# STAT - 427/627 Final Project

Parker Brotman, Christopher Hoffmann, Beau Swallow

5/4/2020

## Reading in the Data and Libraries

```
library(tidyverse)
library(e1071)
```

```
## Warning: package 'e1071' was built under R version 3.6.3
```

```
library(MASS)
library(readr)
library(leaps)
library(car)
library(ISLR)
library(pls)
```

```
## Warning: package 'pls' was built under R version 3.6.3
```

```
library(glmnet)
```

```
## Warning: package 'glmnet' was built under R version 3.6.3
```

```
top50 = read_csv("../data/top50cleaned.csv",
                 col_types = cols(Genre = col_factor()))
attach(top50)
```

## Introduction to the Data

The data we decided to use is called "Top 50 Spotify Songs - 2019" and cand be found **here**

The variables that are used from the dataset are described below:

- Popularity: Numerical rank of song
- Genre: Categorical variable of three categories (Hip Hop, Pop and Latin)
- Beats Per Minute: The tempo of the song

- Energy: Higher values indicate higher levels of energy from the song
- Dancibilty: Higher values indicate higher dancibility of song
- Loudness: Loudness of song measured in decibles
- Liveness: Likelihood song is a live recording
- Valence: The higher the value, the more positive the mood of the song
- Length: Duration of song
- Acousticness: The higher the value, the more acoustic the song is
- Speechiness: Higher values indicate more lyrics

## Predicting Genre with SVM

### Model Selection

First, let's determine which kernel and costs to use.

```
set.seed(1)
S1tuned = tune(svm,Genre ~ . -Genre - X1 - Track.Name - Artist.Name,data = top50,kernel='linear',ranges
```

```
set.seed(1)
S2tuned = tune(svm,Genre ~ . -Genre - X1 - Track.Name - Artist.Name,data = top50,kernel='polynomial',ra
```

```
set.seed(1)
S3tuned = tune(svm,Genre ~ . -Genre - X1 - Track.Name - Artist.Name,data = top50,kernel='radial',ranges
```

```
set.seed(1)
S4tuned = tune(svm,Genre ~ . -Genre - X1 - Track.Name - Artist.Name,data = top50,kernel='sigmoid',range
```

```
summary(S1tuned)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##   0.1
##
## - best performance: 0.28
##
## - Detailed performance results:
##    cost error dispersion
## 1 1e-03  0.50  0.1943651
```

```
## 2 1e-02  0.50  0.1943651
## 3 1e-01  0.28  0.1932184
## 4 1e+00  0.30  0.1414214
## 5 1e+01  0.32  0.1686548
## 6 1e+02  0.30  0.1699673
## 7 1e+03  0.30  0.1699673
```

```r
summary(S2tuned)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##    10
##
## - best performance: 0.34
##
## - Detailed performance results:
##    cost error dispersion
## 1 1e-03  0.50  0.1943651
## 2 1e-02  0.50  0.1943651
## 3 1e-01  0.50  0.1943651
## 4 1e+00  0.42  0.1135292
## 5 1e+01  0.34  0.1646545
## 6 1e+02  0.40  0.1333333
## 7 1e+03  0.40  0.1333333
```

```r
summary(S3tuned)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##     1
##
## - best performance: 0.28
##
## - Detailed performance results:
##    cost error dispersion
## 1 1e-03  0.50  0.1943651
## 2 1e-02  0.50  0.1943651
## 3 1e-01  0.50  0.1943651
## 4 1e+00  0.28  0.1032796
## 5 1e+01  0.34  0.1349897
## 6 1e+02  0.34  0.1349897
## 7 1e+03  0.34  0.1349897
```

```r
summary(S4tuned)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##     1
##
## - best performance: 0.3
##
## - Detailed performance results:
##    cost error dispersion
## 1 1e-03  0.50  0.1943651
## 2 1e-02  0.50  0.1943651
## 3 1e-01  0.50  0.1943651
## 4 1e+00  0.30  0.1699673
## 5 1e+01  0.38  0.1751190
## 6 1e+02  0.42  0.1751190
## 7 1e+03  0.44  0.1837873
```

The two most promising models appear to be the Linear kernel with cost = .1 and the Radial kernel with cost = 1. Now, let's evaluate the predictive accuracy of these two models.

### Evaluating Predictive Accuracy

This is a small dataset, so rather than splitting the data into training/testing, let's use LOOCV to determine predictive accuracy.

```r
n = nrow(top50)
preds = c()
for (i in 1:n) {
  d.train = top50[-i,]
  d.test = top50[i,]
  s.temp = svm(Genre ~ . -Genre - X1 - Track.Name - Artist.Name,data = d.train,kernel='linear',cost=.1)
  yhat = predict(s.temp,d.test['Genre'])[i]
  preds = append(preds,yhat)
}
preds = recode_factor(preds,`1` ='Pop', `2`='Latin',`3`='Hip Hop')
table(Predicted= preds,Actual= Genre)
```

```
##          Actual
## Predicted Pop Latin Hip Hop
##   Pop      21     5       6
##   Latin     3     8       0
##   Hip Hop   1     0       6
```

```
svm.linear.MSPE = mean(preds == Genre)
svm.linear.MSPE
```

```
## [1] 0.7
```

```
n = nrow(top50)
preds = c()
for (i in 1:n) {
  d.train = top50[-i,]
  d.test = top50[i,]
  s.temp = svm(Genre ~ . -Genre - X1 - Track.Name - Artist.Name,data = d.train,kernel='radial',cost=1)
  yhat = predict(s.temp,d.test['Genre'])[i]
  preds = append(preds,yhat)
}
preds = recode_factor(preds,`1` ='Pop', `2`='Latin',`3`='Hip Hop')
table(Predicted= preds,Actual= Genre)
```

```
##           Actual
## Predicted Pop Latin Hip Hop
##   Pop      24     9       4
##   Latin     1     4       0
##   Hip Hop   0     0       8
```

```
mean(preds == Genre)
```

```
## [1] 0.72
```

```
svm.radial.MSPE = mean(preds == Genre)
svm.radial.MSPE
```

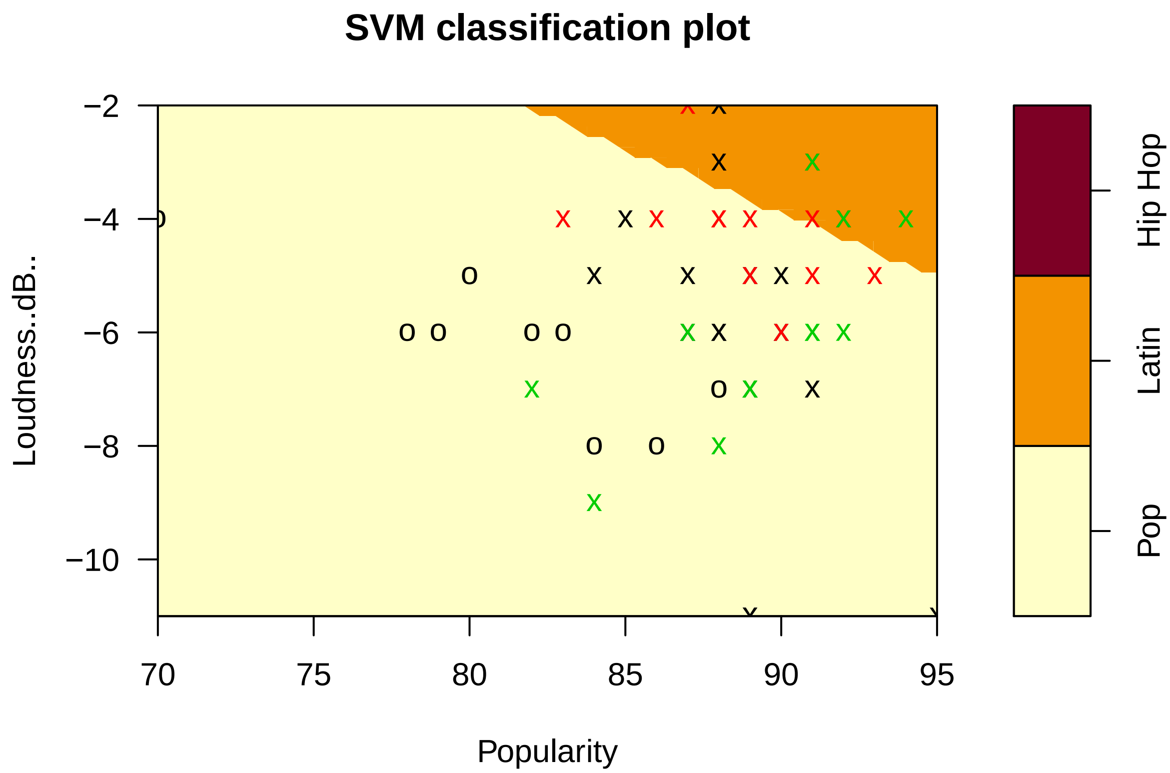
```
## [1] 0.72
```

The Linear kernal model has a classification rate of .7, while the Radial kernal model has a classification rate of .72. The first model is better at classifying Latin, but can predict Hip Hop with only 50% accuracy. Meanwhile, second model is better at classifying Pop and Hip Hop, but misclassifies Latin as Pop over 2/3 of the time.

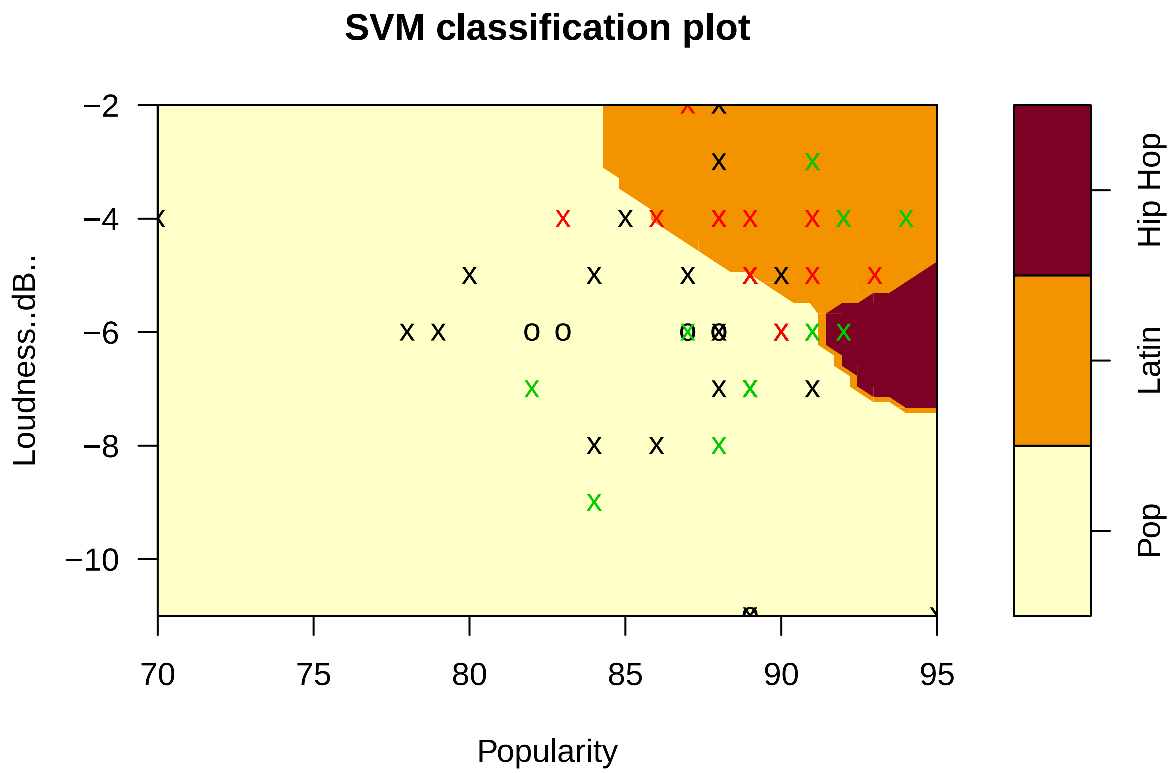
## Visualizing the Models

SVM is not a very interpretable model, and it is tough to visualize with more than two dimensions. However, let's refit SVM models using just Loudness and Popularity, and then visualize them.

```
plot(svm(Genre ~ Loudness..dB.. + Popularity,data = top50,kernel='linear',cost=.1),top50,Loudness..dB..
```




```
plot(svm(Genre ~ Loudness..dB.. + Popularity,data = top50,kernel='radial',cost=1),top50,Loudness..dB..
```




In both models, most songs are classified as Pop, except louder and more popular songs are classified as Latin.

# LDA/QDA

## Evaluating Predictive Accuracy

We will evaluate the MSPE of the models using LOOCV, which is conveniently built into the `lda()` and `qda()` functions as an option.

```
lda.cv = lda(Genre ~ . -Genre - X1 - Track.Name - Artist.Name,data = top50,CV = TRUE)
table(Predicted= lda.cv$class,Actual= Genre)
```

```
##           Actual
## Predicted Pop Latin Hip Hop
##   Pop      20     2       3
##   Latin     2    10       0
##   Hip Hop   3     1       9
```

```
lda.MSPE = mean(Genre == lda.cv$class)
lda.MSPE
```

```
## [1] 0.78
```

```r
qda.cv = qda(Genre ~ . -Genre - X1 - Track.Name - Artist.Name,data = top50,CV = TRUE)
table(Predicted= qda.cv$class,Actual= Genre)
```

```
##          Actual
## Predicted Pop Latin Hip Hop
##   Pop      22    10      10
##   Latin     2     3       1
##   Hip Hop   1     0       1
```

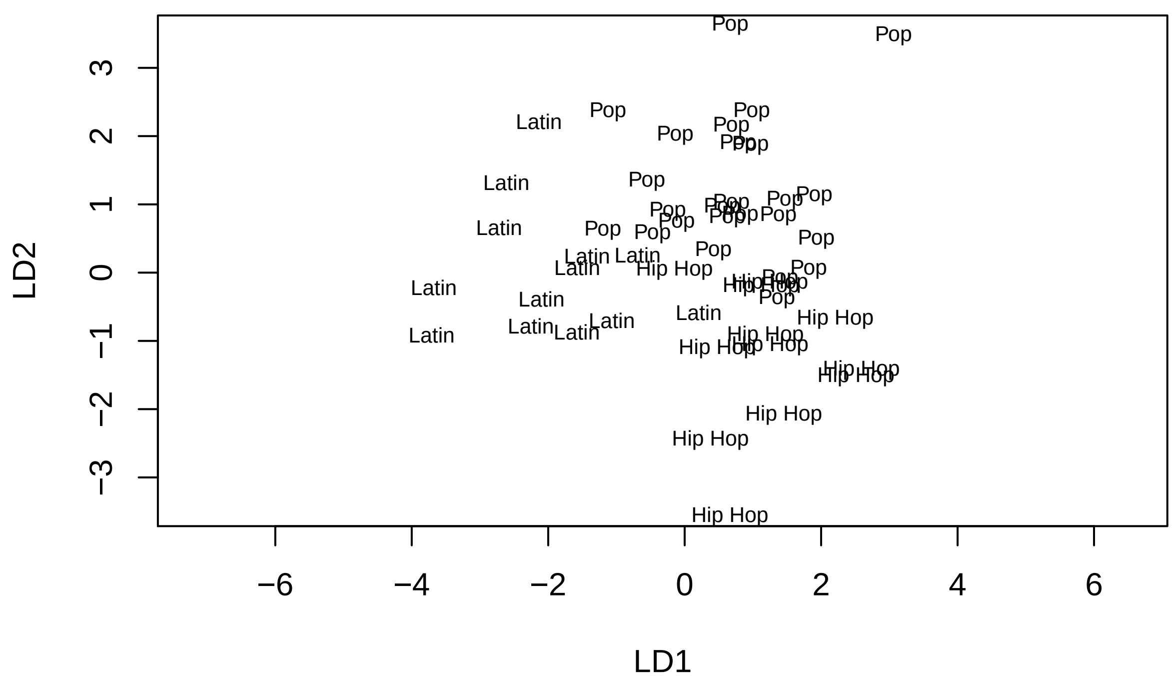
```r
qda.MSPE = mean(Genre == qda.cv$class)
qda.MSPE
```

```
## [1] 0.52
```

The LDA performs which better than the QDA, which indicates that a linear decision boundary, thus equal variances between prior probabilities. The LDA has a classification rate of .78. The QDA, on the other hand, has a dismal classification rate of .52, which is only barely better than guessing.

## Plotting LDA

```r
lda.fit = lda(Genre ~ . -Genre - X1 - Track.Name - Artist.Name,data = top50)
plot(lda.fit)
```



We can see that LDA does a nice job separating the three genres, but that there is some overlap in the middle.

## Comparison

Let us compare the MSPE for each of our four models that predict Genre:

```
c(SVM.Lin= svm.linear.MSPE, SVM.Rad= svm.radial.MSPE, LDA= lda.MSPE, QDA= qda.MSPE)
```

```
## SVM.Lin SVM.Rad     LDA     QDA
##    0.70    0.72    0.78    0.52
```

LDA has the best MSPE, while the two SVM models are not too far behind. QDA, meanwhile has a poor MSPE. It is also worth noting a finding which is not captured by MSPE: LDA is much better at correctly classifying Latin and Hip Hop, whereas SVM takes advantage of the class imbalance by overclassifying to Pop.

## Linear Regression with Variable Selection

This test focuses on variable selection for the response "Popularity." In other words, finding the most optimal linear regression function for the prediction of a song's popularity.

```
reg.fit <- regsubsets(Popularity ~ . -X1 -Track.Name -Artist.Name -Popularity,
           data = top50, nvmax = 11, method = "exhaustive")

summary(reg.fit)
```

```
## Subset selection object
## Call: regsubsets.formula(Popularity ~ . - X1 - Track.Name - Artist.Name -
##     Popularity, data = top50, nvmax = 11, method = "exhaustive")
## 11 Variables  (and intercept)
##                  Forced in Forced out
## GenreLatin           FALSE      FALSE
## GenreHip Hop         FALSE      FALSE
## Beats.Per.Minute     FALSE      FALSE
## Energy               FALSE      FALSE
## Danceability         FALSE      FALSE
## Loudness..dB..       FALSE      FALSE
## Liveness             FALSE      FALSE
## Valence.             FALSE      FALSE
## Length.              FALSE      FALSE
## Acousticness..       FALSE      FALSE
## Speechiness.         FALSE      FALSE
## 1 subsets of each size up to 11
## Selection Algorithm: exhaustive
##           GenreLatin GenreHip Hop Beats.Per.Minute Energy Danceability
## 1  ( 1 )  " "        " "          " "              " "    " "
## 2  ( 1 )  "*"        " "          " "              " "    " "
## 3  ( 1 )  "*"        " "          " "              " "    " "
```

```
## 4  ( 1 )  "*"         " "            " "                " "     " "
## 5  ( 1 )  "*"         " "            " "                " "     " "
## 6  ( 1 )  "*"         "*"            " "                " "     " "
## 7  ( 1 )  "*"         "*"            " "                " "     "*"
## 8  ( 1 )  "*"         "*"            " "                " "     "*"
## 9  ( 1 )  "*"         "*"            "*"                " "     "*"
## 10  ( 1 ) "*"         "*"            " "                "*"     "*"
## 11  ( 1 ) "*"         "*"            "*"                "*"     "*"
##           Loudness..dB.. Liveness Valence. Length. Acousticness.. Speechiness.
## 1  ( 1 )  " "            " "      "*"      " "     " "            " "
## 2  ( 1 )  " "            " "      "*"      " "     " "            " "
## 3  ( 1 )  " "            " "      "*"      "*"     " "            " "
## 4  ( 1 )  " "            "*"      "*"      "*"     " "            " "
## 5  ( 1 )  " "            "*"      "*"      "*"     " "            "*"
## 6  ( 1 )  " "            "*"      "*"      "*"     " "            "*"
## 7  ( 1 )  "*"            "*"      "*"      "*"     " "            " "
## 8  ( 1 )  "*"            "*"      "*"      "*"     "*"            " "
## 9  ( 1 )  "*"            "*"      "*"      "*"     "*"            " "
## 10  ( 1 ) "*"            "*"      "*"      "*"     "*"            "*"
## 11  ( 1 ) "*"            "*"      "*"      "*"     "*"            "*"
```

Using the Adjusted R^2 criterion, the model with the highest adjusted R^2 is the model with 7 predictors. It has an adjusted R^2 value of 0.2319. The 7 chosen predictors are: - GenreLatin*
- GenrePop*
- Danceability
- Loudness
- Liveness
- Valence
- Length

*Note: GenreLatin and GenrePop are dummy variables for the categorical variable Genre, which can either be Hip Hop, Pop or Latin.

```r
summary(reg.fit)$adjr2
```

```
##  [1] 0.0822367 0.1529003 0.2178352 0.2221503 0.2258999 0.2261826 0.2319301
##  [8] 0.2241151 0.2093685 0.1929890 0.1727235
```

```r
which.max(summary(reg.fit)$adjr2)
```

```
## [1] 7
```

```r
summary(reg.fit)$adjr2[7]
```

```
## [1] 0.2319301
```

The Mallow's Cp criterion finds that the closest Cp to p is the model with two predictors: GenrePop and Valence.

```r
summary(reg.fit)$cp
```

```
## [1]  7.250199  4.126215  1.491604  2.311411  3.171731  4.221314  4.994139
## [8]  6.453024  8.228163 10.044631 12.000000
```

BIC chooses the thrid model, with the three predictos: GenreLatin, Valence, Length.

```r
summary(reg.fit)$bic
```

```
## [1]  2.5022934  1.3555912  0.2046576  2.7411261  5.2879014  8.0321832
## [7] 10.3949210 13.6082434 17.2270247 20.8984218 24.7517540
```

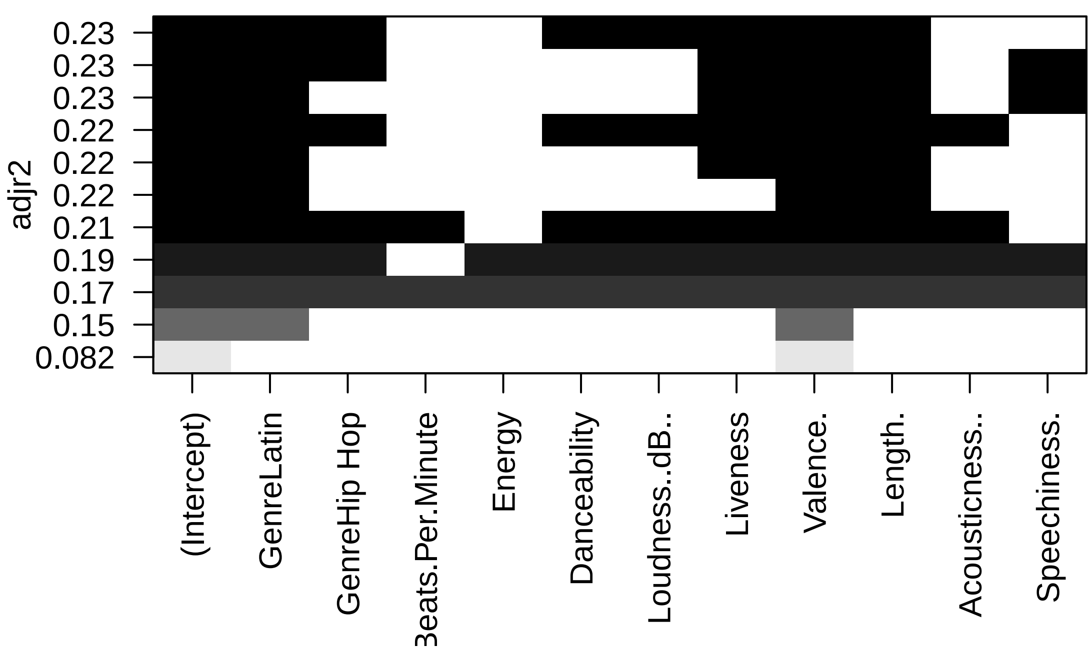
```r
which.min(summary(reg.fit)$bic)
```

```
## [1] 3
```

Looking at Adjusted R^2, Mallow's Cp, and the BIC, the results are all varied. However, I am going to choose the adjusted R^2 since it explains most of a song's popularity compared to the other models.

Looking at the plot we can visually compared the other models and their adjusted R^2.

```r
plot(reg.fit, scale = "adjr2" )
```




The coefficients for the model are below. We can see that only the intercept is very statistically significant, with GenreLatin, Valence and Length being statistically significant only if $\alpha < 10$.

```r
reg.bestfit <- lm(Popularity ~ Genre + Danceability + Loudness..dB.. + Liveness + Valence. + Length., da

summary(reg.bestfit)
```

```
##
## Call:
## lm(formula = Popularity ~ Genre + Danceability + Loudness..dB.. +
##     Liveness + Valence. + Length., data = top50)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -12.9571  -1.4213  -0.0145   2.3484   7.5334
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)    96.19506    5.71113  16.843  < 2e-16 ***
## GenreLatin      6.35060    1.71811   3.696 0.000628 ***
## GenreHip Hop    2.55717    1.75413   1.458 0.152334
## Danceability   -0.06306    0.05426  -1.162 0.251701
## Loudness..dB.. -0.44190    0.32862  -1.345 0.185931
## Liveness        0.07997    0.05428   1.473 0.148151
## Valence.       -0.05879    0.03131  -1.878 0.067331 .
## Length.        -0.03444    0.01745  -1.974 0.054957 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.936 on 42 degrees of freedom
## Multiple R-squared:  0.3417, Adjusted R-squared:  0.2319
## F-statistic: 3.114 on 7 and 42 DF,  p-value: 0.009727
```

```r
Yhat.bestfit <- predict(reg.bestfit)
MSE.bestfit <- mean((Yhat.bestfit - Popularity)^2)
MSE.bestfit
```

```
## [1] 13.01549
```

```r
reg.full <- lm(Popularity ~ . -X1 -Track.Name -Artist.Name -Popularity,
               data = top50)
Yhat.full <- predict(reg.full)
MSE.full <- mean((Yhat.full - Popularity)^2)
MSE.full
```

```
## [1] 12.68367
```

The prediction MSE for the best fit linear model is 13.01549. The prediciton MSE for the full model was slightly smaller at 12.68, however its R^2 was 0.17 compared to 0.23.

## Initialization of Training and Testing Data

```r
set.seed(11)
#Create training and testing data
n=nrow(top50)
training = sample(1:n,n/2)
testing <- -training
spotify.training <- top50[training, ]
spotify.testing <- top50[testing, ]
#Create training and testing matrices
spo.mat.training <- model.matrix(Popularity~.-X1 -Track.Name -Artist.Name -Popularity, data=spotify.tra
spo.mat.testing <- model.matrix(Popularity~.-X1 -Track.Name -Artist.Name -Popularity, data=spotify.test
```

# Using LASSO and Ridge Regression

```r
#LASSO
spo.lasso <- cv.glmnet(spo.mat.training, spotify.training$Popularity, alpha=1)
```

```
## Warning: Option grouped=FALSE enforced in cv.glmnet, since < 3 observations per
## fold
```

```r
(lambda <- spo.lasso$lambda.min)  # optimal lambda
```

```
## [1] 1.375505
```

```r
pred.lasso <- predict(spo.lasso, s=lambda, newx=spo.mat.testing)
(err.lasso <- mean((spotify.testing$Popularity - pred.lasso)^2))
```

```
## [1] 15.8992
```

```r
predict(spo.lasso, s=lambda, type="coefficients")
```

```
## 12 x 1 sparse Matrix of class "dgCMatrix"
##                       1
## (Intercept)       87.08
## GenreLatin          .
## GenreHip Hop        .
## Beats.Per.Minute    .
## Energy              .
## Danceability        .
## Loudness..dB..      .
## Liveness            .
## Valence.            .
## Length.             .
## Acousticness..      .
## Speechiness.        .
```

```r
summary(spo.lasso)
```

```
##            Length Class  Mode
## lambda     62     -none- numeric
## cvm        62     -none- numeric
## cvsd       62     -none- numeric
## cvup       62     -none- numeric
## cvlo       62     -none- numeric
## nzero      62     -none- numeric
## call        4     -none- call
## name        1     -none- character
## glmnet.fit 12     elnet  list
## lambda.min  1     -none- numeric
## lambda.1se  1     -none- numeric
```

```r
#Ridge Regression
spo.ridge <- cv.glmnet(spo.mat.training, spotify.training$Popularity, alpha=0)
```

```
## Warning: Option grouped=FALSE enforced in cv.glmnet, since < 3 observations per
## fold
```

```r
(lambda <- spo.ridge$lambda.min)
```

```
## [1] 1375.505
```

```r
pred.ridge <- predict(spo.ridge, s=lambda, newx=spo.mat.testing)
(err.ridge <- mean((spotify.testing$Popularity - pred.ridge)^2))
```

```
## [1] 15.8992
```

```r
predict(spo.ridge, s=lambda, type="coefficients")
```

```
## 12 x 1 sparse Matrix of class "dgCMatrix"
##                             1
## (Intercept)       8.708000e+01
## GenreLatin        3.094436e-36
## GenreHip Hop      1.406926e-36
## Beats.Per.Minute  7.995687e-39
## Energy           -6.150706e-38
## Danceability      4.069386e-38
## Loudness..dB..   -4.619195e-37
## Liveness          4.705235e-39
## Valence.         -3.129497e-38
## Length.          -6.555672e-39
## Acousticness..   -4.168423e-38
## Speechiness.      9.312137e-38
```

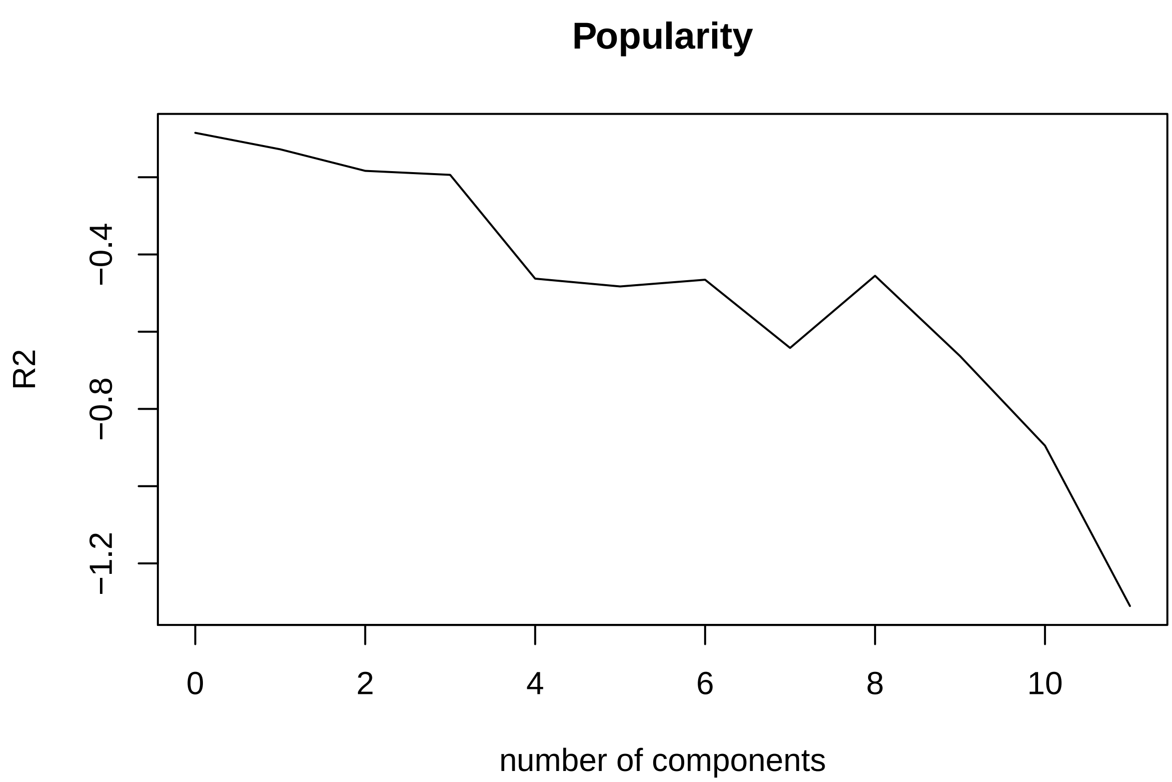
```r
summary(spo.ridge)
```

```
##              Length Class  Mode
## lambda       100    -none- numeric
## cvm          100    -none- numeric
## cvsd         100    -none- numeric
## cvup         100    -none- numeric
## cvlo         100    -none- numeric
## nzero        100    -none- numeric
## call           4    -none- call
## name           1    -none- character
## glmnet.fit    12    elnet  list
## lambda.min     1    -none- numeric
## lambda.1se     1    -none- numeric
```

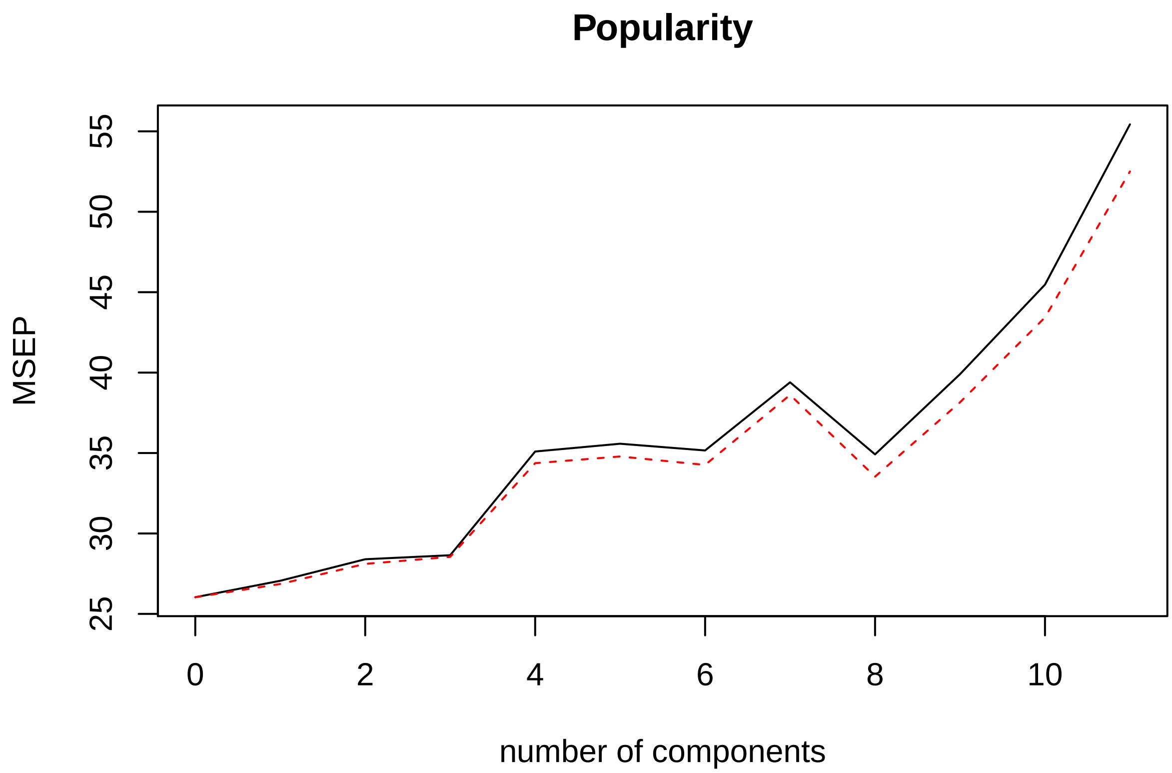
- MSE for Ridge Regression and LASSO methods is 15.9.

## Using PCR

```r
reg.pcr <- pcr(Popularity ~ .-X1 -Track.Name -Artist.Name -Popularity, data = spotify.training, scale =
validationplot(reg.pcr, val.type = "R2")
```




```r
validationplot(reg.pcr, val.type = "MSEP")
```



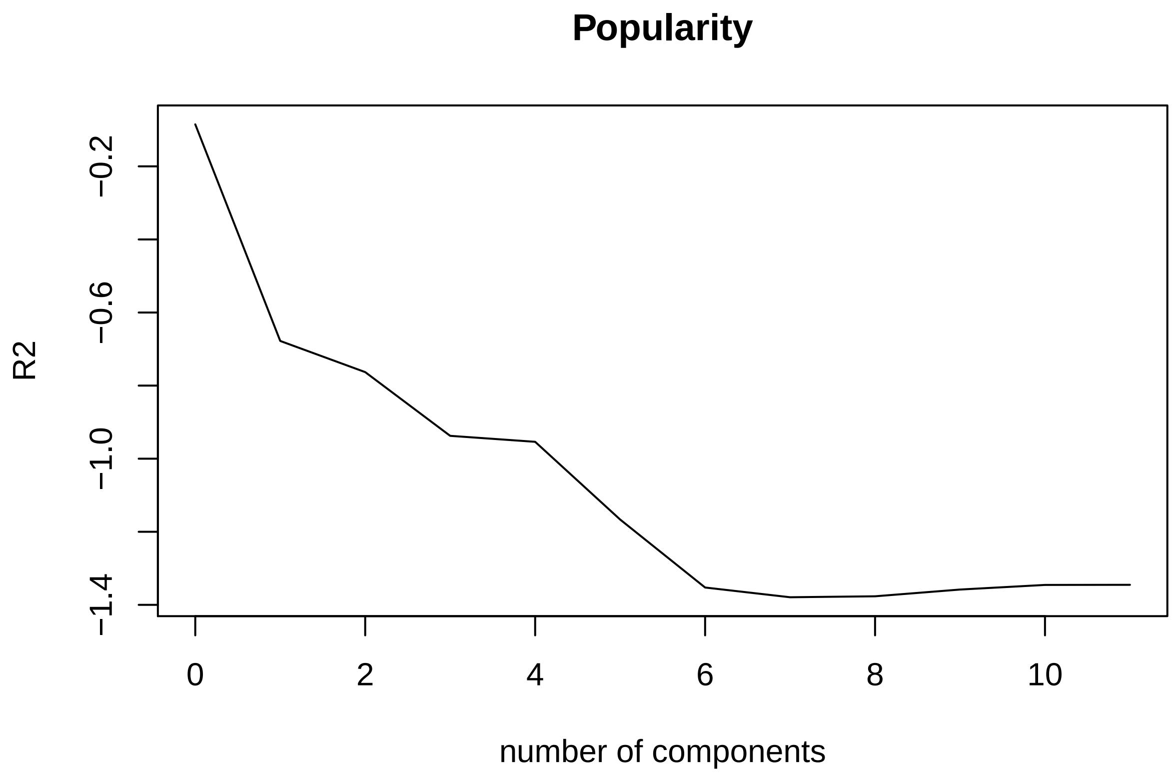

```
pred.pcr <- predict(reg.pcr, spotify.testing, ncomp = 10)
#Calculate MSE
mean((pred.pcr - spotify.testing$Popularity)^2)
```

```
## [1] 20.97863
```

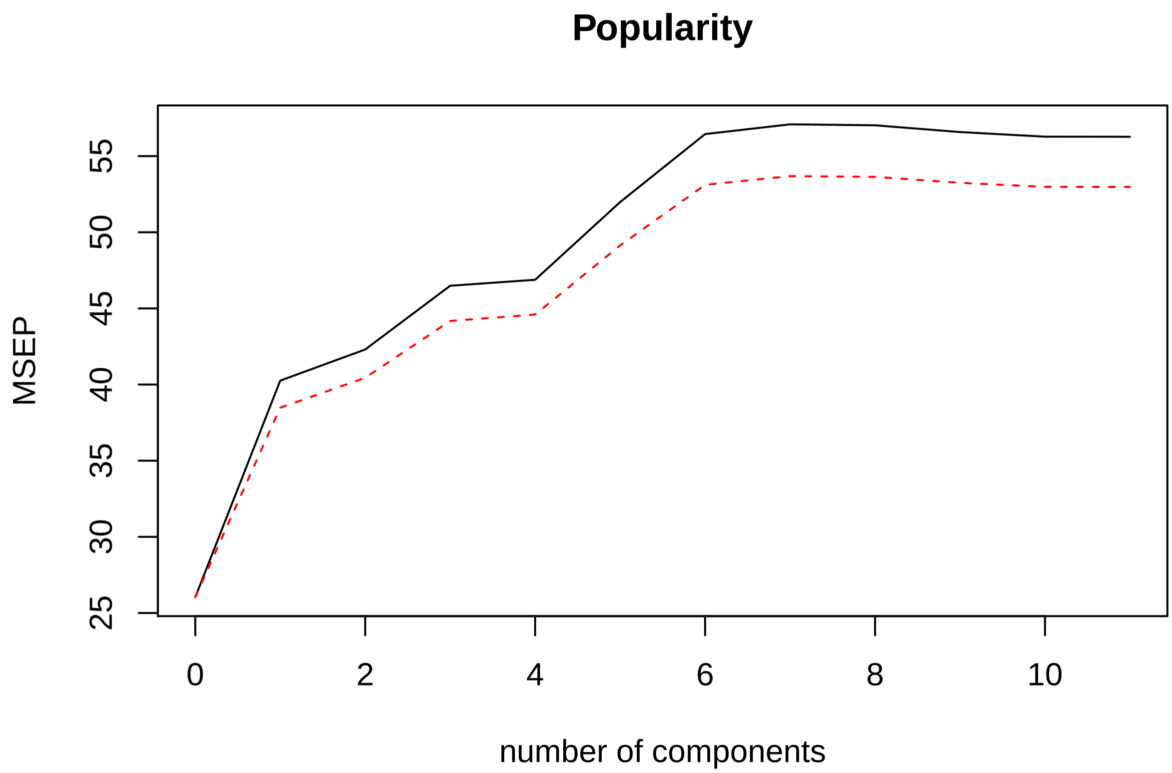
- MSE for PCR method is 19. Oddly, the screeplot is sloped upward which may suggest less components are better at explaining the model.

#Using PLS

```
reg.pls <- plsr(Popularity ~ .-X1 -Track.Name -Artist.Name -Popularity, data = spotify.training, scale =
validationplot(reg.pls, val.type = "R2")
```




```r
validationplot(reg.pls, val.type = "MSEP")
```




```
pred.pls <- predict(reg.pls, spotify.testing, ncomp = 10)
#Calculate MSE
mean((pred.pls - spotify.testing$Popularity)^2)
```

```
## [1] 18.79896
```

- MSE for PLS method is 18.8. Same with PCR, the screeplot is oddly sloped upward which suggests more components will lead to a larger MSE.
- The MSE for Ridge and LASSO is lower than the MSE for PCR and PLS and the screeplots were shaped oddly so I would advise against PCR and PLS here in favor of LASSO and Ridge.