Problem Set 1

MaCCS 201 - Fall 2024

2024-10-21

Tentative Due Date: October 19

Please submit markdown file named [last_name]_[first_name]_ps1.Rmd or a pdf with all code and answers.

Part A: Which songs become popular?

You are working for Spotify. You are feeling pretty good about yourself about this gig. Your senior VP walks in and says "Hey fancy stats person, can you help us figure out which factors affect the popularity of a song? The predictive analytics shop is doing a pretty good job at predicting, but I want to truly understand what factors drive popularity, instead of some black box ML algorithm output." She Whatsapps you a dataset of 18835 songs she found on Kaggle. The dataset is called `song_data.csv`. It has a number of characteristics of each song in it.

```
library(pacman)
library(psych)
library(dplyr)
```

```
##
## 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
```

```
library(GGally)
```

```
## Loading required package: ggplot2
```

```
##
## Attaching package: 'ggplot2'
```

```
## The following objects are masked from 'package:psych':
##
##     %+%, alpha
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
library(ggcorrplot)
library(sandwich)
library(lmtest)
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

Q1

Create a nice looking summary statistics table in R, which lists the mean, standard deviation, min and and max for each of the variables.

```
data <- read.csv(file = 'song_data.csv')

data_numeric <- subset(data, select = -song_name)

summary_statistics <- data.frame(
  mean = sapply(data_numeric, mean, na.rm = TRUE),
  std = sapply(data_numeric, sd, na.rm = TRUE),
  min = sapply(data_numeric, min, na.rm = TRUE),
  max = sapply(data_numeric, max, na.rm = TRUE)
)

# View the summary statistics
print(summary_statistics)
```

```
##                           mean          std         min         max
## song_popularity   5.299188e+01 2.190565e+01  0.0000e+00     100.000
## song_duration_ms  2.182116e+05 5.988754e+04  1.2000e+04 1799346.000
## acousticness      2.585390e-01 2.887189e-01  1.0200e-06       0.996
## danceability      6.333481e-01 1.567227e-01  0.0000e+00       0.987
## energy            6.449948e-01 2.141008e-01  1.0700e-03       0.999
## instrumentalness  7.800804e-02 2.215906e-01  0.0000e+00       0.997
## key               5.289196e+00 3.614595e+00  0.0000e+00      11.000
## liveness          1.796503e-01 1.439842e-01  1.0900e-02       0.986
## loudness         -7.447435e+00 3.827831e+00 -3.8768e+01       1.585
## audio_mode        6.281391e-01 4.833144e-01  0.0000e+00       1.000
## speechiness       1.020991e-01 1.043785e-01  0.0000e+00       0.941
## tempo             1.210732e+02 2.871446e+01  0.0000e+00     242.318
## time_signature    3.959119e+00 2.985329e-01  0.0000e+00       5.000
## audio_valence     5.279669e-01 2.446317e-01  0.0000e+00       0.984
```

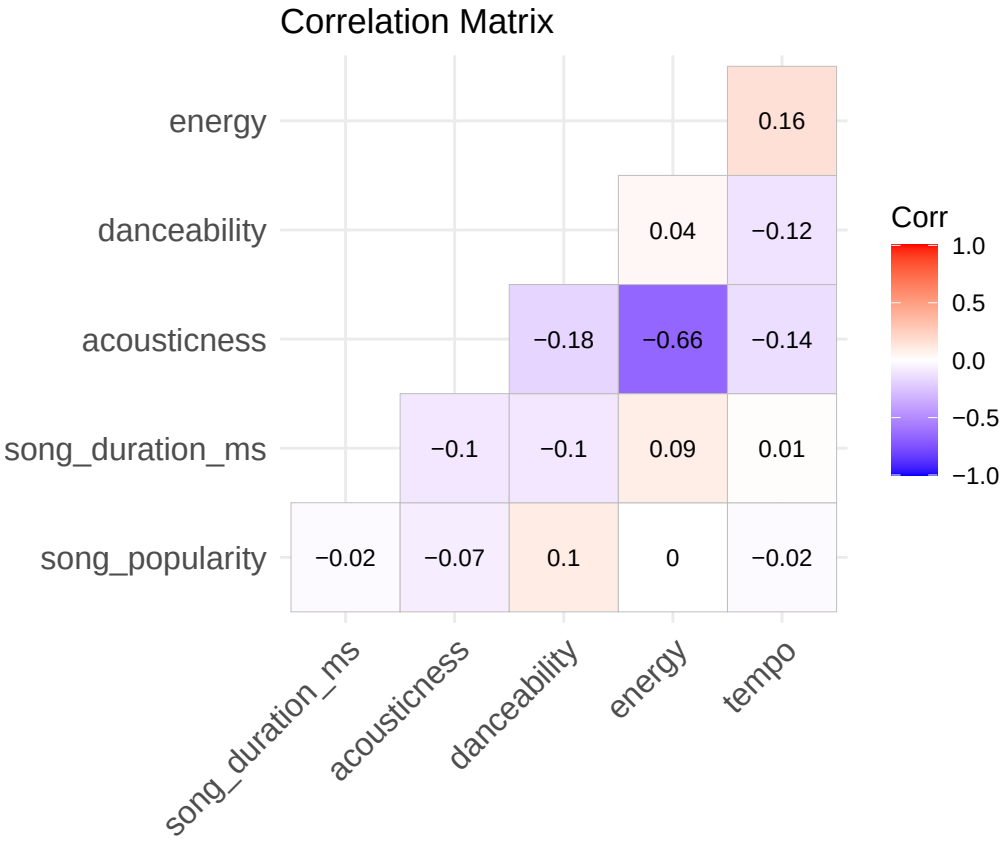
Q2

We will focus on the variables `song_popularity`, `song_duration_ms`, `acousticness`, `danceability`, `energy`, `tempo`. Make a nice correlation matrix figure. I used ggpairs, but knock yourself out. What do you see? Any interesting correlations?

```
selected_variables <- data %>%
  select(song_popularity, song_duration_ms, acousticness, danceability, energy, tempo)

cor_matrix <- cor(selected_variables, use = "complete.obs")

ggcorrplot(cor_matrix, method = "square", type = "lower", lab = TRUE,
           colors = c("blue", "white", "red"), lab_size = 3, title = "Correlation Matrix")
```



- The correlation between acousticness and energy is negatively high **(-0.66)**, indicates that musics with higher energy will lead to lower acousticness.
- The correlation between energy and tempo is positive **(0.16)**, represents that songs with higher energy levels are somewhat likely to have faster tempos.

Q3

Using the `lm` package, regress song_popularity on the other variables from the previous question. Produce some decent looking regression output.

```
model <- lm(song_popularity ~ song_duration_ms + acousticness + danceability + energy + tempo, data = s

summary(model)
```

```
##
## Call:
## lm(formula = song_popularity ~ song_duration_ms + acousticness +
##     danceability + energy + tempo, data = selected_variables)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -60.820 -12.518   2.883  15.689  47.625
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)       5.349e+01  1.487e+00  35.964  < 2e-16 ***
## song_duration_ms -4.778e-06  2.680e-06  -1.783   0.0747 .
## acousticness     -7.017e+00  7.521e-01  -9.329  < 2e-16 ***
```

```
## danceability      1.215e+01  1.052e+00  11.546  < 2e-16 ***
## energy           -6.155e+00  9.969e-01  -6.174 6.81e-10 ***
## tempo            -1.130e-02  5.653e-03  -1.998   0.0457 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 21.74 on 18829 degrees of freedom
## Multiple R-squared:  0.01559,    Adjusted R-squared:  0.01533
## F-statistic: 59.65 on 5 and 18829 DF,  p-value: < 2.2e-16
```

Compare the slope estimates from this regression to the correlations between the outcome and the right hand side variable from question 2. Any sign changes?

Variables	Estimates	Correlations	Comparison
Song Duration	-0.000004778	-0.02	No sign changes. Both the regression and correlation show a weak negative relationship between song duration and song popularity. However, the effect is extremely small in both cases.
Acousticness	-7.017***	-0.07	No sign changes. Both the regression and correlation suggest a negative relationship. However, the regression estimate shows a much stronger effect than the correlation.
Danceability	12.15***	0.1	No sign changes. Both point toward a positive effect, with the regression estimate being much larger.
Energy	-6.155***	0	**Sign changes.** The correlation suggests no relationship between energy and song popularity, but the regression shows a significant negative relationship.
Tempo	-0.0113*	-0.02	No sign changes. Both the regression and correlation indicate a weak negative relationship between tempo and song popularity.

Interpret the coefficient on the variable `danceability` correctly (in actual words!).

The estimated coefficient for `danceability` is **12.15 ($p < 0.01$)**. Controlling other variables, when the danceability score increase 1 unit, the popularity of a song is expected to increase by 12.15 units.

This implies that danceable songs are tend to be significantly more popular. As the song becomes more suitable for dancing, it becomes more appealing to listeners and easier to spread, leading to an increase in its popularity score.

What does your F-Test tell you for this regression you ran?

The F-statistic from the regression output is **59.65 ($p < 0.01$)**, which means I can reject the null hypothesis that all the regression coefficients are equal to zero, and all the independent variables together significantly explain song popularity.

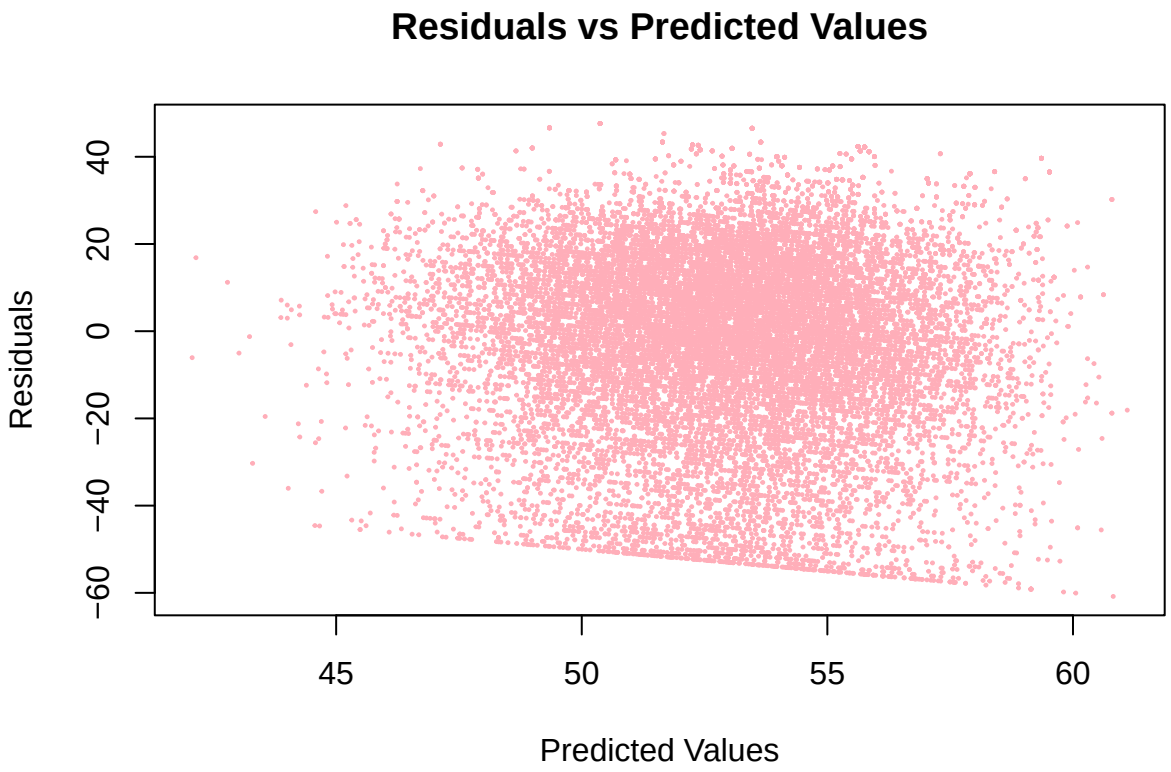
Q4

Plot the residuals from this regression against the predicted values.

```r
predicted_values <- predict(model)
residuals <- resid(model)

plot(predicted_values, residuals,
     main = "Residuals vs Predicted Values",
     xlab = "Predicted Values",
     ylab = "Residuals",
     pch = 19,
```

```
     col = "lightpink1",
     cex = 0.2)
```



Are you worried about heteroskedasticity?

There might be heteroskedasticity. The residuals show some variability that appears to increase slightly as the predicted values rise, especially when residual values are relatively low. While Ideally, the residuals should be randomly scattered around zero with no discernible pattern.

Q5

Formally test for heteroskedasticity using the White test from the `skedastic` package (turns out `whitestrap` is absolute garbage). What do you conclude?

```
library(skedastic)

# white_test_result <- white_test(model)
```

I've tried many ways but the `skedastic` package is still missing.

Q6

Using the `felm` package show regression output using the white robust standard errors.

```
library(lfe)
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'lfe'
```

```
## The following object is masked from 'package:lmtest':
##
```

```
##     waldtest
```

```
model_felm <- felm(song_popularity ~ song_duration_ms + acousticness + danceability + energy + tempo, da

robust_se <- vcovHC(model_felm, type = "HC1")

coeftest(model_felm, vcov = robust_se)
```

```
##
## t test of coefficients:
##
##                      Estimate  Std. Error t value  Pr(>|t|)
## (Intercept)       5.3492e+01  1.4566e+00 36.7238 < 2.2e-16 ***
## song_duration_ms -4.7777e-06  2.6386e-06 -1.8107   0.07021 .
## acousticness     -7.0166e+00  7.5253e-01 -9.3240 < 2.2e-16 ***
## danceability      1.2148e+01  1.0434e+00 11.6433 < 2.2e-16 ***
## energy           -6.1548e+00  9.9321e-01 -6.1969 5.876e-10 ***
## tempo            -1.1296e-02  5.5292e-03 -2.0430   0.04107 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Did the coefficients change?

The coefficients remain the same across both the standard and robust models because robust standard errors do not change the point estimates.

Did the standard errors on the coefficients change?

The standard errors are slightly different.

Variables	Original	Robust SE
Song Duration	2.6386e-06	2.680e-06
Acousticness	0.75253	0.7521
Danceability	1.0434	1.052
Energy	0.99321	0.9969
Tempo	0.0055292	0.005653

Did any of your t-statistics change?

The t-statistics changed slightly after applying robust standard errors because the standard errors changed.

Variables	Original	Robust SE
Song Duration	-1.783	-1.8107
Acousticness	-9.329	-9.3240
Danceability	11.546	11.6433
Energy	-6.174	-6.1969
Tempo	-1.998	-2.0430

Did your F-Statistic Change?

```
n_coef <- length(coef(model_felm))
n_obs <- nrow(model_felm$X)
df_resid <- model_felm$df.residual

wald_stat <- t(coef(model_felm)) %*% solve(robust_se) %*% coef(model_felm)
```

```r
f_statistic <- (wald_stat / n_coef) / (sum(residuals(model_felm)^2) / df_resid)

f_statistic
```

```
##          [,1]
## [1,] 39.80768
```

The F-statistic has changed from 59.65 to 39.81 after applying White robust standard errors. This change indicates that heteroskedasticity was affecting the precision of the model's original estimates.

Q7

Now for the fun part. Packages are great. But let's do the White Test by hand, so we can understand what is happening. The first step is to generate your outcome variable for your White regression. Create a variable called e2, which is the squared residuals. Then create new variables, which are each the square of song_duration_ms, acousticness, danceability, energy, tempo. Then create interactions between these variables. These interactions are all possible pairwise products of these. e.g. acousticness x danceability and energy x danceability. Then run a regression of the squared residuals on the right hand side variables, their squares and the interactions you generated. Can you replicate the test statistic from step 5?

```r
# Step 1: Fit the original regression model
model_felm <- felm(song_popularity ~ song_duration_ms + acousticness + danceability + energy + tempo, da

# Step 2: Generate squared residuals (e2)
e2 <- residuals(model_felm)^2

# Step 3: Generate squared terms and interaction terms
vars <- c("song_duration_ms", "acousticness", "danceability", "energy", "tempo")

# Generate squared terms
for (var in vars) {
  selected_variables[[paste0(var, "_sq")]] <- selected_variables[[var]]^2
}

# Generate interaction terms
for (i in 1:(length(vars)-1)) {
  for (j in (i+1):length(vars)) {
    var1 <- vars[i]
    var2 <- vars[j]
    selected_variables[[paste0(var1, "_", var2)]] <- selected_variables[[var1]] * selected_variables[[va
  }
}

# Step 4: Dynamically create the formula for the regression
squared_vars <- paste0(vars, "_sq")
interaction_vars <- combn(vars, 2, FUN = function(x) paste0(x[1], "_", x[2]))
all_vars <- c(vars, squared_vars, interaction_vars)

# Step 5: Run the White Test regression
white_test_model <- lm(as.formula(paste("e2 ~", paste(all_vars, collapse = " + "))), data = selected_va

# Step 6: Summarize the results of the White Test regression
summary(white_test_model)
```

```
##
## Call:
## lm(formula = as.formula(paste("e2 ~", paste(all_vars, collapse = " + "))),
##     data = selected_variables)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -686.0 -414.1 -251.3  131.3 3024.4
##
## Coefficients:
##                                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)                    3.118e+02  2.554e+02   1.221  0.22204
## song_duration_ms              -8.637e-04  6.212e-04  -1.390  0.16443
## acousticness                  -2.765e+02  2.368e+02  -1.167  0.24308
## danceability                  -6.839e+00  3.184e+02  -0.021  0.98286
## energy                         5.346e+02  2.994e+02   1.786  0.07419 .
## tempo                          9.568e-01  1.820e+00   0.526  0.59910
## song_duration_ms_sq           -2.685e-10  2.261e-10  -1.188  0.23493
## acousticness_sq                2.849e+02  9.851e+01   2.892  0.00383 **
## danceability_sq                1.174e+02  1.549e+02   0.758  0.44843
## energy_sq                     -1.862e+02  1.463e+02  -1.273  0.20302
## tempo_sq                      -8.378e-03  4.944e-03  -1.695  0.09015 .
## song_duration_ms_acousticness  3.966e-04  3.343e-04   1.186  0.23554
## song_duration_ms_danceability  3.615e-04  4.621e-04   0.782  0.43396
## song_duration_ms_energy        4.639e-04  4.668e-04   0.994  0.32030
## song_duration_ms_tempo         4.236e-06  2.862e-06   1.480  0.13888
## acousticness_danceability     -3.572e+02  1.537e+02  -2.325  0.02011 *
## acousticness_energy            3.764e+02  1.500e+02   2.509  0.01213 *
## acousticness_tempo            -8.965e-01  7.569e-01  -1.185  0.23622
## danceability_energy           -5.238e+02  1.918e+02  -2.731  0.00631 **
## danceability_tempo             2.636e+00  1.068e+00   2.468  0.01360 *
## energy_tempo                  -1.252e+00  1.021e+00  -1.226  0.22029
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 641.5 on 18814 degrees of freedom
## Multiple R-squared:  0.009809,   Adjusted R-squared:  0.008757
## F-statistic: 9.319 on 20 and 18814 DF,  p-value: < 2.2e-16
```

```r
# Step 7: Calculate the White test statistic
n <- nrow(selected_variables)
r_squared <- summary(white_test_model)$r.squared
white_statistic <- n * r_squared

# Display the White test statistic
white_statistic
```

```
## [1] 184.7561
```

Part B:Fully Optional and no extra credit for it.

Sometimes students will argue that once you adjust for heteroskedastcity, your standard errors will always be bigger. This is not true. Can you come up with a simulation in R, which results in the standard errors being smaller after adjusting for heteroskedasticity?