Untitled

group16

2024-05-09

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

##   
## Attaching package: 'gridExtra'

## The following object is masked from 'package:dplyr':  
##   
## combine

##   
## Attaching package: 'MASS'

## The following object is masked from 'package:patchwork':  
##   
## area

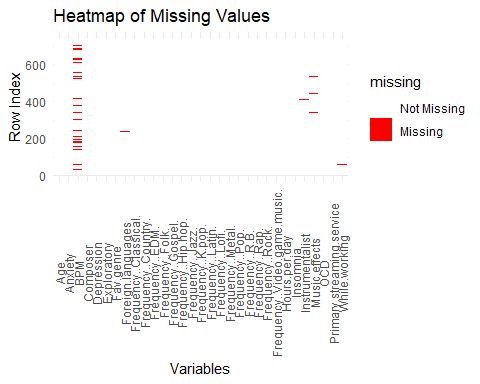
## The following object is masked from 'package:dplyr':  
##   
## select

## Data Overview

# Data overview  
mt.data.raw <- read.csv("mxmh\_survey\_results.csv")  
mt.data.raw.summary <- summary(mt.data.raw)

## Data Preprocessing

# 1. Remove Permission and Timestamp columns  
mt.data.raw <- mt.data.raw[, !(names(mt.data.raw) %in% c("Timestamp", "Permissions"))]  
  
# 2. Checking for duplicates  
mt.data <- mt.data.raw %>% distinct()  
dup.data <- setdiff(mt.data.raw, mt.data)  
  
# 3. Identifying NA's  
missing.matrix <- is.na(mt.data) | (mt.data == "")  
missing.df <- as.data.frame(missing.matrix)  
missing.df$row <- rownames(mt.data)  
missing.df <- tidyr::pivot\_longer(missing.df, -row, names\_to = "column", values\_to = "missing")  
  
heatmap <- ggplot(missing.df, aes(x = column, y = as.numeric(row), fill = missing)) +  
 geom\_tile() +  
 scale\_fill\_manual(values = c("white", "red"), na.value = "red", labels = c("Not Missing", "Missing")) +  
 labs(x = "Variables", y = "Row Index", title = "Heatmap of Missing Values") +  
 theme\_minimal() +  
 theme(axis.text.x = element\_text(angle = 90, vjust = 0.5))  
  
heatmap



# 4. Regress BPM~Fav.Genre  
# Assign scoring to the frequency response  
category.columns <- grep("^Frequency", names(mt.data), value = TRUE)  
  
for (col in category.columns) {  
 mt.data[[col]] <- factor(mt.data[[col]], levels = c("Never", "Rarely", "Sometimes", "Very frequently"))  
 mt.data[[col]] <- factor(as.numeric(mt.data[[col]]))  
}  
  
mt.data[["Fav.genre"]] <- factor(mt.data[["Fav.genre"]])  
  
# Drop NA's in other columns (non BPM)  
missing.df <- missing.df[missing.df$missing == TRUE, ]  
na.cols <- unique(missing.df$column)  
  
missing.count.df <- missing.df %>%  
 group\_by(column) %>%  
 summarise(Count = n()) %>%  
 ungroup()  
  
# Only BPM in missing.count.df has high count of NA values, so we'll drop the NA from the other columns and try to fill in NA of the BPM column through prediction  
na.cols <- na.cols[na.cols != "BPM"]  
print(na.cols)

## [1] "Music.effects" "Age"   
## [3] "While.working" "Primary.streaming.service"  
## [5] "Instrumentalist" "Foreign.languages"   
## [7] "Composer"

mt.data[na.cols] <- lapply(mt.data[na.cols], function(x) {  
 x[x == ""] <- NA  
 return(x)  
})  
  
mt.data <- mt.data %>%  
 filter\_at(vars(all\_of(na.cols)), all\_vars(!is.na(.)))  
  
### 102 entries after removing NA data based on the code belows ###  
# missing.df.check <- as.data.frame(is.na(mt.data) | (mt.data == ""))  
# missing.df.check$row <- rownames(mt.data)  
# missing.df.check <- tidyr::pivot\_longer(missing.df.check, -row, names\_to = "column", values\_to = "missing")  
# missing.df.check <- missing.df.check[missing.df.check$missing == TRUE,]  
  
char.cols <- c("Primary.streaming.service", "While.working", "Instrumentalist", "Composer", "Exploratory", "Foreign.languages", "Music.effects")  
  
for (col in char.cols) {  
 col.val <- unique(mt.data[[col]])  
 mt.data[[col]] <- factor(mt.data[[col]], levels = col.val)  
}

# Apply regression to fill in NAs for BPM  
# Removing BPM outliers  
  
max.bpm <- max(mt.data$BPM, na.rm = TRUE)  
mt.data <- mt.data[mt.data$BPM != max.bpm | is.na(mt.data$BPM), ]  
  
max.bpm <- max(mt.data$BPM, na.rm = TRUE)  
mt.data <- mt.data[mt.data$BPM != max.bpm | is.na(mt.data$BPM), ]  
  
bpm.na <- mt.data[is.na(mt.data[["BPM"]]), ]  
bpm.not.na <- mt.data[!is.na(mt.data[["BPM"]]), ]  
  
bpm.lm <- lm(BPM~., data=bpm.not.na)  
bpm.model <- stepAIC(bpm.lm, direction="both")  
bpm.na$BPM <- predict(bpm.model, newdata = bpm.na)  
  
mt.data$BPM[is.na(mt.data[["BPM"]])] <- bpm.na$BPM  
  
missing.bpm.check <- as.data.frame(is.na(mt.data) | (mt.data == ""))  
missing.bpm.check$row <- rownames(mt.data)  
missing.bpm.check <- tidyr::pivot\_longer(missing.bpm.check, -row, names\_to = "column", values\_to = "missing")  
missing.bpm.check <- missing.bpm.check[missing.bpm.check$missing == TRUE,]

DV.histograms <- list()  
DV.charts <- list()  
# Loop through each column in the dataset  
for (col.name in names(mt.data.raw)) {  
 if (is.numeric(mt.data.raw[[col.name]])) {  
 gg <- ggplot(mt.data.raw, aes(x = !!sym(col.name))) +  
 geom\_histogram(fill = "skyblue", color = "black") +  
 labs(x = col.name, y = "Frequency") +   
 theme(plot.margin = unit(c(0, 0, 0, 0), "cm"))  
 DV.histograms[[col.name]] <- gg  
 } else {  
 if (grepl("^Frequency", col.name)) next  
 gg<- ggplot(mt.data.raw, aes\_string(x = col.name)) +  
 geom\_bar(fill = "skyblue") +  
 labs(x = col.name, y = "Count") +   
 theme(plot.margin = unit(c(0, 0, 0, 0), "cm")) +  
 theme(axis.text.x = element\_text(angle = 45, hjust = 1))  
 DV.charts[[col.name]] <- gg  
 }  
}

## Warning: `aes\_string()` was deprecated in ggplot2 3.0.0.  
## ℹ Please use tidy evaluation idioms with `aes()`.  
## ℹ See also `vignette("ggplot2-in-packages")` for more information.  
## This warning is displayed once every 8 hours.  
## Call `lifecycle::last\_lifecycle\_warnings()` to see where this warning was  
## generated.

DV.histograms.cleaned <- list()  
DV.charts.cleaned <- list()  
# Loop through each column in the dataset  
for (col.name in names(mt.data)) {  
 if (is.numeric(mt.data[[col.name]])) {  
 gg <- ggplot(mt.data, aes(x = !!sym(col.name))) +  
 geom\_histogram(fill = "skyblue", color = "black") +  
 labs(x = col.name, y = "Frequency") +   
 theme(plot.margin = unit(c(0, 0, 0, 0), "cm"))  
 DV.histograms.cleaned[[col.name]] <- gg  
 } else {  
 if (grepl("^Frequency", col.name)) next  
 gg<- ggplot(mt.data, aes\_string(x = col.name)) +  
 geom\_bar(fill = "skyblue") +  
 labs(x = col.name, y = "Count") +   
 theme(plot.margin = unit(c(0, 0, 0, 0), "cm")) +  
 theme(axis.text.x = element\_text(angle = 45, hjust = 1))  
 DV.charts.cleaned[[col.name]] <- gg  
 }  
}  
  
combined.histograms <- wrap\_plots(DV.histograms, ncol = 3)  
# combined.charts <- wrap\_plots(DV.charts, ncol = 1)   
combined.histograms.cleaned <- wrap\_plots(DV.histograms.cleaned, ncol = 3)  
# combined.charts.cleaned <- wrap\_plots(DV.charts.cleaned, ncol = 1)   
  
print(combined.histograms)

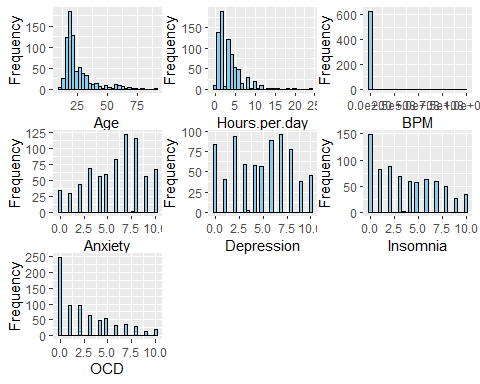
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.

## Warning: Removed 1 row containing non-finite outside the scale range  
## (`stat\_bin()`).

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.

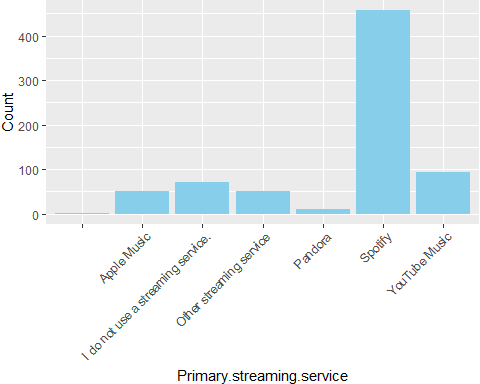
## Warning: Removed 107 rows containing non-finite outside the scale range  
## (`stat\_bin()`).

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.

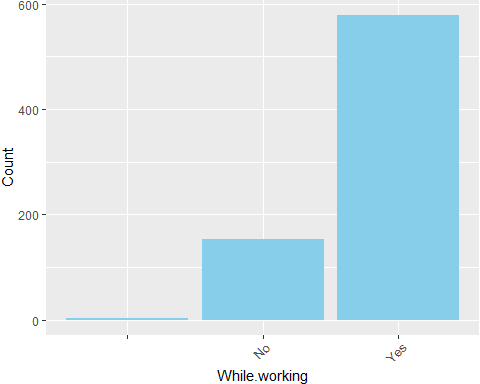


print(DV.charts)

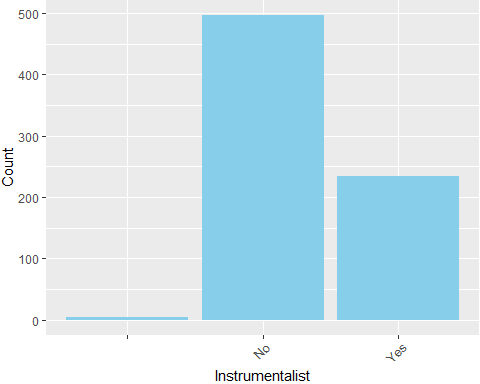
## $Primary.streaming.service



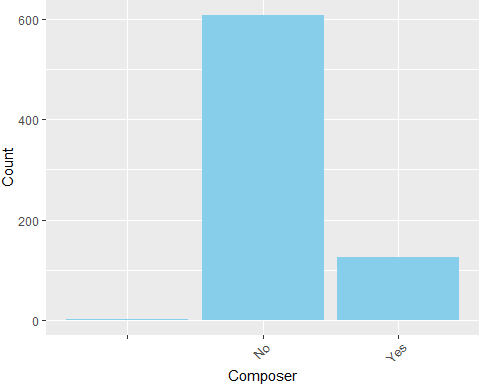
##   
## $While.working



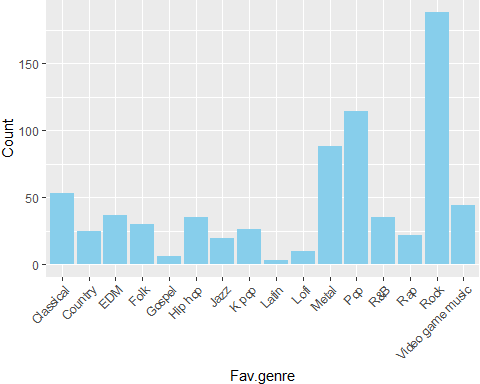
##   
## $Instrumentalist



##   
## $Composer



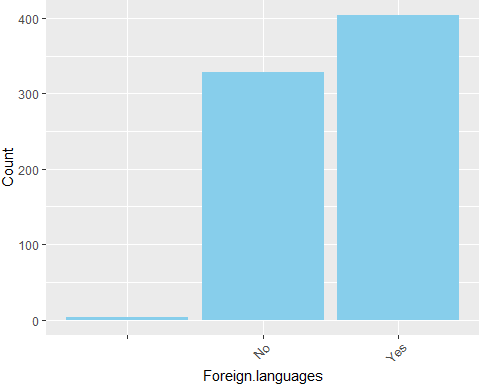
##   
## $Fav.genre



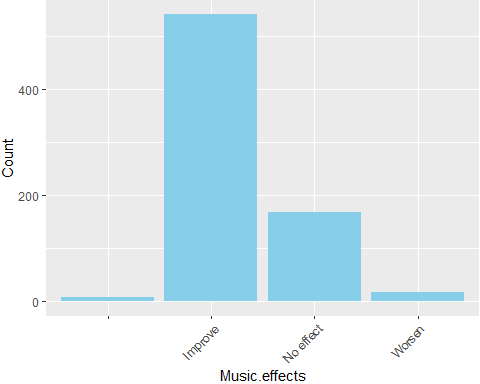
##   
## $Exploratory



##   
## $Foreign.languages

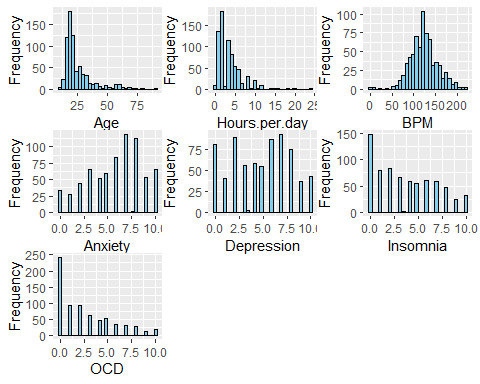


##   
## $Music.effects



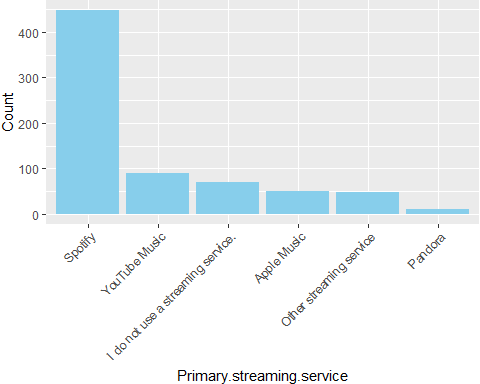
print(combined.histograms.cleaned)

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.

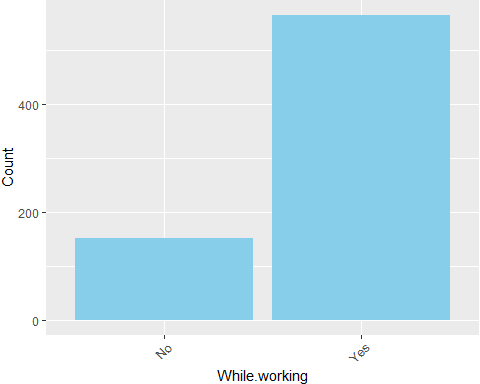


print(DV.charts.cleaned)

## $Primary.streaming.service



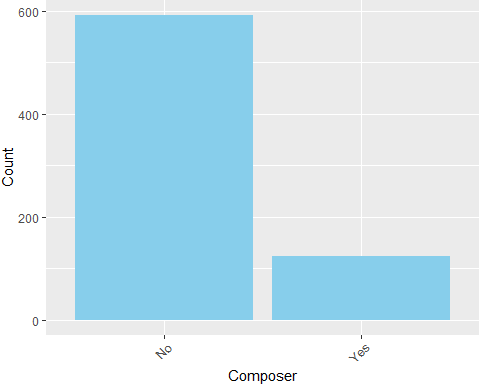
##   
## $While.working



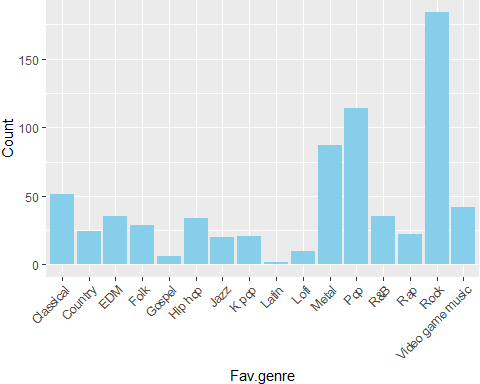
##   
## $Instrumentalist



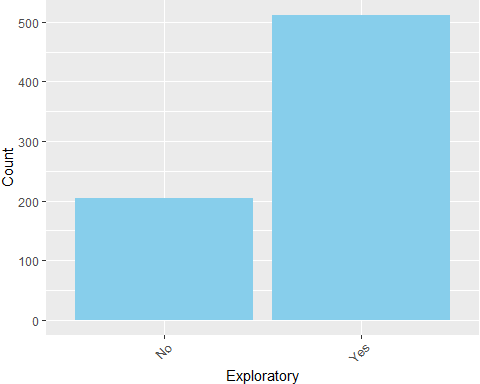
##   
## $Composer



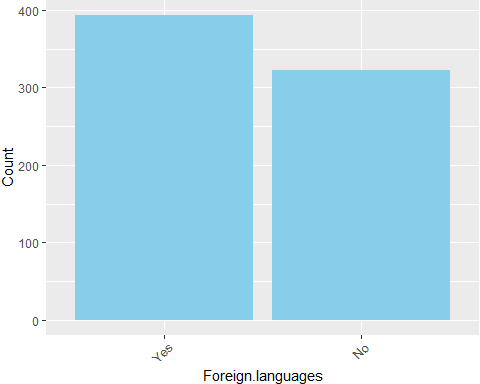
##   
## $Fav.genre



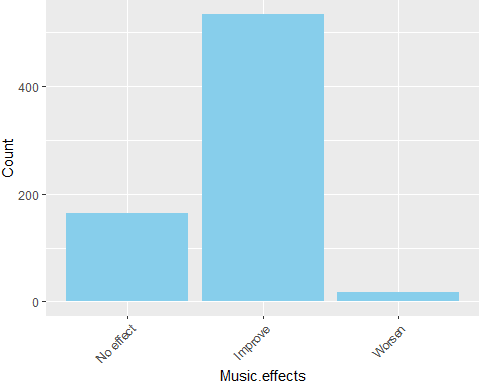
##   
## $Exploratory



##   
## $Foreign.languages



##   
## $Music.effects



write.csv(mt.data, "mt\_data\_cleaned.csv", row.names = FALSE)

* No duplicated data found
* Age has 0.14% values are NA, while BPM has 14.54% NA’s values (moderate level)
* Some contains “” values instead NA so we fill all “” with NA’s values before continue
* Plot heatmap of all missing values
* Plot histogram of BPM before and after removing the max value, then remove BPM > 600 (3 histograms total), justify that we believe these are typos
* Removed the two max values for BPM