MovieLens Project

Recommendation systems help people make better choice depending on their tastes and needs. User ratings are one character which recommendation system follows to recommend choices to the customer. Highly rated and in-demand items are found with help of algorithm for recommending the products. Also depending on the past behaviour of the user an alogrithm can recommend relevant and user-specific items. It helps the user to select the most suitable item from the available options. A happy customer is expected to return to shop frequently and this helps build business.

```r
# Load libraries
library(tidyverse)
library(caret)
```

```r
# Load the data
load("rdas/edx.rda")
str(edx)
```

```
## 'data.frame':    9000055 obs. of  6 variables:
##  $ userId   : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ movieId  : num  122 185 292 316 329 355 356 362 364 370 ...
##  $ rating   : num  5 5 5 5 5 5 5 5 5 5 ...
##  $ timestamp: int  838985046 838983525 838983421 838983392 838983392 838984474 838983653 838984885 83898370
##  $ title    : chr  "Boomerang (1992)" "Net, The (1995)" "Outbreak (1995)" "Stargate (1994)" ...
##  $ genres   : chr  "Comedy|Romance" "Action|Crime|Thriller" "Action|Drama|Sci-Fi|Thriller" "Action|Adventu
```

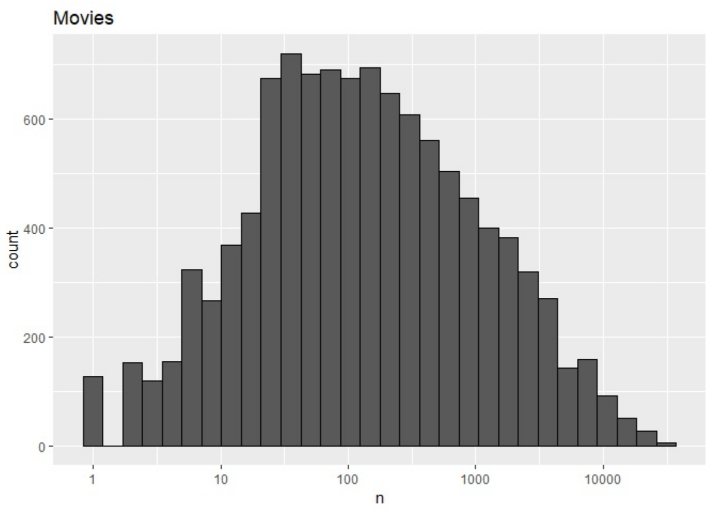
Number of distinct users, movies and genres

```r
edx %>%
  summarize(n_users = n_distinct(userId),
            n_movies = n_distinct(movieId),
            n_genres = n_distinct(genres))
```

```
##   n_users n_movies n_genres
## 1   69878    10677      797
```

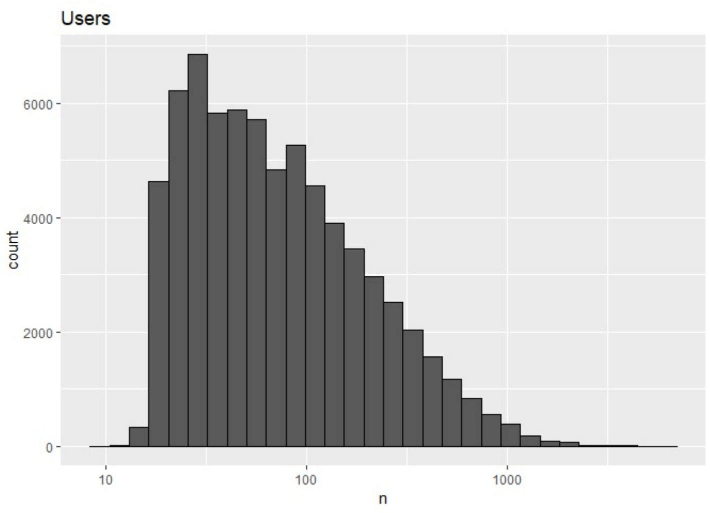
Count gives the number of ratings available for a movie. Explore the times a movie is rated. Can see that some movies are often rated and some very less.

```r
edx %>%
  dplyr::count(movieId) %>%
  ggplot(aes(n)) +
  geom_histogram(bins = 30, color = "black") +
  scale_x_log10() +
  ggtitle("Movies")
```



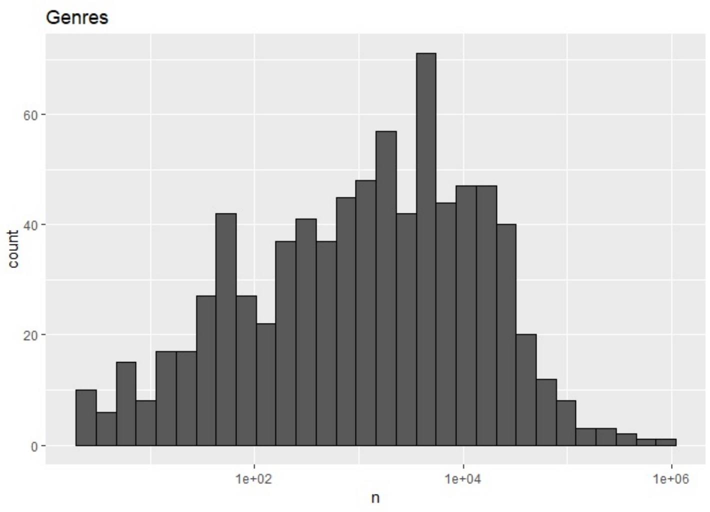
Count gives the number of ratings available from a user. Explore the times a user rates a movie. Can see that some users often give rating and some very less.

```r
edx %>%
  dplyr::count(userId) %>%
  ggplot(aes(n)) +
  geom_histogram(bins = 30, color = "black") +
  scale_x_log10() +
  ggtitle("Users")
```



Same effect is seen in genre also, that some genre is rated often

```
edx %>%
  dplyr::count(genres) %>%
  ggplot(aes(n)) +
  geom_histogram(bins = 30, color = "black") +
  scale_x_log10() +
  ggtitle("Genres")
```



Number of ratings available is important that a larger number gives a more reliable rating. The average of large number of ratings will be more reliable than one person rating a movie too high or too low. One user rating a bad movie too high will spoil the power to predict the movie ratings. So the number of ratings needs to be taken care.

Movie recommendation system here uses User Id, Movie Id, Genres and the Movie rating. Here every user does not rate every movie. So all the user, movie and genre specific effects on rating is used as predictors in movie recommendations.

In movie recommendation the user effect, movie effect and genre effect are incorporated in the algorithm so that we expect the algorithm to predict the ratings of all the movies based on the available ratings along with the biases for the user behaviour, individual movies and the genres.

Mean movie rating

```
mu <- mean(edx$rating)
```

Create a function to calculate the RMSE to help model evaluation

```
RMSE <- function(true_ratings, predicted_ratings){
     sqrt(mean((true_ratings - predicted_ratings)^2))
}
```

Load the validation dataset for evaluating the prediction accuracy

```
load("rdas/validation.rda")
```

Here the mean rating across all movies and users, most naive figure is used as the predicted rating. Calculate the RMSE for the naive model used for prediction. Tabulate the RMSE for the model

```
naive_rmse <- RMSE(validation$rating, mu)
rmse_results <- data_frame(method = "Just the average", RMSE = naive_rmse)
rmse_results
```

```
## # A tibble: 1 x 2
##   method            RMSE
##   <chr>            <dbl>
## 1 Just the average  1.06
```

As expected this model didn't do well and error value is unacceptable. Different users rates same movie very differently depending on their tastes. Also some movies are rated more often. Same difference are expected in different genre of movies. This generates a user specific, movie specific and genre specific effect when the ratings are done. We need to account for these difference in modeling so that the model accounts for the differences when different users rate same movies, or high ratings received for blockbuster movies or a similar kind of bias in genres.

Find the movie bias or movie effects on ratings

```
movie_avgs <- edx %>%
     group_by(movieId) %>%
     summarize(b_i = mean(rating - mu))
```

Find the user bias or user effects on ratings

```
user_avgs <- edx %>%
     left_join(movie_avgs, by='movieId') %>%
     group_by(userId) %>%
     summarize(b_u = mean(rating - mu - b_i))
```

Find the genre bias or genres effects on ratings

```
genres_avgs <- edx %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  group_by(genres) %>%
  summarize(b_gen = mean(rating- mu - b_i - b_u))
```

Mean validation ratings. It is same as the test datset

```
mu_val <- mean(validation$rating)
```

Predict the movie ratings for validation dataset.

```
predicted_ratings <- validation %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  left_join(genres_avgs, by='genres') %>%
  mutate(pred = mu_val + b_i + b_u + b_gen) %>%
  .$pred
```

Model evaluation

```
model_rmse <- RMSE(predicted_ratings,validation$rating)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Movie + User Effects Model + genre effect model",
                                     RMSE = model_rmse ))
rmse_results
```

```
## # A tibble: 2 x 2
##   method                                           RMSE
##   <chr>                                           <dbl>
## 1 Just the average                                1.06
## 2 Movie + User Effects Model + genre effect model 0.865
```

The RMSE has reduced when the user-movie-genre specific effects are introduced in the model.

Lets see how the prediction happened

Get movie titles

```
movie_titles <- edx %>%
  select(movieId, title) %>%
  distinct()
```

See for the top predicted movies on the movie_avgs or the movie effects factor. Look for the number of ratings each movie has

```
all_avgs <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating- mu),
            b_u = mean(rating- mu - b_i),
            b_gen = mean(rating- mu - b_i - b_u))

validation %>%
  dplyr::count(movieId) %>%
  left_join(all_avgs) %>%
  left_join(movie_titles, by="movieId") %>%
  arrange(desc(b_i), desc(b_u), desc(b_gen)) %>%
  select(title, b_i, n) %>%
  slice(1:10) %>%
  knitr::kable()
```

```
## Joining, by = "movieId"
```

title	b_i	n
Hellhounds on My Trail (1999)	1.4875348	1
More (1998)	1.2018205	1
Valerie and Her Week of Wonders (Valerie a tÃ½den divu) (1970)	0.9875348	1
Kansas City Confidential (1952)	0.9875348	1
Shawshank Redemption, The (1994)	0.9426660	3111
Red Desert, The (Deserto rosso, Il) (1964)	0.9042015	1
Godfather, The (1972)	0.9029008	2067
Man Who Planted Trees, The (Homme qui plantait des arbres, L') (1987)	0.8875348	2
Usual Suspects, The (1995)	0.8533885	2389
Schindler's List (1993)	0.8510281	2584

See for the low rating movies in prediction

```
validation %>%
  dplyr::count(movieId) %>%
  left_join(all_avgs) %>%
  left_join(movie_titles, by="movieId") %>%
  arrange(b_i, b_u, b_gen) %>%
  select(title, b_i, n) %>%
  slice(1:10) %>%
  knitr::kable()
```

```
## Joining, by = "movieId"
```

title	b_i	n
Confessions of a Superhero (2007)	-3.012465	1
War of the Worlds 2: The Next Wave (2008)	-3.012465	1
SuperBabies: Baby Geniuses 2 (2004)	-2.717822	5
Disaster Movie (2008)	-2.653090	8
From Justin to Kelly (2003)	-2.610455	17
Criminals (1996)	-2.512465	2
Mountain Eagle, The (1926)	-2.512465	2
When Time Ran Out... (a.k.a. The Day the World Ended) (1980)	-2.512465	2
PokÃ©mon Heroes (2003)	-2.483268	19
Roller Boogie (1979)	-2.479132	2

Noticed that some of the top and low rated movies have few have rated only once. It is rather not a good idea to create a model when movies are predicted based on very few number of ratings. So we try regularization by introducing the parameter 'lambda'. Lambda penalise the values coming from small sample sizes. When n is large the value of lambda is effectively ignored. When value of n is small the values are shrunken towards zero. When lamda is large penality is more. The best value of lambda is the value which gives the prediction of minimum RMSE.

Prediction with regularisation: Series of lambda values are used. Regularisation factor lambda is introduced to userid, movieid and genres. Small estimates in these variables will be penalised. With the help of a function the predictions are done for a series of lambda values and RMSE for those predictions are found out.

Unfortunately it was difficulty to get the knit run on the lambda_rmses function. So the findings are noted here. Scripts can be found in MovieLens.R file. Regularisation gave an RMSE value of 0.864. The top and bottom rated movies were analysed. It was a promising result that all the top rated movies had a minimum number of few 100s ratings. And also that the top movies were actually good ones.