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
```

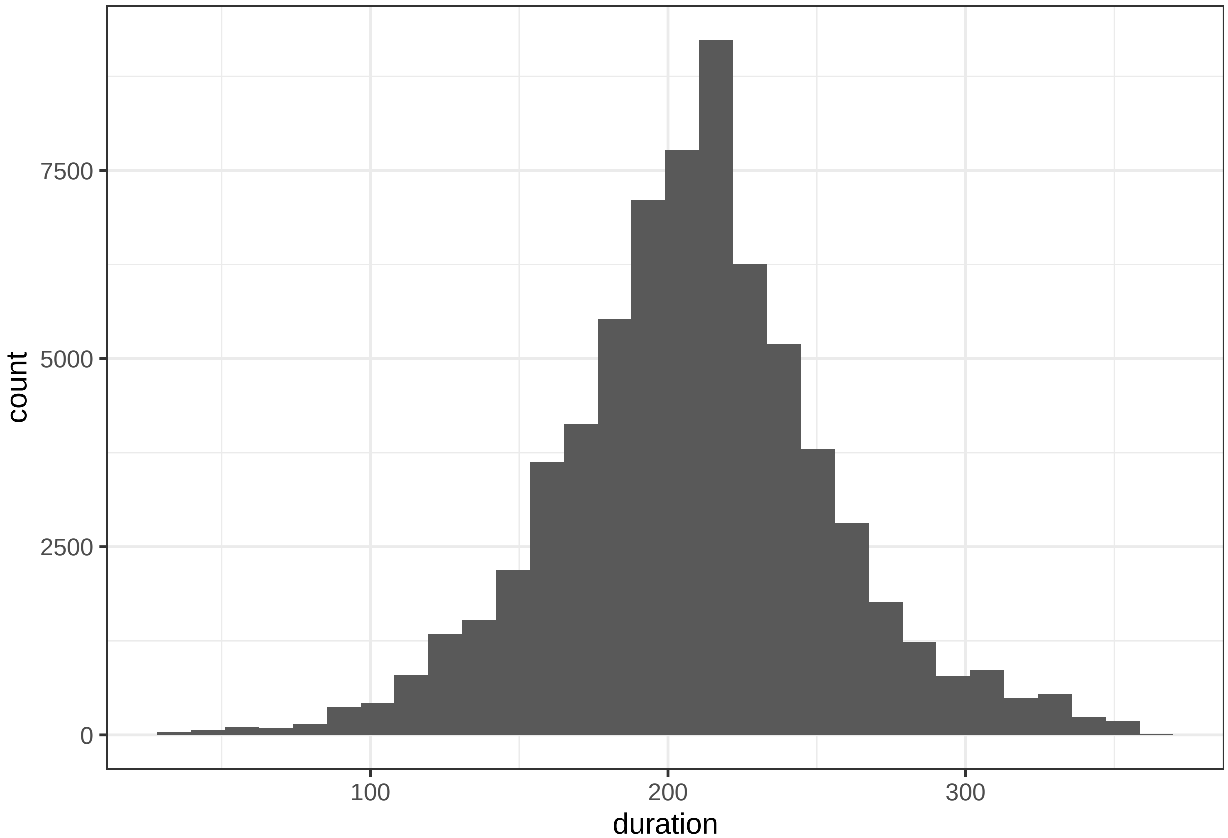
```r
Tracks.bool <- Tracks
Tracks.bool$skipped <- rep(1, length(Tracks$track_id))
c <- rep(1,length(Tracks$track_id))
for (i in 1:length(Tracks$track_id)) {
  c[i] <- i
}
vect <- rep(1,17468)
for (i in 33236:50704) {
  vect[i-33235] <- i
}
Leftover <- Tracks.bool[-vect,]
Tracks.final <- rbind(Tracks.bool, 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 <= 0.7,]
Track_features <- Track_features[Track_features$instrumentalness <= 0.6,]
Track_features <- Track_features[Track_features$duration <= 360,]
```

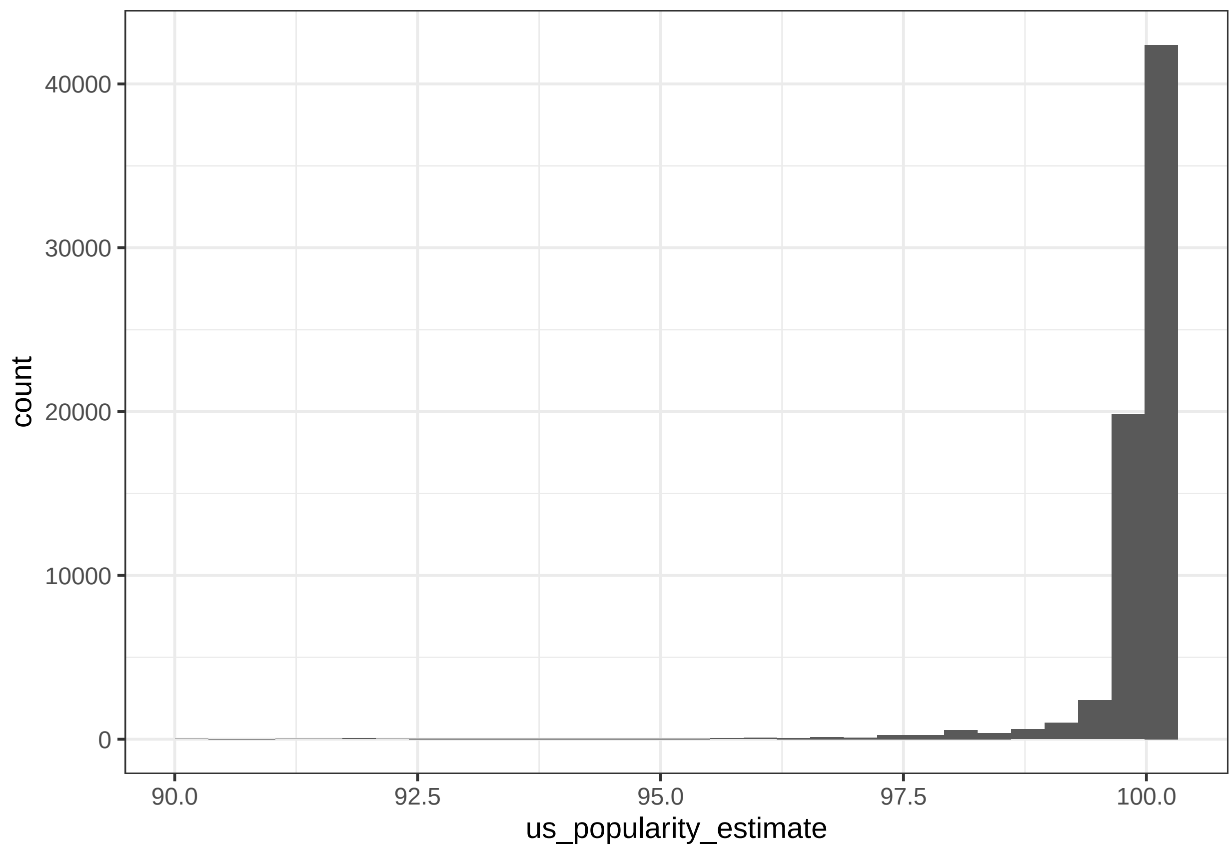
```r
ggplot(data = Track_features, aes(x = duration)) +
  geom_histogram()
```

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



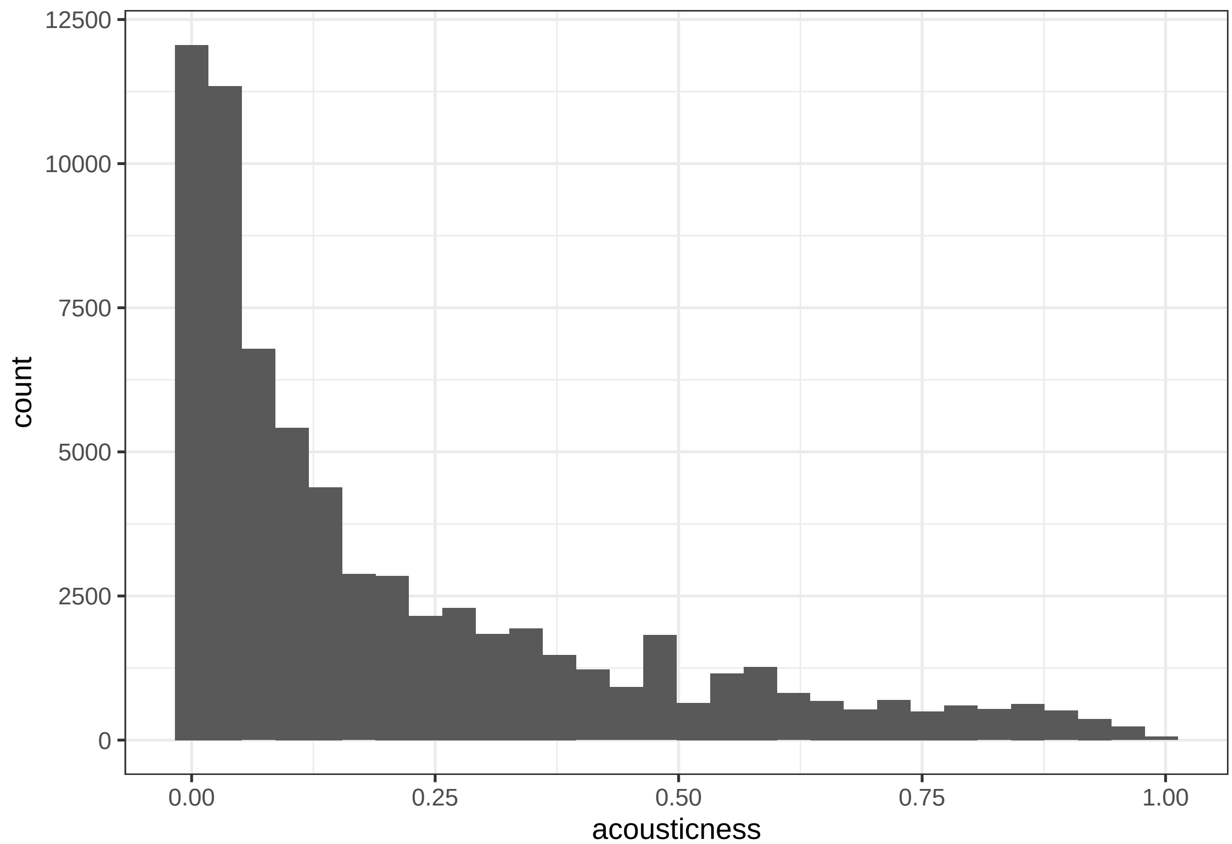

```
ggplot(data = Track_features, aes(x = us_popularity_estimate)) +
  geom_histogram()
```

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



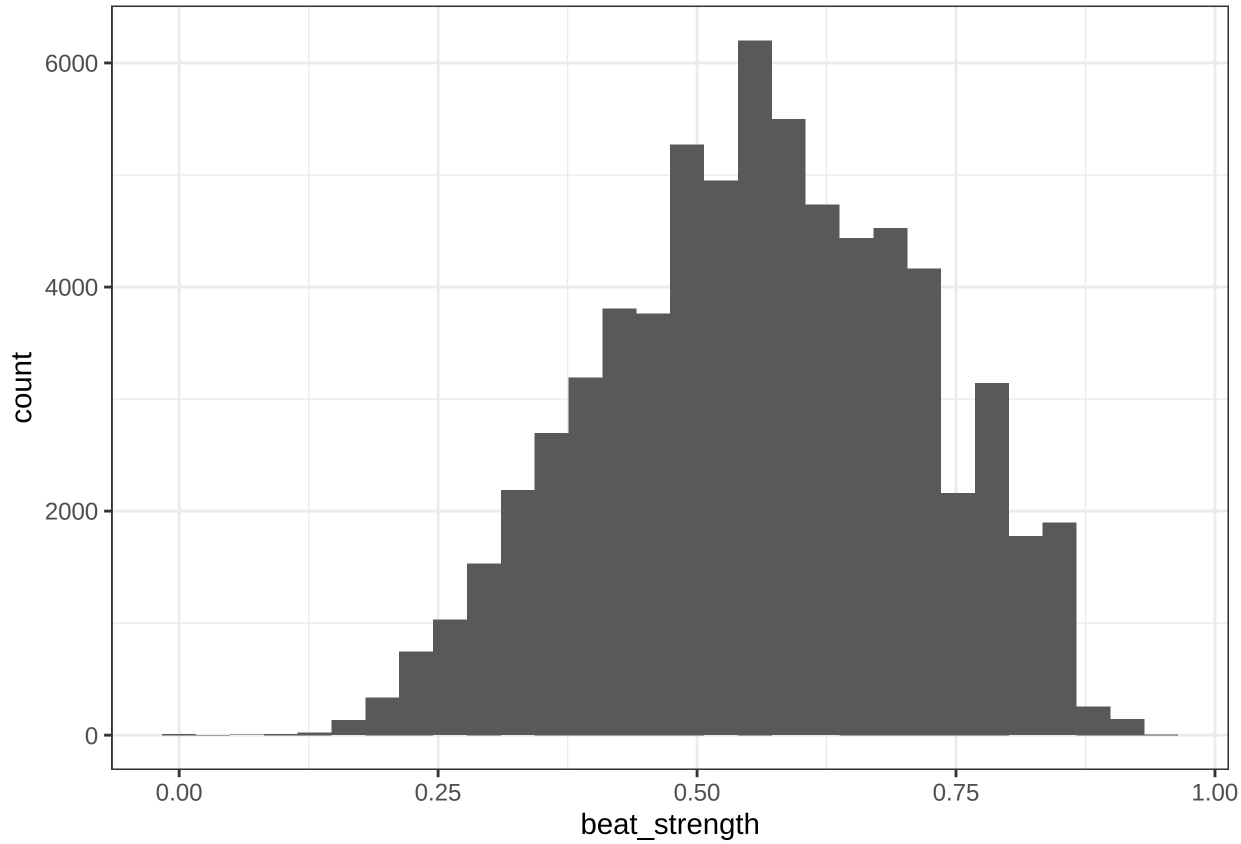

```
ggplot(data = Track_features, aes(x = acousticness)) +
  geom_histogram()
```

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



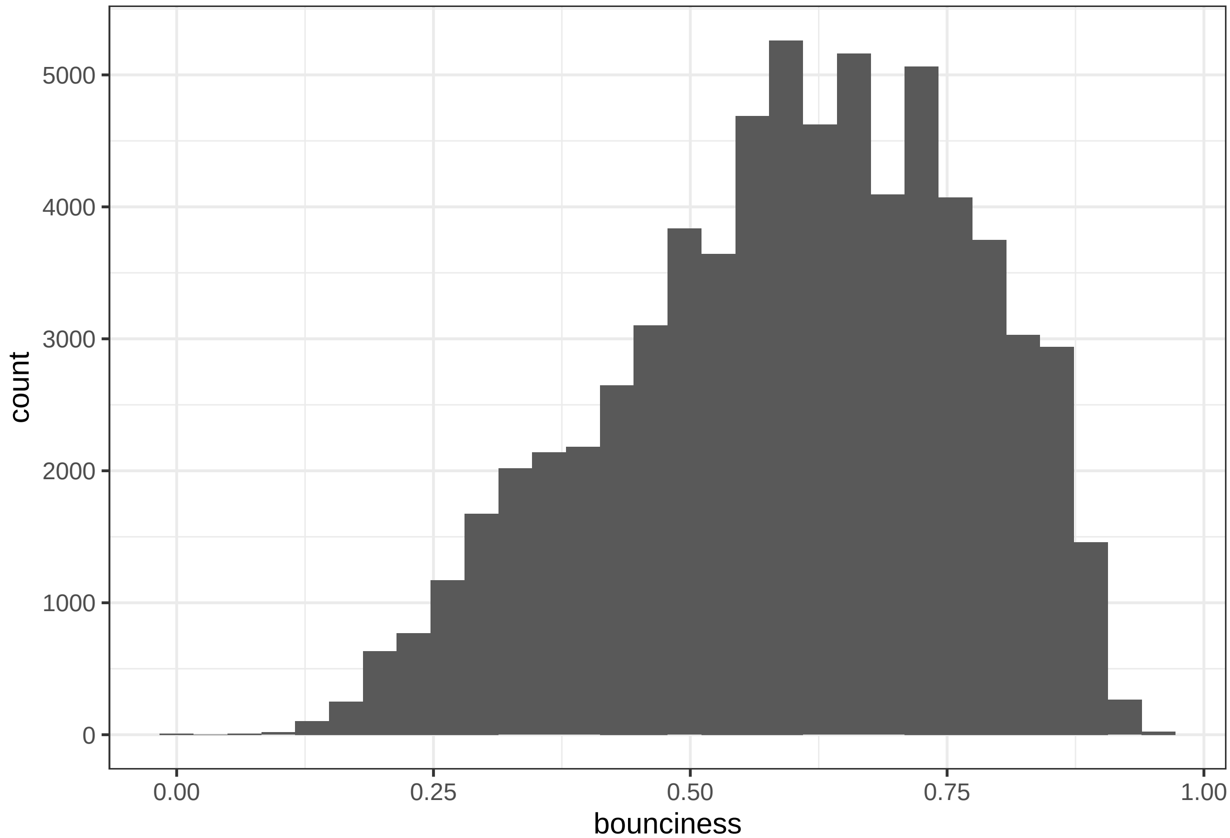

```
ggplot(data = Track_features, aes(x = beat_strength)) +
  geom_histogram()
```

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



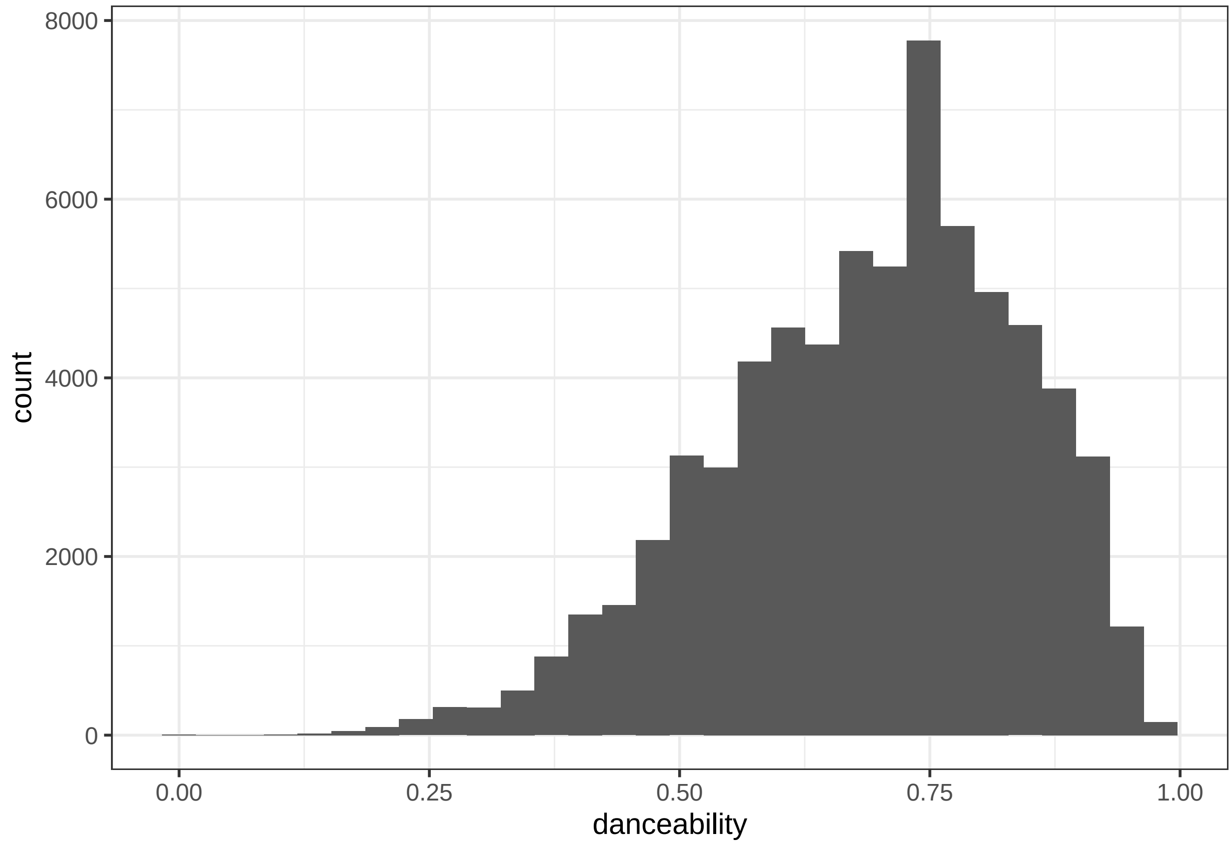

```
ggplot(data = Track_features, aes(x = bounciness)) +
  geom_histogram()
```

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



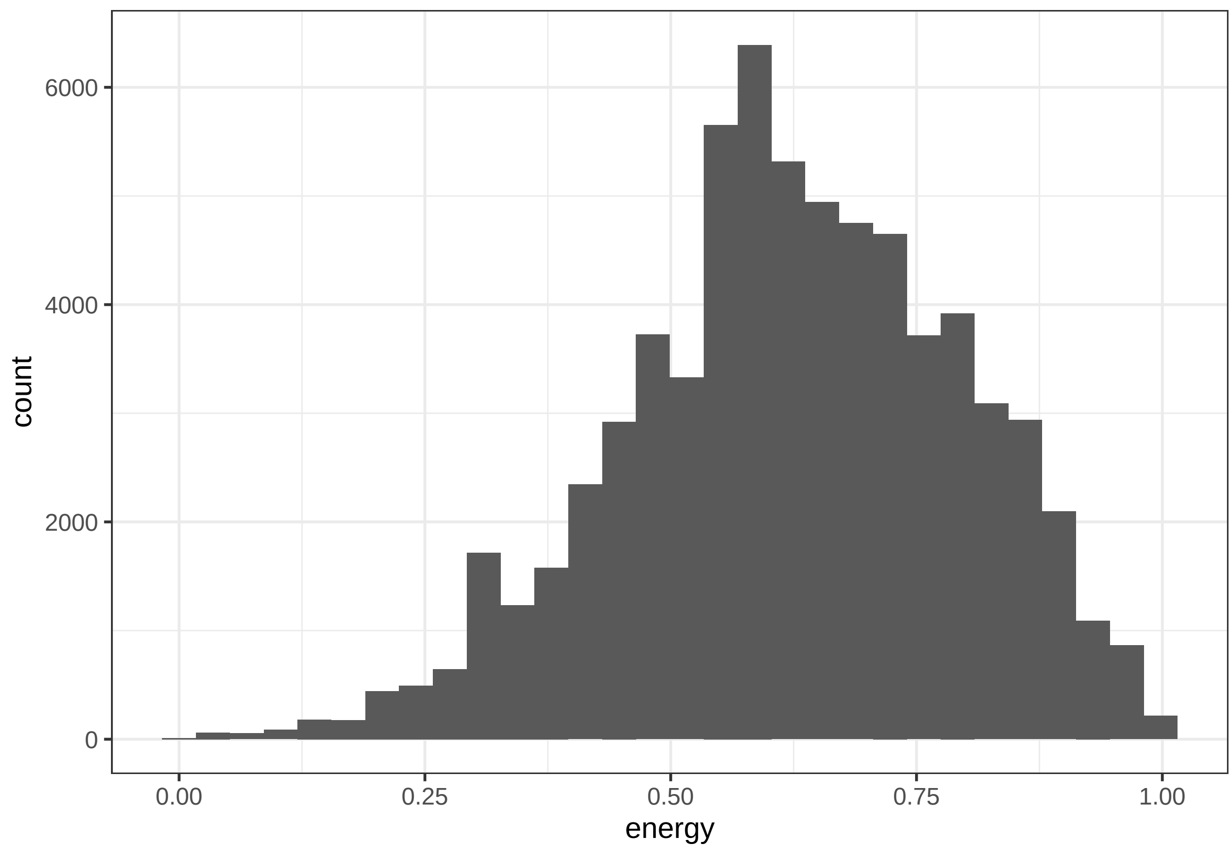

```
ggplot(data = Track_features, aes(x = danceability)) +
  geom_histogram()
```

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



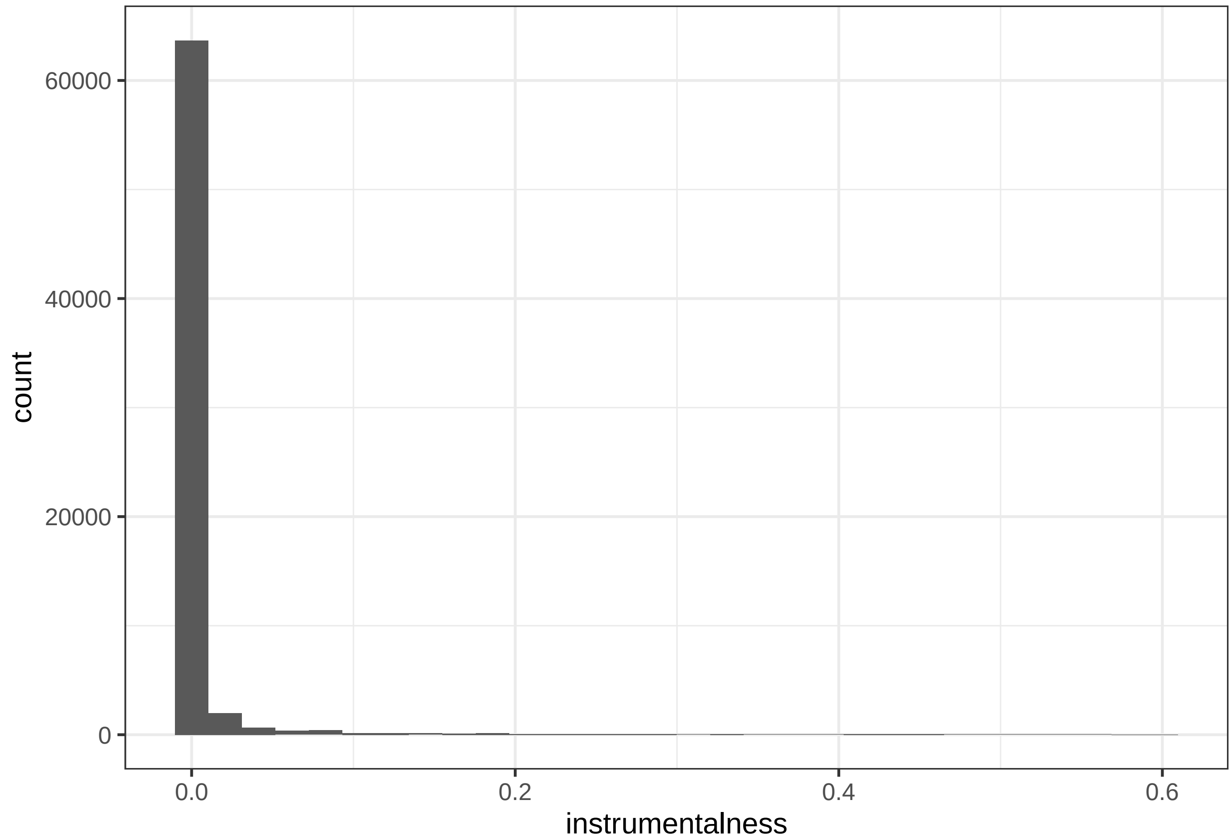

```r
ggplot(data = Track_features, aes(x = energy)) +
  geom_histogram()
```

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

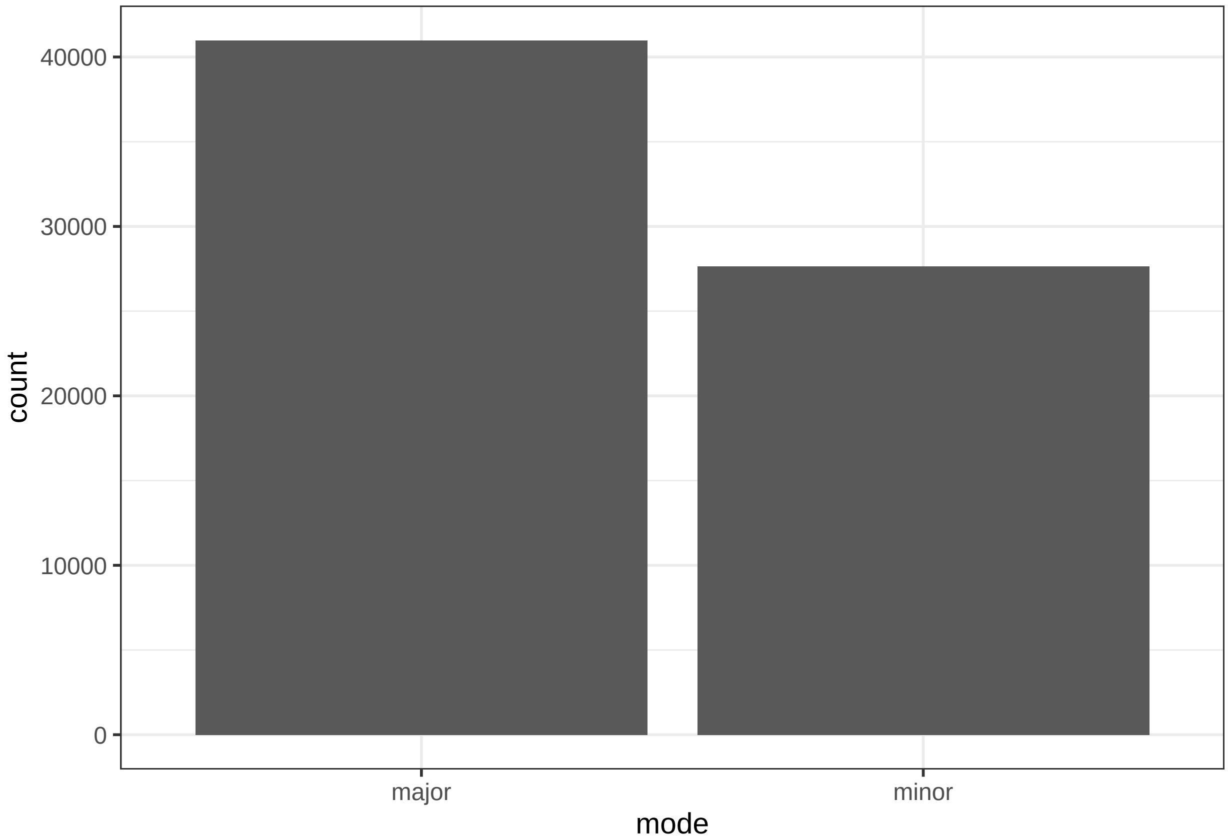



```
ggplot(data = Track_features, aes(x = instrumentalness)) +
  geom_histogram()
```

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

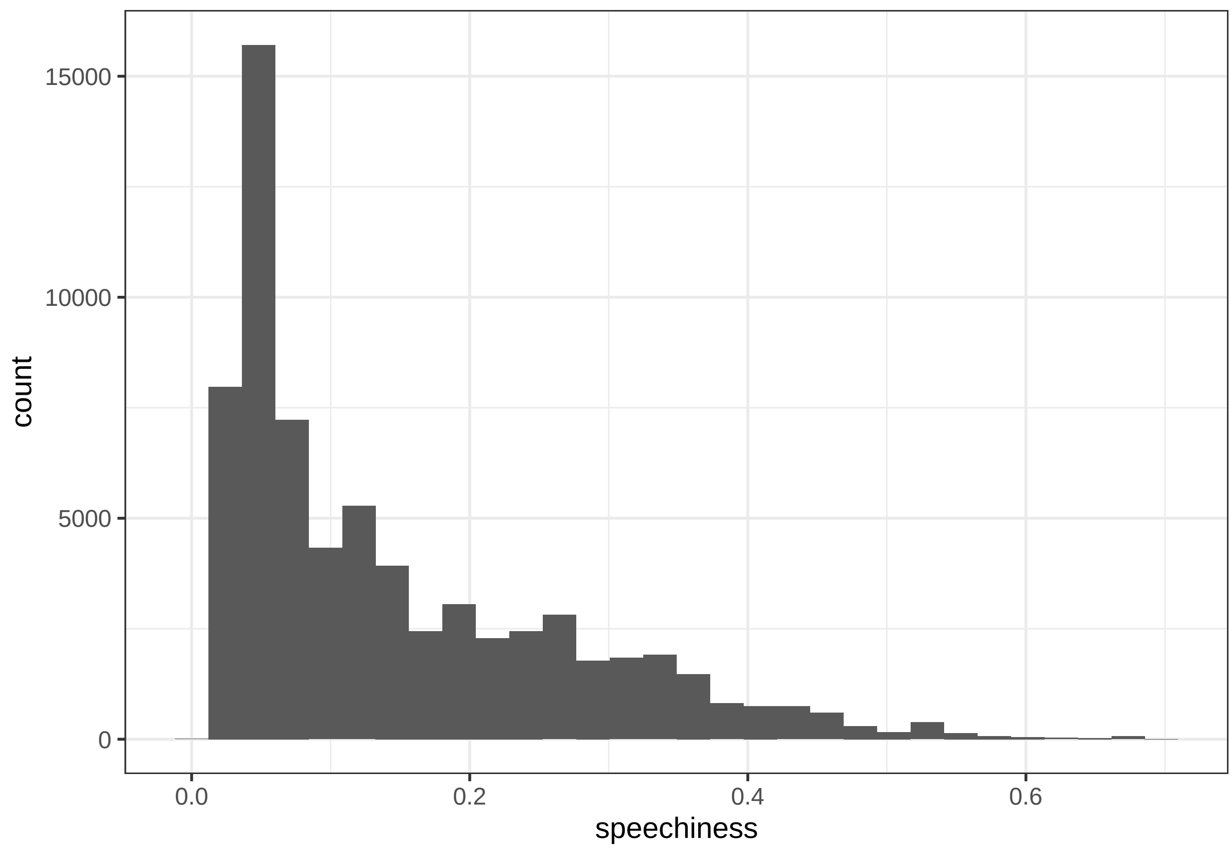



```r
ggplot(data = Track_features, aes(x = mode)) +
  geom_bar()
```



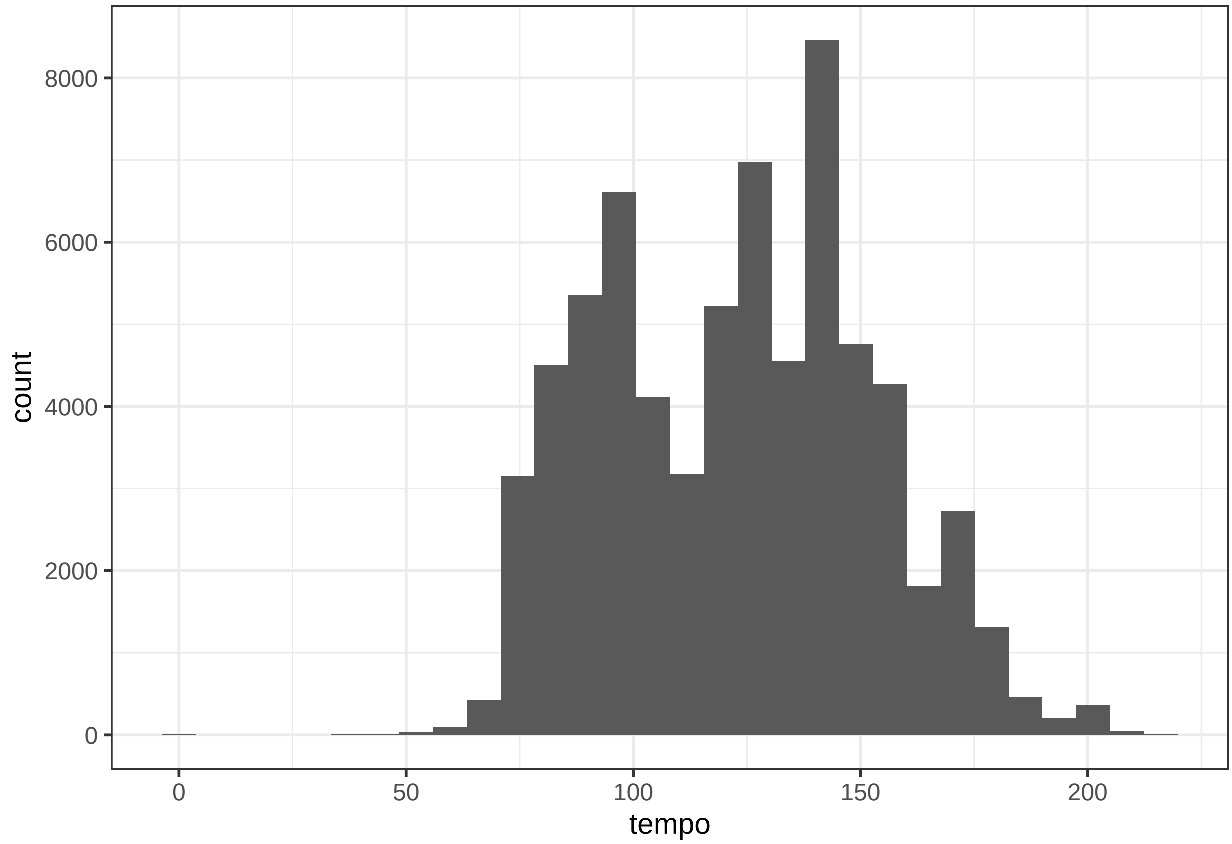

```
ggplot(data = Track_features, aes(x = speechiness)) +
  geom_histogram()
```

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



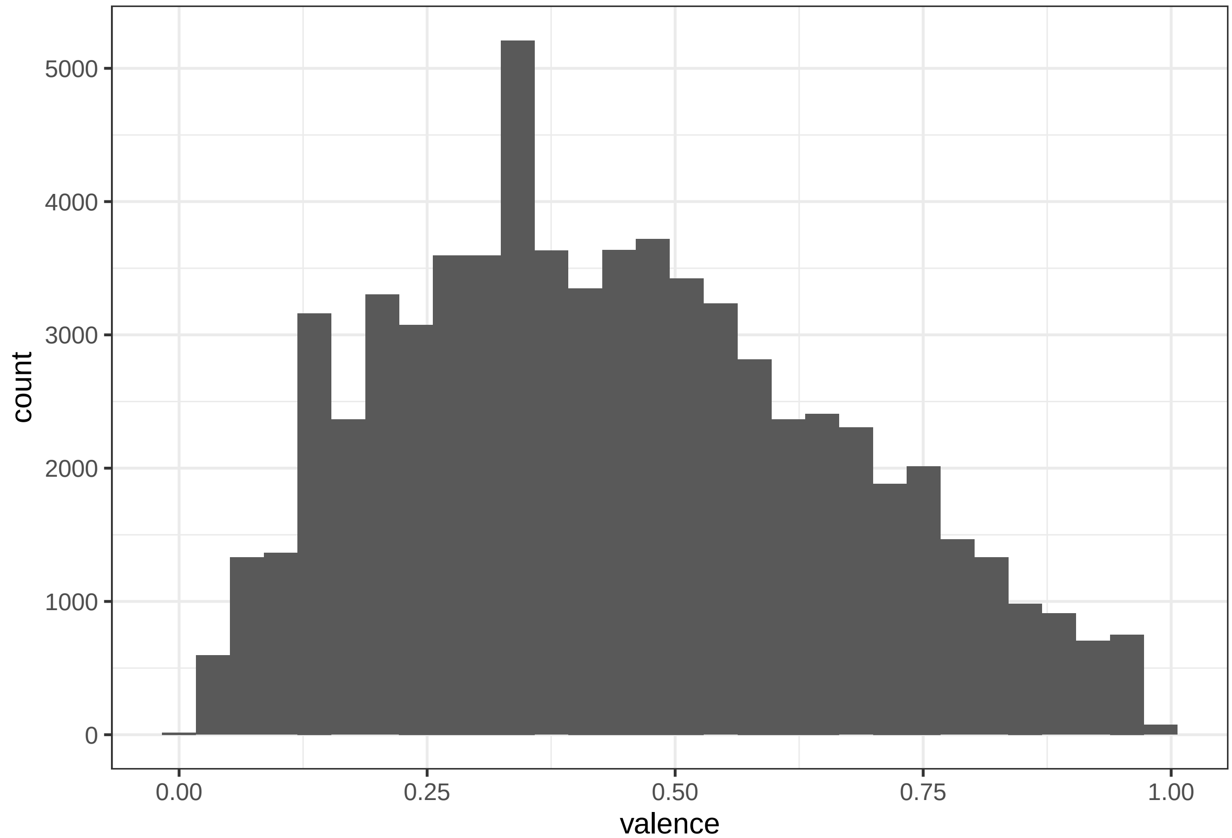

```
ggplot(data = Track_features, aes(x = tempo)) +
  geom_histogram()
```

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

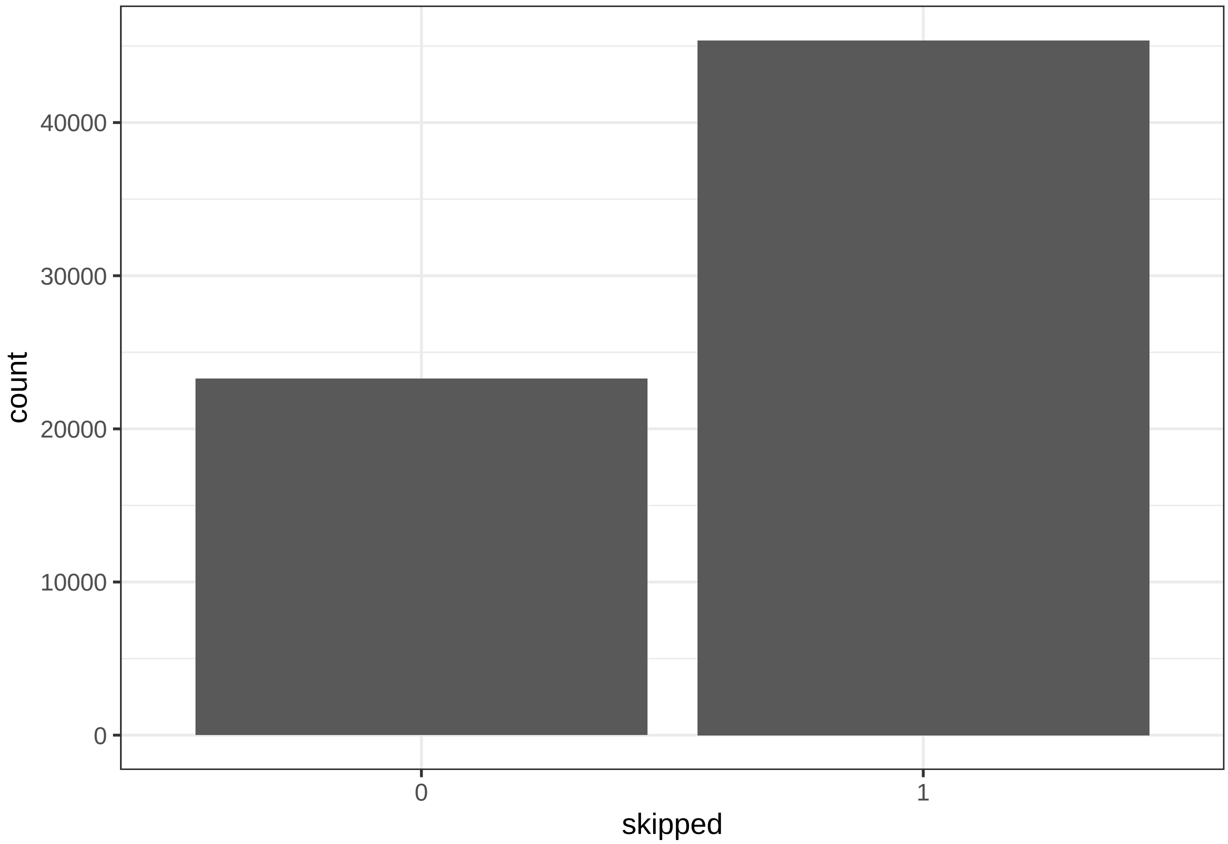



```
ggplot(data = Track_features, aes(x = valence)) +
  geom_histogram()
```

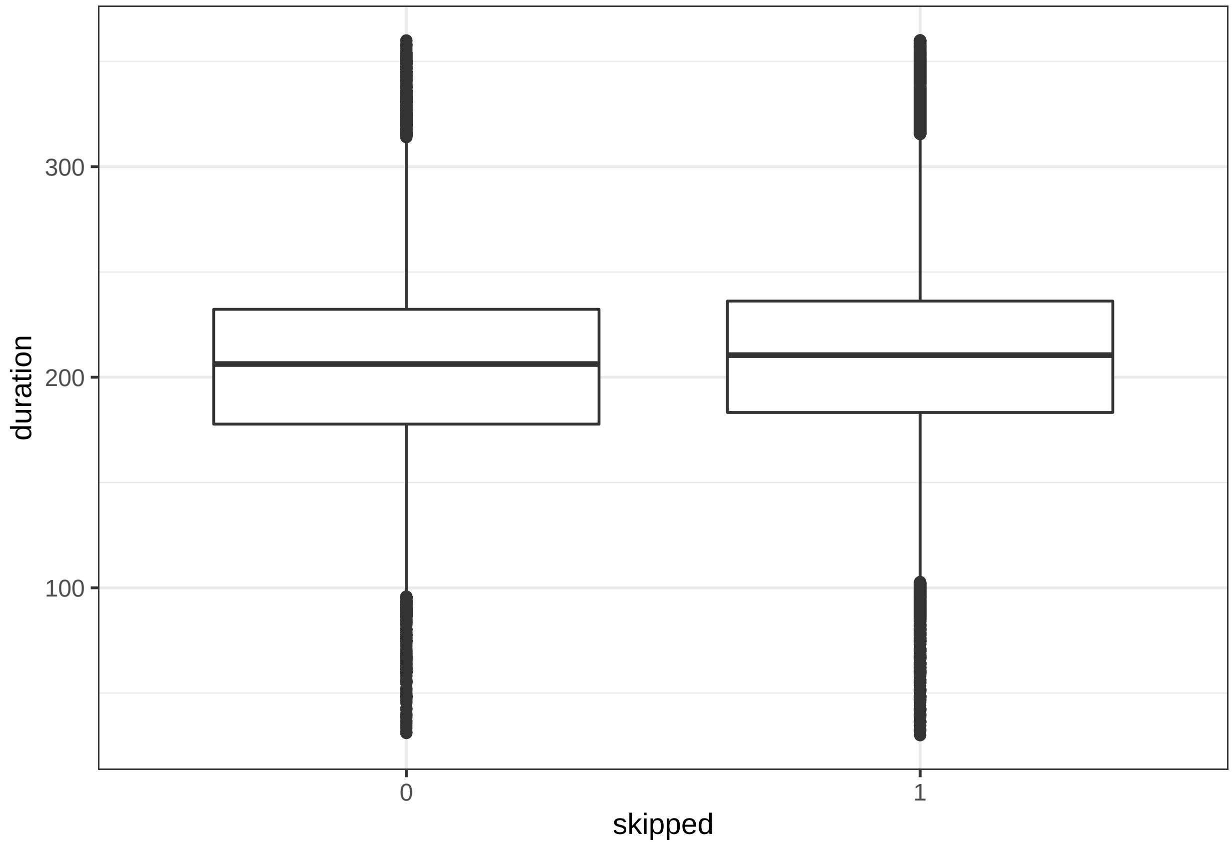
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



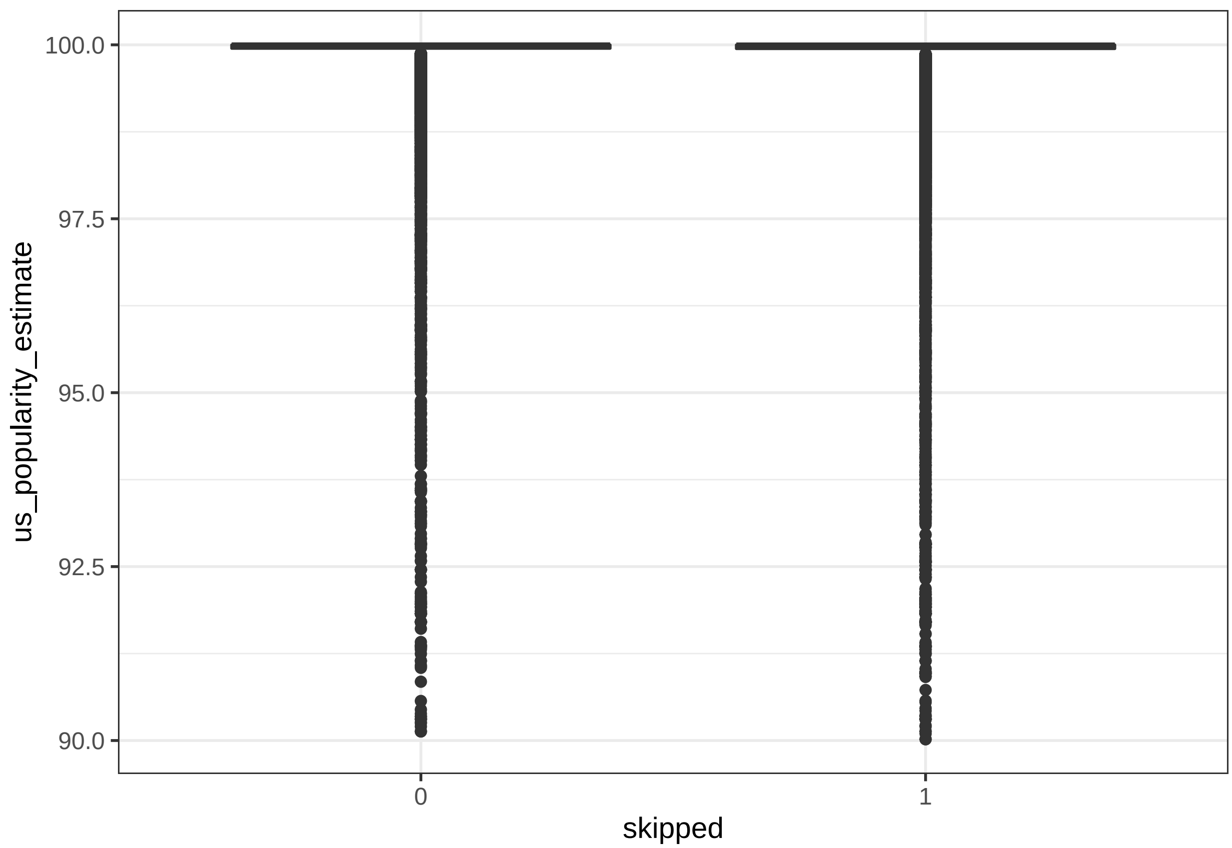

```
ggplot(data = Track_features, aes(x = skipped)) +
  geom_bar()
```



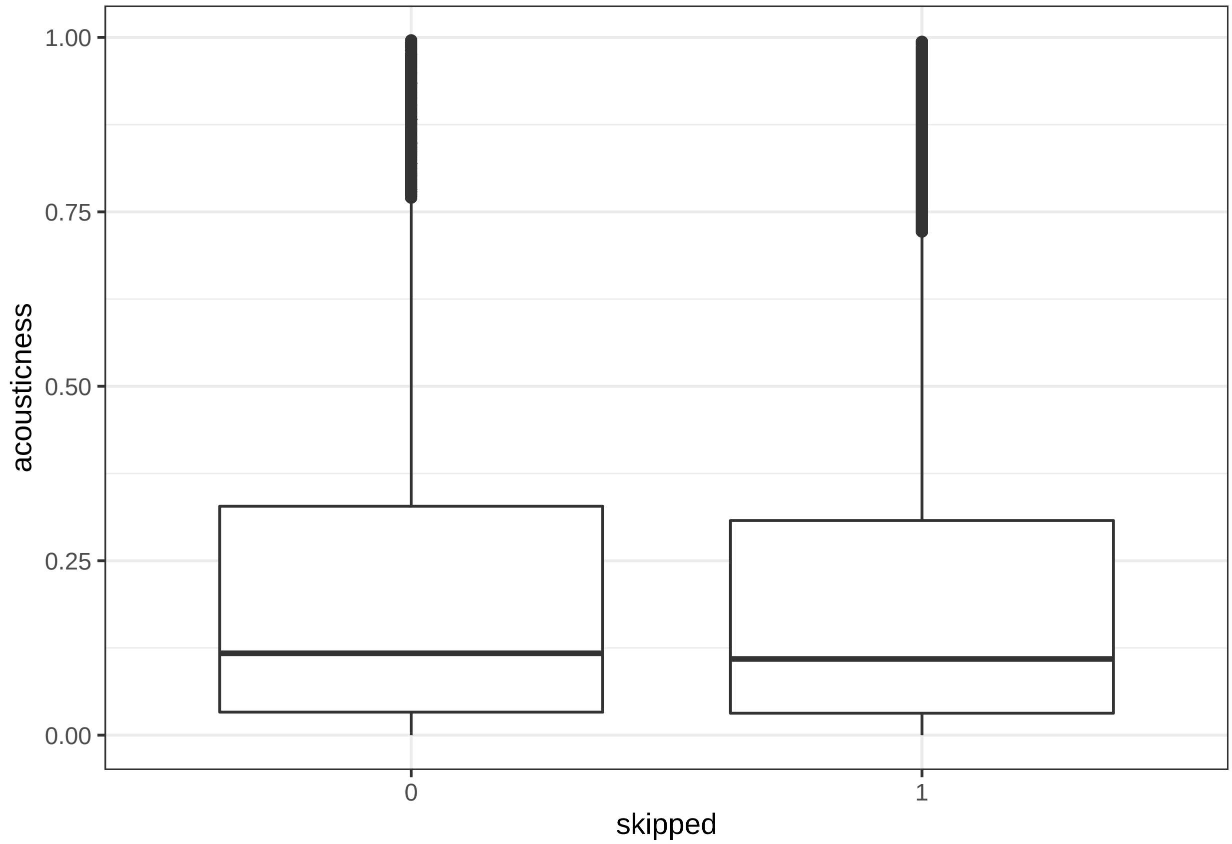

```
ggplot(Track_features, aes(skipped, duration)) + geom_boxplot()
```



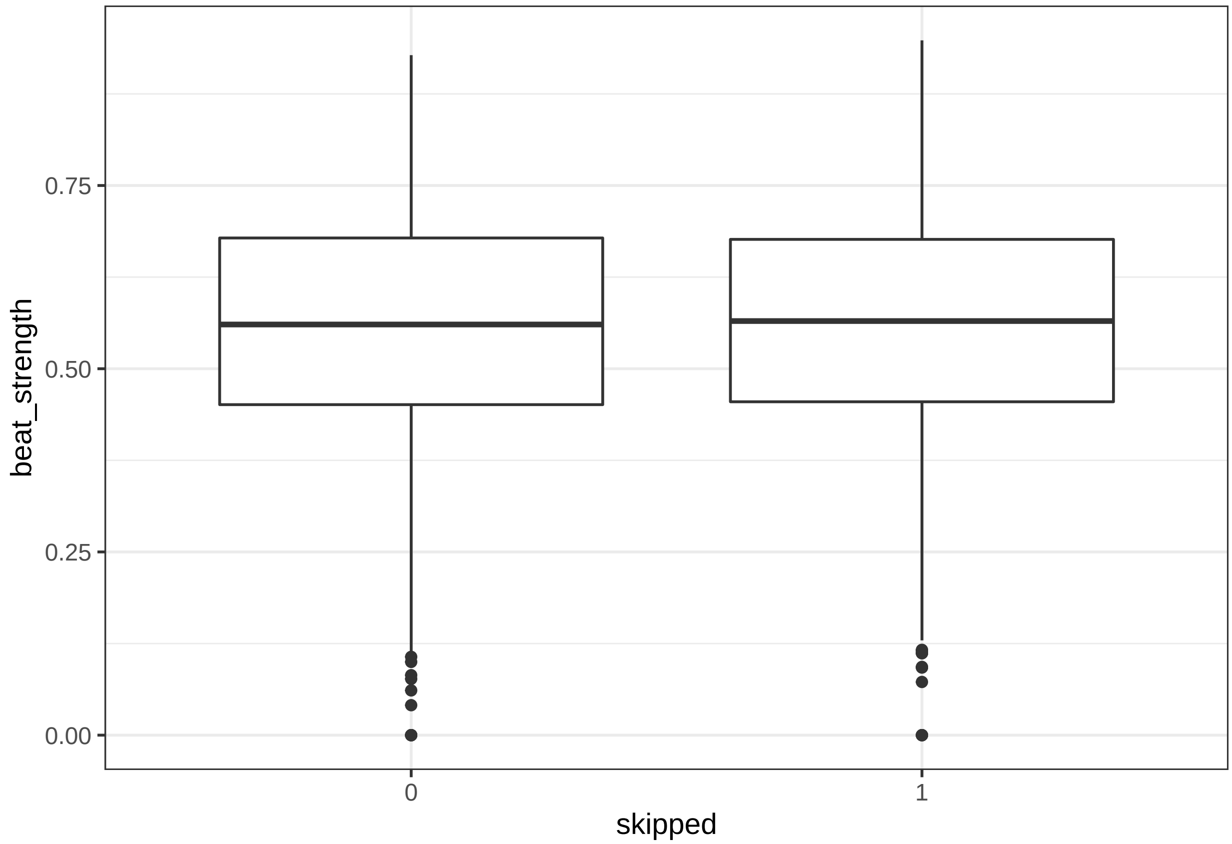

```
ggplot(Track_features, aes(skipped, us_popularity_estimate)) + geom_boxplot()
```



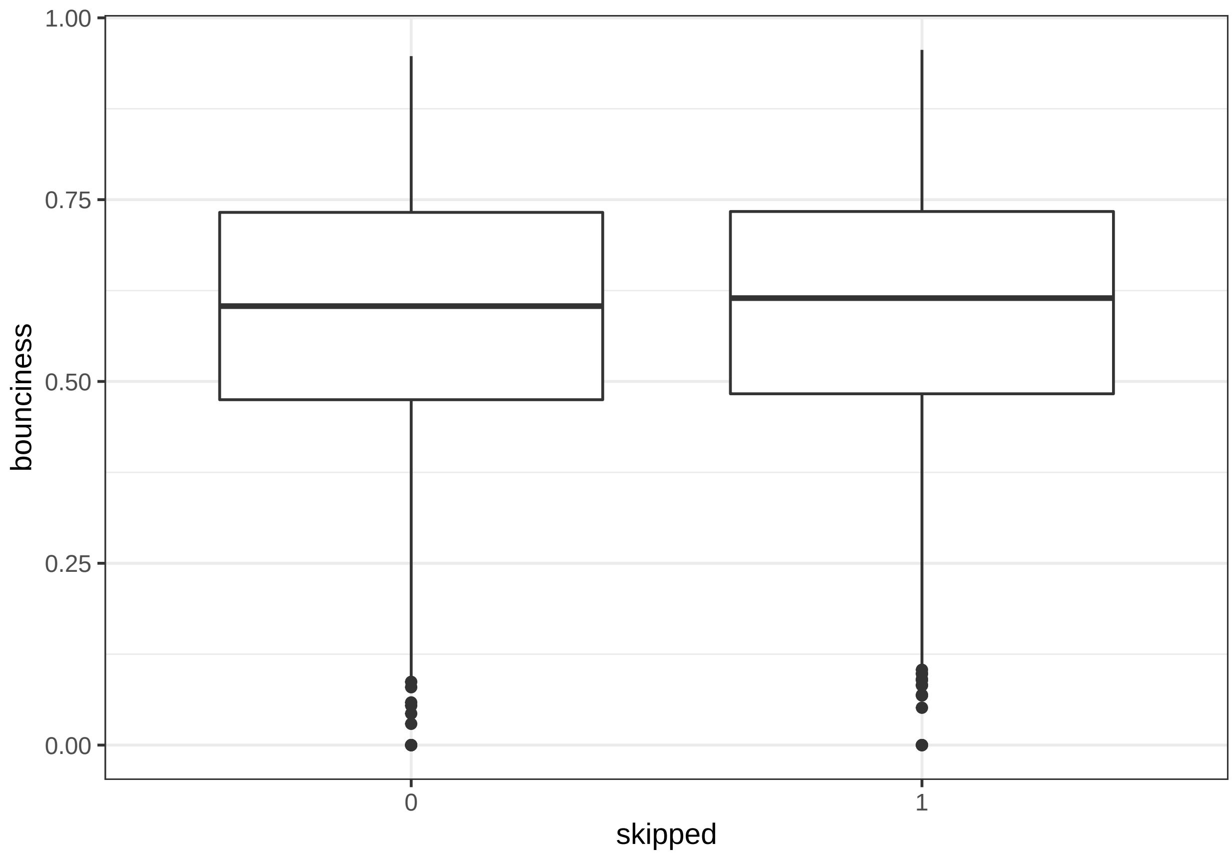

```
ggplot(Track_features, aes(skipped, acousticness)) + geom_boxplot()
```



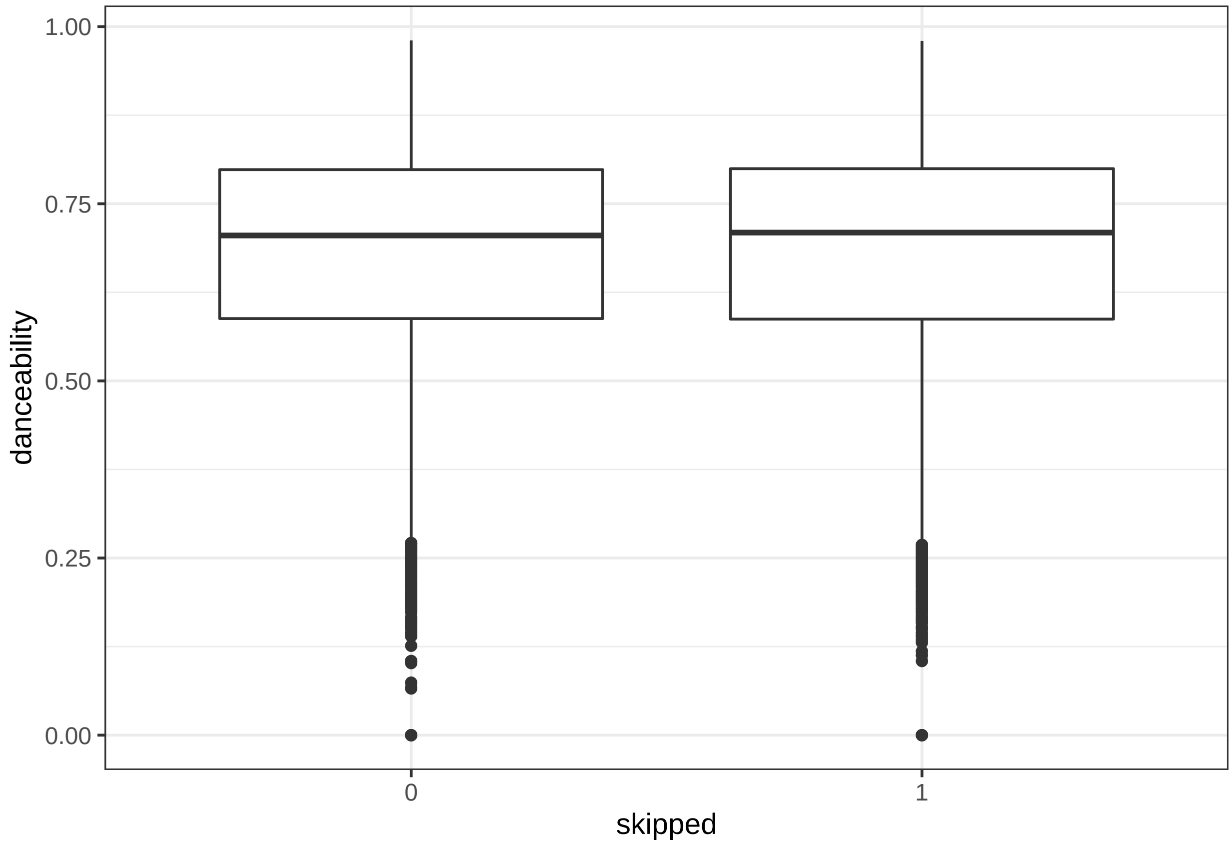

```
ggplot(Track_features, aes(skipped, beat_strength)) + geom_boxplot()
```



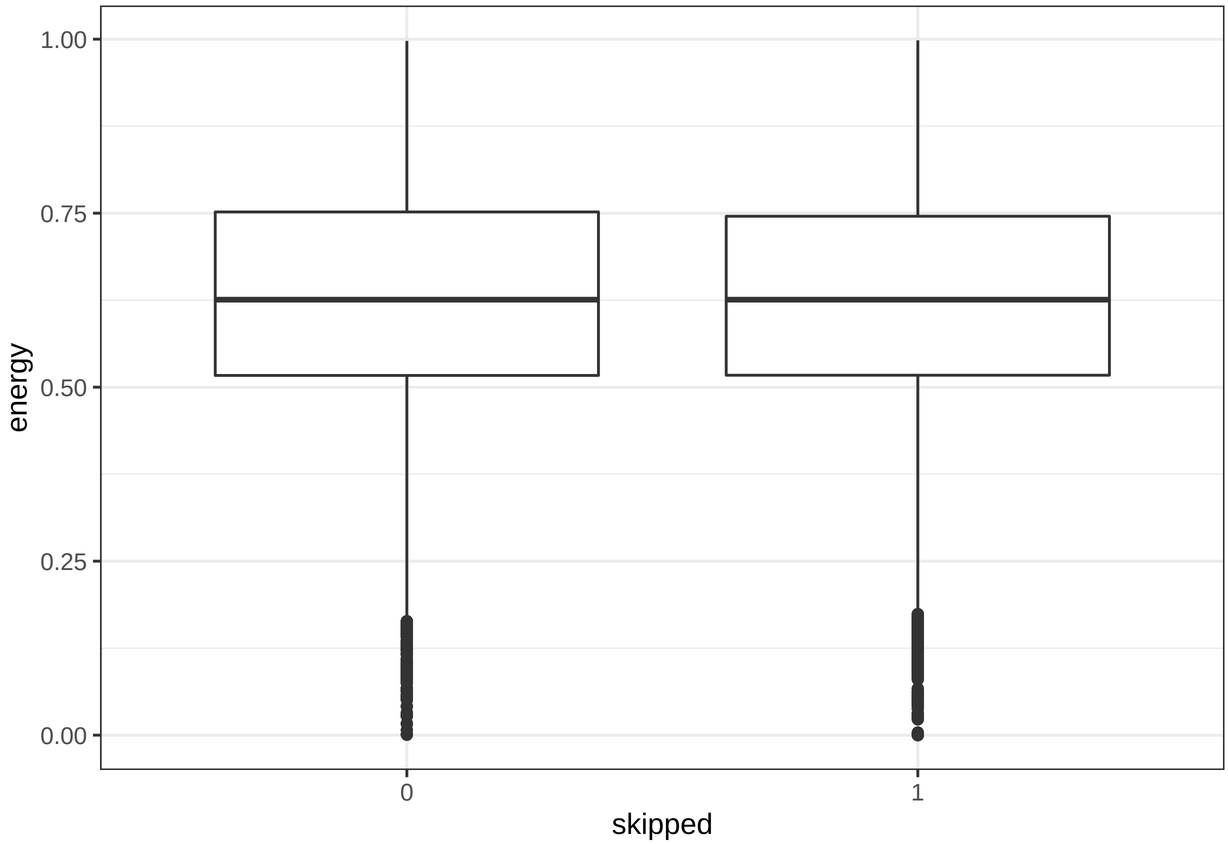

```
ggplot(Track_features, aes(skipped, bounciness)) + geom_boxplot()
```



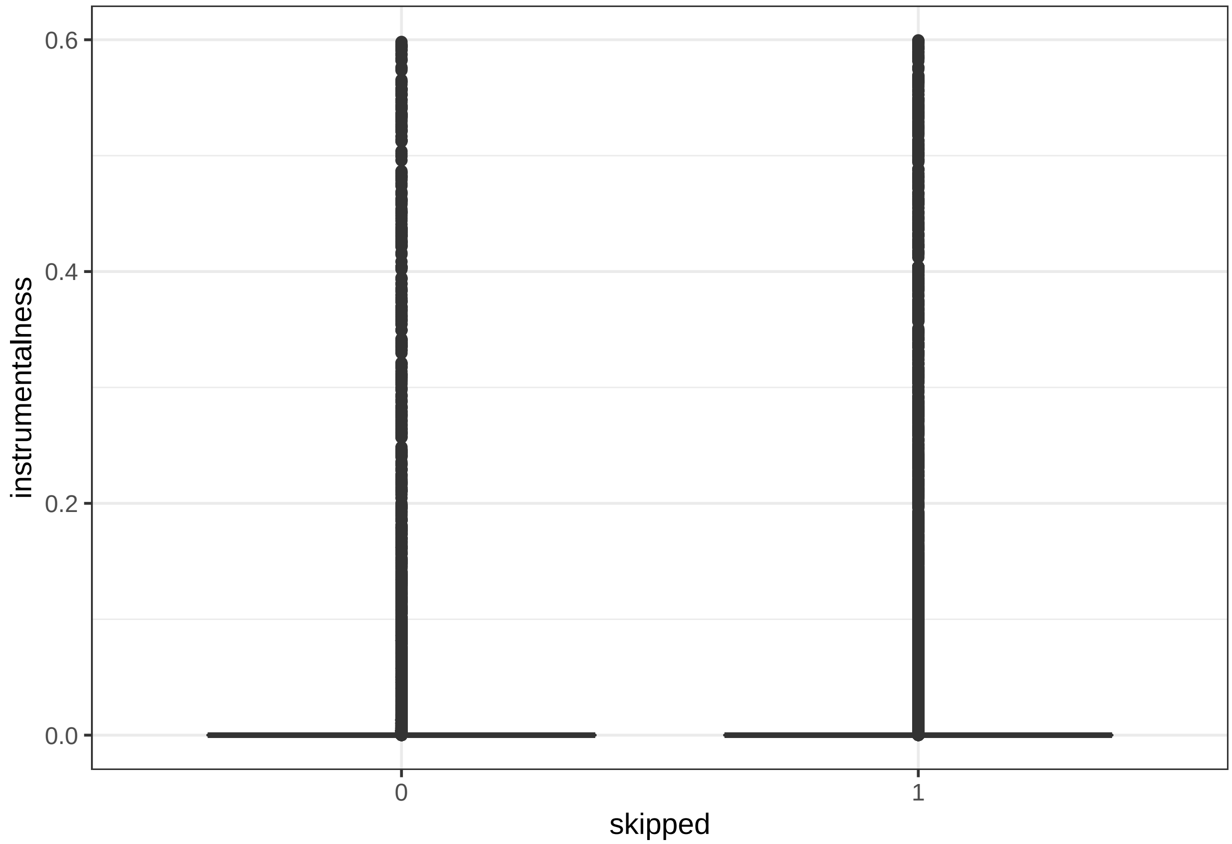

```
ggplot(Track_features, aes(skipped, danceability)) + geom_boxplot()
```



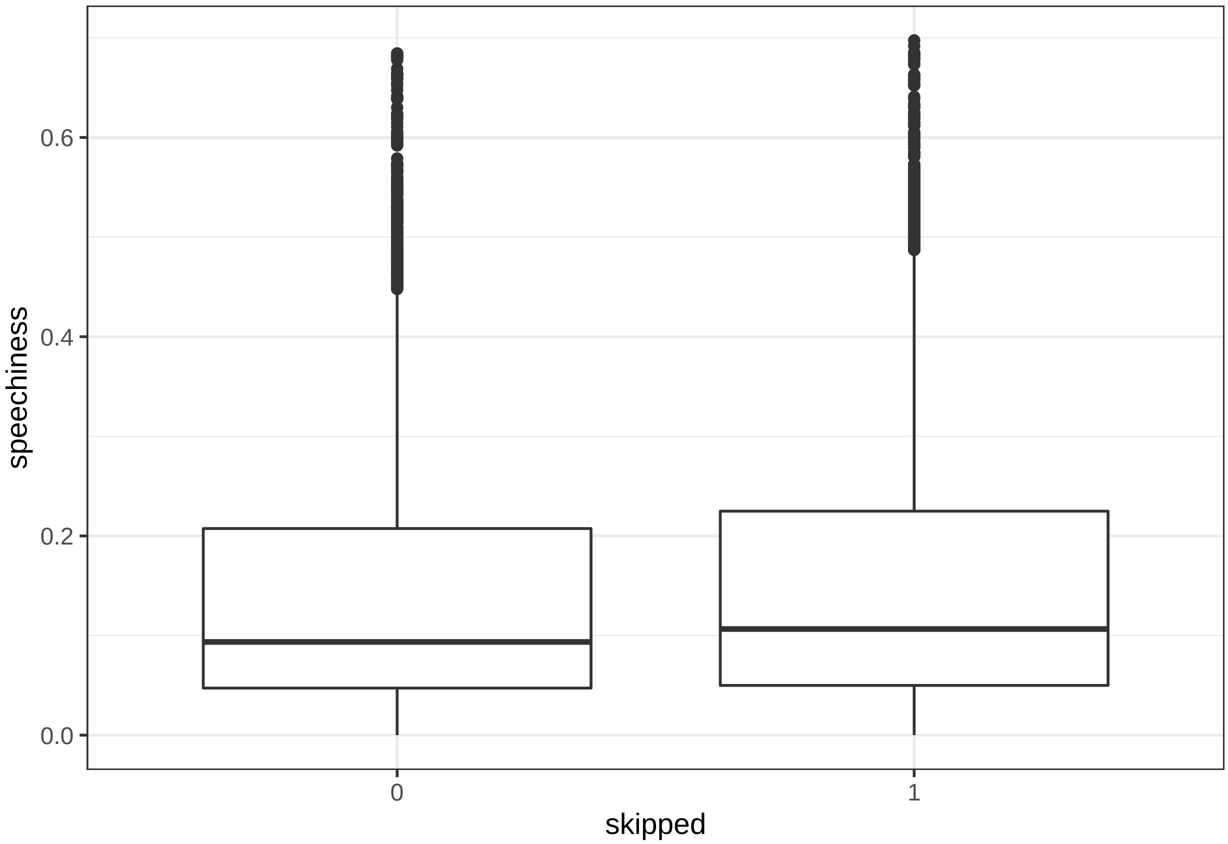

```
ggplot(Track_features, aes(skipped, energy)) + geom_boxplot()
```



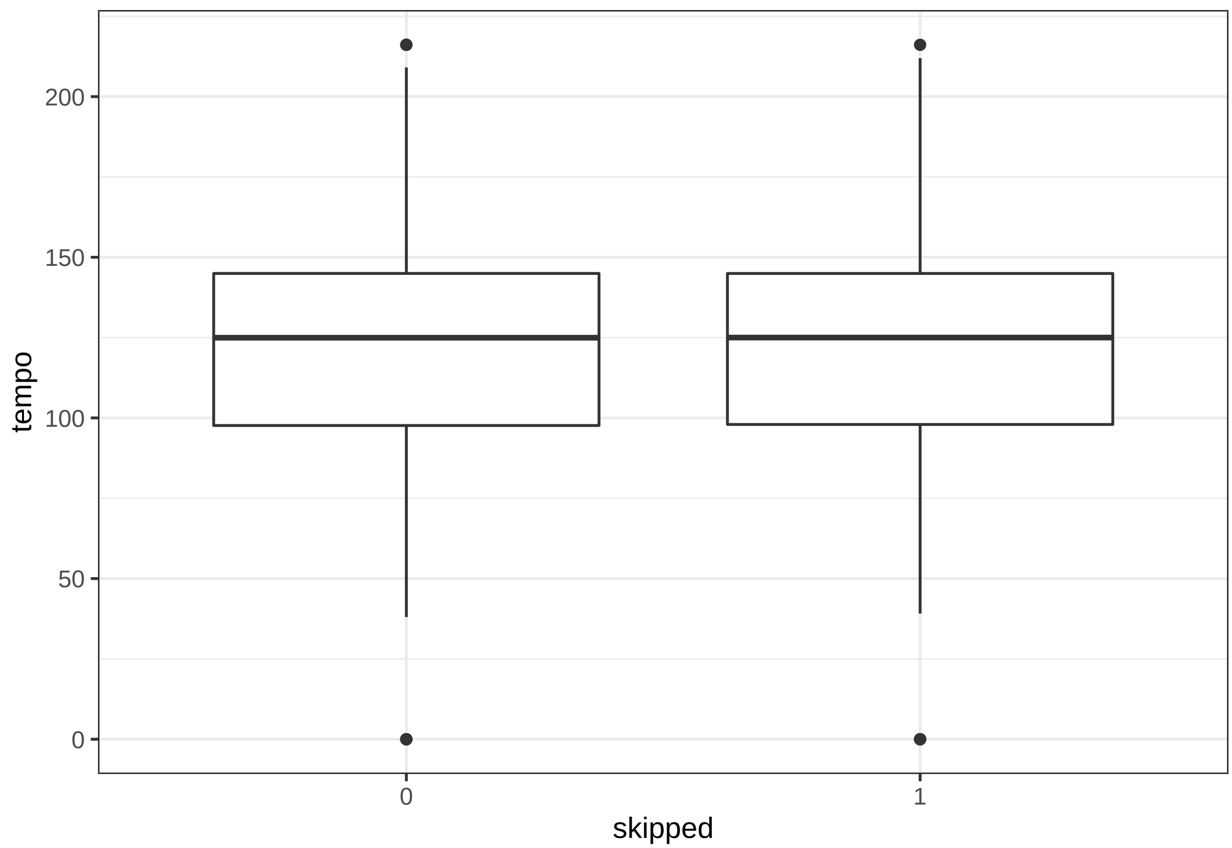

```
ggplot(Track_features, aes(skipped, instrumentalness)) + geom_boxplot()
```



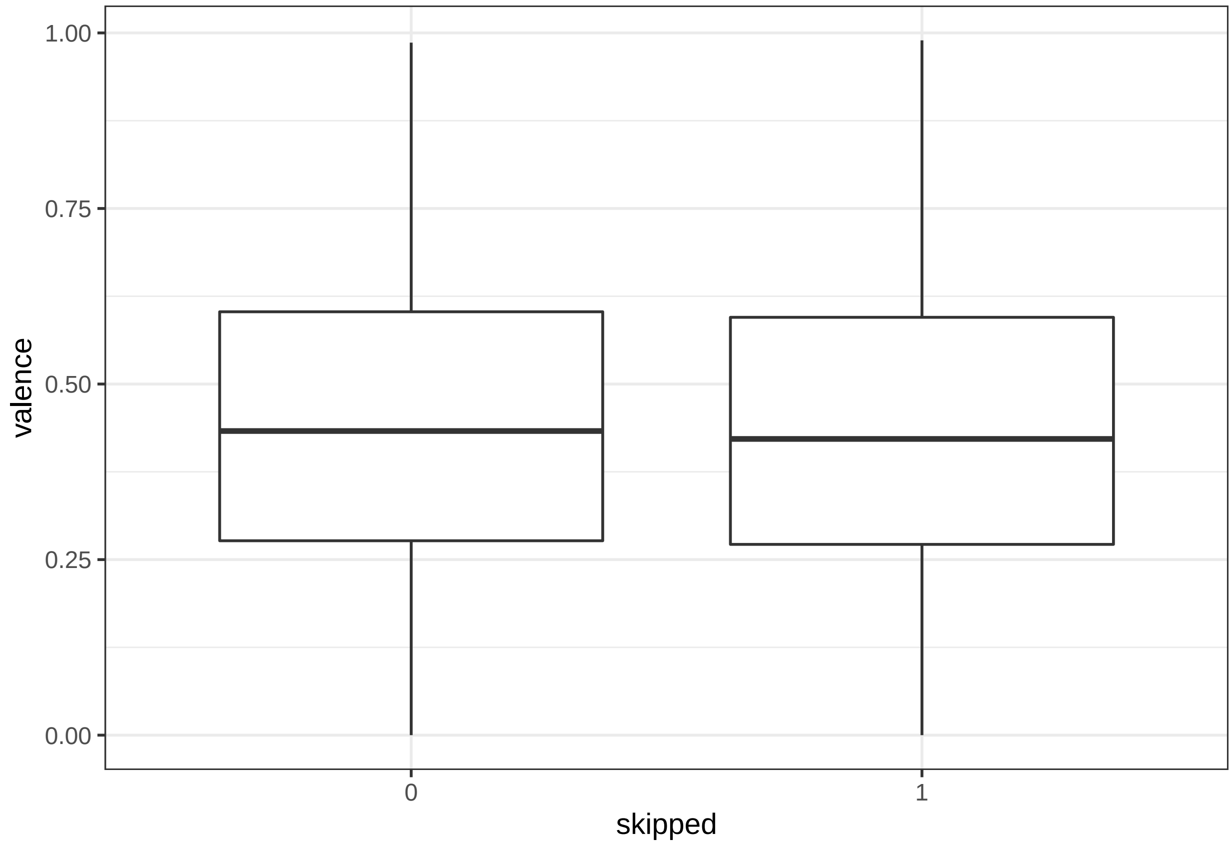

```
ggplot(Track_features, aes(skipped, speechiness)) + geom_boxplot()
```




```
ggplot(Track_features, aes(skipped, tempo)) + geom_boxplot()
```




```
ggplot(Track_features, aes(skipped, valence)) + geom_boxplot()
```




```
set.seed(1)
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)]
```

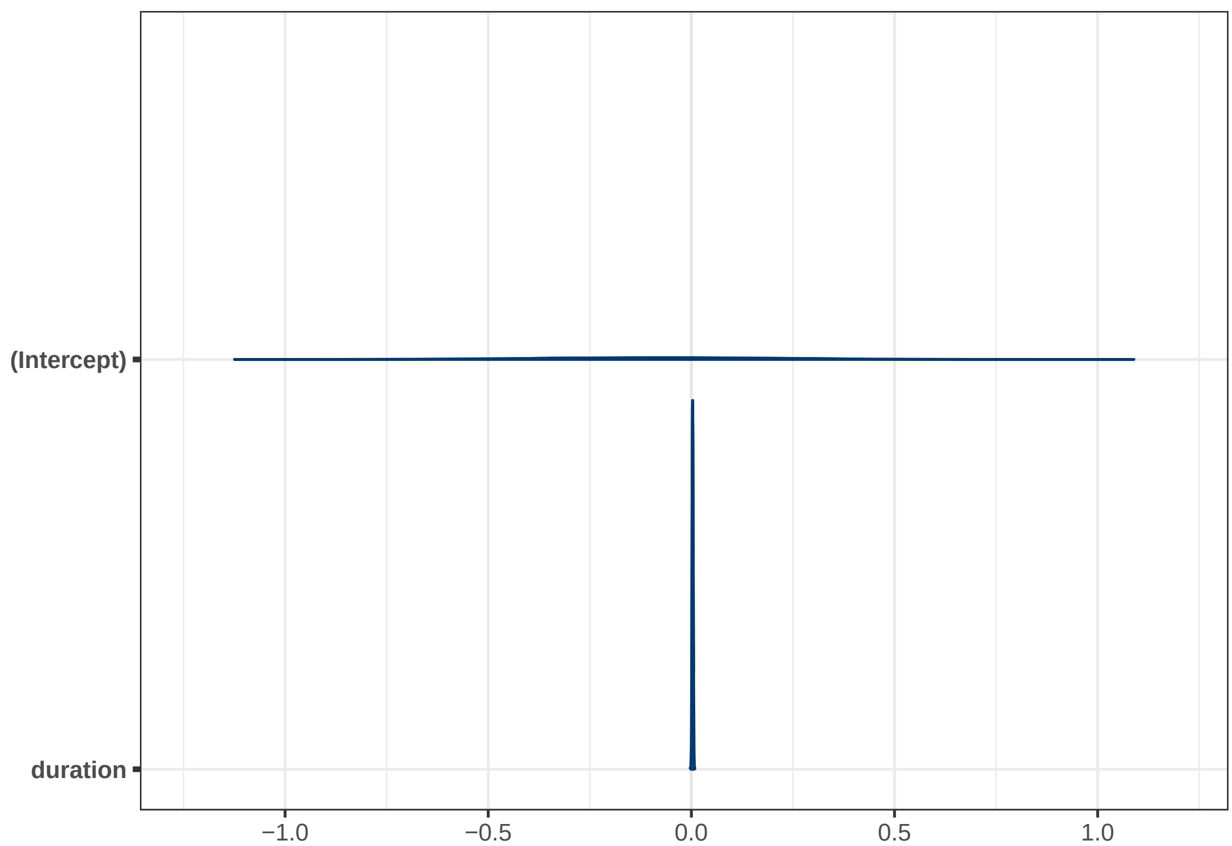
```
seed <- 1
posterior1 <- stan_glm(skipped ~ duration , data = Track_features_a,
                 family = binomial(link = "logit"),
                 prior = normal(0,1), prior_intercept = normal(0,1),
                 seed = seed,
                 refresh = 0)

summary(posterior1)
```

```
##
## Model Info:
##  function:     stan_glm
##  family:       binomial [logit]
##  formula:      skipped ~ duration
##  algorithm:    sampling
##  sample:       4000 (posterior sample size)
##  priors:       see help('prior_summary')
##  observations: 1000
##  predictors:   2
```

```
##
## Estimates:
##               mean   sd   10%   50%   90%
## (Intercept) -0.1    0.3 -0.5  -0.1   0.3
## duration     0.0    0.0  0.0   0.0   0.0
##
## 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)   0.0  1.0  2988
## duration      0.0  1.0  3080
## mean_PPD      0.0  1.0  1882
## log-posterior 0.0  1.0  1484
##
## For each parameter, mcse is Monte Carlo standard error, n_eff is a crude measure of effective sample
```

```
launch_shinystan(posterior1)
```

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



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

```
## (Intercept)    duration
##      -0.079       0.003
```

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

```
##                  5%   95%
## (Intercept) -0.589 0.425
## duration     0.001 0.005
```

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

```
##
## Computed from 4000 by 1000 log-likelihood matrix
##
##          Estimate   SE
## elpd_loo   -653.7  8.9
## p_loo         2.0  0.1
## looic      1307.4 17.8
## ------
## 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   -655.0  8.6
## p_loo         1.0  0.0
## looic      1310.0 17.2
## ------
## 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
## posterior1  0.0       0.0
## post0      -1.3       2.1
```