( ~ Subject, as.table=F) +
  geom_hline(yintercept=0) +
  geom_hline(yintercept=mean(subset_cond$`Conditional Residuals`), color = "Red")

# plot marginal residuals by subject
names(subset_reff) <- c("Subject","Fitted Values","Random Effects Residuals")
ggplot(subset_reff) + geom_point(aes(x=`Fitted Values`, y=`Random Effects Residuals`)) +
  facet_wrap( ~ Subject, as.table=F) +
  geom_hline(yintercept=0) +
  geom_hline(yintercept=mean(subset_reff$`Random Effects Residuals`), color = "Red")

# look at qqnorm plots
par(mfrow = c(2,2))
qqnorm(marginal0,main="Marginal Residuals")
qqline(marginal0)
qqnorm(conditional0,main="Conditional Residuals")
qqline(conditional0)
qqnorm(rand_effects0,main="Random Effects Residuals")
qqline(rand_effects0)

# look at binned plots
library(arm)
par(mfrow = c(2,2))
binnedplot(fit_marg0, marginal0)
binnedplot(fit_cond0, conditional0)
binnedplot(fit_reff0, rand_effects0)

################
### Popular ###
################

# reload the model
```

```
pop <- lmer(Popular ~ Instrument + Harmony + Voice + (Instrument + Harmony|Subject) +
                X16.minus.17 + KnowRob + Composing + Selfdeclare,
              data = ratings, control = lmerControl(optimizer = 'bobyqa'), REML = F)

library(dplyr) # formatting data set for residual visualization
# (need to reload because of masking issues)

# get the residuals of interest using functions from residual-functions.R
marginal0 <- r.marg(pop)
conditional0 <- r.cond(pop)
rand_effects0 <- r.reff(pop)

# get fitted values
fit_marg0 <- yhat.marg(model_pop)
fit_cond0 <- yhat.cond(model_pop)
fit_reff0 <- yhat.reff(model_pop)

# attach row number information that matches observation number
# in the original ratings data (to later merge them for graphing)

marginal <- as.data.frame(marginal0) %>% mutate(X = 1:n())
conditional <- as.data.frame(conditional0) %>% mutate(X = 1:n())
rand_effects <- as.data.frame(rand_effects0) %>% mutate(X = 1:n())

fit_marg <- as.data.frame(fit_marg0) %>% mutate(X = 1:n())
fit_cond <- as.data.frame(fit_cond0) %>% mutate(X = 1:n())
fit_reff <- as.data.frame(fit_reff0) %>% mutate(X = 1:n())

# join residual information to the data table
marg_cls <- ratings[c(1,2)] %>%
  left_join(., fit_marg, by = "X") %>%
  left_join(., marginal, by = "X")

cond_cls <- ratings[c(1,2)] %>%
  left_join(., fit_cond, by = "X") %>%
  left_join(., conditional, by = "X")

reff_cls <- ratings[c(1,2)] %>%
  left_join(., fit_reff, by = "X") %>%
  left_join(., rand_effects, by = "X")

# getting ready to plot by subject (randomly sample 30 of them)
subject <- as.numeric(unique(ratings$Subject))
subset <- sort(sample(subject,30))
subset_marg <- marg_cls[marg_cls$Subject %in% subset,] %>% dplyr::select(-X)
subset_cond <- cond_cls[cond_cls$Subject %in% subset,] %>% dplyr::select(-X)
subset_reff <- reff_cls[reff_cls$Subject %in% subset,] %>% dplyr::select(-X)

# plot marginal residuals by subject
names(subset_marg) <- c("Subject","Fitted Values","Marginal Residuals")
ggplot(subset_marg) + geom_point(aes(x=`Fitted Values`, y=`Marginal Residuals`)) +
  facet_wrap( ~ Subject, as.table=F) +
  geom_hline(yintercept=0) +
```

```
  geom_hline(yintercept=mean(subset_marg$`Marginal Residuals`), color = "Red")

# plot conditional residuals by subject
names(subset_cond) <- c("Subject","Fitted Values","Conditional Residuals")
ggplot(subset_cond) + geom_point(aes(x=`Fitted Values`, y=`Conditional Residuals`)) +
  facet_wrap( ~ Subject, as.table=F) +
  geom_hline(yintercept=0) +
  geom_hline(yintercept=mean(subset_cond$`Conditional Residuals`), color = "Red")

# plot marginal residuals by subject
names(subset_reff) <- c("Subject","Fitted Values","Random Effects Residuals")
ggplot(subset_reff) + geom_point(aes(x=`Fitted Values`, y=`Random Effects Residuals`)) +
  facet_wrap( ~ Subject, as.table=F) +
  geom_hline(yintercept=0) +
  geom_hline(yintercept=mean(subset_reff$`Random Effects Residuals`), color = "Red")

# look at qqnorm plots
par(mfrow = c(2,2))
qqnorm(marginal0,main="Marginal Residuals")
qqline(marginal0)
qqnorm(conditional0,main="Conditional Residuals")
qqline(conditional0)
qqnorm(rand_effects0,main="Random Effects Residuals")
qqline(rand_effects0)

# look at binned plots
library(arm)
par(mfrow = c(2,2))
binnedplot(fit_marg0, marginal0)
binnedplot(fit_cond0, conditional0)
binnedplot(fit_reff0, rand_effects0)

################################################################################
# INTERACTION WITH MUSICIAN
################################################################################

#################
### Classical ###
#################

# Musician interacted with fixed effect variables
cls_musician <- lmer(Classical ~ Musician*(Instrument + Harmony + Voice  +
                X16.minus.17 + ClsListen + MusicClass + PianoPlay) +
                  (Instrument + Harmony|Subject), data = ratings,
                control = lmerControl(optimizer = 'bobyqa'), REML = F)

summary(cls_musician)

# Musician interacted with Harmony was significant, specifically harmony I-V-VI
# (positive coefficient = people who are not musicians had higher classical ratings
# with harmony I-V-VI than people who are musicians)

plot1 <- ggplot(ratings, aes(x = Musician, y = Classical)) +
```

```
  geom_boxplot(fill = "powderblue")
plot2 <- ggplot(ratings, aes(x = Harmony, y = Classical, fill = Musician)) +
  geom_boxplot()
plot3 <- ggplot(ratings, aes(x = Instrument, y = Classical, fill = Musician)) +
  geom_boxplot()
plot4 <- ggplot(ratings, aes(x = Voice, y = Classical, fill = Musician)) +
  geom_boxplot()
grid.arrange(plot1, plot2, plot3, plot4, ncol = 2)

################################################################################
# HARMONY INTERACTED WITH KNOWAXIS AND KNOWROB
################################################################################

#################
### Classical ###
#################

cls_harmony_int <- lmer(Classical ~ Instrument + Harmony*(KnowAxis + KnowRob) +
                        Voice + X16.minus.17 + ClsListen + MusicClass + PianoPlay +
                    (Instrument + Harmony|Subject), data = ratings,
                  control = lmerControl(optimizer = 'bobyqa'), REML = F)

summary(cls_harmony_int)

plot1 <- ggplot(ratings, aes(x = Harmony, y = Classical, fill = KnowRob)) +
  geom_boxplot()
plot2 <- ggplot(ratings, aes(x = Harmony, y = Classical, fill = KnowAxis)) +
  geom_boxplot()
grid.arrange(plot1, plot2, ncol = 2)
```