Predictive Analysis Results and Code

Andrew Mashhadi

Since the R script became an out of control mess, I made this so we can see the code, results, and output pretty easily.

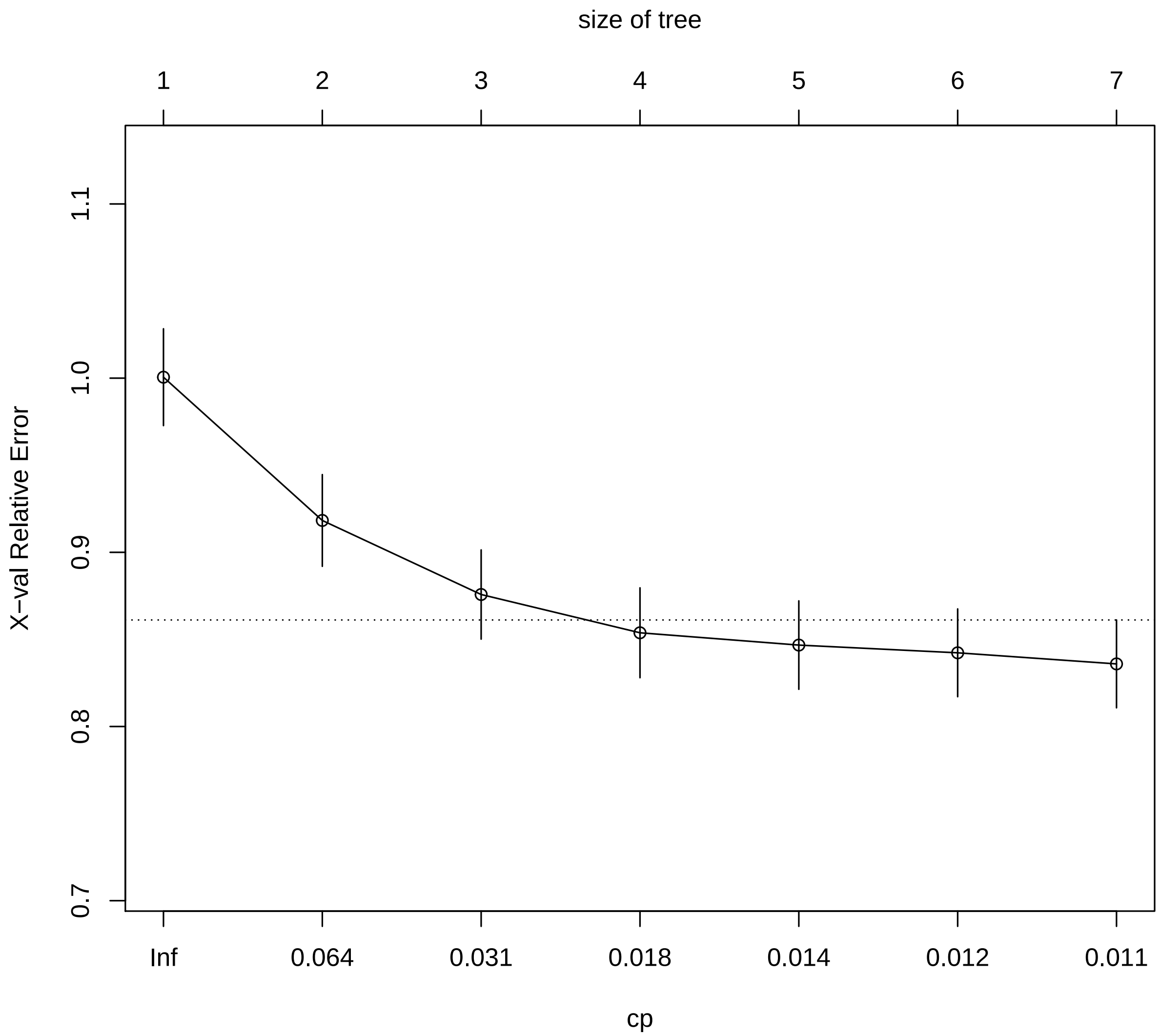
All Libraries Needed

```
library(rpart)
library(partykit)
library(tidyverse)
library(randomForest)
library(tidyverse)
library(tidymodels)
library(vip)
library(ggparty)
```

Predicting imdb-rating with single tree and rf

```
set.seed(777)

imdb_details_extd2 <- read.csv("C:\\Users\\amiro\\Desktop\\Statistics 405\\Week 5\\Final_Project_Brainst
imdb_details_extd2$star_power <- log(imdb_details_extd2$star_power+1)
imdb_details_extd2$wr_pop <- log(imdb_details_extd2$wr_pop+1)

## randomly select genres if more than one
tt<- lapply(imdb_details_extd2$genres, strsplit, ", ")
r_genre <- c()
for (i in 1:length(tt)) {

  if (identical(tt[[i]][[1]], character(0))) {
    name <- "None"
  } else {
    name <- sample(tt[[i]][[1]], 1)
  }
  r_genre <- c(r_genre, name)
}

imdb_details_extd2$genres <- r_genre


## simple single tree
tr <- rpart(imDbRating ~ runtime+genres+rating+dir_pop_fac+co_size+star_power+
              wr_pop+release_period+budget_adj,
```

```
           imdb_details_extd2)

## print plot to help choose cp
plotcp(tr)
```

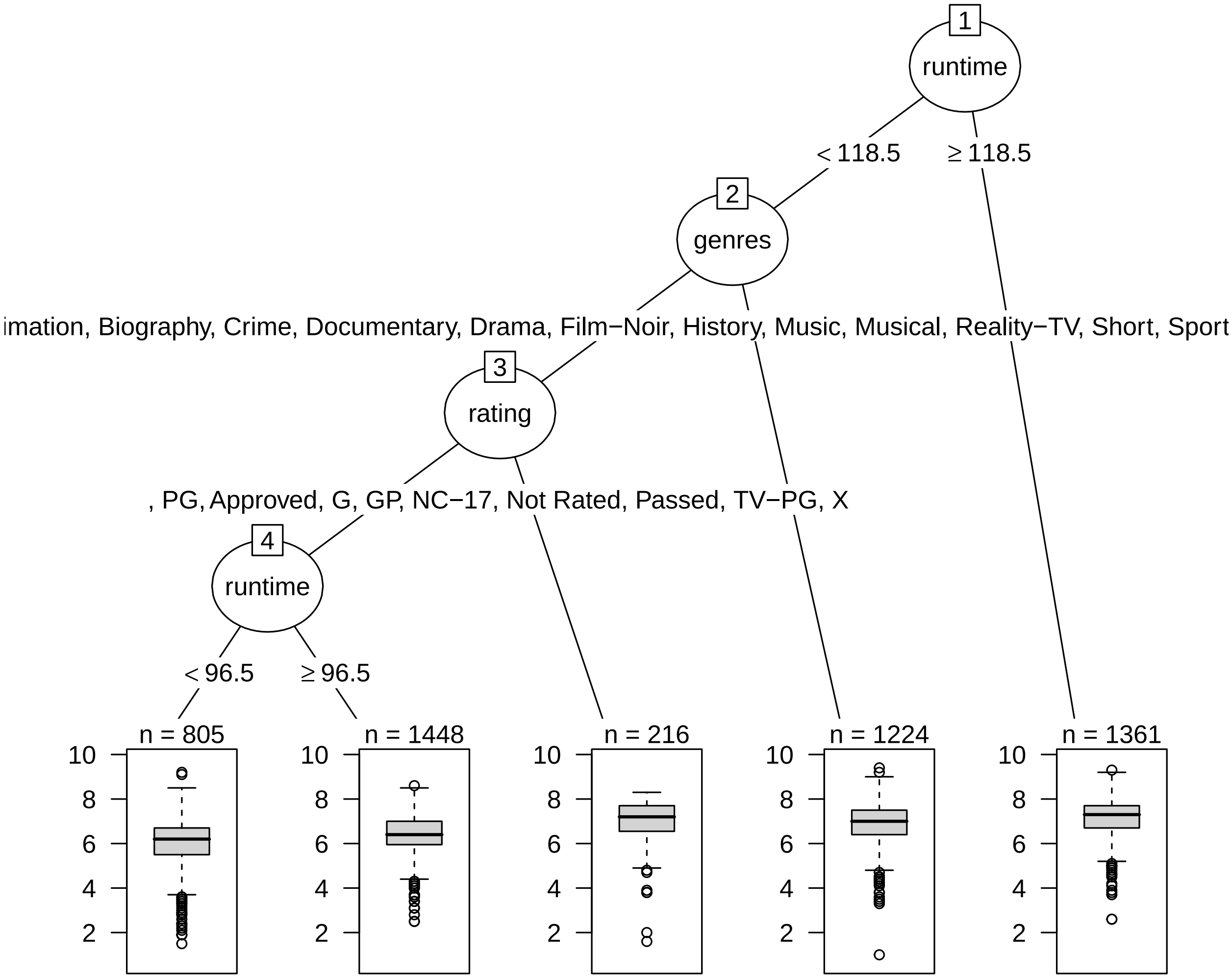



```
## prune the tree
tr_2 <- rpart::prune(tr, cp = 0.014)
print(tr_2)
```

```
## n= 5054
##
## node), split, n, deviance, yval
##       * denotes terminal node
##
##  1) root 5054 4737.3100 6.703720
##    2) runtime< 118.5 3693 3462.9790 6.530301
##      4) genres=Action,Adventure,Comedy,Family,Fantasy,Horror,Mystery,None,Romance,Sci-Fi,Thriller 24
##        8) rating=,PG,PG-13,R,TV-14,TV-MA,Unrated 2253 2013.4540 6.295473
##         16) runtime< 96.5 805  948.4268 6.047329 *
```

```
##          17) runtime>=96.5 1448  987.9022 6.433425 *
##        9) rating=Approved,G,GP,NC-17,Not Rated,Passed,TV-PG,X 216  216.8998 7.000926 *
##      5) genres=Animation,Biography,Crime,Documentary,Drama,Film-Noir,History,Music,Musical,Reality-T
##    3) runtime>=118.5 1361  861.8999 7.174284 *
```

```
## plot the pruned tree
plot(as.party(tr_2), tp_args = list(id = FALSE))
```




```
set.seed(777)
samp <- sample(5090)
rf_errors <- matrix(0, nrow=1, ncol=10)
lm_errors <- matrix(0, nrow=1, ncol=10)
pt_errors <- matrix(0, nrow=1, ncol=10)

for(k in 1:10){
  from <- 1 + (k-1)*509
  to <- 509*k # we will lose the last 8 observations
  test <- na.omit(imdb_details_extd2[samp[from:to],])
  train <- imdb_details_extd2[samp[-(from:to)],]
```

```r
  ## rf
  imdb_rf <- randomForest(imDbRating ~ runtime+genres+rating+dir_pop_fac+
                          co_size+star_power+wr_pop+release_period+budget_adj,
                          data = imdb_details_extd2,
                          mtry = 3,
                          na.action = na.omit)


  # lm
  imdb_lm <- lm(imDbRating ~runtime+rating+dir_pop_fac+
                          co_size+star_power+wr_pop+budget_adj,
                data = imdb_details_extd2,
                na.action = na.omit)

  ## single pruned tree
  tr <- rpart(imDbRating ~runtime+genres+rating+dir_pop_fac+
                          co_size+star_power+wr_pop+release_period+budget_adj,
              data = imdb_details_extd2)
  pt <- rpart::prune(tr, cp = 0.014)

  ## calc errors for this fold
  rf_errors[k] <- mean((test$imDbRating - predict(imdb_rf, test))^2 )
  lm_errors[k] <- mean((test$imDbRating - predict(imdb_lm, test))^2)
  pt_errors[k] <- mean((test$imDbRating - predict(pt, test))^2)
}


## compare K-fold MSEs
mean(rf_errors)
```

```
## [1] 0.68611
```

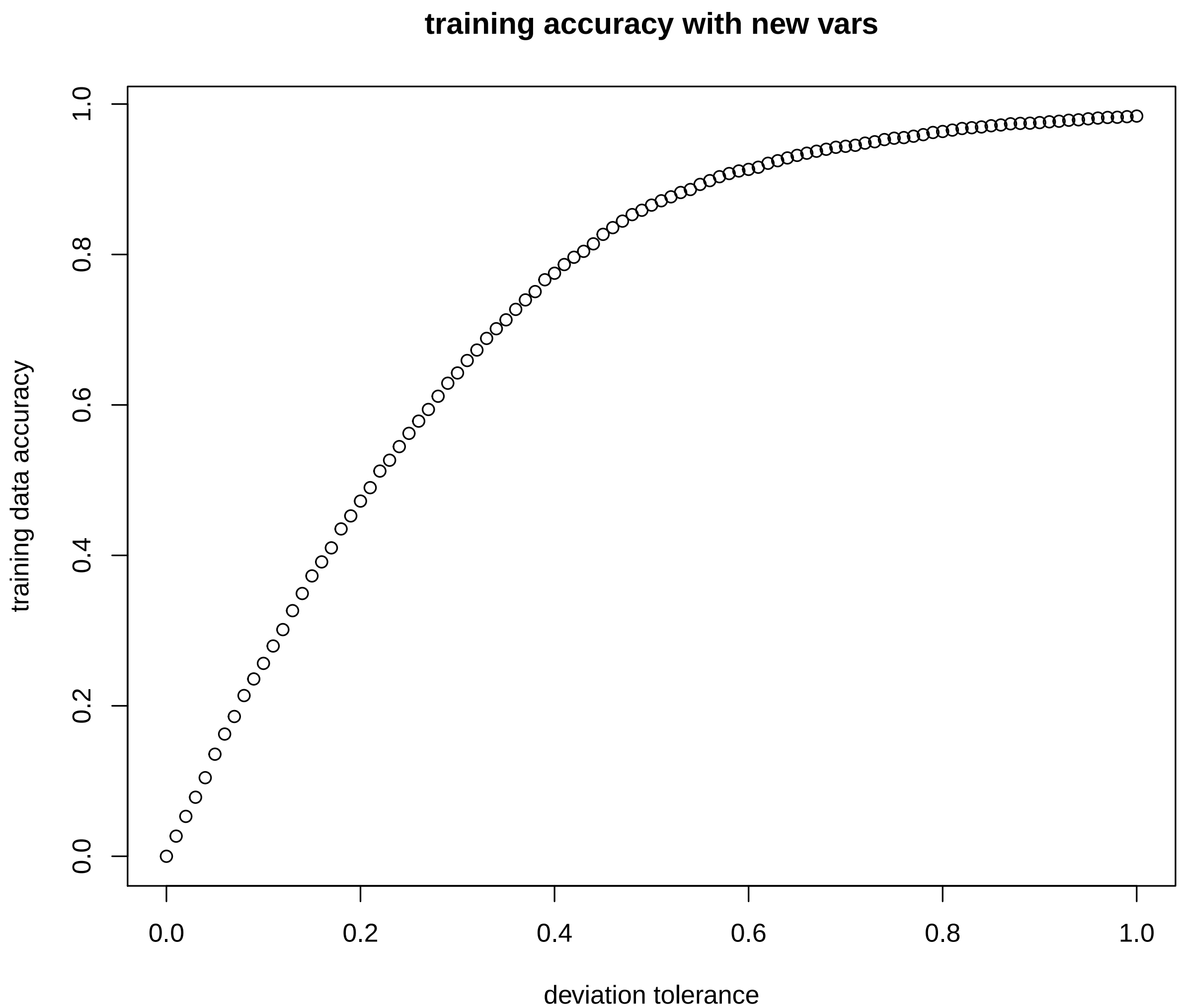
```r
mean(lm_errors)
```

```
## [1] 0.6849048
```

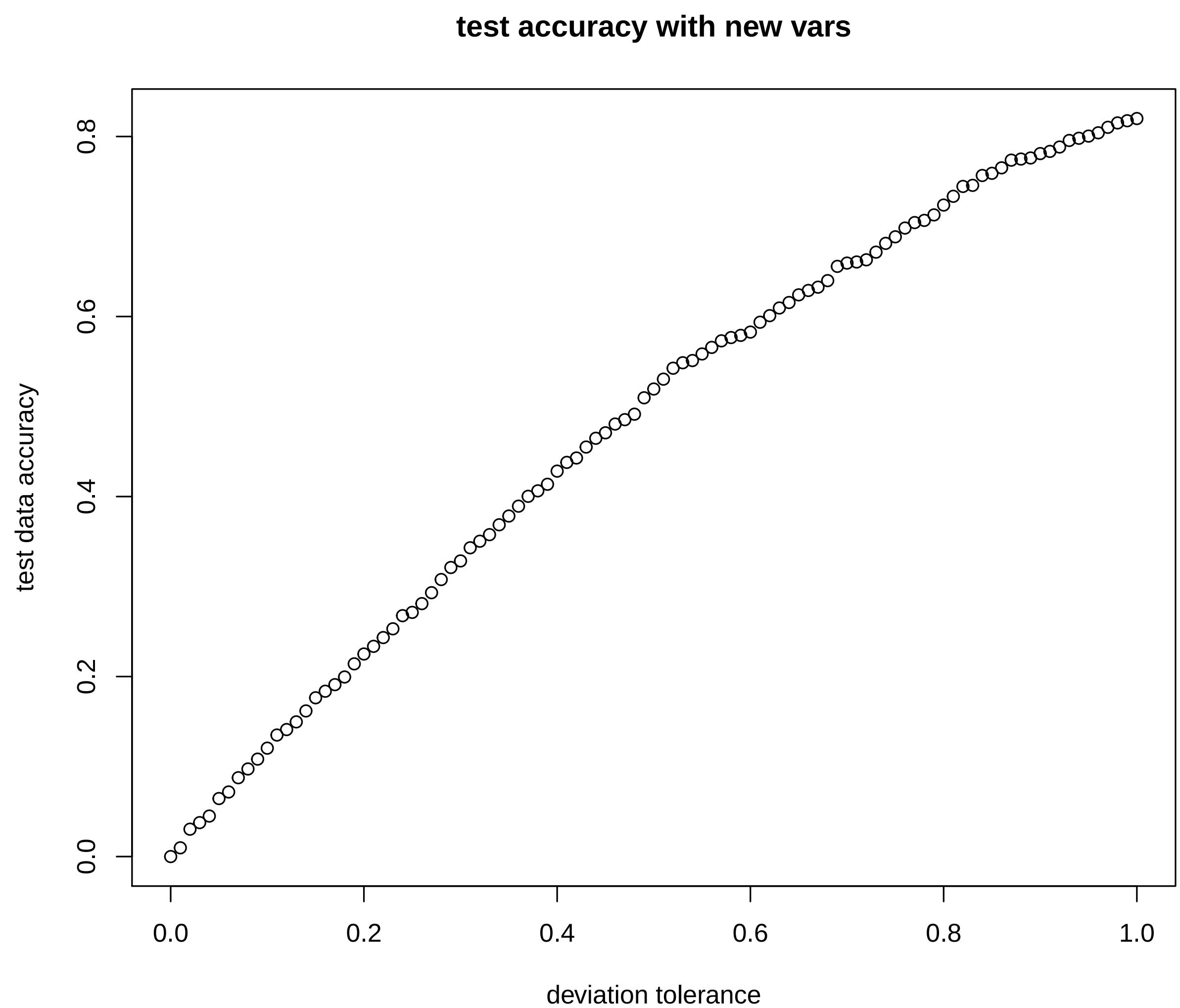
```r
mean(pt_errors)
```

```
## [1] 0.7465697
```

```r
set.seed(777)
## obtain prediction accuracy
imdb_details_no_NA <- na.omit(imdb_details_extd2)
imdb_rf = randomForest(imDbRating ~runtime+genres+rating+dir_pop_fac+
                          co_size+star_power+wr_pop+release_period+budget_adj,
                       data = imdb_details_no_NA,
                       mtry = 3)

## compare the predictions to the data
tr_comp <- data.frame(imDbRating=imdb_details_no_NA$imDbRating, predictedRating=predict(imdb_rf, imdb_de

## approximate training accuracy
devs <- abs(tr_comp$imDbRating - tr_comp$predictedRating)
close_enoughs <- function(x) sum(devs <= x)/ length(devs)
x <- seq(from=0,to=1,by=0.01)
plot(x, sapply(x, close_enoughs), main="training accuracy with new vars", xlab="deviation tolerance", yl
```




```
## approximate test accuracy
samp <- sample(5098)
train <- na.omit(imdb_details_extd2[samp[1:4000],])
test  <- na.omit(imdb_details_extd2[samp[4001:5098],])

imdb_rf = randomForest(imDbRating ~runtime+genres+rating+dir_pop_fac+
                          co_size+star_power+wr_pop+release_period+budget_adj,
                       data = train,
                       mtry = 3)

## compare the predictions to the data
comp <- data.frame(imDbRating=test$imDbRating, predictedRating=predict(imdb_rf, test))

devs <- abs(comp$imDbRating - comp$predictedRating)
close_enoughs <- function(x) sum(devs <= x)/ length(devs)
x <- seq(from=0,to=1,by=0.01)
plot(x, sapply(x, close_enoughs), main="test accuracy with new vars", xlab="deviation tolerance", ylab=
```



Predicting imdb-rating with TidyModels

```
set.seed(777)
trees_df <- filter(imdb_details_extd2, type == "Movie") %>%
  na.omit()

trees_split <- initial_split(trees_df)
trees_train <- training(trees_split)
trees_test <- testing(trees_split)


# build recipe (just instructions)
tree_rec <- recipe(imDbRating ~ runtime+genres+rating+dir_pop_fac+
                          co_size+star_power+wr_pop+release_period+budget_adj,
                   data = trees_train) %>%
            step_other(genres, threshold = 0.03) %>%
```

```r
            step_unknown(genres) %>%
            step_other(rating, threshold = 0.05) %>%
            step_unknown(rating) %>%
            step_dummy(all_nominal(), -all_outcomes())

# prep actually uses the data
tree_prep <- prep(tree_rec)
juiced <- juice(tree_prep)

# run the below to check the step_other results (doesnt work if step_dummy already used)
# juiced %>% count(genres, sort = T)

# report details
summary(tree_rec)
```

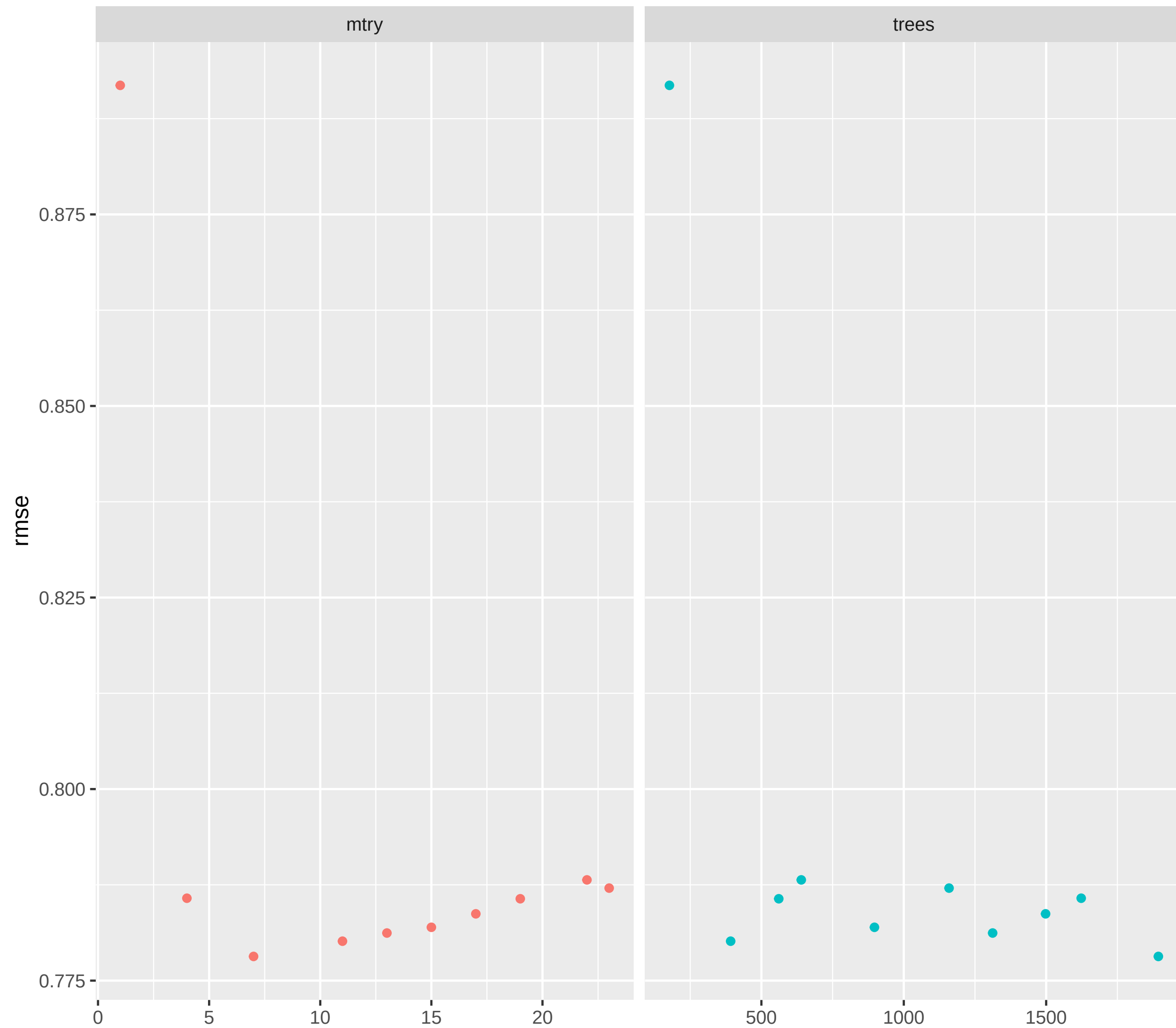
```
## # A tibble: 10 x 4
##    variable       type    role      source
##    <chr>          <chr>   <chr>     <chr>
##  1 runtime        numeric predictor original
##  2 genres         nominal predictor original
##  3 rating         nominal predictor original
##  4 dir_pop_fac    numeric predictor original
##  5 co_size        numeric predictor original
##  6 star_power     numeric predictor original
##  7 wr_pop         numeric predictor original
##  8 release_period nominal predictor original
##  9 budget_adj     numeric predictor original
## 10 imDbRating     numeric outcome   original
```

```r
# build model for tuning
tune_spec <- rand_forest(
  mtry = tune(),
  trees = tune()
) %>%
  set_mode("regression") %>%
  set_engine("ranger")

tune_wf <- workflow() %>%
  add_recipe(tree_rec) %>%
  add_model(tune_spec)

# create a set of cross-validation resamples to use for tuning
trees_folds <- vfold_cv(trees_train)

# choose 10 grid points automatically
tune_res <- tune_grid(
  tune_wf,
  resamples = trees_folds,
  grid = 10
)
```

```
## i Creating pre-processing data to finalize unknown parameter: mtry
```

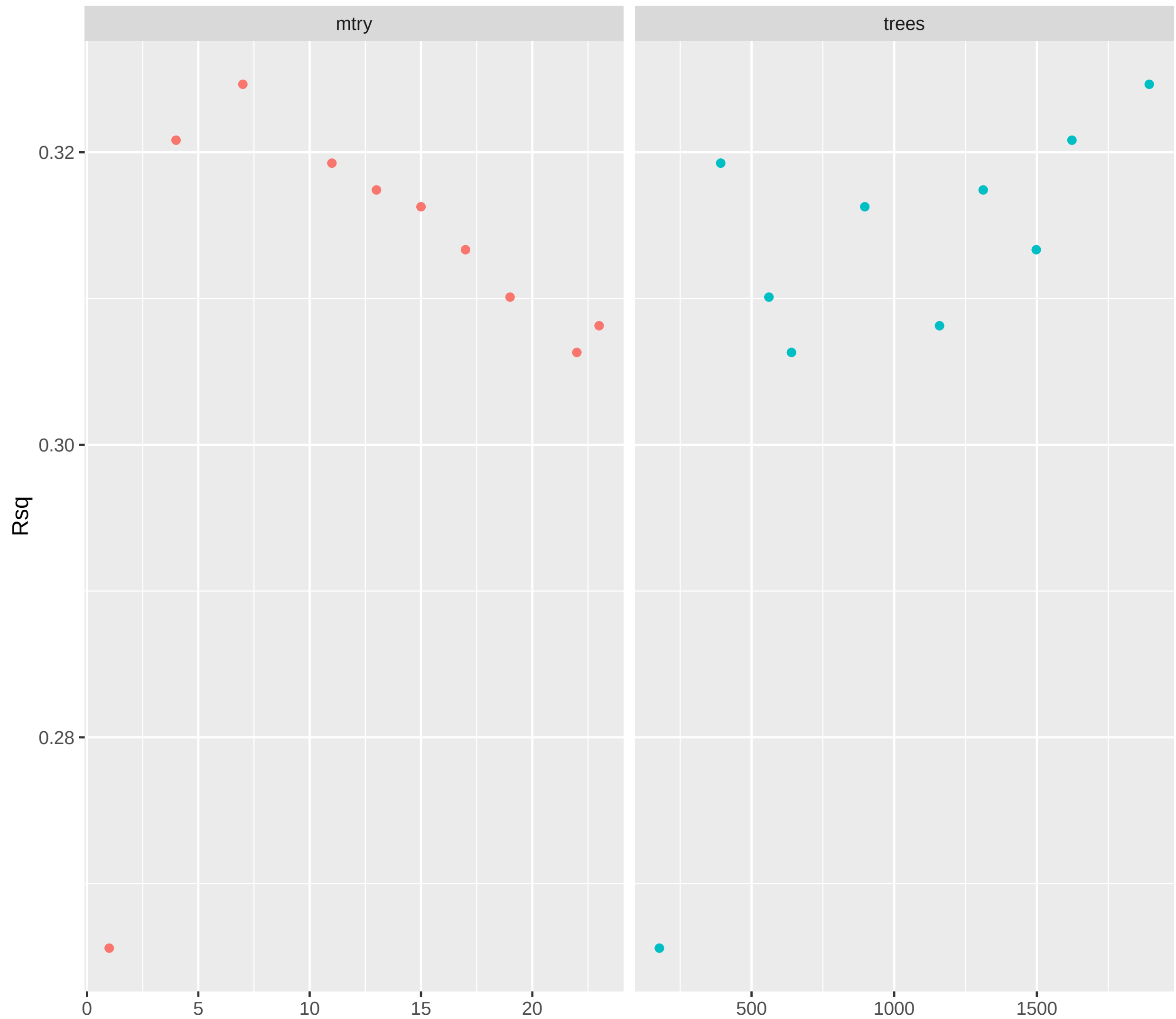
```
## Warning: package 'ranger' was built under R version 4.0.5
```

```r
### rmse plot for tuning mtry and number of trees
tune_res %>%
  collect_metrics() %>%
  filter(.metric == "rmse") %>%
  select(mean, trees, mtry) %>%
  pivot_longer(trees:mtry,
               values_to = "value",
               names_to = "parameter"
  ) %>%
  ggplot(aes(value, mean, color = parameter)) +
  geom_point(show.legend = FALSE) +
  facet_wrap(~parameter, scales = "free_x") +
  labs(x = NULL, y = "rmse")
```



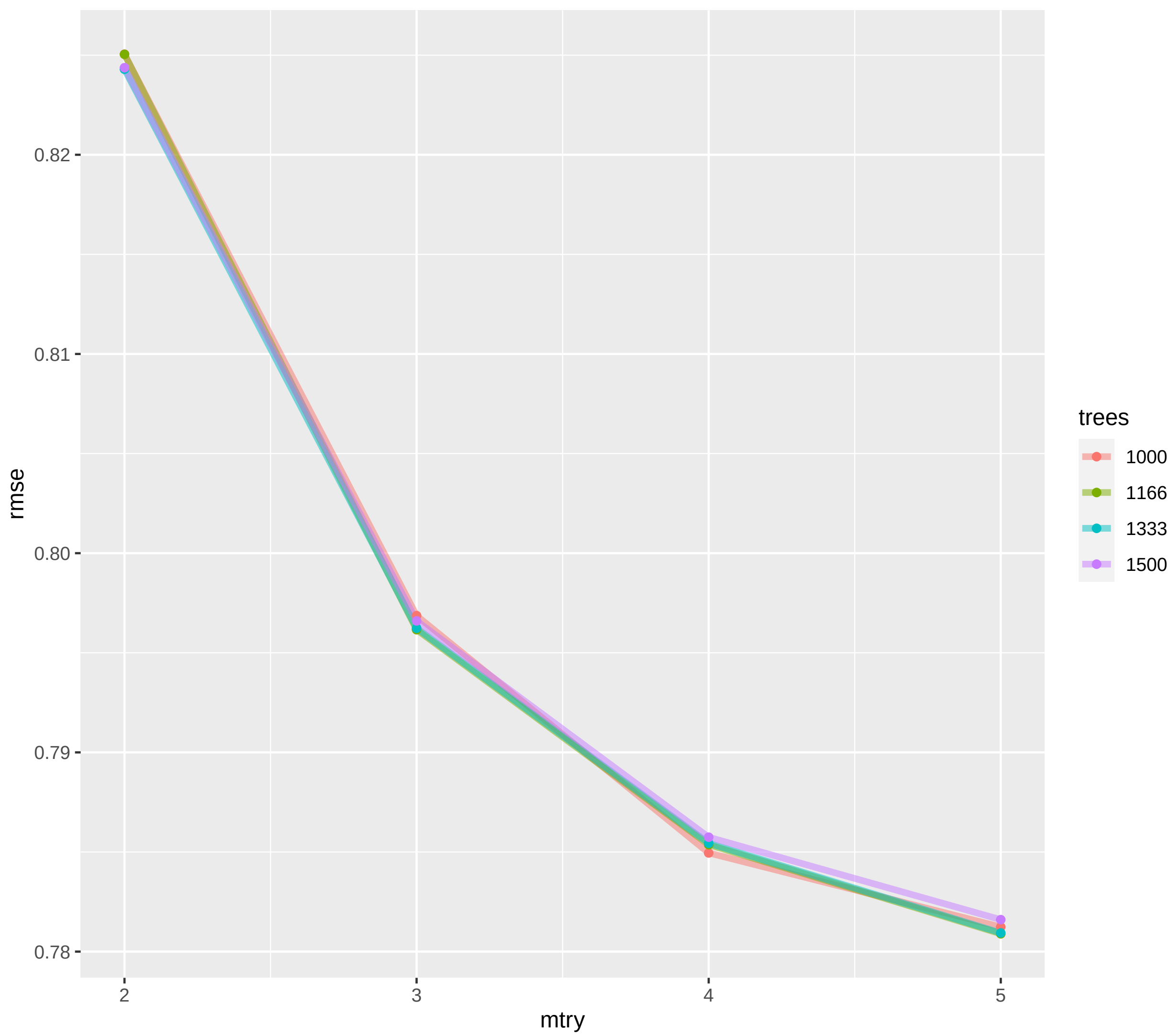

```r
### rsq plot for tuning mtry and number of trees
tune_res %>%
  collect_metrics() %>%
  filter(.metric == "rsq") %>%
  select(mean, trees, mtry) %>%
```

```
  pivot_longer(trees:mtry,
               values_to = "value",
               names_to = "parameter"
  ) %>%
  ggplot(aes(value, mean, color = parameter)) +
  geom_point(show.legend = FALSE) +
  facet_wrap(~parameter, scales = "free_x") +
  labs(x = NULL, y = "Rsq")
```



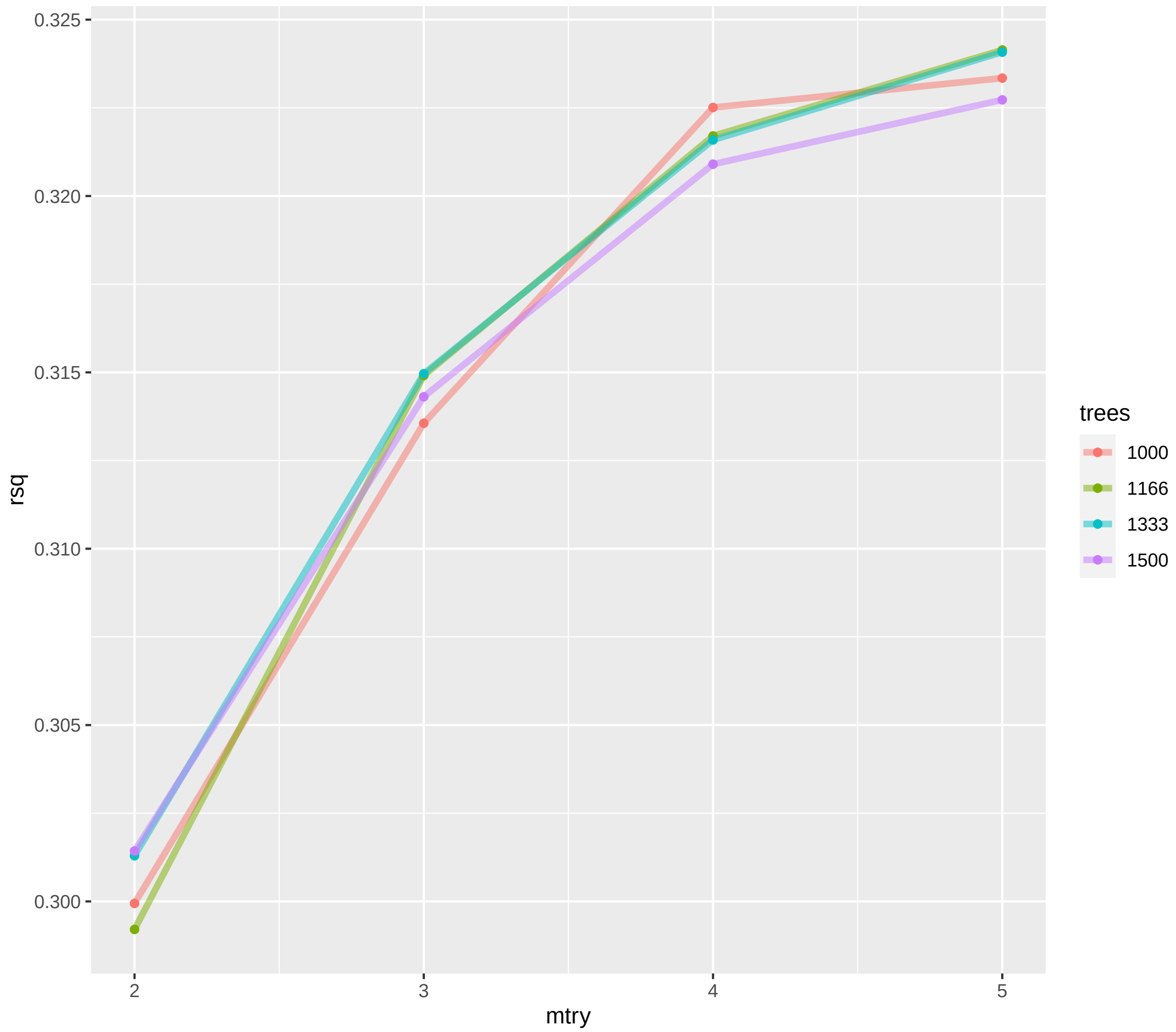

```
## looks like 3 for mtry and 1000-1500 for trees could work best

### taking a closer look now

rf_grid <- grid_regular(
  mtry(range = c(2, 5)),
  trees(range = c(1000, 1500)),
  levels = 4
)
```

```
regular_res <- tune_grid(
  tune_wf,
  resamples = trees_folds,
  grid = rf_grid
)

## rmse plot for tuning mtry and number of trees
regular_res %>%
  collect_metrics() %>%
  filter(.metric == "rmse") %>%
  mutate(trees = factor(trees)) %>%
  ggplot(aes(mtry, mean, color = trees)) +
  geom_line(alpha = 0.5, size = 1.5) +
  geom_point() +
  labs(y = "rmse")
```



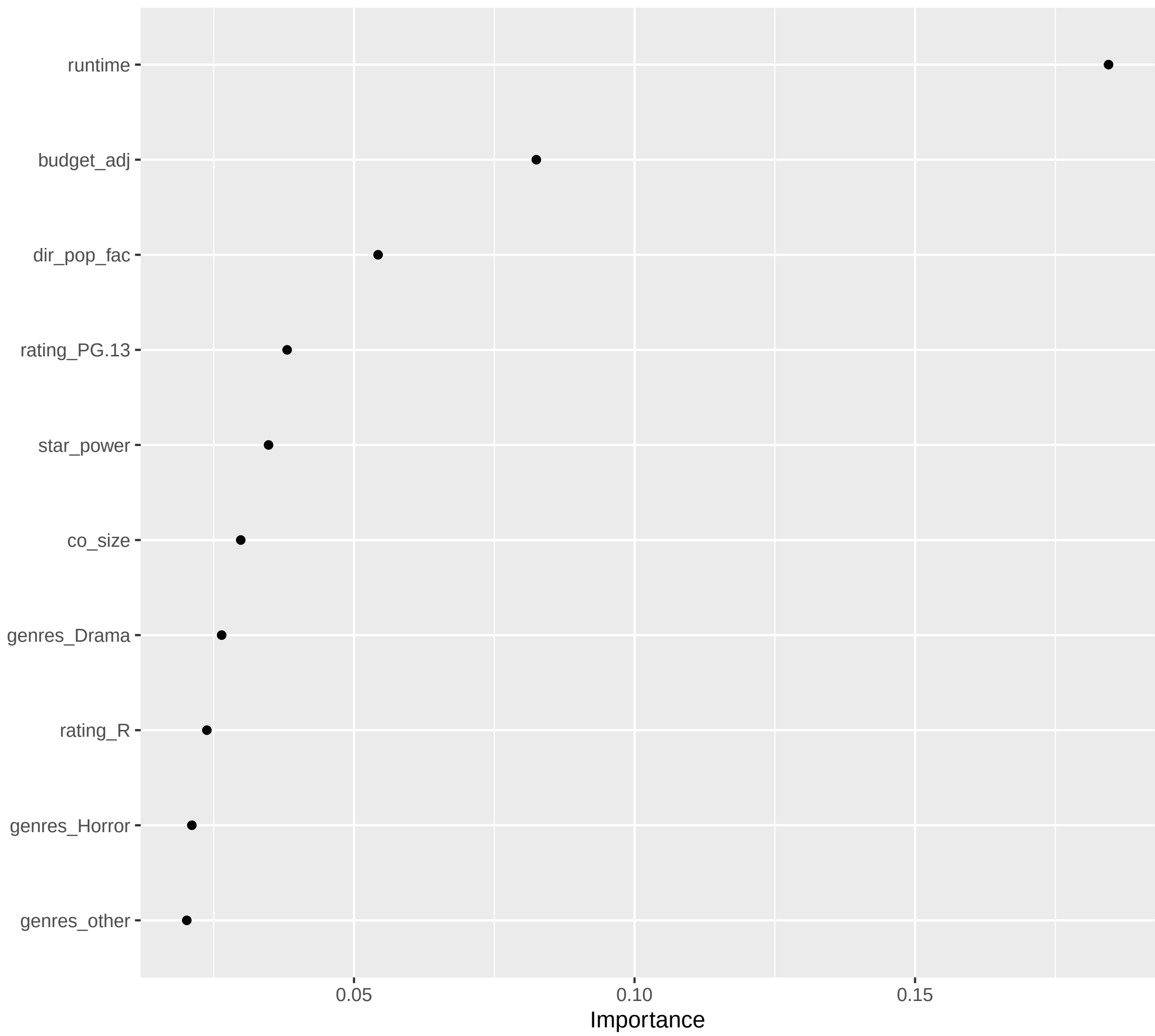

```
## rsq plot for tuning mtry and number of trees
regular_res %>%
  collect_metrics() %>%
```

```
  filter(.metric == "rsq") %>%
  mutate(trees = factor(trees)) %>%
  ggplot(aes(mtry, mean, color = trees)) +
  geom_line(alpha = 0.5, size = 1.5) +
  geom_point() +
  labs(y = "rsq")
```




```
## looks like 3 for mtry and 1166 for trees is optimal

## build model with tuned params
final_rf <- rand_forest(
  mtry = 3,
  trees = 1166,
) %>%
  set_mode("regression") %>%
  set_engine("ranger")

# checking out importance plots
final_rf %>%
```

```
  set_engine("ranger", importance = "permutation") %>%
  fit(imDbRating ~ .,
      data = juice(tree_prep)
  ) %>%
  vip(geom = "point")
```



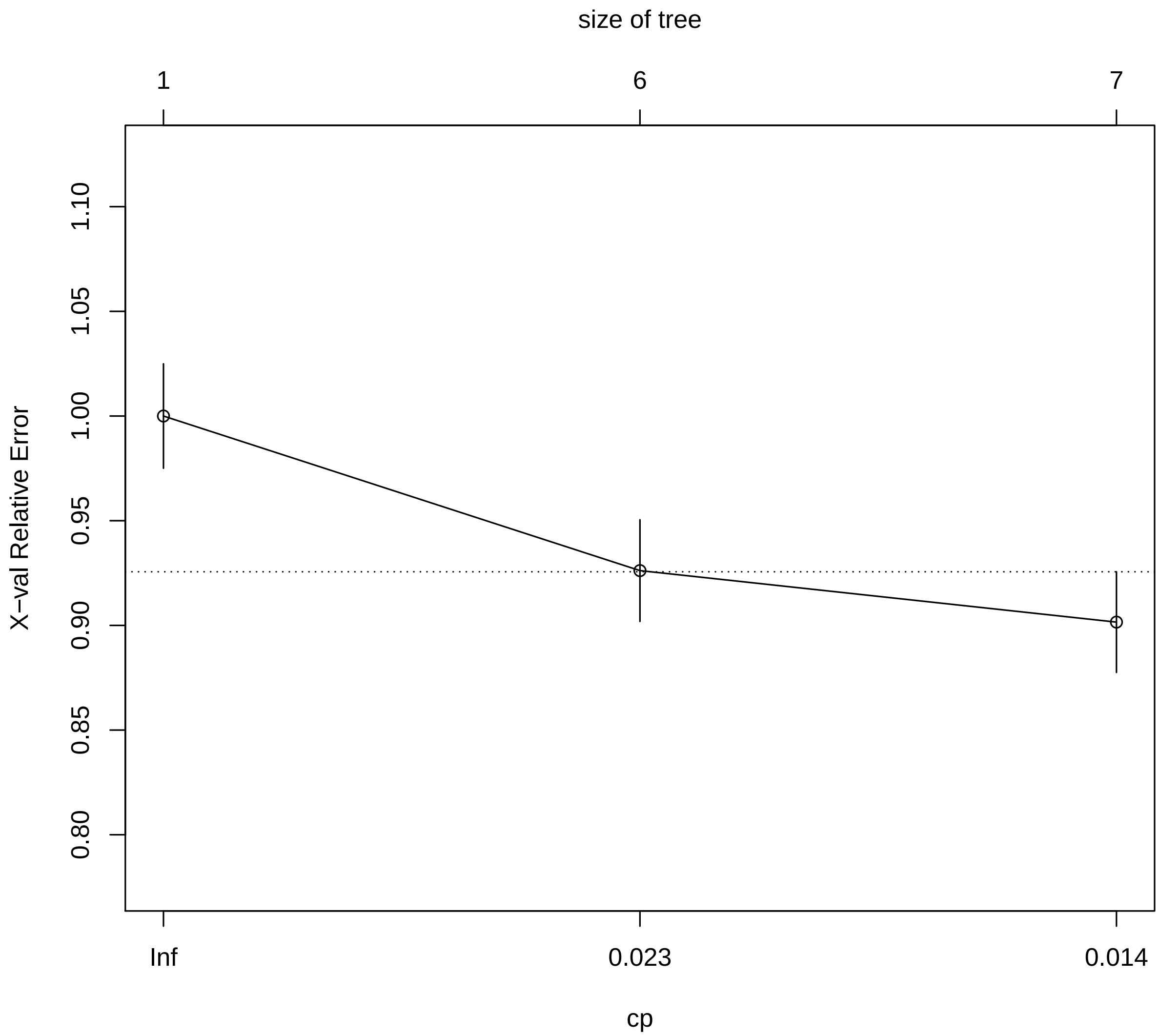

```
### view metrics

final_wf <- workflow() %>%
  add_recipe(tree_rec) %>%
  add_model(final_rf)

final_res <- final_wf %>%
  last_fit(trees_split)

final_res %>%
  collect_metrics()
```

```
## # A tibble: 2 x 4
```

```
##   .metric .estimator .estimate .config
##   <chr>   <chr>          <dbl> <chr>
## 1 rmse    standard       0.816 Preprocessor1_Model1
## 2 rsq     standard       0.326 Preprocessor1_Model1
```

Predicting Oscar-Nomination with single tree and rf

```r
set.seed(777)

imdb_details_extd2 <- read.csv("C:\\Users\\amiro\\Desktop\\Statistics 405\\Week 5\\Final_Project_Brainst
imdb_details_extd2$star_power <- log(imdb_details_extd2$star_power+1)
imdb_details_extd2$wr_pop <- log(imdb_details_extd2$wr_pop+1)

## randomly select genres if more than one
tt<- lapply(imdb_details_extd2$genres, strsplit, ", ")
r_genre <- c()
for (i in 1:length(tt)) {

  if (identical(tt[[i]][[1]], character(0))) {
    name <- "None"
  } else {
    name <- sample(tt[[i]][[1]], 1)
  }
  r_genre <- c(r_genre, name)
}

imdb_details_extd2$genres <- r_genre
imdb_details_extd2$oscar_nom <- as.factor(imdb_details_extd2$oscar_nom)


## simple single tree
tr <- rpart(oscar_nom ~ runtime+genres+rating+dir_pop_fac+co_size+star_power+
              wr_pop+release_period+budget_adj, data=imdb_details_extd2)
## print plot to help choose cp
plotcp(tr)
```

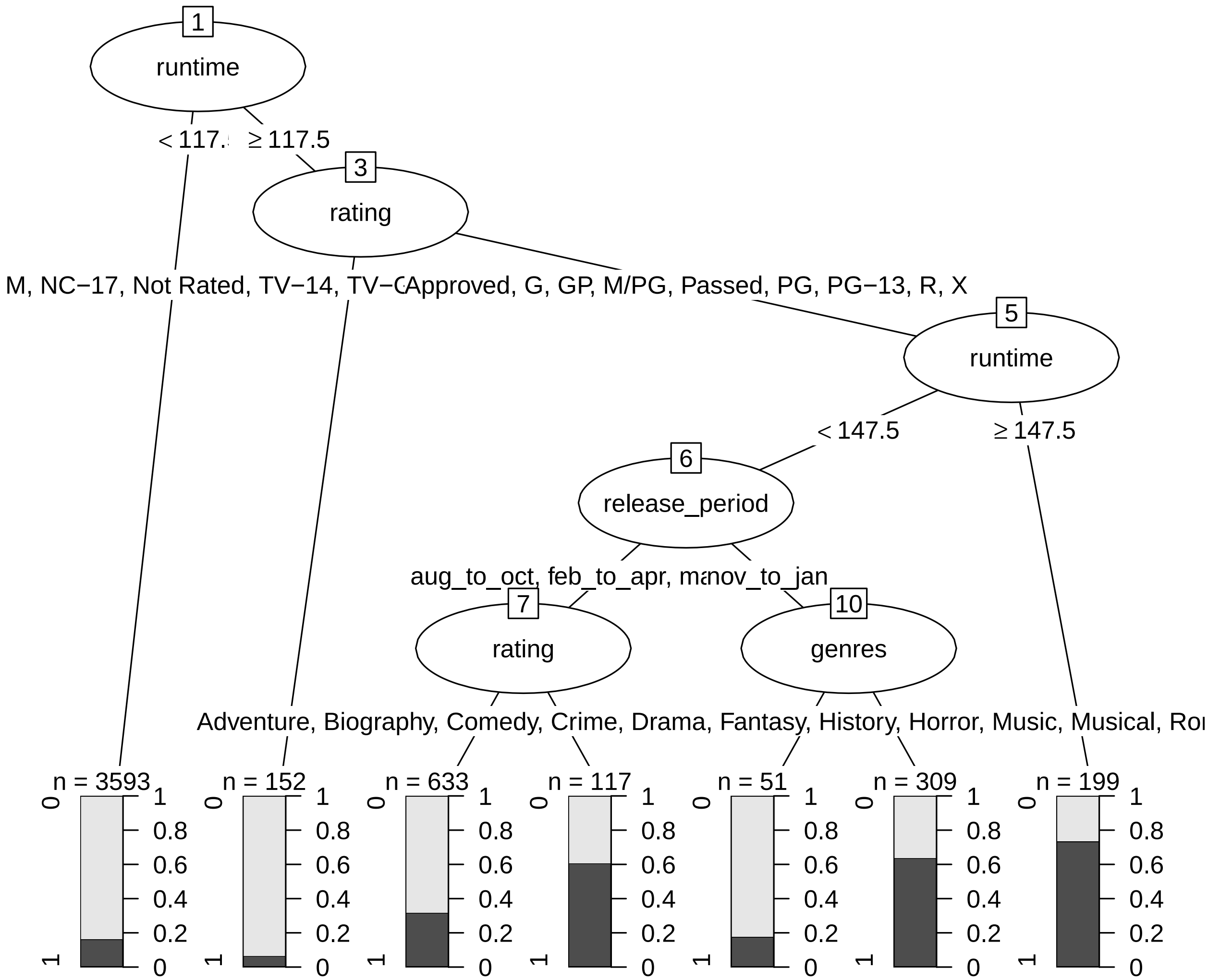



```r
## prune the tree
tr_2 <- rpart::prune(tr, cp = 0.013)
print(tr_2)
```

```
## n= 5054
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 5054 1219 0 (0.75880491 0.24119509)
##    2) runtime< 117.5 3593  585 0 (0.83718341 0.16281659) *
##    3) runtime>=117.5 1461  634 0 (0.56605065 0.43394935)
##      6) rating=,M,NC-17,Not Rated,TV-14,TV-G,TV-MA,TV-PG,Unrated 152   10 0 (0.93421053 0.06578947) *
##      7) rating=Approved,G,GP,M/PG,Passed,PG,PG-13,R,X 1309  624 0 (0.52330023 0.47669977)
##       14) runtime< 147.5 1110  478 0 (0.56936937 0.43063063)
##         28) release_period=aug_to_oct,feb_to_apr,may_to_jul 750  272 0 (0.63733333 0.36266667)
##           56) rating=GP,PG-13,R 633  201 0 (0.68246445 0.31753555) *
##           57) rating=Approved,G,Passed,PG,X 117   46 1 (0.39316239 0.60683761) *
##         29) release_period=nov_to_jan 360  154 1 (0.42777778 0.57222222)
```

```
##           58) genres=Action,Animation,Family,Mystery,Western 51    9 0 (0.82352941 0.17647059) *
##           59) genres=Adventure,Biography,Comedy,Crime,Drama,Fantasy,History,Horror,Music,Musical,Roma
##       15) runtime>=147.5 199   53 1 (0.26633166 0.73366834) *
```

```r
## plot the pruned tree
plot(as.party(tr_2), tp_args = list(id = FALSE))
```




```r
########## COMPARING K-FOLD accuracy FOR MULTIPLE MODELS ##############

samp <- sample(5090)
rf_errors <- matrix(0, nrow=1, ncol=10)
pt_errors <- matrix(0, nrow=1, ncol=10)

for(k in 1:10){
  from <- 1 + (k-1)*509
  to <- 509*k # we will lose the last 8 observations
  test <- na.omit(imdb_details_extd2[samp[from:to],])
  train <- imdb_details_extd2[samp[-(from:to)],]

  ## rf
```

```
  imdb_rf <- randomForest(oscar_nom ~ runtime+genres+rating+dir_pop_fac+co_size+star_power+
                          wr_pop+release_period+budget_adj,
                          data = imdb_details_extd2,
                          mtry = 3,
                          na.action = na.omit)


  ## single pruned tree
  tr <- rpart(oscar_nom ~ runtime+genres+year+rating+dir_pop_fac+co_size+star_power+
              wr_pop+release_period+budget_adj,
              data = imdb_details_extd2)
  pt <- rpart::prune(tr, cp = 0.011)

  ## calc pred accuracy for this fold
  rf_errors[k] <- mean(test$oscar_nom==predict(imdb_rf, test))
  pt_preds <- apply(predict(pt, test), 1, which.max) - 1
  pt_errors[k] <- mean(test$oscar_nom==pt_preds)
}


## compare K-fold prediction accuracy
mean(rf_errors)
```

```
## [1] 0.9170455
```

```
mean(pt_errors)
```

```
## [1] 0.7890319
```

Predicting Oscar Nominations with TidyModels

```
set.seed(777)

trees_df <- filter(imdb_details_extd2, type == "Movie") %>%
  na.omit()

trees_split <- initial_split(trees_df)
trees_train <- training(trees_split)
trees_test <- testing(trees_split)


# build recipe (just instructions)

tree_rec <- recipe(oscar_nom ~  runtime+genres+rating+dir_pop_fac+co_size+star_power+
                   wr_pop+release_period+budget_adj, data = trees_train) %>%
            step_other(genres, threshold = 0.03) %>%
            step_unknown(genres) %>%
            step_other(rating, threshold = 0.05) %>%
            step_unknown(rating) %>%
            step_dummy(all_nominal(), -all_outcomes())

# prep actually uses the data
```

```r
tree_prep <- prep(tree_rec)
juiced <- juice(tree_prep)

# check the step_other results
# juiced %>% count(genres, sort = T)

summary(tree_rec)
```

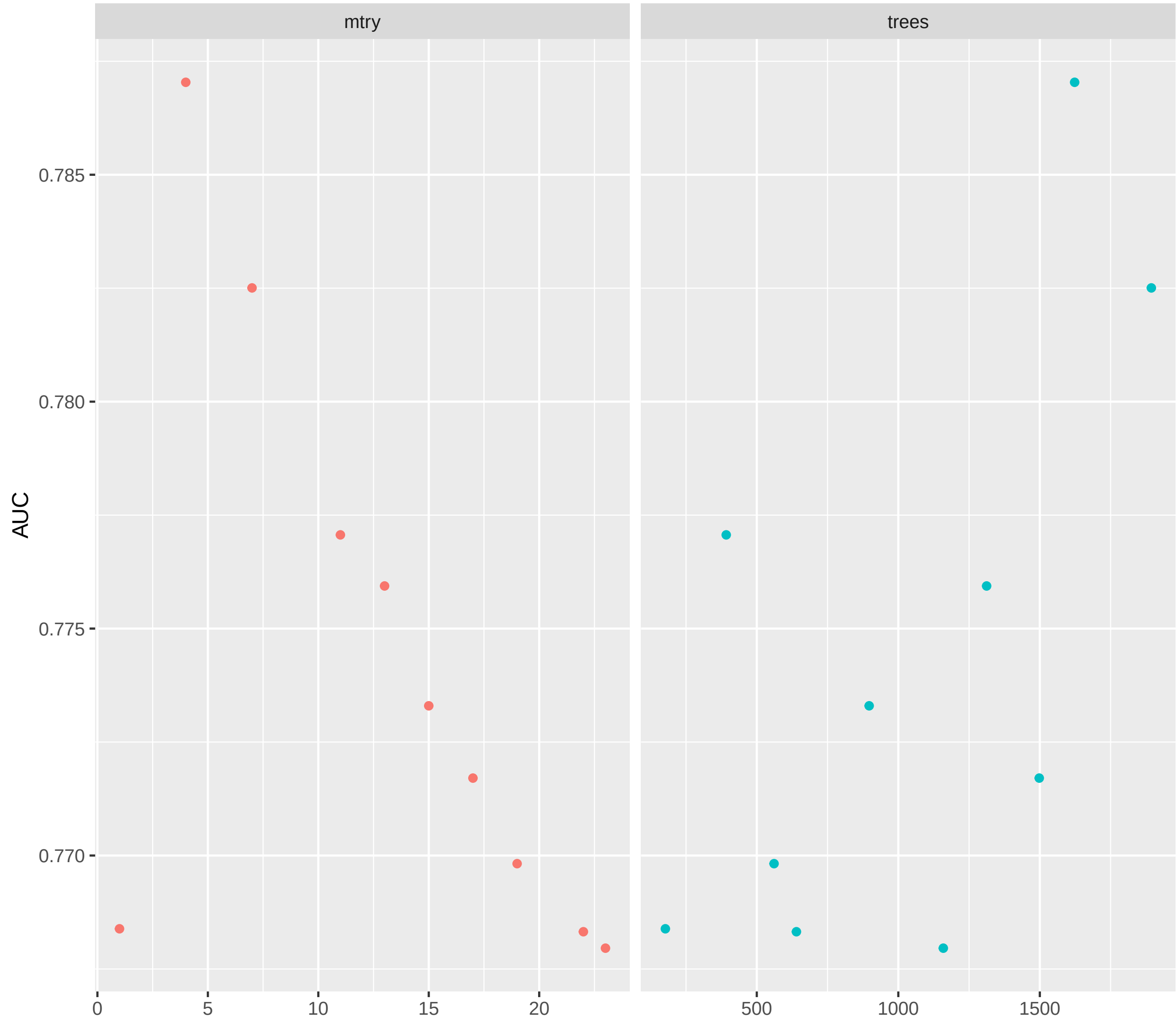
```
## # A tibble: 10 x 4
##    variable       type    role      source
##    <chr>          <chr>   <chr>     <chr>
##  1 runtime        numeric predictor original
##  2 genres         nominal predictor original
##  3 rating         nominal predictor original
##  4 dir_pop_fac    numeric predictor original
##  5 co_size        numeric predictor original
##  6 star_power     numeric predictor original
##  7 wr_pop         numeric predictor original
##  8 release_period nominal predictor original
##  9 budget_adj     numeric predictor original
## 10 oscar_nom      nominal outcome   original
```

```r
# build model
tune_spec <- rand_forest(
  mtry = tune(),
  trees = tune()
) %>%
  set_mode("classification") %>%
  set_engine("ranger")

tune_wf <- workflow() %>%
  add_recipe(tree_rec) %>%
  add_model(tune_spec)

# create a set of cross-validation resamples to use for tuning
trees_folds <- vfold_cv(trees_train)

#doParallel::registerDoParallel() #try me

# choose 10 grid points automatically
tune_res <- tune_grid(
  tune_wf,
  resamples = trees_folds,
  grid = 10
)
```

```
## i Creating pre-processing data to finalize unknown parameter: mtry
```

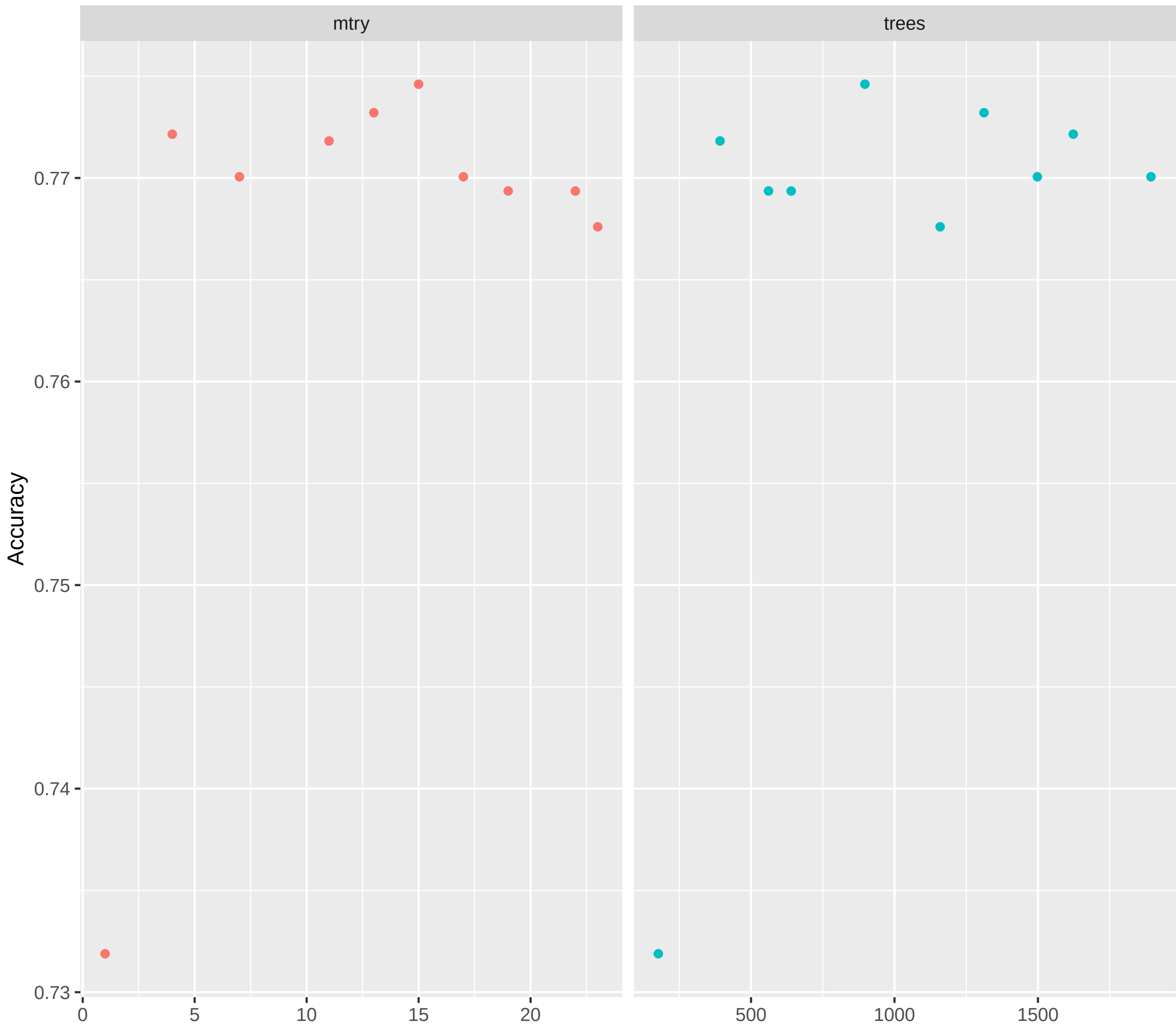
```r
### roc_auc plot for tuning mtry and number of trees
tune_res %>%
  collect_metrics() %>%
  filter(.metric == "roc_auc") %>%
  dplyr::select(mean, trees, mtry) %>%
  pivot_longer(trees:mtry,
               values_to = "value",
```

```
                names_to = "parameter"
  ) %>%
  ggplot(aes(value, mean, color = parameter)) +
  geom_point(show.legend = FALSE) +
  facet_wrap(~parameter, scales = "free_x") +
  labs(x = NULL, y = "AUC")
```



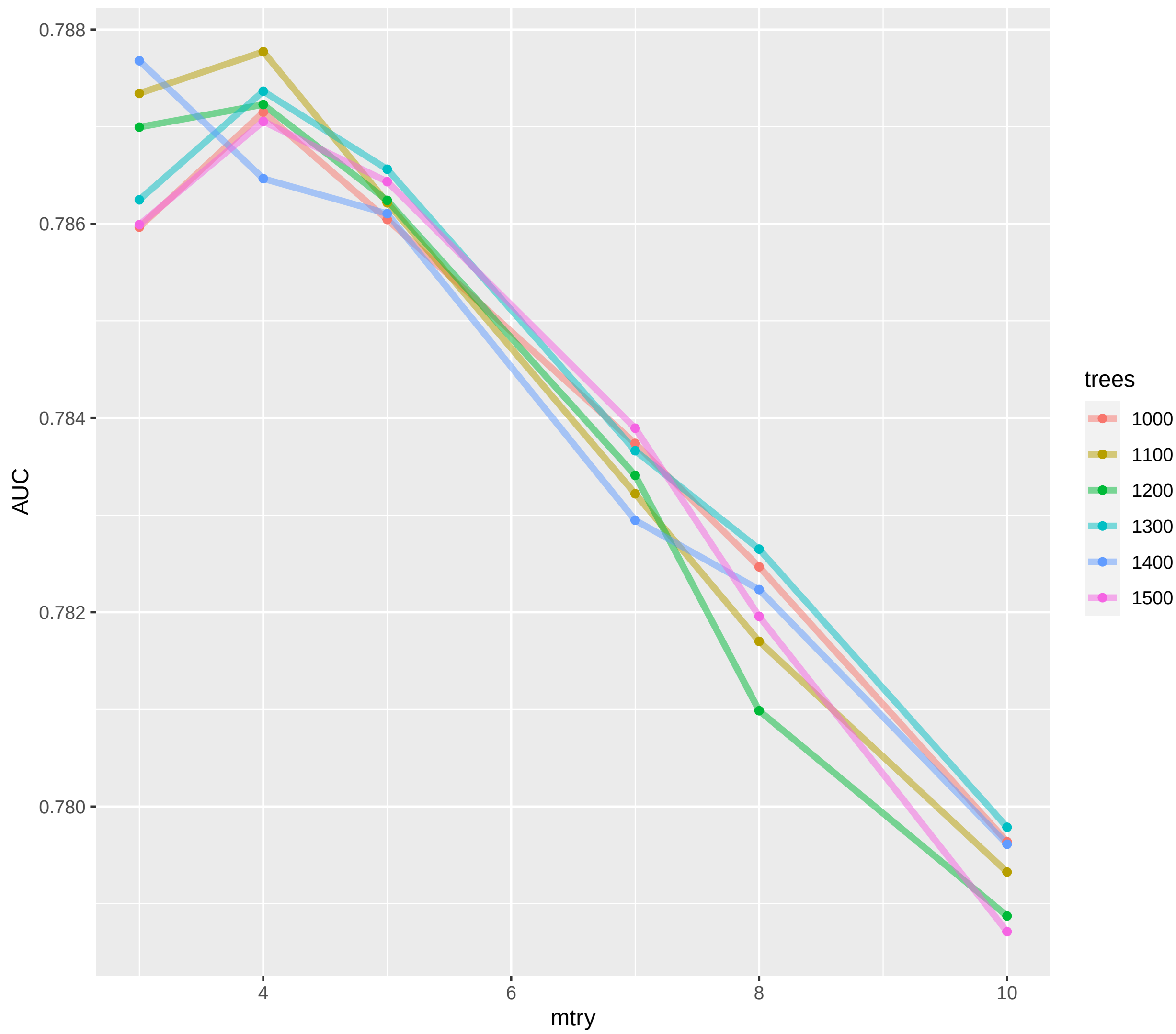

```
### accuracy plot for tuning mtry and number of trees
tune_res %>%
  collect_metrics() %>%
  filter(.metric == "accuracy") %>%
  dplyr::select(mean, trees, mtry) %>%
  pivot_longer(trees:mtry,
                values_to = "value",
                names_to = "parameter"
  ) %>%
  ggplot(aes(value, mean, color = parameter)) +
  geom_point(show.legend = FALSE) +
  facet_wrap(~parameter, scales = "free_x") +
```

```
  labs(x = NULL, y = "Accuracy")
```

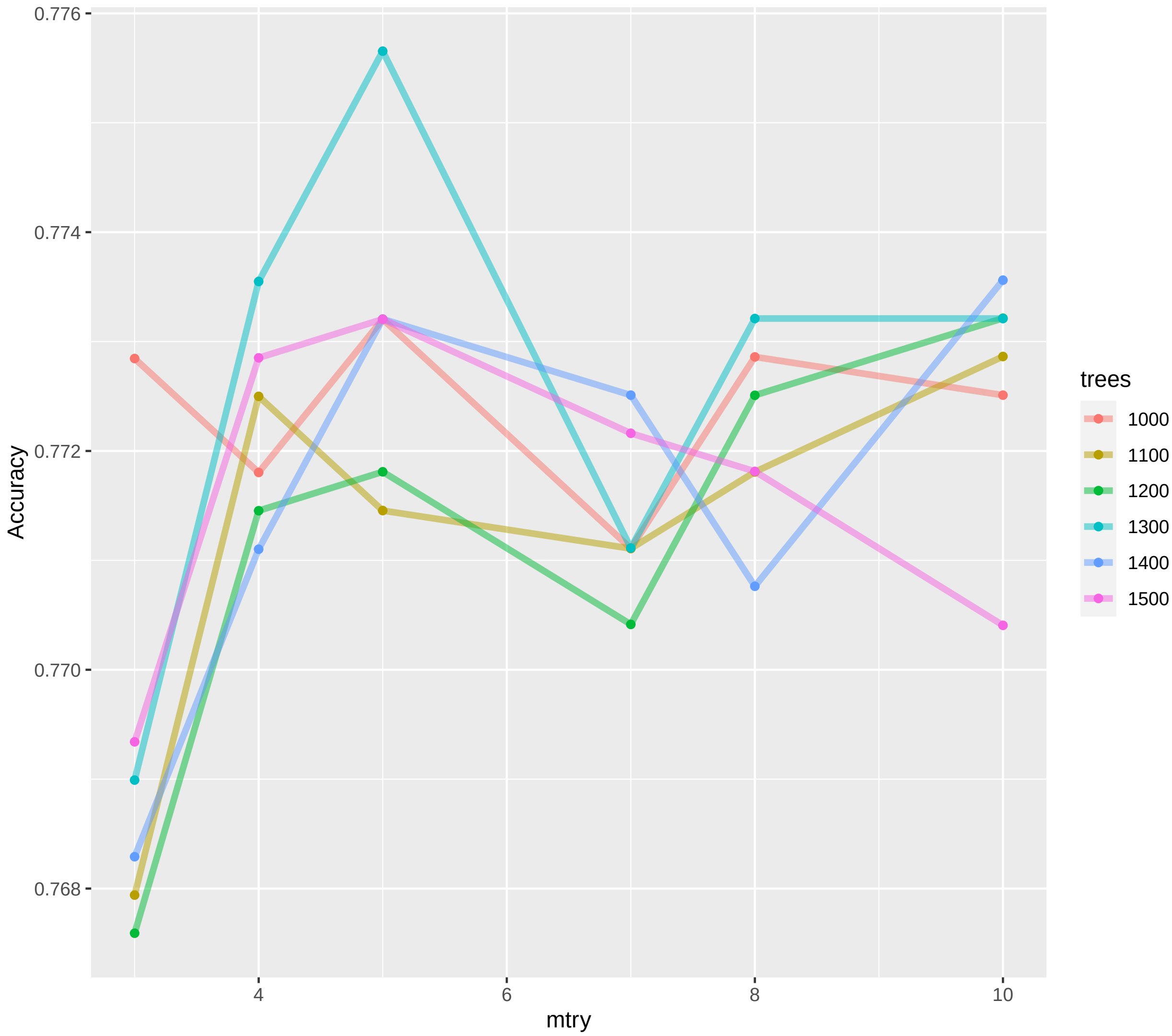



```
## looks like 3 for mtry and 1000-1500 for trees could work best

### taking a closer look now

rf_grid <- grid_regular(
  mtry(range = c(3, 10)),
  trees(range = c(1000, 1500)),
  levels = 6
)

regular_res <- tune_grid(
  tune_wf,
  resamples = trees_folds,
  grid = rf_grid
)

## AUC plot for tuning mtry and number of trees
```

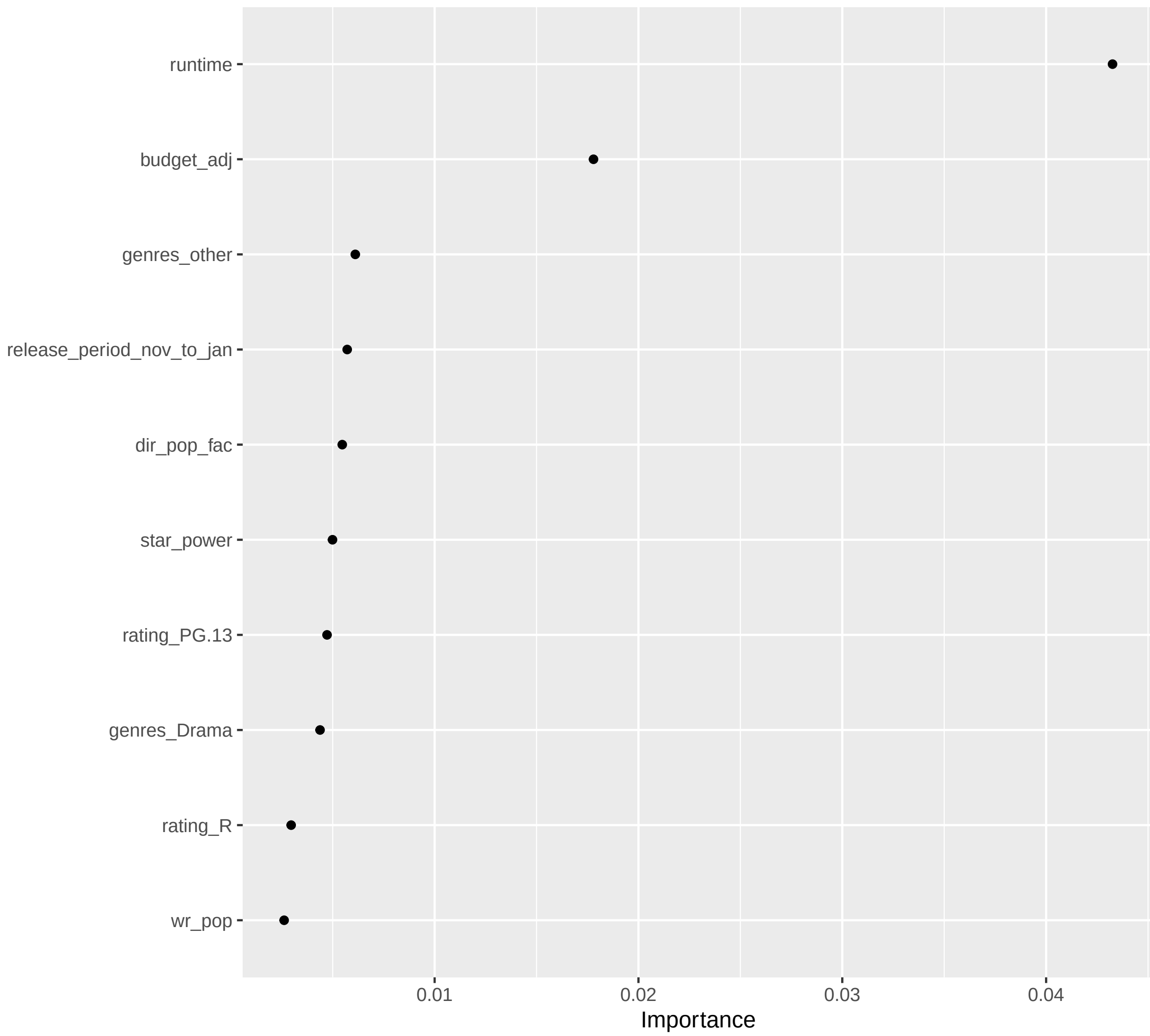
```r
regular_res %>%
  collect_metrics() %>%
  filter(.metric == "roc_auc") %>%
  mutate(trees = factor(trees)) %>%
  ggplot(aes(mtry, mean, color = trees)) +
  geom_line(alpha = 0.5, size = 1.5) +
  geom_point() +
  labs(y = "AUC")
```




```r
## accuracy plot for tuning mtry and number of trees
regular_res %>%
  collect_metrics() %>%
  filter(.metric == "accuracy") %>%
  mutate(trees = factor(trees)) %>%
  ggplot(aes(mtry, mean, color = trees)) +
  geom_line(alpha = 0.5, size = 1.5) +
  geom_point() +
  labs(y = "Accuracy")
```




```r
## looks like 7 for mtry and 1250 for trees is optimal

## build model with tuned params
final_rf <- rand_forest(
  mtry = 7,
  trees = 1250,
) %>%
  set_mode("classification") %>%
  set_engine("ranger")

# checking out importance plots (measure of how much worse predictions get w/o the var)
final_rf %>%
  set_engine("ranger", importance = "permutation") %>%
  fit(oscar_nom ~ .,
      data = juice(tree_prep)
  ) %>%
  vip(geom = "point")
```




```r
### view metrics

final_wf <- workflow() %>%
  add_recipe(tree_rec) %>%
  add_model(final_rf)

final_res <- final_wf %>%
  last_fit(trees_split)

final_res %>%
  collect_metrics()
```

```
## # A tibble: 2 x 4
##   .metric  .estimator .estimate .config
##   <chr>    <chr>          <dbl> <chr>
## 1 accuracy binary         0.801 Preprocessor1_Model1
## 2 roc_auc  binary         0.800 Preprocessor1_Model1
```

Precision-Response AUC and Confusion Matrix for Oscar Nominations (with TidyModels)

This should be compared with Dan's Logistic Regression model.

```
set.seed(777)

## check pr_auc
## good ref: https://machinelearningmastery.com/roc-curves-and-precision-recall-curves-for-imbalanced-c
## want to keep precision AND recall high as threshold increases

fit_rf_oscar_nom <-
  final_wf %>%
  fit(trees_train)

pred_rf_oscar_nom <-
  predict(fit_rf_oscar_nom, trees_test) %>%
  bind_cols(predict(fit_rf_oscar_nom, trees_test, type = "prob")) %>%
  bind_cols(trees_test %>% dplyr::select(oscar_nom))

pred_rf_oscar_nom %>%
  pr_auc(event_level = "second", truth = oscar_nom, .pred_1)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 pr_auc  binary         0.578
```

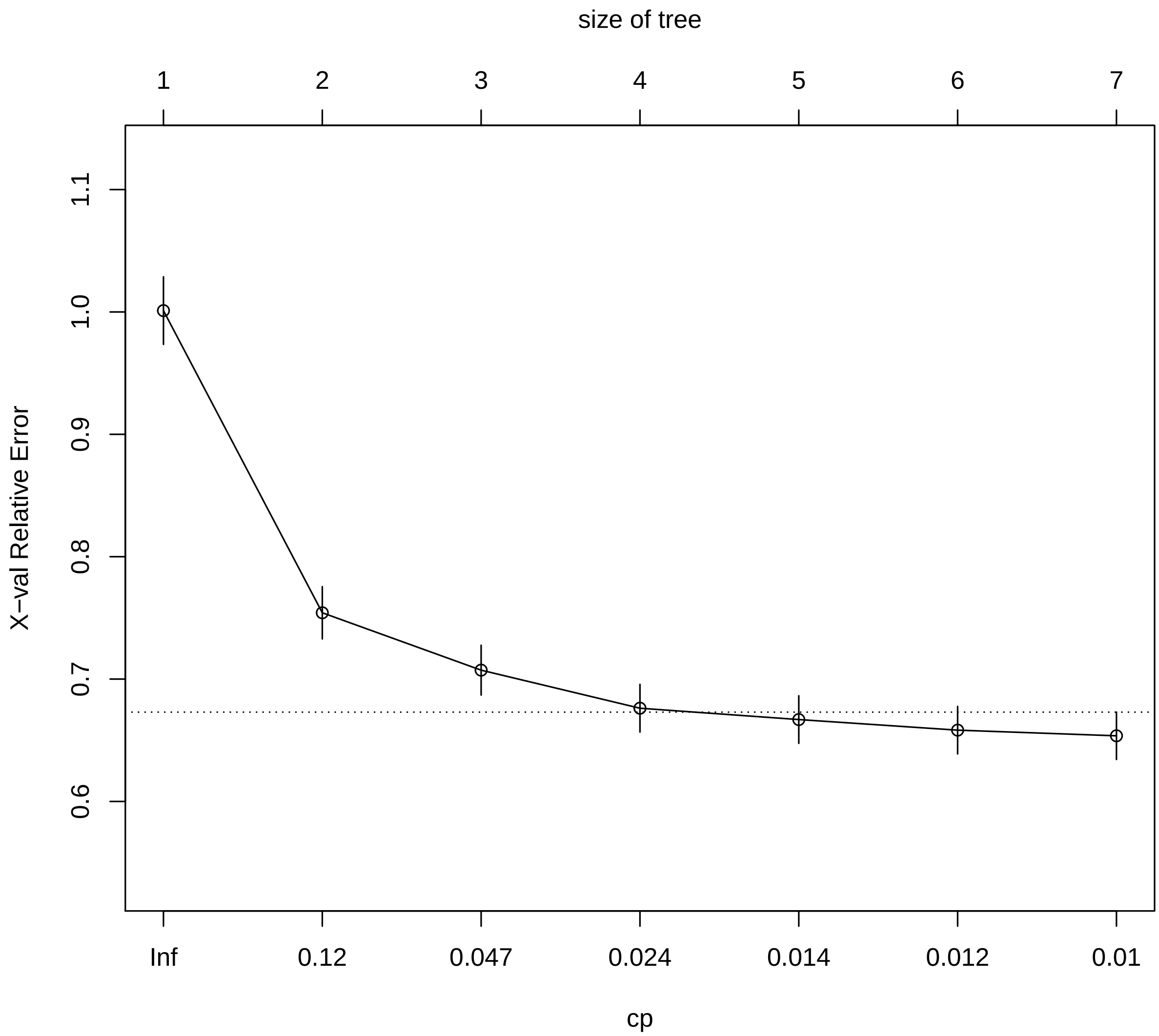
```
###
pred_v_real <- final_res %>%
  dplyr::select(.predictions) %>%
  unnest(.predictions) %>%
  as_tibble() %>%
  dplyr::select(.pred_class, oscar_nom)

conf_mat(pred_v_real, oscar_nom, .pred_class)
```

```
##           Truth
## Prediction   0   1
##          0 662 142
##          1  48 101
```

Predicting Box Office Profits with Single Tree and RF

```
set.seed(777)

imdb_details_extd3 <- read.csv("C:\\Users\\amiro\\Desktop\\Statistics 405\\Week 5\\Final_Project_Brains
imdb_details_extd3$star_power <- log(imdb_details_extd3$star_power+1)
imdb_details_extd3$wr_pop <- log(imdb_details_extd3$wr_pop+1)
imdb_details_extd3$inf_adjusted_gp = imdb_details_extd3$inf_adjusted_gp^.25



## randomly select genres if more than one
```

```r
tt<- lapply(imdb_details_extd3$genres, strsplit, ", ")
r_genre <- c()
for (i in 1:length(tt)) {

  if (identical(tt[[i]][[1]], character(0))) {
    name <- "None"
  } else {
    name <- sample(tt[[i]][[1]], 1)
  }
  r_genre <- c(r_genre, name)
}

imdb_details_extd3$genres <- r_genre


## simple single tree
tr <- rpart(inf_adjusted_gp ~ genres+rating+dir_pop_fac+budget_adj+runtime+release_period+
             co_size+star_power+wr_pop, data=imdb_details_extd3)
## print plot to help choose cp
plotcp(tr)
```

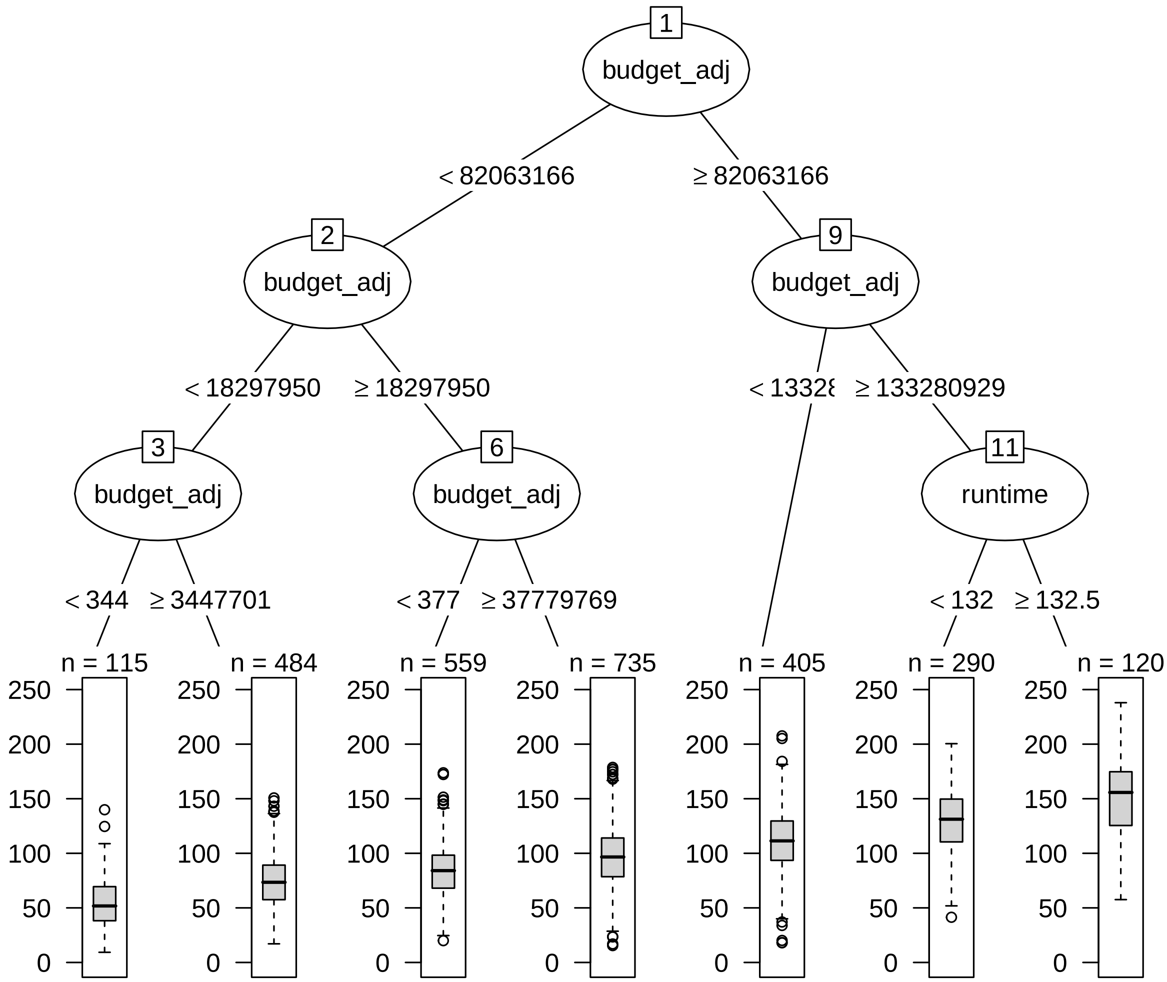



```
## prune the tree
tr_2 <- rpart::prune(tr, cp=0.012)
print(tr_2)
```

```
## n=2708 (700 observations deleted due to missingness)
##
## node), split, n, deviance, yval
##       * denotes terminal node
##
##  1) root 2708 3224647.0  96.51553
##    2) budget_adj< 8.206317e+07 1893 1498279.0  85.07092
##      4) budget_adj< 1.829795e+07 599  387245.1  70.44133 *
##      5) budget_adj>=1.829795e+07 1294  923488.2  91.84305
##       10) budget_adj< 3.777977e+07 559  326229.4  84.85289 *
##       11) budget_adj>=3.777977e+07 735  549171.3  97.15937 *
##    3) budget_adj>=8.206317e+07 815  902527.2 123.09790
##      6) budget_adj< 1.332809e+08 405  352237.6 110.89160 *
##      7) budget_adj>=1.332809e+08 410  430339.7 135.15540
##       14) runtime< 132.5 290  249979.6 128.76830 *
```

```
##        15) runtime>=132.5 120  139939.5 150.59080 *
```

```
## plot the pruned tree
plot(as.party(tr), tp_args = list(id = FALSE))
```



Predicting Box Office Profit with TidyModels

```
df <- filter(imdb_details_extd3, type == "Movie") %>%
  na.omit()

data_split <- initial_split(df)
data_train <- training(data_split)
data_test <- testing(data_split)


# build recipe (just instructions)
model_rec <- recipe(inf_adjusted_gp ~  budget_adj+runtime+genres+rating+dir_pop_fac+release_period+
```

```r
                   runtime+co_size+star_power+wr_pop,
                   data = data_train) %>%
  step_other(genres, threshold = 0.03) %>%
  step_unknown(genres) %>%
  step_other(rating, threshold = 0.05) %>%
  step_unknown(rating) %>%
  step_dummy(all_nominal(), -all_outcomes())

# prep actually uses the data
model_prep <- prep(model_rec)
juiced <- juice(model_prep)

# run the below to check the step_other results (doesnt work if step_dummy already used)
# juiced %>% count(genres, sort = T)

# report details
summary(model_rec)
```

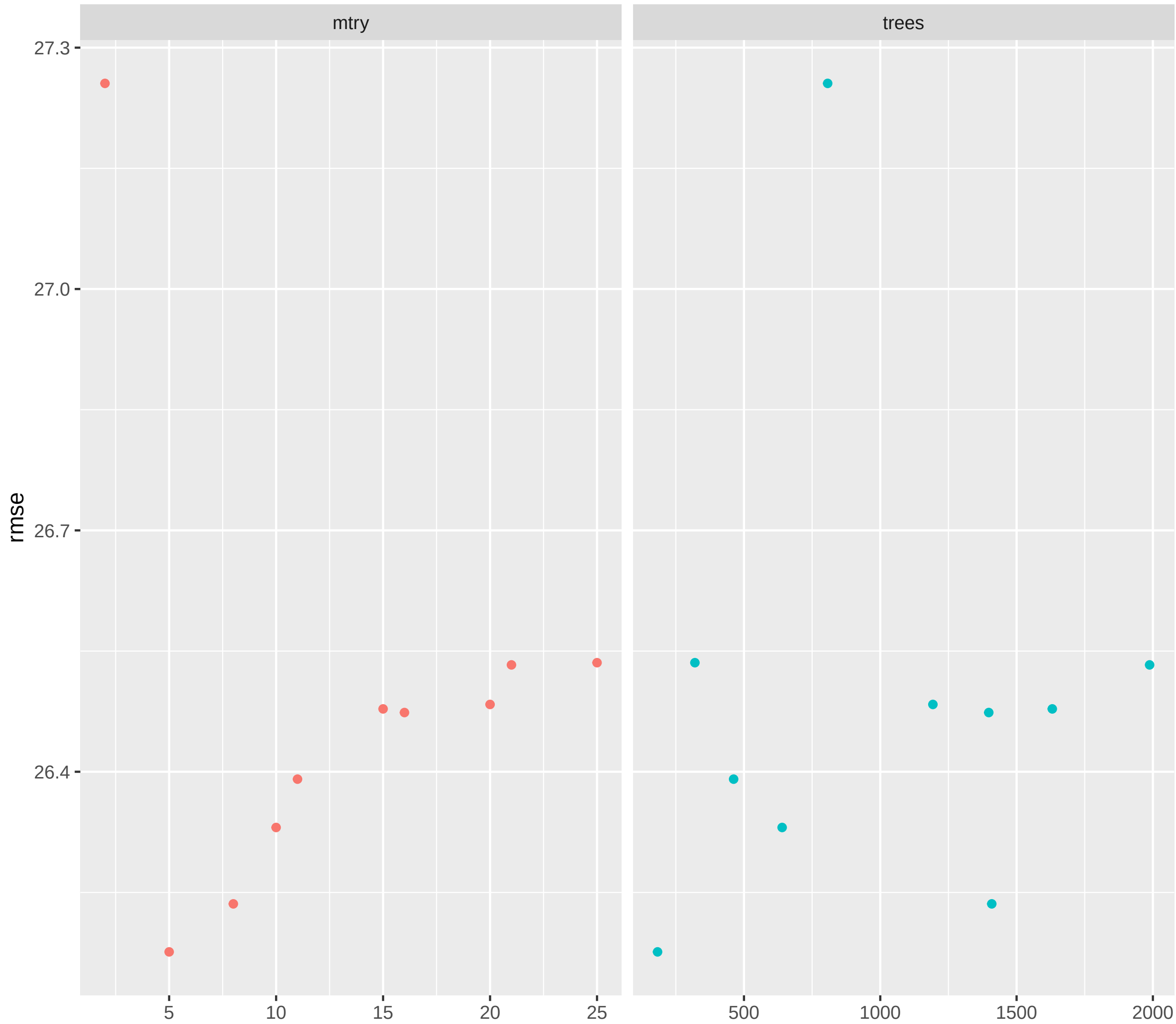
```
## # A tibble: 10 x 4
##    variable        type    role      source
##    <chr>           <chr>   <chr>     <chr>
##  1 budget_adj      numeric predictor original
##  2 runtime         numeric predictor original
##  3 genres          nominal predictor original
##  4 rating          nominal predictor original
##  5 dir_pop_fac     numeric predictor original
##  6 release_period  nominal predictor original
##  7 co_size         numeric predictor original
##  8 star_power      numeric predictor original
##  9 wr_pop          numeric predictor original
## 10 inf_adjusted_gp numeric outcome   original
```

```r
# build model for tuning
rf_bo_profit_spec <- rand_forest(
  mtry = tune(),
  trees = tune()
) %>%
  set_mode("regression") %>%
  set_engine("ranger")

rf_tune_wf <- workflow() %>%
  add_recipe(model_rec) %>%
  add_model(rf_bo_profit_spec)

# create a set of cross-validation resamples to use for tuning
data_folds <- vfold_cv(data_train)

# choose 10 grid points automatically
rf_tune_res <- tune_grid(
  rf_tune_wf,
  resamples = data_folds,
  grid = 10
)
```

```
## i Creating pre-processing data to finalize unknown parameter: mtry
```

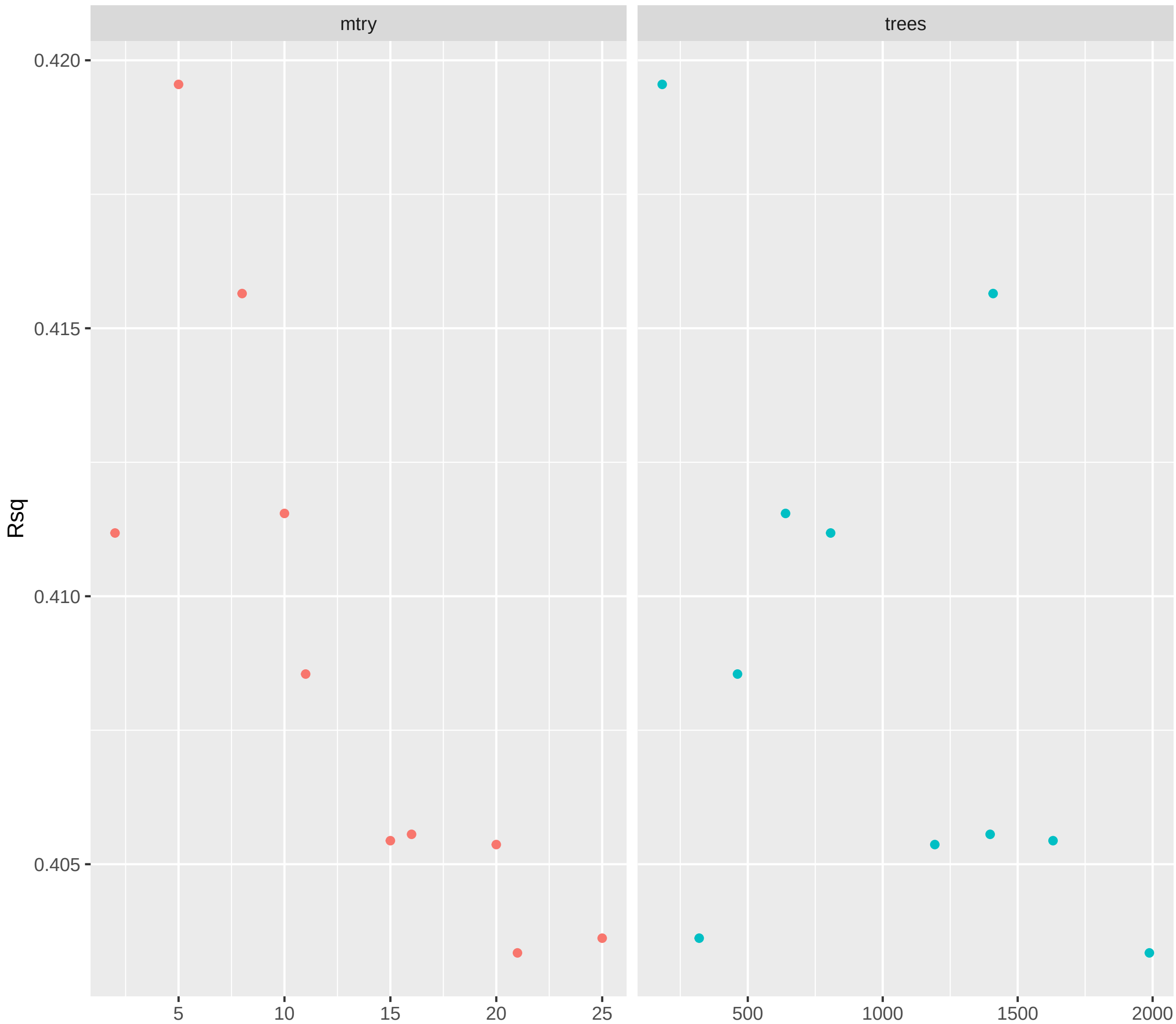
```
## ! Fold04: preprocessor 1/1, model 9/10: 25 columns were requested but there were 2...

## ! Fold07: preprocessor 1/1, model 9/10: 25 columns were requested but there were 2...
```

```
### rmse plot for tuning mtry and number of trees
rf_tune_res %>%
  collect_metrics() %>%
  filter(.metric == "rmse") %>%
  dplyr::select(mean, trees, mtry) %>%
  pivot_longer(trees:mtry,
               values_to = "value",
               names_to = "parameter"
  ) %>%
  ggplot(aes(value, mean, color = parameter)) +
  geom_point(show.legend = FALSE) +
  facet_wrap(~parameter, scales = "free_x") +
  labs(x = NULL, y = "rmse")
```



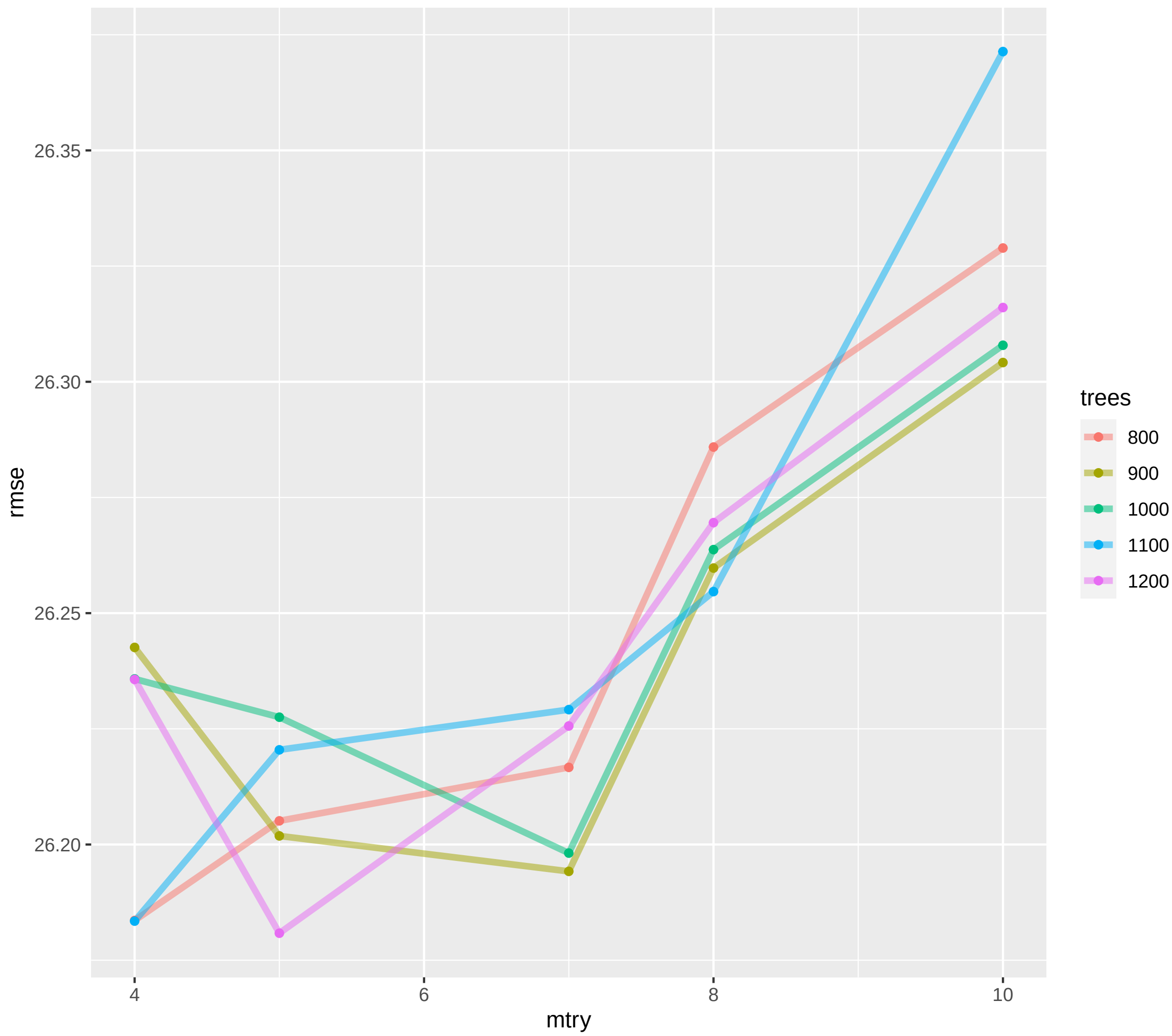

```
### rsq plot for tuning mtry and number of trees
rf_tune_res %>%
```

```
  collect_metrics() %>%
  filter(.metric == "rsq") %>%
  dplyr::select(mean, trees, mtry) %>%
  pivot_longer(trees:mtry,
               values_to = "value",
               names_to = "parameter"
  ) %>%
  ggplot(aes(value, mean, color = parameter)) +
  geom_point(show.legend = FALSE) +
  facet_wrap(~parameter, scales = "free_x") +
  labs(x = NULL, y = "Rsq")
```

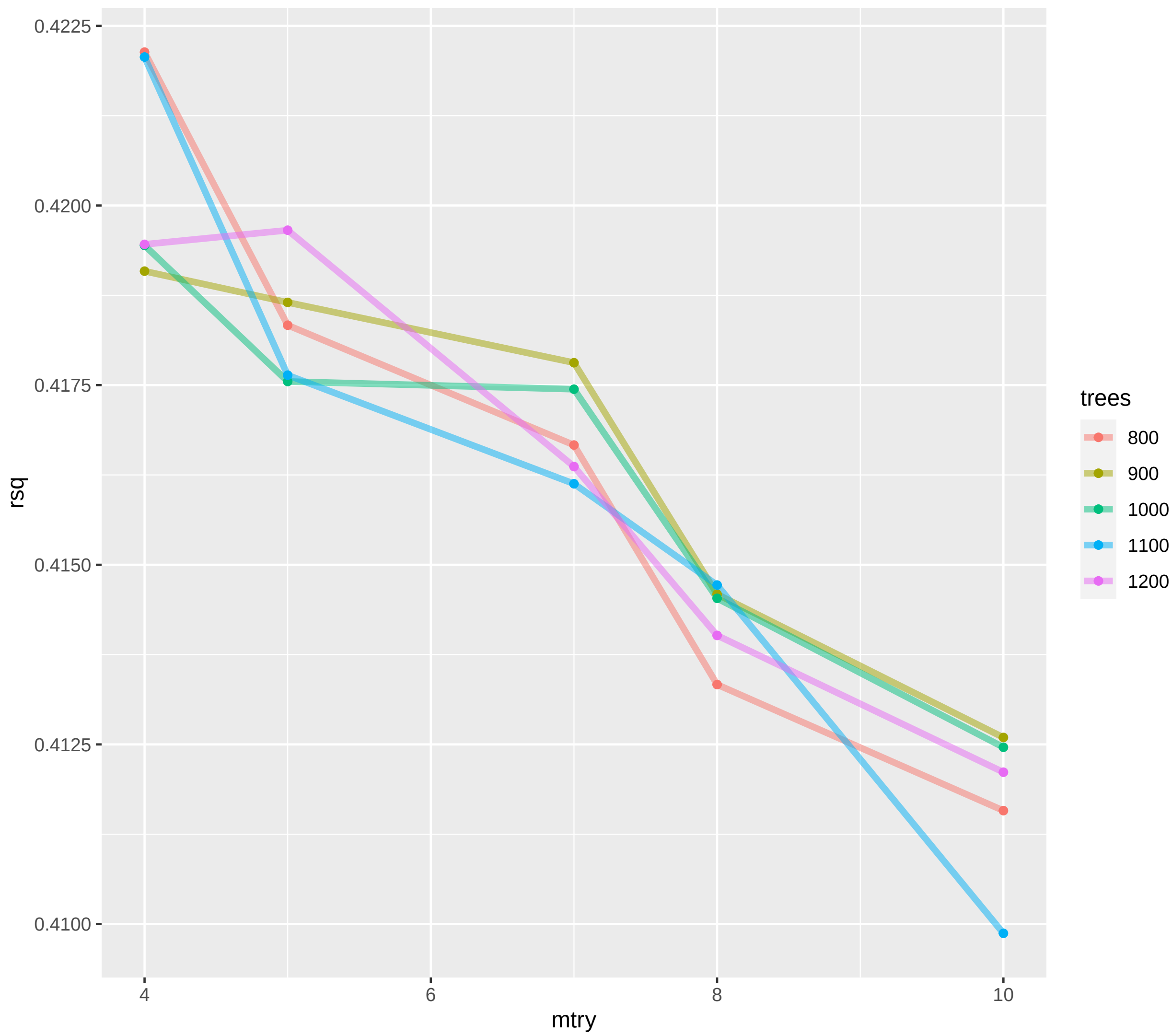



```
## looks like 3 for mtry and 1000-1500 for trees could work best

### taking a closer look now

rf_grid <- grid_regular(
  mtry(range = c(4, 10)),
  trees(range = c(800, 1200)),
```

```
  levels = 5
)

rf_res <- tune_grid(
  rf_tune_wf,
  resamples = data_folds,
  grid = rf_grid
)

## rmse plot for tuning mtry and number of trees
rf_res %>%
  collect_metrics() %>%
  filter(.metric == "rmse") %>%
  mutate(trees = factor(trees)) %>%
  ggplot(aes(mtry, mean, color = trees)) +
  geom_line(alpha = 0.5, size = 1.5) +
  geom_point() +
  labs(y = "rmse")
```

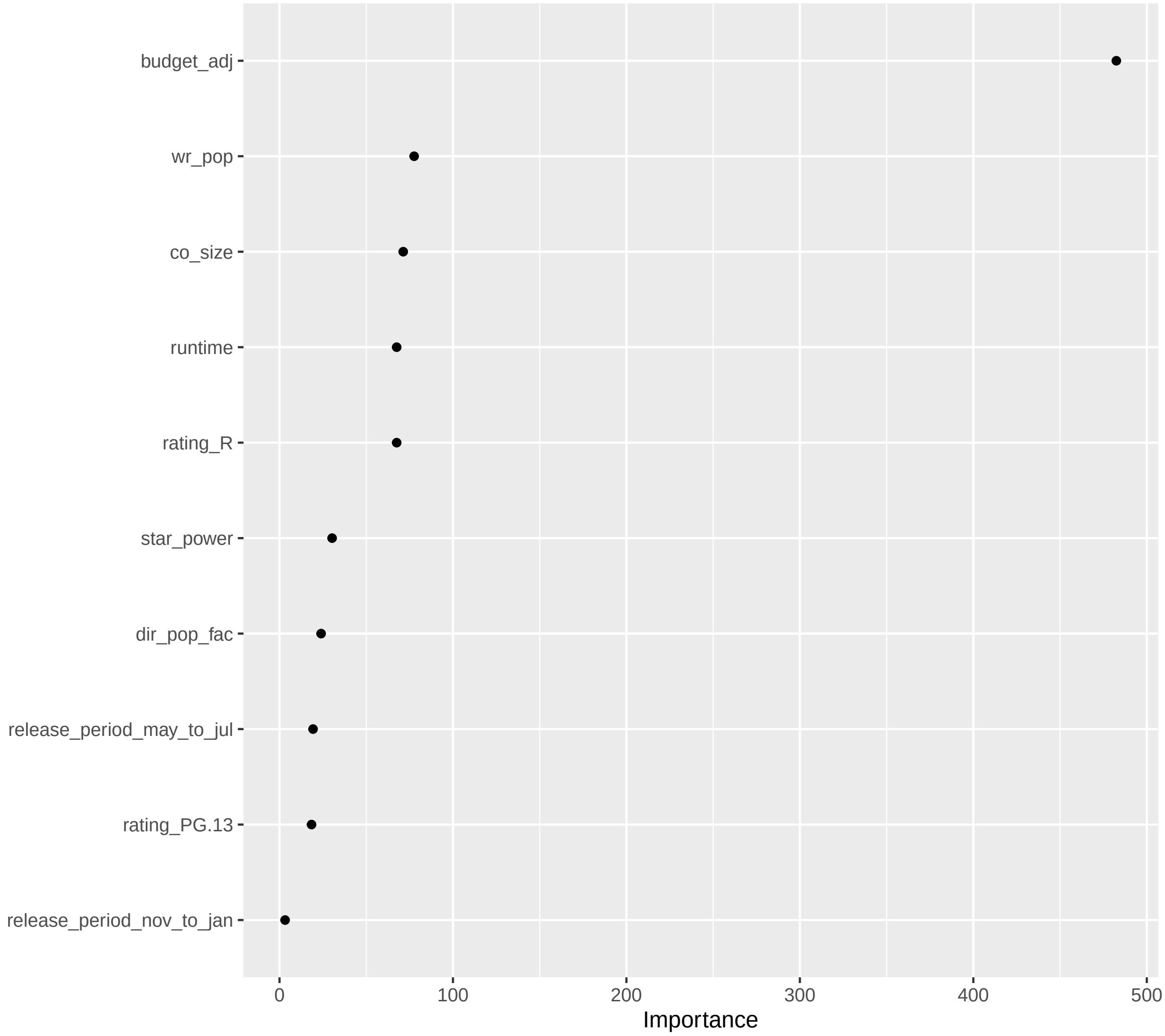


```r
## rsq plot for tuning mtry and number of trees
rf_res %>%
  collect_metrics() %>%
  filter(.metric == "rsq") %>%
  mutate(trees = factor(trees)) %>%
  ggplot(aes(mtry, mean, color = trees)) +
  geom_line(alpha = 0.5, size = 1.5) +
  geom_point() +
  labs(y = "rsq")
```




```r
## looks like 7 for mtry and 1100 for trees is optimal

## build rf model with tuned params
final_rf <- rand_forest(
  mtry = 7,
  trees = 1100,
) %>%
  set_mode("regression") %>%
  set_engine("ranger")
```

```r
# checking out importance plots
final_rf %>%
  set_engine("ranger", importance = "permutation") %>%
  fit(inf_adjusted_gp ~ .,
      data = juice(model_prep)
  ) %>%
  vip(geom = "point")
```




```r
### view metrics

final_wf <- workflow() %>%
  add_recipe(model_rec) %>%
  add_model(final_rf)

final_res <- final_wf %>%
  last_fit(data_split)

final_res %>%
```

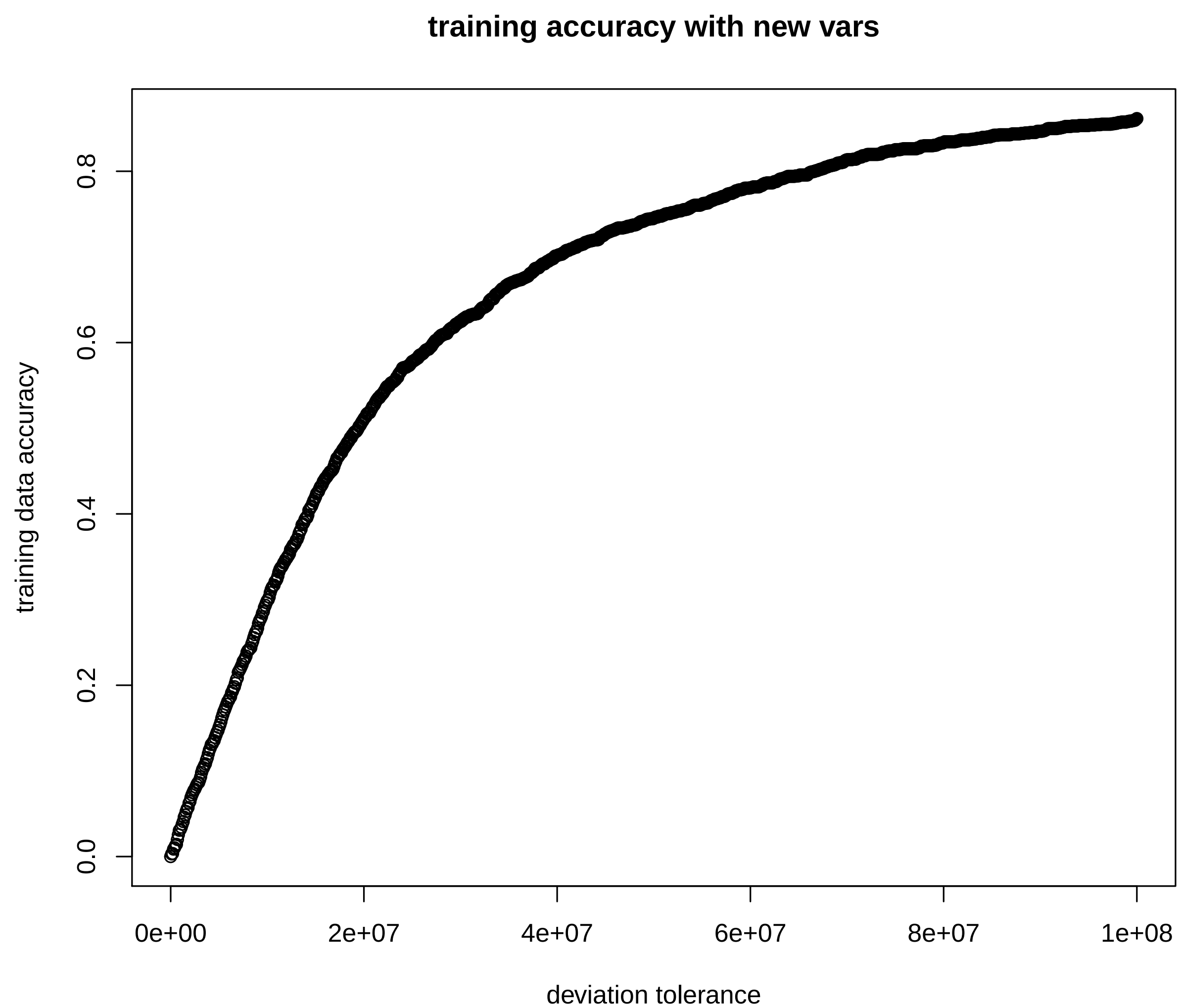
```
  collect_metrics()
```

```
## # A tibble: 2 x 4
##   .metric .estimator .estimate .config
##   <chr>   <chr>          <dbl> <chr>
## 1 rmse    standard      25.6   Preprocessor1_Model1
## 2 rsq     standard       0.457 Preprocessor1_Model1
```

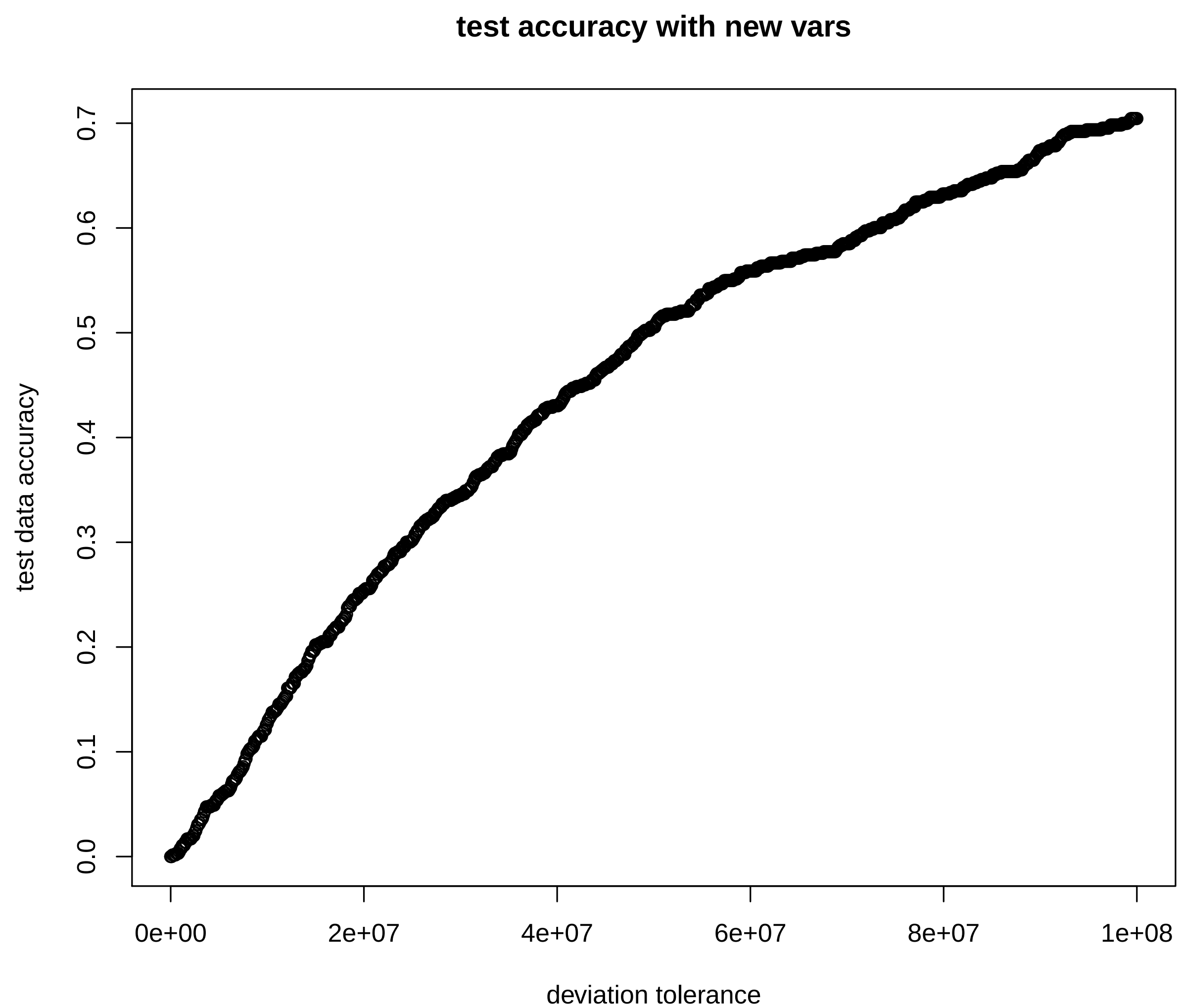
Plot Accuracy For Box Office Profit Random Forest (transformed data back)

Results looks great ... when the data is transformed. After transforming back they look kind of like this.

```
fit_rf_oscar_nom <-
  final_wf %>%
  fit(data_train)

## compare the predictions to the data
tr_comp <- data.frame(true_gp=data_train$inf_adjusted_gp, predict(fit_rf_oscar_nom, data_train))

## approximate training accuracy
devs <- abs((tr_comp$true_gp)^4 - (tr_comp$.pred)^4)
close_enoughs <- function(x) sum(devs <= x)/ length(devs)
x <- seq(from=0,to=100000000,by=100000)
plot(x, sapply(x, close_enoughs), main="training accuracy with new vars", xlab="deviation tolerance", yl
```




```
## compare the predictions to the data
tr_comp <- data.frame(true_gp=data_test$inf_adjusted_gp, predict(fit_rf_oscar_nom, data_test))

## approximate test accuracy
devs <- abs(tr_comp$true_gp^4 - tr_comp$.pred^4)
x <- seq(from=0,to=100000000,by=100000)
plot(x, sapply(x, close_enoughs), main="test accuracy with new vars", xlab="deviation tolerance", ylab=
```



Compare Box Office Profit Random Forest with Linear Model

```
# build recipe (just instructions)
lm_model_rec <- recipe(inf_adjusted_gp ~  budget_adj+runtime+dir_pop_fac+release_period+
                      runtime+co_size+star_power+wr_pop,
                    data = data_train)

## build lm model with tuned params
final_lm <- linear_reg() %>%
  set_engine("lm") %>%
  set_mode("regression")


lm_final_wf <- workflow() %>%
  add_recipe(lm_model_rec) %>%
  add_model(final_lm)
```

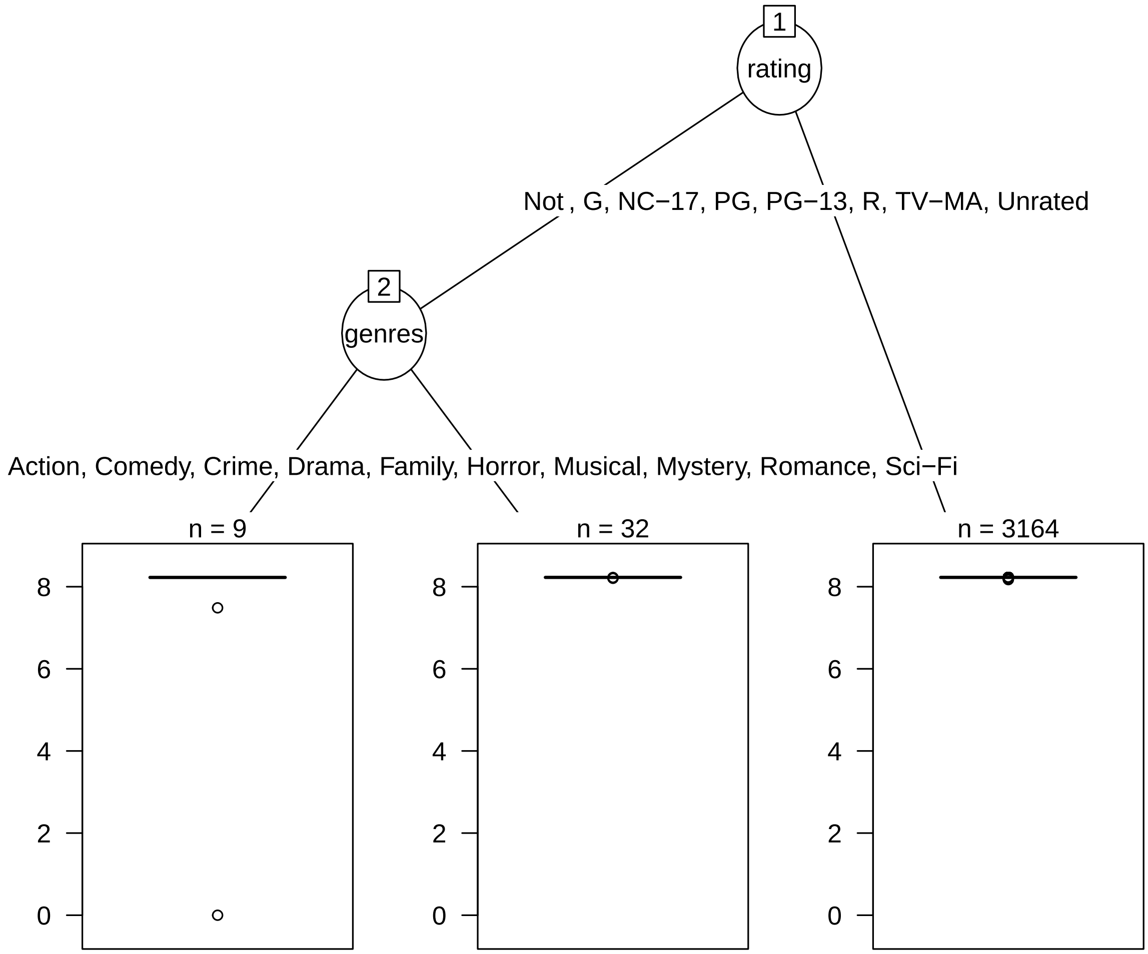
```
lm_final_res <- lm_final_wf %>%
  last_fit(data_split)

lm_final_res %>%
  collect_metrics()
```

```
## # A tibble: 2 x 4
##   .metric .estimator .estimate .config
##   <chr>   <chr>          <dbl> <chr>
## 1 rmse    standard      26.9   Preprocessor1_Model1
## 2 rsq     standard       0.402 Preprocessor1_Model1
```

Predicting GPM with Single Tree and RF

Pretty boring looking tree.

```
set.seed(777)

imdb_details_extd3 <- read.csv("C:\\Users\\amiro\\Desktop\\Statistics 405\\Week 5\\Final_Project_Brains
imdb_details_extd3 <- filter(imdb_details_extd3, type == "Movie") %>%
  na.omit()
imdb_details_extd3$star_power <- log(imdb_details_extd3$star_power+1)
imdb_details_extd3$wr_pop <- log(imdb_details_extd3$wr_pop+1)
C <- min(imdb_details_extd3$gpm)
imdb_details_extd3$gpm = log(imdb_details_extd3$gpm+1-C)


## randomly select genres if more than one
tt<- lapply(imdb_details_extd3$genres, strsplit, ", ")
r_genre <- c()
for (i in 1:length(tt)) {

  if (identical(tt[[i]][[1]], character(0))) {
    name <- "None"
  } else {
    name <- sample(tt[[i]][[1]], 1)
  }
  r_genre <- c(r_genre, name)
}

imdb_details_extd3$genres <- r_genre

## simple single tree
tr <- rpart(gpm ~ genres+rating+dir_pop_fac+budget_adj+runtime+release_period+
              co_size+star_power+wr_pop, data=imdb_details_extd3)


## plot the pruned tree
plot(as.party(tr), tp_args = list(id = FALSE))
```



Predicting GPM with Random Forest in TidyModels

Notice the RMSE is not that great given the transformation, and the Rsq is absolute trash. Not sure if i made a mistake or something. Should discuss this.

```
df <- filter(imdb_details_extd3, type == "Movie") %>%
  na.omit()

data_split <- initial_split(df)
data_train <- training(data_split)
data_test <- testing(data_split)


# build recipe (just instructions)
model_rec <- recipe(gpm ~  budget_adj+runtime+genres+rating+dir_pop_fac+release_period+
                      runtime+co_size+star_power+wr_pop,
                    data = data_train) %>%
```

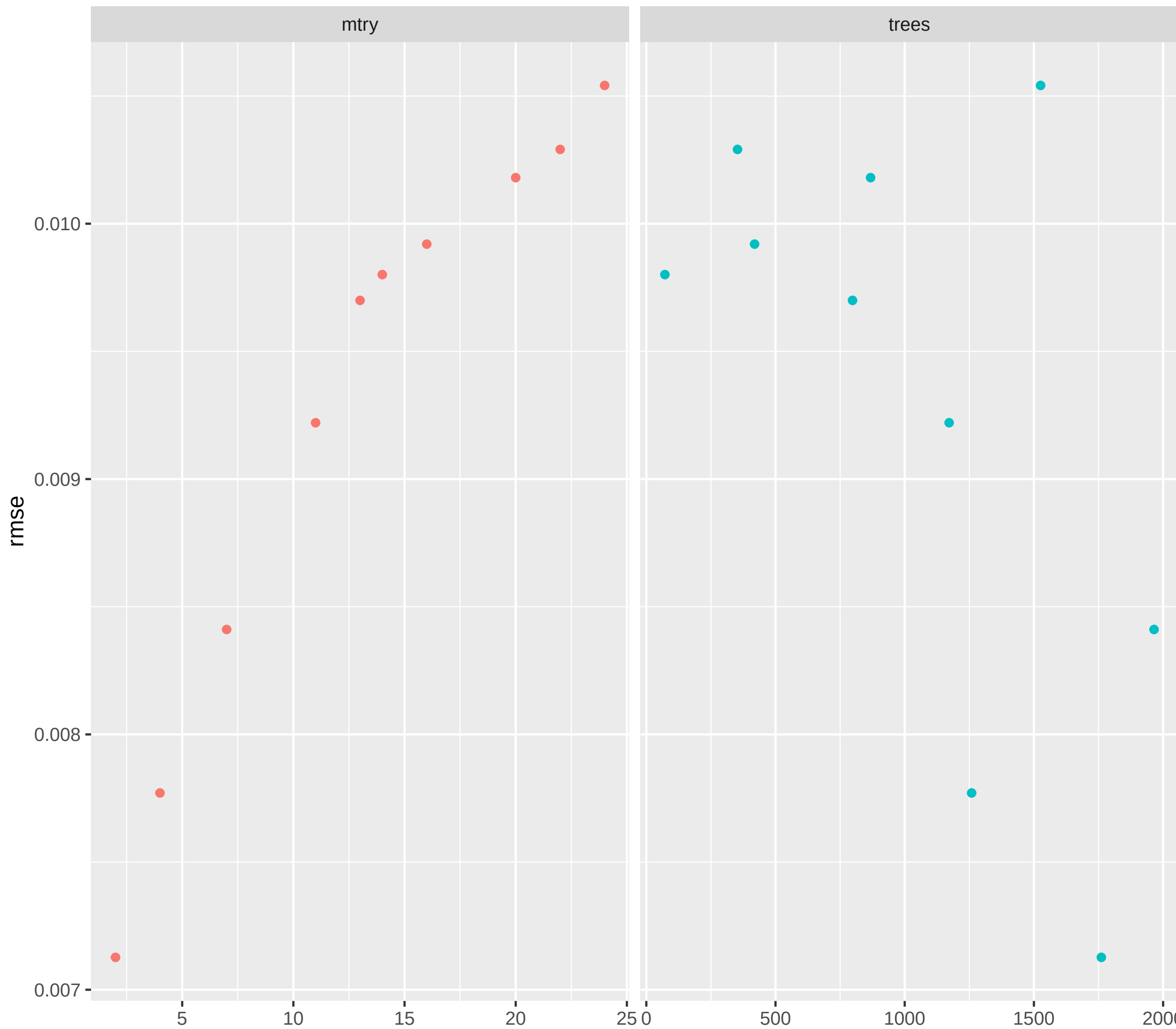
```r
  step_other(genres, threshold = 0.03) %>%
  step_unknown(genres) %>%
  step_other(rating, threshold = 0.05) %>%
  step_unknown(rating) %>%
  step_dummy(all_nominal(), -all_outcomes())

# prep actually uses the data
model_prep <- prep(model_rec)
juiced <- juice(model_prep)

# run the below to check the step_other results (doesnt work if step_dummy already used)
# juiced %>% count(genres, sort = T)

# report details
summary(model_rec)
```

```
## # A tibble: 10 x 4
##    variable       type    role      source
##    <chr>          <chr>   <chr>     <chr>
##  1 budget_adj     numeric predictor original
##  2 runtime        numeric predictor original
##  3 genres         nominal predictor original
##  4 rating         nominal predictor original
##  5 dir_pop_fac    numeric predictor original
##  6 release_period nominal predictor original
##  7 co_size        numeric predictor original
##  8 star_power     numeric predictor original
##  9 wr_pop         numeric predictor original
## 10 gpm            numeric outcome   original
```

```r
# build model for tuning
rf_bo_profit_spec <- rand_forest(
  mtry = tune(),
  trees = tune()
) %>%
  set_mode("regression") %>%
  set_engine("ranger")

rf_tune_wf <- workflow() %>%
  add_recipe(model_rec) %>%
  add_model(rf_bo_profit_spec)

# create a set of cross-validation resamples to use for tuning
data_folds <- vfold_cv(data_train)

# choose 10 grid points automatically
rf_tune_res <- tune_grid(
  rf_tune_wf,
  resamples = data_folds,
  grid = 10
)
```

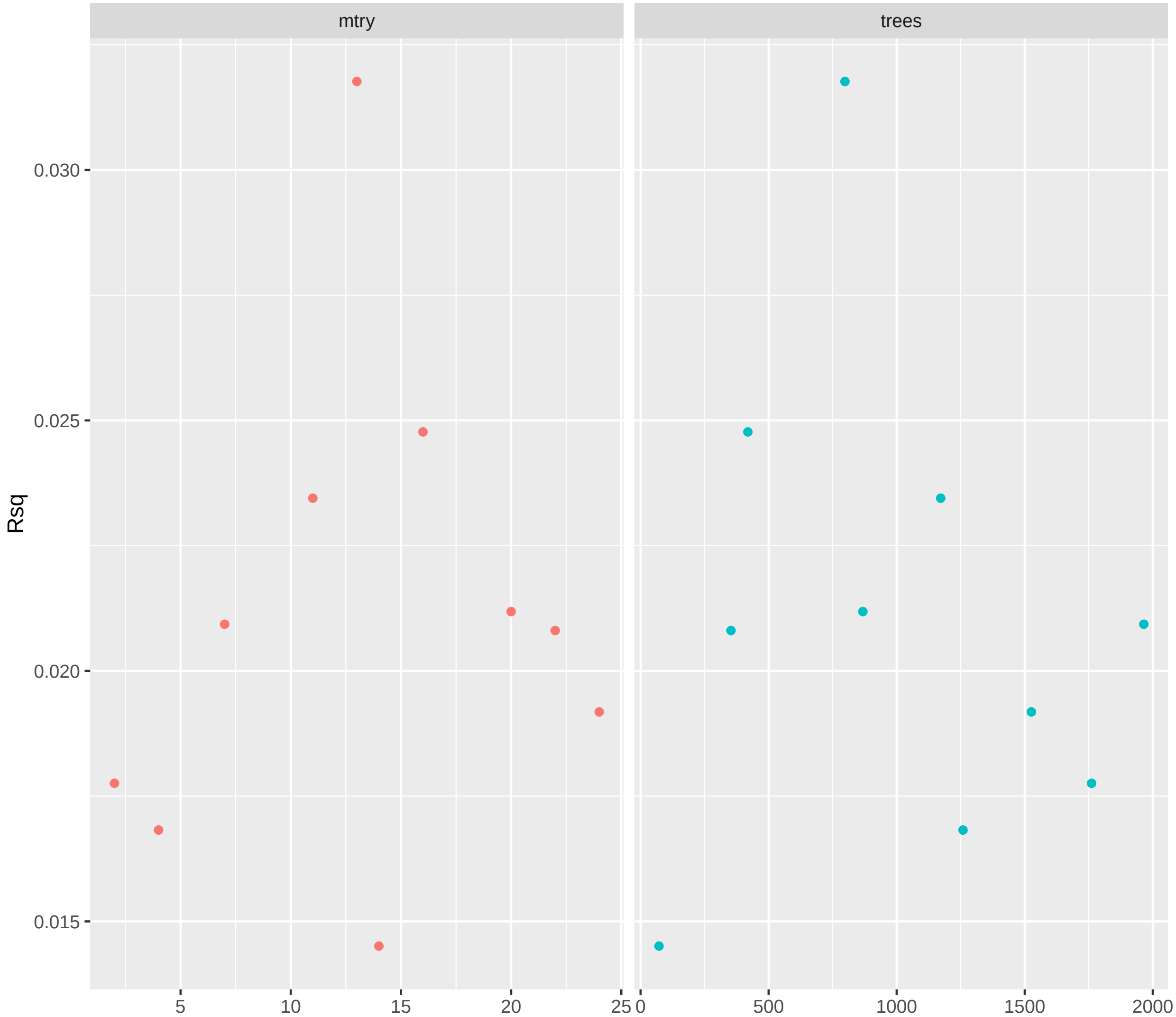
```
## i Creating pre-processing data to finalize unknown parameter: mtry
```

```
### rmse plot for tuning mtry and number of trees
rf_tune_res %>%
  collect_metrics() %>%
  filter(.metric == "rmse") %>%
  dplyr::select(mean, trees, mtry) %>%
  pivot_longer(trees:mtry,
               values_to = "value",
               names_to = "parameter"
  ) %>%
  ggplot(aes(value, mean, color = parameter)) +
  geom_point(show.legend = FALSE) +
  facet_wrap(~parameter, scales = "free_x") +
  labs(x = NULL, y = "rmse")
```



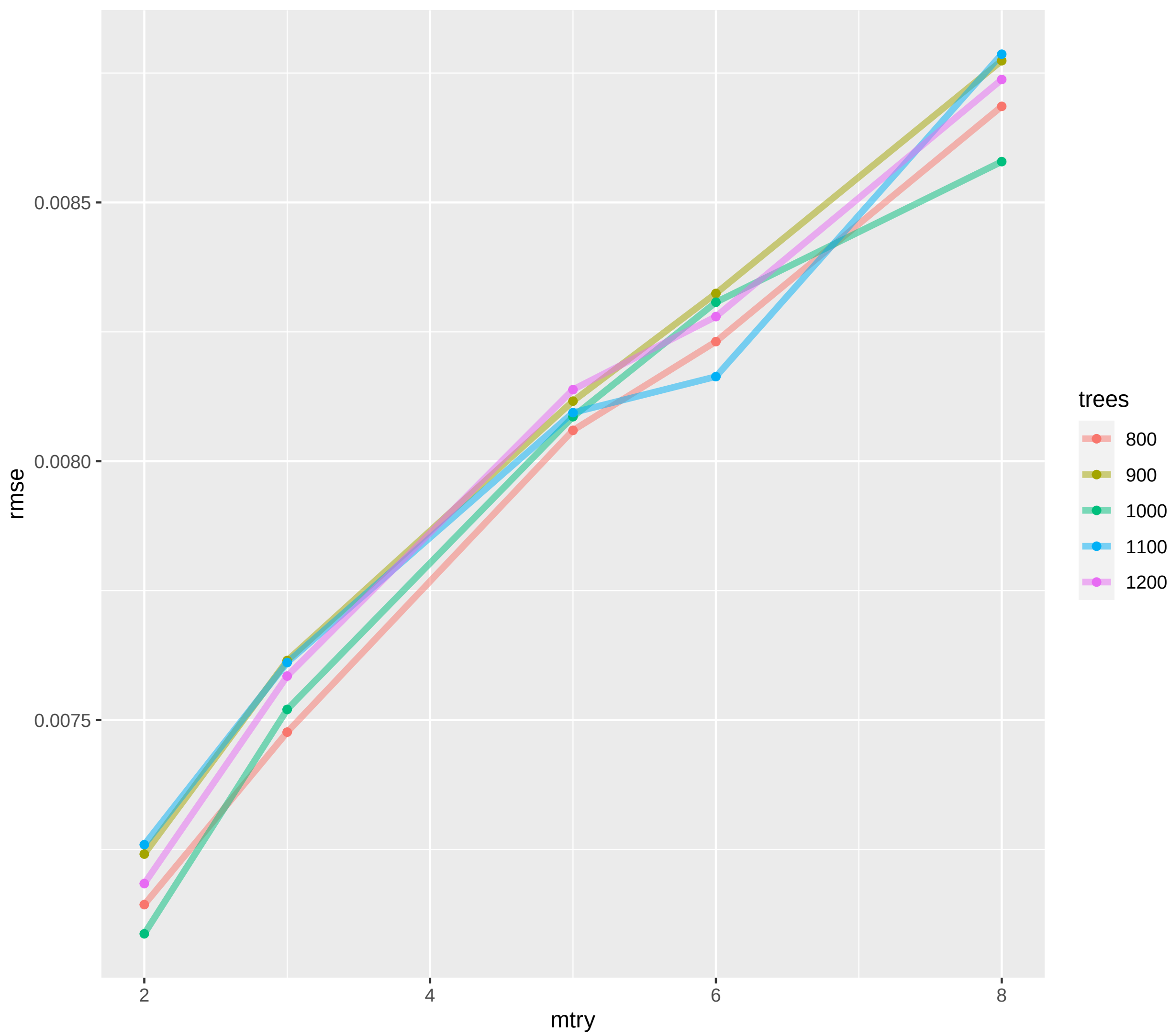

```
### rsq plot for tuning mtry and number of trees
rf_tune_res %>%
  collect_metrics() %>%
  filter(.metric == "rsq") %>%
  dplyr::select(mean, trees, mtry) %>%
```

```
  pivot_longer(trees:mtry,
               values_to = "value",
               names_to = "parameter"
  ) %>%
  ggplot(aes(value, mean, color = parameter)) +
  geom_point(show.legend = FALSE) +
  facet_wrap(~parameter, scales = "free_x") +
  labs(x = NULL, y = "Rsq")
```



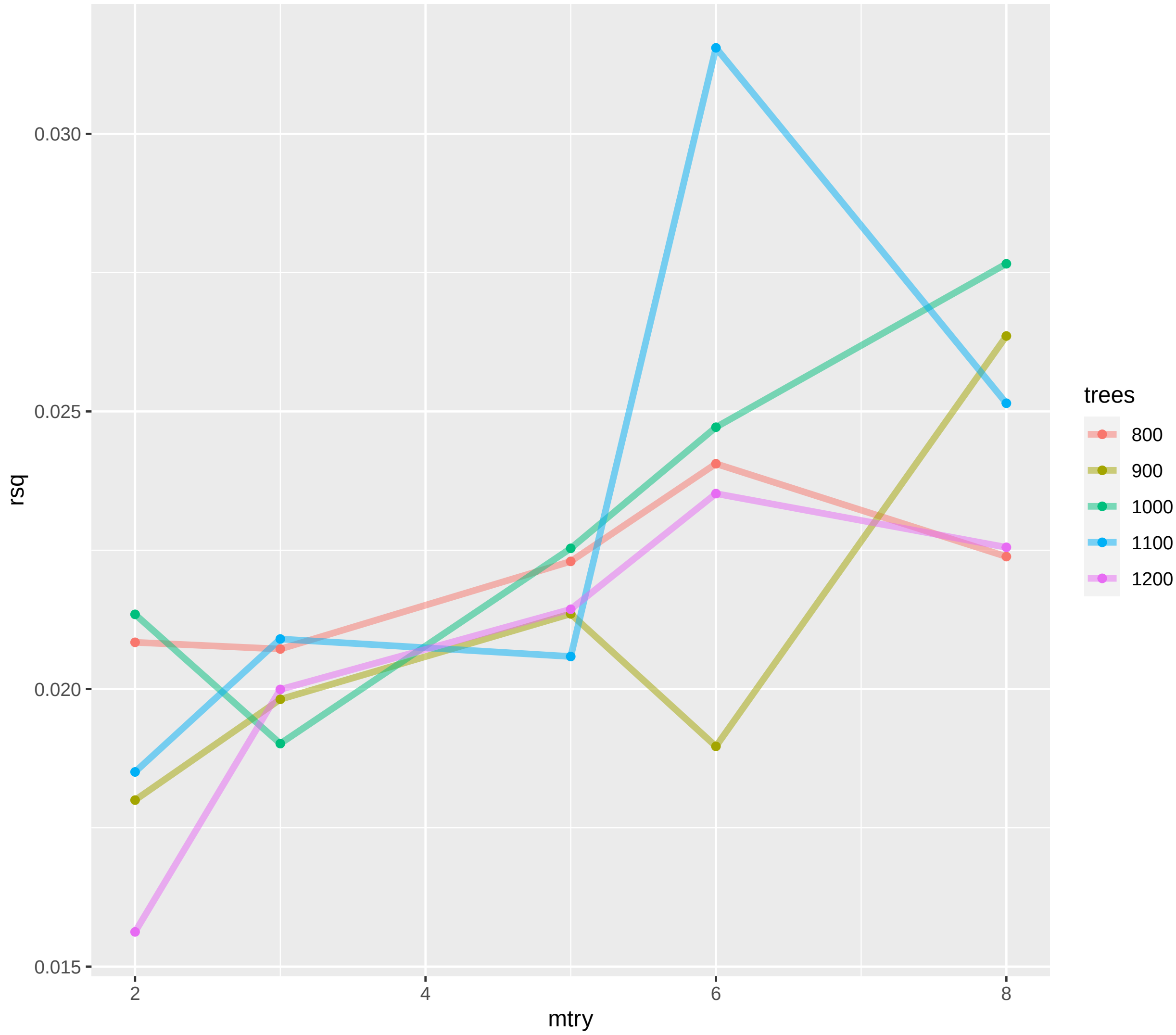

```
## looks like 3 for mtry and 1000-1500 for trees could work best

### taking a closer look now

rf_grid <- grid_regular(
  mtry(range = c(2, 8)),
  trees(range = c(800, 1200)),
  levels = 5
)
```

```
rf_res <- tune_grid(
  rf_tune_wf,
  resamples = data_folds,
  grid = rf_grid
)

## rmse plot for tuning mtry and number of trees
rf_res %>%
  collect_metrics() %>%
  filter(.metric == "rmse") %>%
  mutate(trees = factor(trees)) %>%
  ggplot(aes(mtry, mean, color = trees)) +
  geom_line(alpha = 0.5, size = 1.5) +
  geom_point() +
  labs(y = "rmse")
```



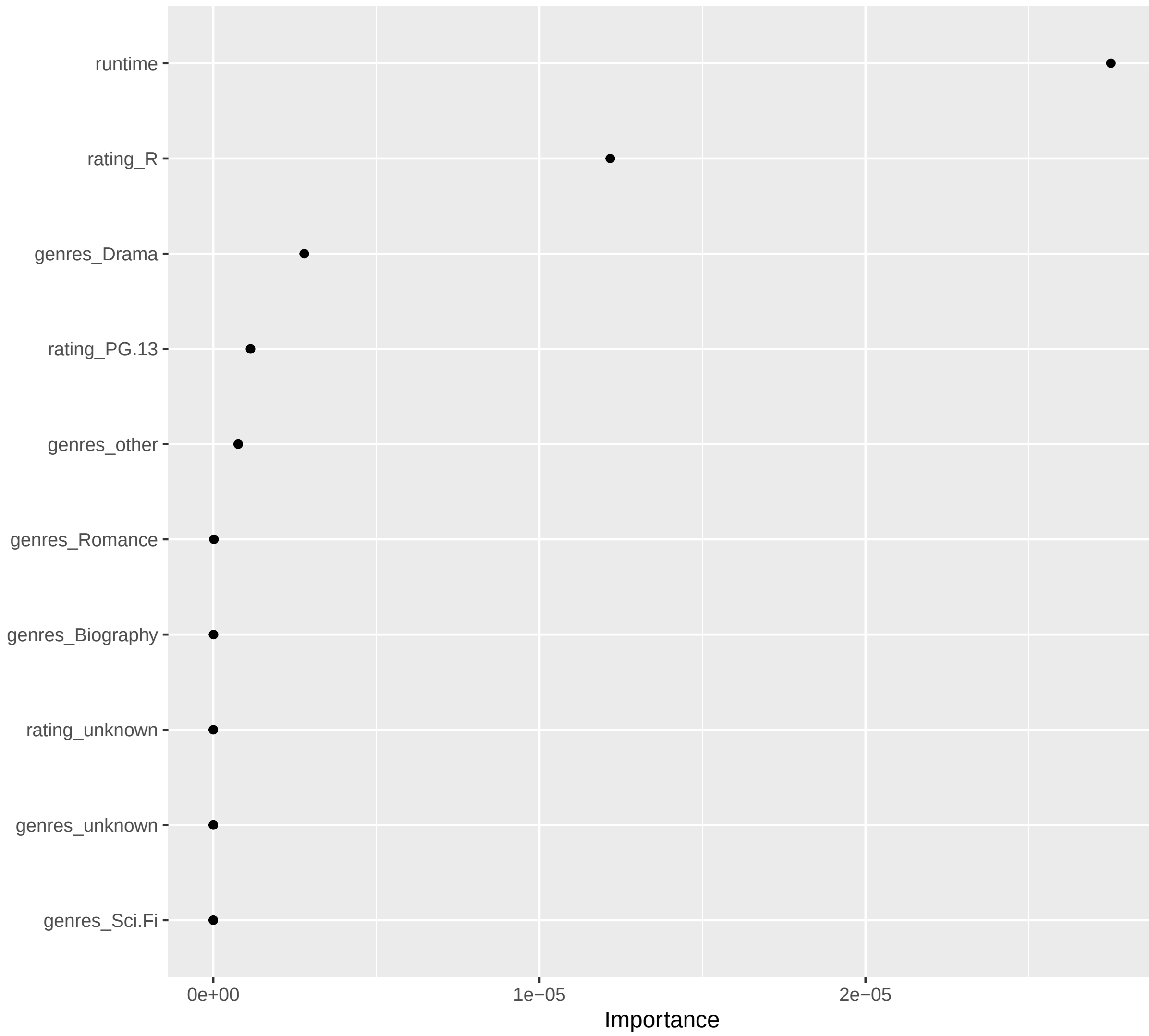

```
## rsq plot for tuning mtry and number of trees
rf_res %>%
  collect_metrics() %>%
```

```
  filter(.metric == "rsq") %>%
  mutate(trees = factor(trees)) %>%
  ggplot(aes(mtry, mean, color = trees)) +
  geom_line(alpha = 0.5, size = 1.5) +
  geom_point() +
  labs(y = "rsq")
```




```
## looks like 4 for mtry and 1000 for trees is optimal

## build rf model with tuned params
final_rf <- rand_forest(
  mtry = 4,
  trees = 1000,
) %>%
  set_mode("regression") %>%
  set_engine("ranger")

# checking out importance plots
final_rf %>%
```

```r
  set_engine("ranger", importance = "permutation") %>%
  fit(gpm ~ .,
      data = juice(model_prep)
  ) %>%
  vip(geom = "point")
```




```r
### view metrics

final_wf <- workflow() %>%
  add_recipe(model_rec) %>%
  add_model(final_rf)

final_res <- final_wf %>%
  last_fit(data_split)

final_res %>%
  collect_metrics()
```

```
## # A tibble: 2 x 4
```

```
##   .metric .estimator .estimate .config
##   <chr>   <chr>          <dbl> <chr>
## 1 rmse    standard     0.290   Preprocessor1_Model1
## 2 rsq     standard     0.00469 Preprocessor1_Model1
```

Compare Box Office Profit Random Forest with Linear Model

```
# build recipe (just instructions)
lm_model_rec <- recipe(gpm ~  budget_adj+runtime+dir_pop_fac+release_period+
                         runtime+co_size+star_power+wr_pop,
                       data = data_train)

## build lm model with tuned params
final_lm <- linear_reg() %>%
  set_engine("lm") %>%
  set_mode("regression")


lm_final_wf <- workflow() %>%
  add_recipe(lm_model_rec) %>%
  add_model(final_lm)

lm_final_res <- lm_final_wf %>%
  last_fit(data_split)

lm_final_res %>%
  collect_metrics()
```

```
## # A tibble: 2 x 4
##   .metric .estimator .estimate .config
##   <chr>   <chr>          <dbl> <chr>
## 1 rmse    standard     0.290   Preprocessor1_Model1
## 2 rsq     standard     0.00230 Preprocessor1_Model1
```