Spotify Data Bayesian Analysis

Nathaniel Maxwell, Jessie Bierschenk

```
Logs <- data.frame(read_csv("data/log_sample_reduced.csv"))
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   track_id_clean = col_character(),
##   not_skipped = col_logical()
## )
```

```
Tracks <- data.frame(read_csv("data/tf_sample_1.csv"))
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   track_id = col_character(),
##   duration = col_double(),
##   release_year = col_double(),
##   us_popularity_estimate = col_double(),
##   acousticness = col_double(),
##   beat_strength = col_double()
## )
```

```
Append <- data.frame(read_csv("data/tf_sample_2.csv"))
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   bounciness = col_double(),
##   danceability = col_double(),
##   energy = col_double(),
##   instrumentalness = col_double(),
##   mode = col_character(),
##   speechiness = col_double(),
##   tempo = col_double(),
##   valence = col_double()
## )
```

```
Tracks$bounciness <- Append$bounciness
Tracks$danceability <- Append$danceability
Tracks$energy <- Append$energy
Tracks$instrumentalness <- Append$instrumentalness
Tracks$mode <- Append$mode
```

```r
Tracks$speechiness <- Append$speechiness
Tracks$tempo <- Append$tempo
Tracks$valence <- Append$valence
```

```r
Tracks.bool <- Tracks
Tracks.bool$skipped <- rep(1, length(Tracks$track_id))
c <- rep(1,length(Tracks$track_id))
for (i in 1:length(Tracks$track_id)) {
  c[i] <- i
}
vect <- rep(1,17468)
for (i in 33236:50704) {
  vect[i-33235] <- i
}
Leftover <- Tracks.bool[-vect,]
Tracks.final <- rbind(Tracks.bool, Leftover)
```

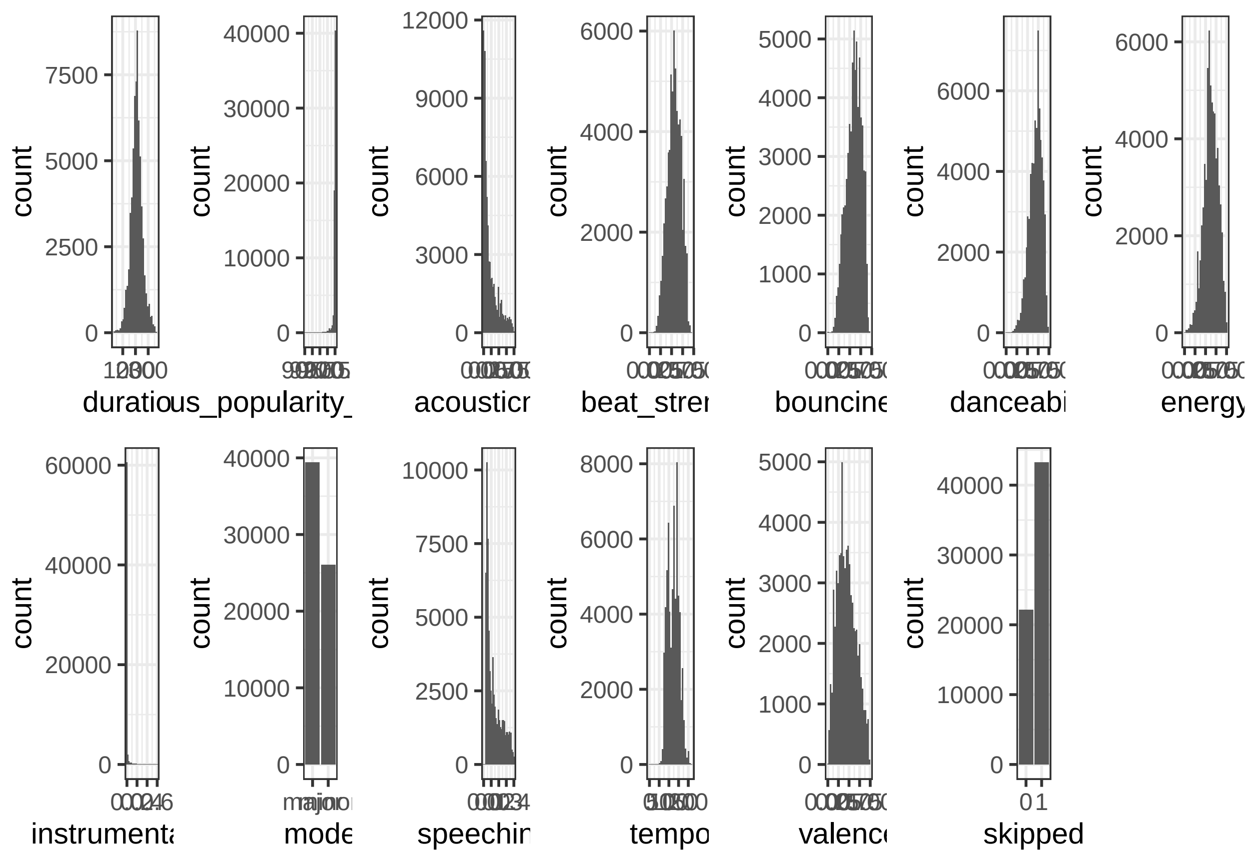
```r
for (i in 1:length(Logs$track_id_clean)) {
  x <- Logs$track_id_clean[[i]]
  y <- which(Tracks$track_id == x)
  bool <- 1
  if (Logs$not_skipped[[i]] == TRUE) {
    bool <- 0
  }
  z <- cbind(Tracks[y,], skipped = bool)
  Tracks.final[i,] <- z
}
```

```r
Tracks.final$skipped <- as.factor(Tracks.final$skipped)
Track_features <- Tracks.final[Tracks.final$release_year >= 2010,]
Track_features <- Track_features[Track_features$speechiness <= 0.4,]
Track_features <- Track_features[Track_features$instrumentalness <= 0.6,]
Track_features <- Track_features[Track_features$duration <= 360,]
```

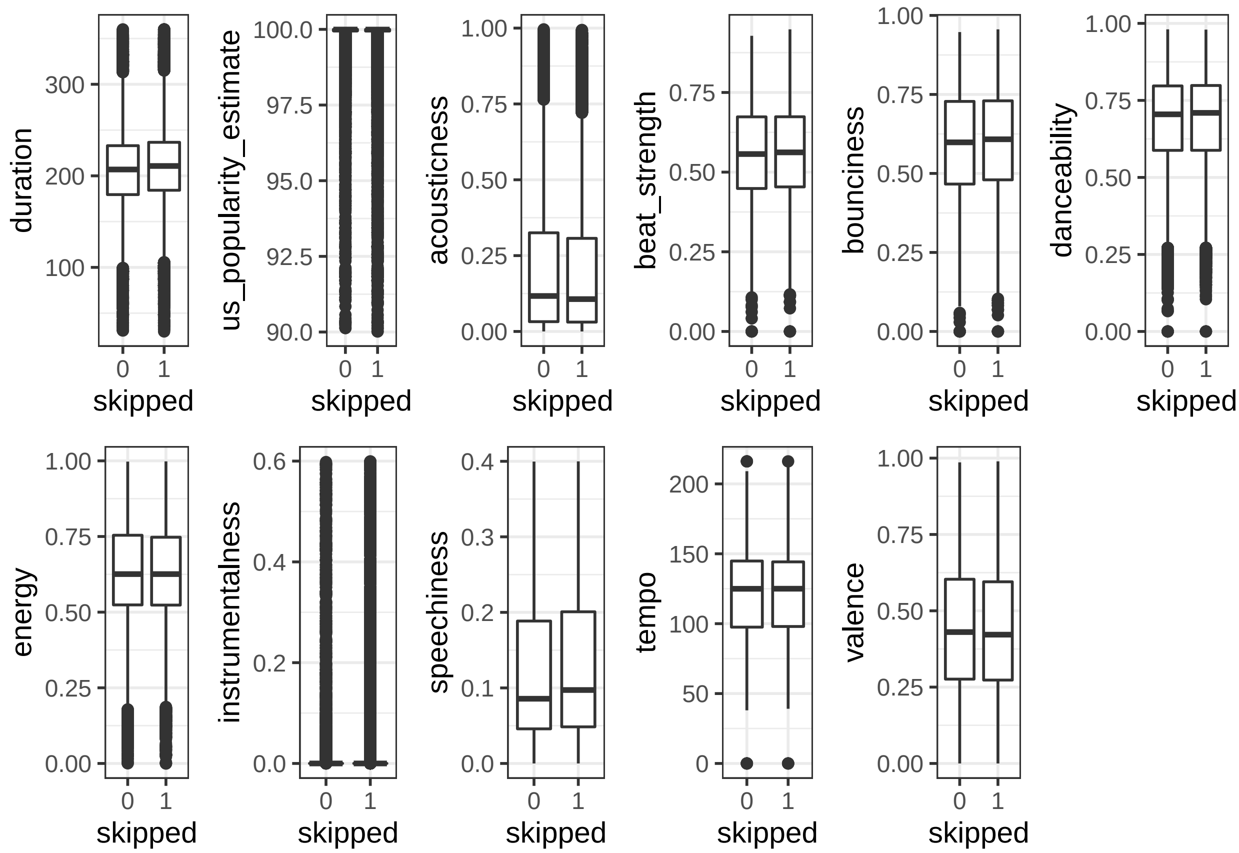
```r
p1= ggplot(data = Track_features, aes(x = duration)) +
  geom_histogram()
p2= ggplot(data = Track_features, aes(x = us_popularity_estimate)) +
  geom_histogram()
p3= ggplot(data = Track_features, aes(x = acousticness)) +
  geom_histogram()
p4= ggplot(data = Track_features, aes(x = beat_strength)) +
  geom_histogram()
p5= ggplot(data = Track_features, aes(x = bounciness)) +
  geom_histogram()
p6=ggplot(data = Track_features, aes(x = danceability)) +
  geom_histogram()
p7= ggplot(data = Track_features, aes(x = energy)) +
  geom_histogram()
p8=ggplot(data = Track_features, aes(x = instrumentalness)) +
  geom_histogram()
p9=ggplot(data = Track_features, aes(x = mode)) +
  geom_bar()
```

```r
p10=ggplot(data = Track_features, aes(x = speechiness)) +
  geom_histogram()
p11= ggplot(data = Track_features, aes(x = tempo)) +
  geom_histogram()
p12=ggplot(data = Track_features, aes(x = valence)) +
  geom_histogram()
p13=ggplot(data = Track_features, aes(x = skipped)) +
  geom_bar()

grid.arrange(p1, p2, p3, p4, p5, p6, p7, p8, p9, p10,p11, p12, p13, nrow=2)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



```r
g1= ggplot(Track_features, aes(skipped, duration)) + geom_boxplot()
g2=ggplot(Track_features, aes(skipped, us_popularity_estimate)) + geom_boxplot()
g3=ggplot(Track_features, aes(skipped, acousticness)) + geom_boxplot()
g4=ggplot(Track_features, aes(skipped, beat_strength)) + geom_boxplot()
g5=ggplot(Track_features, aes(skipped, bounciness)) + geom_boxplot()
g6=ggplot(Track_features, aes(skipped, danceability)) + geom_boxplot()
g7=ggplot(Track_features, aes(skipped, energy)) + geom_boxplot()
g8=ggplot(Track_features, aes(skipped, instrumentalness)) + geom_boxplot()
g9=ggplot(Track_features, aes(skipped, speechiness)) + geom_boxplot()
g10=ggplot(Track_features, aes(skipped, tempo)) + geom_boxplot()
g11=ggplot(Track_features, aes(skipped, valence)) + geom_boxplot()

grid.arrange(g1, g2, g3, g4, g5, g6, g7, g8, g9, g10,g11,  nrow=2)
```




```r
set.seed(4)
a <- sample.int(length(Track_features$track_id), 1000)
Track_features_a <- Track_features[a,]
Track_features_a <- Track_features_a[2:15]
Track_features_a <- Track_features_a[-c(2)]
```

```r
seed <- 1
posterior1 <- stan_glm(skipped ~ ., data = Track_features_a,
                 family = binomial(link = "logit"),
                 prior = normal(0,1), prior_intercept = normal(0,1),
```

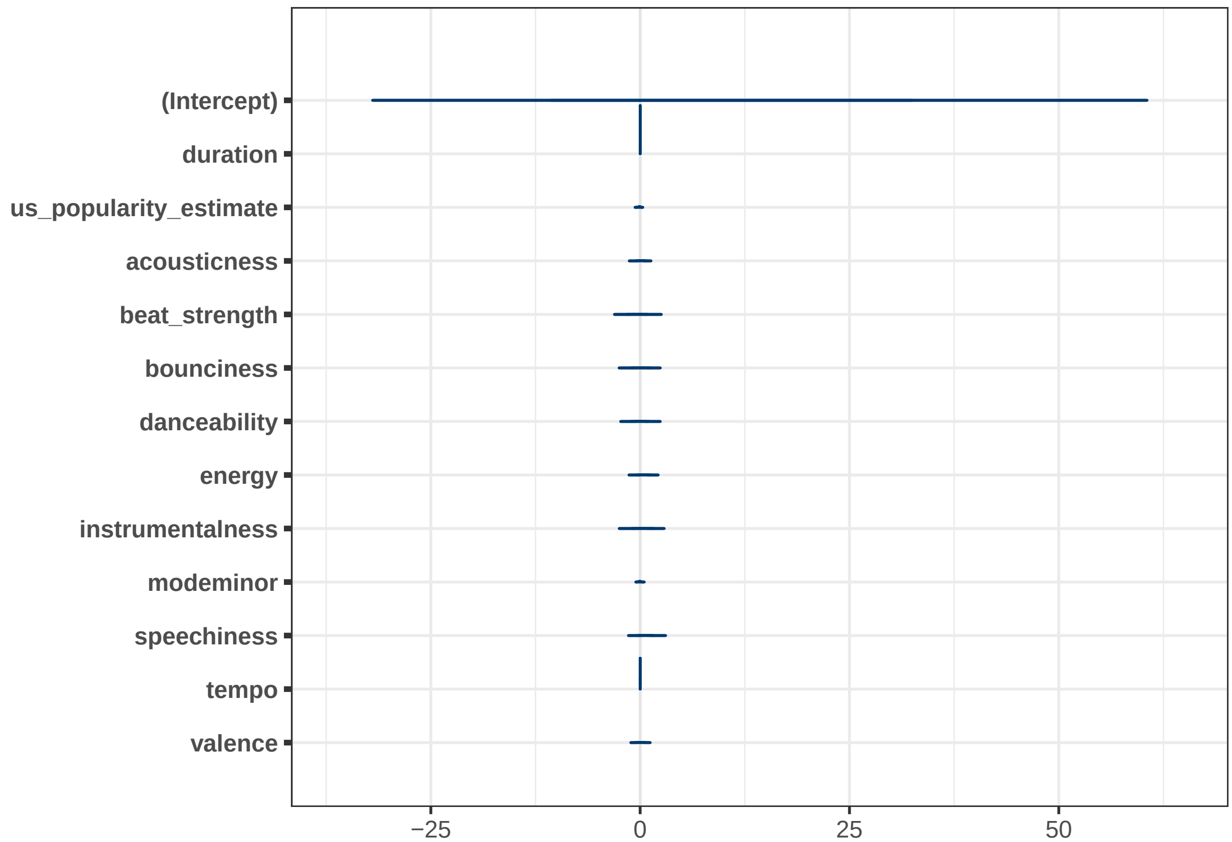
```r
                 seed = seed,
                 refresh = 0)

summary(posterior1)
```

```
##
## Model Info:
##  function:     stan_glm
##  family:       binomial [logit]
##  formula:      skipped ~ .
##  algorithm:    sampling
##  sample:       4000 (posterior sample size)
##  priors:       see help('prior_summary')
##  observations: 1000
##  predictors:   13
##
## Estimates:
##                         mean   sd   10%   50%   90%
## (Intercept)            10.2   13.1 -6.2   9.7  27.3
## duration                0.0    0.0  0.0   0.0   0.0
## us_popularity_estimate -0.1    0.1 -0.3  -0.1   0.1
## acousticness            0.1    0.3 -0.3   0.1   0.5
## beat_strength          -0.4    0.8 -1.4  -0.4   0.6
## bounciness              0.1    0.7 -0.8   0.1   1.0
## danceability            0.0    0.6 -0.9  -0.1   0.8
## energy                  0.4    0.5 -0.2   0.4   1.0
## instrumentalness        0.3    0.8 -0.7   0.3   1.4
## modeminor              -0.1    0.1 -0.2  -0.1   0.1
## speechiness             0.6    0.6 -0.2   0.6   1.3
## tempo                   0.0    0.0  0.0   0.0   0.0
## valence                 0.1    0.3 -0.3   0.1   0.5
##
## Fit Diagnostics:
##            mean   sd   10%   50%   90%
## mean_PPD 0.6    0.0  0.6   0.6   0.7
##
## The mean_ppd is the sample average posterior predictive distribution of the outcome variable (for de
##
## MCMC diagnostics
##                        mcse Rhat n_eff
## (Intercept)            0.2  1.0  5826
## duration               0.0  1.0  3901
## us_popularity_estimate 0.0  1.0  5820
## acousticness           0.0  1.0  4125
## beat_strength          0.0  1.0  3430
## bounciness             0.0  1.0  3314
## danceability           0.0  1.0  4175
## energy                 0.0  1.0  3524
## instrumentalness       0.0  1.0  6275
## modeminor              0.0  1.0  5472
## speechiness            0.0  1.0  5218
## tempo                  0.0  1.0  6757
## valence                0.0  1.0  4111
```

```
## mean_PPD                  0.0  1.0  4438
## log-posterior             0.1  1.0  1739
##
## For each parameter, mcse is Monte Carlo standard error, n_eff is a crude measure of effective sample
```

```r
launch_shinystan(posterior1)
```

```r
mcmc_areas(as.matrix(posterior1), prob = 0.90, prob_outer = 1)
```




```r
round(coef(posterior1), 3)
```

```
##            (Intercept)               duration us_popularity_estimate
##                  9.721                  0.003                 -0.101
##           acousticness          beat_strength             bounciness
##                  0.082                 -0.372                  0.097
##           danceability                 energy       instrumentalness
##                 -0.062                  0.421                  0.295
##              modeminor            speechiness                  tempo
##                 -0.062                  0.561                  0.001
##                valence
##                  0.070
```

```r
round(posterior_interval(posterior1, prob = 0.90), 3)
```

```
##                                5%    95%
## (Intercept)            -10.625 32.464
## duration                  0.001  0.006
## us_popularity_estimate  -0.330  0.103
## acousticness            -0.434  0.631
## beat_strength           -1.649  0.906
## bounciness              -1.096  1.300
## danceability            -1.114  1.021
## energy                  -0.330  1.171
## instrumentalness        -1.008  1.651
## modeminor               -0.282  0.164
## speechiness             -0.422  1.550
## tempo                   -0.003  0.005
## valence                 -0.440  0.595
```

```r
(loo1 <- loo(posterior1, save_psis = TRUE))
```

```
##
## Computed from 4000 by 1000 log-likelihood matrix
##
##          Estimate   SE
## elpd_loo   -652.5 10.0
## p_loo        10.2  0.6
## looic      1304.9 19.9
## ------
## Monte Carlo SE of elpd_loo is 0.0.
##
## All Pareto k estimates are good (k < 0.5).
## See help('pareto-k-diagnostic') for details.
```

```r
post0 <- stan_glm(skipped ~ 1, data = Track_features_a,
                 family = binomial(link = "logit"),
                 prior = normal(0,1), prior_intercept = normal(0,1),
                 seed = seed,
                 refresh = 0)
(loo0 <- loo(post0, save_psis = T))
```

```
##
## Computed from 4000 by 1000 log-likelihood matrix
##
##          Estimate   SE
## elpd_loo   -648.5  9.3
## p_loo         1.0  0.0
## looic      1297.0 18.7
## ------
## Monte Carlo SE of elpd_loo is 0.0.
##
## All Pareto k estimates are good (k < 0.5).
## See help('pareto-k-diagnostic') for details.
```

```r
rstanarm::loo_compare(loo0, loo1)
```

```
##            elpd_diff se_diff
## post0       0.0       0.0
## posterior1 -4.0       3.3
```

###New Data

```r
spotify <- data.frame(read_csv("data/spotify.csv"))
```

```
## Warning: Missing column names filled in: 'X1' [1]
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   X1 = col_double(),
##   acousticness = col_double(),
##   danceability = col_double(),
##   duration_ms = col_double(),
##   energy = col_double(),
##   instrumentalness = col_double(),
##   key = col_double(),
##   liveness = col_double(),
##   loudness = col_double(),
##   mode = col_double(),
##   speechiness = col_double(),
##   tempo = col_double(),
##   time_signature = col_double(),
##   valence = col_double(),
##   target = col_double(),
##   song_title = col_character(),
##   artist = col_character()
## )
```

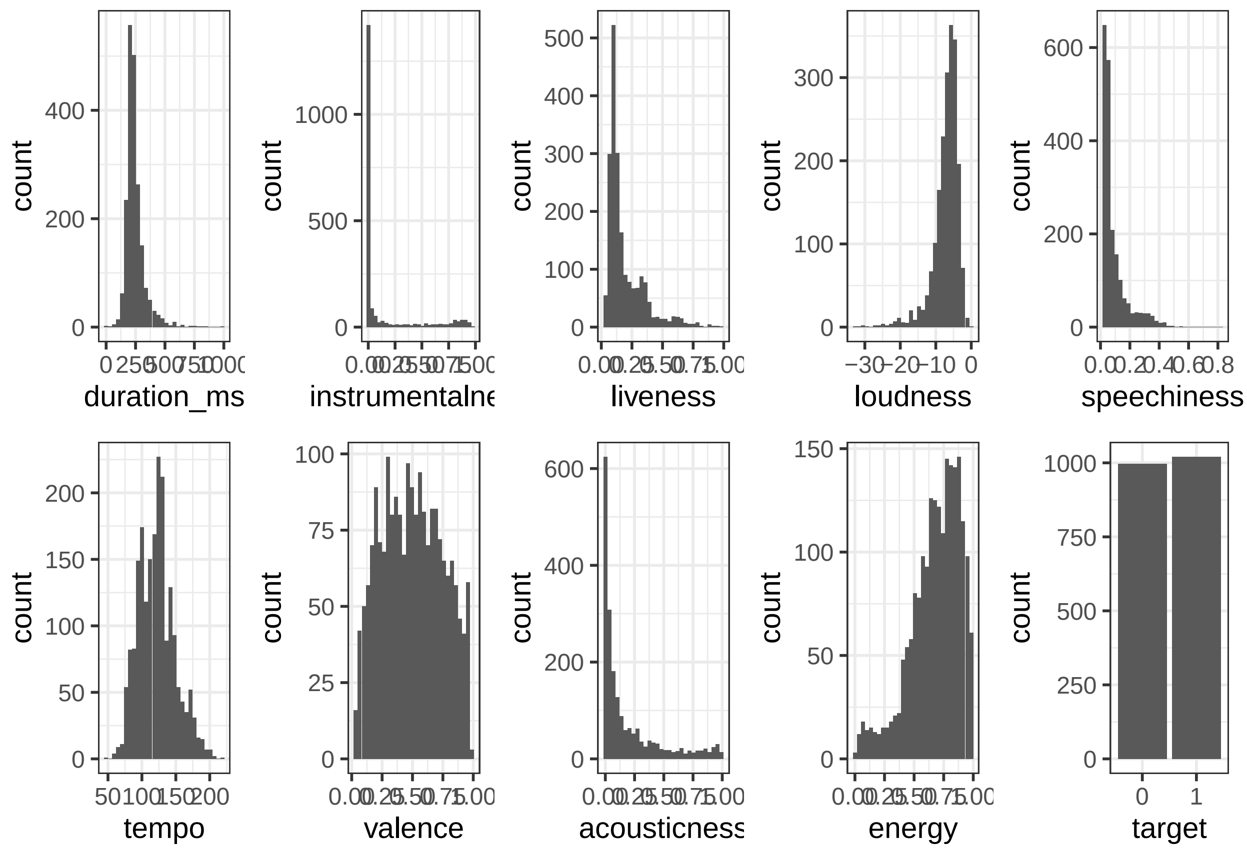
```r
#View(spotify)
```

```r
#Drop un-needed variables
spotify1 <- spotify[-c(1,16,17)]
#View(spotify1)
spotify1$target <- factor(spotify1$target)
spotify1$mode <- factor(spotify1$mode)
spotify1$key <- factor(spotify1$key)
spotify1 <- spotify1 %>%
   mutate(duration_ms = duration_ms / 1000)
```

```r
#EDA
a1= ggplot(data = spotify1, aes(x = duration_ms)) +
  geom_histogram()
a2= ggplot(data = spotify1, aes(x = instrumentalness)) +
  geom_histogram()
a3= ggplot(data = spotify1, aes(x = liveness)) +
```

```
  geom_histogram()
a4= ggplot(data = spotify1, aes(x = loudness)) +
  geom_histogram()
a5= ggplot(data = spotify1, aes(x = speechiness)) +
  geom_histogram()
a6=ggplot(data = spotify1, aes(x = tempo)) +
  geom_histogram()
a7= ggplot(data = spotify1, aes(x = valence)) +
  geom_histogram()
a8=ggplot(data = spotify1, aes(x = acousticness)) +
  geom_histogram()
a9=ggplot(data = spotify1, aes(x = danceability)) +
  geom_histogram()
a9=ggplot(data = spotify1, aes(x = energy)) +
  geom_histogram()
a10=ggplot(data = spotify1, aes(x = target)) +
  geom_bar()
grid.arrange(a1, a2, a3, a4, a5, a6, a7, a8, a9, a10,  nrow=2)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```r
seed=1
post2 <- stan_glm(target ~ ., data = spotify1,
                  family = binomial(link = "logit"),
                  prior = normal(0,1), prior_intercept = normal(0,1),
                  seed = seed,
                  refresh = 0)

summary(post2)
```

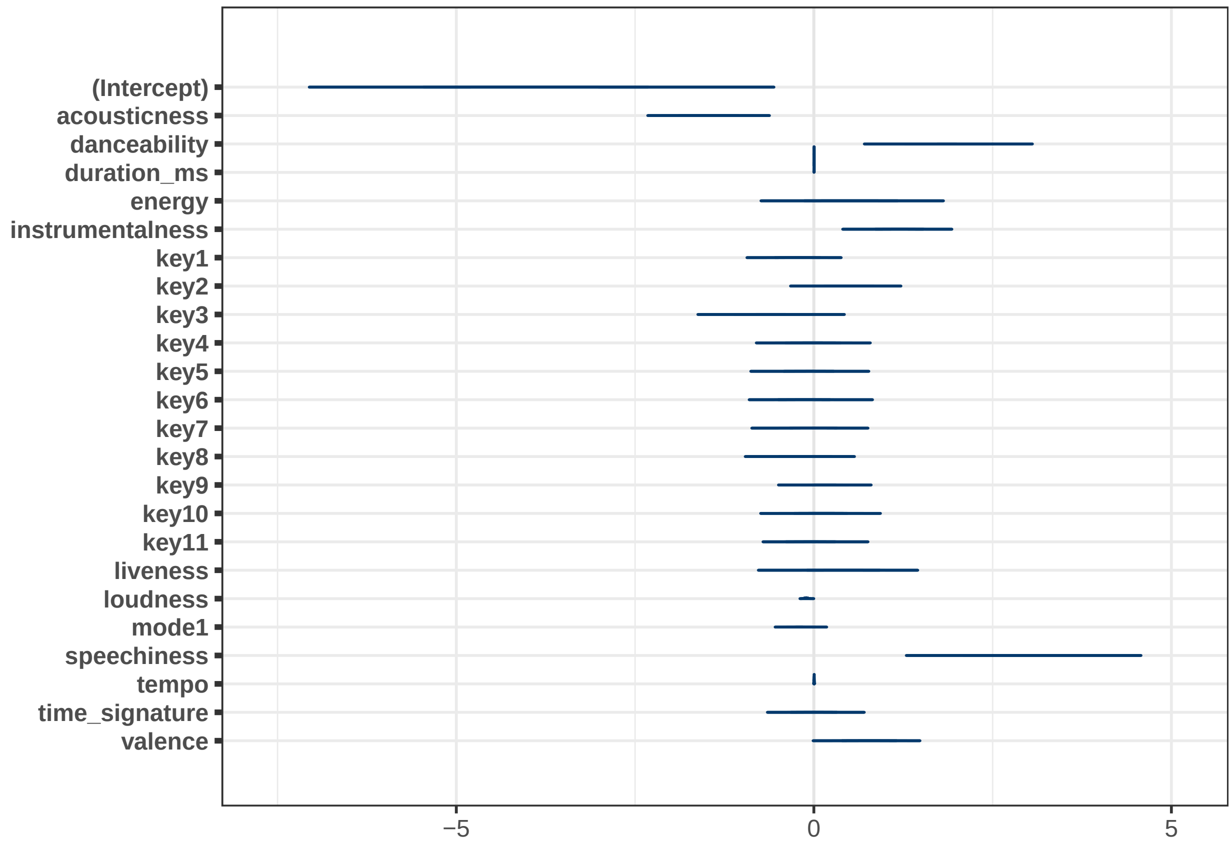
```
##
## Model Info:
##  function:     stan_glm
##  family:       binomial [logit]
##  formula:      target ~ .
##  algorithm:    sampling
##  sample:       4000 (posterior sample size)
##  priors:       see help('prior_summary')
##  observations: 2017
##  predictors:   24
##
## Estimates:
##                    mean   sd   10%   50%   90%
## (Intercept)       -3.9    1.0 -5.1  -3.9  -2.7
## acousticness      -1.5    0.3 -1.8  -1.5  -1.1
## danceability       1.9    0.3  1.4   1.9   2.3
```

```
## duration_ms       0.0    0.0  0.0   0.0   0.0
## energy            0.5    0.4  0.0   0.5   1.0
## instrumentalness  1.2    0.2  0.9   1.2   1.5
## key1             -0.2    0.2 -0.5  -0.2   0.0
## key2              0.5    0.2  0.2   0.5   0.8
## key3             -0.6    0.3 -1.0  -0.6  -0.2
## key4              0.0    0.2 -0.3   0.0   0.3
## key5             -0.1    0.2 -0.3  -0.1   0.2
## key6             -0.1    0.2 -0.4  -0.1   0.1
## key7              0.0    0.2 -0.3   0.0   0.2
## key8             -0.1    0.2 -0.4  -0.1   0.2
## key9              0.2    0.2 -0.1   0.2   0.4
## key10             0.1    0.2 -0.2   0.1   0.4
## key11             0.0    0.2 -0.3   0.0   0.2
## liveness          0.4    0.3  0.0   0.4   0.8
## loudness         -0.1    0.0 -0.1  -0.1  -0.1
## mode1            -0.2    0.1 -0.3  -0.2  -0.1
## speechiness       2.9    0.5  2.3   2.9   3.5
## tempo             0.0    0.0  0.0   0.0   0.0
## time_signature    0.0    0.2 -0.3   0.0   0.3
## valence           0.8    0.2  0.5   0.8   1.1
##
## Fit Diagnostics:
##            mean   sd   10%   50%   90%
## mean_PPD 0.5    0.0  0.5   0.5   0.5
##
## The mean_ppd is the sample average posterior predictive distribution of the outcome variable (for de
##
## MCMC diagnostics
##                  mcse Rhat n_eff
## (Intercept)      0.0  1.0  4989
## acousticness     0.0  1.0  4846
## danceability     0.0  1.0  4350
## duration_ms      0.0  1.0  3965
## energy           0.0  1.0  3140
## instrumentalness 0.0  1.0  5027
## key1             0.0  1.0  1889
## key2             0.0  1.0  2246
## key3             0.0  1.0  3447
## key4             0.0  1.0  2721
## key5             0.0  1.0  2315
## key6             0.0  1.0  2215
## key7             0.0  1.0  2402
## key8             0.0  1.0  2519
## key9             0.0  1.0  2207
## key10            0.0  1.0  2399
## key11            0.0  1.0  2371
## liveness         0.0  1.0  5539
## loudness         0.0  1.0  3396
## mode1            0.0  1.0  5168
## speechiness      0.0  1.0  4780
## tempo            0.0  1.0  4846
## time_signature   0.0  1.0  5077
## valence          0.0  1.0  4466
```

```
## mean_PPD         0.0  1.0  4960
## log-posterior    0.1  1.0  1758
##
## For each parameter, mcse is Monte Carlo standard error, n_eff is a crude measure of effective sample
```

```
launch_shinystan(post2)
```

```
mcmc_areas(as.matrix(post2), prob = 0.90, prob_outer = 1)
```




```
round(coef(post2), 3)
```

```
##      (Intercept)     acousticness     danceability      duration_ms
##           -3.903           -1.478            1.853            0.003
##           energy instrumentalness             key1             key2
##            0.502            1.197           -0.227            0.497
##             key3             key4             key5             key6
##           -0.588           -0.016           -0.061           -0.134
##             key7             key8             key9            key10
##            0.002           -0.121            0.159            0.096
##            key11         liveness         loudness            mode1
##           -0.046            0.409           -0.109           -0.201
##      speechiness            tempo   time_signature          valence
##            2.893            0.004            0.001            0.764
```

```r
round(posterior_interval(post2, prob = 0.90), 3)
```

```
##                       5%    95%
## (Intercept)       -5.458 -2.318
## acousticness      -1.886 -1.056
## danceability       1.277  2.442
## duration_ms        0.002  0.004
## energy            -0.133  1.170
## instrumentalness  0.862  1.532
## key1              -0.544  0.087
## key2               0.149  0.836
## key3              -1.109 -0.071
## key4              -0.397  0.373
## key5              -0.420  0.278
## key6              -0.498  0.225
## key7              -0.333  0.323
## key8              -0.486  0.250
## key9              -0.179  0.481
## key10             -0.274  0.466
## key11             -0.388  0.298
## liveness          -0.094  0.921
## loudness          -0.145 -0.073
## mode1             -0.376 -0.030
## speechiness        2.093  3.715
## tempo              0.001  0.007
## time_signature    -0.323  0.321
## valence            0.395  1.156
```

```r
(loo3 <- loo(post2, save_psis = TRUE))
```

```
##
## Computed from 4000 by 2017 log-likelihood matrix
##
##          Estimate   SE
## elpd_loo  -1268.3 15.7
## p_loo        23.5  0.6
## looic      2536.6 31.5
## ------
## Monte Carlo SE of elpd_loo is 0.1.
##
## All Pareto k estimates are good (k < 0.5).
## See help('pareto-k-diagnostic') for details.
```

```r
post4 <- stan_glm(target ~ 1, data = spotify1,
                 family = binomial(link = "logit"),
                 prior = normal(0,1), prior_intercept = normal(0,1),
                 seed = seed,
                 refresh = 0)
(loo2 <- loo(post4, save_psis = T))
```

```
##
```

```
## Computed from 4000 by 2017 log-likelihood matrix
##
##          Estimate  SE
## elpd_loo  -1398.9 0.5
## p_loo         1.0 0.0
## looic      2797.9 1.0
## ------
## Monte Carlo SE of elpd_loo is 0.0.
##
## All Pareto k estimates are good (k < 0.5).
## See help('pareto-k-diagnostic') for details.
```

```
rstanarm::loo_compare(loo2, loo3)
```

```
##       elpd_diff se_diff
## post2    0.0       0.0
## post4 -130.6      15.7
```

```
preds <- posterior_linpred(post2, transform = TRUE)
```

```
## Instead of posterior_linpred(..., transform=TRUE) please call posterior_epred(), which provides equi
```

```
pred <- colMeans(preds)
```

```
pr <- as.integer(pred >= 0.5)
# have the students calculate this themselves?
round(mean(xor(pr,as.integer(spotify1$target == 0))),3)
```

```
## [1] 0.674
```

```
ploo = E_loo(preds, loo3$psis_object, type="mean", log_ratios = -log_lik(post2))$value
round(mean(xor(ploo>0.5,as.integer(spotify1$target==0))),3)
```

```
## [1] 0.661
```