Problem Set 2

MaCCS 201 - Fall 2024

2025-10-29

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.

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

```
songs <- read.csv("song_data_2025.csv")
describe(songs)
```

```
##                   vars     n      mean       sd    median   trimmed      mad
## X                    1 17835   9408.39  5434.85   9395.00   9406.90  6975.63
## song_name*           2 17835   6274.69  3621.80   6283.00   6275.36  4642.02
## song_popularity      3 17835     52.99    21.88     55.00     54.30    20.76
## song_duration_ms     4 17835 217922.94 59097.92 211186.00 213437.41 42719.64
## acousticness         5 17835      0.26     0.29      0.13      0.21     0.19
## danceability         6 17835      0.63     0.16      0.64      0.64     0.16
## energy               7 17835      0.65     0.21      0.67      0.66     0.22
## instrumentalness     8 17835      0.08     0.22      0.00      0.01     0.00
## key                  9 17835      5.30     3.61      5.00      5.26     4.45
## liveness            10 17835      0.18     0.14      0.12      0.15     0.06
## loudness            11 17835     -7.44     3.83     -6.54     -6.94     2.84
## audio_mode          12 17835      0.63     0.48      1.00      0.66     0.00
## speechiness         13 17835      0.10     0.10      0.06      0.08     0.03
## tempo               14 17835    121.09    28.72    120.02    119.55    29.72
## time_signature      15 17835      3.96     0.30      4.00      4.00     0.00
## audio_valence       16 17835      0.53     0.24      0.53      0.53     0.29
##                       min        max      range  skew kurtosis     se
## X                    1.00   18835.00   18834.00  0.00    -1.20  40.70
## song_name*           1.00   12519.00   12518.00  0.00    -1.20  27.12
## song_popularity      0.00     100.00     100.00 -0.49    -0.18   0.16
## song_duration_ms 12000.00 1799346.00 1787346.00  3.01    43.33 442.52
## acousticness         0.00       1.00       1.00  1.07    -0.09   0.00
## danceability         0.00       0.98       0.98 -0.39    -0.09   0.00
```

```
## energy               0.00       1.00       1.00 -0.62    -0.13   0.00
## instrumentalness     0.00       1.00       1.00  3.00     7.69   0.00
## key                  0.00      11.00      11.00  0.00    -1.31   0.03
## liveness             0.01       0.99       0.98  2.22     5.79   0.00
## loudness           -38.77       1.58      40.35 -1.93     6.54   0.03
## audio_mode           0.00       1.00       1.00 -0.53    -1.72   0.00
## speechiness          0.00       0.94       0.94  2.28     6.55   0.00
## tempo                0.00     242.32     242.32  0.44    -0.22   0.22
## time_signature       0.00       5.00       5.00 -5.01    46.14   0.00
## audio_valence        0.00       0.98       0.98 -0.02    -0.97   0.00
```

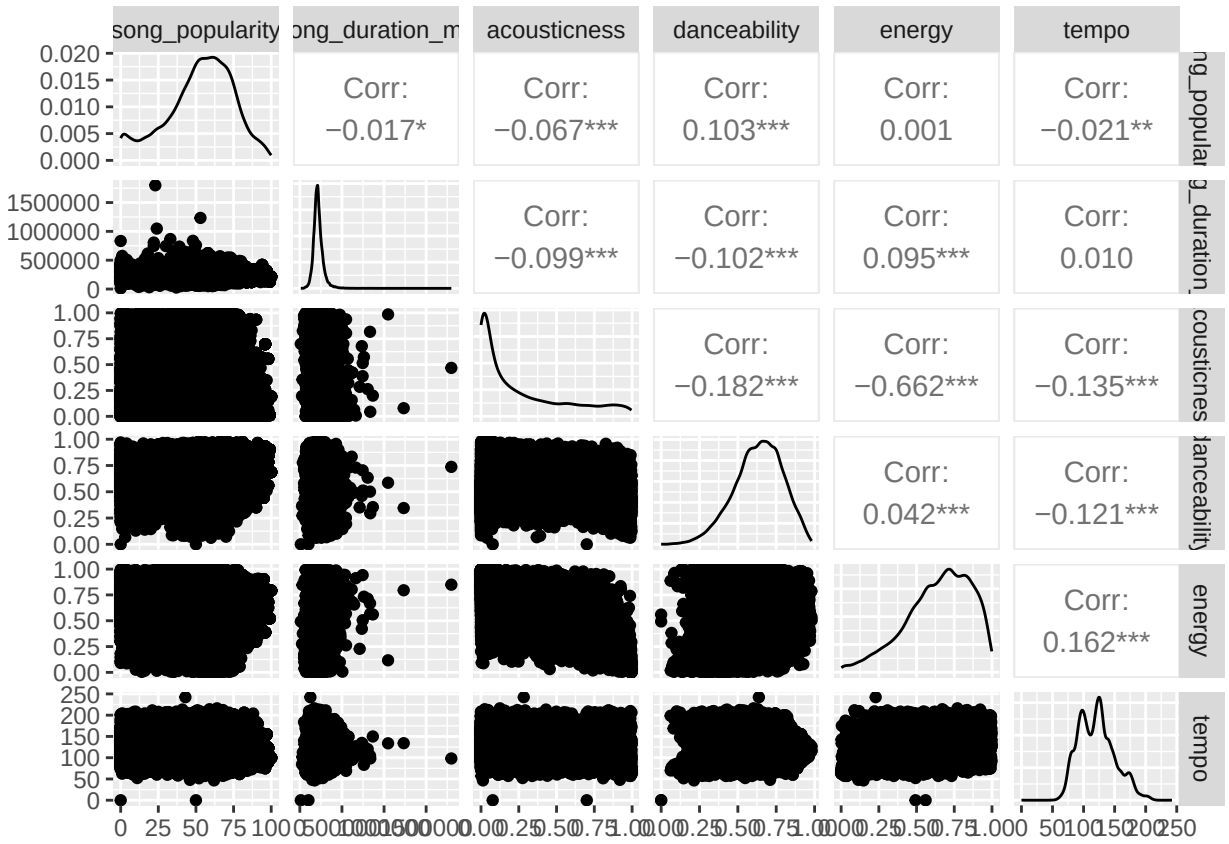
```
stargazer(songs %>% as.data.frame(),
          type = "text")
```

```
##
## ===============================================================
## Statistic           N       Mean      St. Dev.    Min      Max
## ---------------------------------------------------------------
## X                17,835  9,408.387  5,434.853     1     18,835
## song_popularity  17,835   52.995      21.878      0       100
## song_duration_ms 17,835 217,922.900 59,097.920 12,000  1,799,346
## acousticness     17,835    0.258      0.289    0.00000   0.996
## danceability     17,835    0.634      0.157     0.000    0.981
## energy           17,835    0.645      0.214     0.001    0.999
## instrumentalness 17,835    0.077      0.220     0.000    0.997
## key              17,835    5.296      3.615       0       11
## liveness         17,835    0.179      0.144     0.011    0.986
## loudness         17,835   -7.439      3.828    -38.768   1.585
## audio_mode       17,835    0.628      0.483       0        1
## speechiness      17,835    0.102      0.105     0.000    0.941
## tempo            17,835   121.094     28.720    0.000   242.318
## time_signature   17,835    3.959      0.296       0        5
## audio_valence    17,835    0.529      0.244     0.000    0.984
## ---------------------------------------------------------------
```

2. 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?

```
songs2 <- songs %>%
  dplyr::select(song_popularity, song_duration_ms, acousticness, danceability, energy, tempo)

ggpairs(songs2)
```



3. Using the lm package, regress song_popularity on the other variables from the previous question. Produce some decent looking regression output. 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? Interpret the coefficient on the variable danceability correctly (in actual words!) What does your F-Test tell you for this regression you ran?

Answer: - Energy is the only variable whose coefficient sign is different in the correlation matrix compared to the linear regression.

- The interpretation of the coefficient associated to the danceability variable is: "a one-unit increase in the danceability variable is associated with a 11.94 unit increase in the song popularity variable, holding all other variables fixed.
- With a F-statistic of 57 and a p-value smaller than 0.01, we reject the null hypothesis of all coefficients being jointly equal to 0.

```
model_1 <- lm( song_popularity ~ song_duration_ms + acousticness + danceability + energy + tempo, data=s
summary(model_1)
```

```
##
## Call:
## lm(formula = song_popularity ~ song_duration_ms + acousticness +
##     danceability + energy + tempo, data = songs)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -60.961 -12.527   2.834  15.695  47.799
##
## Coefficients:
```

```
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)       5.363e+01  1.530e+00  35.048  < 2e-16 ***
## song_duration_ms -4.246e-06  2.787e-06  -1.524   0.1276
## acousticness     -7.344e+00  7.718e-01  -9.516  < 2e-16 ***
## danceability      1.194e+01  1.081e+00  11.044  < 2e-16 ***
## energy           -6.519e+00  1.024e+00  -6.363 2.03e-10 ***
## tempo            -9.733e-03  5.800e-03  -1.678   0.0933 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 21.71 on 17829 degrees of freedom
## Multiple R-squared:  0.01573,    Adjusted R-squared:  0.01546
## F-statistic:    57 on 5 and 17829 DF,  p-value: < 2.2e-16
```

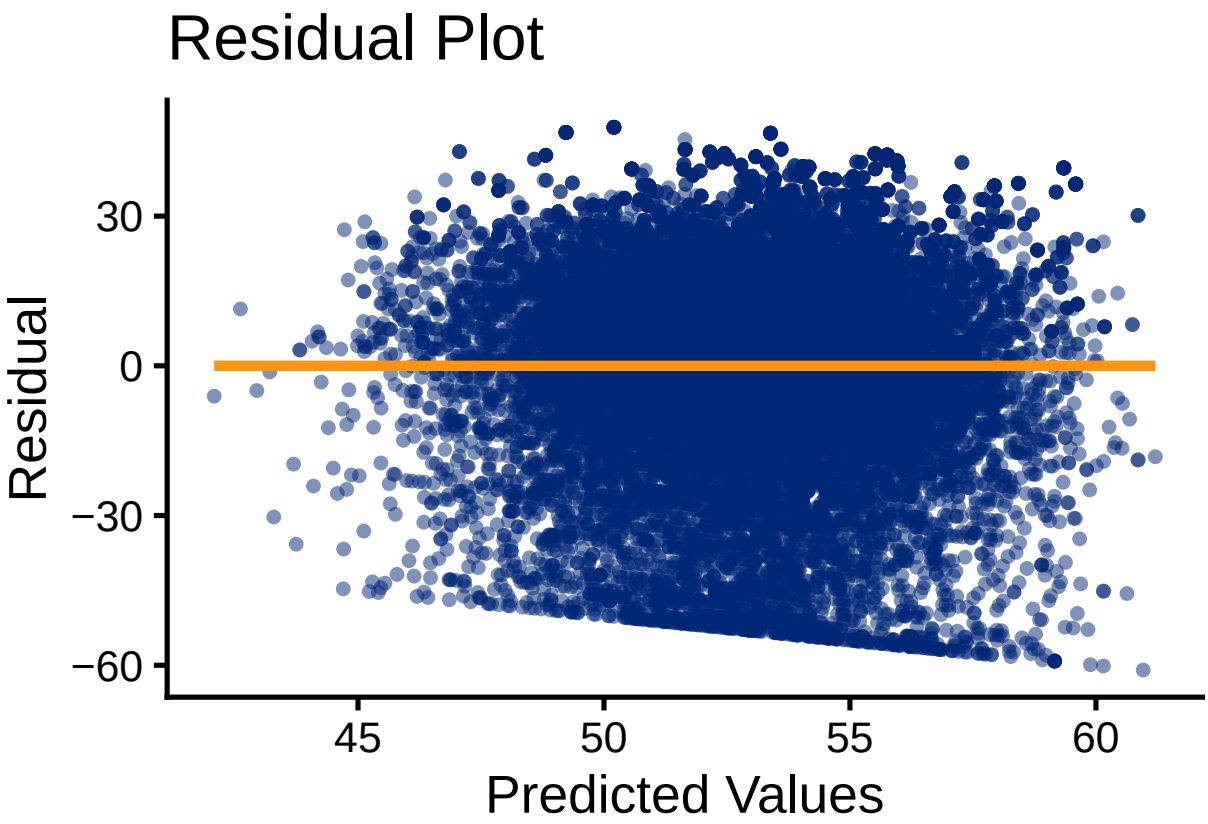
4. Plot the residuals from this regression against the predicted values. Are you worried about heteroskedasticity?

Answer: there doesn't seem to be heteroskedasticity, although the censoring of the data reveals some other type of issue. The lowess curve also looks flat and centered at 0.

```
songs$res <- model_1$resid
songs$yhat <- songs$song_popularity - songs$res

ggplot(songs, aes(x=yhat, y=res)) +
  geom_point(alpha=0.5, shape=16, fill="#002676", color="#002676", size=2)+
  geom_smooth(method=lm, se=FALSE, color="#FC9313")+
  labs(title="Residual Plot",
       x="Predicted Values", y = "Residual")+
  theme_classic(base_size = 20)
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



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

Answer: White's test rejects the null hypothesis of homoskedasticity, with a statistic of 144 and a p-value much smaller than 0.001.

```
skedastic::white(model_1,
                 interactions = T)
```

```
## # A tibble: 1 x 5
##   statistic  p.value parameter method       alternative
##       <dbl>    <dbl>     <dbl> <chr>        <chr>
## 1      171. 5.85e-26        20 White's Test greater
```

6. Using the felm package show regression output using the white robust standard errors. Did the coefficients change? Did the standard errors on the coefficients change? Did any of your t-statistics change? Did your F-Statistic Change?

Answer: Neither the coefficients or standard errors changed much. As a result, the t-statistics didn't really change either. The F-statistics are virtually identical.

```
model_2 <- felm( song_popularity ~ song_duration_ms + acousticness + danceability + energy + tempo, data
summary(model_2,robust = T)
```

```
##
## Call:
##    felm(formula = song_popularity ~ song_duration_ms + acousticness +      danceability + energy + t
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -60.961 -12.527   2.834  15.695  47.799
```

```
##
## Coefficients:
##                    Estimate Robust s.e t value Pr(>|t|)
## (Intercept)       5.363e+01  1.498e+00  35.811  < 2e-16 ***
## song_duration_ms -4.246e-06  2.740e-06  -1.550   0.1212
## acousticness     -7.344e+00  7.727e-01  -9.505  < 2e-16 ***
## danceability      1.194e+01  1.071e+00  11.142  < 2e-16 ***
## energy           -6.519e+00  1.021e+00  -6.383 1.78e-10 ***
## tempo            -9.733e-03  5.672e-03  -1.716   0.0862 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 21.71 on 17829 degrees of freedom
## Multiple R-squared(full model): 0.01573   Adjusted R-squared: 0.01546
## Multiple R-squared(proj model): 0.01573   Adjusted R-squared: 0.01546
## F-statistic(full model, *iid*):   57 on 5 and 17829 DF, p-value: < 2.2e-16
## F-statistic(proj model): 56.82 on 5 and 17829 DF, p-value: < 2.2e-16
```

```
summary(model_2,robust = F)
```

```
##
## Call:
##    felm(formula = song_popularity ~ song_duration_ms + acousticness +      danceability + energy + te
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -60.961 -12.527   2.834  15.695  47.799
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)       5.363e+01  1.530e+00  35.048  < 2e-16 ***
## song_duration_ms -4.246e-06  2.787e-06  -1.524   0.1276
## acousticness     -7.344e+00  7.718e-01  -9.516  < 2e-16 ***
## danceability      1.194e+01  1.081e+00  11.044  < 2e-16 ***
## energy           -6.519e+00  1.024e+00  -6.363 2.03e-10 ***
## tempo            -9.733e-03  5.800e-03  -1.678   0.0933 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 21.71 on 17829 degrees of freedom
## Multiple R-squared(full model): 0.01573   Adjusted R-squared: 0.01546
## Multiple R-squared(proj model): 0.01573   Adjusted R-squared: 0.01546
## F-statistic(full model):   57 on 5 and 17829 DF, p-value: < 2.2e-16
## F-statistic(proj model):    57 on 5 and 17829 DF, p-value: < 2.2e-16
```

7. 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
# You could do the following by hand, but I am a bit lazy. Generate the squares and interactions.
# Put rhs variables in a frame
songsw <- songs |> dplyr::select(song_duration_ms, acousticness, danceability, energy, tempo)
whites <- make_sq_inter(data_frame = songsw, is_square = TRUE, is_inter = TRUE, keep_marginal = TRUE)
whites$dep <- (songs$res)^2
white_t_manual <- lm(dep ~  . , data=whites)
white_r2 <- summary(white_t_manual)$r.squared
# Calculate the White test statistic
white_stat <- nrow(whites) * white_r2
white_stat
```

```
## [1] 170.8797
```

```r
# Calculate the p-value
pchisq(q = white_stat, df = 20, lower.tail = F)
```

```
## [1] 5.845471e-26
```

```r
# Matches what we got above!
```