Problem Set 9

Geoffrey Hughes

11/15/2019

Can tree models predict movie profit?

a. Apply cleaning code

```
library("tidyverse")
```

```
## -- Attaching packages ------------------------------------------------------------------------

## v ggplot2 3.2.1     v purrr   0.3.3
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   1.0.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0

## -- Conflicts ---------------------------------------------------------------------------------
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library("ElemStatLearn")
library('partykit')
```

```
## Loading required package: grid

## Loading required package: libcoin

## Loading required package: mvtnorm
```

```
library('magrittr')
```

```
##
## Attaching package: 'magrittr'

## The following object is masked from 'package:purrr':
##
##     set_names

## The following object is masked from 'package:tidyr':
##
##     extract
```

```
library('caret')
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift
```

```
library('randomForest')
```

```
## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:dplyr':
##
##     combine

## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library('randomForestExplainer')
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
options(scipen = 50)
set.seed(1861)
movies <- read.csv(here::here("datasets", "movie_metadata.csv"))
movies <- movies %>% filter(budget < 4e+08) %>% filter(content_rating !=
"", content_rating != "Not Rated", plot_keywords != "", !is.na(gross))
movies <- movies %>% mutate(genre_main = unlist(map(strsplit(as.character(movies$genres),
"\\|"), 1)), plot_main = unlist(map(strsplit(as.character(movies$plot_keywords),
"\\|"), 1)), grossM = gross/1e+06, budgetM = budget/1e+06)
movies <- movies %>% mutate(genre_main = fct_lump(genre_main,
7), plot_first = fct_lump(plot_main, 20), content_rating = fct_lump(content_rating,
4), country = fct_lump(country, 8), language = fct_lump(language,
4), cast_total_facebook_likes000s = cast_total_facebook_likes/1000,
) %>% drop_na()
top_director <- movies %>% group_by(director_name) %>% summarize(num_films = n()) %>%
top_frac(0.1) %>% mutate(top_director = 1) %>% select(-num_films)
```

```
## Selecting by num_films
```

```r
movies <- movies %>% left_join(top_director, by = "director_name") %>%
mutate(top_director = replace_na(top_director, 0)) %>% select(-c(director_name,
actor_2_name, gross, genres, actor_1_name, movie_title, actor_3_name,
plot_keywords, movie_imdb_link, budget, color, aspect_ratio,
plot_main, actor_3_facebook_likes, actor_2_facebook_likes,
color, num_critic_for_reviews, num_voted_users, num_user_for_reviews,
actor_2_facebook_likes))
sapply(movies %>% select_if(is.factor), table)
```

```
## $language
##
##  English   French Mandarin  Spanish    Other
##     3576       32       13       22       70
##
## $country
## Australia    Canada     China    France   Germany Hong Kong     Spain
##        39        57        13        97        79        13        19
##        UK       USA     Other
##       315      2974       107
##
## $content_rating
##
##     G    PG PG-13     R Other
##    87   565  1306  1694    61
##
## $genre_main
##
##    Action Adventure Biography    Comedy     Crime     Drama    Horror
##       952       367       204       979       250       654       163
##     Other
##       144
##
## $plot_first
##
##            1950s            1970s            actor african american
##               18               18               24               24
##            alien        apartment             army         assassin
##               69               19               20               26
##             baby             bank              bar       basketball
##               22               19               18               18
##           battle            beach      best friend  box office flop
##               26               19               32               28
##              boy        christmas              cia          college
##               36               18               19               22
##            death           friend            Other
##               40               21             3157
```

```r
train_idx <- sample(1:nrow(movies), size = floor(0.75 * nrow(movies)))
movies_train <- movies %>% slice(train_idx)
movies_test <- movies %>% slice(-train_idx)
```

b. Ridgeline plot showing grossM against plot_first

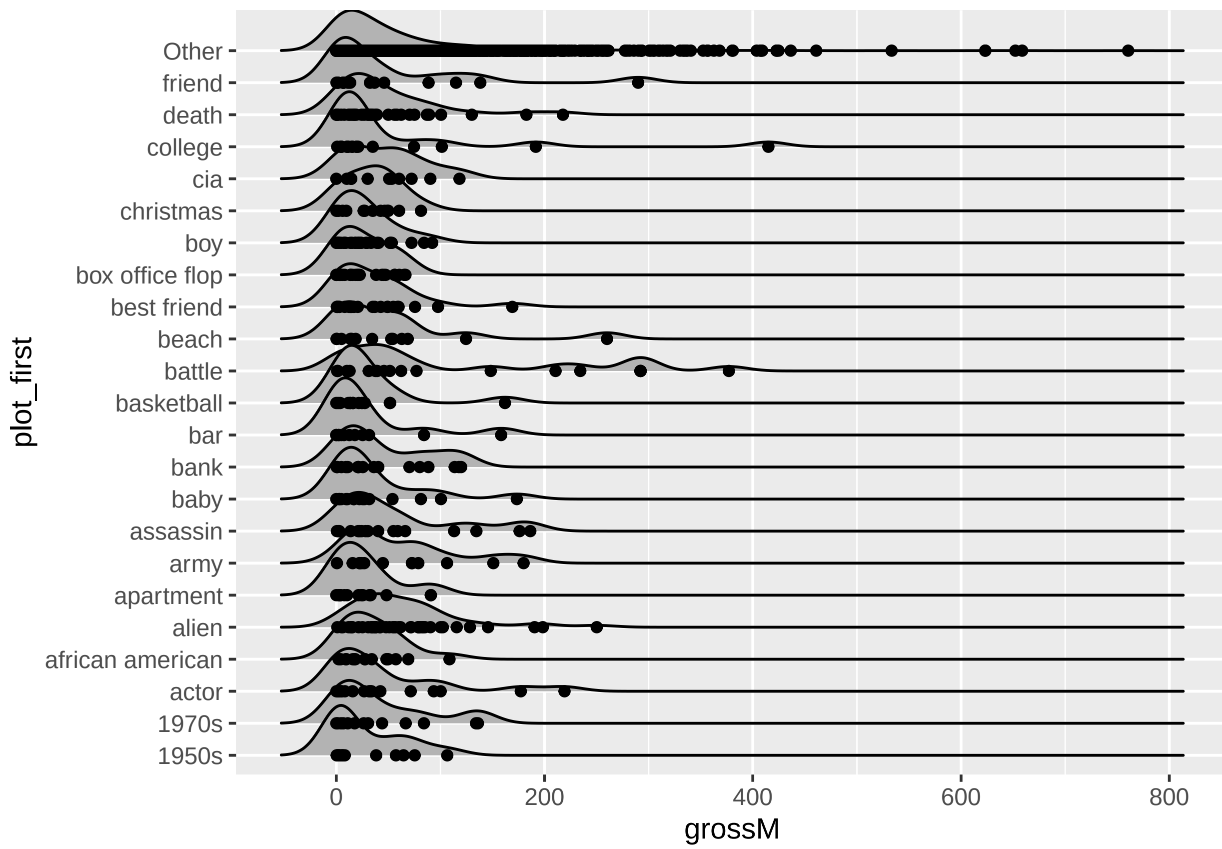
```r
library('ggridges')
```

```
##
## Attaching package: 'ggridges'

## The following object is masked from 'package:ggplot2':
##
##     scale_discrete_manual
```

```r
ridge_p <- ggplot(movies_train, aes(grossM, plot_first, )) + geom_density_ridges() + geom_point()
ridge_p
```

```
## Picking joint bandwidth of 17.6
```



* Plot keywords associated with the most blockbusters (>300M) are college, battle, and other!

c. Bagging model using 100 regression trees to predict grossM with every other variable. Bootstrap size = 2000

```r
B <- 100
num_b <- 2000

boot_mods <- list()
```

```r
train_preds <- movies_train %>% rownames_to_column() %>%
  mutate(rowname = as.numeric(rowname))


for(i in 1:B)
{

  boot_idx <- sample(1:nrow(movies_train),
                     size = num_b,
                     replace = FALSE)

  boot_tree <- ctree(grossM ~ .,
                     data = movies_train %>%
                       slice(boot_idx))

  boot_mods[[i]] <- boot_tree

  preds_boot <- data.frame(
    preds_boot = predict(boot_tree),
    rowname = boot_idx
  )

  names(preds_boot)[1] <- paste("preds_boot", i, sep = "")

  train_preds <- left_join(x = train_preds,
                           y = preds_boot,
                           by = "rowname")

}
```

d. Summarize across the 100 bags to generate average preds for each movie

```r
train_preds %<>% mutate(preds_bag =
                          select(., preds_boot1:preds_boot100) %>%
                          rowMeans(na.rm = TRUE))
```

e. R2, RMSE, and Mean Absolute Error

```r
R2(train_preds$preds_bag, movies_train$grossM)
```

```
## [1] 0.6295367
```

```r
RMSE(train_preds$preds_bag, movies_train$grossM)
```

```
## [1] 42.81847
```

```r
MAE(train_preds$preds_bag, movies_train$grossM)
```

```
## [1] 27.16414
```

- The model is not that great, having some pretty bad RMSE and MAE values for the grossM variable values. Also the R2 just passes the >0.6 threshhold which indicates the model is pretty decent, but could definitely be better.

f. Random Forest with 500 trees! Figure out mtry

```
rf_fit <- randomForest(grossM ~ .,
                       data = movies_train,
                       type = classification,
                       ntree = 500,
                       importance = TRUE,
                       localImp = TRUE)

rf_fit
```

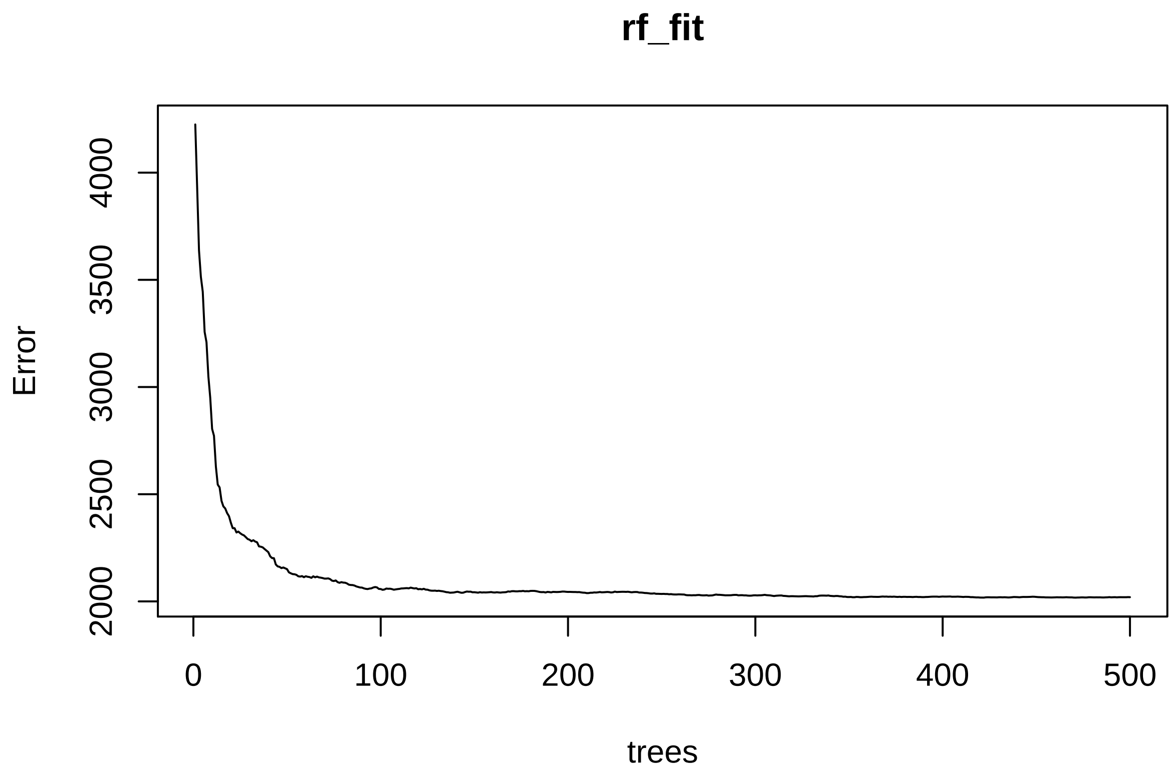
```
##
## Call:
##  randomForest(formula = grossM ~ ., data = movies_train, type = classification,      ntree = 500, imp
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 5
##
##           Mean of squared residuals: 2019.658
##                     % Var explained: 58.19
```

g. Why not mtry = sqrt(16)?

- The model without a set mtry parameter defaults to 5, and that makes sense. I ran it with 3, then 4, then 5, and 6. 5 seemed to explain the most variables with a google Mean of squared residuals. Also I believe we are supposed to round up from sqrt(16 or 17) = 4 to 5 as I think we covered in class? But yeah that is why I chose 5 as mtry. sqrt(variables) + 1. (which also is what the model defaulted to!)
- OH! And the model actually creates more columns to use in the model, which makes the number of variables used closest to 25, and so it rounds to sqrt(~25) = 5

h. How does the model improve with number of trees?

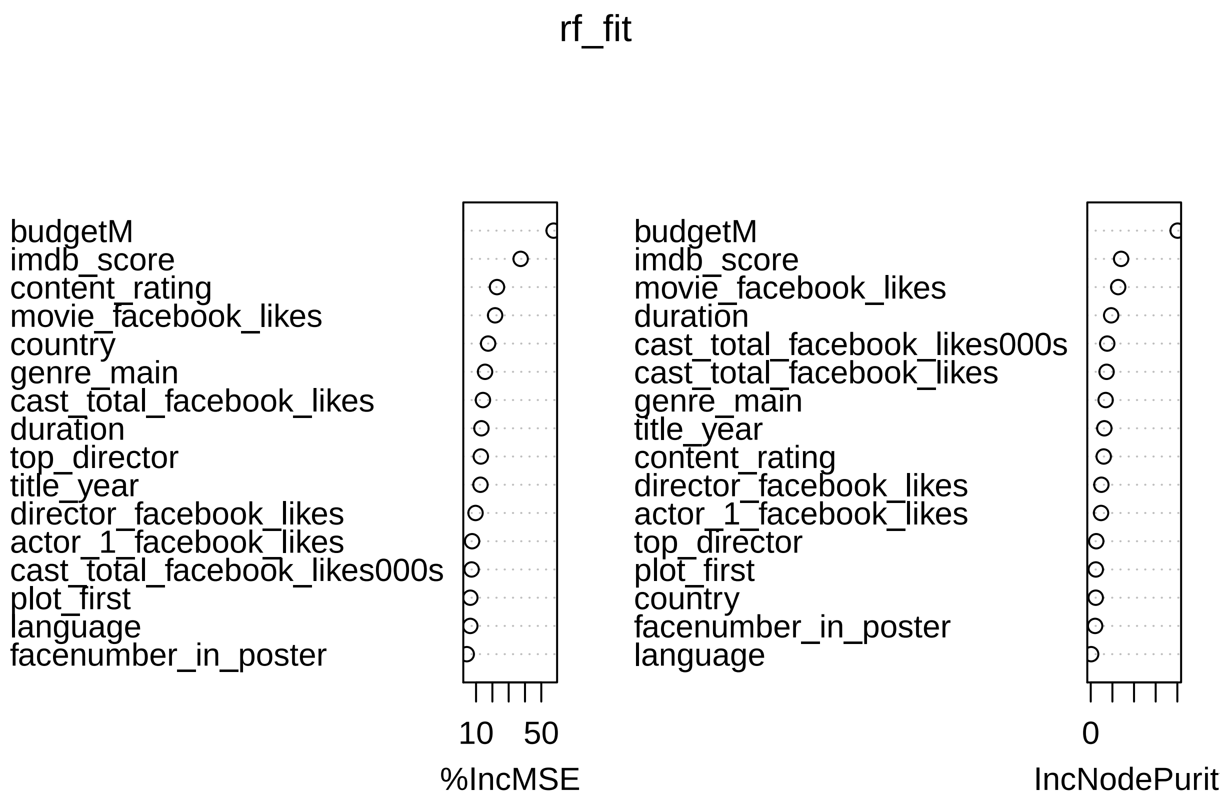
```
plot(rf_fit)
```



* The model cuts its error in half (from 4000 to 2000) when changing from using just around 1-15 to using ~100 trees. Then at 200 trees, the error stagnates at 2000ish. So I would use 200 trees to keep both error and processing time low.

i. Which variable are the most important?

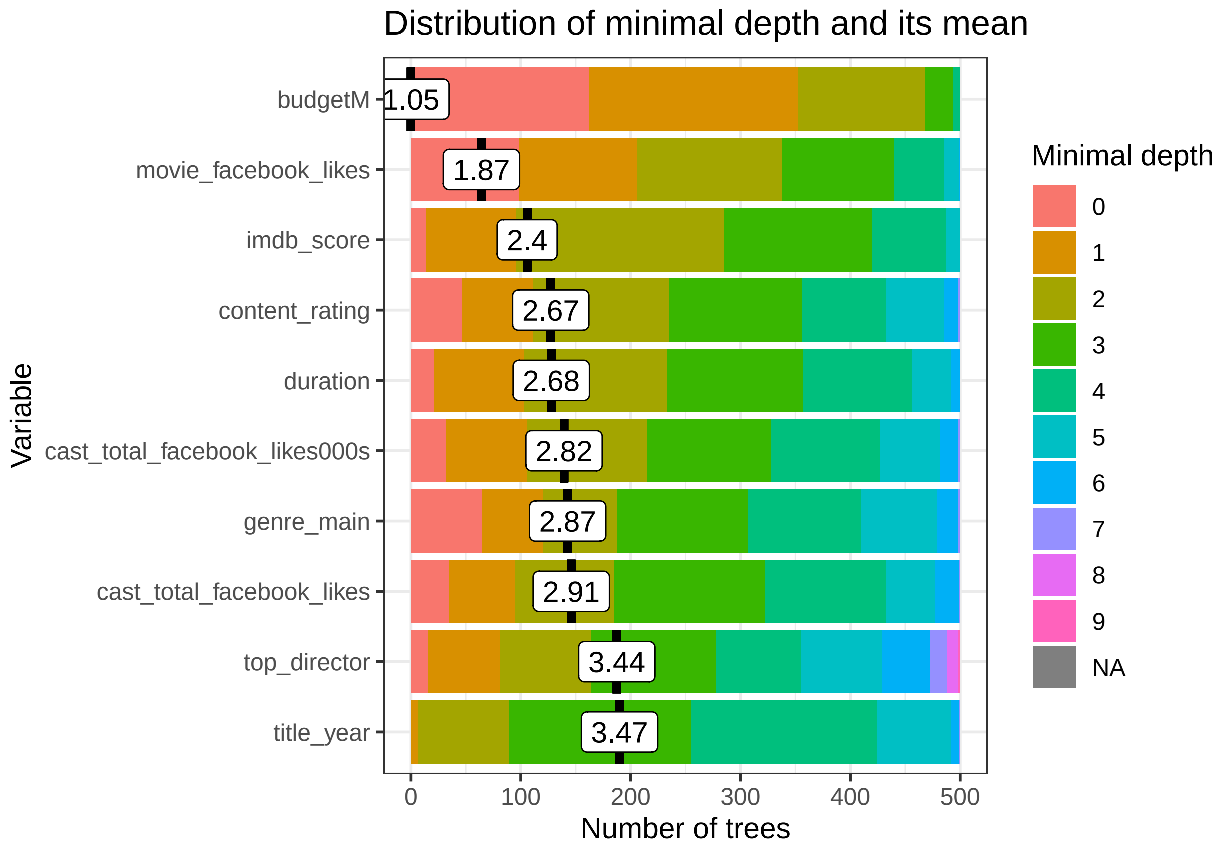
```r
varImpPlot(rf_fit)
```



* The top 5 most important variables are: budgetM, imdb_score, content_rating, movie_facebook_likes, and country!

j. Explore minimum depth by variable. How would I explain these findings to someone not well versed in machine learning?

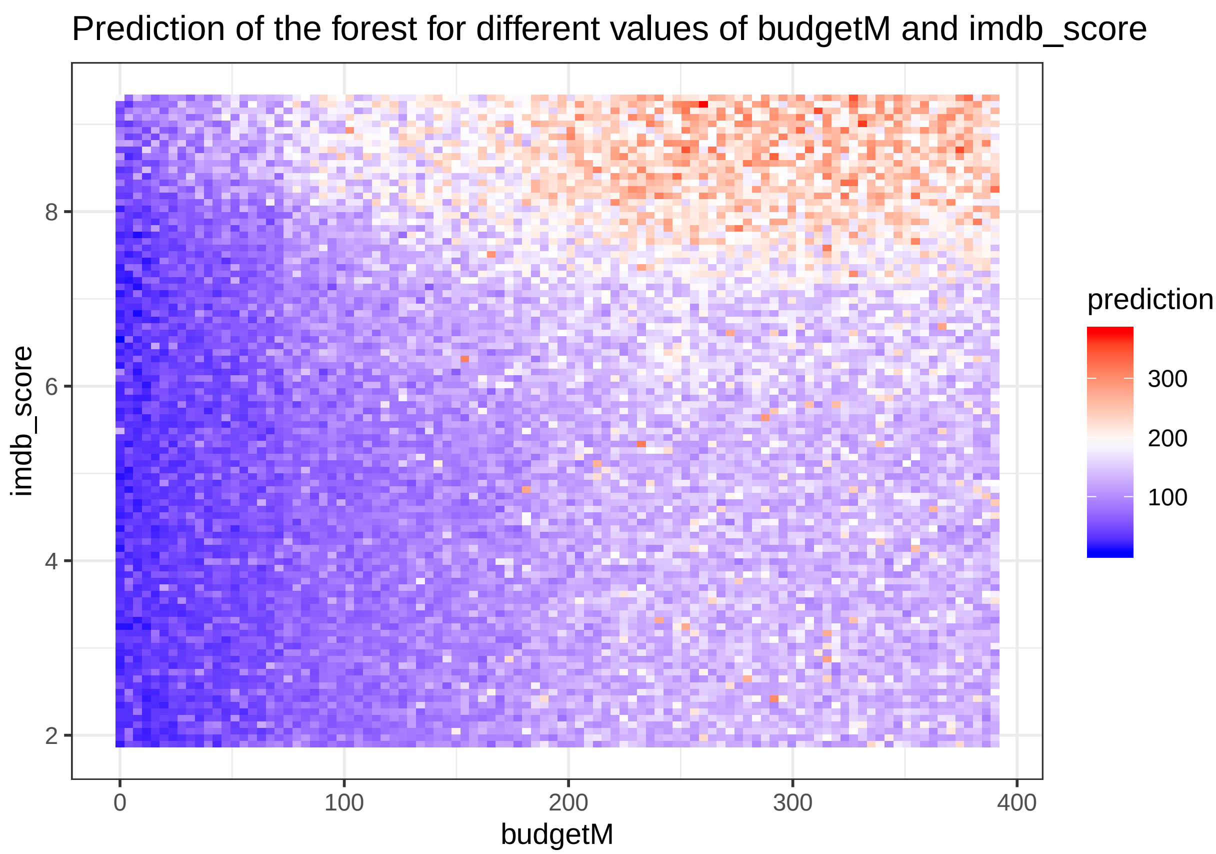
```
plot_min_depth_distribution(rf_fit)
```



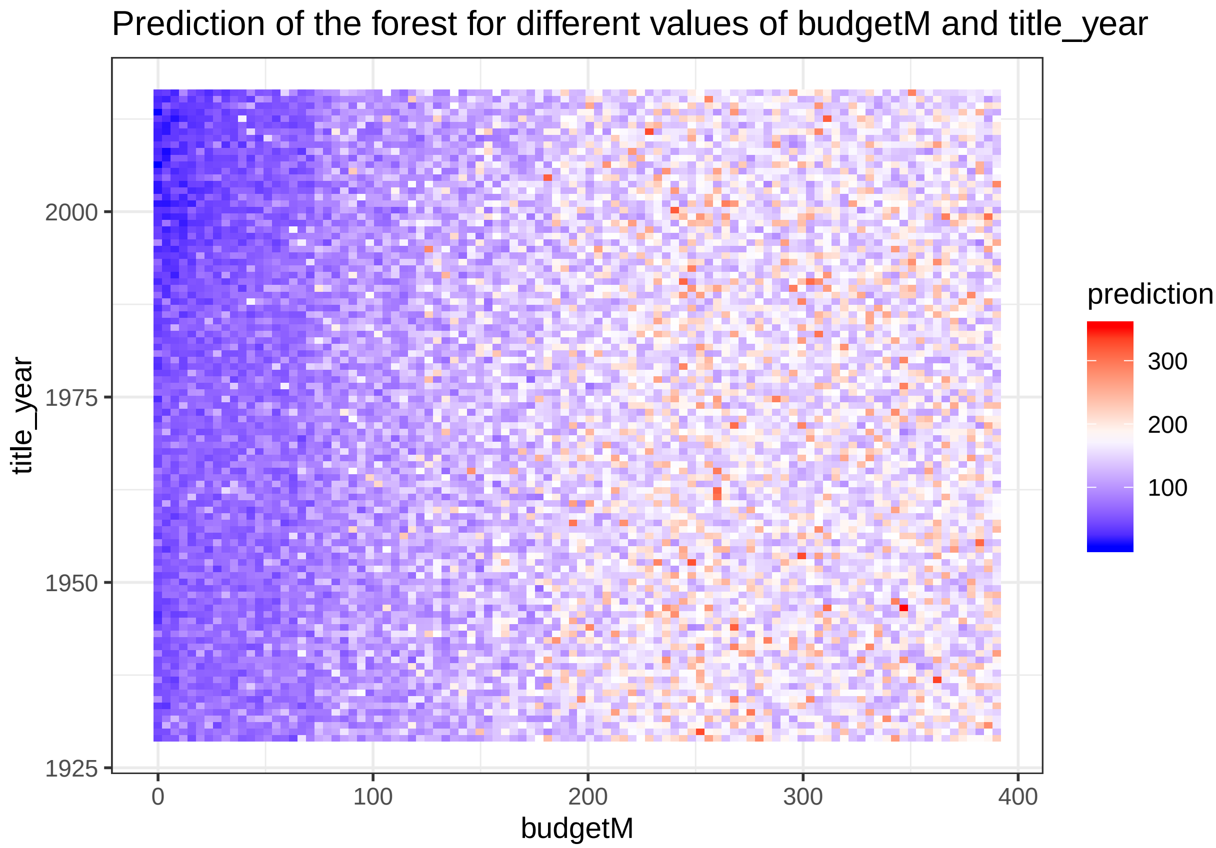
* FINALLY! That took a while. Graph looks sick tho. * This plot shows a bunch of iterations of decision trees. The respective variables are used to predict (within the decision trees) and in this graph we can see the average depth of each variable. When a variable is in a shallow depth, that indicates it is more important in deciding the prediction in our model. For example, budgetM has a avarage depth of 1.09, which means that it rarely is far down the decision tree. This shows that budgetM is important in predicting grossM. Variables farther down, like title_year, are less important, but still boast a high average depth of 3.61, compared to a lot of other variables - think of the depth as the importance to the model's prediction of that given variable.

k. Explore interactions between budgetM and imdb_score, and also budgetM and title_year

```
plot_predict_interaction(rf_fit, movies_train, "budgetM", "imdb_score")
```




```r
plot_predict_interaction(rf_fit, movies_train, "budgetM", "title_year")
```



* These plots show the prediction of grossM when looking at two variables. For budgetM and imdb_score, it is rare to get a great grossM with just a high imdb_score or just a high budgetM, what this plot shows is that it takes a combination of both to reach high grossM. After around 200M budget, the imdb_score definitely is a deciding factor for grossM, as where budgetM > 200, and imdb_score approaches 7, 8, 9, and 10, we start to see a grossM return of 250M+! * In a similar way, budgetM and title_year work together to predict grossM, but in this case we can clearly see that it is only budgetM that makes a significant impact on grossM. However, there is a trend where newer movies (~2000+) require at least a budget of ~50M to not preform super poorly (see the blue cluster in the top left). This trend is not seen as drastically in earlier years, although is can still be seen to a lesser degree.

1. Test preds, in-bag preds, out-of-bag preds. Patterns?!?!

```
# Test Predictions
movies_test_preds<- predict(rf_fit, newdata=movies_test)
R2(movies_test_preds, movies_test$grossM)
```

```
## [1] 0.6228651
```

```
RMSE(movies_test_preds, movies_test$grossM)
```

```
## [1] 45.41287
```

```
# In-Bag Predictions
R2(train_preds$preds_bag, movies_train$grossM)
```

```
## [1] 0.6295367
```

```r
RMSE(train_preds$preds_bag, movies_train$grossM)
```

```
## [1] 42.81847
```

```r
# Out-of-Bag Predictions
movies_oob_preds <- predict(rf_fit)
R2(movies_oob_preds, movies_train$grossM)
```

```
## [1] 0.5884627
```

```r
RMSE(movies_oob_preds, movies_train$grossM)
```

```
## [1] 44.94061
```

- Here we see that our random forest model preforms pretty equally across all three different types of predictions. Our test and out-of-bag predictions are a bit worse than the in-bag. That is probably because the OOB predictions and test predcitons are using new data to predict the model, whereas the in-bag predictions use its own data to make predictions in the model!