Dancebility of a Song

MA 575 Linear Models Final Project, Spring 2022

Duc Anh V Bui

U86632708

Theme: Entertainment

Electronic Submission Date: 5/4/2022

Table of Contents

Introduction

Spotify has been around for years in the music industry where it allows people to upload their products. Here, I will look for the danceability of songs by using Spotify API is extract various features of the songs including parameters like acousticness, energy, instrumentalness, liveness, loudness, speechiness, tempo, key, and valence. To understand the data deeper, I will split the data into 2 parts by 2 genres: Pop and Classic Rock music.

Throughout this project, I must choose an alpha level for all the hypothesis tests. For this case, I will choose an alpha level of 0.05. The reason for choosing this alpha is because that smaller the alpha level, the less likely the null hypothesis is rejected. However, if the alpha is too small, it leads to Type II error; if the alpha is too big, it creates Type I error. So, an alpha level of 0.05 is a perfect balance between Type I and II errors.

Univariate Data Analysis – Mean Testing

As mentioned above, I want to understand more about the Danceability of songs, so Danceability will be used as dependent variable.

Let the danceability of songs being measured [0, 100] with 0 as the least danceable and 100 as the most danceable. I believe that on average, songs would have the danceability of 60 because in my opinion, most songs that I have listened to are quite "danceable" so using an average of 60 is reasonable.

Data set:

Spotify-2000

Index	Title	Artist	Top Genre	Year	Danceability	Tempo	Energy	Loudness	Liveness	Valence	Length	Acousticness	Speechiness	Popularity
1	Sunrise	Norah Jones	adult standards	2004	53	157	30	-14	11	68	201	94	3	71
2	Black Night	Deep Purple	album rock	2000	50	135	79	-11	17	81	207	17	7	39
3	Clint Eastwood	Gorillaz	alternative hip hop	2001	66	168	69	-9	7	52	341	2	17	69
4	The Pretender	Foo Fighters	alternative metal	2007	43	173	96	-4	3	37	269	0	4	76
5	Waitin' On A Sunny Day	Bruce Springsteen	classic rock	2002	58	106	82	-5	10	87	256	1	3	59
6	The Road Ahead (Miles Of The Unknown)	City To City	alternative pop rock	2004	54	99	46	-9	14	14	247	0	2	45
7	She Will Be Loved	Maroon 5	pop	2002	71	102	71	-6	13	54	257	6	3	74
8	Knights of Cydonia	Muse	modern rock	2006	37	137	96	-5	12	21	366	0	14	69
9	Mr. Brightside	The Killers	modern rock	2004	36	148	92	-4	10	23	223	0	8	77
10	Without Me	Eminem	detroit hip hop	2002	91	112	67	-3	24	66	290	0	7	82
11	Love Me Tender	Elvis Presley	adult standards	2002	44	109	5	-16	11	31	162	88	4	49
12	Seven Nation Army	The White Stripes	alternative rock	2003	74	124	46	-8	26	32	232	1	8	74
13	Als Het Golft	De Dijk	dutch indie	2000	54	102	88	-6	53	59	214	2	3	34
14	I'm going home	Ten Years After	album rock	2005	38	117	93	-2	81	40	639	18	10	26
15	Fluorescent Adolescent	Arctic Monkeys	garage rock	2007	65	112	81	-5	14	82	173	0	3	66
16	Zonder Jou	Paul de Leeuw	dutch cabaret	2006	42	133	42	-10	16	25	236	84	4	48
17	Speed of Sound	Coldplay	permanent wave	2005	52	123	90	-7	7	36	288	0	6	69
18	Uninvited	Alanis Morissette	alternative rock	2005	38	127	54	-5	9	19	276	2	3	57
19	Music	John Miles	classic uk pop	2004	27	87	31	-13	63	12	352	1	3	46
20	Cry Me a River	Justin Timberlake	dance pop	2002	62	74	65	-7	10	56	288	57	18	74

Thanks to Kaggel, I was able to get this data measuring the danceability of Spotify songs. The dataset includes a total of 1990 songs.

For more information, here's the url:

https://www.kaggle.com/code/mithalishenoy/music-trends-throughout-the-years

where I downloaded the dataset.

For how the author generated the dataset, he referred to another website called:

http://sortyourmusic.playlistmachinery.com/

to generated top 2000 songs with the according attributes from Spotify.

R- code:

```r
#import library and help function
library(dplyr)
source("nemolm2.R")

#import and clean data
data <- read.csv("project/Spotify-2000.csv")
#drop unneccessary for this project
data <- data[, !(colnames(data) %in% c("Index","Title","Popularity","Length","Popularity"))]

#Project Part 1

keep <- c("Danceability","Tempo","Valence","Artist","Top.Genre")
df <- data[keep]

Sy <- df$Danceability
```

Figure: Import library, helper function Clean data and define Y variable

Hypothesis testing:

$$H0: \mu = 60$$

$$H1: \mu \neq 60$$

For this test, I'm using two-tailed t-test to test for the mean = 60 with alpha = 0.05.

Assume dancebility data has a normal distribution.

Let random variable Y is dancebility => $Y \sim N(\mu, \sigma^2)$

Y is randomly selected

Because length(Y) = 1990, we can consider this dataset is large.

```r
#Degree of freedom
v <- n-1
#t critical val
tcrit <- qt(alpha/2, df=v, lower.tail = F)
#mean of danceability
ybar <- mean(Sy)
#claim average
mu0 <- 60
#standard deviation, standard error, and margin error of danceability
sy <- sd(Sy)
SE <- sy/sqrt(n)
eps <- tcrit*SE
#t stat
tstat <- (ybar - mu0)/SE
#p-value
pvalA <- 2*pt(-abs(tstat),df=v, lower.tail = T)
#Confidence Interval
CIL_A <- ybar - eps
CIU_A <- ybar + eps
#table summary
metric_name_A <- c("CI Lower","CI Upper", "Claim mu", "T.stat","T.crit", "p-value","alpha value")
metric_val_A <- c(CIL_A, CIU_A, mu0, tstat, tcrit, pvalA, alpha)
metric_summary_A <- data.frame(metric_name_A, metric_val_A)
```

Figure: R code

```
> metric_summary_A
  metric_name_A  metric_val_A
1      CI Lower  5.254345e+01
2      CI Upper  5.389373e+01
3      Claim mu  6.000000e+01
4        T.stat -1.969874e+01
5        T.crit  1.961157e+00
6       p-value  4.666233e-79
7   alpha value  5.000000e-02
```

Figure: Metric summary for data mean against proposed mean

Because the p-value < alpha value ($4.666e^{-79} < 0.05$), we reject the null hypothesis. This means that I am 95% confident that mean of danceability is not 60 as predicted as there's only a 5% change that the null hypothesis is true.

Because the data is generated by the top 2000 songs from 1940 until 2019 on Spotify, it may hold a certain degree of biasness since their the top song. However with over 2000 songs as data, I do not think it will affect too much when comparing to population values.

Univariate Data Analysis – Standard Deviation Testing

Considering the standard deviation, I would want the standard deviation to be as small as possible which means I'm trying to test that whether the data is concentrated around the mean, in this case the danceability level. Even then I think a standard deviation = 10 is reasonable because a lot of songs are not meant to be danced on but rather to feel the meanings behind the lyrics.
For this hypothesis testing, since I'm testing its standard deviation, I want to use two-tailed chi-squared test.

Hypothesis testing:

H0: $\sigma = 10$

H1: $\sigma \neq 10$

Using Chi-squared test.

```
> metric_summary_B
  metric_name_B  metric_val_B
1      CI Lower  1.489434e+01
2      CI Upper  1.584965e+01
3      Claim sd  1.000000e+01
4      Chi.stat  4.690839e+03
5      Chi.crit  2.114501e+03
6       p-value 6.885697e-219
7   alpha value  5.000000e-02
```

Figure: metric values for data standard deviation equal to proposed standard deviation

Because p-value < alpha ($4.31*10^{-283} < 0.05$), we reject null hypothesis. Which means I am 95% confident that the standard deviation is not equal to 10 as predicted, as there is only a 5% chance that the null hypothesis is true.

Because the data is generated by the top 2000 songs from 1940 until 2019 on Spotify, it may holds a certain degree of biasness since their the top song. However with over 2000 songs as data, I do not think it will affect too much when comparing to population values.

```r
#Part B
# H0: sd = 0.1
# H1: sd <> 0.1
# claim standard deviation
sd0 <- 10

#chi crit val
chicrit <- qchisq(alpha/2, df = v, lower.tail = F)

#chi stat
chistat <- (v)*((sy/sd0)^2)
#p value
pvalB <- pchisq(q=chistat, df = v, lower.tail = F)
#Confidence Interval
CIL_B <- sqrt((v*sy^2)/qchisq(alpha/2, df = v, lower.tail = F))
CIU_B <- sqrt((v*sy^2)/qchisq(1-alpha/2, df = v, lower.tail = F))

#Table summary
metric_name_B <- c("CI Lower","CI Upper", "Claim sd", "Chi.stat","Chi.crit", "p-value","alpha value")
metric_val_B <- c(CIL_B, CIU_B, sd0, chistat, chicrit, pvalB, alpha)
metric_summary_B <- data.frame(metric_name_B, metric_val_B)
```

Figure: R code

Normality of random variables

For this data, I choose tempo and valence as the two numerical random variables. The data sets are included in the same picture as part I.

Description of the 2 variables of I chose:

- Tempo: The overall estimated tempo of track in beats per minute (BPM).
- Valence: A measure from [0, 100] describing the musical positiveness conveys by the song. Tracks with high valence sound more positive (e.g happy, cheerful, euphoric), while tracks with low valence sound more negative (e.g sad, depressed, angry).

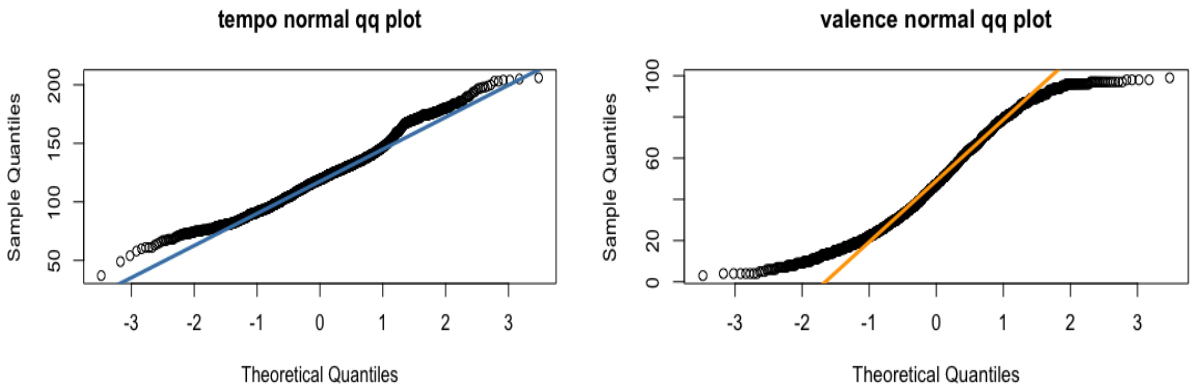


Figure: NQQ plots for Tempo and Valence variables

```
> #correlation coefficient for QQ plot tempo
> cor(V1$x,V1$y)
[1] 0.9910652
> #correlation coefficient for QQ plot valence
> cor(V2$x,V2$y)
[1] 0.9819298
```

Figure: Correlation coefficient for QQ plot of Tempo and Valence

As seen in the images above for NQQ plots of valence and tempo in respectively, we can see clearly that both variable have almost normal distribution with some outliers with tempo have a more normal distribution than valence.

With further investigation from finding the coefficient correlation, the distribution of each random variable is explained with the coefficient listed above in that the higher the correlation coefficient, the more its data follows a normal distribution, the better for my future model.

```
#Part C
#define random variables
X1 <- df$Tempo
X2 <- df$Valence
#Q-Q plot for tempo
V1 <- qqnorm(X1, main = "tempo normal qq plot")
qqline(X1,col="steel blue",lwd=3)
#Q-Q plot for valence
V2 <- qqnorm(X2, main ="valence normal qq plot")
qqline(X2,col="orange", lwd=3)
#correlation coefficient for QQ plot tempo
cor(V1$x,V1$y)
#correlation coefficient for QQ plot valence
cor(V2$x,V2$y)
```

Figure: R code

Parameter Comparisons for Means

Hear, I'm interested in comparing the mean between the danceability of pop genre and classic rock genre
Hypothesis:

H0: $\mu_1 = \mu_2$

H1: $\mu_1 =/= \mu_2$

```
> metric_summary_D
        metric_name_D metric_val_D
1           CI Lower 1.111209e+01
2           CI Upper 1.246237e+01
3 Claim mu difference 0.000000e+00
4             T.stat 3.402344e+00
5             T.crit 1.986086e+00
6            p-value 9.906244e-04
7        alpha value 5.000000e-02
```

Figure: Metric values for equal means testing.

Because the p-value < alpha value ($9.906e^{-04} < 0.05$), we reject the null hypothesis ($\mu_1 = \mu_2$), which means that the mean of dancebility between Pop and Classic Rock genre are not the same. Because of the alpha that we chose, we are 95% confident to reject the Null hypothesis because there's only a 5% change the Null hypothesis is true.

Because the data is generated by the top 2000 songs from 1940 until 2019 on Spotify, it may hold a certain degree of biasness since they're the top songs. However, with over 60 songs as data for each set of genres, I do not think it will affect too much when comparing to population values.

```r
pop_gen <- filter(df, Top.Genre =="pop")$Danceability
c.rock_gen <- filter(df, Top.Genre =="classic rock")$Danceability
popbar <- mean(pop_gen)
c.rockbar <- mean(c.rock_gen)
std_pop <- sd(pop_gen)
std_c.rock <- sd(c.rock_gen)
n1 <- length(pop_gen)
n2 <- length(c.rock_gen)

#degree of freedom with equal variance assumption
v_D <- (n1 + n2)-2

#t-crit part D
tcrit_D <- qt(alpha/2, df = v_D,lower.tail = F)

#Standard error
SE_12 <- sqrt(std_pop^2/n1 + std_c.rock^2/n2)
#t stat for part D
tstat_D <- (popbar - c.rockbar)/SE_12

pval_D <- 2*pt(-abs(tstat_D),df=v_D, lower.tail = T)

#margin of error
eps_D <- tstat_D*SE_12

#Confidence Interval
CIL_D <- (popbar-c.rockbar)-eps
CIU_D <- (popbar-c.rockbar)+eps

#Table summary
metric_name_D <- c("CI Lower","CI Upper", "Claim mu difference", "T.stat","T.crit", "p-value","alpha value")
metric_val_D <- c(CIL_D, CIU_D, 0, tstat_D, tcrit_D, pval_D, alpha)
metric_summary_D <- data.frame(metric_name_D, metric_val_D)
```

Figure: R code

Parameter Comparisons for Variances

Because the question is asking for hypothesis test for equal variances, the appropriate test here would be F test

Hypothesis:

$$H0: \sigma_1^2 = \sigma_2^2$$

$$H1: \sigma_1^2 = \sigma_2^2$$

```
> metric_summary_E
          metric_name_E metric_val_E
1             CI Lower 4.861036e+04
2             CI Upper 1.566802e+05
3 Claim var difference 0.000000e+00
4               F.stat 8.727131e+04
5               F.crit 5.377254e+00
6              p-value 1.000000e+00
7          alpha value 5.000000e-02
```

Figure: metrics for equal variance assumption for Pop and Classic Rock genre

Base of the table, with p-value > alpha value (1.00 > 0.05) , we fail to reject the null hypothesis, which means that it is statistically insignificant. Because of the alpha that we chose, we are 95% confident to reject the Null hypothesis because there's only a 5% change the Null hypothesis is true. So, the assumption of equal variance is held for alpha level of 0.05.

Because the data is generated by the top 2000 songs from 1940 until 2019 on Spotify, it may hold a certain degree of biasness since they're the top songs. However, with over 60 songs as data for each set of genres, I do not think it will affect too much when comparing to population values.

```r
rpop_c.rock <- cor(pop_gen,c.rock_gen)
SSE <- std_c.rock^2*(n1-1)*(1-rpop_c.rock^2)
MSE <- SSE/(n-2)
SST <- std_c.rock^2*(n-1)
MST <- SST/(n-1)
SSM <- SST - SSE
MSM <- SSM/1
# F stat value
Fstat <- MSM/MSE
# F crit value
Fcrit <- qf(alpha/2, 1, n2-2, lower.tail = F)
Fcrit2 <- qf(1-alpha/2, 1, n2-2, lower.tail = F)
# p val
pval_E <- pf(Fstat, 1, n2-2)
#Confidence Interval
CIL_E <- Fstat * qf(alpha/2, n2-1, n1-1)
CIU_E <- Fstat * 1/qf(alpha/2, n1-1, n2-1)

#Table summary
metric_name_E <- c("CI Lower","CI Upper", "Claim var difference", "F.stat","F.crit", "p-value","alpha value")
metric_val_E <- c(CIL_E, CIU_E, 0, Fstat, Fcrit, pval_E, alpha)
metric_summary_E <- data.frame(metric_name_E, metric_val_E)
```

Figure: R code

Introduction to Modelling the data

Next, after carefully examine the data that I have, I want to predict Danceability with the attributes above. To do this, I must fit models to the data and examine them. Upon trying to predict the Danceability of given song, I must fit appropriate models to the data, then compare which model is the best by comparing their MSE (which has the minimum MSE can be considered the best model).

For simplicity, I will use a helper function called "nemolm2.R" for all the statistical results of the models generated in the rest of this project. The calculations of these full be provided throughout the paper.

```
nemolm2 <- function(Y, Xk, ridge=0){
  #ridge = lambda >= 0 (defaulting to 0 results in OLS)
  n <- length(Y)
  v1s <- rep(1, n)
  X <- cbind(v1s,Xk)
  p <- dim(X)[2]-1

  # Ridge Regression If-Statement
  if(ridge != 0){
    lambda = ridge
    S <- svd(t(X)%*%X + lambda^2*diag(p+1))
  }
  else{
    S <- svd(t(X)%*%X)
  }

  # Singular Value Decomposition
  U <- S$u
  D <- diag(S$d)
  V <- S$v

  # Condition Number for XtX
  kappa <- max(S$d)/min(S$d)

  betahat <- V%*%solve(D)%*%t(U)%*%t(X)%*%Y
  Yhat <- X%*%betahat
  H <- X%*%V%*%solve(D)%*%t(U)%*%t(X)
  lv <- diag(H)

  res <- Y - Yhat

  SSE <- sum(res^2)
  MSE <- SSE/(n-p-1)
  SST <- sd(Y)^2*(n-1)
  MST <- SST/(n-1)
  SSM <- SST - SSE
  MSM <- SSM/p

  sres <- res/(sqrt(MSE)*sqrt(1-lv))
  SEbetahat <- sqrt(MSE)*sqrt(diag(V%*%solve(D)%*%t(U)))

  Fstat <- MSM/MSE
  pval <- pf(Fstat, p, n-p-1, lower.tail = F)

  r2 <- 1-SSE/SST
  r2adj <- 1- MSE/MST

  results <- list("predicted" = Yhat,
                  "residual" = res,
                  "sres" = sres,
                  "condition" = kappa,
                  "leverage"= lv,
                  "sse" = SSE,
                  "mse" = MSE,
                  "ssm" = SSM,
                  "msm" = MSM,
                  "pval" = pval,
                  "betahat" = betahat,
                  "SEbetahat" = SEbetahat,
                  "r2" = r2,
                  "r2adj" = r2adj
  )

  return(results)
}
```

Figure: R code for helper function nemolm2.R

```
#Phase 2
# clean data a bit more
data <- data[, !(colnames(data) %in% c("Index","Title","Artist","Top.Genre","Popularity","Length","Popularity"))]
```

Figure: R code for cleaning the data

However, with 8 variables that could be used as predictors, first, I must create a correlation table with Danceability against the 8 variables.

Simple Linear Regression

For a Simple Linear Regression, I'm using "Valence" as the one explanatory variable to explain/predict the "Danceability" of a song.

Danceability = betahat0 + betahat1*Valence (1)

When generating the correlation table between "Danceability" (as y variable) and "Valence" (as x1 variable), it shows a correlation coefficient of 0.51456 (highest coefficient according to figure below). With this correlation coefficient, it shows that there's a certain degree of correlation between Valence and Danceability, so it is appropriate to fit a Simple Linear Regression to predict Danceability using Valence.

```
> corr_coef
                    Year Danceability       Tempo      Energy    Loudness    Liveness     Valence Acousticness Speechiness
Year          1.00000000   0.07749327  0.01256997  0.1472349  0.34376421  0.01901677 -0.16616312  -0.13294603  0.05409671
Danceability  0.07749327   1.00000000 -0.14060233  0.1396163  0.04423531 -0.10306258  0.51456376  -0.13576888  0.12522900
Tempo         0.01256997  -0.14060233  1.00000000  0.1566444  0.09292650  0.01625639  0.05965322  -0.12247181  0.08559821
Energy        0.14723485   0.13961627  0.15664444  1.0000000  0.73571088  0.17411770  0.40517478  -0.66515636  0.20586499
Loudness      0.34376421   0.04423531  0.09292650  0.7357109  1.00000000  0.09825705  0.14704112  -0.45163499  0.12508975
Liveness      0.01901677  -0.10306258  0.01625639  0.1741177  0.09825705  1.00000000  0.05066664  -0.04620551  0.09259447
Valence      -0.16616312   0.51456376  0.05965322  0.4051748  0.14704112  0.05066664  1.00000000  -0.23972907  0.10710188
Acousticness -0.13294603  -0.13576888 -0.12247181 -0.6651564 -0.45163499 -0.04620551 -0.23972907   1.00000000 -0.09825610
Speechiness   0.05409671   0.12522900  0.08559821  0.2058650  0.12508975  0.09259447  0.10710188  -0.09825610  1.00000000
```

Figure 2.1: Correlation Coefficient table of Danceability and Everything

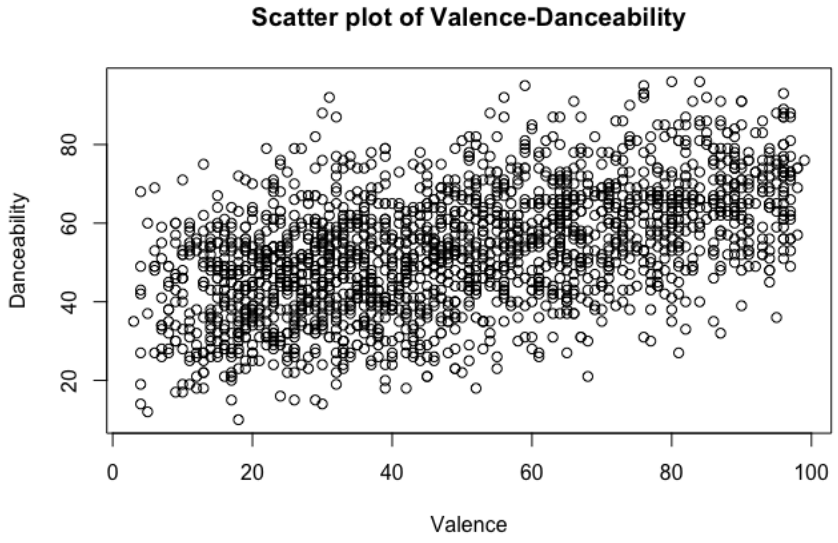


Figure 2.2: Scatter Plot of Danceability ~ Valence

Looking at the Figure 2.2, we can generally see a pattern between "Valence" and "Danceability", and it looks like a straight line.

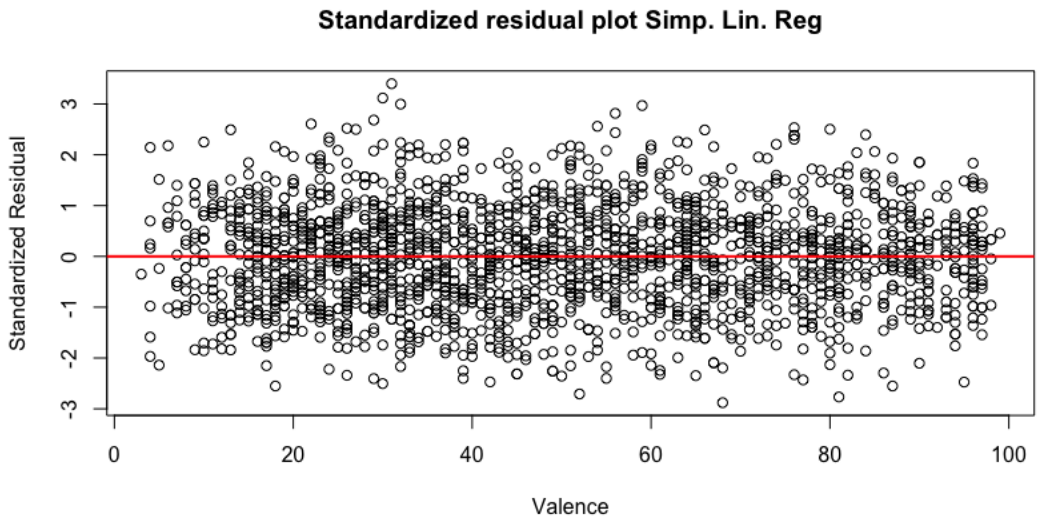


Figure 2.3: Standardized Residual Plot

```
> mod_A$betahat
             [,1]
[1,] 37.5119553
[2,]  0.3180355
```

Figure 2.4: Betahat for Simple Linear Regression

```
> mod_A$SEbetahat
[1] 0.65749838 0.01189489
```

Figure 2.5: Standard error of Betahat for Simple Linear Regression

Here, Betahat0 = 37.5120 and Betahat1 = 0.3180
With the betahat results from above and plug into the equation (1), we get

Danceability = 37.5120 + 0.3180*Valence (2)

```
> metric.sum_A
  metric.title    metric.val
1           r2  2.644869e-01
2  Adjusted r2  2.641170e-01
3      P-Value 8.545215e-135
```

Figure 2.6: R^2, Adjusted R^2, P-Value of Simple Linear Regression

Based on the result of figure 2.6:

- Adjusted R^2 is 0.26442. This means that the model of simple linear regression only explain about 26.44% of the data, which means there is a better model to fit the data.
- P-Value of 8.5452e-135 is much lower than the alpha (0.05) which shows that we reject the Null Hypothesis (Betahat1 = 0) and accept the Alternate Hypothesis (Betahat1 is not 0) , which generally means that we are 95% confidence that Betahat1 is statistically significant in this model.

```r
#part A: Simple linear Regression
df <- data[c("Danceability","Valence")]
y <- df$Danceability
x1 <- df$Valence

corr_table <- cor(df)

plot(x1,y, xlab = "Valence", ylab = "Danceability", main ="Scatter plot of Valence-Danceability")

#Simple linear with danceability and tempo
mod_A <- nemolm2(y,cbind(x1,x1^2))

#Standardized Resid
pA_sres <- mod_A$sres

plot(x1, pA_sres, xlab = "Valence", ylab ="Standardized Residual", main = "Standardized residual plot Simp. Lin. Reg")
abline(0,0, col = "red", lwd=2)

#betahat0 and betahat1
pA_betahat0 <- mod_A$betahat[1]
pA_betahat1 <- mod_A$betahat[2]

#SEbetahat0 and SEbetahat1
pA_SEbeta0 <- mod_A$SEbetahat[1]
pA_SEbeta1 <- mod_A$SEbetahat[2]

#r2, r2adj, p-val
pA_r2 <- mod_A$r2
pA_r2adj <- mod_A$r2adj
pA_pval <- mod_A$pval
metric.title <- c("r2","Adjusted r2", "P-Value")
metric.val <- c(pA_r2,pA_r2adj,pA_pval)
metric.sum_A <- data.frame(metric.title, metric.val)
```

Figure: R code for Simple Linear Regression

Simple Quadratic Regression

Proposed Formula:

When looking at the scatter plot from figure 2.2, a simple linear regression model might not be enough to predict the Danceability. Furthermore, figure 2.6 proposes that there is a better model to predict the Danceability.

$$\text{Danceability} = \text{betahat0} + \text{betahat1}*\text{Valence} + \text{betahat2}*\text{Valence}^2 \quad (3)$$

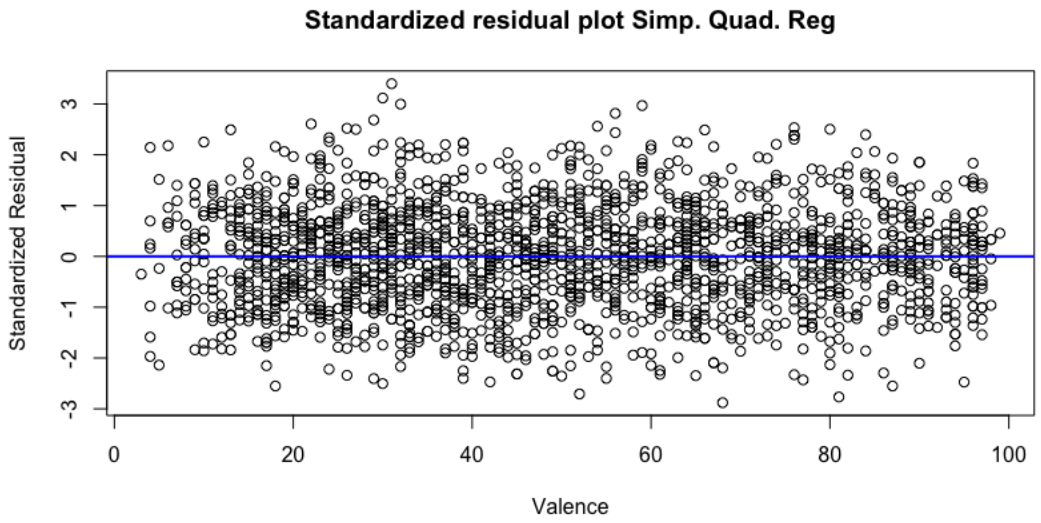


Figure 2.7: Standardized Residual Plot of Betahat for Simple Quadratic Regression.

```
> mod_B$betahat
              [,1]
[1,] 3.883070e+01
[2,] 2.519290e-01
[3,] 6.369098e-04
```

Figure 2.8: Betahat for Simple Quadratic Regression

```
> mod_B$SEbetahat
[1] 1.2278981579 0.0533307681 0.0005008818
```

Figure 2.9: Standard Error of betahat for Simple Quadratic Regression

Here, Betahat0 = 38.8307, Betahat1 = 0.2519, and Betahat2 = 0.000637
With the betahat results from above and plug into the equation (1), we get

$$\text{Danceability} = 44.2252 + 0.2519*\text{Valence} + 0.000637*\text{Valence}^2 \qquad (4)$$

```
> metric.sum_B
  metric.title  metric.val_B
1           r2  2.650850e-01
2  Adjusted r2  2.643452e-01
3      P-Value 1.278183e-133
```

Figure 2.10: R^2, Adjusted R^2, P-Value of Simple Quadratic Regression

Based on the result of figure 2.10:

- Adjusted R2 is 0.2643. This means that the model of simple Quadratic regression only explains about 26.43% of the data, which means there is a better model to fit the data.
- P-Value of $1.2782e^{-133}$ is lower than the alpha (0.05) which shows that we reject the Null Hypothesis (Betahat1 = 0) and accept the Alt Hypothesis (Betahat1 is not 0) , which generally means that we are 95% confidence that Betahat1 is statistically significant in this model.
- If there is nothing else known, except for Adjusted R^2, Simple Linear Regression shows a better result than Simple Quadratic Regression.

```r
#part B: Simple Quadratic Regression
mod_B <- nemolm2(y, cbind(x1,x1^2))
pB_sres <- mod_B$sres

#betahat0 and betahat1 and betahat2
pB_betahat0 <- mod_B$betahat[1]
pB_betahat1 <- mod_B$betahat[2]
pB_betahat2 <- mod_B$betahat[3]

#SEbetahat0 and SEbetahat1
pB_SEbeta0 <- mod_B$SEbetahat[1]
pB_SEbeta1 <- mod_B$SEbetahat[2]
pB_SEbeta2 <- mod_B$SEbetahat[3]

#r2, r2adj, p-val
pB_r2 <- mod_B$r2
pB_r2adj <- mod_B$r2adj
pB_pval <- mod_B$pval
metric.val_B <- c(pB_r2,pB_r2adj,pB_pval)
metric.sum_B <- data.frame(metric.title, metric.val_B)
plot(x1, pB_sres, xlab="Valence", ylab="Standardized Residual",main="Standardized residual plot Simp. Quad. Reg")
abline(0,0, col="blue", lwd=2)
```

Figure: R code for Simple Quadratic Regression

Multiple Linear Regression

Now, what if the Danceability of a song is determined not only by Valence, but other variables as well?

In this case, I choose Tempo and Energy based on the result of correlation coefficient table in figure 2.1.

Where Tempo is the positivity of the song (The higher the value, the higher the speed of the song) measured [0, 100].

Energy is the Energy of the song (The higher the value, the higher the energy of the song) measured [1, 100].

- ⇨ The equation now is:
  Danceability = Betahat0 + Betahat1*Valence + Betahat2*Tempo + Betahat3*Energy (5)

```
> anova.table
      anova.title          anova.valueM     anova.valueE     anova.valueT
1 Source of Error                 Model       Error Term            Total
2             DOF                     3             1986             1989
3   Sum of Square     139067.531266823 330016.380793479 469083.912060302
4     Mean Square     46355.8437556076 166.171390127633 235.839070920212
5           Fstat     278.964048624752               :)               :)
6         P-Value 4.38089442058307e-151               :)               :)
```

Figure 3.1: Anova Table for Multiple Linear Regression

VIF:

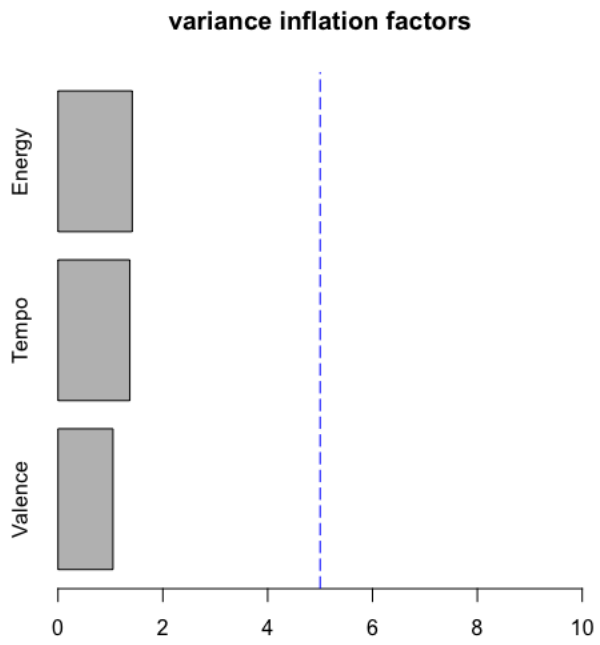


Figure 3.2: Variance Inflation Factor

Figure 3.2 shows whether the predictors used in equation (5) have multicollinearity. A VIF of 1 means the 3 variables are not correlated while VIF of 5 indicates high correlation. Because one of the assumptions made for multiple linear regression is that there is no multicollinearity so figure 3.2 shows that the predictors I chose have very low correlations.

Added Variable Plots:

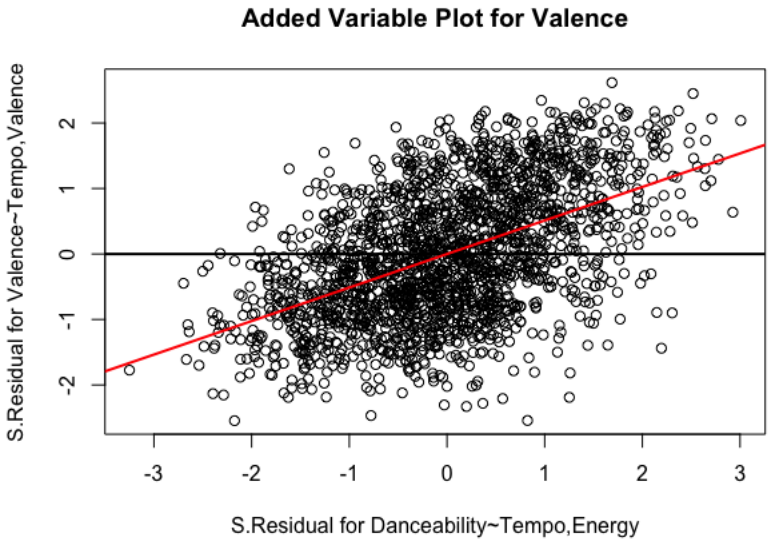


Figure 3.3: Added Variable Plot for Valence

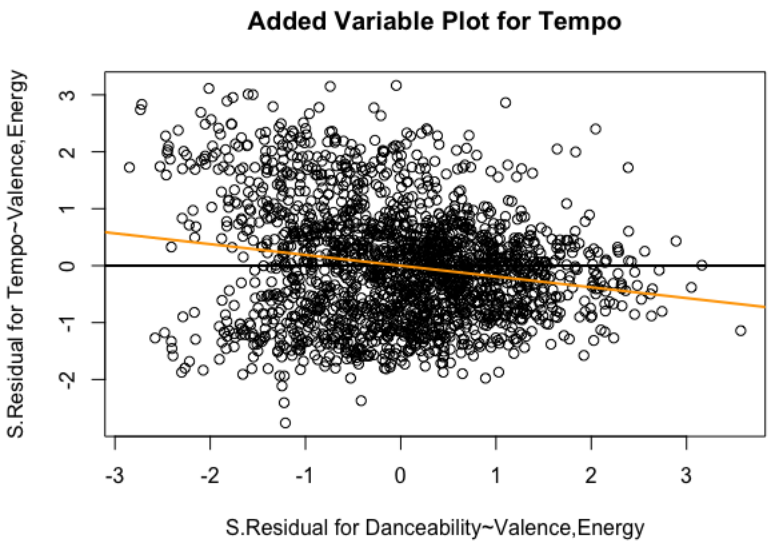


Figure 3.4: Added Variable Plot for Tempo

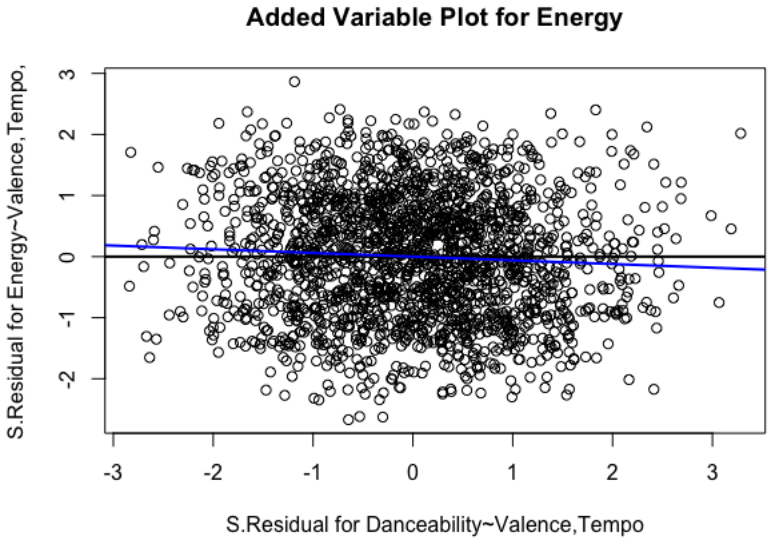


Figure 3.5: Added Variable Plot for Energy

Interpretation:

Comparing the figures 3.3, 3.4, 3.5 above, we can see that figure 3.5 (AVP for Energy) has a fit line closest to 0; while figure 3.3 (AVP for Valence) has the steepest fit line. All these are showing that based on the AVPs, we now have evidence that Energy is not statistically significant as the other 2 predictors. Further investigation (Partial F test) is needed to know whether the model is better fit without Energy as a predictor.

The reason to use Partial F-test is to test whether the proposed reduced model is a better fit than the full model that we have at (5).

```
> pval_partial_f
[1] 0.006221666
```
Figure 3.6: P-Value for partial F test

With the p-value in figure 3.6 above, and a chosen alpha level of 0.05, we reject the null hypothesis (betahat3 = 0). This means that we are 95% confident that betahat3 is not equal to 0, which means betahat3 is statistically significant.

- ⇨ This means that getting rid of Energy does not make the model (4) significantly better.

```
> mod_C$betahat
             [,1]
[1,] 49.66123475
[2,]  0.33809480
[3,] -0.08995219
[4,] -0.03895324
```
Figure 3.7: Betahat for Multiple Linear Regression

Here:

Betahat0 = 49.6612, Betahat1 = 0.3381, Betahat2 = -0.0899, and Betahat3 = -0.0389

With this result, we have a model:

$$\text{Danceability} = 49.6612 + 0.3381*\text{Valence} - 0.0899*\text{Tempo} - 0.0389*\text{Energy} \quad (6)$$

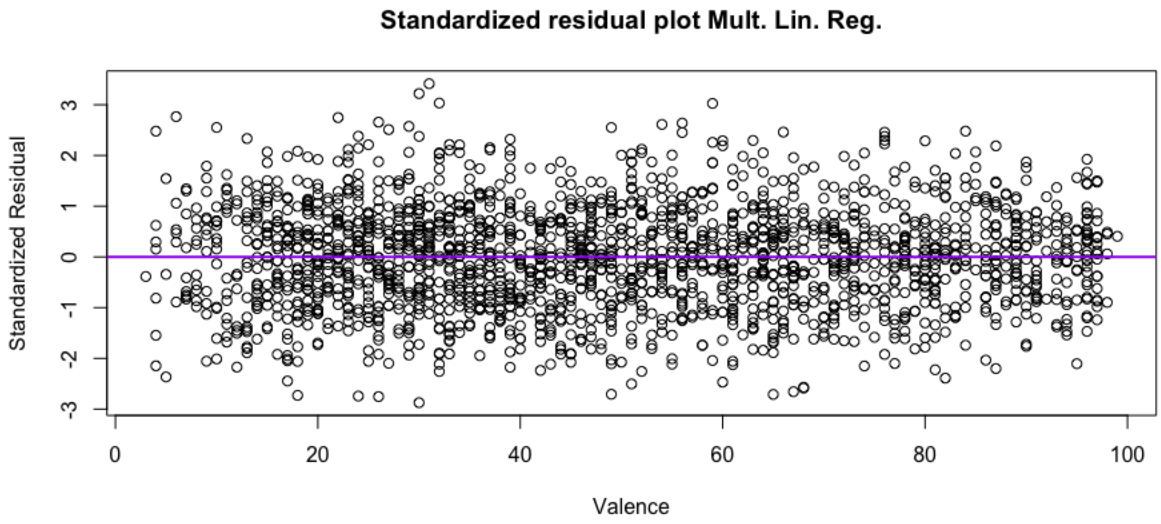


Figure 3.8: Standardized Residual Plot for Multiple Linear Regression

Because this is the Standardized Residual Plot against Valence, and there is no noticeable formation of data around the standardized residual=0 line, meaning they bounce randomly around that line. It is possible to say that Valence is a good predictor. However, since Valence is already in the proposed model at (6), this plot does not offer any other new information.

```r
#part C: Multiple Linear Regression
keep2 <- c("Danceability","Valence","Tempo","Energy")
df2 <- data[keep2]
corr_table2 <- cor(df2)
x2 <- df2$Tempo
x3 <- df2$Energy
mod_C <- nemolm2(y, cbind(x1,x2,x3))

n <- length(y)

SSE <- mod_C$sse
MSE <- mod_C$mse
SSM <- mod_C$ssm
MSM <- mod_C$msm
SST <- SSE + SSM
MST <- SST/(n-1)

pC_pval <- mod_C$pval
pC_Fstat <- MSM/MSE

#anova table
anova.title <- c("Source of Error", "DOF", "Sum of Square", "Mean Square", "Fstat", "P-Value")
anova.valueM <- c("Model", 3, SSM, MSM, pC_Fstat, pC_pval)
anova.valueE <- c("Error Term", n-4, SSE, MSE, ":)", ":)")
anova.valueT <- c("Total", n-1, SST,MST, ":)", ":)")
anova.table <- data.frame(anova.title,anova.valueM,anova.valueE,anova.valueT)
```

```r
#variance inflation factor, barplot
MYv123 <- nemolm2(y, cbind(x1,x2,x3), 0)
MYv12 <- nemolm2(y, cbind(x1,x2), 0)
MYv13 <- nemolm2(y, cbind(x1,x3), 0)
MYv23 <- nemolm2(y, cbind(x2,x3), 0)

M1v23 <- nemolm2(x1, cbind(x2,x3))
M2v13 <- nemolm2(x2, cbind(x1,x3))
M3v12 <- nemolm2(x3, cbind(x1,x2))
vif1 <- 1/(1-MYv23$r2)
vif2 <- 1/(1-MYv13$r2)
vif3 <- 1/(1-MYv12$r2)

vif <- c(vif1,vif2,vif3)

barplot(vif, horiz = T,
        main = "variance inflation factors",
        names.arg = c("Valence","Tempo","Energy"),
        xlim = c(0,10))
abline(v=5,col="blue",lty="longdash")

pC_sres <- mod_C$sres
plot(x1, pC_sres, xlab = "Valence", ylab ="Standardized Residual", main = "Standardized residual plot Mult. Lin. Reg.")
abline(0,0, col = "purple", lwd=2)
```

Figure: R code for Multiple Linear Regression

Time Series

I suspect that the danceability of songs can also be predicted with Time Series model, which means that at Danceability in 2010 can be predicted based on the Danceability in 2009.

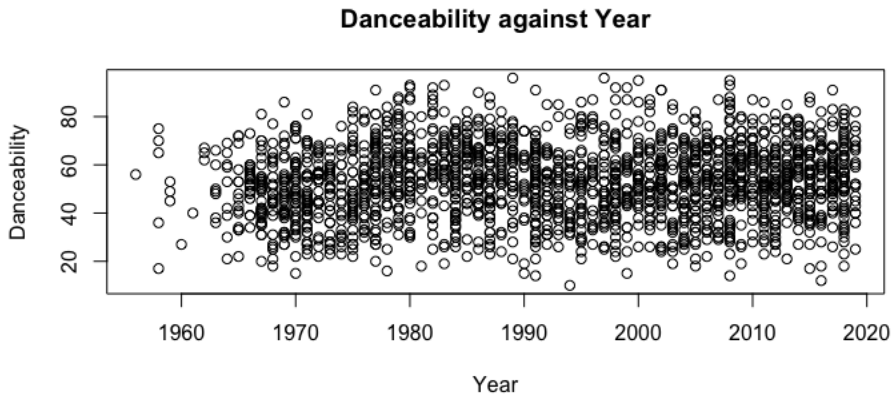


Figure 4.1: Scatter Plot of Danceability against Year

If I only base my prediction on the results in figure 4.1, there is nothing much to say as the data points are in categorical manners, with Year as the groups.

So, taking the mean of each year and plot them against year is a better approach towards fitting a time series model.

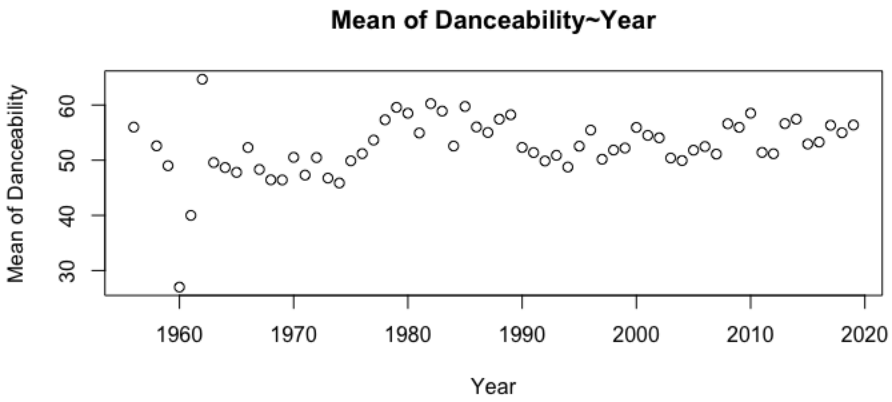


Figure 4.2: Mean of Danceability against Year

Looking at figure 4.2, we can see that from 1960 to 2020, the mean danceability of top songs produced has not changed much. However, there are some noticeable drops/increases in the mean danceability of songs through the year.
For example:

From 1960-1970, the danceability follows a huge decline danceability. This can be explained by during that time, the WW2 just ended so not a lot of artists may not wanted to produce very danceable songs.

Then, from 1970-1980, the trend increases significantly. This can be explained by rock music started becoming a trending genre.

From 1980-2020, there are some increases and decreases in danceability.

Looking closely at the year 1960, the data shows an outlier which pulls everything down, skewed the data significantly. It is appropriate to delete that point.

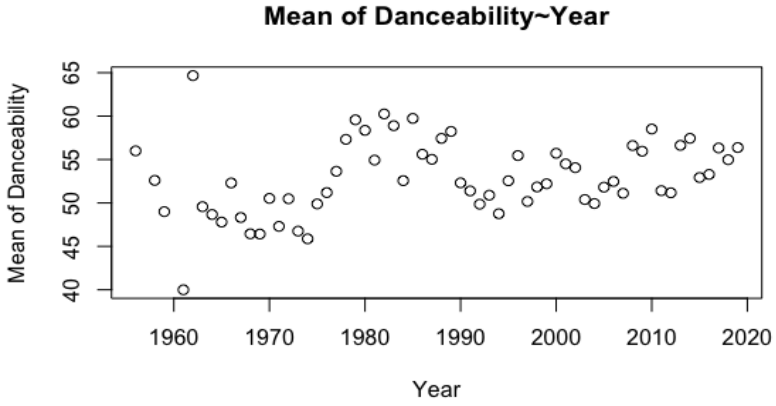


Figure 4.3: Mean of Standardized Danceability against Year after deleting outlier

Interpretation:

- When I count the number of noticeable peaks and troughs, there are a total of 4 points. For safety, I will generate a polynomial regression model with degree of 5 for the "tail" predicted data points to be closer to the real data point in figure 4.3 above.

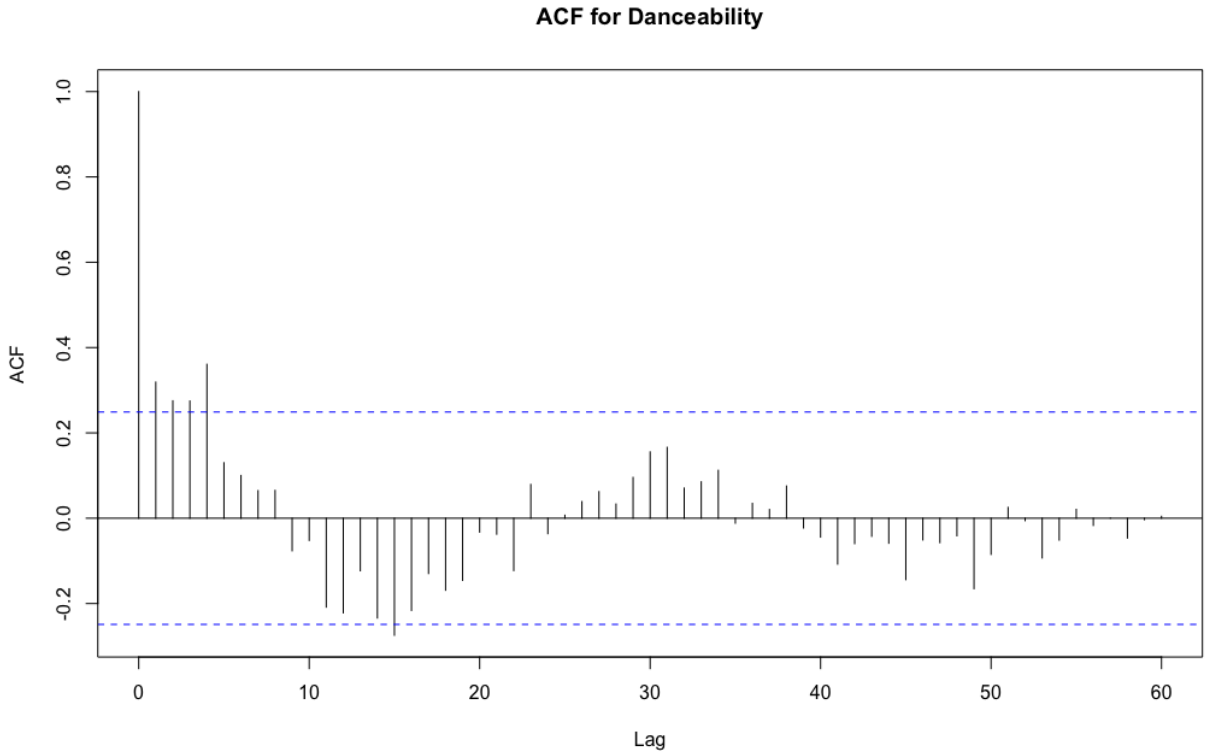


Figure 4.3: ACF for Danceability

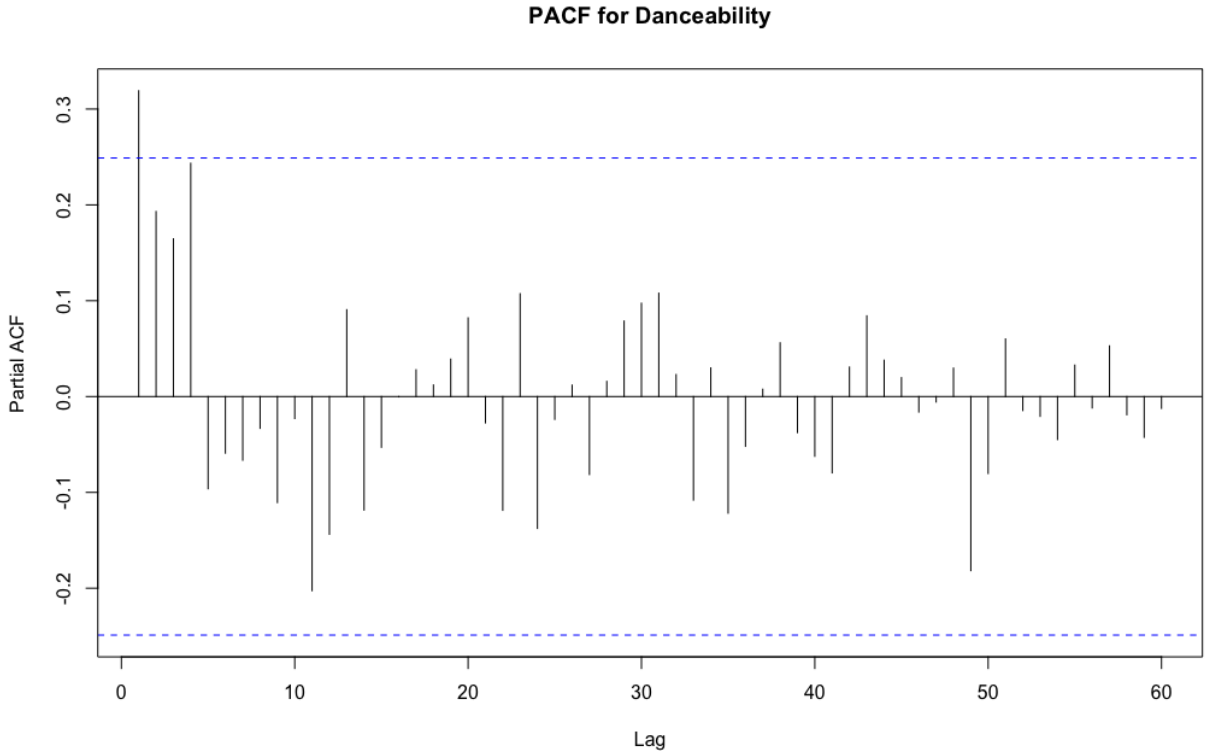


Figure 4.4: PACF for Danceability

If we look closely at figure 4.3, we can noticed that at lag = (2,3,4,15), the ACF value is above the blue dash line, which represents the significance of time series. This also shows that the data is stationary.

```
> mod_time$betahat
          [,1]
[1,] 54.798766
[2,] -3.452079
[3,] -4.951367
[4,]  6.546400
[5,]  1.667214
[6,] -1.784586
>
```

Figure 4.4: Polynomial Regression Betahat

With the betahat results from figure 4.4 above, I can derived an equation for prediction with a 5 degree Polynomial Regression model:

$$Danceability = 54.7988 - 3.4521*Year - 4.9514*Year^2 + 6.5464*Year^3 + 1.6672*Year^4 - 1.7845*Year^5 \quad (8)$$

```
> mod_time2$betahat
          [,1]
[1,] 52.94377
[2,]  1.27879
>
```

Figure 4.5: Linear Regression Betahat

With the betahat results from figure 4.4 above, I can derived an equation for prediction with a Linear Regression model:

$$Danceability = 52.94377 + 1.2788*Year \quad (9)$$

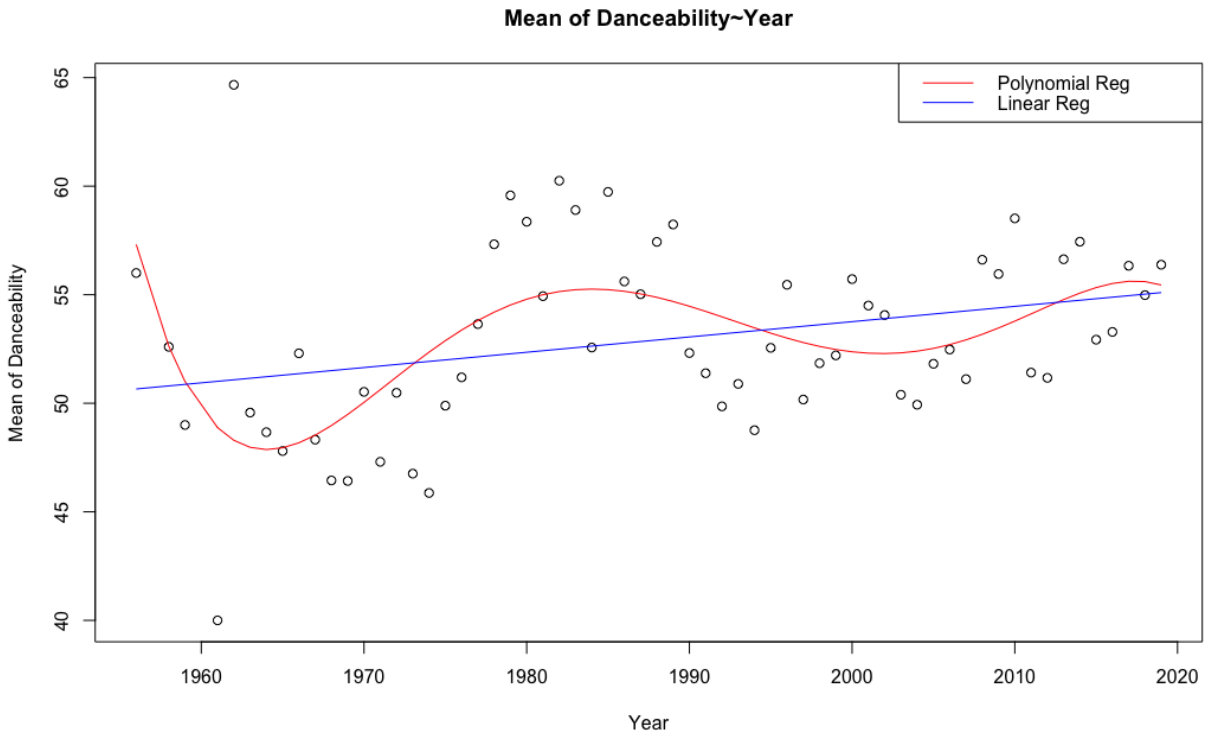


Figure 4.5: Scatter Plot of Mean of Standardized Danceability against Year

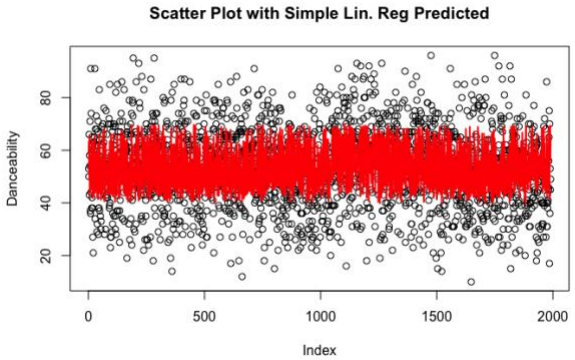
According to the blue line, it is easy to see that the trend of Danceability is increasing, even though slowly, whereas the red line indicates the stationary of the data.

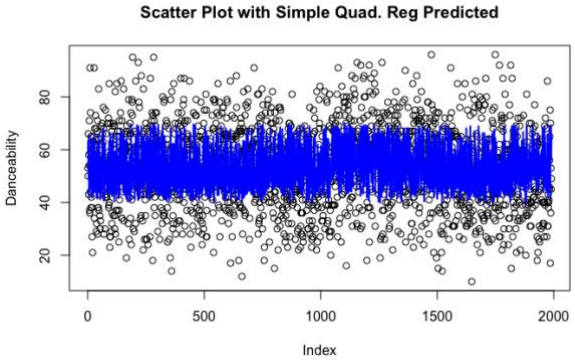
```r
#part 4: Time Series
keep3 <- c("Year", "Danceability")
df3 <- data[keep3]
df3 %>%
  plot(ylab= "Danceability", main="Danceability against Year")
# transformed the danceability into mean by each year
df4 <- df3 %>%
  group_by(Year) %>%
  summarize(mean=mean(Danceability))
# delete the outlier
df4 <- filter(df4, Year != "1960")
df4 %>%
  plot(xlab= "Year", ylab="Mean of Danceability",main="Mean of Danceability~Year")

# standardized the year variable
x_year <- df4$Year
y_mean <- df4$mean
x_year <- (x_year - mean(x_year))/sd(x_year)
# ACF, PACF
acf_plot <- acf(y_mean, lag=60, main="ACF for Danceability")
pacf_plot <- pacf(y_mean, lag=60, main="PACF for Danceability")

# Models for Polynomial and Linear Regression
mod_time <- nemolm2(y_mean, cbind(x_year, x_year^2,x_year^3,x_year^4,x_year^5))
mod_time2 <- nemolm2(y_mean, x_year)

df4 %>%
  plot(xlab= "Year", ylab="Mean of Danceability",main="Mean of Danceability~Year")
lines(df4$Year,mod_time$predicted, col="red", main="Polynomial Reg")
lines(df4$Year,mod_time2$predicted, col="blue")
legend("topright", legend=c("Polynomial Reg", "Linear Reg"),
       col=c("red", "blue"), lty = 1)
```

Figure: R code for Time Series

End note

My goal of this project is to propose a best model to predict the Danceability of a song if other mentioned attributes are known.





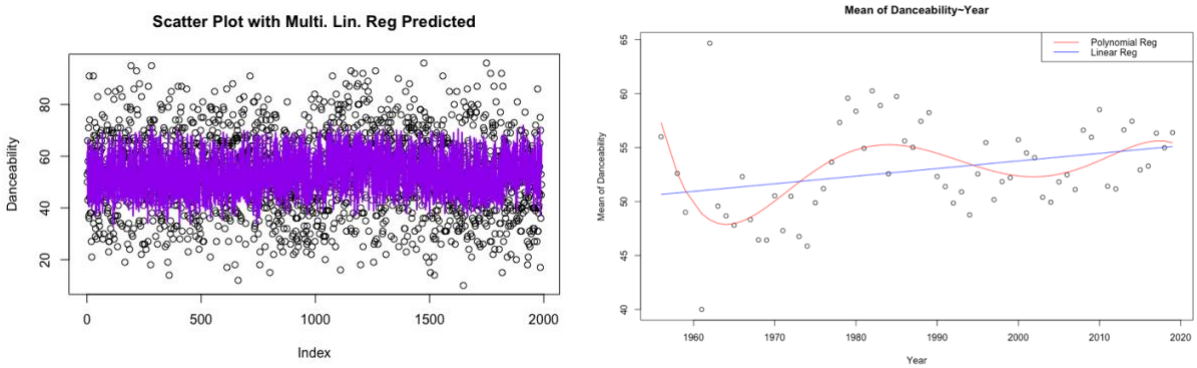


Figure: Scatter plots with Predictions from all the models mentioned above

As mentioned before, in order to compare which model is the best between all the models mentioned, we have to calculate the MSE, whichever has the min MSE indicate it is the best model.

In this case, I'm not going to compare the time series models as these models that I use are only to see the trend of the danceability and to predict the trend, not to predict the exact danceability using the attributes.

```
> metric.sum_final
             metric.reg metric.mse
1    Simple Linear Reg   173.4961
2 Simple Quadratic Reg   173.4961
3  Multiple Linear Reg   166.1714
```

Figure: MSE for the 3 types of Regression models.

From the results of the figure above, using Multiple Linear Regression with the mentioned attributes seem like the best model amongst the 3. However, to make sure that this model is an appropriate and the best one, I need to create many more proposed models with different combinations of attributes. One of the steps in proposing a model that has not been mentioned in this project is the Variable Selection step. This can be done using AIC's, BIC's, MLE, Ridge, or Lasso. The way I chose the predictor variables is based only on the correlation table between the dependent and independent variables.