MATHS 7107 Data Taming Assignment Final Report

Chang Dong

2023-04-24

# Appendix

pacman::p\_load(skimr, tidyverse, tidymodels, themis, recipes, dials,kknn, vip,forcats,caret,MASS,discrim,yardstick,pROC)

## 1.1 Data import

spotify\_songs\_origin <- readr::read\_csv('https://raw.githubusercontent.com/rfordatascience/tidytuesday/master/data/2020/2020-01-21/spotify\_songs.csv')

## Rows: 32833 Columns: 23  
## ── Column specification ────────────────────────────────────────────────────────  
## Delimiter: ","  
## chr (10): track\_id, track\_name, track\_artist, track\_album\_id, track\_album\_na...  
## dbl (13): track\_popularity, danceability, energy, key, loudness, mode, speec...  
##   
## ℹ Use `spec()` to retrieve the full column specification for this data.  
## ℹ Specify the column types or set `show\_col\_types = FALSE` to quiet this message.

skim(spotify\_songs\_origin)

Data summary

|  |  |
| --- | --- |
| Name | spotify\_songs\_origin |
| Number of rows | 32833 |
| Number of columns | 23 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |
| Column type frequency: |  |
| character | 10 |
| numeric | 13 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |
| Group variables | None |

**Variable type: character**

| skim\_variable | n\_missing | complete\_rate | min | max | empty | n\_unique | whitespace |
| --- | --- | --- | --- | --- | --- | --- | --- |
| track\_id | 0 | 1 | 22 | 22 | 0 | 28356 | 0 |
| track\_name | 5 | 1 | 1 | 144 | 0 | 23449 | 0 |
| track\_artist | 5 | 1 | 2 | 69 | 0 | 10692 | 0 |
| track\_album\_id | 0 | 1 | 22 | 22 | 0 | 22545 | 0 |
| track\_album\_name | 5 | 1 | 1 | 151 | 0 | 19743 | 0 |
| track\_album\_release\_date | 0 | 1 | 4 | 10 | 0 | 4530 | 0 |
| playlist\_name | 0 | 1 | 6 | 120 | 0 | 449 | 0 |
| playlist\_id | 0 | 1 | 22 | 22 | 0 | 471 | 0 |
| playlist\_genre | 0 | 1 | 3 | 5 | 0 | 6 | 0 |
| playlist\_subgenre | 0 | 1 | 4 | 25 | 0 | 24 | 0 |

**Variable type: numeric**

| skim\_variable | n\_missing | complete\_rate | mean | sd | p0 | p25 | p50 | p75 | p100 | hist |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| track\_popularity | 0 | 1 | 42.48 | 24.98 | 0.00 | 24.00 | 45.00 | 62.00 | 100.00 | ▆▆▇▆▁ |
| danceability | 0 | 1 | 0.65 | 0.15 | 0.00 | 0.56 | 0.67 | 0.76 | 0.98 | ▁▁▃▇▃ |
| energy | 0 | 1 | 0.70 | 0.18 | 0.00 | 0.58 | 0.72 | 0.84 | 1.00 | ▁▁▅▇▇ |
| key | 0 | 1 | 5.37 | 3.61 | 0.00 | 2.00 | 6.00 | 9.00 | 11.00 | ▇▂▅▅▆ |
| loudness | 0 | 1 | -6.72 | 2.99 | -46.45 | -8.17 | -6.17 | -4.64 | 1.27 | ▁▁▁▂▇ |
| mode | 0 | 1 | 0.57 | 0.50 | 0.00 | 0.00 | 1.00 | 1.00 | 1.00 | ▆▁▁▁▇ |
| speechiness | 0 | 1 | 0.11 | 0.10 | 0.00 | 0.04 | 0.06 | 0.13 | 0.92 | ▇▂▁▁▁ |
| acousticness | 0 | 1 | 0.18 | 0.22 | 0.00 | 0.02 | 0.08 | 0.26 | 0.99 | ▇▂▁▁▁ |
| instrumentalness | 0 | 1 | 0.08 | 0.22 | 0.00 | 0.00 | 0.00 | 0.00 | 0.99 | ▇▁▁▁▁ |
| liveness | 0 | 1 | 0.19 | 0.15 | 0.00 | 0.09 | 0.13 | 0.25 | 1.00 | ▇▃▁▁▁ |
| valence | 0 | 1 | 0.51 | 0.23 | 0.00 | 0.33 | 0.51 | 0.69 | 0.99 | ▃▇▇▇▃ |
| tempo | 0 | 1 | 120.88 | 26.90 | 0.00 | 99.96 | 121.98 | 133.92 | 239.44 | ▁▂▇▂▁ |
| duration\_ms | 0 | 1 | 225799.81 | 59834.01 | 4000.00 | 187819.00 | 216000.00 | 253585.00 | 517810.00 | ▁▇▇▁▁ |

## 1.2 Data Cleanning Method

These have n\_unique more than 1000 should be considered as Text instead of Categorical data (“track\_id”,“track\_name”, “track\_artist”, “track\_album\_id”, “track\_album\_name”), , “track\_album\_release\_date” is not included, because we know it is a time series should be numerical.

text\_variables <- c("track\_id", "track\_name", "track\_artist", "track\_album\_id", "track\_album\_name")  
  
spotify\_songs <- spotify\_songs\_origin %>%  
 dplyr::select(-dplyr::any\_of(text\_variables))

Categorical Data (factor)

categorical\_variables <- c("playlist\_name", "playlist\_id", "playlist\_genre", "playlist\_subgenre")  
  
spotify\_songs <- spotify\_songs %>%  
 mutate(across(all\_of(categorical\_variables), as.factor))

Numerical Data (numerical)

numerical\_variables <- c("track\_popularity","danceability", "energy", "key",   
 "loudness","mode", "speechiness","acousticness",  
 "instrumentalness", "liveness", "valence",   
 "tempo", "duration\_ms")   
  
spotify\_songs <- spotify\_songs %>%  
 mutate(across(all\_of(numerical\_variables), as.numeric))  
  
spotify\_songs <- spotify\_songs %>%  
 mutate(track\_album\_release\_year = as.numeric(format(as.Date(track\_album\_release\_date, format = "%Y-%m-%d"), "%Y"))) %>%  
 dplyr::select(-track\_album\_release\_date)  
  
numerical\_variables <- c(numerical\_variables, "track\_album\_release\_year")

skim(spotify\_songs)

Data summary

|  |  |
| --- | --- |
| Name | spotify\_songs |
| Number of rows | 32833 |
| Number of columns | 18 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |
| Column type frequency: |  |
| factor | 4 |
| numeric | 14 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |
| Group variables | None |

**Variable type: factor**

| skim\_variable | n\_missing | complete\_rate | ordered | n\_unique | top\_counts |
| --- | --- | --- | --- | --- | --- |
| playlist\_name | 0 | 1 | FALSE | 449 | Ind: 308, 202: 247, Per: 244, Har: 219 |
| playlist\_id | 0 | 1 | FALSE | 471 | 4Jk: 247, 37i: 198, 6Kn: 195, 3xM: 189 |
| playlist\_genre | 0 | 1 | FALSE | 6 | edm: 6043, rap: 5746, pop: 5507, r&b: 5431 |
| playlist\_subgenre | 0 | 1 | FALSE | 24 | pro: 1809, sou: 1675, ind: 1672, lat: 1656 |

**Variable type: numeric**

| skim\_variable | n\_missing | complete\_rate | mean | sd | p0 | p25 | p50 | p75 | p100 | hist |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| track\_popularity | 0 | 1.00 | 42.48 | 24.98 | 0.00 | 24.00 | 45.00 | 62.00 | 100.00 | ▆▆▇▆▁ |
| danceability | 0 | 1.00 | 0.65 | 0.15 | 0.00 | 0.56 | 0.67 | 0.76 | 0.98 | ▁▁▃▇▃ |
| energy | 0 | 1.00 | 0.70 | 0.18 | 0.00 | 0.58 | 0.72 | 0.84 | 1.00 | ▁▁▅▇▇ |
| key | 0 | 1.00 | 5.37 | 3.61 | 0.00 | 2.00 | 6.00 | 9.00 | 11.00 | ▇▂▅▅▆ |
| loudness | 0 | 1.00 | -6.72 | 2.99 | -46.45 | -8.17 | -6.17 | -4.64 | 1.27 | ▁▁▁▂▇ |
| mode | 0 | 1.00 | 0.57 | 0.50 | 0.00 | 0.00 | 1.00 | 1.00 | 1.00 | ▆▁▁▁▇ |
| speechiness | 0 | 1.00 | 0.11 | 0.10 | 0.00 | 0.04 | 0.06 | 0.13 | 0.92 | ▇▂▁▁▁ |
| acousticness | 0 | 1.00 | 0.18 | 0.22 | 0.00 | 0.02 | 0.08 | 0.26 | 0.99 | ▇▂▁▁▁ |
| instrumentalness | 0 | 1.00 | 0.08 | 0.22 | 0.00 | 0.00 | 0.00 | 0.00 | 0.99 | ▇▁▁▁▁ |
| liveness | 0 | 1.00 | 0.19 | 0.15 | 0.00 | 0.09 | 0.13 | 0.25 | 1.00 | ▇▃▁▁▁ |
| valence | 0 | 1.00 | 0.51 | 0.23 | 0.00 | 0.33 | 0.51 | 0.69 | 0.99 | ▃▇▇▇▃ |
| tempo | 0 | 1.00 | 120.88 | 26.90 | 0.00 | 99.96 | 121.98 | 133.92 | 239.44 | ▁▂▇▂▁ |
| duration\_ms | 0 | 1.00 | 225799.81 | 59834.01 | 4000.00 | 187819.00 | 216000.00 | 253585.00 | 517810.00 | ▁▇▇▁▁ |
| track\_album\_release\_year | 1886 | 0.94 | 2012.20 | 10.40 | 1957.00 | 2010.00 | 2017.00 | 2019.00 | 2020.00 | ▁▁▁▁▇ |

spotify\_songs <- spotify\_songs %>%  
 drop\_na()  
  
skim(spotify\_songs)

Data summary

|  |  |
| --- | --- |
| Name | spotify\_songs |
| Number of rows | 30947 |
| Number of columns | 18 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |
| Column type frequency: |  |
| factor | 4 |
| numeric | 14 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |
| Group variables | None |

**Variable type: factor**

| skim\_variable | n\_missing | complete\_rate | ordered | n\_unique | top\_counts |
| --- | --- | --- | --- | --- | --- |
| playlist\_name | 0 | 1 | FALSE | 449 | Ind: 299, 202: 247, Per: 212, Har: 211 |
| playlist\_id | 0 | 1 | FALSE | 471 | 4Jk: 247, 6Kn: 195, 37i: 190, 3xM: 189 |
| playlist\_genre | 0 | 1 | FALSE | 6 | edm: 5969, rap: 5471, pop: 5303, r&b: 5094 |
| playlist\_subgenre | 0 | 1 | FALSE | 24 | pro: 1760, ind: 1647, lat: 1573, neo: 1547 |

**Variable type: numeric**

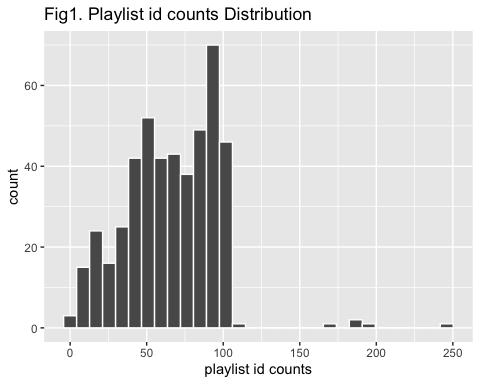
| skim\_variable | n\_missing | complete\_rate | mean | sd | p0 | p25 | p50 | p75 | p100 | hist |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| track\_popularity | 0 | 1 | 42.75 | 24.96 | 0.00 | 25.00 | 45.00 | 62.00 | 100.00 | ▆▆▇▆▁ |
| danceability | 0 | 1 | 0.66 | 0.14 | 0.00 | 0.57 | 0.67 | 0.76 | 0.98 | ▁▁▃▇▃ |
| energy | 0 | 1 | 0.70 | 0.18 | 0.00 | 0.58 | 0.72 | 0.84 | 1.00 | ▁▁▅▇▇ |
| key | 0 | 1 | 5.37 | 3.61 | 0.00 | 2.00 | 6.00 | 9.00 | 11.00 | ▇▂▅▅▆ |
| loudness | 0 | 1 | -6.64 | 2.95 | -46.45 | -8.07 | -6.09 | -4.61 | 1.27 | ▁▁▁▂▇ |
| mode | 0 | 1 | 0.56 | 0.50 | 0.00 | 0.00 | 1.00 | 1.00 | 1.00 | ▆▁▁▁▇ |
| speechiness | 0 | 1 | 0.11 | 0.10 | 0.00 | 0.04 | 0.06 | 0.13 | 0.92 | ▇▂▁▁▁ |
| acousticness | 0 | 1 | 0.18 | 0.22 | 0.00 | 0.02 | 0.08 | 0.26 | 0.99 | ▇▂▁▁▁ |
| instrumentalness | 0 | 1 | 0.09 | 0.23 | 0.00 | 0.00 | 0.00 | 0.01 | 0.99 | ▇▁▁▁▁ |
| liveness | 0 | 1 | 0.19 | 0.15 | 0.00 | 0.09 | 0.13 | 0.25 | 1.00 | ▇▃▁▁▁ |
| valence | 0 | 1 | 0.51 | 0.23 | 0.00 | 0.33 | 0.51 | 0.69 | 0.99 | ▃▇▇▇▃ |
| tempo | 0 | 1 | 120.94 | 26.85 | 0.00 | 99.97 | 122.00 | 133.52 | 239.44 | ▁▂▇▂▁ |
| duration\_ms | 0 | 1 | 223950.10 | 59113.89 | 4000.00 | 186750.00 | 214400.00 | 251133.00 | 517810.00 | ▁▇▇▁▁ |
| track\_album\_release\_year | 0 | 1 | 2012.20 | 10.40 | 1957.00 | 2010.00 | 2017.00 | 2019.00 | 2020.00 | ▁▁▁▁▇ |

## 1.3 Exploratory Data Analysis(EDA) Method

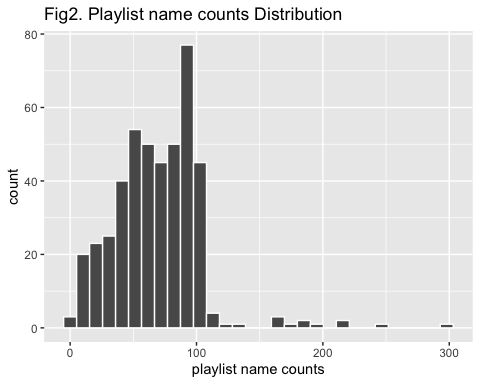
### Categorical variable

**playlist\_id**

playlist\_id\_counts <- spotify\_songs %>%  
 count(playlist\_id) %>%  
 arrange(desc(n))  
playlist\_id\_counts %>% ggplot(aes(x= n)) + geom\_histogram(bins = 30, color = "white") +  
 ggtitle("Fig1. Playlist id counts Distribution") +xlab("playlist id counts") +ylab("count")

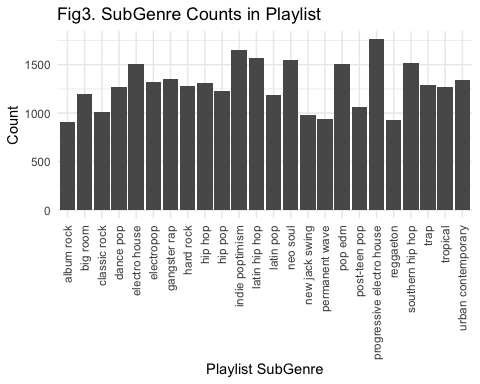
 **playlist\_name**

playlist\_name\_counts <- spotify\_songs %>%  
 count(playlist\_name) %>%  
 arrange(desc(n))  
playlist\_name\_counts %>% ggplot(aes(x= n)) + geom\_histogram(bins = 30, color = "white") +  
 ggtitle("Fig2. Playlist name counts Distribution") +xlab("playlist name counts") +ylab("count")



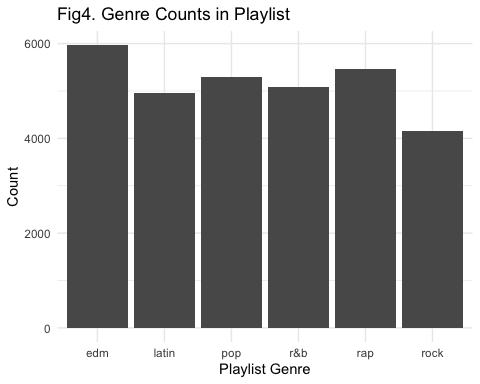
**playlist\_subgenre**

playlist\_subgenre\_counts <- spotify\_songs %>%  
 count(playlist\_subgenre) %>%  
 arrange(desc(n))  
  
playlist\_subgenre\_counts %>%  
 ggplot(aes(x = playlist\_subgenre, y = n)) +  
 geom\_bar(stat = "identity") +  
 theme\_minimal() +  
 theme(axis.text.x = element\_text(angle = 90, vjust = 0.5, hjust = 1)) +  
 labs(title = "Fig3. SubGenre Counts in Playlist",  
 x = "Playlist SubGenre",  
 y = "Count")



**playlist\_genre (Response variable)**

playlist\_genre\_counts <- spotify\_songs %>%  
 count(playlist\_genre) %>%  
 arrange(desc(n))  
  
playlist\_genre\_counts %>%  
 ggplot(aes(x = playlist\_genre, y = n)) +  
 geom\_bar(stat = "identity") +  
 theme\_minimal() +  
 labs(title = "Fig4. Genre Counts in Playlist",  
 x = "Playlist Genre",  
 y = "Count")



#### -Test (categorical data versus categorical data)

chisq\_test <- function(variable\_name) {  
 contingency\_table <- spotify\_songs %>%  
 count(!!sym(variable\_name), playlist\_genre) %>%  
 spread(key = playlist\_genre, value = n, fill = 0) %>%  
 column\_to\_rownames(var= variable\_name)  
   
 chi\_square\_test <- chisq.test(contingency\_table)  
 cat(variable\_name)  
 print(chi\_square\_test)  
}  
  
for (variable in setdiff(categorical\_variables, "playlist\_genre")) {  
 chisq\_test(variable)  
}

## Warning in chisq.test(contingency\_table): Chi-squared approximation may be  
## incorrect

## playlist\_name  
## Pearson's Chi-squared test  
##   
## data: contingency\_table  
## X-squared = 152976, df = 2240, p-value < 2.2e-16

## Warning in chisq.test(contingency\_table): Chi-squared approximation may be  
## incorrect

## playlist\_id  
## Pearson's Chi-squared test  
##   
## data: contingency\_table  
## X-squared = 152976, df = 2350, p-value < 2.2e-16  
##   
## playlist\_subgenre  
## Pearson's Chi-squared test  
##   
## data: contingency\_table  
## X-squared = 154735, df = 115, p-value < 2.2e-16

spotify\_songs <- spotify\_songs %>%   
 dplyr::select(-setdiff(categorical\_variables, "playlist\_genre"))

## Numerical variable

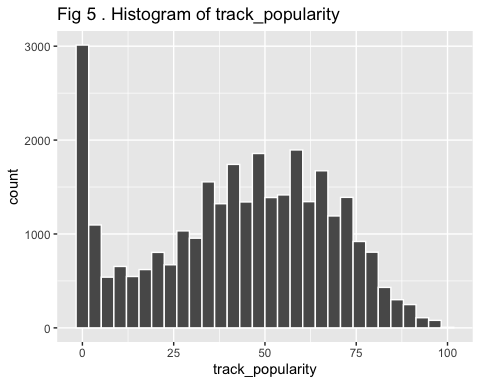
**Compile a standard function for EDA**

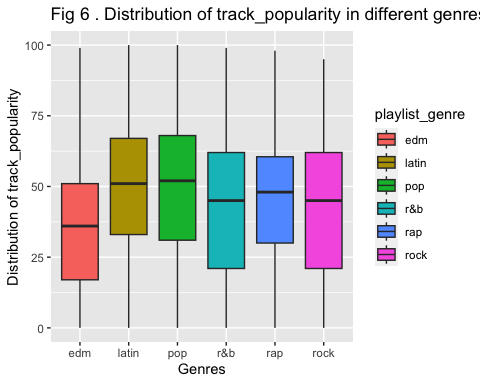
* plot a histogram to check the distribution of each numerical variable
* plot a box plot to check the distribution of each numerical variable in different genres
* Anova test to check the distribution differences in different genres

analyze\_variable\_hist <- function(variable\_name) {  
   
 # histogram  
 idx = 4 + 2\*which(numerical\_variables == variable\_name)-1  
 histo\_gram <- spotify\_songs %>% ggplot(aes\_string(variable\_name)) +   
 geom\_histogram(bins = 30,color="white") +  
 ggtitle(paste("Fig",idx,". Histogram of", variable\_name)) +   
 xlab(variable\_name) + ylab("count")  
 print(histo\_gram)  
   
}  
  
analyze\_variable\_box <- function(variable\_name) {  
# boxplot  
 idx = 4 + 2\*which(numerical\_variables == variable\_name)  
 box\_plot <- spotify\_songs %>% ggplot(aes\_string(x = "playlist\_genre",   
 y = variable\_name,  
 fill = "playlist\_genre")) +  
 geom\_boxplot() + ggtitle(paste("Fig",idx,". Distribution of", variable\_name, "in different genres")) +   
 xlab("Genres") + ylab(paste("Distribution of", variable\_name))  
 print(box\_plot)  
}  
  
analyze\_variable\_ANOVA <- function(variable\_name) {  
 #ANOVA  
 avg\_by\_genre <- tapply(spotify\_songs[[variable\_name]], spotify\_songs$playlist\_genre, mean)  
 print(avg\_by\_genre)  
   
 formula\_str <- paste(variable\_name, "~ playlist\_genre")  
 anova\_result <- aov(as.formula(formula\_str), data = spotify\_songs)  
 print(summary(anova\_result))  
   
 tukey\_result <- TukeyHSD(anova\_result)  
 print(tukey\_result)  
}

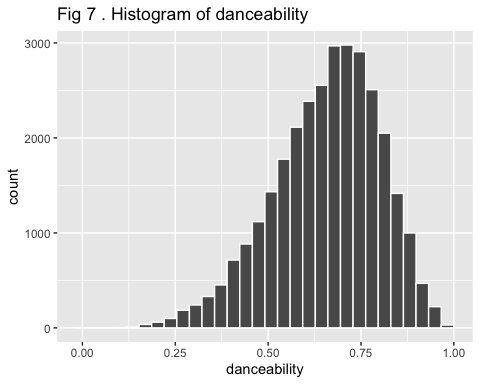
for (variable in numerical\_variables) {  
 cat("\n")  
 cat("\n")  
 analyze\_variable\_hist(variable)  
 cat("\n")  
 cat("\n")  
 analyze\_variable\_box(variable)  
 cat("\n")  
 cat(variable)  
 cat("\n")  
 analyze\_variable\_ANOVA (variable)  
 print("NEXT>>>>NEXT>>>>NEXT>>>>NEXT>>>>NEXT>>>>NEXT>>>>NEXT>>>>")  
}

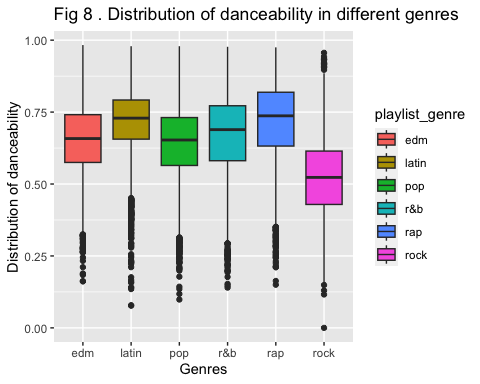
## Warning: `aes\_string()` was deprecated in ggplot2 3.0.0.  
## ℹ Please use tidy evaluation ideoms with `aes()`



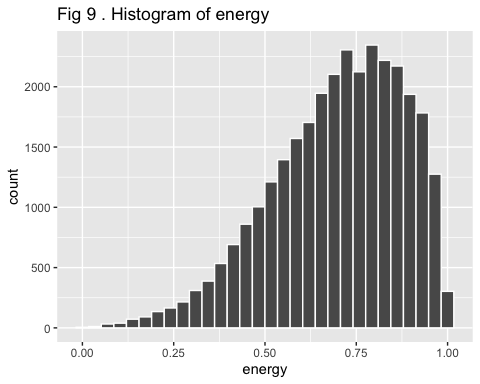


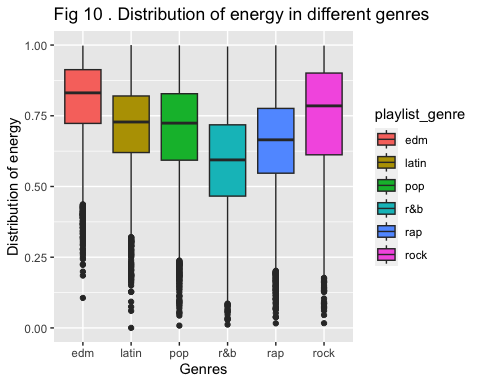
##   
## track\_popularity  
## edm latin pop r&b rap rock   
## 34.93885 47.56679 48.07147 41.81017 43.62128 41.42247   
## Df Sum Sq Mean Sq F value Pr(>F)   
## playlist\_genre 5 645473 129095 214.4 <2e-16 \*\*\*  
## Residuals 30941 18627072 602   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
## Tukey multiple comparisons of means  
## 95% family-wise confidence level  
##   
## Fit: aov(formula = as.formula(formula\_str), data = spotify\_songs)  
##   
## $playlist\_genre  
## diff lwr upr p adj  
## latin-edm 12.6279435 11.2847705 13.9711166 0.0000000  
## pop-edm 13.1326183 11.8131651 14.4520714 0.0000000  
## r&b-edm 6.8713181 5.5376085 8.2050277 0.0000000  
## rap-edm 8.6824251 7.3737436 9.9911066 0.0000000  
## rock-edm 6.4836233 5.0701367 7.8971099 0.0000000  
## pop-latin 0.5046747 -0.8762588 1.8856082 0.9039405  
## r&b-latin -5.7566255 -7.1511871 -4.3620638 0.0000000  
## rap-latin -3.9455185 -5.3161635 -2.5748734 0.0000000  
## rock-latin -6.1443202 -7.6153624 -4.6732780 0.0000000  
## r&b-pop -6.2613002 -7.6330308 -4.8895695 0.0000000  
## rap-pop -4.4501932 -5.7976020 -3.1027843 0.0000000  
## rock-pop -6.6489949 -8.0984113 -5.1995785 0.0000000  
## rap-r&b 1.8111070 0.4497344 3.1724796 0.0020783  
## rock-r&b -0.3876947 -1.8501012 1.0747117 0.9747460  
## rock-rap -2.1988017 -3.6384192 -0.7591843 0.0001952  
##   
## [1] "NEXT>>>>NEXT>>>>NEXT>>>>NEXT>>>>NEXT>>>>NEXT>>>>NEXT>>>>"



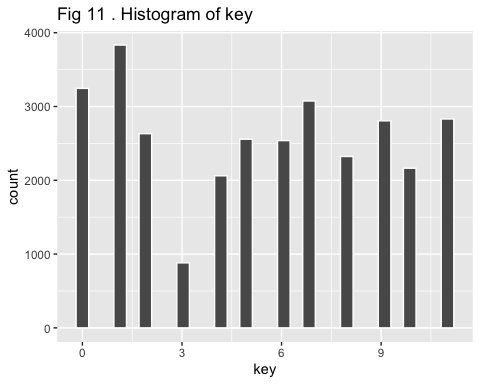


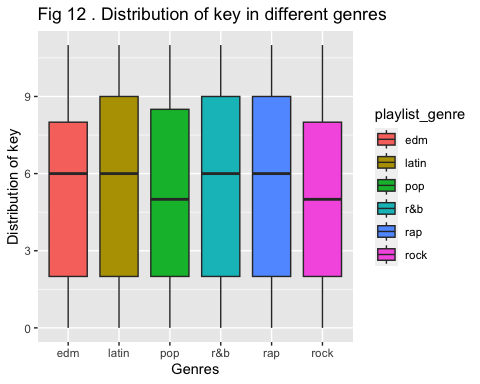
##   
## danceability  
## edm latin pop r&b rap rock   
## 0.6546405 0.7134672 0.6406509 0.6697148 0.7175184 0.5201989   
## Df Sum Sq Mean Sq F value Pr(>F)   
## playlist\_genre 5 115.7 23.149 1364 <2e-16 \*\*\*  
## Residuals 30941 525.3 0.017   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
## Tukey multiple comparisons of means  
## 95% family-wise confidence level  
##   
## Fit: aov(formula = as.formula(formula\_str), data = spotify\_songs)  
##   
## $playlist\_genre  
## diff lwr upr p adj  
## latin-edm 0.058826742 0.051694070 0.065959414 0.0000000  
## pop-edm -0.013989618 -0.020996330 -0.006982906 0.0000002  
## r&b-edm 0.015074287 0.007991868 0.022156705 0.0000000  
## rap-edm 0.062877894 0.055928383 0.069827405 0.0000000  
## rock-edm -0.134441537 -0.141947596 -0.126935478 0.0000000  
## pop-latin -0.072816359 -0.080149551 -0.065483168 0.0000000  
## r&b-latin -0.043752455 -0.051158016 -0.036346894 0.0000000  
## rap-latin 0.004051152 -0.003227405 0.011329709 0.6077803  
## rock-latin -0.193268278 -0.201079976 -0.185456581 0.0000000  
## r&b-pop 0.029063904 0.021779583 0.036348226 0.0000000  
## rap-pop 0.076867512 0.069712346 0.084022677 0.0000000  
## rock-pop -0.120451919 -0.128148776 -0.112755062 0.0000000  
## rap-r&b 0.047803607 0.040574290 0.055032925 0.0000000  
## rock-r&b -0.149515823 -0.157281662 -0.141749985 0.0000000  
## rock-rap -0.197319431 -0.204964253 -0.189674609 0.0000000  
##   
## [1] "NEXT>>>>NEXT>>>>NEXT>>>>NEXT>>>>NEXT>>>>NEXT>>>>NEXT>>>>"



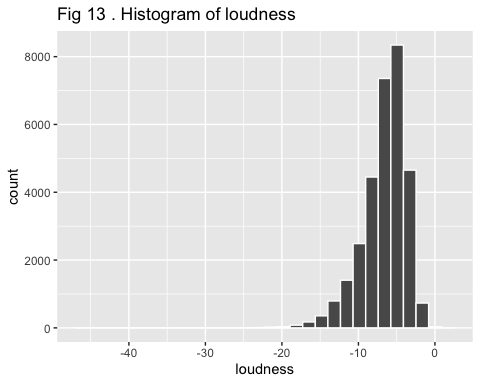


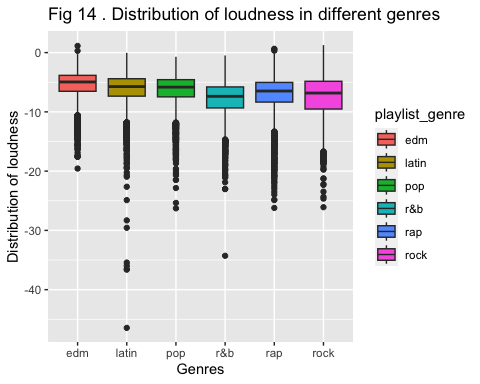
##   
## energy  
## edm latin pop r&b rap rock   
## 0.8029283 0.7076550 0.6983240 0.5889020 0.6502701 0.7385113   
## Df Sum Sq Mean Sq F value Pr(>F)   
## playlist\_genre 5 146.1 29.212 1045 <2e-16 \*\*\*  
## Residuals 30941 864.6 0.028   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
## Tukey multiple comparisons of means  
## 95% family-wise confidence level  
##   
## Fit: aov(formula = as.formula(formula\_str), data = spotify\_songs)  
##   
## $playlist\_genre  
## diff lwr upr p adj  
## latin-edm -0.09527329 -0.10442416 -0.0861224264 0.0000000  
## pop-edm -0.10460426 -0.11359353 -0.0956149974 0.0000000  
## r&b-edm -0.21402625 -0.22311265 -0.2049398600 0.0000000  
## rap-edm -0.15265824 -0.16157412 -0.1437423549 0.0000000  
## rock-edm -0.06441699 -0.07404689 -0.0547870819 0.0000000  
## pop-latin -0.00933097 -0.01873910 0.0000771546 0.0533869  
## r&b-latin -0.11875296 -0.12825393 -0.1092519889 0.0000000  
## rap-latin -0.05738494 -0.06672297 -0.0480469105 0.0000000  
## rock-latin 0.03085631 0.02083428 0.0408783312 0.0000000  
## r&b-pop -0.10942199 -0.11876742 -0.1000765634 0.0000000  
## rap-pop -0.04805397 -0.05723370 -0.0388742455 0.0000000  
## rock-pop 0.04018728 0.03031259 0.0500619677 0.0000000  
## rap-r&b 0.06136802 0.05209316 0.0706428778 0.0000000  
## rock-r&b 0.14960927 0.13964608 0.1595724573 0.0000000  
## rock-rap 0.08824125 0.07843332 0.0980491802 0.0000000  
##   
## [1] "NEXT>>>>NEXT>>>>NEXT>>>>NEXT>>>>NEXT>>>>NEXT>>>>NEXT>>>>"



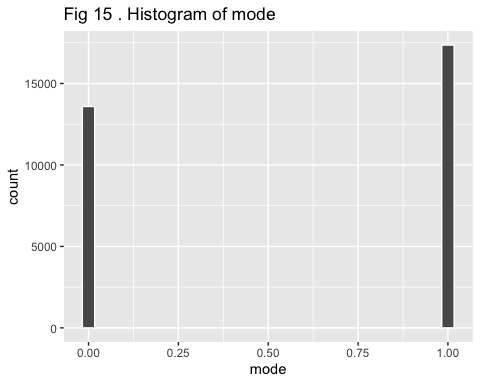


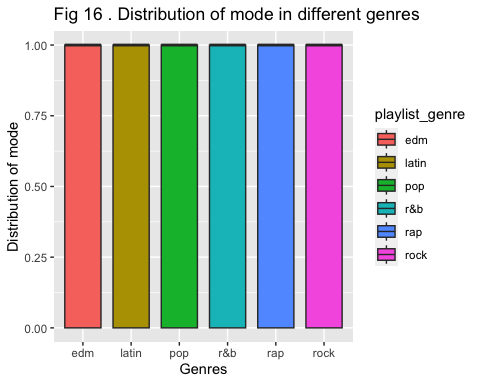
##   
## key  
## edm latin pop r&b rap rock   
## 5.351148 5.481161 5.301150 5.390459 5.455675 5.203279   
## Df Sum Sq Mean Sq F value Pr(>F)   
## playlist\_genre 5 246 49.22 3.77 0.00206 \*\*  
## Residuals 30941 403958 13.06   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
## Tukey multiple comparisons of means  
## 95% family-wise confidence level  
##   
## Fit: aov(formula = as.formula(formula\_str), data = spotify\_songs)  
##   
## $playlist\_genre  
## diff lwr upr p adj  
## latin-edm 0.13001299 -0.06778771 0.32781369 0.4188392  
## pop-edm -0.04999730 -0.24430492 0.14431031 0.9778599  
## r&b-edm 0.03931177 -0.15709531 0.23571885 0.9929300  
## rap-edm 0.10452778 -0.08819355 0.29724912 0.6345533  
## rock-edm -0.14786812 -0.35602345 0.06028722 0.3282987  
## pop-latin -0.18001030 -0.38337174 0.02335115 0.1175059  
## r&b-latin -0.09070122 -0.29606959 0.11466714 0.8075283  
## rap-latin -0.02548521 -0.22733154 0.17636112 0.9992140  
## rock-latin -0.27788111 -0.49451229 -0.06124993 0.0034988  
## r&b-pop 0.08930907 -0.11269712 0.29131526 0.8068408  
## rap-pop 0.15452509 -0.04389939 0.35294957 0.2287320  
## rock-pop -0.09787081 -0.31131730 0.11557567 0.7815225  
## rap-r&b 0.06521602 -0.13526482 0.26569685 0.9396816  
## rock-r&b -0.18717988 -0.40253934 0.02817957 0.1309320  
## rock-rap -0.25239590 -0.46439936 -0.04039244 0.0090504  
##   
## [1] "NEXT>>>>NEXT>>>>NEXT>>>>NEXT>>>>NEXT>>>>NEXT>>>>NEXT>>>>"



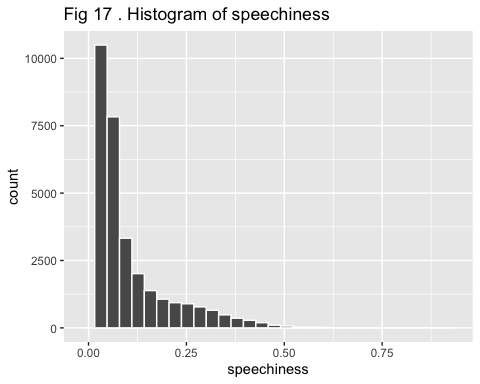


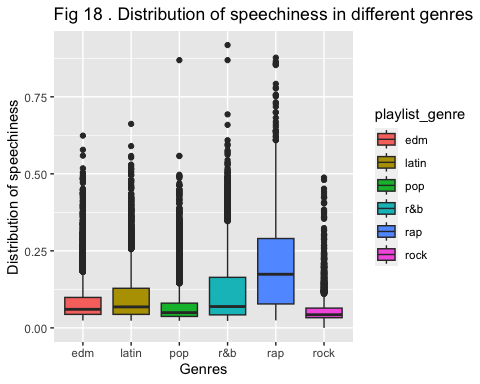
##   
## loudness  
## edm latin pop r&b rap rock   
## -5.411837 -6.222340 -6.300329 -7.807318 -7.013525 -7.410332   
## Df Sum Sq Mean Sq F value Pr(>F)   
## playlist\_genre 5 20647 4129 514.2 <2e-16 \*\*\*  
## Residuals 30941 248461 8   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
## Tukey multiple comparisons of means  
## 95% family-wise confidence level  
##   
## Fit: aov(formula = as.formula(formula\_str), data = spotify\_songs)  
##   
## $playlist\_genre  
## diff lwr upr p adj  
## latin-edm -0.81050326 -0.9656307 -0.65537584 0.000000  
## pop-edm -0.88849278 -1.0408807 -0.73610485 0.000000  
## r&b-edm -2.39548156 -2.5495160 -2.24144711 0.000000  
## rap-edm -1.60168884 -1.7528327 -1.45054497 0.000000  
## rock-edm -1.99849539 -2.1617435 -1.83524724 0.000000  
## pop-latin -0.07798952 -0.2374780 0.08149897 0.731055  
## r&b-latin -1.58497830 -1.7460407 -1.42391586 0.000000  
## rap-latin -0.79118558 -0.9494858 -0.63288533 0.000000  
## rock-latin -1.18799213 -1.3578876 -1.01809670 0.000000  
## r&b-pop -1.50698878 -1.6654144 -1.34856316 0.000000  
## rap-pop -0.71319606 -0.8688127 -0.55757944 0.000000  
## rock-pop -1.11000261 -1.2774004 -0.94260481 0.000000  
## rap-r&b 0.79379272 0.6365634 0.95102206 0.000000  
## rock-r&b 0.39698617 0.2280881 0.56588423 0.000000  
## rock-rap -0.39680655 -0.5630726 -0.23054046 0.000000  
##   
## [1] "NEXT>>>>NEXT>>>>NEXT>>>>NEXT>>>>NEXT>>>>NEXT>>>>NEXT>>>>"



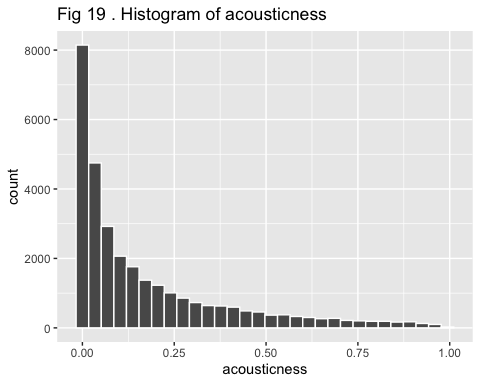


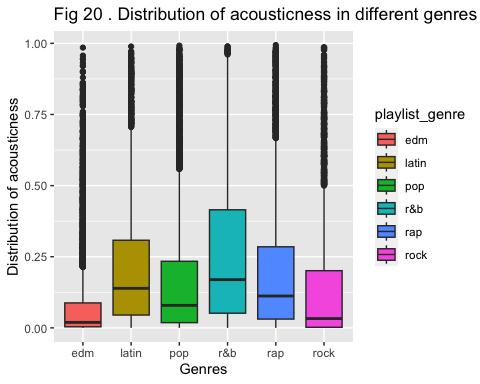
##   
## mode  
## edm latin pop r&b rap rock   
## 0.5191824 0.5617570 0.5855176 0.5188457 0.5189179 0.6956836   
## Df Sum Sq Mean Sq F value Pr(>F)   
## playlist\_genre 5 108 21.519 88.61 <2e-16 \*\*\*  
## Residuals 30941 7514 0.243   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
## Tukey multiple comparisons of means  
## 95% family-wise confidence level  
##   
## Fit: aov(formula = as.formula(formula\_str), data = spotify\_songs)  
##   
## $playlist\_genre  
## diff lwr upr p adj  
## latin-edm 4.257456e-02 0.015596958 0.06955216 0.0001005  
## pop-edm 6.633519e-02 0.039834002 0.09283638 0.0000000  
## r&b-edm -3.367418e-04 -0.027124270 0.02645079 1.0000000  
## rap-edm -2.645117e-04 -0.026549350 0.02602033 1.0000000  
## rock-edm 1.765012e-01 0.148111338 0.20489103 0.0000000  
## pop-latin 2.376063e-02 -0.003975389 0.05149665 0.1421710  
## r&b-latin -4.291130e-02 -0.070921040 -0.01490156 0.0001838  
## rap-latin -4.283907e-02 -0.070368446 -0.01530970 0.0001344  
## rock-latin 1.339266e-01 0.104380776 0.16347247 0.0000000  
## r&b-pop -6.667193e-02 -0.094223109 -0.03912075 0.0000000  
## rap-pop -6.659970e-02 -0.093662378 -0.03953702 0.0000000  
## rock-pop 1.101660e-01 0.081054500 0.13927749 0.0000000  
## rap-r&b 7.223008e-05 -0.027270908 0.02741537 1.0000000  
## rock-r&b 1.768379e-01 0.147465526 0.20621033 0.0000000  
## rock-rap 1.767657e-01 0.147851012 0.20568038 0.0000000  
##   
## [1] "NEXT>>>>NEXT>>>>NEXT>>>>NEXT>>>>NEXT>>>>NEXT>>>>NEXT>>>>"



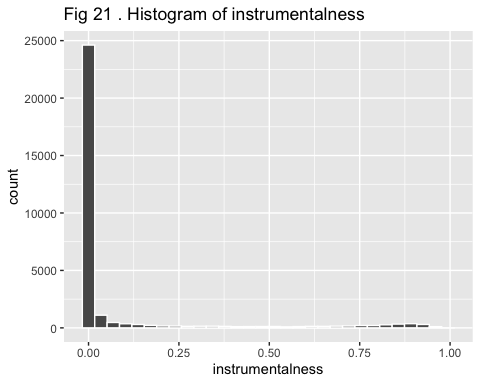


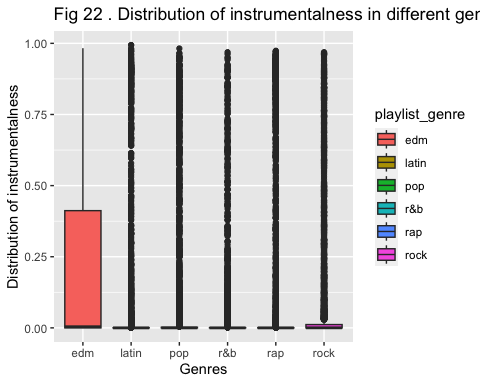
##   
## speechiness  
## edm latin pop r&b rap rock   
## 0.08687212 0.10327175 0.07482399 0.11890188 0.19613449 0.05865602   
## Df Sum Sq Mean Sq F value Pr(>F)   
## playlist\_genre 5 61.81 12.362 1478 <2e-16 \*\*\*  
## Residuals 30941 258.76 0.008   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
## Tukey multiple comparisons of means  
## 95% family-wise confidence level  
##   
## Fit: aov(formula = as.formula(formula\_str), data = spotify\_songs)  
##   
## $playlist\_genre  
## diff lwr upr p adj  
## latin-edm 0.01639963 0.01139338 0.02140587 0  
## pop-edm -0.01204814 -0.01696597 -0.00713030 0  
## r&b-edm 0.03202976 0.02705879 0.03700073 0  
## rap-edm 0.10926237 0.10438468 0.11414006 0  
## rock-edm -0.02821611 -0.03348442 -0.02294779 0  
## pop-latin -0.02844776 -0.03359475 -0.02330078 0  
## r&b-latin 0.01563013 0.01043236 0.02082791 0  
## rap-latin 0.09286274 0.08775410 0.09797138 0  
## rock-latin -0.04461573 -0.05009857 -0.03913290 0  
## r&b-pop 0.04407790 0.03896521 0.04919058 0  
## rap-pop 0.12131050 0.11628847 0.12633254 0  
## rock-pop -0.01616797 -0.02157020 -0.01076574 0  
## rap-r&b 0.07723261 0.07215853 0.08230668 0  
## rock-r&b -0.06024587 -0.06569652 -0.05479522 0  
## rock-rap -0.13747847 -0.14284418 -0.13211276 0  
##   
## [1] "NEXT>>>>NEXT>>>>NEXT>>>>NEXT>>>>NEXT>>>>NEXT>>>>NEXT>>>>"



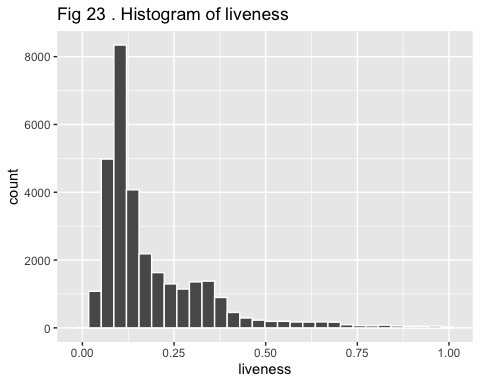


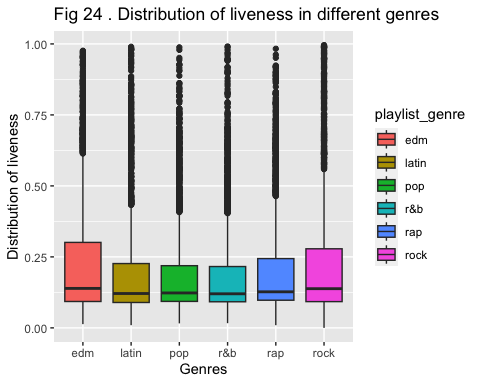
##   
## acousticness  
## edm latin pop r&b rap rock   
## 0.0816723 0.2110486 0.1730262 0.2629757 0.1959889 0.1400041   
## Df Sum Sq Mean Sq F value Pr(>F)   
## playlist\_genre 5 105.3 21.070 468 <2e-16 \*\*\*  
## Residuals 30941 1392.9 0.045   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
## Tukey multiple comparisons of means  
## 95% family-wise confidence level  
##   
## Fit: aov(formula = as.formula(formula\_str), data = spotify\_songs)  
##   
## $playlist\_genre  
## diff lwr upr p adj  
## latin-edm 0.12937634 0.11776115 0.140991532 0.0000000  
## pop-edm 0.09135387 0.07994379 0.102763938 0.0000000  
## r&b-edm 0.18130339 0.16977004 0.192836749 0.0000000  
## rap-edm 0.11431655 0.10299963 0.125633477 0.0000000  
## rock-edm 0.05833177 0.04610854 0.070555005 0.0000000  
## pop-latin -0.03802247 -0.04996420 -0.026080746 0.0000000  
## r&b-latin 0.05192705 0.03986748 0.063986631 0.0000000  
## rap-latin -0.01505979 -0.02691254 -0.003207029 0.0039839  
## rock-latin -0.07104457 -0.08376552 -0.058323620 0.0000000  
## r&b-pop 0.08994953 0.07808738 0.101811672 0.0000000  
## rap-pop 0.02296269 0.01131087 0.034614509 0.0000003  
## rock-pop -0.03302209 -0.04555603 -0.020488156 0.0000000  
## rap-r&b -0.06698684 -0.07875941 -0.055214266 0.0000000  
## rock-r&b -0.12297162 -0.13561789 -0.110325351 0.0000000  
## rock-rap -0.05598478 -0.06843398 -0.043535581 0.0000000  
##   
## [1] "NEXT>>>>NEXT>>>>NEXT>>>>NEXT>>>>NEXT>>>>NEXT>>>>NEXT>>>>"



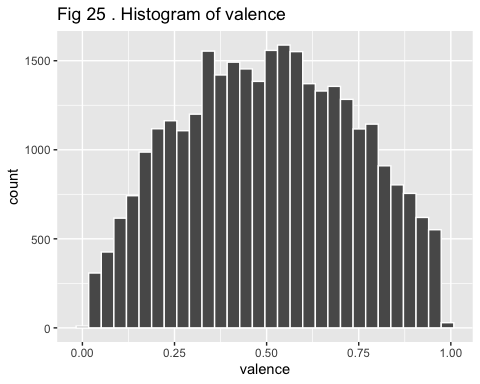


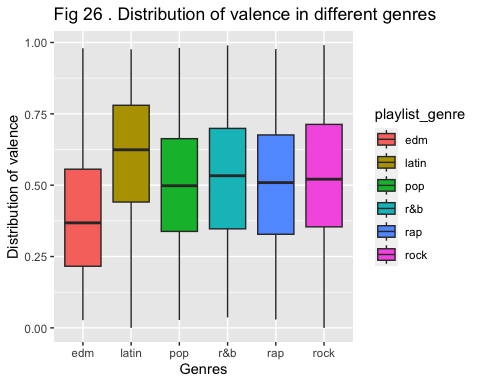
##   
## instrumentalness  
## edm latin pop r&b rap rock   
## 0.21881824 0.04495271 0.05858493 0.02913776 0.07864474 0.06558976   
## Df Sum Sq Mean Sq F value Pr(>F)   
## playlist\_genre 5 136.1 27.222 575.3 <2e-16 \*\*\*  
## Residuals 30941 1464.0 0.047   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
## Tukey multiple comparisons of means  
## 95% family-wise confidence level  
##   
## Fit: aov(formula = as.formula(formula\_str), data = spotify\_songs)  
##   
## $playlist\_genre  
## diff lwr upr p adj  
## latin-edm -0.173865530 -0.185773460 -0.1619575999 0.0000000  
## pop-edm -0.160233306 -0.171930947 -0.1485356657 0.0000000  
## r&b-edm -0.189680485 -0.201504516 -0.1778564528 0.0000000  
## rap-edm -0.140173500 -0.151775645 -0.1285713561 0.0000000  
## rock-edm -0.153228478 -0.165759774 -0.1406971814 0.0000000  
## pop-latin 0.013632224 0.001389527 0.0258749198 0.0188423  
## r&b-latin -0.015814955 -0.028178471 -0.0034514381 0.0036354  
## rap-latin 0.033692030 0.021540546 0.0458435136 0.0000000  
## rock-latin 0.020637052 0.007595496 0.0336786089 0.0000950  
## r&b-pop -0.029447178 -0.041608286 -0.0172860705 0.0000000  
## rap-pop 0.020059806 0.008114323 0.0320052890 0.0000252  
## rock-pop 0.007004829 -0.005845004 0.0198546614 0.6293906  
## rap-r&b 0.049506984 0.037437706 0.0615762630 0.0000000  
## rock-r&b 0.036452007 0.023487011 0.0494170031 0.0000000  
## rock-rap -0.013054977 -0.025817937 -0.0002920174 0.0414948  
##   
## [1] "NEXT>>>>NEXT>>>>NEXT>>>>NEXT>>>>NEXT>>>>NEXT>>>>NEXT>>>>"



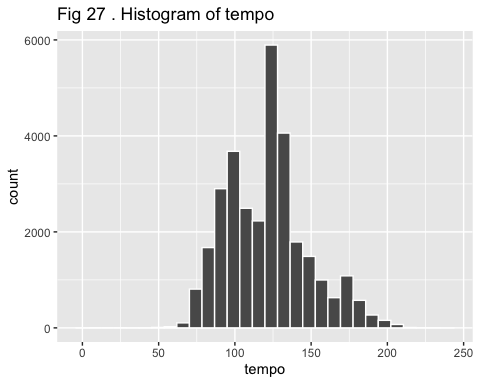


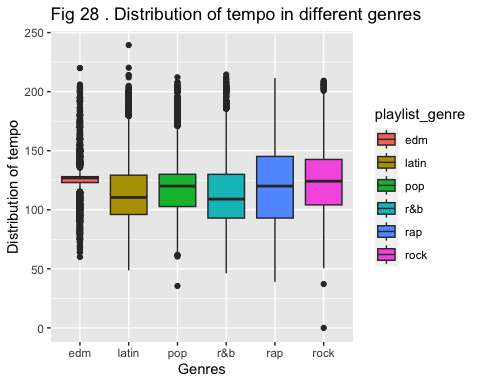
##   
## liveness  
## edm latin pop r&b rap rock   
## 0.2125270 0.1803575 0.1767067 0.1749040 0.1899724 0.2045341   
## Df Sum Sq Mean Sq F value Pr(>F)   
## playlist\_genre 5 6.5 1.2929 55.04 <2e-16 \*\*\*  
## Residuals 30941 726.9 0.0235   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
## Tukey multiple comparisons of means  
## 95% family-wise confidence level  
##   
## Fit: aov(formula = as.formula(formula\_str), data = spotify\_songs)  
##   
## $playlist\_genre  
## diff lwr upr p adj  
## latin-edm -0.032169489 -0.040559929 -0.0237790481 0.0000000  
## pop-edm -0.035820312 -0.044062580 -0.0275780433 0.0000000  
## r&b-edm -0.037622962 -0.045954287 -0.0292916373 0.0000000  
## rap-edm -0.022554632 -0.030729613 -0.0143796511 0.0000000  
## rock-edm -0.007992933 -0.016822604 0.0008367368 0.1022232  
## pop-latin -0.003650823 -0.012277143 0.0049754965 0.8341357  
## r&b-latin -0.005453474 -0.014164925 0.0032579772 0.4761509  
## rap-latin 0.009614857 0.001052806 0.0181769074 0.0172714  
## rock-latin 0.024176555 0.014987351 0.0333657595 0.0000000  
## r&b-pop -0.001802650 -0.010371482 0.0067661814 0.9910974  
## rap-pop 0.013265680 0.004848780 0.0216825805 0.0001033  
## rock-pop 0.027827378 0.018773265 0.0368814923 0.0000000  
## rap-r&b 0.015068330 0.006564202 0.0235724586 0.0000066  
## rock-r&b 0.029630029 0.020494770 0.0387652880 0.0000000  
## rock-rap 0.014561698 0.005568796 0.0235546010 0.0000579  
##   
## [1] "NEXT>>>>NEXT>>>>NEXT>>>>NEXT>>>>NEXT>>>>NEXT>>>>NEXT>>>>"



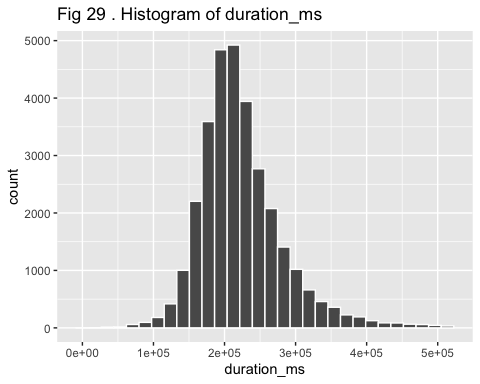


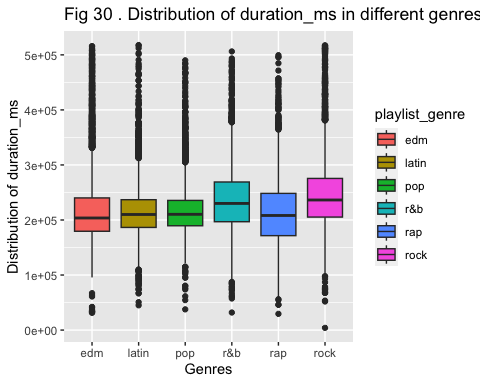
##   
## valence  
## edm latin pop r&b rap rock   
## 0.3984501 0.6027073 0.5004120 0.5236551 0.5000706 0.5311139   
## Df Sum Sq Mean Sq F value Pr(>F)   
## playlist\_genre 5 120 24.00 477.1 <2e-16 \*\*\*  
## Residuals 30941 1556 0.05   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
## Tukey multiple comparisons of means  
## 95% family-wise confidence level  
##   
## Fit: aov(formula = as.formula(formula\_str), data = spotify\_songs)  
##   
## $playlist\_genre  
## diff lwr upr p adj  
## latin-edm 0.2042571750 0.191979625 0.21653472 0.0000000  
## pop-edm 0.1019619220 0.089901189 0.11402265 0.0000000  
## r&b-edm 0.1252050148 0.113013968 0.13739606 0.0000000  
## rap-edm 0.1016204632 0.089658191 0.11358274 0.0000000  
## rock-edm 0.1326637807 0.119743516 0.14558405 0.0000000  
## pop-latin -0.1022952530 -0.114917960 -0.08967255 0.0000000  
## r&b-latin -0.0790521602 -0.091799437 -0.06630488 0.0000000  
## rap-latin -0.1026367118 -0.115165375 -0.09010805 0.0000000  
## rock-latin -0.0715933943 -0.085039758 -0.05814703 0.0000000  
## r&b-pop 0.0232430927 0.010704507 0.03578168 0.0000019  
## rap-pop -0.0003414588 -0.012657727 0.01197481 0.9999996  
## rock-pop 0.0307018586 0.017453170 0.04395055 0.0000000  
## rap-r&b -0.0235845516 -0.036028458 -0.01114065 0.0000010  
## rock-r&b 0.0074587659 -0.005908661 0.02082619 0.6051412  
## rock-rap 0.0310433175 0.017884198 0.04420244 0.0000000  
##   
## [1] "NEXT>>>>NEXT>>>>NEXT>>>>NEXT>>>>NEXT>>>>NEXT>>>>NEXT>>>>"



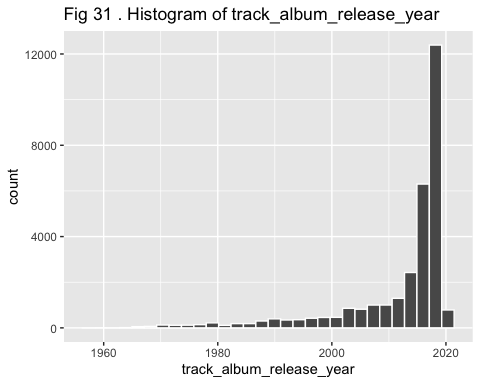


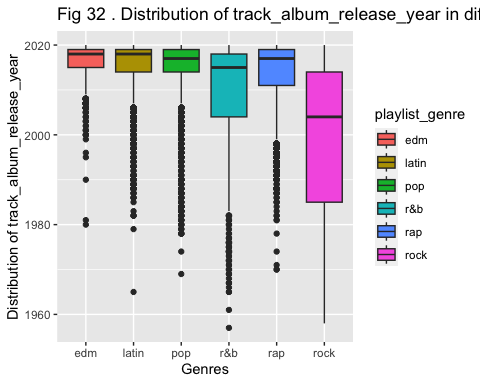
##   
## tempo  
## edm latin pop r&b rap rock   
## 125.7464 118.7115 120.4868 114.2492 121.1452 125.2136   
## Df Sum Sq Mean Sq F value Pr(>F)   
## playlist\_genre 5 467636 93527 132.5 <2e-16 \*\*\*  
## Residuals 30941 21841480 706   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
## Tukey multiple comparisons of means  
## 95% family-wise confidence level  
##   
## Fit: aov(formula = as.formula(formula\_str), data = spotify\_songs)  
##   
## $playlist\_genre  
## diff lwr upr p adj  
## latin-edm -7.0348422 -8.4892986 -5.5803857 0.0000000  
## pop-edm -5.2595762 -6.6883475 -3.8308049 0.0000000  
## r&b-edm -11.4971479 -12.9413568 -10.0529390 0.0000000  
## rap-edm -4.6011697 -6.0182768 -3.1840625 0.0000000  
## rock-edm -0.5328016 -2.0633971 0.9977939 0.9206764  
## pop-latin 1.7752660 0.2799206 3.2706113 0.0093529  
## r&b-latin -4.4623057 -5.9724083 -2.9522032 0.0000000  
## rap-latin 2.4336725 0.9494680 3.9178770 0.0000437  
## rock-latin 6.5020406 4.9091209 8.0949602 0.0000000  
## r&b-pop -6.2375717 -7.7229517 -4.7521917 0.0000000  
## rap-pop 0.6584065 -0.8006366 2.1174497 0.7929058  
## rock-pop 4.7267746 3.1572725 6.2962767 0.0000000  
## rap-r&b 6.8959782 5.4218144 8.3701420 0.0000000  
## rock-r&b 10.9643463 9.3807779 12.5479147 0.0000000  
## rock-rap 4.0683681 2.5094767 5.6272594 0.0000000  
##   
## [1] "NEXT>>>>NEXT>>>>NEXT>>>>NEXT>>>>NEXT>>>>NEXT>>>>NEXT>>>>"





##   
## duration\_ms  
## edm latin pop r&b rap rock   
## 221647.9 215760.6 216827.5 236093.9 211773.5 247319.9   
## Df Sum Sq Mean Sq F value Pr(>F)   
## playlist\_genre 5 4.461e+12 8.922e+11 266.2 <2e-16 \*\*\*  
## Residuals 30941 1.037e+14 3.351e+09   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
## Tukey multiple comparisons of means  
## 95% family-wise confidence level  
##   
## Fit: aov(formula = as.formula(formula\_str), data = spotify\_songs)  
##   
## $playlist\_genre  
## diff lwr upr p adj  
## latin-edm -5887.289 -9056.156 -2718.422 0.0000018  
## pop-edm -4820.402 -7933.309 -1707.496 0.0001484  
## r&b-edm 14445.948 11299.407 17592.489 0.0000000  
## rap-edm -9874.436 -12961.930 -6786.943 0.0000000  
## rock-edm 25671.992 22337.238 29006.746 0.0000000  
## pop-latin 1066.887 -2191.067 4324.840 0.9380135  
## r&b-latin 20333.237 17043.131 23623.342 0.0000000  
## rap-latin -3987.148 -7220.828 -753.467 0.0058983  
## rock-latin 31559.281 28088.740 35029.823 0.0000000  
## r&b-pop 19266.350 16030.108 22502.591 0.0000000  
## rap-pop -5054.034 -8232.895 -1875.174 0.0000860  
## rock-pop 30492.394 27072.874 33911.915 0.0000000  
## rap-r&b -24320.384 -27532.189 -21108.580 0.0000000  
## rock-r&b 11226.045 7775.877 14676.212 0.0000000  
## rock-rap 35546.429 32150.026 38942.832 0.0000000  
##   
## [1] "NEXT>>>>NEXT>>>>NEXT>>>>NEXT>>>>NEXT>>>>NEXT>>>>NEXT>>>>"

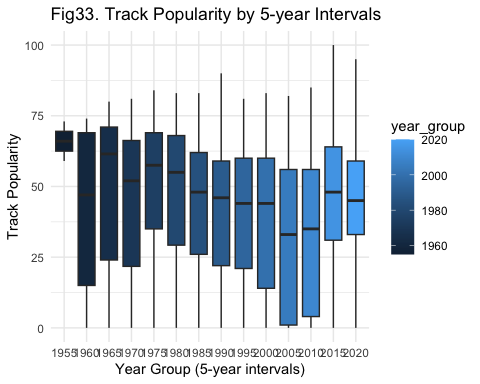




##   
## track\_album\_release\_year  
## edm latin pop r&b rap rock   
## 2016.780 2015.176 2014.843 2010.361 2013.411 1999.331   
## Df Sum Sq Mean Sq F value Pr(>F)   
## playlist\_genre 5 918246 183649 2341 <2e-16 \*\*\*  
## Residuals 30941 2427434 78   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
## Tukey multiple comparisons of means  
## 95% family-wise confidence level  
##   
## Fit: aov(formula = as.formula(formula\_str), data = spotify\_songs)  
##   
## $playlist\_genre  
## diff lwr upr p adj  
## latin-edm -1.6037595 -2.0886385 -1.1188805 0.0000000  
## pop-edm -1.9365664 -2.4128826 -1.4602501 0.0000000  
## r&b-edm -6.4188497 -6.9003124 -5.9373869 0.0000000  
## rap-edm -3.3686033 -3.8410310 -2.8961755 0.0000000  
## rock-edm -17.4487799 -17.9590418 -16.9385181 0.0000000  
## pop-latin -0.3328069 -0.8313173 0.1657034 0.4005061  
## r&b-latin -4.8150902 -5.3185202 -4.3116602 0.0000000  
## rap-latin -1.7648438 -2.2596401 -1.2700475 0.0000000  
## rock-latin -15.8450205 -16.3760596 -15.3139813 0.0000000  
## r&b-pop -4.4822833 -4.9774714 -3.9870952 0.0000000  
## rap-pop -1.4320369 -1.9184450 -0.9456288 0.0000000  
## rock-pop -15.5122135 -16.0354459 -14.9889812 0.0000000  
## rap-r&b 3.0502464 2.5587975 3.5416954 0.0000000  
## rock-r&b -11.0299302 -11.5578519 -10.5020086 0.0000000  
## rock-rap -14.0801767 -14.5998716 -13.5604817 0.0000000  
##   
## [1] "NEXT>>>>NEXT>>>>NEXT>>>>NEXT>>>>NEXT>>>>NEXT>>>>NEXT>>>>"

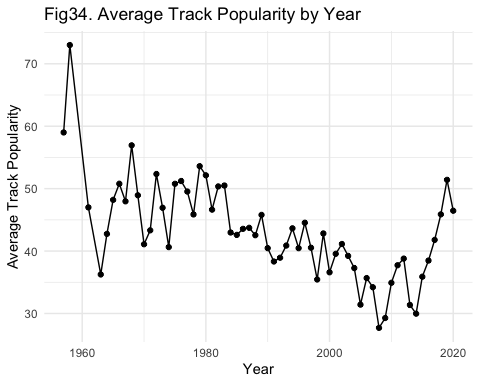
How does track popularity change over time

spotify\_songs <- spotify\_songs %>%  
 mutate(year\_group = 5 \* (track\_album\_release\_year %/% 5))  
  
spotify\_songs %>%  
 ggplot(aes(x = factor(year\_group),   
 y = track\_popularity,  
 fill = year\_group)) +  
 geom\_boxplot() +  
 xlab("Year Group (5-year intervals)") +  
 ylab("Track Popularity") +  
 ggtitle("Fig33. Track Popularity by 5-year Intervals") +  
 theme\_minimal()

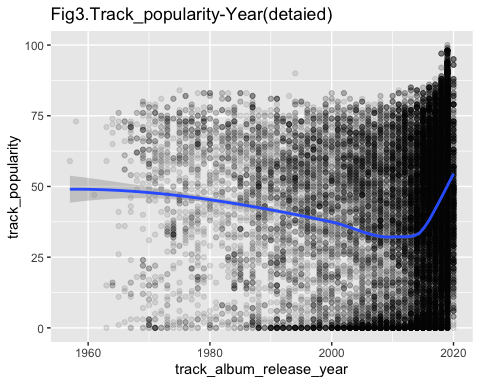


spotify\_songs <- spotify\_songs %>%   
 dplyr::select(-year\_group)

spotify\_songs %>%  
 group\_by(track\_album\_release\_year) %>%  
 summarize(avg\_popularity = mean(track\_popularity)) %>%  
 ggplot(aes(x = track\_album\_release\_year, y = avg\_popularity)) +  
 geom\_point() +  
 geom\_line() +  
 xlab("Year") +  
 ylab("Average Track Popularity") +  
 ggtitle("Fig34. Average Track Popularity by Year") +  
 theme\_minimal()



spotify\_songs %>% ggplot(aes(x = track\_album\_release\_year, y =track\_popularity)) +  
 geom\_point(alpha = 0.1) +  
 geom\_smooth( formula = y ~ x, method = "loess") +  
 ggtitle("Fig3.Track\_popularity-Year(detaied)")



linear\_model <- lm(track\_popularity ~track\_album\_release\_year,   
 data = spotify\_songs)  
  
summary(linear\_model)

##   
## Call:  
## lm(formula = track\_popularity ~ track\_album\_release\_year, data = spotify\_songs)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -43.998 -18.037 2.524 19.364 56.162   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -279.38805 27.39308 -10.20 <2e-16 \*\*\*  
## track\_album\_release\_year 0.16009 0.01361 11.76 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 24.9 on 30945 degrees of freedom  
## Multiple R-squared: 0.004449, Adjusted R-squared: 0.004417   
## F-statistic: 138.3 on 1 and 30945 DF, p-value: < 2.2e-16

**PCA**

spotify\_songs\_recipe\_PCA <- recipe( playlist\_genre ~ ., data = spotify\_songs ) %>%   
 step\_normalize(all\_predictors() ) %>% # Normalise our predictors   
 step\_pca(all\_predictors() ) # Do the PCA.  
spotify\_songs\_recipe\_PCA

##

## ── Recipe ──────────────────────────────────────────────────────────────────────

##

## ── Inputs

## Number of variables by role

## outcome: 1  
## predictor: 14

##

## ── Operations

## • Centering and scaling for: all\_predictors()

## • PCA extraction with: all\_predictors()

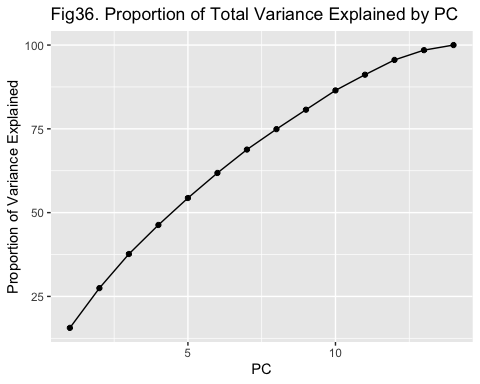
spotify\_songs\_prepped <- spotify\_songs\_recipe\_PCA %>% prep()   
tidy(spotify\_songs\_prepped)

## # A tibble: 2 × 6  
## number operation type trained skip id   
## <int> <chr> <chr> <lgl> <lgl> <chr>   
## 1 1 step normalize TRUE FALSE normalize\_aDMUP  
## 2 2 step pca TRUE FALSE pca\_41mbw

sdev <- spotify\_songs\_prepped$steps[[2]]$res$sdev  
ve <- sdev^2 / sum(sdev ^2)  
  
variance\_explained <- ve \* 100  
  
sorted\_var\_explained <- sort(variance\_explained, decreasing = TRUE)  
  
cumulative\_var\_explained <- cumsum(sorted\_var\_explained)  
PC\_CUM <- data.frame(PC = 1:length(cumulative\_var\_explained),   
 cumulative\_var\_explained = cumulative\_var\_explained)  
PC\_CUM %>% filter(cumulative\_var\_explained > 95)

## PC cumulative\_var\_explained  
## 1 12 95.56142  
## 2 13 98.48400  
## 3 14 100.00000

PC\_CUM %>% ggplot(aes(x = PC , y = cumulative\_var\_explained)) + geom\_point() + geom\_line() +  
 ggtitle("Fig36. Proportion of Total Variance Explained by PC") + xlab("PC") +  
 ylab("Proportion of Variance Explained")



## 1.4 Modeling

### Sampling to reduce data set size

set.seed(1897402)  
songs\_per\_genre <- 1000  
  
spotify\_songs\_sampled <- spotify\_songs %>%  
 group\_by(playlist\_genre) %>%  
 sample\_n(songs\_per\_genre) %>%  
 ungroup()  
  
skim(spotify\_songs\_sampled)

Data summary

|  |  |
| --- | --- |
| Name | spotify\_songs\_sampled |
| Number of rows | 6000 |
| Number of columns | 15 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |
| Column type frequency: |  |
| factor | 1 |
| numeric | 14 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |
| Group variables | None |

**Variable type: factor**

| skim\_variable | n\_missing | complete\_rate | ordered | n\_unique | top\_counts |
| --- | --- | --- | --- | --- | --- |
| playlist\_genre | 0 | 1 | FALSE | 6 | edm: 1000, lat: 1000, pop: 1000, r&b: 1000 |

**Variable type: numeric**

| skim\_variable | n\_missing | complete\_rate | mean | sd | p0 | p25 | p50 | p75 | p100 | hist |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| track\_popularity | 0 | 1 | 42.77 | 24.69 | 0.00 | 25.00 | 46.00 | 62.00 | 98.00 | ▆▆▇▇▂ |
| danceability | 0 | 1 | 0.65 | 0.15 | 0.08 | 0.56 | 0.67 | 0.76 | 0.97 | ▁▂▅▇▃ |
| energy | 0 | 1 | 0.70 | 0.18 | 0.02 | 0.58 | 0.72 | 0.84 | 1.00 | ▁▁▅▇▇ |
| key | 0 | 1 | 5.42 | 3.63 | 0.00 | 2.00 | 6.00 | 9.00 | 11.00 | ▇▂▅▅▇ |
| loudness | 0 | 1 | -6.70 | 3.02 | -36.51 | -8.17 | -6.14 | -4.60 | 0.64 | ▁▁▁▅▇ |
| mode | 0 | 1 | 0.56 | 0.50 | 0.00 | 0.00 | 1.00 | 1.00 | 1.00 | ▆▁▁▁▇ |
| speechiness | 0 | 1 | 0.11 | 0.10 | 0.02 | 0.04 | 0.06 | 0.13 | 0.86 | ▇▁▁▁▁ |
| acousticness | 0 | 1 | 0.18 | 0.22 | 0.00 | 0.02 | 0.08 | 0.26 | 0.98 | ▇▂▁▁▁ |
| instrumentalness | 0 | 1 | 0.08 | 0.22 | 0.00 | 0.00 | 0.00 | 0.00 | 0.99 | ▇▁▁▁▁ |
| liveness | 0 | 1 | 0.19 | 0.15 | 0.01 | 0.09 | 0.13 | 0.25 | 1.00 | ▇▂▁▁▁ |
| valence | 0 | 1 | 0.51 | 0.23 | 0.00 | 0.33 | 0.52 | 0.70 | 0.98 | ▃▆▇▇▅ |
| tempo | 0 | 1 | 120.79 | 27.19 | 52.65 | 99.90 | 121.04 | 134.03 | 220.25 | ▂▇▇▂▁ |
| duration\_ms | 0 | 1 | 224650.09 | 59104.81 | 31893.00 | 187510.25 | 214323.50 | 252233.25 | 517125.00 | ▁▇▅▁▁ |
| track\_album\_release\_year | 0 | 1 | 2011.48 | 11.26 | 1958.00 | 2009.00 | 2016.00 | 2019.00 | 2020.00 | ▁▁▁▁▇ |

## Split Train and Test

set.seed(1897402)  
spotify\_songs\_split <- initial\_split(spotify\_songs\_sampled,   
 strata = playlist\_genre)   
spotify\_songs\_split

## <Training/Testing/Total>  
## <4500/1500/6000>

spotify\_songs\_train <- training(spotify\_songs\_split)  
spotify\_songs\_test <- testing(spotify\_songs\_split)  
head(spotify\_songs\_train)

## # A tibble: 6 × 15  
## track\_pop…¹ playl…² dance…³ energy key loudn…⁴ mode speec…⁵ acous…⁶ instr…⁷  
## <dbl> <fct> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 29 edm 0.478 0.904 0 -5.41 0 0.0648 0.00221 0   
## 2 48 edm 0.355 0.993 5 -0.627 0 0.221 0.0108 1.13e-1  
## 3 71 edm 0.687 0.707 4 -6.19 1 0.0328 0.0536 9.29e-5  
## 4 55 edm 0.891 0.794 3 -3.44 0 0.0526 0.383 1.99e-4  
## 5 46 edm 0.755 0.83 8 -6.34 0 0.124 0.104 6.6 e-1  
## 6 37 edm 0.893 0.748 7 -5.60 0 0.051 0.311 2.59e-5  
## # … with 5 more variables: liveness <dbl>, valence <dbl>, tempo <dbl>,  
## # duration\_ms <dbl>, track\_album\_release\_year <dbl>, and abbreviated variable  
## # names ¹​track\_popularity, ²​playlist\_genre, ³​danceability, ⁴​loudness,  
## # ⁵​speechiness, ⁶​acousticness, ⁷​instrumentalness

spotify\_songs\_train %>% count(playlist\_genre) %>%  
 mutate(percent = prop.table(n) \* 100)

## # A tibble: 6 × 3  
## playlist\_genre n percent  
## <fct> <int> <dbl>  
## 1 edm 750 16.7  
## 2 latin 750 16.7  
## 3 pop 750 16.7  
## 4 r&b 750 16.7  
## 5 rap 750 16.7  
## 6 rock 750 16.7

spotify\_songs\_test %>% count(playlist\_genre) %>%  
 mutate(percent = prop.table(n) \* 100)

## # A tibble: 6 × 3  
## playlist\_genre n percent  
## <fct> <int> <dbl>  
## 1 edm 250 16.7  
## 2 latin 250 16.7  
## 3 pop 250 16.7  
## 4 r&b 250 16.7  
## 5 rap 250 16.7  
## 6 rock 250 16.7

# Preprocessing

spotify\_songs\_rcp <- recipe(playlist\_genre ~ ., data = spotify\_songs\_train) %>%  
 step\_zv(all\_predictors()) %>%   
 step\_normalize(all\_predictors()) %>%   
 step\_corr(all\_predictors()) %>%   
 prep()  
  
spotify\_songs\_rcp

##

## ── Recipe ──────────────────────────────────────────────────────────────────────

##

## ── Inputs

## Number of variables by role

## outcome: 1  
## predictor: 14

##

## ── Training information

## Training data contained 4500 data points and no incomplete rows.

##

## ── Operations

## • Zero variance filter removed: <none> | Trained

## • Centering and scaling for: track\_popularity, danceability, ... | Trained

## • Correlation filter on: <none> | Trained

spotify\_train\_prep <- juice(spotify\_songs\_rcp)  
  
spotify\_train\_prep %>% head()

## # A tibble: 6 × 15  
## track\_…¹ dance…² energy key loudn…³ mode speec…⁴ acous…⁵ instr…⁶ liven…⁷  
## <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 -0.554 -1.16 1.13 -1.47 0.428 -1.14 -0.417 -0.795 -0.366 1.42   
## 2 0.214 -1.99 1.61 -0.0991 2.00 -1.14 1.10 -0.757 0.144 0.996  
## 3 1.14 0.244 0.0521 -0.374 0.172 0.878 -0.727 -0.567 -0.365 -0.193  
## 4 0.496 1.62 0.527 -0.649 1.08 -1.14 -0.535 0.893 -0.365 -0.707  
## 5 0.133 0.701 0.724 0.725 0.122 -1.14 0.156 -0.344 2.61 -0.667  
## 6 -0.231 1.63 0.276 0.451 0.367 -1.14 -0.551 0.574 -0.365 1.17   
## # … with 5 more variables: valence <dbl>, tempo <dbl>, duration\_ms <dbl>,  
## # track\_album\_release\_year <dbl>, playlist\_genre <fct>, and abbreviated  
## # variable names ¹​track\_popularity, ²​danceability, ³​loudness, ⁴​speechiness,  
## # ⁵​acousticness, ⁶​instrumentalness, ⁷​liveness

spotify\_test\_prep <- bake(spotify\_songs\_rcp, new\_data = spotify\_songs\_test)  
spotify\_test\_prep %>% head()

## # A tibble: 6 × 15  
## track\_…¹ dance…² energy key loudn…³ mode speec…⁴ acous…⁵ instr…⁶ liven…⁷  
## <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 0.577 -0.570 0.380 0.176 0.185 -1.14 -0.253 -0.335 -0.366 -0.673  
## 2 -0.554 0.876 1.06 -0.374 0.0944 -1.14 0.602 -0.633 0.932 -0.671  
## 3 0.0923 0.607 0.467 -0.374 0.632 0.878 -0.643 -0.739 -0.366 0.730  
## 4 -1.08 0.439 1.52 1.55 1.07 -1.14 0.602 -0.613 -0.197 1.24   
## 5 -1.64 0.432 0.560 -1.20 0.403 -1.14 -0.507 -0.759 -0.366 0.840  
## 6 1.14 -2.14 1.57 -0.0991 1.27 -1.14 -0.500 -0.660 3.10 0.236  
## # … with 5 more variables: valence <dbl>, tempo <dbl>, duration\_ms <dbl>,  
## # track\_album\_release\_year <dbl>, playlist\_genre <fct>, and abbreviated  
## # variable names ¹​track\_popularity, ²​danceability, ³​loudness, ⁴​speechiness,  
## # ⁵​acousticness, ⁶​instrumentalness, ⁷​liveness

set.seed(1897402)  
spotify\_cv <- vfold\_cv(   
 data = spotify\_train\_prep,   
 v = 5,   
 strata = playlist\_genre)  
  
spotify\_cv %>%  
 slice( 1 ) %>%  
 pull( splits )

## [[1]]  
## <Analysis/Assess/Total>  
## <3600/900/4500>

set.seed(1897402)  
spotify\_bootstrap <- bootstraps(spotify\_train\_prep, times = 10, strata = playlist\_genre )

###LDA

lda\_model <- discrim\_linear(mode = "classification") %>%  
 set\_engine("MASS")  
  
spotify\_resamples <- fit\_resamples(  
 object = lda\_model,  
 preprocessor = recipe(playlist\_genre ~ ., data = spotify\_train\_prep),  
 resamples = spotify\_bootstrap,  
 control = control\_resamples(save\_pred = T)  
)  
  
prediction\_lda <- spotify\_resamples %>% collect\_predictions()  
metrics\_lda <- spotify\_resamples %>% collect\_metrics()  
  
lda\_10cfm <- prediction\_lda %>%   
 group\_by(id) %>%  
 conf\_mat(truth = playlist\_genre, estimate = .pred\_class) %>%  
 pull(conf\_mat)  
  
lda\_boost\_result <- spotify\_resamples %>%  
 collect\_predictions() %>%  
 group\_by(id)  
  
confusionMatrix(lda\_boost\_result$.pred\_class, as\_factor(lda\_boost\_result$playlist\_genre))

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction edm latin pop r&b rap rock  
## edm 1550 277 357 103 288 191  
## latin 324 1282 554 381 428 123  
## pop 561 434 1139 413 246 456  
## r&b 114 240 325 869 325 191  
## rap 183 365 162 634 1405 23  
## rock 18 111 205 357 81 1798  
##   
## Overall Statistics  
##   
## Accuracy : 0.4871   
## 95% CI : (0.4794, 0.4947)  
## No Information Rate : 0.1685   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.3846   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Statistics by Class:  
##   
## Class: edm Class: latin Class: pop Class: r&b Class: rap  
## Sensitivity 0.56364 0.47324 0.41539 0.31520 0.50667  
## Specificity 0.91165 0.86888 0.84678 0.91313 0.90051  
## Pos Pred Value 0.56038 0.41462 0.35057 0.42103 0.50685  
## Neg Pred Value 0.91271 0.89367 0.87915 0.86933 0.90044  
## Prevalence 0.16654 0.16405 0.16605 0.16696 0.16793  
## Detection Rate 0.09387 0.07764 0.06898 0.05263 0.08508  
## Detection Prevalence 0.16750 0.18725 0.19675 0.12499 0.16787  
## Balanced Accuracy 0.73764 0.67106 0.63108 0.61416 0.70359  
## Class: rock  
## Sensitivity 0.6463  
## Specificity 0.9438  
## Pos Pred Value 0.6996  
## Neg Pred Value 0.9294  
## Prevalence 0.1685  
## Detection Rate 0.1089  
## Detection Prevalence 0.1556  
## Balanced Accuracy 0.7950

### KNN

knn\_model <- nearest\_neighbor(  
 mode = "classification",  
 neighbors = tune()  
) %>%  
 set\_engine("kknn")  
  
k\_grid <- grid\_regular(  
 levels = 20,  
 neighbors(range = c(1, 100))) %>%  
 as\_tibble()  
  
spotify\_knn\_tune <- tune\_grid(  
 object = knn\_model,  
 resamples = spotify\_bootstrap,  
 grid = k\_grid,  
 preprocessor = recipe(playlist\_genre ~., data = spotify\_train\_prep),  
)  
  
  
best.k <- spotify\_knn\_tune %>%   
 select\_best("accuracy")  
  
best.k

## # A tibble: 1 × 2  
## neighbors .config   
## <int> <chr>   
## 1 89 Preprocessor1\_Model18

knn\_model\_best <-   
 nearest\_neighbor(mode = "classification",   
 neighbors = best.k$neighbors) %>%  
 set\_engine( "kknn" )  
  
knn\_model\_best

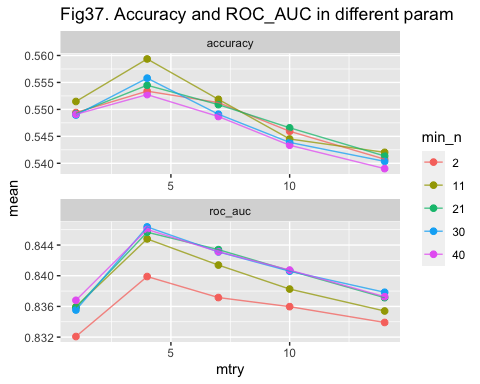
## K-Nearest Neighbor Model Specification (classification)  
##   
## Main Arguments:  
## neighbors = best.k$neighbors  
##   
## Computational engine: kknn

spotify\_knn\_tune %>% collect\_metrics()

## # A tibble: 40 × 7  
## neighbors .metric .estimator mean n std\_err .config   
## <int> <chr> <chr> <dbl> <int> <dbl> <chr>   
## 1 1 accuracy multiclass 0.417 10 0.00317 Preprocessor1\_Model01  
## 2 1 roc\_auc hand\_till 0.650 10 0.00188 Preprocessor1\_Model01  
## 3 6 accuracy multiclass 0.432 10 0.00321 Preprocessor1\_Model02  
## 4 6 roc\_auc hand\_till 0.742 10 0.00226 Preprocessor1\_Model02  
## 5 11 accuracy multiclass 0.451 10 0.00382 Preprocessor1\_Model03  
## 6 11 roc\_auc hand\_till 0.764 10 0.00275 Preprocessor1\_Model03  
## 7 16 accuracy multiclass 0.462 10 0.00432 Preprocessor1\_Model04  
## 8 16 roc\_auc hand\_till 0.776 10 0.00290 Preprocessor1\_Model04  
## 9 21 accuracy multiclass 0.471 10 0.00444 Preprocessor1\_Model05  
## 10 21 roc\_auc hand\_till 0.783 10 0.00283 Preprocessor1\_Model05  
## # … with 30 more rows

### RF

rf\_spec <- rand\_forest(mode = "classification",  
 trees = 100,  
 mtry = tune(),  
 min\_n = tune()) %>%  
 set\_engine("ranger", importance = "permutation")  
  
rf\_grid <- grid\_regular(   
 finalize( mtry(),   
 spotify\_train\_prep%>%   
 dplyr::select( -playlist\_genre ) ),  
 min\_n(),  
 levels = 5 )  
  
set.seed(1897402)  
doParallel::registerDoParallel() # This makes macs run a little faster  
spotify\_rf\_tune <- tune\_grid(object = rf\_spec,  
 preprocessor = recipe(playlist\_genre ~ ., data = spotify\_train\_prep),  
 resamples = spotify\_bootstrap,  
 grid = rf\_grid)  
  
spotify\_rf\_tune %>%   
 collect\_metrics() %>%   
 mutate( min\_n = as.factor( min\_n ) ) %>%   
 ggplot( aes( x = mtry, y = mean, colour = min\_n ) ) +  
 geom\_point( size = 2 ) +  
 geom\_line( alpha = 0.75 ) +  
 ggtitle("Fig37. Accuracy and ROC\_AUC in different param")+  
 facet\_wrap( ~ .metric, scales = "free", nrow = 3 )



best\_rf\_acc <- select\_best(spotify\_rf\_tune, "accuracy" )  
best\_rf\_acc

## # A tibble: 1 × 3  
## mtry min\_n .config   
## <int> <int> <chr>   
## 1 4 11 Preprocessor1\_Model07

rf\_model\_best <- finalize\_model( rf\_spec, best\_rf\_acc )  
rf\_model\_best

## Random Forest Model Specification (classification)  
##   
## Main Arguments:  
## mtry = 4  
## trees = 100  
## min\_n = 11  
##   
## Engine-Specific Arguments:  
## importance = permutation  
##   
## Computational engine: ranger

## 1.5 Model Evaluation Method

### standard model build function

model\_build\_function <- function(model) {  
 model.cv <- fit\_resamples(   
 object = model,  
 preprocessor = recipe(playlist\_genre ~ . ,   
 data = spotify\_train\_prep),  
 resamples = spotify\_cv,   
 control = control\_resamples(save\_pred = T))  
   
 model.metrics <- model.cv %>%   
 collect\_metrics()  
   
 model.prediction <- model.cv %>%   
 collect\_predictions()  
   
 return(list(metrics = model.metrics, prediction = model.prediction))  
}

genre = c(".pred\_edm",".pred\_latin",".pred\_pop",".pred\_r&b",".pred\_rap",".pred\_rock" )  
genrename = c("edm","latin","pop","r&b","rap","rock" )  
  
ROC\_plot <- function(prediction,i,idx,model\_name){  
   
 index = idx+i  
 predictions <-   
 prediction %>%   
 mutate(.pred\_other = 1 - across(genre[i])) %>%   
 mutate(playlist\_genre = ifelse(playlist\_genre == genrename[i],genrename[i], "other")) %>%   
 mutate(playlist\_genre = factor(playlist\_genre, levels = c(genrename[i], "other"))) %>%   
 mutate(.pred\_class = ifelse(.pred\_class == genrename[i],genrename[i],"other"))  
  
 plot <- predictions %>% group\_by( id ) %>%   
 roc\_curve( truth = as\_factor(playlist\_genre), estimate = genre[i] ) %>%   
 ggplot(aes(x= 1-specificity, y = sensitivity)) + geom\_point(alpha = 0.2, color = "red") +  
 ggtitle(paste("Fig",index,". ROC curve of ", genrename[i],"for ", model\_name)) + theme\_minimal()  
 return(plot)  
}  
  
AUC\_value <- function(prediction,i){  
   
 predictions <-   
 prediction %>%   
 mutate(.pred\_other = 1 - across(genre[i])) %>%   
 mutate(playlist\_genre = ifelse(playlist\_genre == genrename[i],genrename[i], "other")) %>%   
 mutate(playlist\_genre = factor(playlist\_genre, levels = c(genrename[i], "other"))) %>%   
 mutate(.pred\_class = ifelse(.pred\_class == genrename[i],genrename[i],"other"))  
  
 roc\_data <- predictions %>% group\_by( id ) %>%   
 roc\_curve( truth = as\_factor(playlist\_genre), estimate = genre[i])  
 return(roc\_data)  
}

### LDA

lda.cv.model <- model\_build\_function(lda\_model)  
lda.prediction <- lda.cv.model$prediction  
  
lda.cv.model$metrics

## # A tibble: 2 × 6  
## .metric .estimator mean n std\_err .config   
## <chr> <chr> <dbl> <int> <dbl> <chr>   
## 1 accuracy multiclass 0.489 5 0.00338 Preprocessor1\_Model1  
## 2 roc\_auc hand\_till 0.806 5 0.00204 Preprocessor1\_Model1

lda.cfm <- confusionMatrix(lda.prediction$.pred\_class, as\_factor(lda.prediction$playlist\_genre))  
lda.cfm

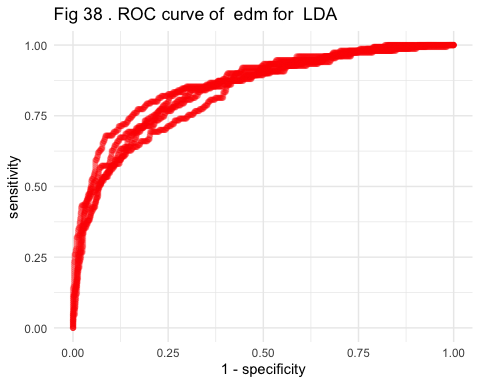
## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction edm latin pop r&b rap rock  
## edm 418 72 94 31 78 53  
## latin 87 360 147 103 108 31  
## pop 159 120 313 109 69 123  
## r&b 30 64 93 238 82 53  
## rap 52 102 45 171 389 6  
## rock 4 32 58 98 24 484  
##   
## Overall Statistics  
##   
## Accuracy : 0.4893   
## 95% CI : (0.4746, 0.5041)  
## No Information Rate : 0.1667   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.3872   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Statistics by Class:  
##   
## Class: edm Class: latin Class: pop Class: r&b Class: rap  
## Sensitivity 0.55733 0.4800 0.41733 0.31733 0.51867  
## Specificity 0.91253 0.8731 0.84533 0.91413 0.89973  
## Pos Pred Value 0.56032 0.4306 0.35050 0.42500 0.50850  
## Neg Pred Value 0.91156 0.8936 0.87885 0.87005 0.90335  
## Prevalence 0.16667 0.1667 0.16667 0.16667 0.16667  
## Detection Rate 0.09289 0.0800 0.06956 0.05289 0.08644  
## Detection Prevalence 0.16578 0.1858 0.19844 0.12444 0.17000  
## Balanced Accuracy 0.73493 0.6765 0.63133 0.61573 0.70920  
## Class: rock  
## Sensitivity 0.6453  
## Specificity 0.9424  
## Pos Pred Value 0.6914  
## Neg Pred Value 0.9300  
## Prevalence 0.1667  
## Detection Rate 0.1076  
## Detection Prevalence 0.1556  
## Balanced Accuracy 0.7939

plot <- lapply(1:6, function(x) (ROC\_plot(lda.prediction,x,37,"LDA")))

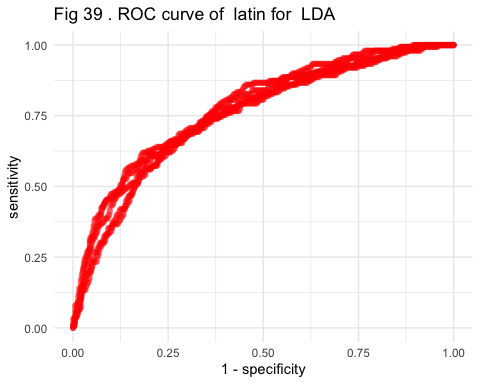
## Warning: Returning more (or less) than 1 row per `summarise()` group was deprecated in  
## dplyr 1.1.0.  
## ℹ Please use `reframe()` instead.  
## ℹ When switching from `summarise()` to `reframe()`, remember that `reframe()`  
## always returns an ungrouped data frame and adjust accordingly.  
## ℹ The deprecated feature was likely used in the yardstick package.  
## Please report the issue at <]8;;https://github.com/tidymodels/yardstick/issueshttps://github.com/tidymodels/yardstick/issues]8;;>.

print(plot)

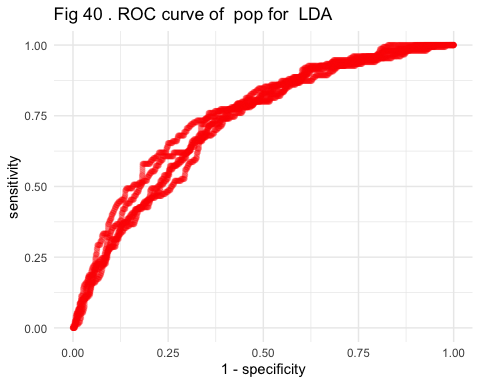
## [[1]]



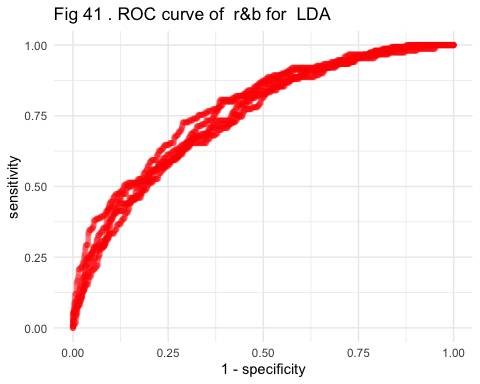
##   
## [[2]]



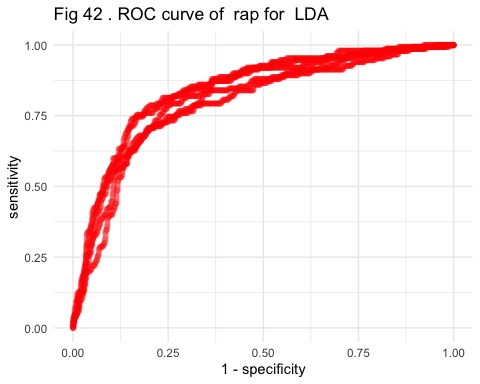
##   
## [[3]]



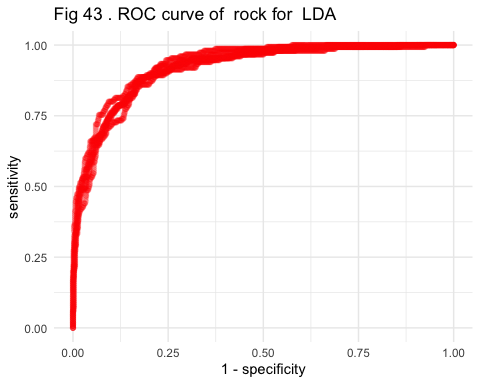
##   
## [[4]]



##   
## [[5]]



##   
## [[6]]

 ### KNN

knn.cv.model <- model\_build\_function(knn\_model\_best)  
knn.prediction <- knn.cv.model$prediction  
  
knn.cv.model$metrics

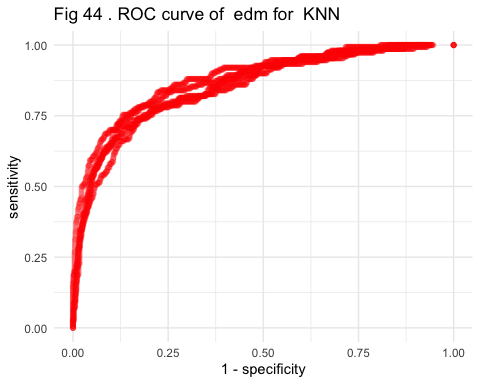
## # A tibble: 2 × 6  
## .metric .estimator mean n std\_err .config   
## <chr> <chr> <dbl> <int> <dbl> <chr>   
## 1 accuracy multiclass 0.509 5 0.00675 Preprocessor1\_Model1  
## 2 roc\_auc hand\_till 0.820 5 0.00221 Preprocessor1\_Model1

knn.cmf <- confusionMatrix(knn.prediction$.pred\_class, as\_factor(knn.prediction$playlist\_genre))  
knn.cmf

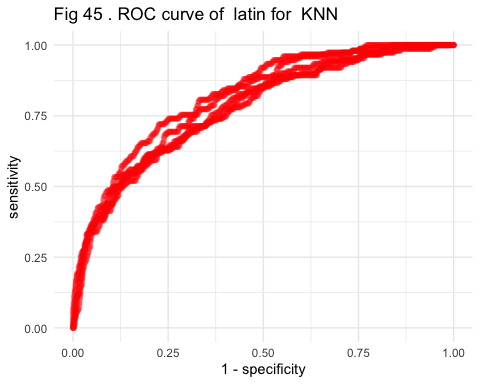
## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction edm latin pop r&b rap rock  
## edm 494 78 127 41 83 80  
## latin 61 345 114 106 102 24  
## pop 137 154 331 125 88 108  
## r&b 17 73 74 273 82 57  
## rap 35 78 48 150 377 9  
## rock 6 22 56 55 18 472  
##   
## Overall Statistics  
##   
## Accuracy : 0.5093   
## 95% CI : (0.4946, 0.524)  
## No Information Rate : 0.1667   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.4112   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Statistics by Class:  
##   
## Class: edm Class: latin Class: pop Class: r&b Class: rap  
## Sensitivity 0.6587 0.46000 0.44133 0.36400 0.50267  
## Specificity 0.8909 0.89147 0.83680 0.91920 0.91467  
## Pos Pred Value 0.5471 0.45878 0.35101 0.47396 0.54089  
## Neg Pred Value 0.9288 0.89194 0.88220 0.87844 0.90192  
## Prevalence 0.1667 0.16667 0.16667 0.16667 0.16667  
## Detection Rate 0.1098 0.07667 0.07356 0.06067 0.08378  
## Detection Prevalence 0.2007 0.16711 0.20956 0.12800 0.15489  
## Balanced Accuracy 0.7748 0.67573 0.63907 0.64160 0.70867  
## Class: rock  
## Sensitivity 0.6293  
## Specificity 0.9581  
## Pos Pred Value 0.7504  
## Neg Pred Value 0.9282  
## Prevalence 0.1667  
## Detection Rate 0.1049  
## Detection Prevalence 0.1398  
## Balanced Accuracy 0.7937

plot <- lapply(1:6, function(x) (ROC\_plot(knn.prediction,x,43,"KNN")))  
print(plot)

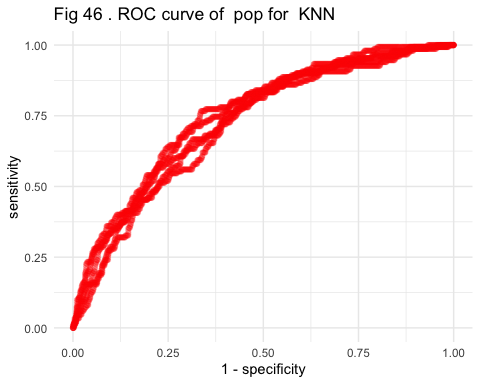
## [[1]]



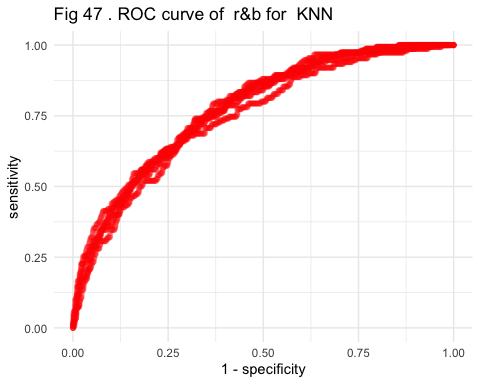
##   
## [[2]]



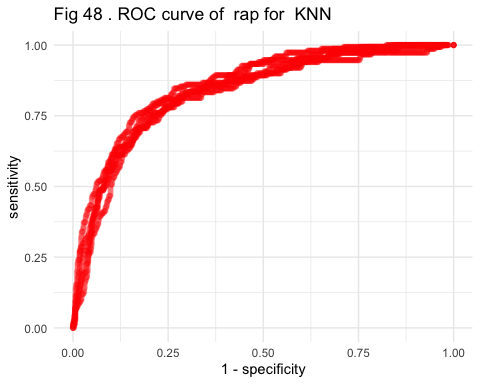
##   
## [[3]]



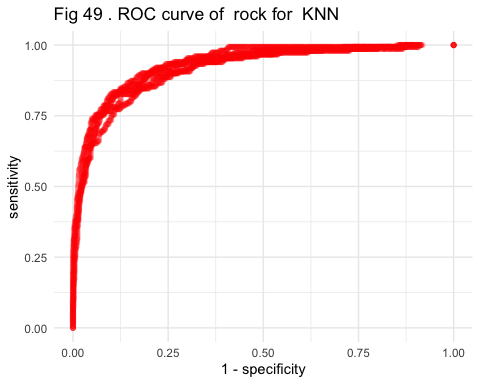
##   
## [[4]]



##   
## [[5]]



##   
## [[6]]



### RF

rf.cv.model <- model\_build\_function(rf\_model\_best)  
rf.prediction <- rf.cv.model$prediction  
  
rf.cv.model$metrics

## # A tibble: 2 × 6  
## .metric .estimator mean n std\_err .config   
## <chr> <chr> <dbl> <int> <dbl> <chr>   
## 1 accuracy multiclass 0.561 5 0.00606 Preprocessor1\_Model1  
## 2 roc\_auc hand\_till 0.849 5 0.00176 Preprocessor1\_Model1

confusionMatrix(rf.prediction$.pred\_class, as\_factor(rf.prediction$playlist\_genre))

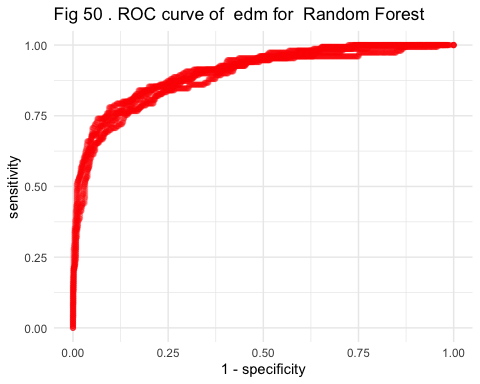
## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction edm latin pop r&b rap rock  
## edm 499 57 104 23 49 19  
## latin 49 343 112 68 62 12  
## pop 112 126 284 94 46 58  
## r&b 27 90 109 320 93 55  
## rap 43 108 52 175 480 8  
## rock 20 26 89 70 20 598  
##   
## Overall Statistics  
##   
## Accuracy : 0.5609   
## 95% CI : (0.5462, 0.5755)  
## No Information Rate : 0.1667   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.4731   
##   
## Mcnemar's Test P-Value : 7.121e-08   
##   
## Statistics by Class:  
##   
## Class: edm Class: latin Class: pop Class: r&b Class: rap  
## Sensitivity 0.6653 0.45733 0.37867 0.42667 0.6400  
## Specificity 0.9328 0.91920 0.88373 0.90027 0.8971  
## Pos Pred Value 0.6644 0.53096 0.39444 0.46110 0.5543  
## Neg Pred Value 0.9330 0.89440 0.87672 0.88702 0.9257  
## Prevalence 0.1667 0.16667 0.16667 0.16667 0.1667  
## Detection Rate 0.1109 0.07622 0.06311 0.07111 0.1067  
## Detection Prevalence 0.1669 0.14356 0.16000 0.15422 0.1924  
## Balanced Accuracy 0.7991 0.68827 0.63120 0.66347 0.7685  
## Class: rock  
## Sensitivity 0.7973  
## Specificity 0.9400  
## Pos Pred Value 0.7266  
## Neg Pred Value 0.9587  
## Prevalence 0.1667  
## Detection Rate 0.1329  
## Detection Prevalence 0.1829  
## Balanced Accuracy 0.8687

rf.cfm <- confusionMatrix(rf.prediction$.pred\_class, as\_factor(rf.prediction$playlist\_genre))  
rf.cfm

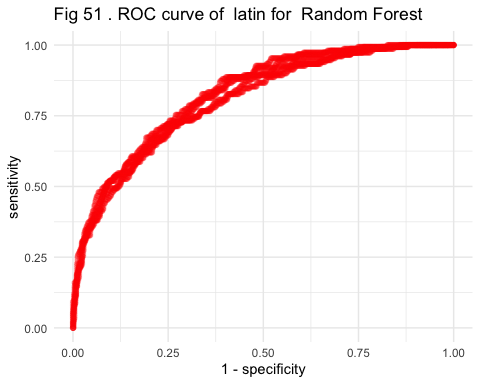
## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction edm latin pop r&b rap rock  
## edm 499 57 104 23 49 19  
## latin 49 343 112 68 62 12  
## pop 112 126 284 94 46 58  
## r&b 27 90 109 320 93 55  
## rap 43 108 52 175 480 8  
## rock 20 26 89 70 20 598  
##   
## Overall Statistics  
##   
## Accuracy : 0.5609   
## 95% CI : (0.5462, 0.5755)  
## No Information Rate : 0.1667   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.4731   
##   
## Mcnemar's Test P-Value : 7.121e-08   
##   
## Statistics by Class:  
##   
## Class: edm Class: latin Class: pop Class: r&b Class: rap  
## Sensitivity 0.6653 0.45733 0.37867 0.42667 0.6400  
## Specificity 0.9328 0.91920 0.88373 0.90027 0.8971  
## Pos Pred Value 0.6644 0.53096 0.39444 0.46110 0.5543  
## Neg Pred Value 0.9330 0.89440 0.87672 0.88702 0.9257  
## Prevalence 0.1667 0.16667 0.16667 0.16667 0.1667  
## Detection Rate 0.1109 0.07622 0.06311 0.07111 0.1067  
## Detection Prevalence 0.1669 0.14356 0.16000 0.15422 0.1924  
## Balanced Accuracy 0.7991 0.68827 0.63120 0.66347 0.7685  
## Class: rock  
## Sensitivity 0.7973  
## Specificity 0.9400  
## Pos Pred Value 0.7266  
## Neg Pred Value 0.9587  
## Prevalence 0.1667  
## Detection Rate 0.1329  
## Detection Prevalence 0.1829  
## Balanced Accuracy 0.8687

plot <- lapply(1:6, function(x) (ROC\_plot(rf.prediction,x,49,"Random Forest")))  
print(plot)

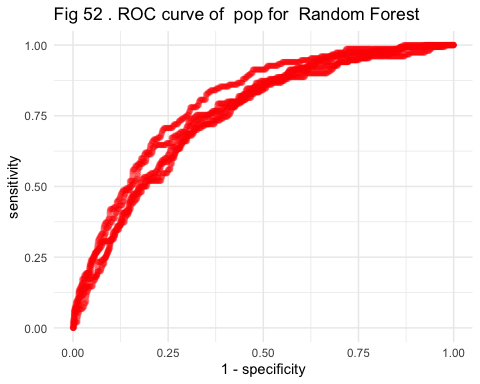
## [[1]]



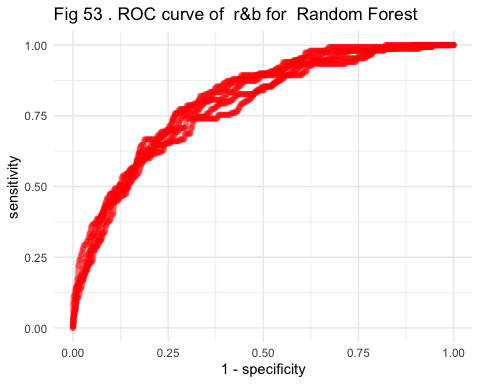
##   
## [[2]]



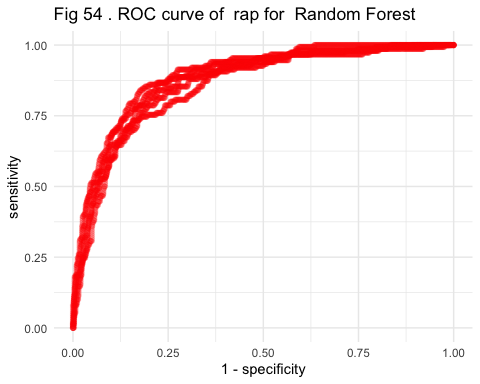
##   
## [[3]]



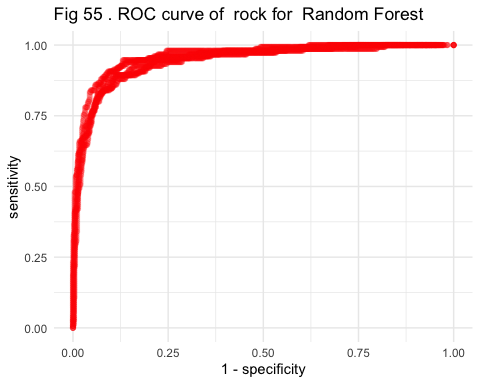
##   
## [[4]]



##   
## [[5]]

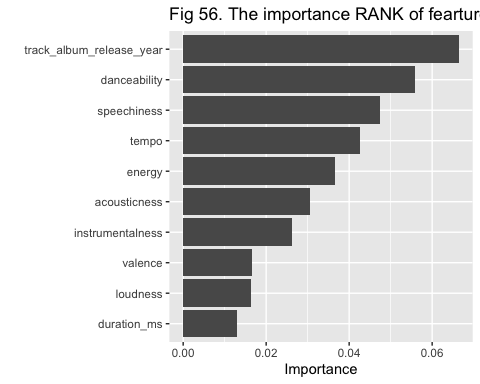


##   
## [[6]]



## 2.prediction

set.seed(1897402)  
rf.final <- rf\_model\_best %>%  
 fit( playlist\_genre ~ . , data = spotify\_train\_prep)  
  
rf.final %>% vip() + ggtitle("Fig 56. The importance RANK of fearturesc")



test\_pred <- predict(rf.final, # The predictions  
 new\_data = spotify\_test\_prep) %>%   
 bind\_cols( spotify\_test\_prep %>% # Add the truth  
 dplyr::select( playlist\_genre ) )  
  
  
confusionMatrix(test\_pred$.pred\_class,as\_factor(test\_pred$playlist\_genre))

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction edm latin pop r&b rap rock  
## edm 162 24 36 7 11 7  
## latin 16 101 34 26 31 5  
## pop 41 50 85 29 15 16  
## r&b 10 22 41 123 25 19  
## rap 18 41 28 50 159 5  
## rock 3 12 26 15 9 198  
##   
## Overall Statistics  
##   
## Accuracy : 0.552   
## 95% CI : (0.5264, 0.5774)  
## No Information Rate : 0.1667   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.4624   
##   
## Mcnemar's Test P-Value : 0.007047   
##   
## Statistics by Class:  
##   
## Class: edm Class: latin Class: pop Class: r&b Class: rap  
## Sensitivity 0.6480 0.40400 0.34000 0.4920 0.6360  
## Specificity 0.9320 0.91040 0.87920 0.9064 0.8864  
## Pos Pred Value 0.6559 0.47418 0.36017 0.5125 0.5282  
## Neg Pred Value 0.9298 0.88423 0.86946 0.8992 0.9241  
## Prevalence 0.1667 0.16667 0.16667 0.1667 0.1667  
## Detection Rate 0.1080 0.06733 0.05667 0.0820 0.1060  
## Detection Prevalence 0.1647 0.14200 0.15733 0.1600 0.2007  
## Balanced Accuracy 0.7900 0.65720 0.60960 0.6992 0.7612  
## Class: rock  
## Sensitivity 0.7920  
## Specificity 0.9480  
## Pos Pred Value 0.7529  
## Neg Pred Value 0.9580  
## Prevalence 0.1667  
## Detection Rate 0.1320  
## Detection Prevalence 0.1753  
## Balanced Accuracy 0.8700

test\_pred\_prob <- predict(rf.final,  
 spotify\_test\_prep,  
 type = "prob" )   
  
colnames(test\_pred\_prob) <- genrename  
   
pROC::multiclass.roc(spotify\_test\_prep$playlist\_genre,   
 as.matrix(test\_pred\_prob))

##   
## Call:  
## multiclass.roc.default(response = spotify\_test\_prep$playlist\_genre, predictor = as.matrix(test\_pred\_prob))  
##   
## Data: multivariate predictor as.matrix(test\_pred\_prob) with 6 levels of spotify\_test\_prep$playlist\_genre: edm, latin, pop, r&b, rap, rock.  
## Multi-class area under the curve: 0.843