Movie Recommendations

Matthew Higgins

5/5/2019

Contents

1 Introduction

This project will work with the "Movielens 10M" dataset. Which, according to the creators;

> This data set contains 10000054 ratings and 95580 tags applied to 10681 movies by 71567 users of the online movie recommender service MovieLens.

The purpose of this project is to establish the best method to estimate the rating a user is likely to give a movie based on factors from the data. This will include some exploratory data analysis, improving the usability the data and testing a number of different approaches to prediction, before concluding which provides the lowest error using the Root Mean Square Error (RMSE) formula.

From some research, especially around the Netflix Prize, which aimed to find a solution to a similar problem, a target of less than `0.870` seems like a reasonable point to aim for.

After this optimal solution is found, it will be used in an R script which can be run independently of this report to create recommendations.

2 Creating the test and validation sets

Before getting into the report, the following code is taken directly from the project specification, and creates the `edx` (training) and `validation` (test) datasets ready for use.

```
##########################################################
# Create edx set, validation set, and submission file
##########################################################

# Note: this process could take a couple of minutes

if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")

# MovieLens 10M dataset:
# https://grouplens.org/datasets/movielens/10m/
# http://files.grouplens.org/datasets/movielens/ml-10m.zip

dl <- tempfile()
download.file("http://files.grouplens.org/datasets/movielens/ml-10m.zip", dl)

ratings <- read.table(text = gsub("::", "\t", readLines(unzip(dl, "ml-10M100K/ratings.dat"))),
                      col.names = c("userId", "movieId", "rating", "timestamp"))

movies <- str_split_fixed(readLines(unzip(dl, "ml-10M100K/movies.dat")), "\\::", 3)
colnames(movies) <- c("movieId", "title", "genres")
movies <- as.data.frame(movies) %>% mutate(movieId = as.numeric(levels(movieId))[movieId],
                                           title = as.character(title),
                                           genres = as.character(genres))

movielens <- left_join(ratings, movies, by = "movieId")

# Validation set will be 10% of MovieLens data

set.seed(1)
test_index <- createDataPartition(y = movielens$rating, times = 1, p = 0.1, list = FALSE)
edx <- movielens[-test_index,]
temp <- movielens[test_index,]

# Make sure userId and movieId in validation set are also in edx set

validation <- temp %>%
     semi_join(edx, by = "movieId") %>%
     semi_join(edx, by = "userId")

# Add rows removed from validation set back into edx set

removed <- anti_join(temp, validation)
edx <- rbind(edx, removed)

# To make testing super-speedy, let's take the first 10K results

# n <- 10000
# edx <- head(edx, n)
# validation <- head(edx, n)
```

```
# rm(n)

rm(dl, ratings, movies, test_index, temp, movielens, removed)
```

3 Exploring and cleaning the data

Before making any predictions, it's worth spending some time examining the data. The first step in exploring the data is taking a few rows to understand the format and the contents.

```
head(edx, 8)
```

	userId	movieId	rating	timestamp	title	genres
1	1	122	5	838985046	Boomerang (1992)	Comedy|Romance
2	1	185	5	838983525	Net, The (1995)	Action|Crime|Thriller
4	1	292	5	838983421	Outbreak (1995)	Action|Drama|Sci-Fi|Thriller
5	1	316	5	838983392	Stargate (1994)	Action|Adventure|Sci-Fi
6	1	329	5	838983392	Star Trek: Generations (1994)	Action|Adventure|Drama|Sci-Fi
7	1	355	5	838984474	Flintstones, The (1994)	Children|Comedy|Fantasy
8	1	356	5	838983653	Forrest Gump (1994)	Comedy|Drama|Romance|War
9	1	362	5	838984885	Jungle Book, The (1994)	Adventure|Children|Romance

Here user and movie IDs, as well as ratings, timestamps, a title column (which contains both the moie title and a year), and a list of genres can be seen. The next step will be to look at these in turn.

3.1 Dates

The `timestamp` column contains a UNIX timestamp (the number of seconds since Jnuary 1 1970), which can be converted to a more exploration-friendly date, using the following code;

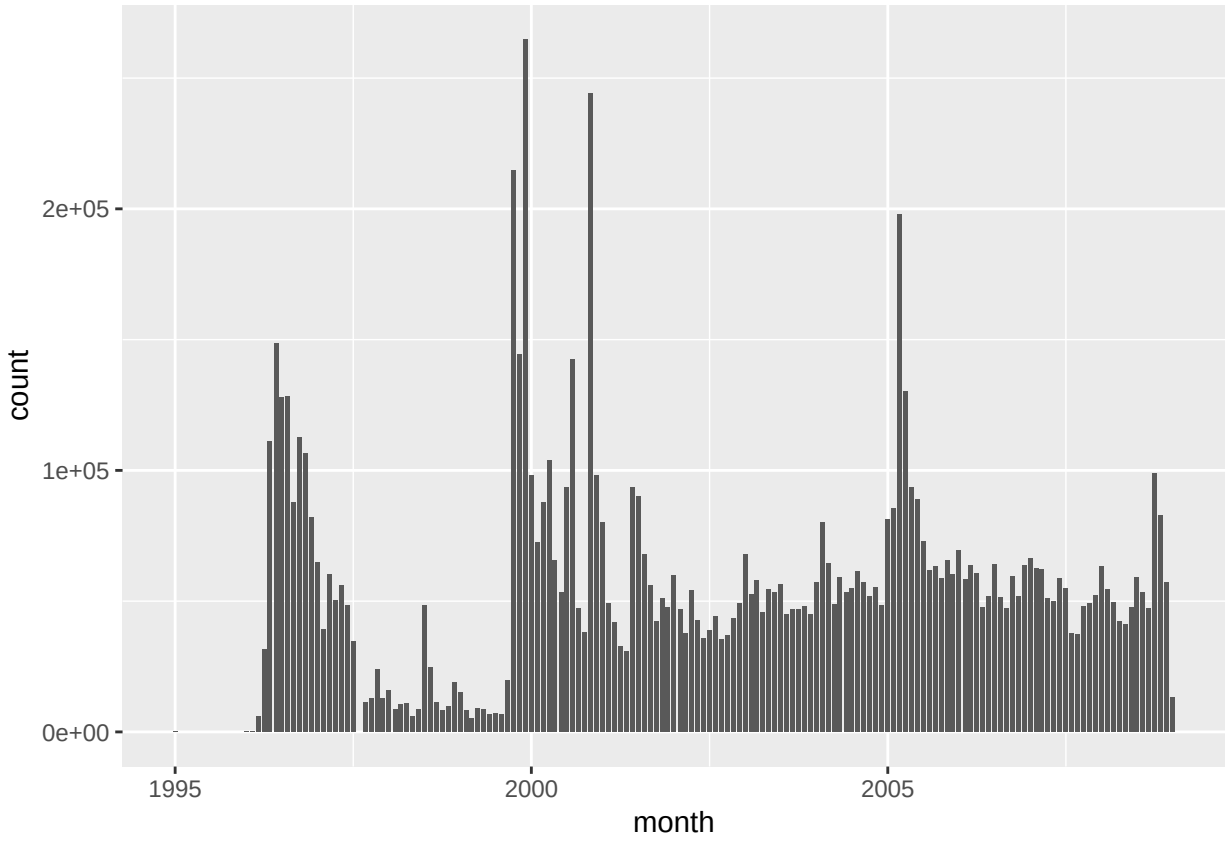
```
edx <- edx %>% mutate(datetime = as_datetime(timestamp))
head(edx$datetime)
```

```
## [1] "1996-08-02 11:24:06 UTC" "1996-08-02 10:58:45 UTC"
## [3] "1996-08-02 10:57:01 UTC" "1996-08-02 10:56:32 UTC"
## [5] "1996-08-02 10:56:32 UTC" "1996-08-02 11:14:34 UTC"
```

This has been added in the addition to the `timestamp` column, so the `edx` object now includes the additional column.

Using these dates, it's possible to graph how many ratings were added each month.

```
ratings_per_month <- edx %>%
                        group_by(month=floor_date(datetime, "month")) %>%
                        summarize(count = n())
ggplot(ratings_per_month, aes(x = month, y = count)) + geom_col()
```

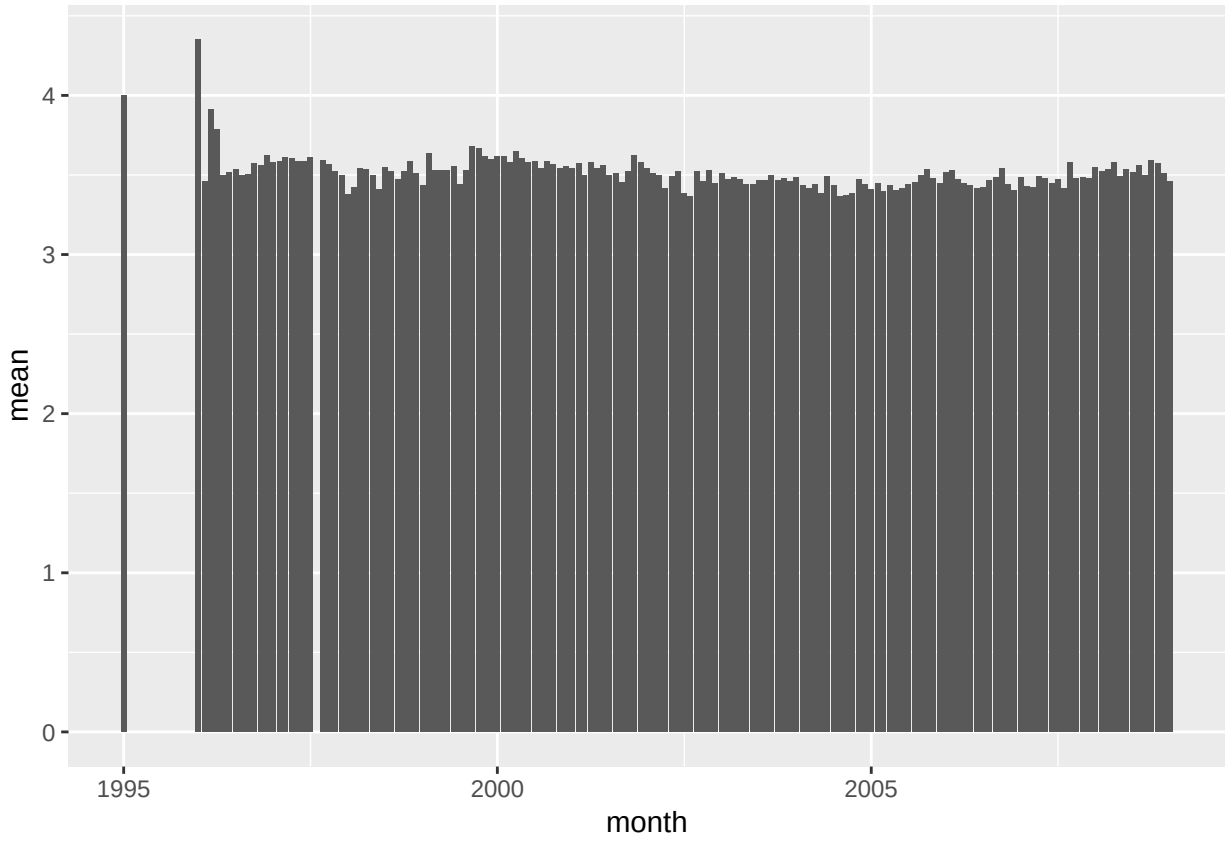



```
rm(ratings_per_month)
```

This shows some spikes, but from around the year 2000, steady use, so it seems unlikely that there will be any bias on the basis of the rating being especially old or new.

It's also possible to see whether the average rating given changes over time, for example, due to changes in societal attitudes to film critique.

```
average_per_month <- edx %>%
                      group_by(month=floor_date(datetime, "month")) %>%
                      summarize(mean = mean(rating))
ggplot(average_per_month, aes(x = month, y = mean)) + geom_col()
```



This doesn't seem to change much, so it seems unlikely that this could have an effect that is useful when predicting ratings.

Additionally, while it may have been possible to find a general trend, it's important to note that a recommendation engine wouldn't have this context, as it's predicting how a user might rate movies in the future, so exploring the date was always of limited value.

```
rm(average_per_month)
```

3.2 Users

Another column in the dataset is `userId`. As it contains numeric data, R has treated it as an `integer`, which isn't very useful, as it is really categorical, and the categories happen to be represented by numbers.

```
class(edx$userId)
```

```
## [1] "integer"
```

```
summary(edx$userId)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       1   18124   35738   35870   53607   71567
```

It can also be converted to a factor.

```
edx <- edx %>% mutate(userId = as.factor(userId))
```

Now we can count up how they are distributed.

```
users <- as_data_frame(table(edx$userId)) %>% arrange(desc(n))
head(users)
```

Var1	n
59269	6616
67385	6360
14463	4648
68259	4036
27468	4023
19635	3771

```
summary(users)
```

```
##      Var1                 n         
##  Length:69878       Min.   :  10.0  
##  Class :character   1st Qu.:  32.0  
##  Mode  :character   Median :  62.0  
##                     Mean   : 128.8  
##                     3rd Qu.: 141.0  
##                     Max.   :6616.0
```

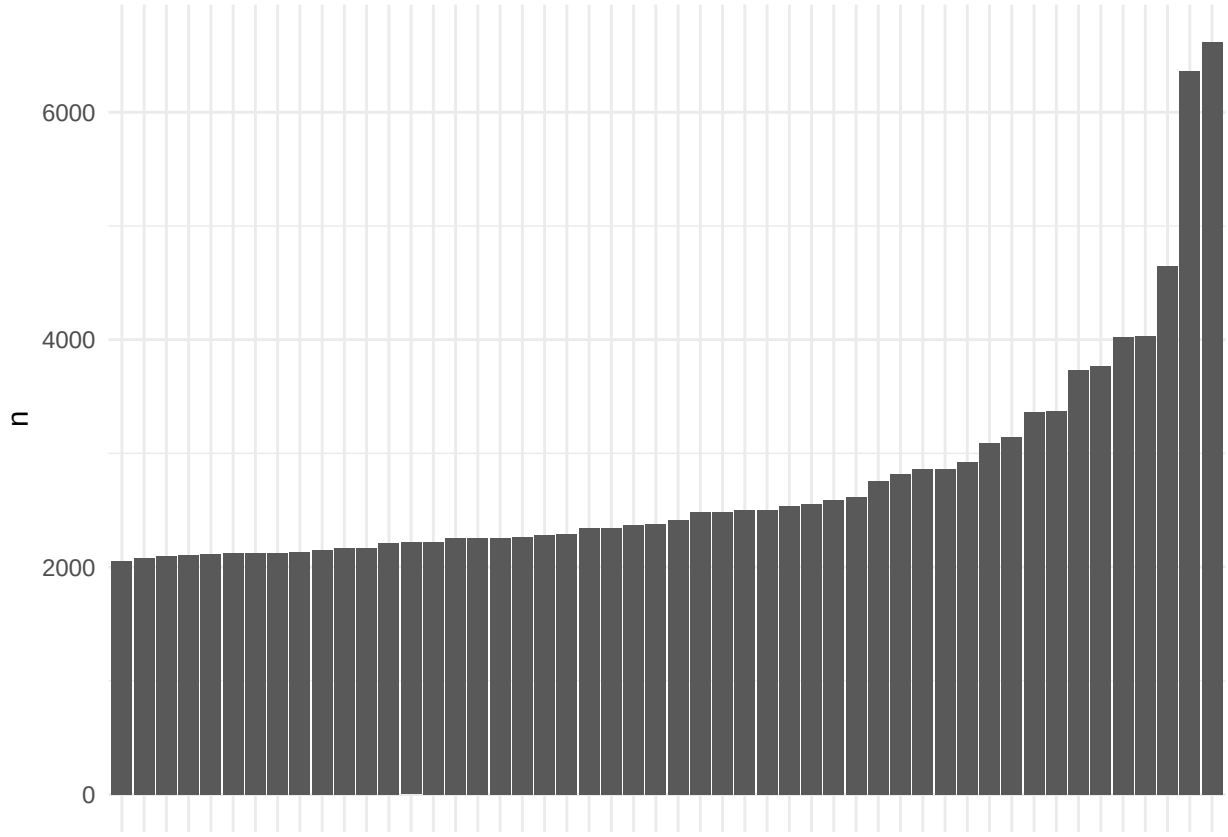
According to this table, two users have over 6000 movies rated, three have over 4000. But most, by far, have less than 100, the median is 62. The minimum is 10, so all users have multiple ratings.

The following chart shows this distribution clearly.

```
ggplot(users, aes(x = reorder(Var1, n), y = n)) +
  geom_col() +
  theme_minimal() +
  theme(axis.text.x=element_blank(), axis.title.x = element_blank())
```

As this is very this, and somewhat difficult to read, it is also possible to create a "zoomed in" view of just the top 50.

```
ggplot(top_n(users, 50), aes(x = reorder(Var1, n), y = n)) +
  geom_col() +
  theme_minimal() +
  theme(axis.text.x=element_blank(), axis.title.x = element_blank())
```



The following code finds the proportion of the ratings made by the 1% of most active users.

```
sum(top_n(users, (round(nrow(users) * 0.01)))$n) / nrow(edx)
```

```
## [1] 0.1062188
```

So around 10% of ratings come from the most active 1% of users.

The following provides a view of the proportion of users by the average ratings they give movies.

```
user_average_ratings <- edx %>%
                          group_by(userId) %>%
                          summarize(average_rating = mean(rating))
ggplot(user_average_ratings, aes(x = reorder(userId, average_rating), y = average_rating)) +
  geom_col() +
  theme_minimal() +
  theme(axis.text.x=element_blank(), axis.title.x = element_blank())
```

In summary, the findings from the users column;

- There are nearly 70,000 users
- The median movies rated by a user are 62
- The top 1% of users accounted for 10% of the ratings
- The lower quartile is 32 ratings, so 75% of users have rated at least 32 movies
- Users have individual average ratings that span the full range of ratings, but are mostly quite consistent, with an interquartile range of 0.546

Overall, it appears that based on this level of activity users have, the user's own history, how many ratings they have made and how positive they generally are will be important factors for the predictions.

```
rm(users, user_average_ratings)
```

3.3 Movies

The data also contain a column called `movieId`, lets take a look at how these are stored and distributed.

```
class(edx$movieId)
```

```
## [1] "numeric"
```

```
summary(edx$movieId)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       1     648    1834    4122    3626   65133
```

These have the same issues as users, namely that categorical data is being represented as numeric data, so can also be conveted into a factor.

```
edx$movieId <- as.factor(edx$movieId)
```

The data also contain a column including the totle of the movie. These can be used to count the frequency with which different movies appear in the results.

```
movies <- as_data_frame(table(edx$title)) %>% arrange(desc(n))
head(movies)
```

Var1	n
Pulp Fiction (1994)	31362
Forrest Gump (1994)	31079
Silence of the Lambs, The (1991)	30382
Jurassic Park (1993)	29360
Shawshank Redemption, The (1994)	28015
Braveheart (1995)	26212

```
summary(movies$n)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     1.0    30.0   122.0   843.0   565.2 31362.0
```

```
nrow(movies)
```

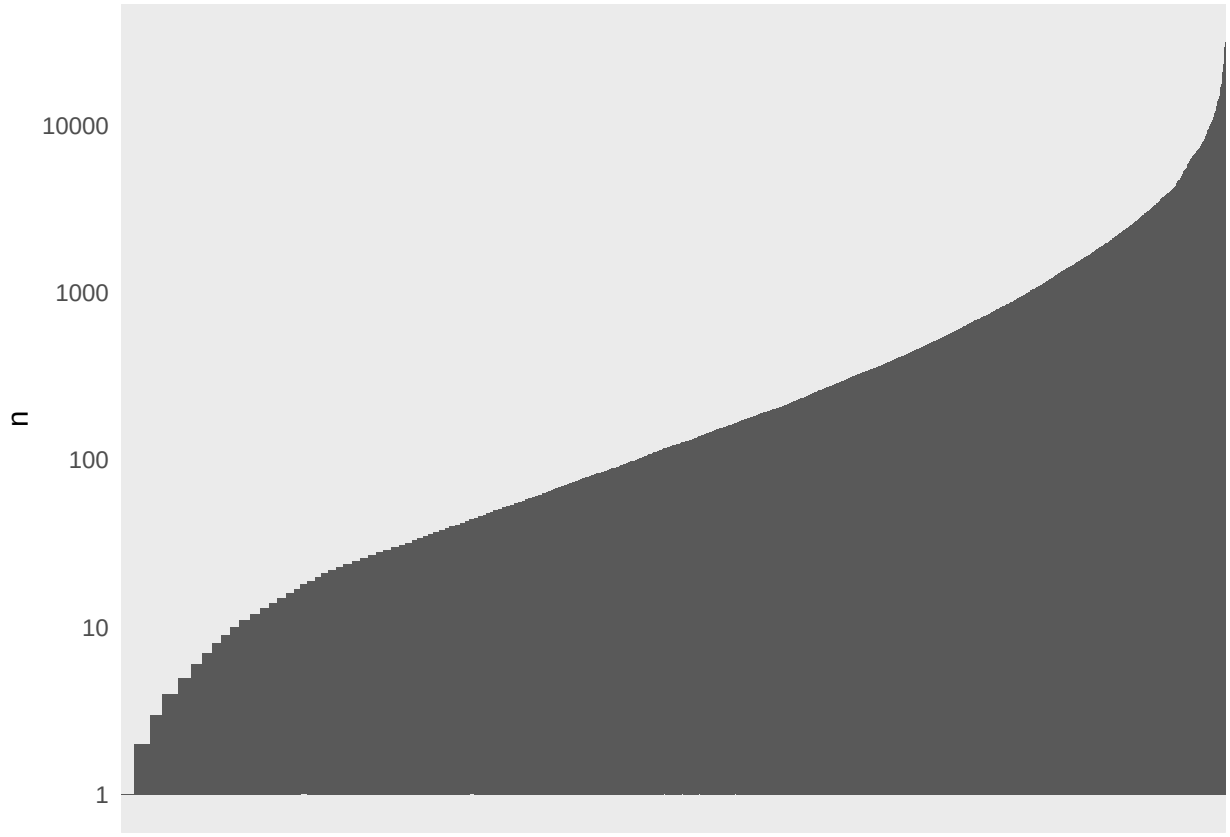
```
## [1] 10676
```

This shows that some movies are rated more than 30 000 times. The most-rated movies are all very well-known. The mean of the counts is 842.9, and the median is 122.

```
ggplot(movies, aes(x = reorder(Var1, n), y = n)) +
  geom_col() +
  theme_minimal() +
  theme(axis.text.x=element_blank(), axis.title.x = element_blank())
```

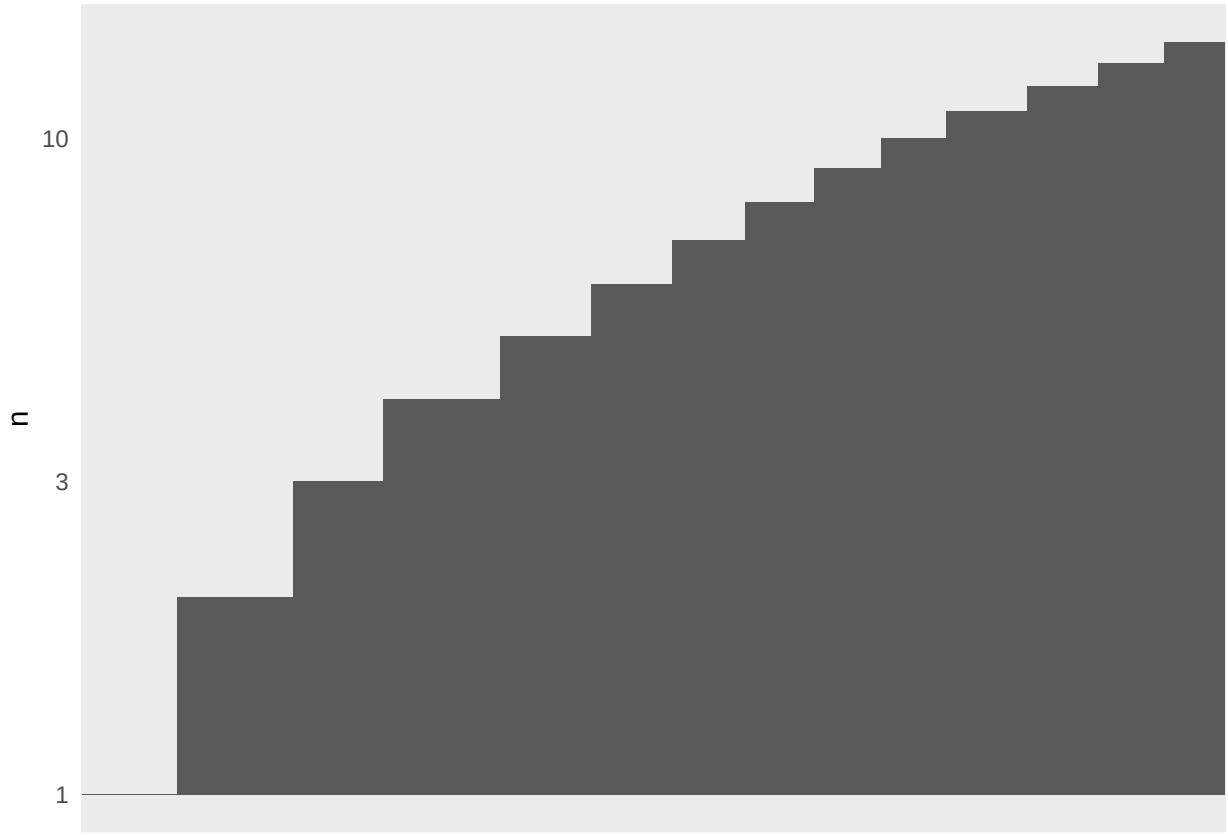
This shows that many movies have only a few ratings. Let's try this again with a log scale to see if it's easier to read.

```
ggplot(movies, aes(x = reorder(Var1, n), y = n)) +
  geom_col() +
  scale_y_log10() +
  theme_minimal() +
  theme(axis.text.x=element_blank(), axis.title.x = element_blank())
```



So, quite a few movies are at the low end of the scale, but most have over 100 ratings. Now, to "zoom in" on the low end.

```
ggplot(top_n(movies, 1500, -n), aes(x = reorder(Var1, n), y = n)) +
  geom_col() +
  scale_y_log10() +
  theme_minimal() +
  theme(axis.text.x=element_blank(), axis.title.x = element_blank())
```



And lets see some of the least-frequently-rated movies.

```
tail(movies)
```

Var1	n
Vinci (2004)	1
When Time Ran Out… (a.k.a. The Day the World Ended) (1980)	1
Where A Good Man Goes (Joi gin a long) (1999)	1
Won't Anybody Listen? (2000)	1
Young Unknowns, The (2000)	1
Zona Zamfirova (2002)	1

So, in summary;

- The dataset contains just over 10,000 movies.
- Each movie has been rated 1-31,362 times.
- Most movies are rated between 100 and 10,000 times.

3.4 Movie Production Years

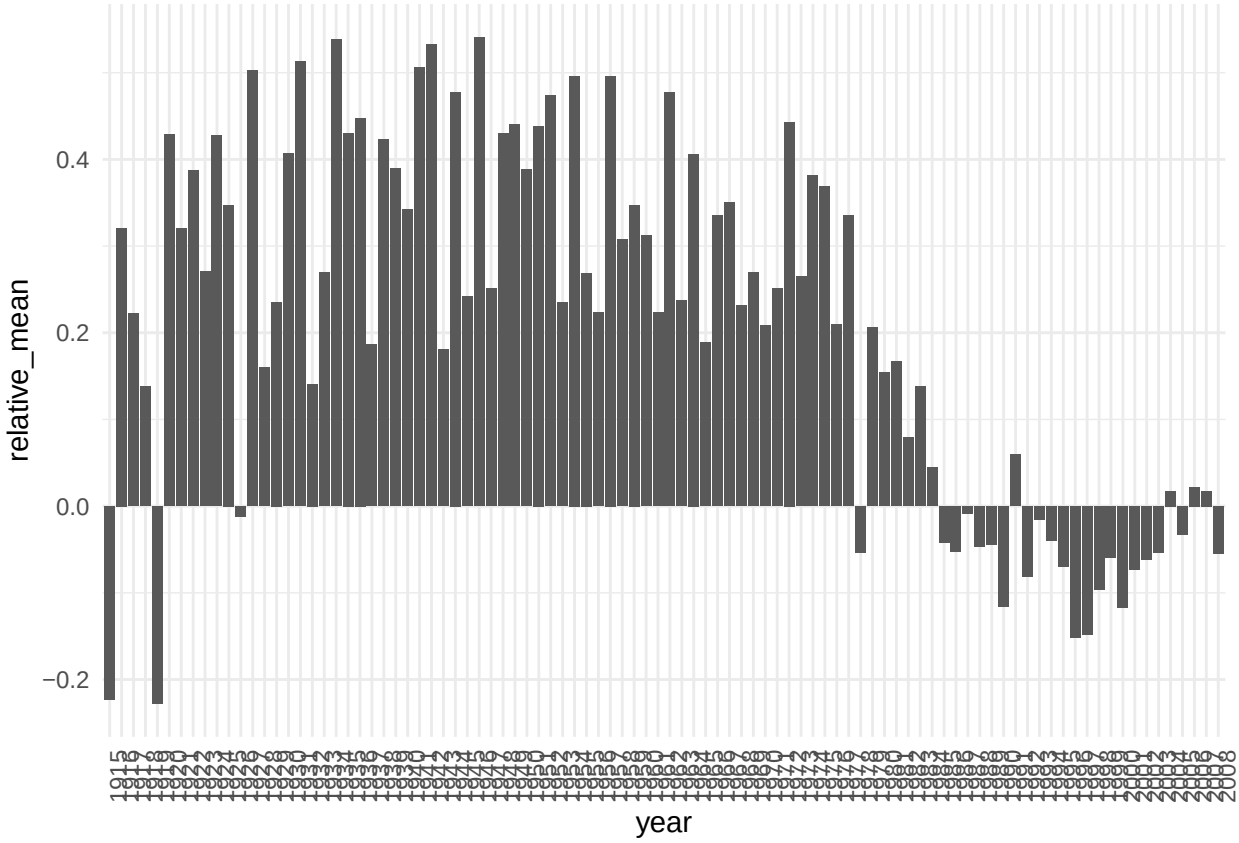
The `title` column is in the format of `[title] ([year])`. The year can be copied into a separate column using a regular expression.

```
years_in_brackets <- str_extract(edx$title, "\\(\\d{4}\\)$")
years_without_brackets <- str_remove_all(years_in_brackets, "[()]")
years_as_factor <- as.factor(years_without_brackets)
```

```
edx$year <- years_as_factor
rm(years_in_brackets, years_without_brackets, years_as_factor)
```

Now it's possible to visualise how these affect the average rating.

```
overall_average <- mean(edx$rating)
years <- edx %>%
          group_by(year) %>%
          summarize(mean = mean(rating), relative_mean = mean - overall_average, count = n())
ggplot(years, aes(x = year, y = relative_mean)) +
  geom_col() +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```



This provides a small change, but it seems to diminish in more recent years. It can, however, be included in the further explorations, as it may hold some predictive value.

3.5 Distribution of Ratings

The data also contain a `rating`. This will be the "target" column.

According to the README file from the dataset;

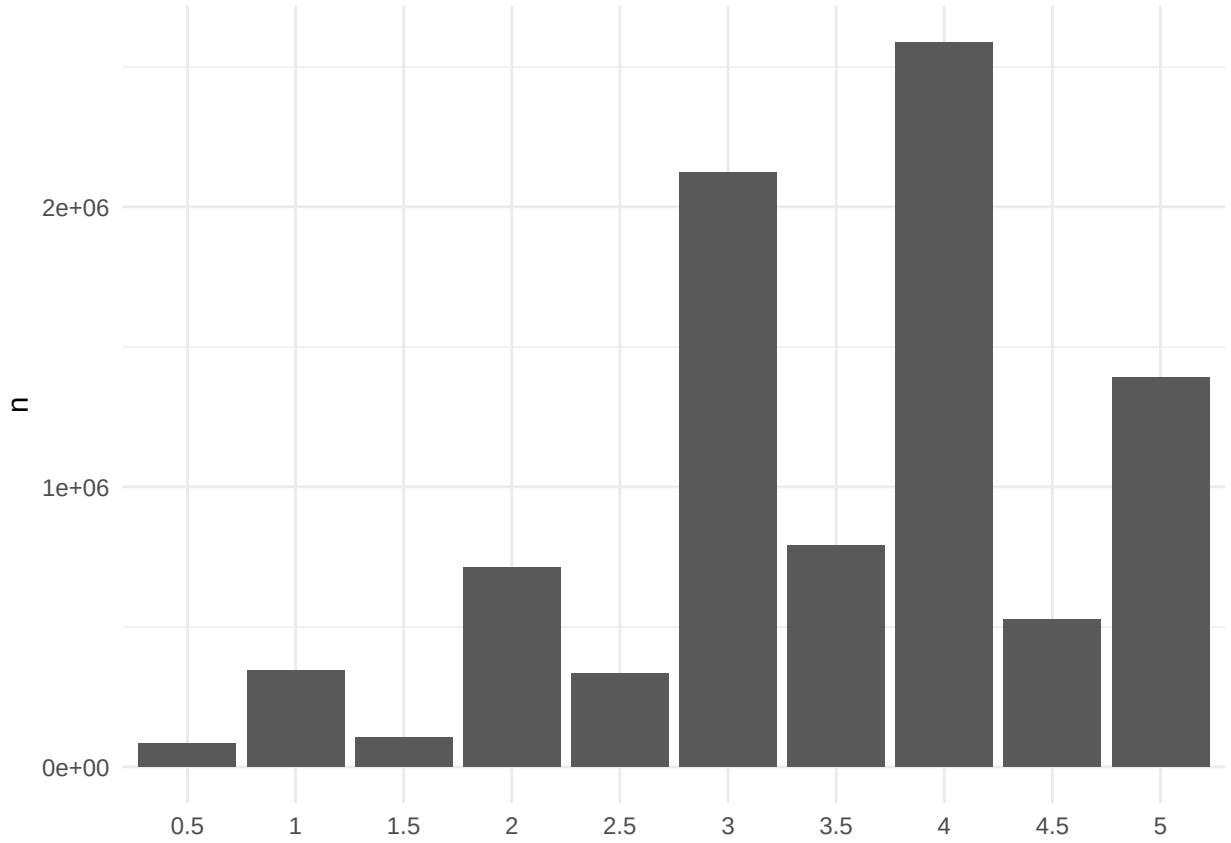
> Ratings are made on a 5-star scale, with half-star increments.

```
summary(edx$rating)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.500   3.000   4.000   3.512   4.000   5.000
```

The proportions can be plotted.

```
ratings <- as_data_frame(table(edx$rating)) %>% arrange(desc(n))
ggplot(ratings, aes(x = Var1, y = n)) +
  geom_col() +
  theme_minimal() +
  theme(axis.title.x = element_blank())
```



Half-star ratings seem less popular then whole-star ratings.

Ratings are numeric, and can be left that way. While they are provided in categories (intervals of 0.5), it makes more sense to predict the value wherever it falls, than a specific category a user can rate.

For example, when recommending movies, it makes sense to show one where they user is predicted to rate `4.9586` more prominently than one the user is expected to rate `4.7501`, even though both would round to `5`.

The following can be observed from the data;

- Whole number ratings are more common than their nearest half ratings.
- On average, users would seem to rate more movies they like than movies they don't.

3.6 Ratings per Genre

The genres are strings, containing the names of one or more genre, separated with a `|` (pipe character).

Splitting them into single genres is more challenging than some of the work with other columns, it involves splitting the strings into their unique values, then, later using the unique values to test for their presence.

This method was chosen as it was the fastest performing, and genres will be handled rowwise, but there are alternatives, for example, breaking the column into several boolean columns.

It's worth noting that this would not work if a genre included another. For example if the data included both `Romantic Comedy` and `Comedy`, a different approach would be needed. Luckily, this isn't an issue in these data.

First, get a complete list of every genre that's included.

```
genres_list <- unique(unlist(strsplit(edx$genres, "|", fixed=TRUE)))
genres_list <- data.frame(name = genres_list)
genres_list
```

name
Comedy
Romance
Action
Crime
Thriller
Drama
Sci-Fi
Adventure
Children
Fantasy
War
Animation
Musical
Western
Mystery
Film-Noir
Horror
Documentary
IMAX
(no genres listed)

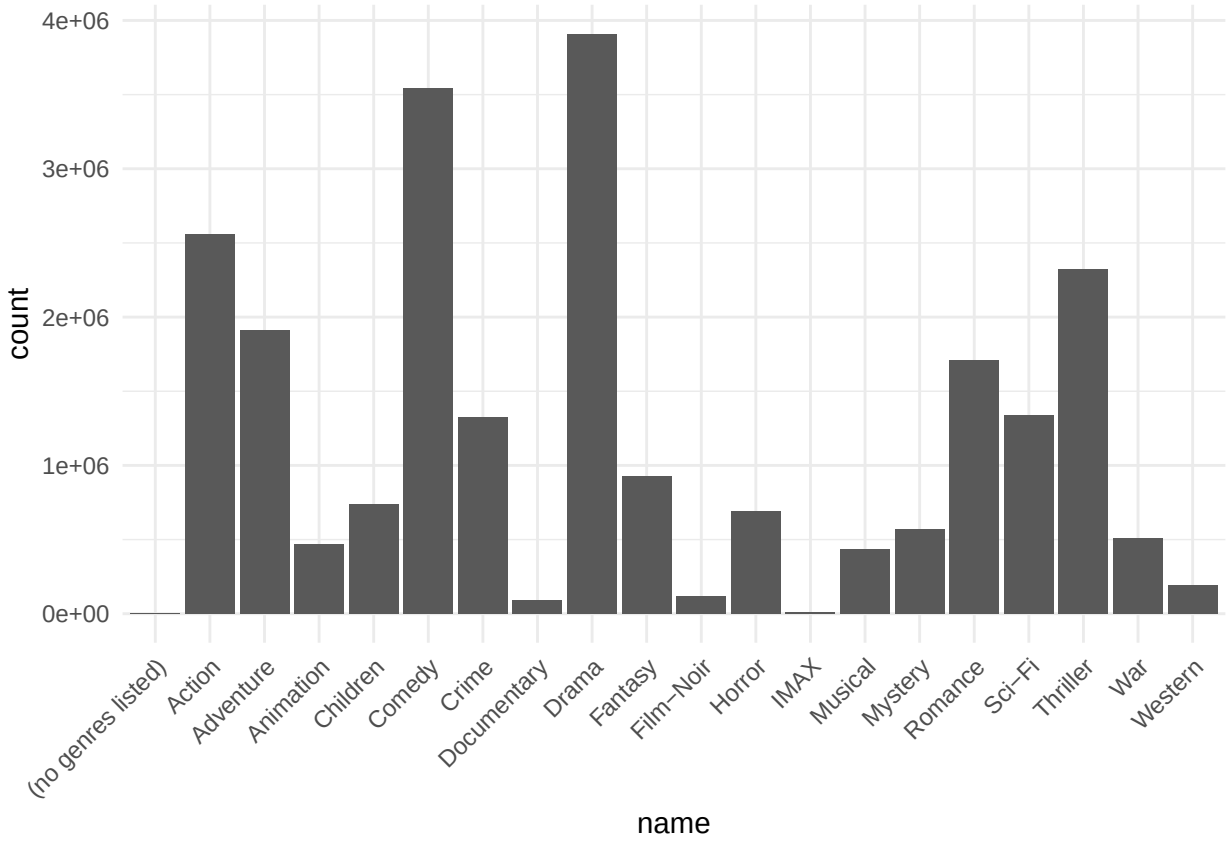
The number of movies including each can be counted.

```
includes_genre <- function(genre_list, genre) grepl(genre_list, genre)
genres_list <- genres_list %>%
                rowwise() %>%
                mutate(count = sum(unlist(lapply(edx$genres, includes_genre, name))))
row.names(genres_list) <- genres_list$name
genres_list <- genres_list %>% as.tibble(rownames = genres_list$name)
head(genres_list)
```

name	count
Comedy	3540930
Romance	1712100
Action	2560545
Crime	1327715
Thriller	2325899
Drama	3910127

Now these can be plotted on a bar chart.

```r
ggplot(genres_list, aes(x = name, y = count)) +
  geom_col() +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



Some final notes about the genres;

- Seven movies have no listed genres - these could be hard to work with if genre is significant, but they represent a tiny proportion of the data.
- A small number of movies are listed as `IMAX`. It will be interesting to see if this has an impact, as it is not a genre in the conventional sense, but may correlate with certain other genres.

3.7 Genre-Specific Ratings

The impact that the present genres have on the movies can be found. It's important to note that this average can't take account of combinations (maybe audiences more often find combinations like `Comedy|Romance` more enjoyable than, say, `Comedy|War`), but only of single genres.

```r
overall_average <- mean(edx$rating)
genres_list[,"average_rating"] <- NA
for(i in 1:nrow(genres_list)){
  matching_ratings <- edx %>%
                      rowwise() %>%
                      filter(includes_genre(genres, genres_list[[i, 'name']]))
  new_value <- mean(matching_ratings$rating)
  genres_list$average_rating[[i]] <- new_value
}
```

```
genres_list <- genres_list %>%
                 mutate(effect = average_rating - overall_average) %>%
                 as.tibble(rownames = genres_list$name)
row.names(genres_list) <- genres_list$name

genres_list
```

	name	count	average_rating	effect
Comedy	Comedy	3540930	3.436908	-0.0755571
Romance	Romance	1712100	3.553813	0.0413482
Action	Action	2560545	3.421405	-0.0910606
Crime	Crime	1327715	3.665925	0.1534601
Thriller	Thriller	2325899	3.507676	-0.0047895
Drama	Drama	3910127	3.673131	0.1606655
Sci-Fi	Sci-Fi	1341183	3.395743	-0.1167220
Adventure	Adventure	1908892	3.493544	-0.0189213
Children	Children	737994	3.418715	-0.0937497
Fantasy	Fantasy	925637	3.501946	-0.0105190
War	War	511147	3.780813	0.2683474
Animation	Animation	467168	3.600644	0.0881785
Musical	Musical	433080	3.563305	0.0508395
Western	Western	189394	3.555918	0.0434526
Mystery	Mystery	568332	3.677001	0.1645361
Film-Noir	Film-Noir	118541	4.011625	0.4991595
Horror	Horror	691485	3.269815	-0.2426502
Documentary	Documentary	93066	3.783487	0.2710218
IMAX	IMAX	8181	3.767693	0.2552282
(no genres listed)	(no genres listed)	7	3.642857	0.1303919

- Genres have a small effect on the average rating in this dataset, with some causing up to a `0.25` change in the average rating.

4 Testing Prediction Methods

4.1 RMSE Calculation

The `calculate_rmse` function will be used to calculate the Root Mean Squared Error (RMSE) from the predicted values (the `ratings` as a list) and the corresponding true values (in this case, the ratings from the `validation` set).

```
calculate_rmse <- function(predictions, actual_values){
  sqrt(mean((actual_values - predictions) ^ 2))
}
```

To double-check that it is working correctly, it is passed the same values for predictions`and`actual_values‘, and as there is no error between them, the RMSE will be zero.

```
calculate_rmse(edx$rating, edx$rating)
```

```
## [1] 0
```

This seems to be working correctly.

4.2 Baseline RMSE

To see how well different predictive approaches can work, the mean of all of the ratings gives a realistic possible rating which is likely to have lower error than other approaches (like using random numbers, for example), so this makes a good baseline against which to compare the RMSE values of different methods.

```
mean_rating = mean(edx$rating)
print(mean_rating)
```

```
## [1] 3.512465
```

And the RMSE of using the mean for every prediction;

```
rmse <- calculate_rmse(rep(mean_rating, nrow(validation)), validation$rating)
```

At this point, the results will be stored in a data frame;

```
peak_rmse_results <- data.frame(method = c("Mean Rating"),
                                rmse = c(rmse),
                                stringsAsFactors = FALSE)
peak_rmse_results
```

method	rmse
Mean Rating	1.061202

As this is over 1, it is, as expected, a poor way of predicting ratings.

4.3 Single Predictors

During the data exploration, each potential predictor was explored to establish potential predictive value.

Now, each potentially useful predictor can be examined, and the RMSE when it is used as the only predictor will be compared.

Again, a data frame will be used to store the values being compared.

```
rmse_results <- data.frame(method = c("Mean Rating"),
                           rmse = c(rmse),
                           stringsAsFactors = FALSE)
rm(rmse)
```

4.3.1 The movie effect

This method considers only the movie, and predicts that the user will rate each movie close to the average of the ratings other users gave;

```
movie_effects <- edx %>%
                  group_by(movieId) %>%
                  summarize(movie_effect = mean(rating - mean_rating),
                            ratings_for_movie = n())
predictions <- validation %>%
               mutate(movieId = as.factor(movieId)) %>%
               left_join(movie_effects, by='movieId') %>%
               mutate(prediction = (mean_rating + movie_effect)) %>%
               .$prediction
rmse <- print(calculate_rmse(predictions, validation$rating))
```

```
## [1] 0.9439087
```

```
rmse_results[nrow(rmse_results) + 1,] <- list("Mean + Movie Effect",rmse)
rm(rmse, predictions)
rmse_results
```

method	rmse
Mean Rating	1.0612018
Mean + Movie Effect	0.9439087

4.3.2 The user effect

This method considers only the ratings the user has given, and predicts that the user will rate the movie with the average of ratings that they have given other movies.

```
user_effects <- edx %>%
                  group_by(userId) %>%
                  summarize(user_effect = mean(rating - mean_rating),
                            ratings_by_user = n())
predictions <- validation %>%
                 mutate(userId = as.factor(userId)) %>%
                 left_join(user_effects, by='userId') %>%
                 mutate(prediction = (mean_rating + user_effect)) %>%
                 .$prediction
rmse <- calculate_rmse(predictions, validation$rating)
rmse_results[nrow(rmse_results) + 1,] <- list("Mean + User Effect",rmse)
rm(rmse, predictions)
rmse_results
```

method	rmse
Mean Rating	1.0612018
Mean + Movie Effect	0.9439087
Mean + User Effect	0.9783360

4.3.3 The year effect

This method looks at movies made in the same year, and predicts that the user will rate the movie close to the average of the ratings given to other movies made the same year.

```
year_effects <- edx %>%
                  group_by(year) %>%
                  summarize(year_effect = mean(rating - mean_rating))
predictions <- validation %>%
                 mutate(movieId = as.factor(movieId), userId = as.factor(userId)) %>%
                 mutate(year = as.factor(str_remove_all(str_extract(title,
                                                                    "\\(\\d{4}\\)$"), "[()]"))) %>%
                 left_join(year_effects, by='year') %>%
                 mutate(prediction = (mean_rating + year_effect)) %>%
                 .$prediction
rmse <- print(calculate_rmse(predictions, validation$rating))
```

```
## [1] 1.050026
```

```
rmse_results[nrow(rmse_results) + 1,] <- list("Mean + Year Effect",rmse)
rm(rmse, predictions)
```

```
rmse_results
```

method	rmse
Mean Rating	1.0612018
Mean + Movie Effect	0.9439087
Mean + User Effect	0.9783360
Mean + Year Effect	1.0500259

4.3.4 The “genres combination” effect"

Next, the string containing all of the genres can be explored.

This, awkwardly, relies on there being a single combination of genres that has an effect, when in reality, the genres likely each have different effects.

```
genre_combination_effects <- edx %>%
                                group_by(genres) %>%
                                summarize(genre_combination_effect = mean(rating - mean_rating))
predictions <- validation %>%
                left_join(genre_combination_effects, by='genres') %>%
                mutate(prediction = (mean_rating + genre_combination_effect)) %>%
                .$prediction
rmse <-calculate_rmse(predictions, validation$rating)
rmse_results[nrow(rmse_results) + 1,] <- list("Mean + Combined Genres Effect",rmse)
rm(rmse, predictions)
rmse_results
```

method	rmse
Mean Rating	1.0612018
Mean + Movie Effect	0.9439087
Mean + User Effect	0.9783360
Mean + Year Effect	1.0500259
Mean + Combined Genres Effect	1.0184056

4.3.5 Sum of single genre effects

Next, a new function that calculates a combined effect of the movie’s listed genres can be used. This will be called `calculate_genre_effect_sum`.

As well as creating the function, it can be tested, passing it a string containing a `|`-separated list of genres, and it should return the effect.

```
calculate_genre_effect_sum <- function(genres_string) {
  genres_as_list <- strsplit(genres_string, "|", fixed=TRUE)
  single_genre_effects <- map(genres_as_list, function(genre_name) { genres_list[genre_name, 'effect'] }
  do.call(sum, single_genre_effects)
}
calculate_genre_effect_sum("Comedy|Drama")
```

```
## [1] 0.08510839
```

Now to measure whether it improves the predictions, and by how much.

```
predictions <- validation %>%
                rowwise() %>%
                mutate(genres_effect = calculate_genre_effect_sum(genres)) %>%
                ungroup() %>%
                mutate(prediction = (mean_rating + genres_effect)) %>%
                .$prediction
rmse <-calculate_rmse(predictions, validation$rating)
rmse_results[nrow(rmse_results) + 1,] <- list("Mean + Sum of Single Genre Effects",rmse)
rm(rmse, predictions)
rmse_results
```

method	rmse
Mean Rating	1.0612018
Mean + Movie Effect	0.9439087
Mean + User Effect	0.9783360
Mean + Year Effect	1.0500259
Mean + Combined Genres Effect	1.0184056
Mean + Sum of Single Genre Effects	1.0461273

4.3.6 Single Predictor Results

The results of the single predictors are as follows;

```
rmse_results
```

method	rmse
Mean Rating	1.0612018
Mean + Movie Effect	0.9439087
Mean + User Effect	0.9783360
Mean + Year Effect	1.0500259
Mean + Combined Genres Effect	1.0184056
Mean + Sum of Single Genre Effects	1.0461273

Here the "movie effect" can be seen as the most impactful, but it's likely that when combined, these factors build on each other and create much better predictions.

```
peak_rmse_results[nrow(peak_rmse_results) + 1,] <- list("Movie Effect (Single Predictor)", min(rmse_resu
```

4.4 Stacking Single Predictors

The next tests will combine several effects. To do this systematically, five effects will be run in all possible combinations, creating 32 possible combinations

Again, the results, including a baseline, will be stored in a data frame.

```
mean_rating = mean(edx$rating)
rmse <- calculate_rmse(rep(mean_rating, nrow(validation)), validation$rating)
stacked_rmse_results <- data.frame(movie_effect = c(FALSE),
                                   user_effect = c(FALSE),
                                   year_effect = c(FALSE),
                                   combined_genres_effect = c(FALSE),
                                   sum_single_genres_effect = c(FALSE),
                                   rmse = c(rmse),
```

```
                                    stringsAsFactors = FALSE)
rm(rmse)
stacked_rmse_results
```

movie_effect	user_effect	year_effect	combined_genres_effect	sum_single_genres_effect	rmse
FALSE	FALSE	FALSE	FALSE	FALSE	1.061202

Since the logic has already been explained, the code contains simple comments noting which combination is represented by each section, and why certain decisions were made.

```
# To reduce duplicate work, we will work on a copy of validation, with the year
# column added, and a few other small mutations made. Making these once makes
# this whole chunk quite speedy.

working_validation <- validation %>%
                        mutate(movieId = as.factor(movieId)) %>%
                        mutate(userId = as.factor(userId)) %>%
                        mutate(year = as.factor(str_remove_all(str_extract(title,
                                                                           "\\(\\d{4}\\)$"), "[()]"))) %
                        rowwise() %>%
                        mutate(sum_single_genres_effect = calculate_genre_effect_sum(genres)) %>%
                        ungroup() %>%
                        left_join(movie_effects, by='movieId') %>%
                        left_join(user_effects, by='userId') %>%
                        left_join(year_effects, by='year') %>%
                        left_join(genre_combination_effects, by='genres')

# Starting with the movie effects only, note that movie_effects is already defined
predictions <- working_validation %>%
               mutate(prediction = (mean_rating + movie_effect)) %>%
               .$prediction
rmse <- print(calculate_rmse(predictions, validation$rating))
```

```
## [1] 0.9439087
```

```
stacked_rmse_results[nrow(stacked_rmse_results) + 1,] <- list(TRUE, # Movie effect
                                                              FALSE, # User effect
                                                              FALSE, # Year effect
                                                              FALSE, # Combined genres effect
                                                              FALSE, # Sum of single genre effects
                                                              rmse)
rm(rmse, predictions)

# Next, user effect only, again, user_effects is defined

predictions <- working_validation %>%
                 mutate(prediction = (mean_rating + user_effect)) %>%
                 .$prediction
rmse <- calculate_rmse(predictions, validation$rating)
stacked_rmse_results[nrow(stacked_rmse_results) + 1,] <- list(FALSE, # Movie effect
                                                              TRUE, # User effect
                                                              FALSE, # Year effect
                                                              FALSE, # Combined genres effect
```

```
                                                        FALSE, # Sum of single genre effects
                                                        rmse)
rm(rmse, predictions)

# This time, the combination of user and movie effects
predictions <- working_validation %>%
                mutate(combined_effect = user_effect + movie_effect) %>%
                mutate(prediction = mean_rating + combined_effect) %>%
                .$prediction
rmse <- calculate_rmse(predictions, validation$rating)
stacked_rmse_results[nrow(stacked_rmse_results) + 1,] <- list(TRUE, # Movie effect
                                                        TRUE, # User effect
                                                        FALSE, # Year effect
                                                        FALSE, # Combined genres effect
                                                        FALSE, # Sum of single genre effects
                                                        rmse)
rm(rmse, predictions)

# Next, year effect only, again, year_effects is defined

predictions <- working_validation %>%
                mutate(prediction = (mean_rating + year_effect)) %>%
                .$prediction
rmse <- calculate_rmse(predictions, validation$rating)
stacked_rmse_results[nrow(stacked_rmse_results) + 1,] <- list(FALSE, # Movie effect
                                                        FALSE, # User effect
                                                        TRUE, # Year effect
                                                        FALSE, # Combined genres effect
                                                        FALSE, # Sum of single genre effects
                                                        rmse)
rm(rmse, predictions)

# Movie and year
predictions <- working_validation %>%
                mutate(combined_effect = movie_effect + year_effect) %>%
                mutate(prediction = (mean_rating + combined_effect)) %>%
                .$prediction
rmse <- calculate_rmse(predictions, validation$rating)
stacked_rmse_results[nrow(stacked_rmse_results) + 1,] <- list(TRUE, # Movie effect
                                                        FALSE, # User effect
                                                        TRUE, # Year effect
                                                        FALSE, # Combined genres effect
                                                        FALSE, # Sum of single genre effects
                                                        rmse)
rm(rmse, predictions)

# User and year
predictions <- working_validation %>%
                mutate(combined_effect = user_effect + year_effect) %>%
                mutate(prediction = (mean_rating + combined_effect)) %>%
                .$prediction
rmse <- calculate_rmse(predictions, validation$rating)
stacked_rmse_results[nrow(stacked_rmse_results) + 1,] <- list(FALSE, # Movie effect
```

```
                                                                TRUE, # User effect
                                                                TRUE, # Year effect
                                                                FALSE, # Combined genres effect
                                                                FALSE, # Sum of single genre effects
                                                                rmse)
rm(rmse, predictions)

# Now the first with three; movie, user and year
predictions <- working_validation %>%
                mutate(combined_effect = movie_effect + user_effect + year_effect) %>%
                mutate(prediction = (mean_rating + combined_effect)) %>%
                .$prediction
rmse <- calculate_rmse(predictions, validation$rating)
stacked_rmse_results[nrow(stacked_rmse_results) + 1,] <- list(TRUE, # Movie effect
                                                                TRUE, # User effect
                                                                TRUE, # Year effect
                                                                FALSE, # Combined genres effect
                                                                FALSE, # Sum of single genre effects
                                                                rmse)
rm(rmse, predictions)

# Now for the combined genres effect
predictions <- working_validation %>%
                mutate(prediction = (mean_rating + genre_combination_effect)) %>%
                .$prediction
rmse <- calculate_rmse(predictions, validation$rating)
stacked_rmse_results[nrow(stacked_rmse_results) + 1,] <- list(FALSE, # Movie effect
                                                                FALSE, # User effect
                                                                FALSE, # Year effect
                                                                TRUE, # Combined genres effect
                                                                FALSE, # Sum of single genre effects
                                                                rmse)
rm(rmse, predictions)

# Now for the movie effect and the combined genres effect
predictions <- working_validation %>%
                mutate(combined_effect = movie_effect + genre_combination_effect) %>%
                mutate(prediction = (mean_rating + combined_effect)) %>%
                .$prediction
rmse <- calculate_rmse(predictions, validation$rating)
stacked_rmse_results[nrow(stacked_rmse_results) + 1,] <- list(TRUE, # Movie effect
                                                                FALSE, # User effect
                                                                FALSE, # Year effect
                                                                TRUE, # Combined genres effect
                                                                FALSE, # Sum of single genre effects
                                                                rmse)
rm(rmse, predictions)

# Now for the user effect and the combined genres effect
predictions <- working_validation %>%
                mutate(combined_effect = user_effect + genre_combination_effect) %>%
                mutate(prediction = (mean_rating + combined_effect)) %>%
                .$prediction
```

```
rmse <- calculate_rmse(predictions, validation$rating)
stacked_rmse_results[nrow(stacked_rmse_results) + 1,] <- list(FALSE, # Movie effect
                                                              TRUE, # User effect
                                                              FALSE, # Year effect
                                                              TRUE, # Combined genres effect
                                                              FALSE, # Sum of single genre effects
                                                              rmse)
rm(rmse, predictions)

# Now for the movie effect, user effect and the combined genres effect
predictions <- working_validation %>%
                mutate(combined_effect = movie_effect + user_effect + genre_combination_effect) %>%
                mutate(prediction = (mean_rating + combined_effect)) %>%
                .$prediction
rmse <- calculate_rmse(predictions, validation$rating)
stacked_rmse_results[nrow(stacked_rmse_results) + 1,] <- list(TRUE, # Movie effect
                                                              TRUE, # User effect
                                                              FALSE, # Year effect
                                                              TRUE, # Combined genres effect
                                                              FALSE, # Sum of single genre effects
                                                              rmse)
rm(rmse, predictions)

# Now for the year and the combined genres effect
predictions <- working_validation %>%
                mutate(combined_effect = year_effect + genre_combination_effect) %>%
                mutate(prediction = (mean_rating + combined_effect)) %>%
                .$prediction
rmse <- calculate_rmse(predictions, validation$rating)
stacked_rmse_results[nrow(stacked_rmse_results) + 1,] <- list(FALSE, # Movie effect
                                                              FALSE, # User effect
                                                              TRUE, # Year effect
                                                              TRUE, # Combined genres effect
                                                              FALSE, # Sum of single genre effects
                                                              rmse)
rm(rmse, predictions)

# Now for the movie, the year and the combined genres effect
predictions <- working_validation %>%
                mutate(combined_effect = movie_effect + year_effect + genre_combination_effect) %>%
                mutate(prediction = (mean_rating + combined_effect)) %>%
                .$prediction
rmse <- calculate_rmse(predictions, validation$rating)
stacked_rmse_results[nrow(stacked_rmse_results) + 1,] <- list(TRUE, # Movie effect
                                                              FALSE, # User effect
                                                              TRUE, # Year effect
                                                              TRUE, # Combined genres effect
                                                              FALSE, # Sum of single genre effects
                                                              rmse)
rm(rmse, predictions)

# Now for the movie, the year and the combined genres effect
predictions <- working_validation %>%
```

```
                mutate(combined_effect = user_effect + year_effect + genre_combination_effect) %>%
                mutate(prediction = (mean_rating + combined_effect)) %>%
                .$prediction
rmse <- calculate_rmse(predictions, validation$rating)
stacked_rmse_results[nrow(stacked_rmse_results) + 1,] <- list(FALSE, # Movie effect
                                                              TRUE, # User effect
                                                              TRUE, # Year effect
                                                              TRUE, # Combined genres effect
                                                              FALSE, # Sum of single genre effects
                                                              rmse)
rm(rmse, predictions)

# Now for the movie, user, year and the combined genres effect
predictions <- working_validation %>%
                mutate(combined_effect = movie_effect + user_effect + year_effect + genre_combination_
                mutate(prediction = (mean_rating + combined_effect)) %>%
                .$prediction
rmse <- calculate_rmse(predictions, validation$rating)
stacked_rmse_results[nrow(stacked_rmse_results) + 1,] <- list(TRUE, # Movie effect
                                                              TRUE, # User effect
                                                              TRUE, # Year effect
                                                              TRUE, # Combined genres effect
                                                              FALSE, # Sum of single genre effects
                                                              rmse)
rm(rmse, predictions)

# Now for the combination of the single genres effects
predictions <- working_validation %>%
                mutate(prediction = (mean_rating + sum_single_genres_effect)) %>%
                .$prediction
rmse <- calculate_rmse(predictions, validation$rating)
stacked_rmse_results[nrow(stacked_rmse_results) + 1,] <- list(FALSE, # Movie effect
                                                              FALSE, # User effect
                                                              FALSE, # Year effect
                                                              FALSE, # Combined genres effect
                                                              TRUE, # Sum of single genre effects
                                                              rmse)
rm(rmse, predictions)

# Now for the movie effect, plus the combination of the single genres effects
predictions <- working_validation %>%
                mutate(combined_effect = movie_effect + sum_single_genres_effect) %>%
                mutate(prediction = (mean_rating + combined_effect)) %>%
                .$prediction
rmse <- calculate_rmse(predictions, validation$rating)
stacked_rmse_results[nrow(stacked_rmse_results) + 1,] <- list(TRUE, # Movie effect
                                                              FALSE, # User effect
                                                              FALSE, # Year effect
                                                              FALSE, # Combined genres effect
                                                              TRUE, # Sum of single genre effects
                                                              rmse)
rm(rmse, predictions)
```

```r
# Now for the user effect, plus the combination of the single genres effects
predictions <- working_validation %>%
                mutate(combined_effect = user_effect + sum_single_genres_effect) %>%
                mutate(prediction = (mean_rating + combined_effect)) %>%
                .$prediction
rmse <- calculate_rmse(predictions, validation$rating)
stacked_rmse_results[nrow(stacked_rmse_results) + 1,] <- list(FALSE, # Movie effect
                                                              TRUE, # User effect
                                                              FALSE, # Year effect
                                                              FALSE, # Combined genres effect
                                                              TRUE, # Sum of single genre effects
                                                              rmse)
rm(rmse, predictions)

# Now for the sum of the movie effect, user effect and the combination of the single genres effects
predictions <- working_validation %>%
                mutate(combined_effect = movie_effect + user_effect + sum_single_genres_effect) %>%
                mutate(prediction = (mean_rating + combined_effect)) %>%
                .$prediction
rmse <- calculate_rmse(predictions, validation$rating)
stacked_rmse_results[nrow(stacked_rmse_results) + 1,] <- list(TRUE, # Movie effect
                                                              TRUE, # User effect
                                                              FALSE, # Year effect
                                                              FALSE, # Combined genres effect
                                                              TRUE, # Sum of single genre effects
                                                              rmse)
rm(rmse, predictions)

# Now for the sum of the year effect and the combination of the single genres effects
predictions <- working_validation %>%
                mutate(combined_effect = year_effect + sum_single_genres_effect) %>%
                mutate(prediction = (mean_rating + combined_effect)) %>%
                .$prediction
rmse <- calculate_rmse(predictions, validation$rating)
stacked_rmse_results[nrow(stacked_rmse_results) + 1,] <- list(FALSE, # Movie effect
                                                              FALSE, # User effect
                                                              TRUE, # Year effect
                                                              FALSE, # Combined genres effect
                                                              TRUE, # Sum of single genre effects
                                                              rmse)
rm(rmse, predictions)

# Now for the sum of the movie effect, year effect and the combination of the single genres effects
predictions <- working_validation %>%
                mutate(combined_effect = movie_effect + year_effect + sum_single_genres_effect) %>%
                mutate(prediction = (mean_rating + combined_effect)) %>%
                .$prediction
rmse <- calculate_rmse(predictions, validation$rating)
stacked_rmse_results[nrow(stacked_rmse_results) + 1,] <- list(TRUE, # Movie effect
                                                              FALSE, # User effect
                                                              TRUE, # Year effect
                                                              FALSE, # Combined genres effect
                                                              TRUE, # Sum of single genre effects
```

```
                                                                rmse)
rm(rmse, predictions)

# Now for the sum of the user effect, year effect and the combination of the single genres effects
predictions <- working_validation %>%
                mutate(combined_effect = user_effect + year_effect + sum_single_genres_effect) %>%
                mutate(prediction = (mean_rating + combined_effect)) %>%
                .$prediction
rmse <- calculate_rmse(predictions, validation$rating)
stacked_rmse_results[nrow(stacked_rmse_results) + 1,] <- list(FALSE, # Movie effect
                                                              TRUE, # User effect
                                                              TRUE, # Year effect
                                                              FALSE, # Combined genres effect
                                                              TRUE, # Sum of single genre effects
                                                              rmse)
rm(rmse, predictions)

# Now for four; movie, user and year effects and the combination of the single genres effects
predictions <- working_validation %>%
                mutate(combined_effect = movie_effect + user_effect + year_effect + sum_single_genres_
                mutate(prediction = (mean_rating + combined_effect)) %>%
                .$prediction
rmse <- calculate_rmse(predictions, validation$rating)
stacked_rmse_results[nrow(stacked_rmse_results) + 1,] <- list(TRUE, # Movie effect
                                                              TRUE, # User effect
                                                              TRUE, # Year effect
                                                              FALSE, # Combined genres effect
                                                              TRUE, # Sum of single genre effects
                                                              rmse)
rm(rmse, predictions)

# Now for completeness, the following are included, but since they include
# two different ways of finding the gnere effect, I don't expect them to
# be too helpful

# Combined genres + single genres
predictions <- working_validation %>%
                mutate(combined_effect = genre_combination_effect + sum_single_genres_effect) %>%
                mutate(prediction = (mean_rating + combined_effect)) %>%
                .$prediction
rmse <- calculate_rmse(predictions, validation$rating)
stacked_rmse_results[nrow(stacked_rmse_results) + 1,] <- list(FALSE, # Movie effect
                                                              FALSE, # User effect
                                                              FALSE, # Year effect
                                                              TRUE, # Combined genres effect
                                                              TRUE, # Sum of single genre effects
                                                              rmse)
rm(rmse, predictions)

# Movie, combined genres + single genres
predictions <- working_validation %>%
                mutate(combined_effect = movie_effect + genre_combination_effect + sum_single_genres_e
                mutate(prediction = (mean_rating + combined_effect)) %>%
```

```
                .$prediction
rmse <- calculate_rmse(predictions, validation$rating)
stacked_rmse_results[nrow(stacked_rmse_results) + 1,] <- list(TRUE, # Movie effect
                                                              FALSE, # User effect
                                                              FALSE, # Year effect
                                                              TRUE, # Combined genres effect
                                                              TRUE, # Sum of single genre effects
                                                              rmse)
rm(rmse, predictions)

# User, combined genres + single genres
predictions <- working_validation %>%
                mutate(combined_effect = user_effect + genre_combination_effect + sum_single_genres_ef
                mutate(prediction = (mean_rating + combined_effect)) %>%
                .$prediction
rmse <- calculate_rmse(predictions, validation$rating)
stacked_rmse_results[nrow(stacked_rmse_results) + 1,] <- list(FALSE, # Movie effect
                                                              TRUE, # User effect
                                                              FALSE, # Year effect
                                                              TRUE, # Combined genres effect
                                                              TRUE, # Sum of single genre effects
                                                              rmse)
rm(rmse, predictions)

# Movie, user, combined genres + single genres
predictions <- working_validation %>%
                mutate(combined_effect = movie_effect + user_effect + genre_combination_effect + sum_s
                mutate(prediction = (mean_rating + combined_effect)) %>%
                .$prediction
rmse <- calculate_rmse(predictions, validation$rating)
stacked_rmse_results[nrow(stacked_rmse_results) + 1,] <- list(TRUE, # Movie effect
                                                              TRUE, # User effect
                                                              FALSE, # Year effect
                                                              TRUE, # Combined genres effect
                                                              TRUE, # Sum of single genre effects
                                                              rmse)
rm(rmse, predictions)

# Year, combined genres + single genres
predictions <- working_validation %>%
                mutate(combined_effect = year_effect + genre_combination_effect + sum_single_genres_ef
                mutate(prediction = (mean_rating + combined_effect)) %>%
                .$prediction
rmse <- calculate_rmse(predictions, validation$rating)
stacked_rmse_results[nrow(stacked_rmse_results) + 1,] <- list(FALSE, # Movie effect
                                                              FALSE, # User effect
                                                              TRUE, # Year effect
                                                              TRUE, # Combined genres effect
                                                              TRUE, # Sum of single genre effects
                                                              rmse)
rm(rmse, predictions)

# Movie, year, combined genres + single genres
```

```
predictions <- working_validation %>%
                mutate(combined_effect = movie_effect + year_effect + genre_combination_effect + sum_si
                mutate(prediction = (mean_rating + combined_effect)) %>%
                .$prediction
rmse <- calculate_rmse(predictions, validation$rating)
stacked_rmse_results[nrow(stacked_rmse_results) + 1,] <- list(TRUE, # Movie effect
                                                              FALSE, # User effect
                                                              TRUE, # Year effect
                                                              TRUE, # Combined genres effect
                                                              TRUE, # Sum of single genre effects
                                                              rmse)
rm(rmse, predictions)

# User, year, combined genres + single genres
predictions <- working_validation %>%
                mutate(combined_effect = user_effect + year_effect + genre_combination_effect + sum_sin
                mutate(prediction = (mean_rating + combined_effect)) %>%
                .$prediction
rmse <- calculate_rmse(predictions, validation$rating)
stacked_rmse_results[nrow(stacked_rmse_results) + 1,] <- list(FALSE, # Movie effect
                                                              TRUE, # User effect
                                                              TRUE, # Year effect
                                                              TRUE, # Combined genres effect
                                                              TRUE, # Sum of single genre effects
                                                              rmse)
rm(rmse, predictions)

# Movie, user, year, combined genres + single genres
predictions <- working_validation %>%
                mutate(combined_effect = user_effect + year_effect + genre_combination_effect + sum_sin
                mutate(prediction = (mean_rating + combined_effect)) %>%
                .$prediction
rmse <- calculate_rmse(predictions, validation$rating)
stacked_rmse_results[nrow(stacked_rmse_results) + 1,] <- list(TRUE, # Movie effect
                                                              TRUE, # User effect
                                                              TRUE, # Year effect
                                                              TRUE, # Combined genres effect
                                                              TRUE, # Sum of single genre effects
                                                              rmse)
rm(rmse, predictions)

stacked_rmse_results %>% arrange(rmse)
```

movie_effect	user_effect	year_effect	combined_genres_effect	sum_single_genres_effect	rmse
TRUE	TRUE	FALSE	FALSE	FALSE	0.8850398
TRUE	TRUE	TRUE	FALSE	FALSE	0.9015546
TRUE	TRUE	FALSE	FALSE	TRUE	0.9139289
TRUE	TRUE	TRUE	FALSE	TRUE	0.9324077
FALSE	TRUE	TRUE	TRUE	FALSE	0.9412588
FALSE	TRUE	FALSE	TRUE	FALSE	0.9436451
TRUE	FALSE	FALSE	FALSE	FALSE	0.9439087
TRUE	TRUE	FALSE	TRUE	FALSE	0.9453827
TRUE	FALSE	TRUE	FALSE	FALSE	0.9561340

movie_effect	user_effect	year_effect	combined_genres_effect	sum_single_genres_effect	rmse
FALSE	TRUE	TRUE	FALSE	TRUE	0.9599412
TRUE	FALSE	FALSE	FALSE	TRUE	0.9664966
FALSE	TRUE	FALSE	FALSE	TRUE	0.9665365
TRUE	TRUE	TRUE	TRUE	FALSE	0.9675092
FALSE	TRUE	TRUE	FALSE	FALSE	0.9694447
FALSE	TRUE	FALSE	TRUE	TRUE	0.9707140
FALSE	TRUE	TRUE	TRUE	TRUE	0.9707730
TRUE	TRUE	TRUE	TRUE	TRUE	0.9707730
FALSE	TRUE	FALSE	FALSE	FALSE	0.9783360
TRUE	FALSE	TRUE	FALSE	TRUE	0.9807941
TRUE	FALSE	FALSE	TRUE	FALSE	0.9894909
TRUE	FALSE	TRUE	TRUE	FALSE	1.0075421
TRUE	TRUE	FALSE	TRUE	TRUE	1.0101916
FALSE	FALSE	TRUE	TRUE	FALSE	1.0131017
FALSE	FALSE	FALSE	TRUE	FALSE	1.0184056
FALSE	FALSE	TRUE	TRUE	TRUE	1.0363343
FALSE	FALSE	TRUE	FALSE	TRUE	1.0370149
FALSE	FALSE	FALSE	TRUE	TRUE	1.0393031
FALSE	FALSE	FALSE	FALSE	TRUE	1.0461273
TRUE	FALSE	FALSE	TRUE	TRUE	1.0473836
FALSE	FALSE	TRUE	FALSE	FALSE	1.0500259
FALSE	FALSE	FALSE	FALSE	FALSE	1.0612018
TRUE	FALSE	TRUE	TRUE	TRUE	1.0666180

These data seem to show that the model using only `user` and `movie` performed the best of all, but it still offers some room for improvement.

```
peak_rmse_results[nrow(peak_rmse_results) + 1,] <- c("Movie & User Effects (Simple Stacked)",
                                                     min(stacked_rmse_results$rmse))
```

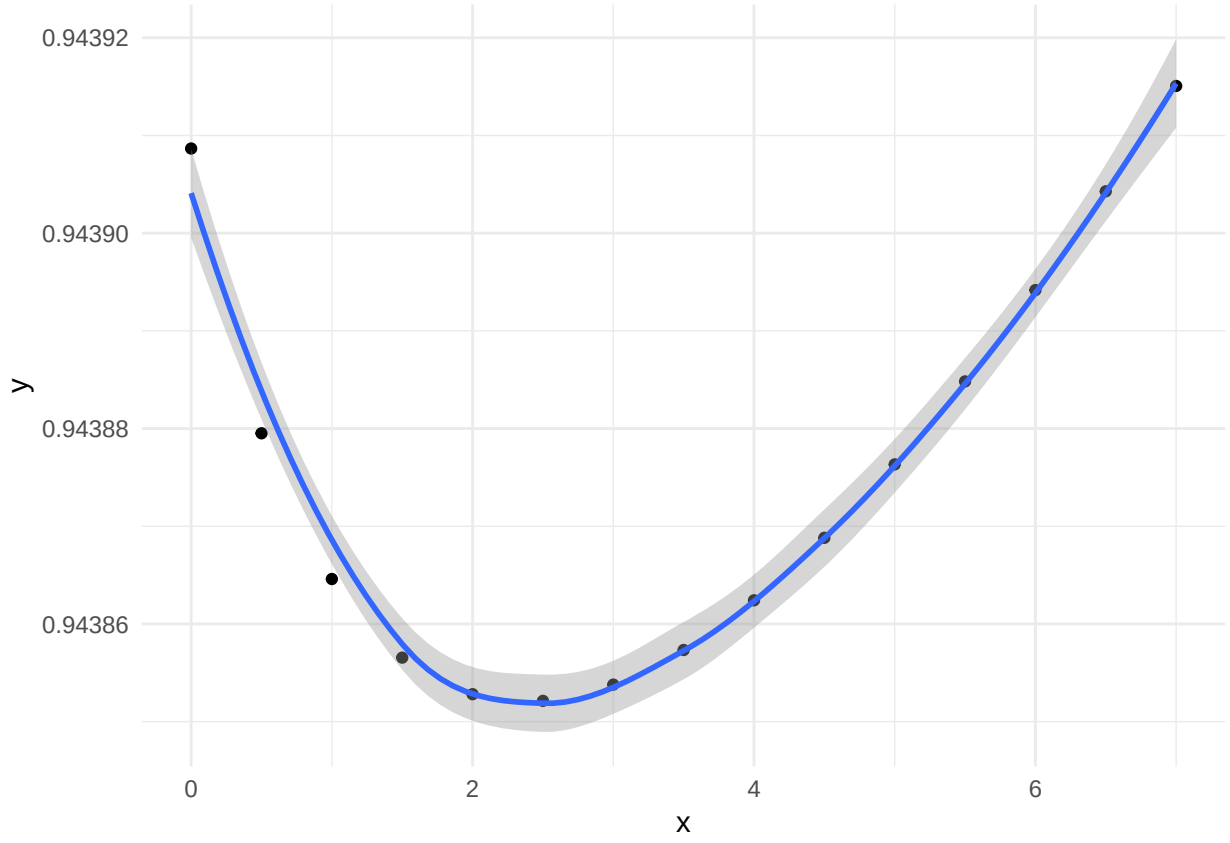
4.5 Regularising the Movie and User Effects

Using regularisation, a lambda value can be found for the effects, starting with the movie effect.

This takes into account the number of ratings, and scales the impact of the movie effect to how many items in the training set relate to the movie - the higher this count, the more reliable this effect.

```
values_to_test <- seq(0, 7, 0.5)

rmse_values <- sapply(values_to_test, function(lambda){
  regularised_movie_effects <- edx %>%
                               group_by(movieId) %>%
                               summarize(movie_effect = mean(rating - mean_rating),
                                         ratings_for_movie = n(),
                                         regularised_effect = sum(rating - mean_rating) / (n() + lambd

  predictions <- validation %>%
                 mutate(movieId = as.factor(movieId)) %>%
                 left_join(regularised_movie_effects, by='movieId') %>%
                 mutate(prediction = (mean_rating + regularised_effect)) %>%
                 .$prediction
  return(calculate_rmse(predictions, validation$rating))
})
```

```
ggplot(data.frame(y = rmse_values, x = values_to_test), aes(x = x, y = y)) + geom_point() + geom_smooth
```

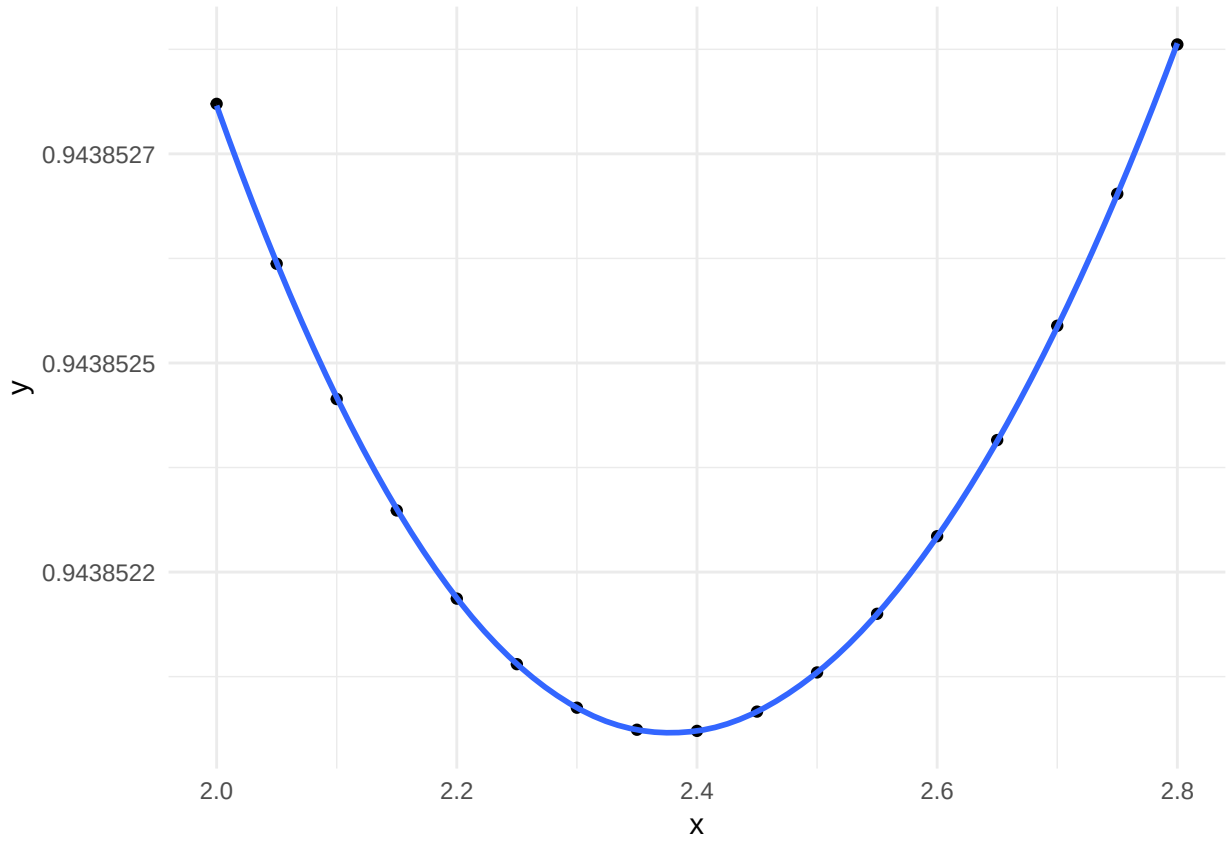



```
rm(rmse_values, values_to_test)
```

The optimal lambda value can be seen at around 2.5.

This can be repeated to find a more precise value.

```
values_to_test <- seq(2, 2.8, 0.05)

rmse_values <- sapply(values_to_test, function(lambda){
  regularised_movie_effects <- edx %>%
                               group_by(movieId) %>%
                               summarize(movie_effect = mean(rating - mean_rating),
                                         ratings_for_movie = n(),
                                         regularised_effect = sum(rating - mean_rating) / (n() + lambd

  predictions <- validation %>%
                 mutate(movieId = as.factor(movieId)) %>%
                 left_join(regularised_movie_effects, by='movieId') %>%
                 mutate(prediction = (mean_rating + regularised_effect)) %>%
                 .$prediction
  return(calculate_rmse(predictions, validation$rating))
})

ggplot(data.frame(y = rmse_values, x = values_to_test), aes(x = x, y = y)) + geom_point() + geom_smooth
```

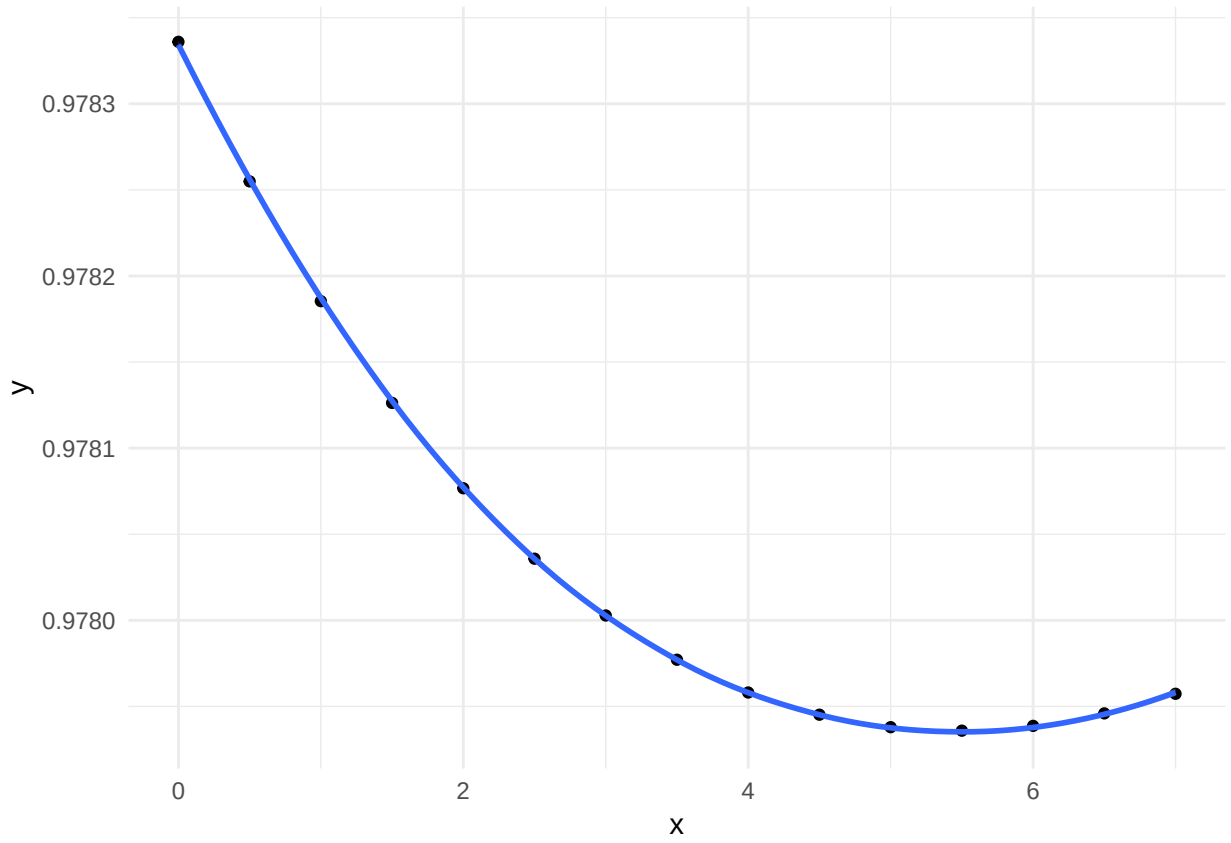



```
rm(rmse_values, values_to_test)
```

The plot shows that the movie effect has a lambda value of approximately `2.38`.

This can also be repeated for the user effect.

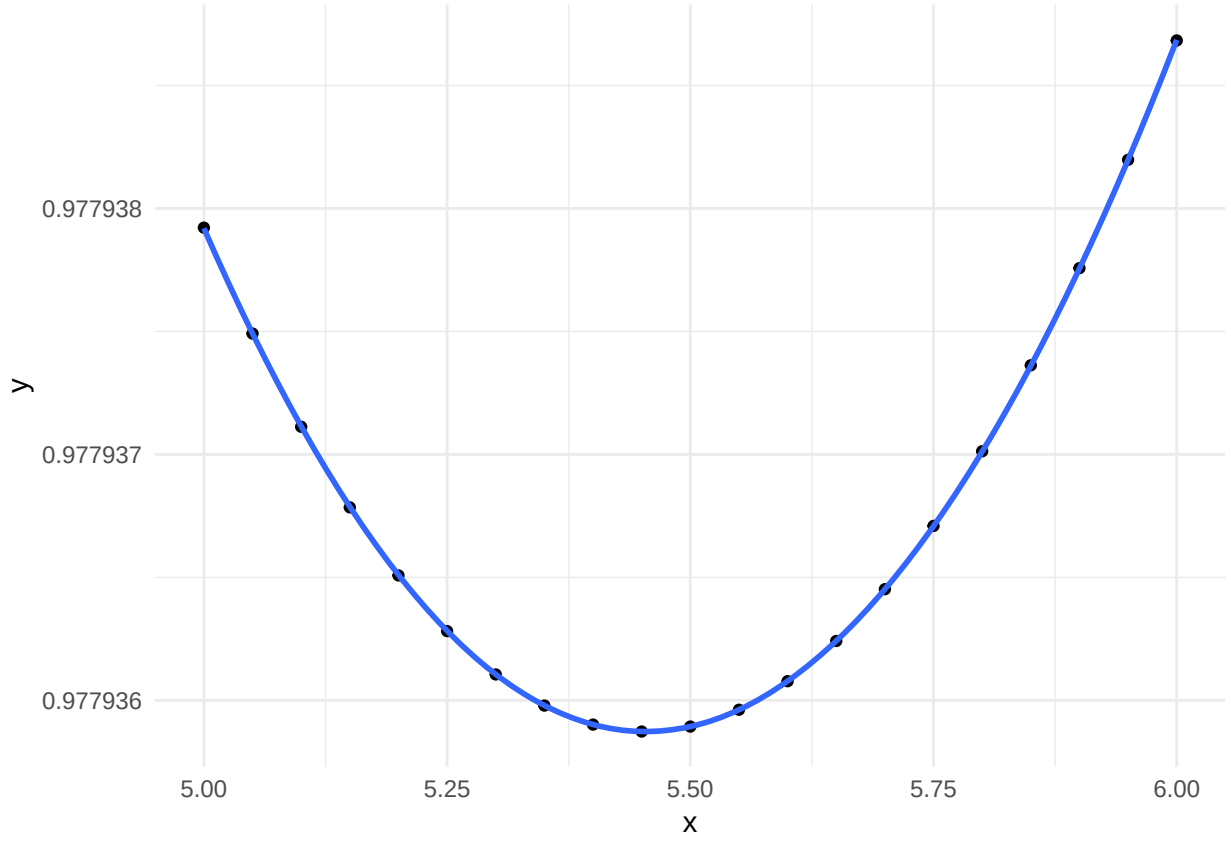
```
values_to_test <- seq(0, 7, 0.5)

rmse_values <- sapply(values_to_test, function(lambda){
  regularised_user_effects <- edx %>%
                              group_by(userId) %>%
                              summarize(userEffect = mean(rating - mean_rating),
                                        ratings_by_user = n(),
                                        regularised_effect = sum(rating - mean_rating) / (n() + lambda

  predictions <- validation %>%
                 mutate(userId = as.factor(userId)) %>%
                 left_join(regularised_user_effects, by='userId') %>%
                 mutate(prediction = (mean_rating + regularised_effect)) %>%
                 .$prediction
  return(calculate_rmse(predictions, validation$rating))
})

ggplot(data.frame(y = rmse_values, x = values_to_test), aes(x = x, y = y)) + geom_point() + geom_smooth
```




```
rm(rmse_values, values_to_test)
```

It looks like the lambda value is between 5 and 6, let's zoom in one more time;

```
values_to_test <- seq(5, 6, 0.05)

rmse_values <- sapply(values_to_test, function(lambda){
  regularised_user_effects <- edx %>%
                              group_by(userId) %>%
                              summarize(userEffect = mean(rating - mean_rating),
                                        ratings_by_user = n(),
                                        regularised_effect = sum(rating - mean_rating) / (n() + lambda

  predictions <- validation %>%
                 mutate(userId = as.factor(userId)) %>%
                 left_join(regularised_user_effects, by='userId') %>%
                 mutate(prediction = (mean_rating + regularised_effect)) %>%
                 .$prediction
  return(calculate_rmse(predictions, validation$rating))
})

ggplot(data.frame(y = rmse_values, x = values_to_test), aes(x = x, y = y)) + geom_point() + geom_smooth
```




```
rm(rmse_values, values_to_test)
```

We'll go with 5.45. Now let's try using these in the "winning" model, which uses only user ID and movie ID.

```
moviesLambda <- 2.38
usersLambda <- 5.45

regularised_movie_effects <- edx %>%
                              group_by(movieId) %>%
                              summarize(movie_effect = mean(rating - mean_rating),
                                        ratings_for_movie = n(),
                                        regularised_movie_effect = sum(rating - mean_rating) / (n() + r

regularised_user_effects <- edx %>%
                            group_by(userId) %>%
                            summarize(user_effect = mean(rating - mean_rating),
                                      ratings_by_user = n(),
                                      regularised_user_effect = sum(rating - mean_rating) / (n() + use

predictions <- validation %>%
               mutate(userId = as.factor(userId)) %>%
               mutate(movieId = as.factor(movieId)) %>%
               left_join(regularised_movie_effects, by='movieId') %>%
               left_join(regularised_user_effects, by='userId') %>%
               mutate(prediction = (mean_rating + (regularised_movie_effect + regularised_user_effect))
               .$prediction
```

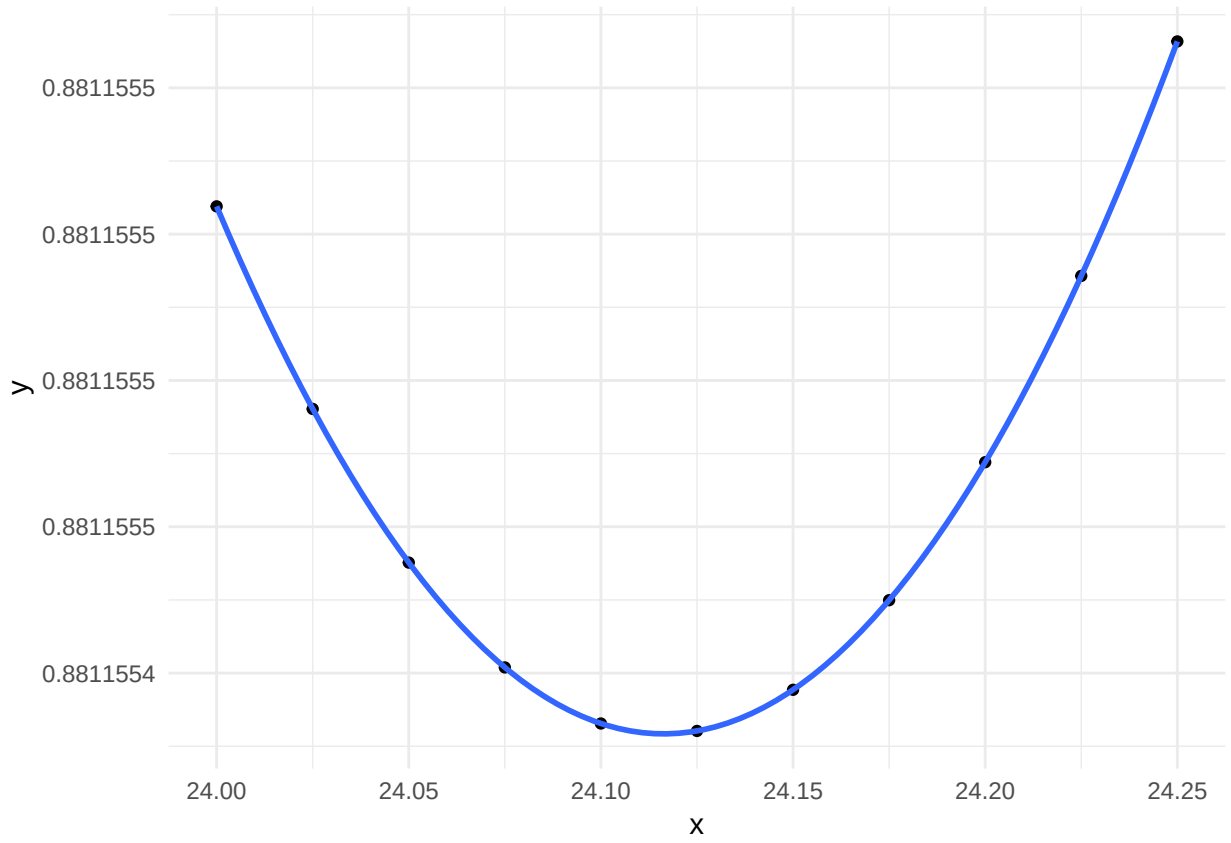
```
print(calculate_rmse(predictions, validation$rating))
```

```
## [1] 0.8830816
```

```
rm(predictions, regularised_user_effects, regularised_movie_effects, moviesLambda, usersLambda)
```

Now this can be repeated to find a lambda that works for both.

```
values_to_test <- seq(24, 24.25, 0.025)

rmse_values <- sapply(values_to_test, function(lambda){
  regularised_movie_effects <- edx %>%
                               group_by(movieId) %>%
                               summarize(movie_effect = mean(rating - mean_rating),
                                         ratings_for_movie = n(),
                                         regularised_movie_effect = sum(rating - mean_rating) / (n() +

  regularised_user_effects <- edx %>%
                              group_by(userId) %>%
                              summarize(user_effect = mean(rating - mean_rating),
                                        ratings_by_user = n(),
                                        regularised_user_effect = sum(rating - mean_rating) / (n() + la

  predictions <- validation %>%
                 mutate(userId = as.factor(userId)) %>%
                 mutate(movieId = as.factor(movieId)) %>%
                 left_join(regularised_movie_effects, by='movieId') %>%
                 left_join(regularised_user_effects, by='userId') %>%
                 mutate(prediction = (mean_rating + (regularised_movie_effect + regularised_user_effec
                 .$prediction

  return(calculate_rmse(predictions, validation$rating))
})

ggplot(data.frame(y = rmse_values, x = values_to_test), aes(x = x, y = y)) + geom_point() + geom_smooth
```




```
rm(values_to_test, rmse_values)
```

The lambda values seems to be `24.125`. Let's calculate the precise RMSE with that value.

```
lambda <- 24.125

regularised_movie_effects <- edx %>%
                               group_by(movieId) %>%
                               summarize(movie_effect = mean(rating - mean_rating),
                                         ratings_for_movie = n(),
                                         regularised_movie_effect = sum(rating - mean_rating) / (n() + l

regularised_user_effects <- edx %>%
                             group_by(userId) %>%
                             summarize(user_effect = mean(rating - mean_rating),
                                       ratings_by_user = n(),
                                       regularised_user_effect = sum(rating - mean_rating) / (n() + lam

predictions <- validation %>%
               mutate(userId = as.factor(userId)) %>%
               mutate(movieId = as.factor(movieId)) %>%
               left_join(regularised_movie_effects, by='movieId') %>%
               left_join(regularised_user_effects, by='userId') %>%
               mutate(prediction = (mean_rating + (regularised_movie_effect + regularised_user_effect))
               .$prediction

rmse <- calculate_rmse(predictions, validation$rating)
```

```
print(rmse)
```

```
## [1] 0.8811554
```

```
rm(predictions, regularised_user_effects, regularised_movie_effects, lambda)
```

This gives a small improvement oer the separate lambda values. The results are now getting close to the target and can be added to the list of results to compare.

```
peak_rmse_results[nrow(peak_rmse_results) + 1,] <- c("Movie & User Effects (Regularised)", rmse)
```

4.6 The user-genre effect

An idea was explored that involved finding the user-specific effect of each present genre. So, to tell if a user is likely to enjoy a `Comedy|Romance`, all ratings of movies including `Comedy` will be used to find the user-specific "comedy effect", and all those with `Romance` rated highly will be used to find the "romance effect" for that user.

For users who have enough other movies rated, this may give better results.

Unfortunately, the running time of this method (despite attempts at caching and optimisation) made it impossible to include in the final analysis. The code is included for reference, and in case anyone wants to try this with more computing power and time available.

```
# user_genre_effect <- function(queriedUserId, genre){
#   matching_ratings <- edx %>% filter(userId == queriedUserId) %>% rowwise() %>% filter(includes_genre
#   if(nrow(matching_ratings) == 0){
#     0
#   }else{
#     mean(matching_ratings$rating) - mean_rating
#   }
# }

# m_user_genre_effect <- memoise(user_genre_effect, cache = fscache)

# user_genres_effect <- function(queriedUserId, genresList){
#   genres_as_list <- strsplit(genresList, "|", fixed=TRUE)
#   single_genre_effects <- map(unlist(genres_as_list), function(genre_name) { m_user_genre_effect(quer
#   mean(unlist(single_genre_effects))
# }

# m_user_genres_effect <- memoise(user_genres_effect)

# benchmark_start_time <- Sys.time()
# predictions <- validation %>%
#                 rowwise() %>%
#                 mutate(user_genres_effect = m_user_genres_effect(userId, genres)) %>%
#                 ungroup() %>%
#                 mutate(prediction = mean_rating + user_genres_effect)  %>%
#                 .$prediction # This takes ~57s for 1000.
# benchmark_end_time <- Sys.time()
# print(benchmark_end_time - benchmark_start_time)
# rm(benchmark_end_time, benchmark_start_time)
# rmse <- calculate_rmse(predictions, validation$rating)
# print(rmse)
# rm(rmse, predictions)
```

When this was successfully run, this gave an improvement in error over all of the other genre-based methods.

The following code would give an indication of this combined with user and movie effects.

```
# benchmark_start_time <- Sys.time()
# predictions <- validation %>%
#                 mutate(movieId = as.factor(movieId), userId = as.factor(userId)) %>%
#                 left_join(movie_effects, by='movieId') %>%
#                 left_join(per_user_effects, by='userId') %>%
#                 rowwise() %>%
#                 mutate(user_genres_effect = m_user_genres_effect(userId, genres)) %>%
#                 ungroup() %>%
#                 mutate(combined_effect = (user_effect + movie_effect + user_genres_effect) / 3) %>%
#                 mutate(prediction = mean_rating + combined_effect)  %>%
#                 .$prediction
# benchmark_end_time <- Sys.time()
# print(benchmark_end_time - benchmark_start_time)
# rm(benchmark_end_time, benchmark_start_time)
# rmse <- calculate_rmse(predictions, validation$rating)
# print(rmse)
# rm(predictions)
```

```
# peak_rmse_results[nrow(peak_rmse_results) + 1,] <- list("User-specific genre effect (combined with mo
```

4.7 Adapting the User & Movie Effects Model

The user and movie effects are good, but don't take account of one another at all. The method that was used to calculate them assumes that they are the only thing affecting the result, when, in fact, both play a part.

Since the effects stack in a certain order, modifying all of the effects is not necessary. Here, the previous movie effects code is used.

```
movie_effects <- edx %>% group_by(movieId) %>% summarize(movie_effect = mean(rating - mean_rating),
                                                         ratings_for_movie = n())
```

New code finds the user effects; this time, the movie effect (found in the previous code chunk) is deducted, before making the calculation of the user effect either way.

```
user_effects <- edx %>%
                  group_by(userId) %>%
                  left_join(movie_effects, by = "movieId") %>%
                  summarize(user_effect = mean(rating - movie_effect - mean_rating),
                            ratings_by_user = n())
```

Now, a new set of predictions can be made, using the sum of the simple movie effect and the user effect which takes account of it.

```
predictions <- validation %>%
                 mutate(movieId = as.factor(movieId), userId = as.factor(userId)) %>%
                 left_join(movie_effects, by = "movieId") %>%
                 left_join(user_effects, by = "userId") %>%
                 mutate(combined_effect = user_effect + movie_effect) %>%
                 mutate(prediction = mean_rating + combined_effect) %>%
                 .$prediction
rmse <- calculate_rmse(predictions, validation$rating)
peak_rmse_results[nrow(peak_rmse_results) + 1,] <- list("Movie & User Effects (Accommodative)", rmse)
peak_rmse_results
```

method	rmse
Mean Rating	1.06120181029262
Movie Effect (Single Predictor)	0.943908662806309
Movie & User Effects (Simple Stacked)	0.88503979532595
Movie & User Effects (Regularised)	0.881155436046365
Movie & User Effects (Accommodative)	0.865348824577316

```
rm( predictions)
```

This gives a noticable improvement, and meets our originally stated target.

5 Results

```
peak_rmse_results
```

method	rmse
Mean Rating	1.06120181029262
Movie Effect (Single Predictor)	0.943908662806309
Movie & User Effects (Simple Stacked)	0.88503979532595
Movie & User Effects (Regularised)	0.881155436046365
Movie & User Effects (Accommodative)	0.865348824577316

The table shows the RMSE values for each attempted prediction method.

The lowest RMSE value was `0.8653`, and was calculated using the effects of the movie, added to the the effects from the user corrected to take account of the movie effect.

This technique can be used to create the script.

6 Conclusion

The nature of these data prevent analysis using more common (and potentially more precise) machine-learning approaches. A training data set of around 9 million rows prevents this type of analysis on a local computer due to memory constraints.

If this were to be repeated with access to High Performance Computing facilities, like those used at universities, some especially interesting mehtods of analysis might have been possible.

Despite the restrictions, an RMSE close to the target specified in the 'Netflix Prize' competition, which set out to solve a similar problem, was able to be attained.

The significance of the "user-specific genre effects" would be an intersting follow-up. While tests showed it as comparable to the final "movie and user combination" results, it may have been possible to improve it further, and to find highly-specific results that are even more specific to users' preferences.

Despite being unable to dig into this deeper at this time, the results found acheived the stated aims, estimating the rating a user will give a movie with relatively low error, and this has been developed into the standalone script which accompanies this report.