Project 1

Daniela Vela

3/15/2020

```
library(formatR)
library(tidyverse)
library(spotifyr)
library(knitr)
Sys.setenv(SPOTIFY_CLIENT_ID = "e8b310e7c042479989dd9570c8772f8d")
Sys.setenv(SPOTIFY_CLIENT_SECRET = "510cd6b14547421a9f2a08d25dce6e83")
```

Introduction: #The two datasets I chose were from Spotify. I chose both Ariana Grande and Taylor Swifts music because they are 2 of the leading female pop superstars. I wanted to see if there was any intersection in their music to tell us why it is so widely listened to. The variables I chose pertain to that. They involve the loudness, valence, acousticness, danceability and energy of their songs. I expect to find similar levels in both these artists music because they give off the same vibe when listening to their discography.

```
# Tidying
library(spotifyr)
library(tidyverse)
library(knitr)
taylorfull <- get_artist_audio_features("taylor swift")
arianafull <- get_artist_audio_features("ariana grande")
taylor <- taylorfull %>% select(danceability, energy, valence)
ariana <- arianafull %>% select(loudness, acousticness, valence)
taylor2 <- taylorfull %>% select(loudness, acousticness, danceability,
    energy, valence, artist_name)
ariana2 <- arianafull %>% select(loudness, acousticness, valence,
    artist_name, danceability, energy)
arianamessy <- ariana %>% pivot_longer(c("acousticness", "valence",
    "loudness"), names_to = "categories", values_to = "levels")
taylormessy <- taylor %>% pivot_longer(c("danceability", "valence",
    "energy"), names_to = "categories", values_to = "levels")
arianatidy <- arianamessy %>% pivot_wider(names_from = "categories",
    values_from = "levels")
```

```
## Warning: Values in `levels` are not uniquely identified; output will contain list-
cols.
## * Use `values_fn = list(levels = list)` to suppress this warning.
## * Use `values_fn = list(levels = length)` to identify where the duplicates arise
## * Use `values_fn = list(levels = summary_fun)` to summarise duplicates
```

```
taylortidy <- taylormessy %>% pivot_wider(names_from = "categories",
    values_from = "levels")
```

```
## Warning: Values in `levels` are not uniquely identified; output will contain list-
cols.
## * Use `values_fn = list(levels = list)` to suppress this warning.
## * Use `values_fn = list(levels = length)` to identify where the duplicates arise
## * Use `values_fn = list(levels = summary_fun)` to summarise duplicates
```

```
# The data i collected from SpotifyR was already tidy. I
# selected the variables I wanted to use for my 2 datasets
# since intially they consisted of 39 variables. I made the
# data sets messy by using pivot_longer. Then I re-tidied
# them using pivot_wider.
```

```
# Joining
joinvalence <- ariana %>% inner_join(taylor)
joinvalencename <- ariana2 %>% full_join(taylor2)
# The datasets were inner joined by valence. The reason for
# using inner join as opposed to any other form was to avoid
# potential NA's that would be present in the data if we used
# full join. The joining was successful because of their
# already common variable of valence.
```

```
# Wrangling
ariana %>% summarize(`average valence` = mean(valence, na.rm = T),
    `average acousticness` = mean(acousticness, na.rm = T), `average loudness` = mean
(loudness,
        na.rm = T), `sd valence` = sd(valence, na.rm = T), `sd acousticness` = sd(aco
usticness,
        na.rm = T), `sd loudness` = sd(loudness, na.rm = T),
    `median valence` = median(valence, na.rm = T), `median acousticness` = median(aco
usticness,
        na.rm = T), `median loudness` = median(loudness, na.rm = T),
    `minimum valence` = min(valence, na.rm = T), `minimum acousticness` = min(acousti
cness,
        na.rm = T), `minimum loudness` = min(loudness, na.rm = T),
    `maximum valence` = max(valence, na.rm = T), `maximum acousticness` = max(acousti
cness,
        na.rm = T), `maximum loudness` = max(loudness, na.rm = T),
    )
```

```
##   average valence average acousticness average loudness sd valence
## 1       0.4324369            0.2038412        -5.851003  0.2063674
##   sd acousticness sd loudness median valence median acousticness
## 1       0.2525574    1.707092          0.383               0.0815
##   median loudness minimum valence minimum acousticness minimum loudness
## 1          -5.634          0.0392             0.000304          -13.623
##   maximum valence maximum acousticness maximum loudness
## 1           0.869                0.969          -2.047
```

```
taylor %>% summarize(`average valence` = mean(valence, na.rm = T),
    `average danceability` = mean(danceability, na.rm = T), `average energy` = mean(e
nergy,
        na.rm = T), `sd valence` = sd(valence, na.rm = T), `sd danceability` = sd(dan
ceability,
        na.rm = T), `sd energy` = sd(energy, na.rm = T), `median valence` = median(va
lence,
        na.rm = T), `median danceability` = median(danceability,
        na.rm = T), `median energy` = median(energy, na.rm = T),
    `minimum danceability` = min(danceability, na.rm = T), `minimum valence` = min(va
lence,
        na.rm = T), `minimum energy` = min(energy, na.rm = T),
    `maximum valence` = max(valence, na.rm = T), `maximum danceability` = max(danceab
ility,
        na.rm = T), `maximum energy` = max(energy, na.rm = T),
    )
```

```
##   average valence average danceability average energy sd valence
## 1       0.4463621            0.6107031      0.5812933  0.2007284
##   sd danceability sd energy median valence median danceability median energy
## 1       0.1073133 0.2052444          0.451               0.607         0.621
##   minimum danceability minimum valence minimum energy maximum valence
## 1                0.292          0.0499          0.151           0.966
##   maximum danceability maximum energy
## 1                0.897           0.95
```

```
joinvalence %>% select(danceability, energy, valence, acousticness,
    loudness) %>% summarize(`average valence` = mean(valence,
    na.rm = T), `average danceability` = mean(danceability, na.rm = T),
    `average energy` = mean(energy, na.rm = T), `average acousticness` = mean(acousti
cness,
        na.rm = T), `average loudness` = mean(loudness, na.rm = T),
    `sd valence` = sd(valence, na.rm = T), `sd danceability` = sd(danceability,
        na.rm = T), `sd energy` = sd(energy, na.rm = T), `sd acousticness` = sd(acous
ticness,
        na.rm = T), `sd loudness` = sd(loudness, na.rm = T),
    `median valence` = median(valence, na.rm = T), `median danceability` = median(dan
ceability,
        na.rm = T), `median energy` = median(energy, na.rm = T),
    `median acousticness` = median(acousticness, na.rm = T),
    `median loudness` = median(loudness, na.rm = T), `minimum danceability` = min(dan
ceability,
        na.rm = T), `minimum valence` = min(valence, na.rm = T),
    `minimum energy` = min(energy, na.rm = T), `minimum acousticness` = min(acousticn
ess,
        na.rm = T), `minimum loudness` = min(loudness, na.rm = T),
    `maximum valence` = max(valence, na.rm = T), `maximum danceability` = max(danceab
ility,
        na.rm = T), `maximum energy` = max(energy, na.rm = T),
    `maximum acousticness` = max(acousticness, na.rm = T), `maximum loudness` = max(l
oudness,
        na.rm = T), )
```

```
##   average valence average danceability average energy average acousticness
## 1       0.4155118            0.5819331      0.6120512            0.2676856
##   average loudness sd valence sd danceability sd energy sd acousticness
## 1        -6.182937  0.1536668       0.1000789 0.1831971       0.2563234
##   sd loudness median valence median danceability median energy
## 1    1.516864          0.374              0.5865         0.636
##   median acousticness median loudness minimum danceability minimum valence
## 1               0.169          -5.959                0.327            0.11
##   minimum energy minimum acousticness minimum loudness maximum valence
## 1          0.151             0.000304          -10.747             0.84
##   maximum danceability maximum energy maximum acousticness maximum loudness
## 1                0.854          0.949                0.943           -2.047
```

```
ariana %>% filter(between(valence, 0.1, 0.7)) %>% arrange(valence) %>%
    mutate(valence_levels = case_when(valence > 0.7 ~ "high",
        0.3 <= valence & valence <= 0.7 ~ "med", valence < 0.3 ~
            "low")) %>% group_by(valence)
```

```
## # A tibble: 243 x 4
## # Groups:   valence [128]
##    loudness acousticness valence valence_levels
##       <dbl>        <dbl>   <dbl> <chr>
##  1    -5.00       0.104    0.103 low
##  2    -4.98       0.0953   0.103 low
##  3    -4.98       0.0953   0.103 low
##  4    -5.04       0.093    0.104 low
##  5    -8.30       0.418    0.11  low
##  6    -8.29       0.422    0.111 low
##  7    -6.51       0.226    0.146 low
##  8    -6.00       0.716    0.152 low
##  9    -4.18       0.158    0.158 low
## 10    -5.96       0.692    0.16  low
## # … with 233 more rows
```

```
taylor %>% filter(between(valence, 0.1, 0.7)) %>% arrange(valence) %>%
    mutate(valence_levels = case_when(valence > 0.7 ~ "high",
        0.3 <= valence & valence <= 0.7 ~ "med", valence < 0.3 ~
            "low")) %>% group_by(valence)
```

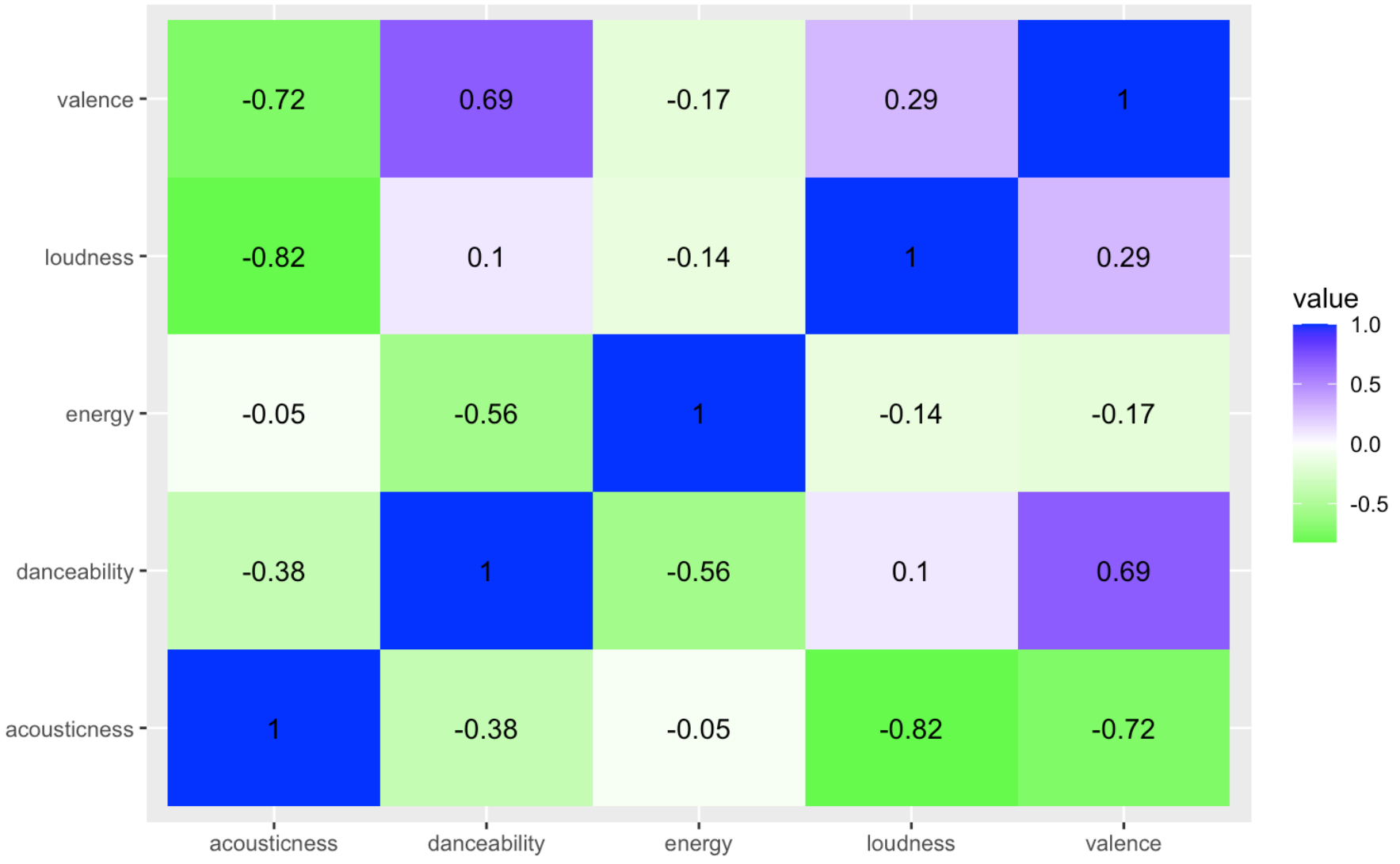
```
## # A tibble: 481 x 4
## # Groups:   valence [296]
##    danceability energy valence valence_levels
##           <dbl>  <dbl>   <dbl> <chr>
##  1        0.589  0.47    0.102 low
##  2        0.462  0.418   0.105 low
##  3        0.731  0.445   0.106 low
##  4        0.481  0.435   0.107 low
##  5        0.481  0.435   0.107 low
##  6        0.459  0.409   0.11  low
##  7        0.586  0.466   0.115 low
##  8        0.575  0.279   0.118 low
##  9        0.505  0.443   0.123 low
## 10        0.505  0.443   0.123 low
## # … with 471 more rows
```

```
joinvalence %>% select(danceability, energy, valence, acousticness,
    loudness) %>% filter(between(valence, 0.1, 0.7)) %>% arrange(valence) %>%
    mutate(valence_levels = case_when(valence > 0.7 ~ "high",
        0.3 <= valence & valence <= 0.7 ~ "med", valence < 0.3 ~
            "low")) %>% group_by(valence)
```

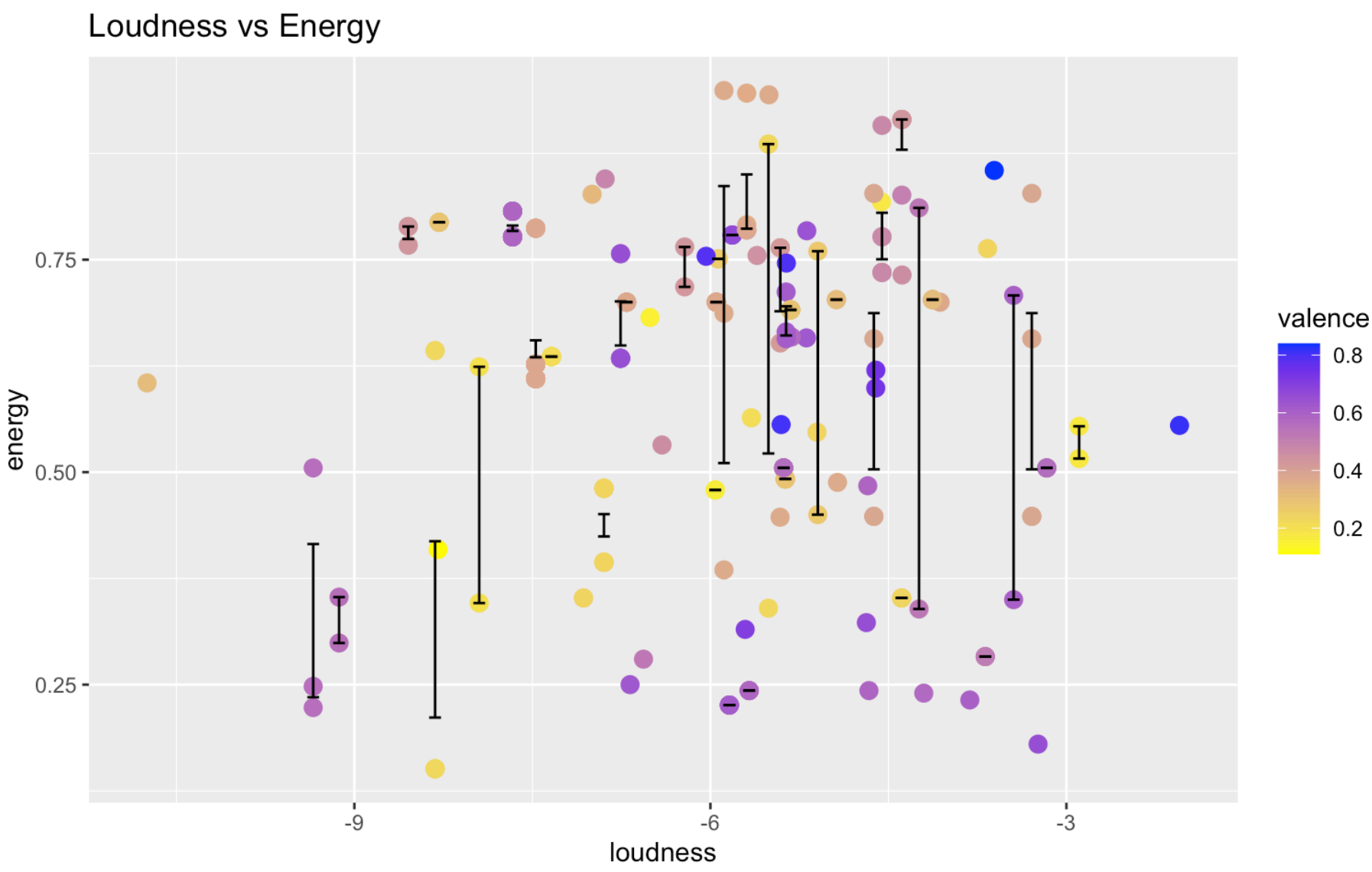
```
## # A tibble: 246 x 6
## # Groups:   valence [65]
##    danceability energy valence acousticness loudness valence_levels
##           <dbl>  <dbl>   <dbl>        <dbl>    <dbl> <chr>
##  1        0.459  0.409   0.11         0.418    -8.30 low
##  2        0.412  0.682   0.146        0.226    -6.51 low
##  3        0.491  0.479   0.16         0.692    -5.96 low
##  4        0.491  0.479   0.16         0.692    -5.96 low
##  5        0.491  0.479   0.16         0.692    -5.96 low
##  6        0.374  0.516   0.171        0.0454   -2.89 low
##  7        0.514  0.554   0.171        0.0454   -2.89 low
##  8        0.338  0.818   0.177        0.109    -4.56 low
##  9        0.464  0.624   0.195        0.78     -7.95 low
## 10        0.585  0.346   0.195        0.78     -7.95 low
## # … with 236 more rows
```

```
joinvalence2 <- joinvalence %>% select(danceability, energy,
    valence, acousticness, loudness) %>% cor(joinvalence) %>%
    as.data.frame
# I first used summarizing methods to see how Taylor Swift
# and Ariana Grande compared in values of standard deviation,
# median, average, minimum and maximum with accordance to the
# variables I am using. I then mutated the data to see how
# high or low the valence is in both of these artists.
```

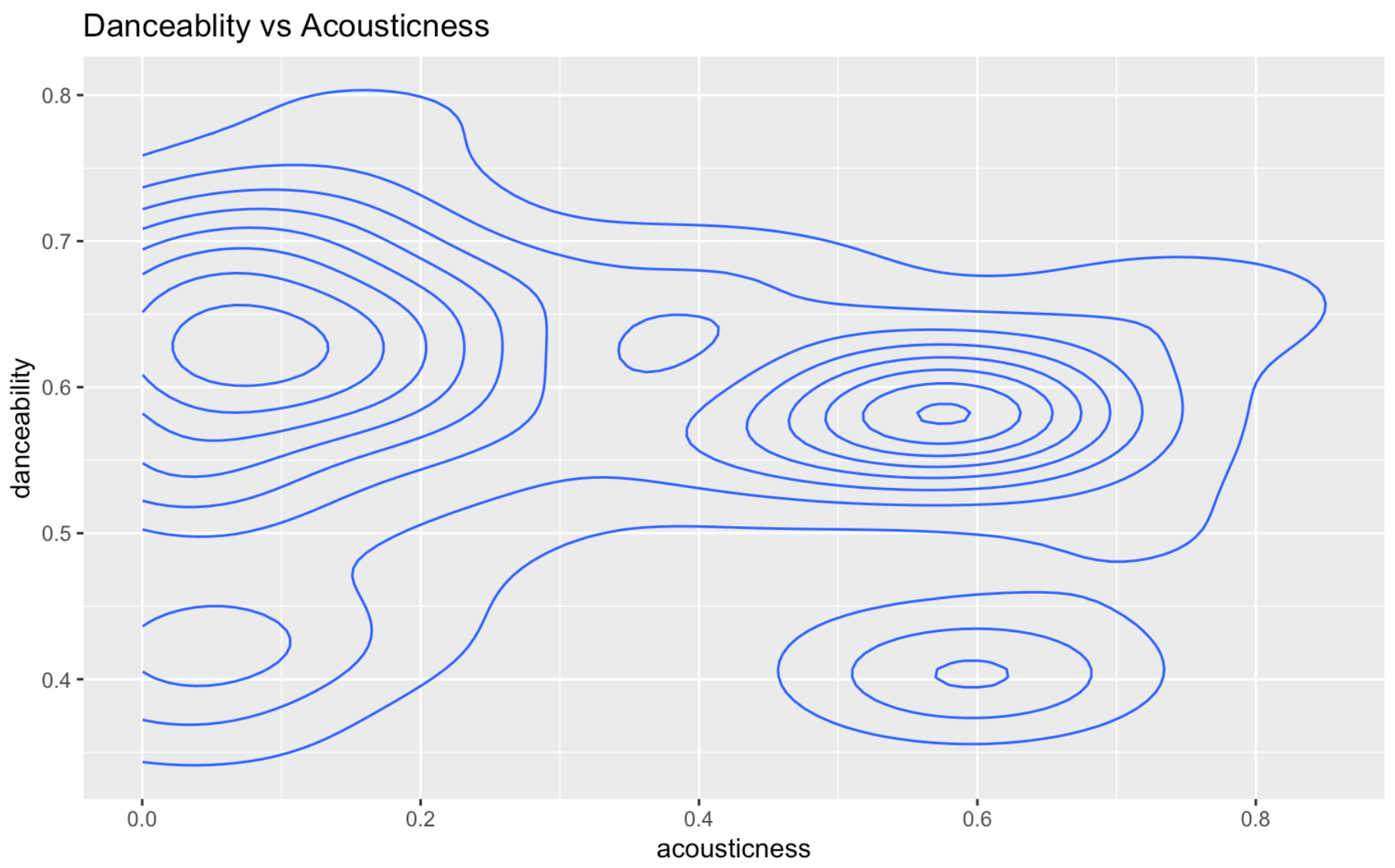
```
# Visualizing
joinvalence2 %>% select_if(is.numeric) %>% cor %>% as.data.frame %>%
    rownames_to_column %>% pivot_longer(-1) %>% ggplot(aes(rowname,
    name, fill = value)) + geom_tile() + geom_text(aes(label = round(value,
    2))) + xlab("") + ylab("") + scale_fill_gradient2(low = "green",
    high = "blue")
```




```
ggplot(joinvalence, aes(loudness, energy, dancebility)) + geom_point(size = 3,
    aes(color = valence)) + scale_color_gradient(low = "yellow",
    high = "blue") + ggtitle("Loudness vs Energy") + geom_errorbar(stat = "summary",
    width = 0.1)
```




```
ggplot(joinvalence, aes(acousticness, danceability, valence)) +
    geom_density_2d() + scale_y_continuous() + ggtitle("Danceablity vs Acousticness")
```




```
# The first graph shows the correlation between all the
# different songs produced by both artist. As we can see by
# the color gradient, the more blue the higher the levels of
# correlation and the more green the lowest. As shown in the
# graph, loudness and accousticness had the lowest
# correlation while valance and dancebility had the highest.
# The correlation means the songs with higher valance are
# easier to dance to while the songs that were the loudest
# were lowest on the accoustic scale.  The second graph shows
# the realtionship between loudness and energy. The colors
# are based on valance. As seen in the graph, there does not
# seem to be any strong correlation between the loudness,
# energy or valance. This is also shown by the error lines.
# The third and final chart is that shows the relationship
# between danceability and acousticness. The danceability
# seems to remain in the middle range of the graph while the
# acousticness is on the lower end. This means the songs are
# more danceable than they are acoustic.
```

```
# Dimentionality Reduction
library(cluster)
joinvalence %>% select_if(is.numeric) %>% cov
```

```
##                 loudness acousticness      valence danceability       energy
## loudness      2.30087501 -0.204514587  0.029464750  0.010001170 -0.010189920
## acousticness -0.20451459  0.065701660 -0.015964012 -0.003216496 -0.002275255
## valence       0.02946475 -0.015964012  0.023613476  0.007652339  0.001515057
## danceability  0.01000117 -0.003216496  0.007652339  0.010015786 -0.003322392
## energy       -0.01018992 -0.002275255  0.001515057 -0.003322392  0.033561195
```

```
eig1 <- joinvalence %>% select_if(is.numeric) %>% cov %>% eigen()
eig1
```

```
## eigen() decomposition
## $values
## [1] 2.319936026 0.054132515 0.033613547 0.020261123 0.005823915
##
## $vectors
##              [,1]       [,2]        [,3]       [,4]         [,5]
## [1,]  0.995791922 -0.0858954 -0.01544088 -0.0274599  0.005287963
## [2,] -0.090439899 -0.8863779 -0.09944346 -0.4340950  0.088471950
## [3,]  0.013418108  0.4148250  0.10108170 -0.7796644  0.457871271
## [4,]  0.004488078  0.1055017  0.17689889 -0.4354811 -0.876305998
## [5,] -0.004345678  0.1540965 -0.97383872 -0.1152696 -0.120774374
```

```
eig1$vectors
```

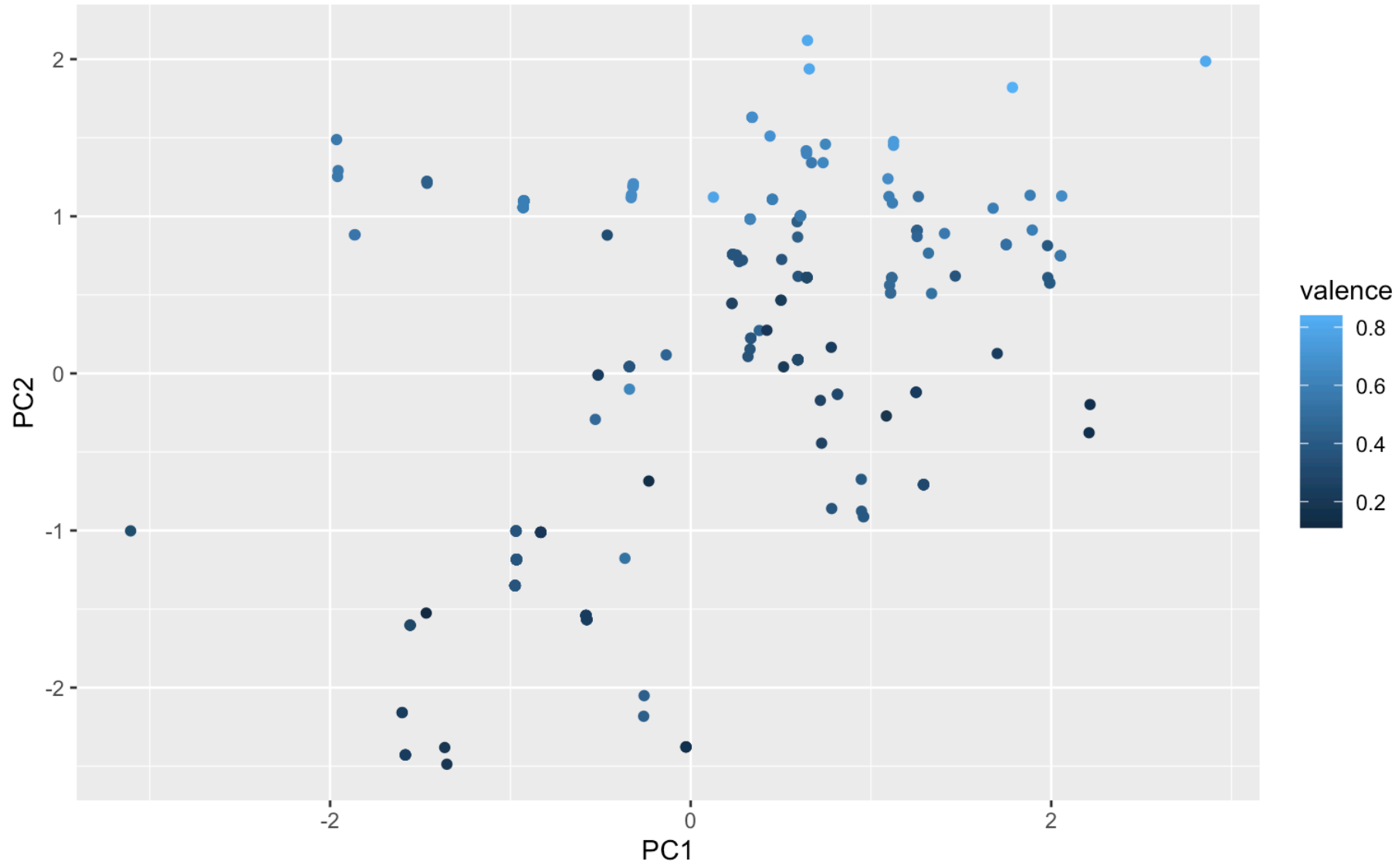
```
##              [,1]       [,2]        [,3]       [,4]         [,5]
## [1,]  0.995791922 -0.0858954 -0.01544088 -0.0274599  0.005287963
## [2,] -0.090439899 -0.8863779 -0.09944346 -0.4340950  0.088471950
## [3,]  0.013418108  0.4148250  0.10108170 -0.7796644  0.457871271
## [4,]  0.004488078  0.1055017  0.17689889 -0.4354811 -0.876305998
## [5,] -0.004345678  0.1540965 -0.97383872 -0.1152696 -0.120774374
```

```
X <- joinvalence %>% select(1:5) %>% scale
PCAscores <- t(t(eig1$vectors) %*% t(X))
PCAscores[, 1:2]
```

```
##               [,1]        [,2]
##   [1,]  1.26312985  1.125579959
##   [2,]  0.77977423  0.166015203
##   [3,]  1.33618505  0.508488713
##   [4,]  1.31902415  0.765305467
##   [5,]  1.46771840  0.619656194
##   [6,] -0.23215961 -0.685238622
##   [7,]  0.72624214 -0.444300842
##   [8,]  0.71938183 -0.171947928
##   [9,]  2.20981391 -0.377681052
##  [10,]  2.21519085 -0.198131388
```

```
##  [11,]  1.08542208 -0.270998164
##  [12,]  1.11556973  0.609073915
##  [13,]  1.11556973  0.609073915
##  [14,]  1.11556973  0.609073915
##  [15,]  1.10325923  0.561677199
##  [16,]  1.10892291  0.511574941
##  [17,]  1.10892291  0.511574941
##  [18,]  1.88270991  1.133446172
##  [19,]  1.89461039  0.912432105
##  [20,]  1.09433489  1.238833153
##  [21,]  0.81419984 -0.132966431
##  [22,]  0.81419984 -0.132966431
##  [23,]  0.38054885  0.273111772
##  [24,] -0.46279652  0.880198336
##  [25,]  1.67758559  1.051321638
##  [26,]  1.70049325  0.126583694
##  [27,] -1.86483832  0.881866224
##  [28,] -1.86158417  0.882828203
##  [29,]  0.95870084 -0.911546040
##  [30,]  0.94618880 -0.674135016
##  [31,]  0.95865599 -0.912600225
##  [32,]  0.94773380 -0.877006241
##  [33,]  0.50107141  0.466062566
##  [34,]  0.50107141  0.466062566
##  [35,]  0.51550315  0.041582110
##  [36,] -1.36431985 -2.381203002
##  [37,] -1.35229903 -2.487486641
##  [38,]  0.73441404  1.341039386
##  [39,] -1.46693868 -1.525317618
##  [40,] -0.26067426 -2.182142383
##  [41,]  0.22903683  0.445373621
##  [42,]  0.22903683  0.445373621
##  [43,] -0.13568985  0.117837297
##  [44,]  1.12517357  1.451853351
##  [45,] -0.31810053  1.206328529
##  [46,] -0.31810053  1.206328529
##  [47,] -0.32841774  1.135849584
##  [48,]  0.59555057  0.617729195
##  [49,] -0.51402921 -0.010877063
##  [50,] -0.36457537 -1.176920662
##  [ reached getOption("max.print") -- omitted 204 rows ]
```

```
joinvalence %>% mutate(PC1 = PCAscores[, 1], PC2 = PCAscores[,
    2]) %>% ggplot(aes(PC1, PC2, color = valence)) + geom_point() +
    scale_fill_gradient2(low = "red", high = "blue")
```




```
# As shown in this graph, we are comparing PC1 vs PC2 and the
# color gradient is once again by valance. The higher valance
# seems to stay at the top of the graph while the lower
# valance is lower.
```

```
## R version 3.6.2 (2019-12-12)
## Platform: x86_64-apple-darwin15.6.0 (64-bit)
## Running under: macOS Sierra 10.12.6
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/3.6/Resources/lib/libRblas.0.dyli
b
## LAPACK: /Library/Frameworks/R.framework/Versions/3.6/Resources/lib/libRlapack.dyli
b
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] cluster_2.1.0   spotifyr_2.1.1  forcats_0.5.0   stringr_1.4.0
##  [5] dplyr_0.8.5     purrr_0.3.3     readr_1.3.1     tidyr_1.0.2
##  [9] tibble_2.1.3    ggplot2_3.3.0   tidyverse_1.3.0 formatR_1.7
## [13] knitr_1.28
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_1.0.3        lubridate_1.7.4   lattice_0.20-38   assertthat_0.2.1
##  [5] digest_0.6.25     utf8_1.1.4        R6_2.4.1          cellranger_1.1.0
##  [9] plyr_1.8.6        backports_1.1.5   reprex_0.3.0      evaluate_0.14
## [13] httr_1.4.1        pillar_1.4.3      rlang_0.4.5       curl_4.3
## [17] readxl_1.3.1      rstudioapi_0.11   Matrix_1.2-18     rmarkdown_2.1
## [21] labeling_0.3      tidytext_0.2.3    munsell_0.5.0     broom_0.5.5
## [25] compiler_3.6.2    janeaustenr_0.1.5 modelr_0.1.6      xfun_0.12
## [29] pkgconfig_2.0.3   htmltools_0.4.0   tidyselect_1.0.0  fansi_0.4.1
## [33] crayon_1.3.4      dbplyr_1.4.2      withr_2.1.2       MASS_7.3-51.4
## [37] SnowballC_0.6.0   grid_3.6.2        nlme_3.1-142      jsonlite_1.6.1
## [41] gtable_0.3.0      lifecycle_0.2.0   DBI_1.1.0         magrittr_1.5
## [45] scales_1.1.0      tokenizers_0.2.1  cli_2.0.2         stringi_1.4.6
## [49] farver_2.0.3      reshape2_1.4.3    fs_1.3.2          xml2_1.2.2
## [53] generics_0.0.2    vctrs_0.2.4       tools_3.6.2       glue_1.3.1
## [57] hms_0.5.3         genius_2.2.0      yaml_2.2.1        colorspace_1.4-1
## [61] isoband_0.2.0     rvest_0.3.5       haven_2.2.0
```

```
## [1] "2020-04-13 23:43:56 CDT"
```

```
##
sysname
##
"Darwin"
##
release
##
"16.7.0"
##
version
## "Darwin Kernel Version 16.7.0: Thu Jun 15 17:36:27 PDT 2017; root:xnu-3789.70.16~2
/RELEASE_X86_64"
##
nodename
##                                                                          "Danis-MacB
ook-Air-2.local"
##
machine
##
"x86_64"
##
login
##
"danivela"
##
user
##
"danivela"
##
effective_user
##
"danivela"
```