# Exercise set 4

30949

```
candidate_number <- 30949
set.seed(candidate_number)
```

## Simulation exercise

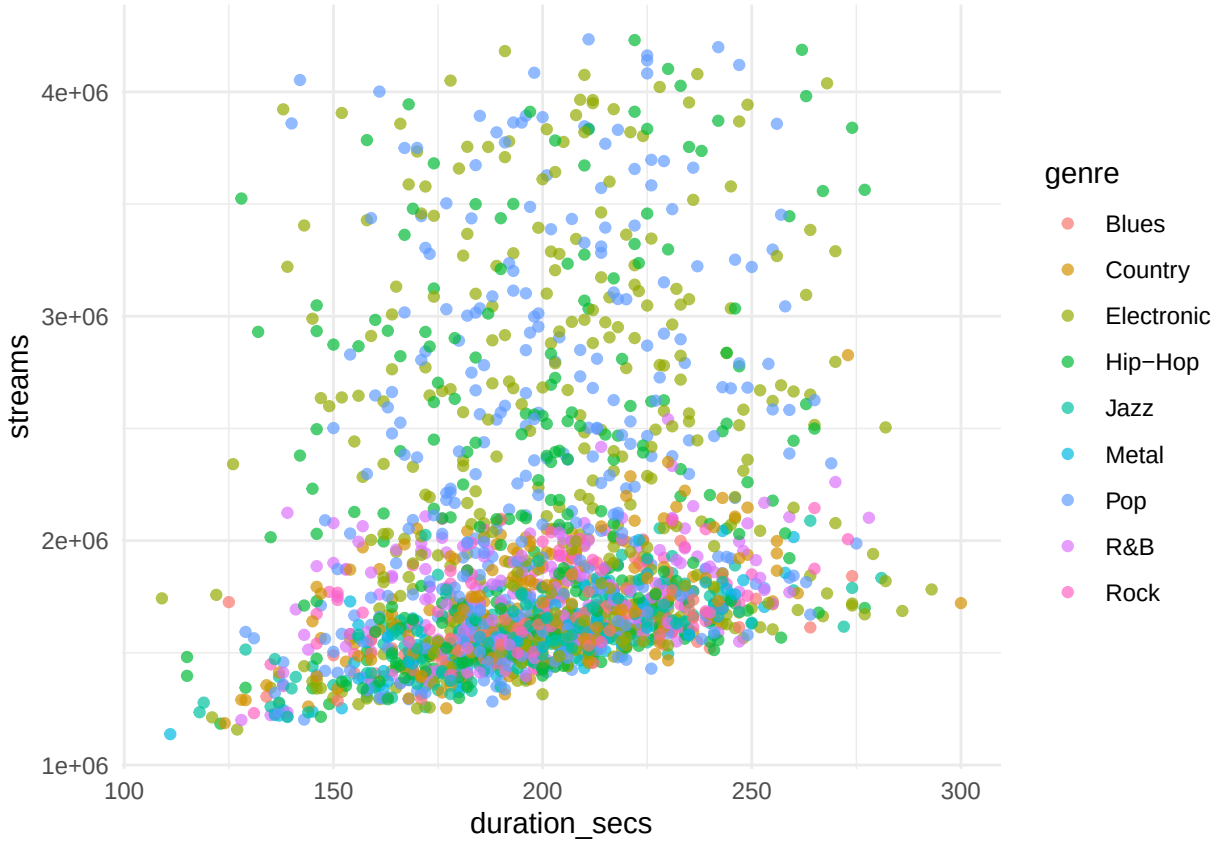
### Data generation

```
n <- 2000
# Generate categorical predictor(s)
genres <- c("Blues", "Country", "Electronic", "Hip-Hop", "Jazz", "Pop", "R&B", "Rock", "Metal")
genre <- factor(sample(genres, n, prob = c(1,2,4,3,2,4,2,1,1), replace = TRUE))
```

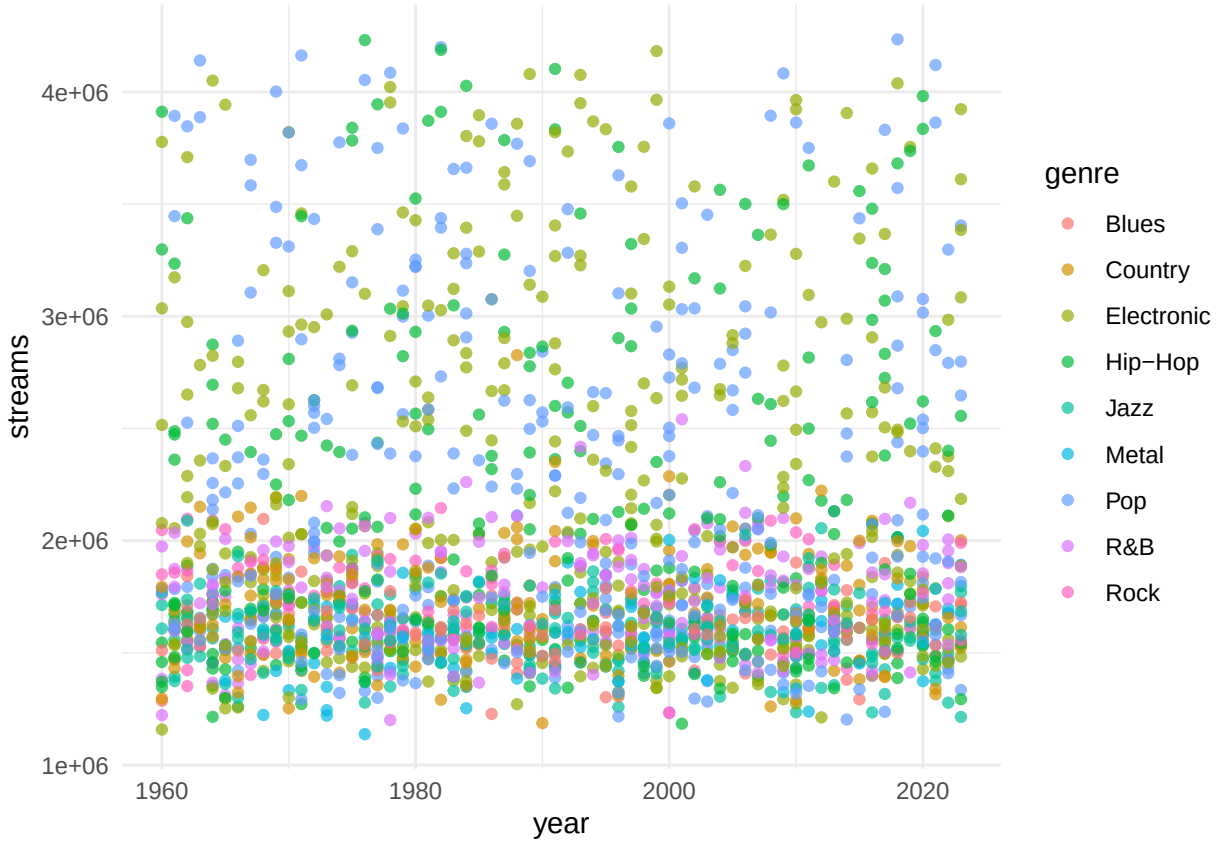
```
# Generate numeric predictor(s)
duration_secs <- round(rnorm(n, mean = 200, sd = 30))
year <- sample(1960:2023, n, replace = TRUE)
rating <- round(runif(n, min = 0, max = 5), digits = 2)
num_artists <- ceiling(rexp(n, rate = 1))
# Data frame version of predictors
predictors <- data.frame(duration_secs, year, rating, genre, num_artists)
head(predictors)
```

```
##   duration_secs year rating genre num_artists
## 1           204 2013   2.32   Pop           1
## 2           184 1992   2.17   Pop           1
## 3           201 1964   3.54   Pop           1
## 4           186 2006   0.16   R&B           5
## 5           203 1967   2.72 Blues           2
## 6           227 2003   1.07  Jazz           4
```

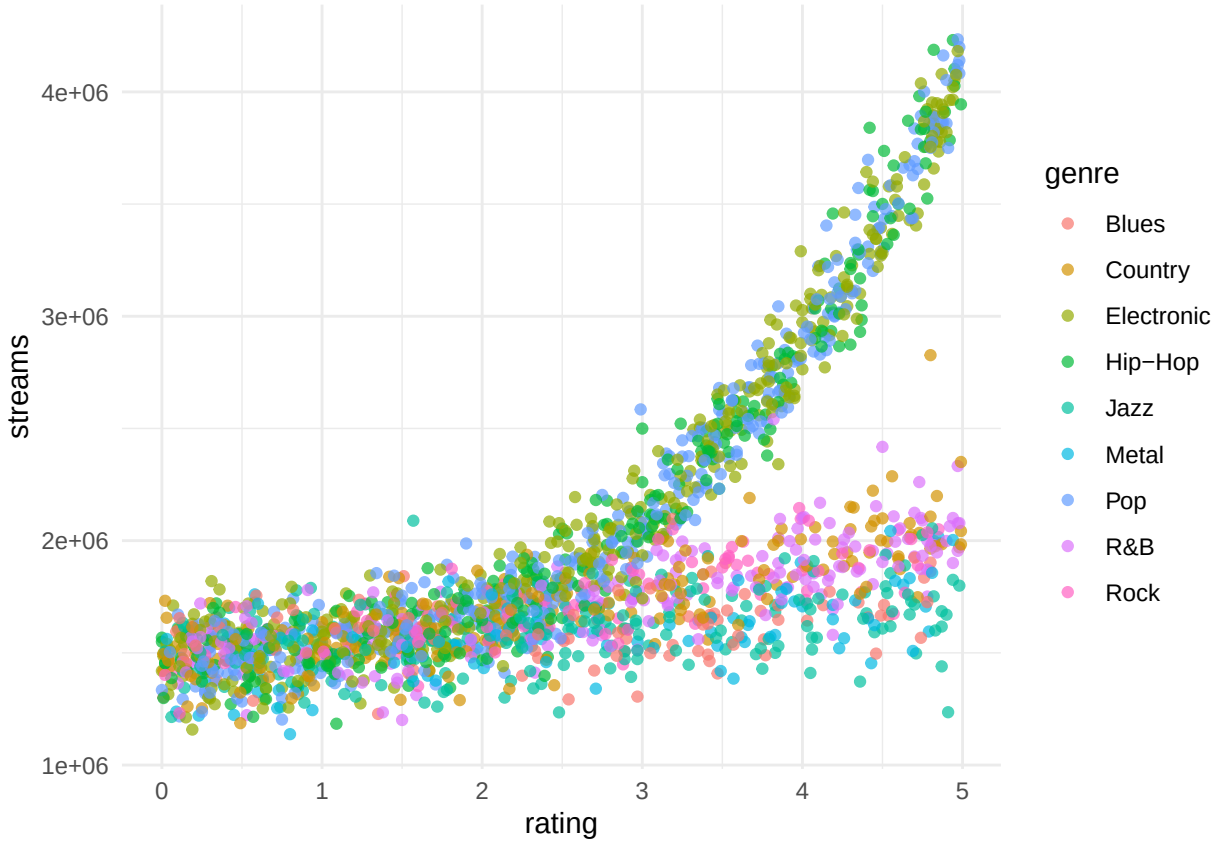
```
# CEF
CEF <- function(duration_secs, year, rating, genre, num_artists) {
  if_else(genre %in% c("Electronic", "Hip-Hop", "Pop"),
          50000 + 100000*sqrt(duration_secs) + 50000*sin(year) + 20000*(rating^3) + 5000*num_artists + 1
          if_else(genre %in% c("Country", "R&B", "Rock"),
                  50000 + 100000*sqrt(duration_secs) + 50000*sin(year) + 20000*(rating^2) + 5000*num_ar
                  50000 + 100000*sqrt(duration_secs) + 50000*sin(year) + 20000*rating + 5000*num_artists
}
training_data <- predictors |> mutate(streams = CEF(duration_secs, year, rating, genre, num_artists) + r
ggplot(training_data, aes(duration_secs, streams, colour = genre)) + geom_point(alpha = .7)
```



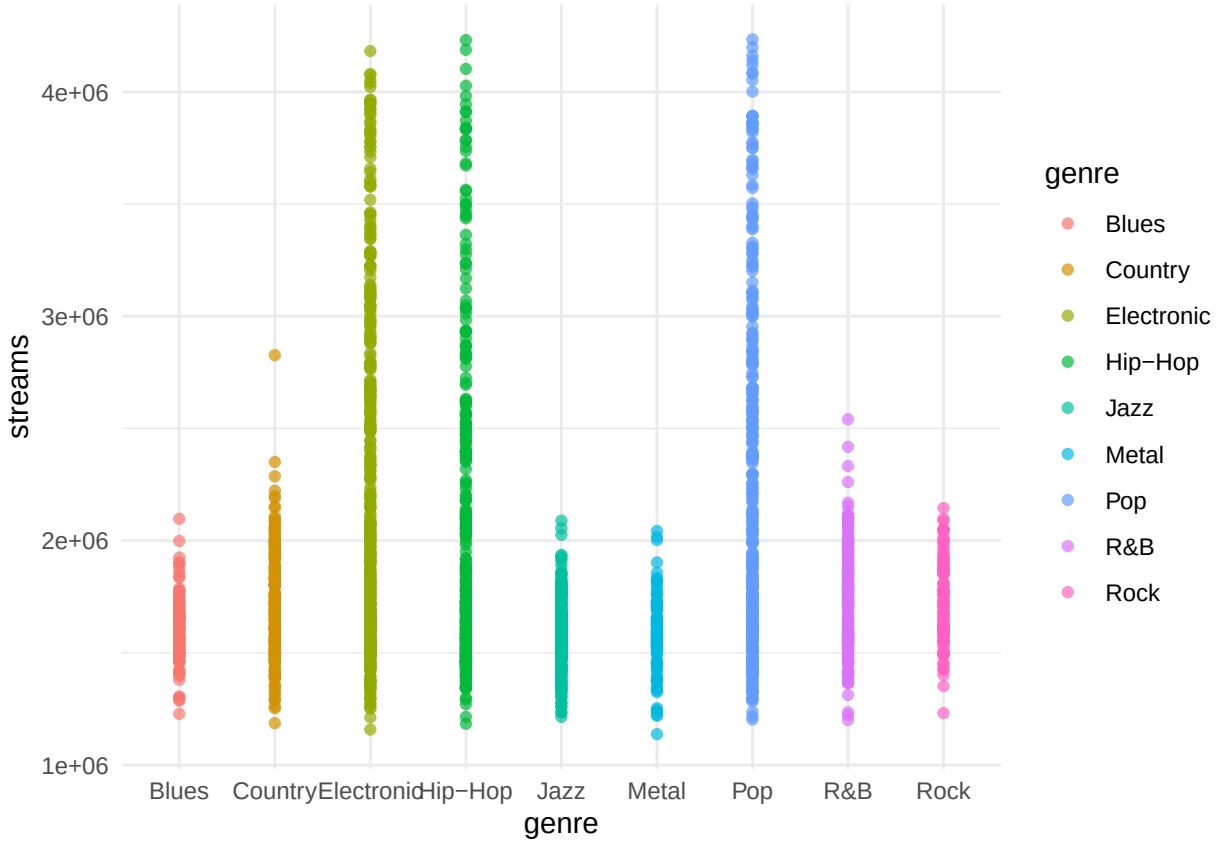

```
ggplot(training_data, aes(year, streams, colour = genre)) + geom_point(alpha = .7)
```



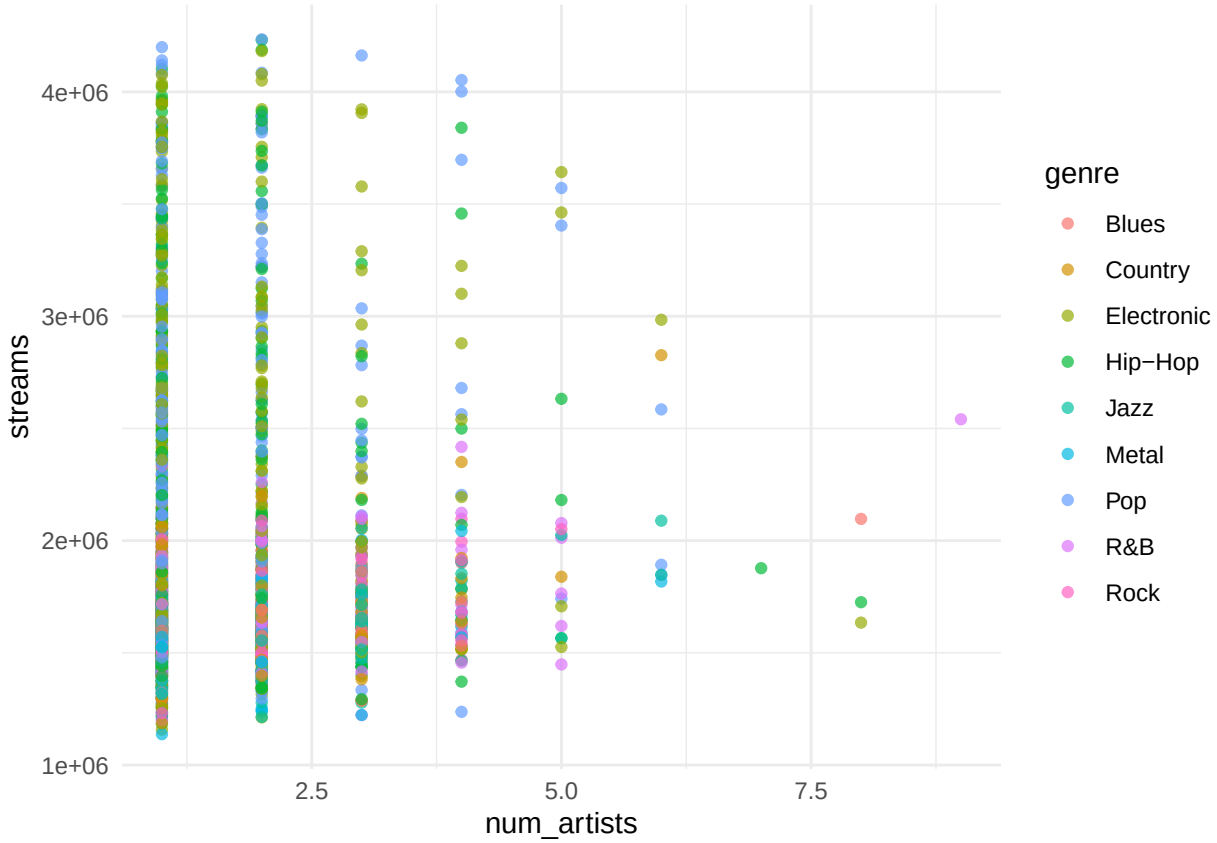

```
ggplot(training_data, aes(rating, streams, colour = genre)) + geom_point(alpha = .7)
```




```
ggplot(training_data, aes(genre, streams, colour = genre)) + geom_point(alpha = .7)
```

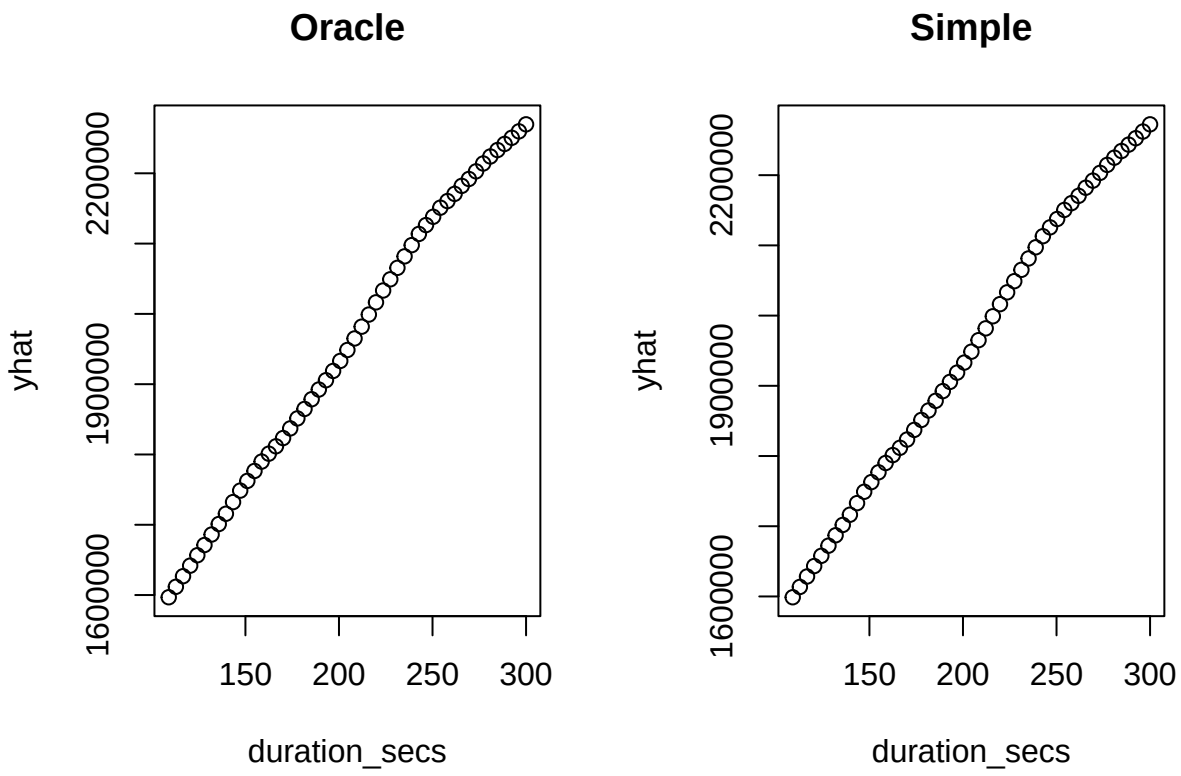



```
ggplot(training_data, aes(num_artists, streams, colour = genre)) + geom_point(alpha = .7)
```

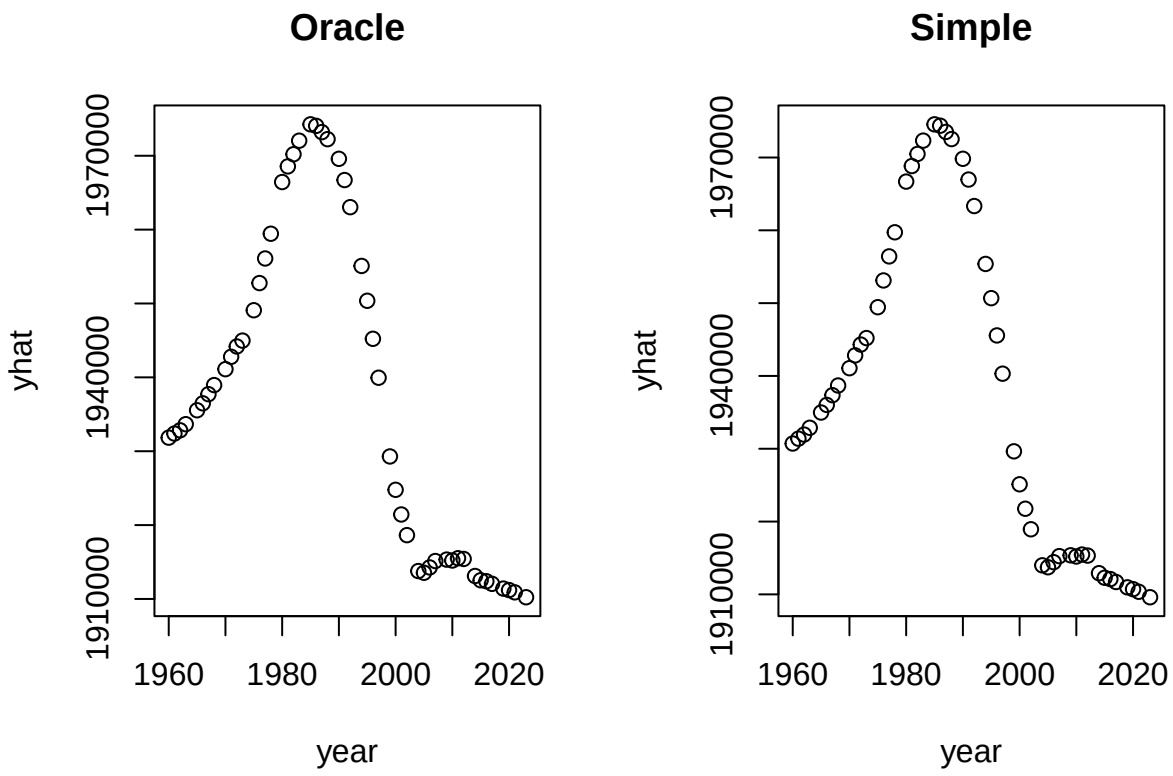



### Additive models

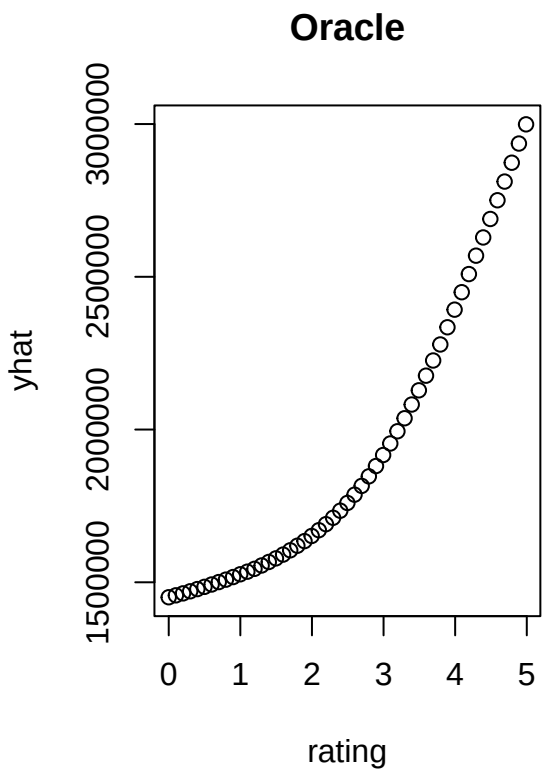
```
gam_oracle <- gam(streams ~ lo(duration_secs) + lo(year) + lo(rating) + genre + lo(num_artists) + lo(nu


gam_simple <- gam(streams ~ lo(duration_secs) + lo(year) + lo(rating) + genre + lo(num_artists), data =

par(mfrow = c(1,2))
plot(partial(gam_oracle, pred.var = "duration_secs"), main = "Oracle")
plot(partial(gam_simple, pred.var = "duration_secs"), main = "Simple")
```

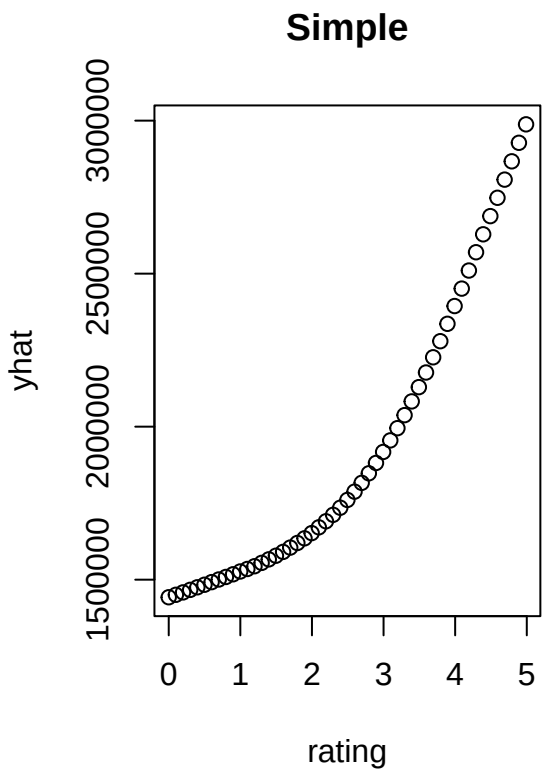



```
plot(partial(gam_oracle, pred.var = "year"), main = "Oracle")
plot(partial(gam_simple, pred.var = "year"), main = "Simple")
```

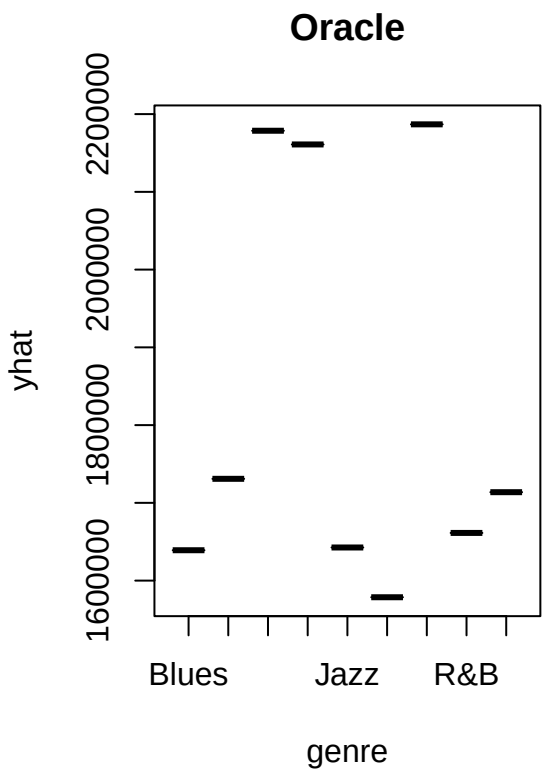


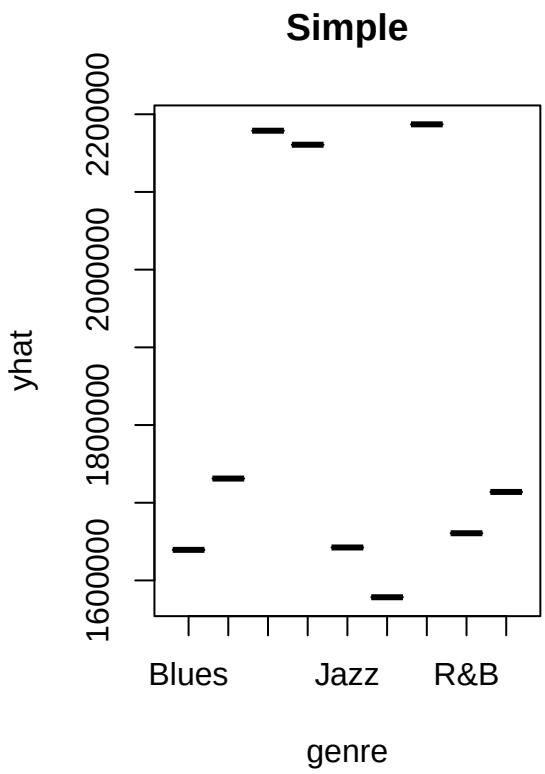

```
plot(partial(gam_oracle, pred.var = "rating"), main = "Oracle")
plot(partial(gam_simple, pred.var = "rating"), main = "Simple")
```



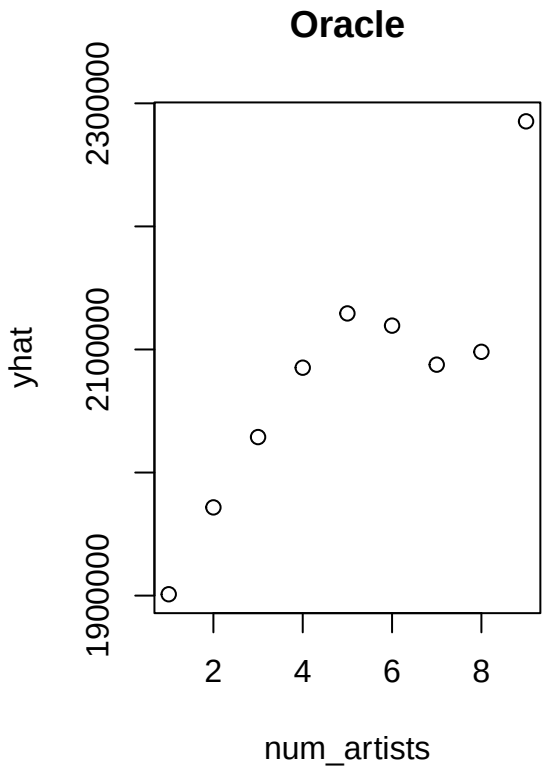



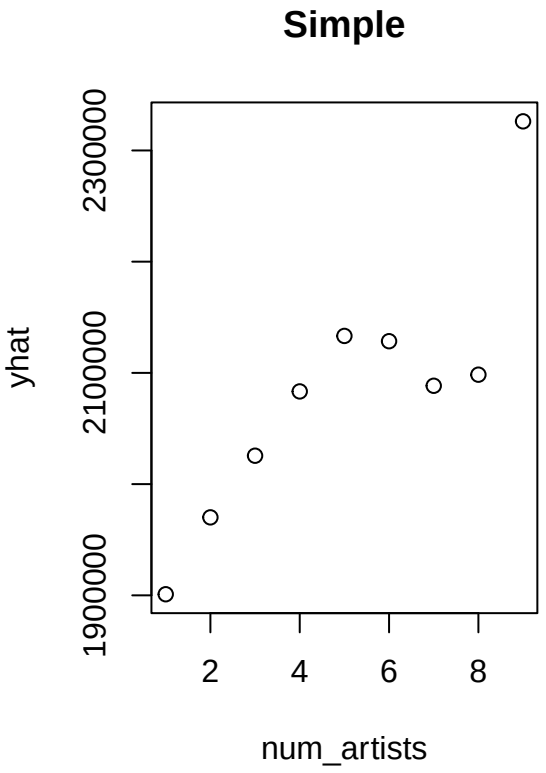

```
plot(partial(gam_oracle, pred.var = "genre"), main = "Oracle")
plot(partial(gam_simple, pred.var = "genre"), main = "Simple")
```



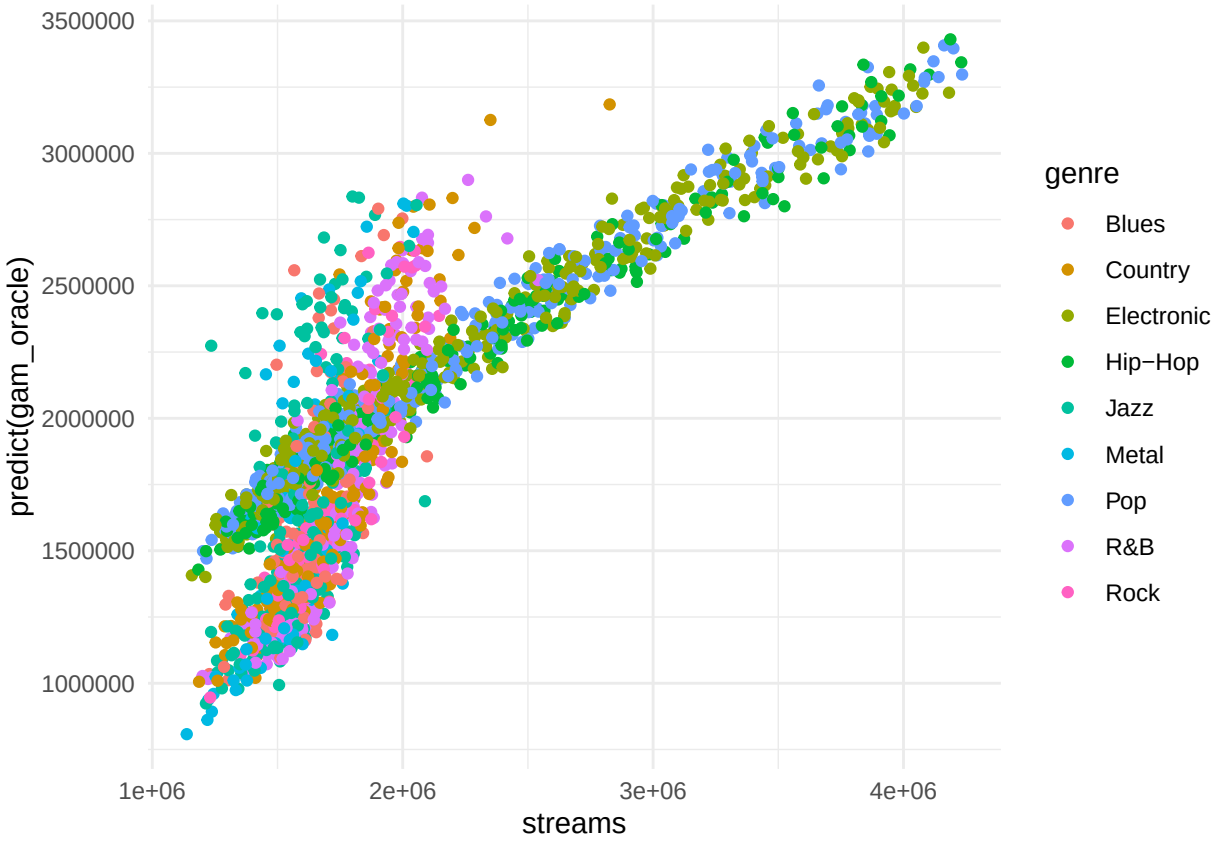




```
plot(partial(gam_oracle, pred.var = "num_artists"), main = "Oracle")
plot(partial(gam_simple, pred.var = "num_artists"), main = "Simple")
```

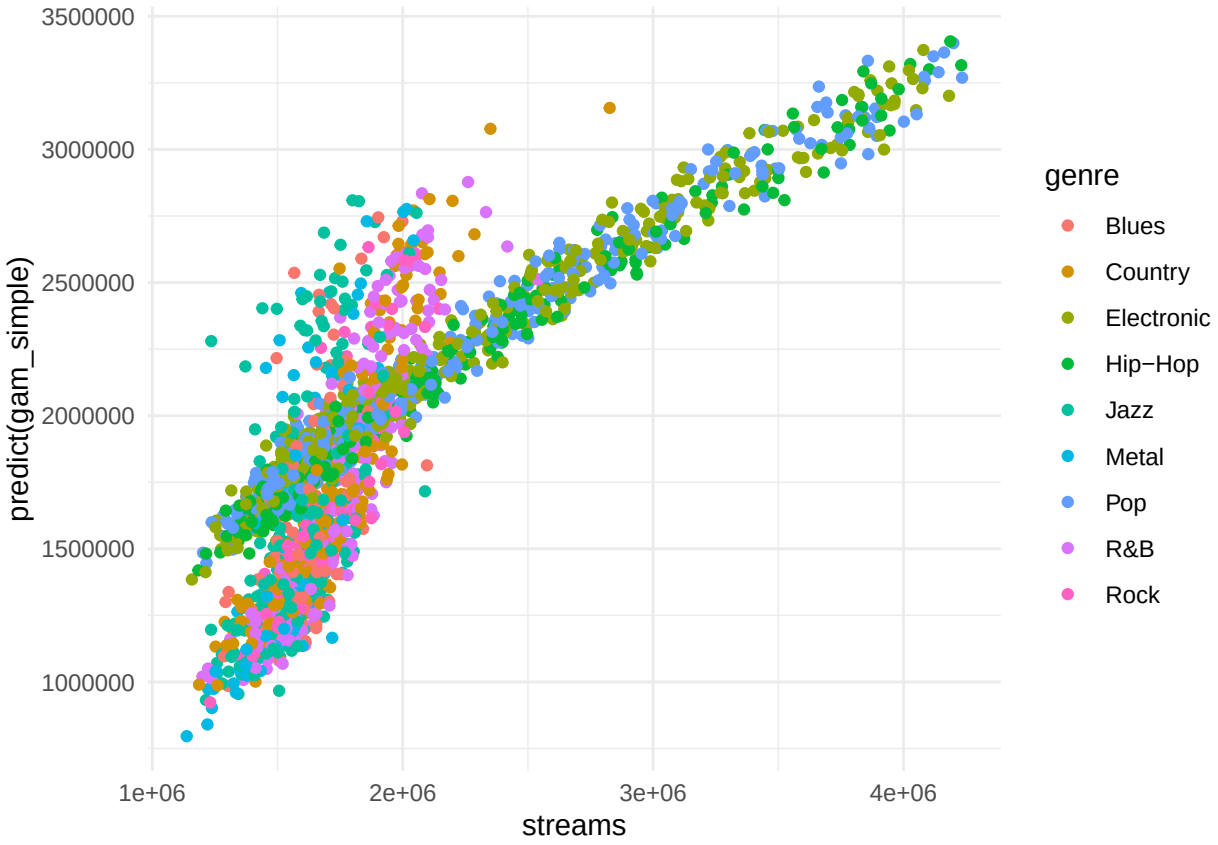



```
par(mfrow = c(1,1))
plot_oracle <- ggplot(training_data, aes(x = streams, y = predict(gam_oracle))) +
  geom_point(aes(colour = genre))
plot_oracle
```

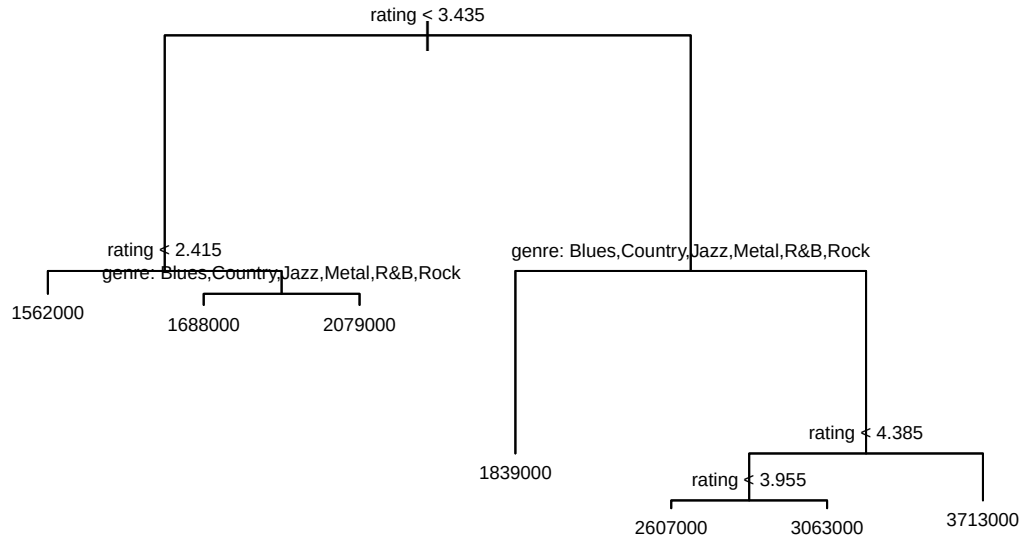



```
plot_simple <- ggplot(training_data, aes(x = streams, y = predict(gam_simple))) +
  geom_point(aes(colour = genre))
plot_simple
```

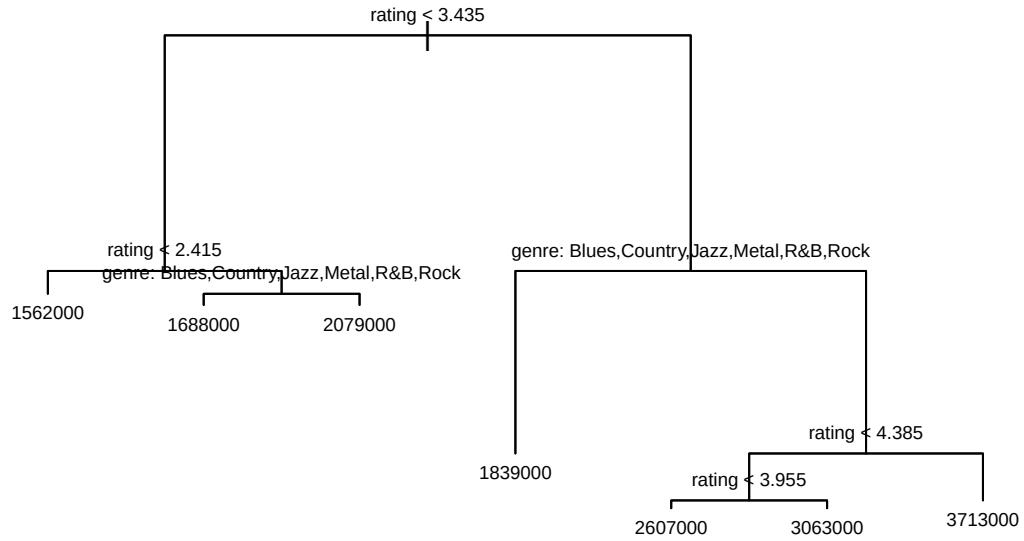



**Tree based models**

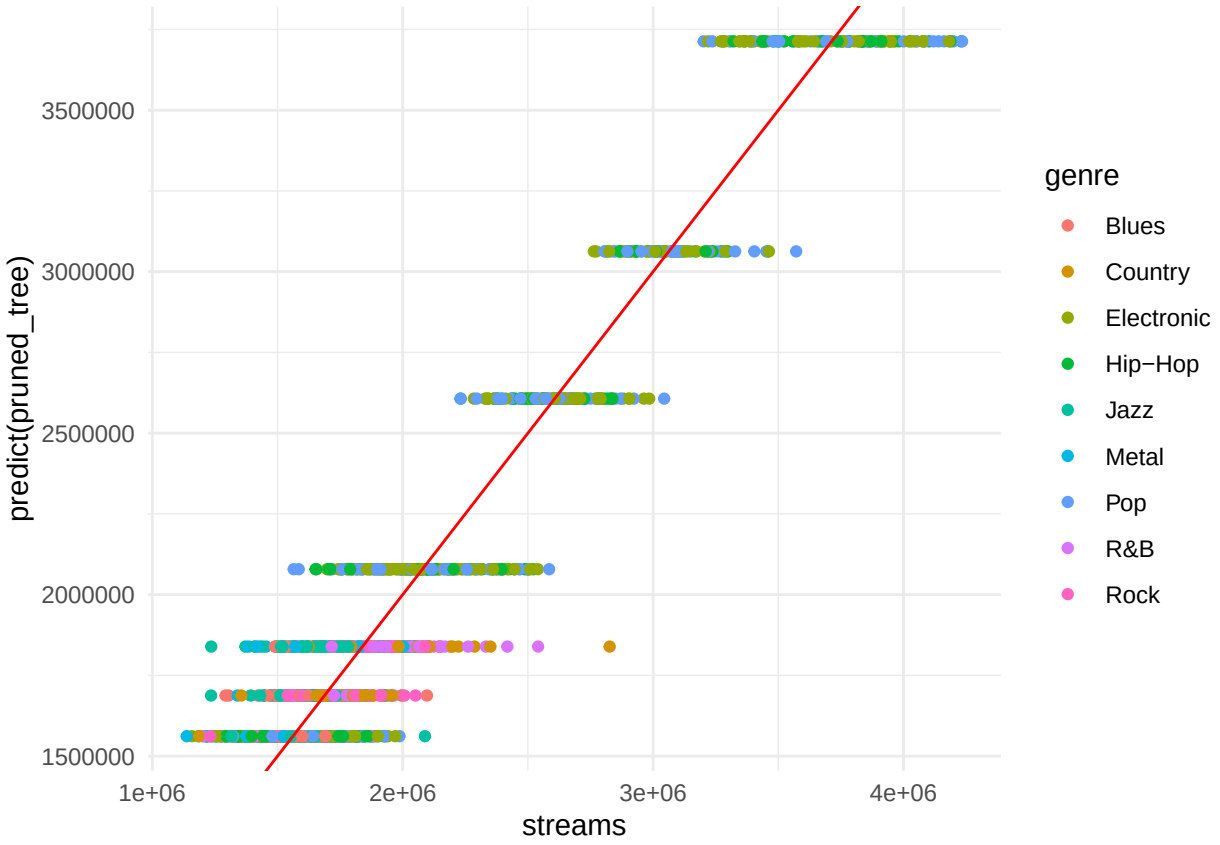
```
single_tree <- tree(streams ~ ., data = training_data)
plot(single_tree, main = "Original Tree")
text(single_tree, pretty = 0, cex = 0.6)
```



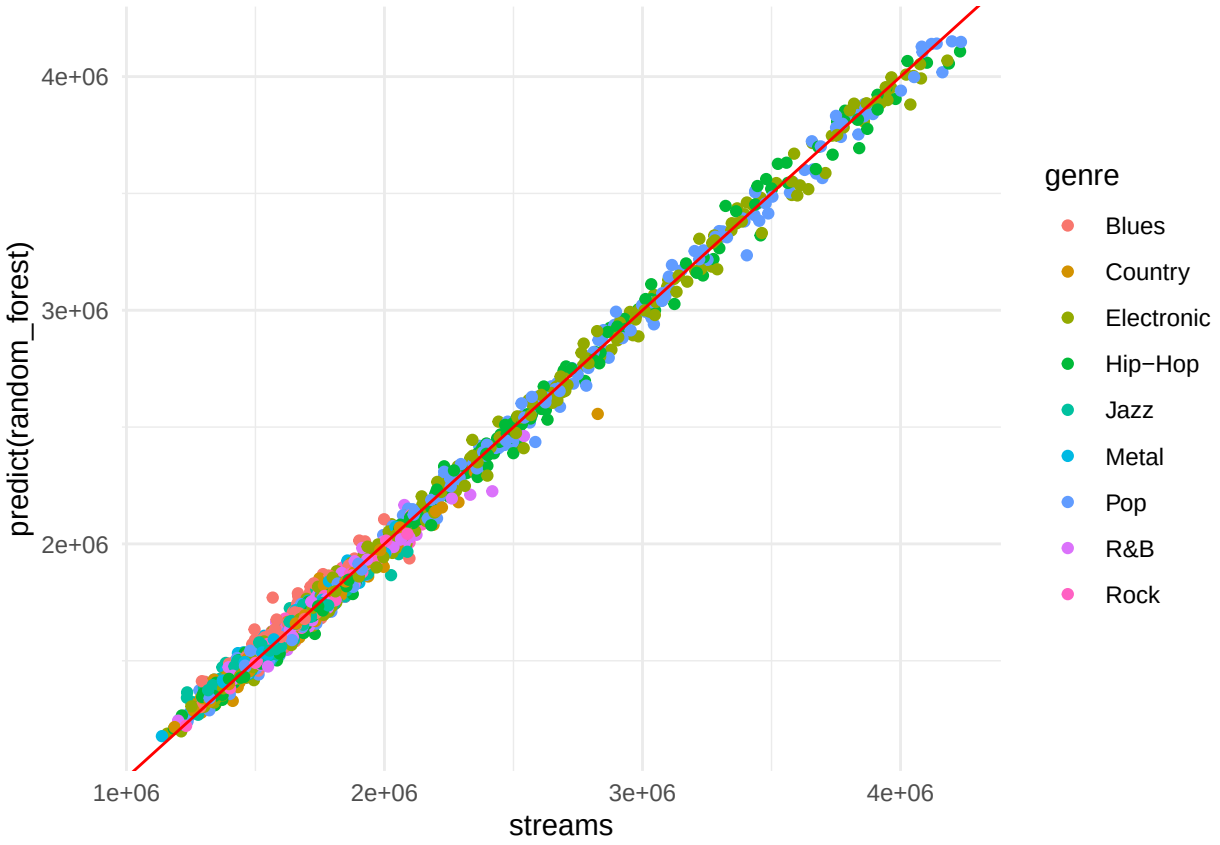

```r
cv_result <- cv.tree(single_tree, K = 10)
best_size <- cv_result$size[which.min(cv_result$dev)]
pruned_tree <- prune.tree(single_tree, best = best_size)
plot(pruned_tree, main = "Pruned Tree")
text(pruned_tree, pretty = 0, cex = 0.6)
```



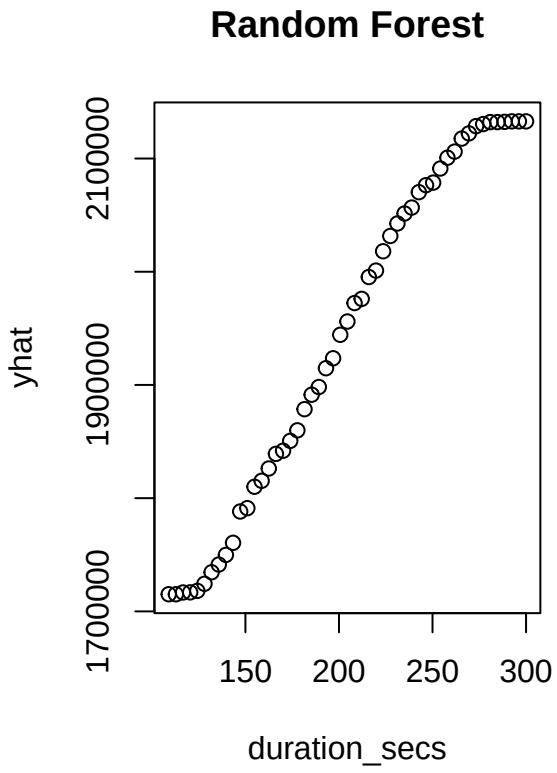

```
plot_tree <- ggplot(training_data, aes(x = streams, y = predict(pruned_tree))) +
  geom_point(aes(colour = genre)) +
  geom_abline(intercept = 0, slope = 1, color = "red")
plot_tree
```

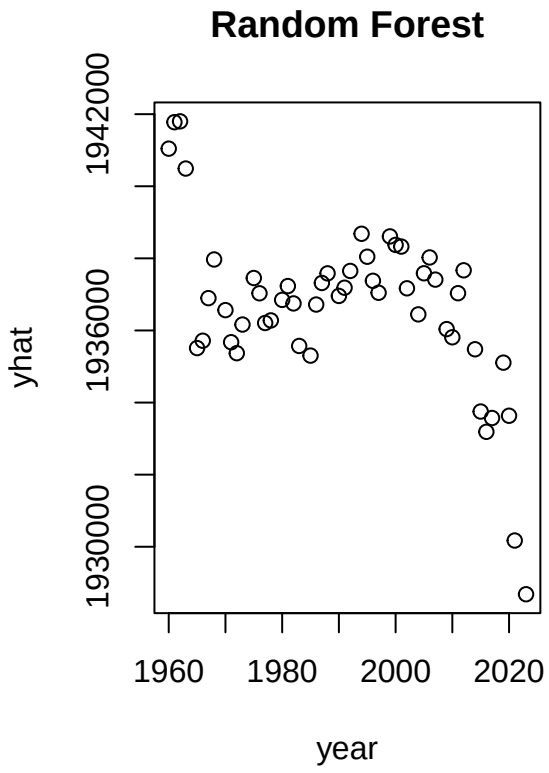



```
train_ctrl <- trainControl(method = "cv", number = 10)
random_forest <- train(streams ~ .,
                       data = training_data,
                       method = "rf",
                       trControl = train_ctrl)

plot_rf <- ggplot(training_data, aes(x = streams, y = predict(random_forest))) +
  geom_point(aes(colour = genre)) +
  geom_abline(intercept = 0, slope = 1, color = "red")
plot_rf
```

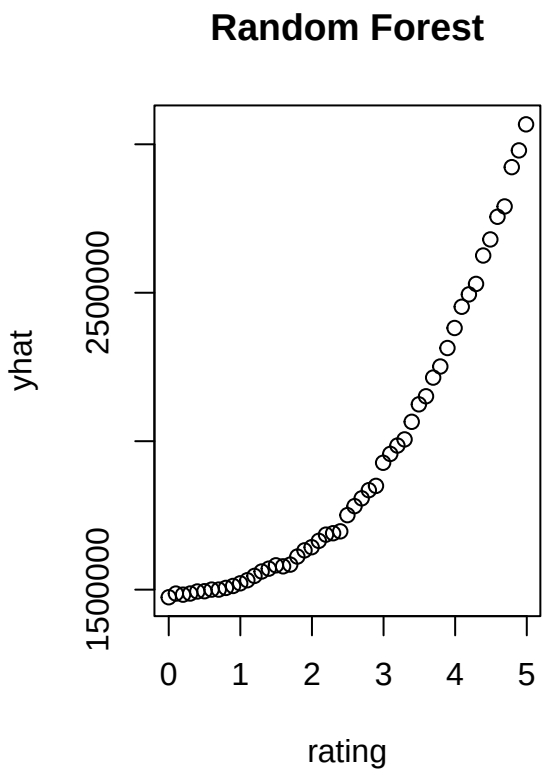


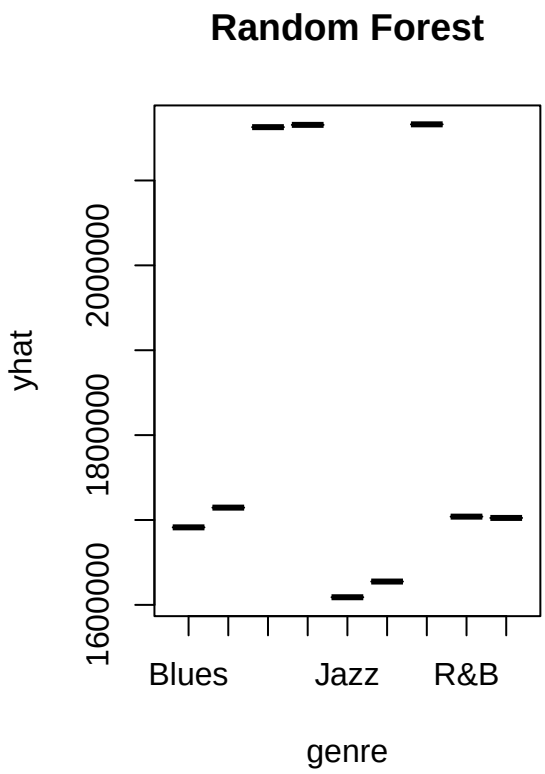

```
par(mfrow = c(1,2))
plot(partial(random_forest, pred.var = "duration_secs"), main = "Random Forest")
plot(partial(random_forest, pred.var = "year"), main = "Random Forest")
```



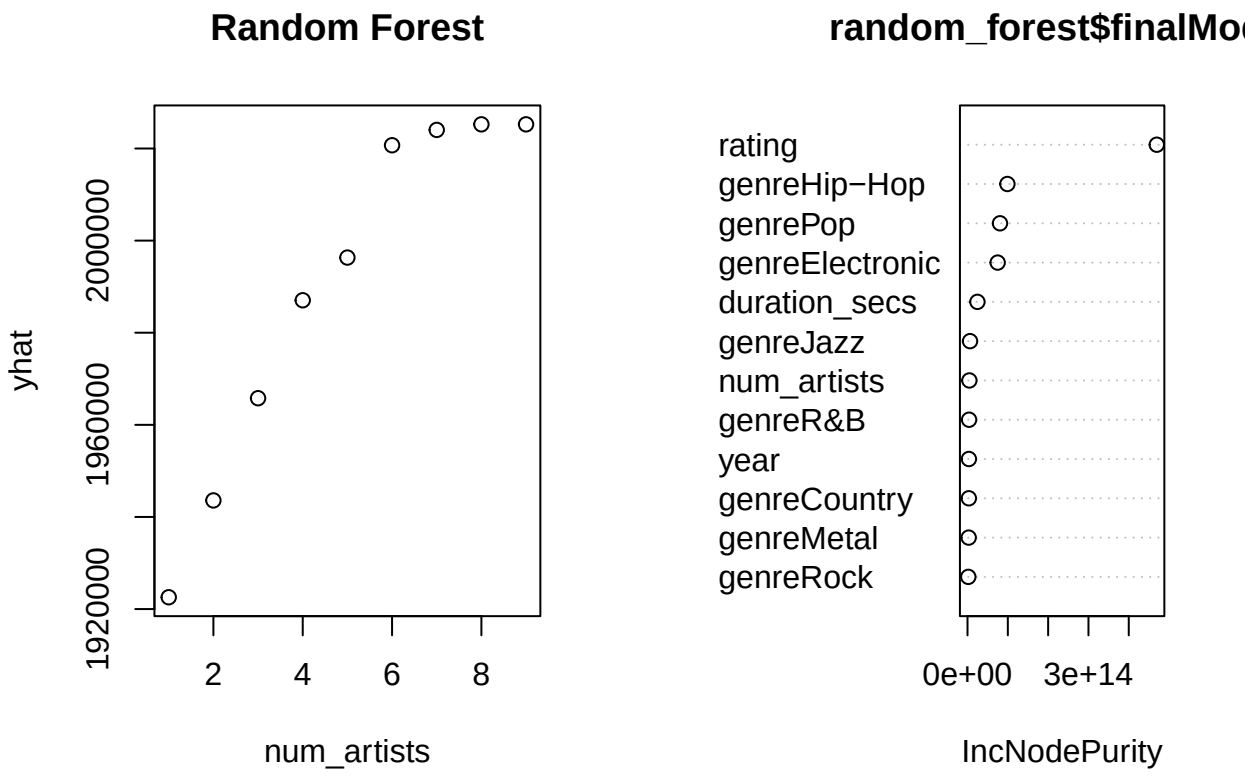




```r
plot(partial(random_forest, pred.var = "rating"), main = "Random Forest")
plot(partial(random_forest, pred.var = "genre"), main = "Random Forest")
```

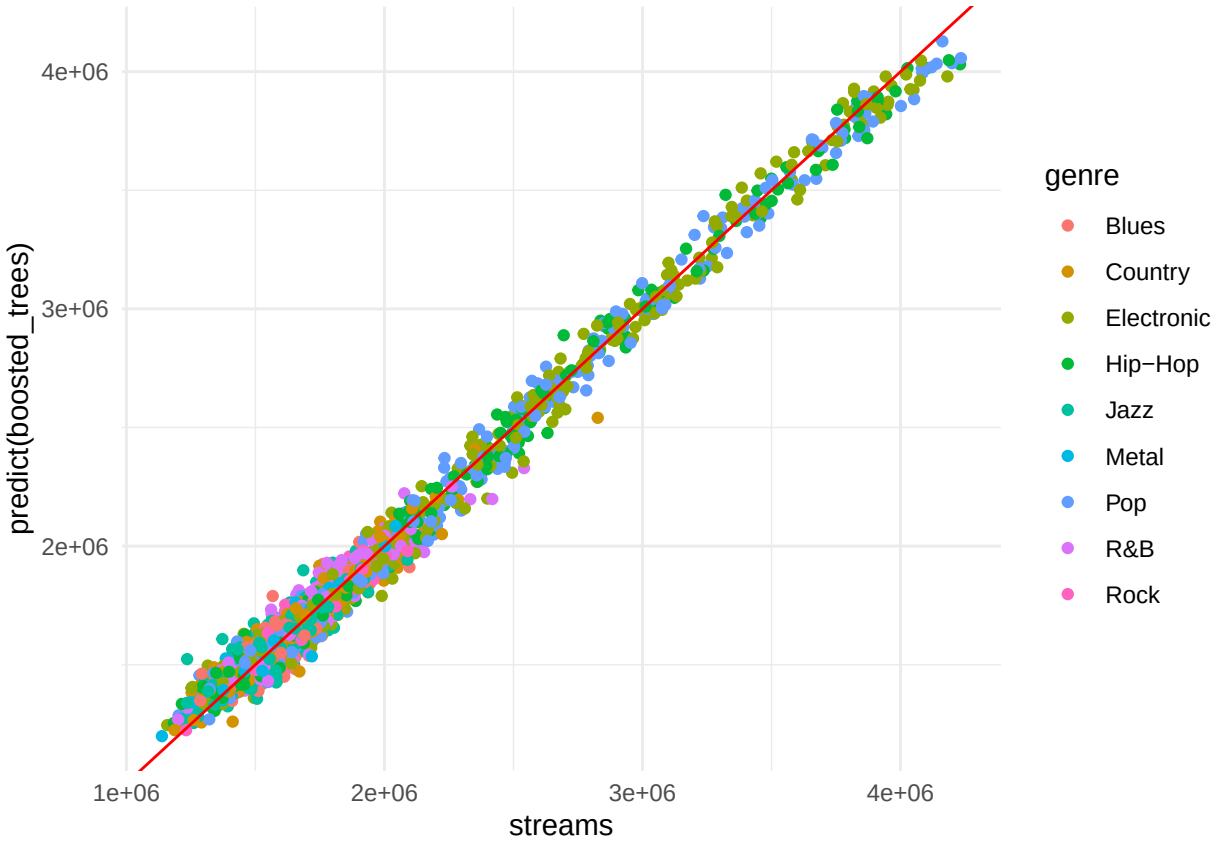


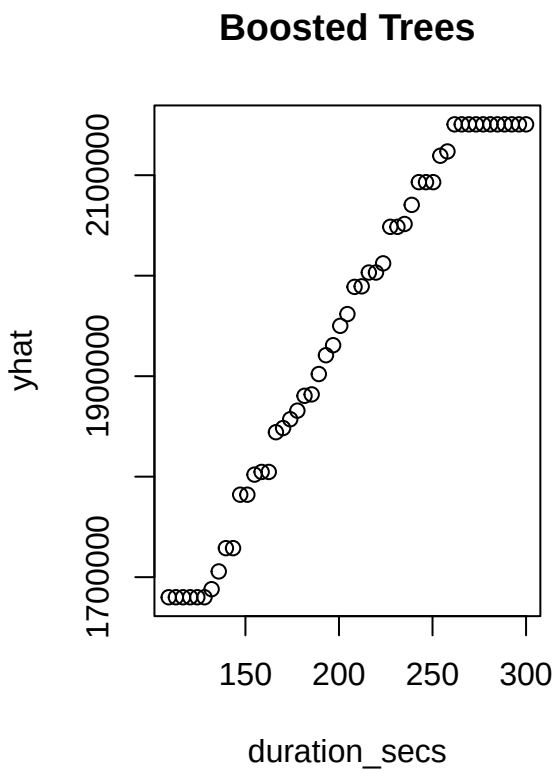

```r
plot(partial(random_forest, pred.var = "num_artists"), main = "Random Forest")
varImpPlot(random_forest$finalModel)
```

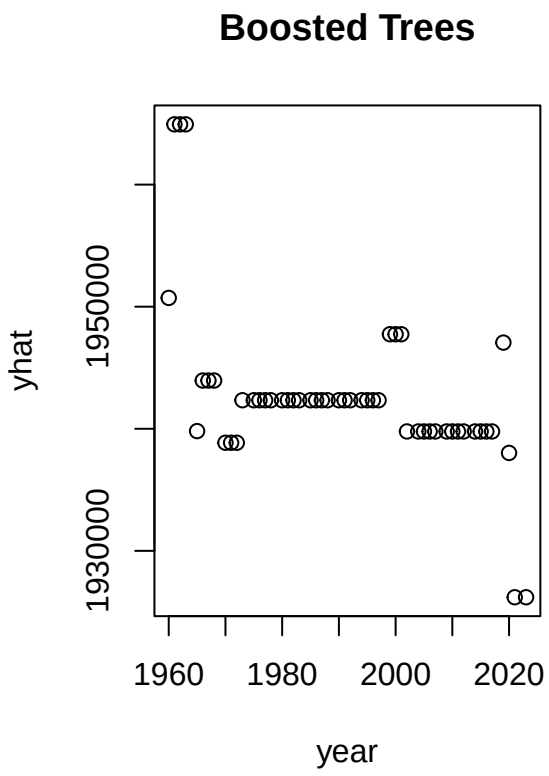



```
boosted_trees <- gbm(
  formula = streams ~ .,
  distribution = "gaussian",
  data = training_data,
  n.trees = 100,
  interaction.depth = 3,
  shrinkage = 0.1,
  cv.folds = 10
)

plot_bt <- ggplot(training_data, aes(x = streams, y = predict(boosted_trees))) +
  geom_point(aes(colour = genre)) +
  geom_abline(intercept = 0, slope = 1, color = "red")
plot_bt
```

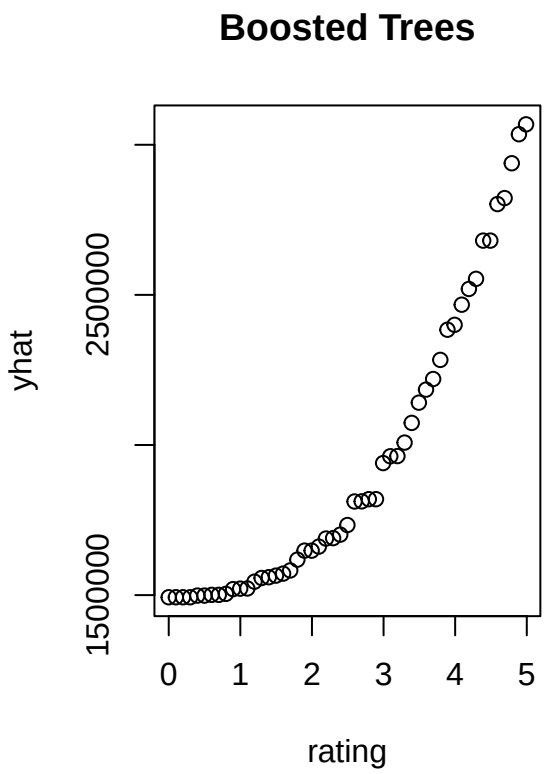


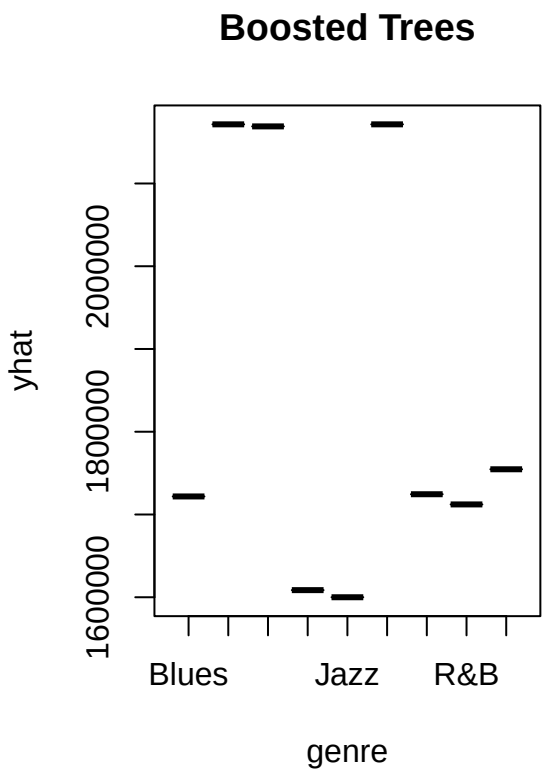

```
plot(partial(boosted_trees, pred.var = "duration_secs", n.trees = 100), main = "Boosted Trees")
plot(partial(boosted_trees, pred.var = "year", n.trees = 100), main = "Boosted Trees")
```



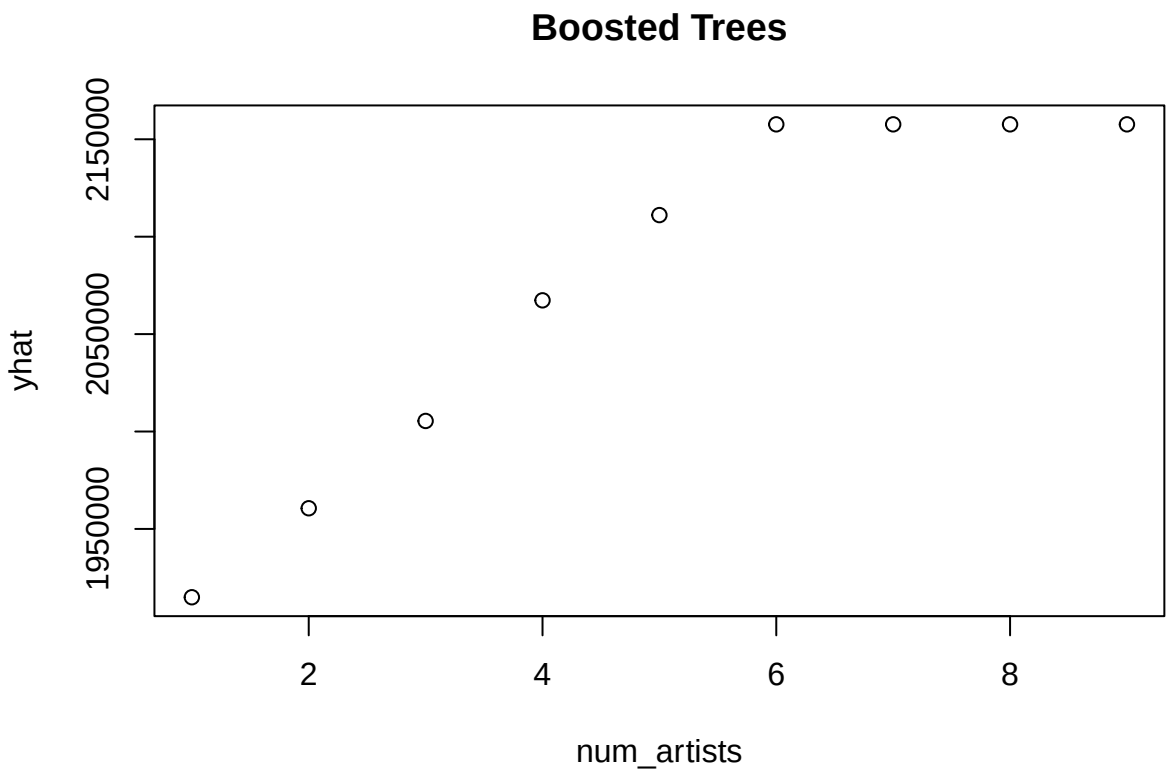




```
plot(partial(boosted_trees, pred.var = "rating", n.trees = 100), main = "Boosted Trees")
plot(partial(boosted_trees, pred.var = "genre", n.trees = 100), main = "Boosted Trees")
```




```
par(mfrow = c(1,1))
plot(partial(boosted_trees, pred.var = "num_artists", n.trees = 100), main = "Boosted Trees")
```




```
genre <- factor(sample(genres, n, prob = c(1,2,4,3,2,4,2,1,1), replace = TRUE))
duration_secs <- round(rnorm(n, mean = 200, sd = 30))
year <- sample(1960:2023, n, replace = TRUE)
rating <- round(runif(n, min = 0, max = 5), digits = 2)
num_artists <- ceiling(rexp(n, rate = 1))
predictors <- data.frame(duration_secs, year, rating, genre, num_artists)
test_data <- predictors |> mutate(streams = CEF(duration_secs, year, rating, genre, num_artists) + rnorm

predictions_st <- predict(pruned_tree, newdata = test_data)
mse_st <- mean((predictions_st - test_data$streams)^2)

predictions_rf <- predict(random_forest, newdata = test_data)
mse_rf <- mean((predictions_rf - test_data$streams)^2)

predictions_bt <- predict(boosted_trees, newdata = test_data)
mse_bt <- mean((predictions_bt - test_data$streams)^2)

mses <- data.frame(Model = c("Single Tree", "Random Forest", "Boosted Trees"), MSE = c(mse_st,mse_rf,mse
mses
```

```
##           Model         MSE
## 1   Single Tree 34613605618
## 2 Random Forest  8650858416
## 3 Boosted Trees  5294622142
```

In the single decision tree model, we see that the predictions are grouped into discrete "buckets", which

clearly does not align with the continuous spread of streams values.

In the random forest model, we see that the linear trend has been captured for 'duration_secs' in the central range of data, with the trend not fully being captured at extreme values. Some of the sinusoidal signal of 'year' has been captured, however contains noise due to this predictor not contributing hugely to the 'streams'. The polynomial relationship between 'rating' and 'streams' and the effect of 'genre' seem to be well captured. The linear relationship between 'num_artists' and 'streams' is well captured for low values, which could be due to the exponential sampling distribution causing a very small number of data points to have high values of 'num_artists'. The model correctly identifies that 'rating' has the highest effect on 'streams', since it has a degree 3 polynomial relationship with 'streams'.

We also notice that the boosted tree model demonstrates a very similar pattern to that of the random forest model.

### ID generalisation

The boosted trees model has the lowest test MSE, and hence is the most accurate.

### OOD generalisation

```
CEF_new_worse <- function(duration_secs, year, rating, genre, num_artists) {
  if_else(genre %in% c("Electronic", "Hip-Hop", "Pop"),
          50000 + 100000*sqrt(duration_secs) + 50000*sin(year) + 20000*(rating) + 5000*num_artists + 20(
          if_else(genre %in% c("Country", "R&B", "Rock"),
                  50000 + 100000*sqrt(duration_secs) + 50000*sin(year) + 20000*(rating^2) + 5000*num_ar
                  50000 + 100000*sqrt(duration_secs) + 50000*sin(year) + 20000*rating + 5000*num_artists
}
test_data_concept_worse <- predictors |> mutate(streams = CEF_new_worse(duration_secs, year, rating, ge
predictions_concept_worse <- predict(boosted_trees, newdata = test_data_concept_worse)
mse_concept_worse <- mean((predictions_concept_worse - test_data_concept_worse$streams)^2)

CEF_new_better <- function(duration_secs, year, rating, genre, num_artists) {
  if_else(genre %in% c("Electronic", "Hip-Hop", "Pop"),
          50000 + 100000*sqrt(duration_secs) + 20000*(rating^3) + 5000*num_artists + 20000*num_artists*
          if_else(genre %in% c("Country", "R&B", "Rock"),
                  50000 + 100000*sqrt(duration_secs) + 20000*(rating^2) + 5000*num_artists + 20000*num_a
                  50000 + 100000*sqrt(duration_secs) + 20000*rating + 5000*num_artists + 20000*num_artis
}

test_data_concept_better <- predictors |> mutate(streams = CEF_new_better(duration_secs, year, rating, g
predictions_concept_better <- predict(boosted_trees, newdata = test_data_concept_worse)
mse_concept_better <- mean((predictions_concept_better - test_data_concept_better$streams)^2)

mses_concept_shift <- data.frame(MSE = c("Original MSE", "Worse MSE - Concept Shift", "Better MSE - Con
mses_concept_shift
```

### Concept shift

```
##                          MSE        Value
## 1               Original MSE   5294622142
## 2  Worse MSE - Concept Shift 401924799524
## 3 Better MSE - Concept Shift   4606199545
```

We notice that by changing the 'rating' term from a polynomial to a linear in one of the branches of the CEF, we achieve a significantly worse accuracy. This makes sense, as from the previous variable importance plot, we see that 'rating' is the most important variable in determining 'streams'.

By removing the 'year' predictor from the CEF, we gain an improvement due it having relatively low importance in determining 'streams', and this simulataneously allows us to build a simpler model that is more resistant to overfitting.

```
rating_new <- round(runif(n, min = 1, max = 5), digits = 2)

predictors <- data.frame(duration_secs, year, rating_new, genre, num_artists)
test_data_cov_worse <- predictors |> mutate(streams = CEF(duration_secs, year, rating_new, genre, num_ar
predictions_cov_worse <- predict(boosted_trees, newdata = test_data_cov_worse)
mse_cov_worse <- mean((predictions_cov_worse - test_data_cov_worse$streams)^2)


duration_secs <- round(rnorm(n, mean = 200, sd = 10))

predictors <- data.frame(duration_secs, year, rating, genre, num_artists)
test_data_cov_better <- predictors |> mutate(streams = CEF(duration_secs, year, rating, genre, num_artis
predictions_cov_better <- predict(boosted_trees, newdata = test_data_cov_better)
mse_cov_better <- mean((predictions_cov_better - test_data_cov_better$streams)^2)

mses_cov_shift <- data.frame(MSE = c("Original MSE", "Worse MSE - Covariate Shift", "Better MSE - Covar
mses_cov_shift
```

**Covariate shift**

```
##                             MSE        Value
## 1                  Original MSE   5294622142
## 2  Worse MSE - Covariate Shift 587060052788
## 3 Better MSE - Covariate Shift   5083363721
```

We already established that the 'rating' predictor has a strong influence on the outcome variable. Since the model was trained on data that included the full range of ratings, it struggles to accurately predict on new data that does not include the lowest values of 'rating' (0 to 1), as it expects to see those data points there and as a result underestimate the 'streams'.

By reducing the variation in the 'num_artists' predictor, we essentially reduce the noise that is present in the data. Hence, the model is able to more accurately predict the outcome variable.