# Movie recommendation system: Predicting movie ratings in the MovieLens dataset

## Introduction

This report describes the development of an algorithm, implemented in the R language (R Core Team 2018), to predict ratings using a subset of the MovieLens dataset. This subset is reffered here as the edx dataset and contains 9.000.055 ratings of 10677 movies by 69878 users, with six columns, 'rating', 'userId', 'movieId', 'timestamp', 'title' and 'genres'. In this analysis, 'rating' is used as the response variable and the others are used as possible predictors. A detailed description of each column and possible columns derived for these six original is given in the next section (Method). The code below were used to extract the basic information from the dataset.

```
# It is assumed tha the tidyverse library is already installed and loaded and that the subset of the Mo

# Number of ratings
n_ratings = nrow(edx)

# Number of movies
n_movies = edx %>%
  group_by(movieId) %>%
  sample_n(1) %>%
  ungroup() %>%
  summarize(n_movies = n())

# Number of users
n_users = edx %>%
  group_by(userId) %>%
  sample_n(1) %>%
  ungroup() %>%
  summarize(n_users = n())

tibble(n_ratings, n_movies, n_users)
```

```
## # A tibble: 1 x 3
##   n_ratings n_movies n_users
##       <int>    <int>   <int>
## 1   9000055    10677   69878
```

```
# Columns
names(edx)
```

```
## [1] "userId"    "movieId"   "rating"    "timestamp" "title"     "genres"
```

The goal of the algorithm is to predict the ratings of different movies by different users, so that it can be used to recommend movies based on users ratings. To do this, the original data was preprocessed and analyzed following the instructions available in (Irizarry 2019)

This report is organized in three additional sections: the Method section describes the steps of data cleaning, data exploration and modeling, the Results and Discussions section describes the analysis of overall accuracy and RMSE of each step, and the Conclusion section brings a summary of the results with limitations and suggestions for future work.

## Method

The columns 'userId' and 'movieId', as the names suggest, contains unique numerical identifications of each user and movie, respectivelly. These columns were used as found in the edx dataset. The column 'title', with the titles and year of release of the movies, was split in two columns, 'title' and 'year'. [The same is true for the 'genres' column, but it is worth noticing that since...]. The column timestamp, with the timestamp of the rating, was used in the creating of five additional columns that enabled better exploratory analysis. From the timestamps, the columns 'rating_year', 'rating_month', 'rating_day', 'rating_wday', 'rating_hour' were created, containing, respectively, the year of the rating, the month of the rating, the day of the month of the rating (1 to 31), the day of the week of the rating (1 to 7), and the hour of the rating (0 to 23). The idea is to explore is the moment of the rating is a predictor of rating.

### Data cleaning

The following code was used to create the additional columns from the timestamps. The column 'years_since_release' was also created, [since older movies tend to be better rated than newer movies].

```r
# Extracts the year of the movie from the column 'title'.
# The pattern matches four number inside parenthesis and the capture group isolates only the numbers.
edx_small = edx_small %>%
  mutate(
    year = str_match(title, '\\((\\d{4})\\)')[,2] %>% as.numeric(),
    time = as_datetime(timestamp),
    rating_year = year(time),
    rating_month = month(time),
    rating_day = day(time),
    rating_wday = wday(time),
    rating_hour = hour(time),
    years_since_release = rating_year - year)
```

From the 'genres' column, first a list of all the genres was created. Then, for each genre, a column was created with a binary variable representing whether a given movie if of a particular genre or not.

```r
genres = edx %$%
  genres %>% str_split('\\|') %>% unlist() %>% unique()

genres
```

```
##  [1] "Comedy"             "Romance"            "Action"
##  [4] "Crime"              "Thriller"           "Drama"
##  [7] "Sci-Fi"             "Adventure"          "Children"
## [10] "Fantasy"            "War"                "Animation"
## [13] "Musical"            "Western"            "Mystery"
## [16] "Film-Noir"          "Horror"             "Documentary"
## [19] "IMAX"               "(no genres listed)"
```

```r
for (g in genres) {
  genre = str_glue('is_', g)

  edx_small = edx_small %>%
    mutate(!!genre := ifelse(str_detect(.$genres, g), 1, 0))
}
```

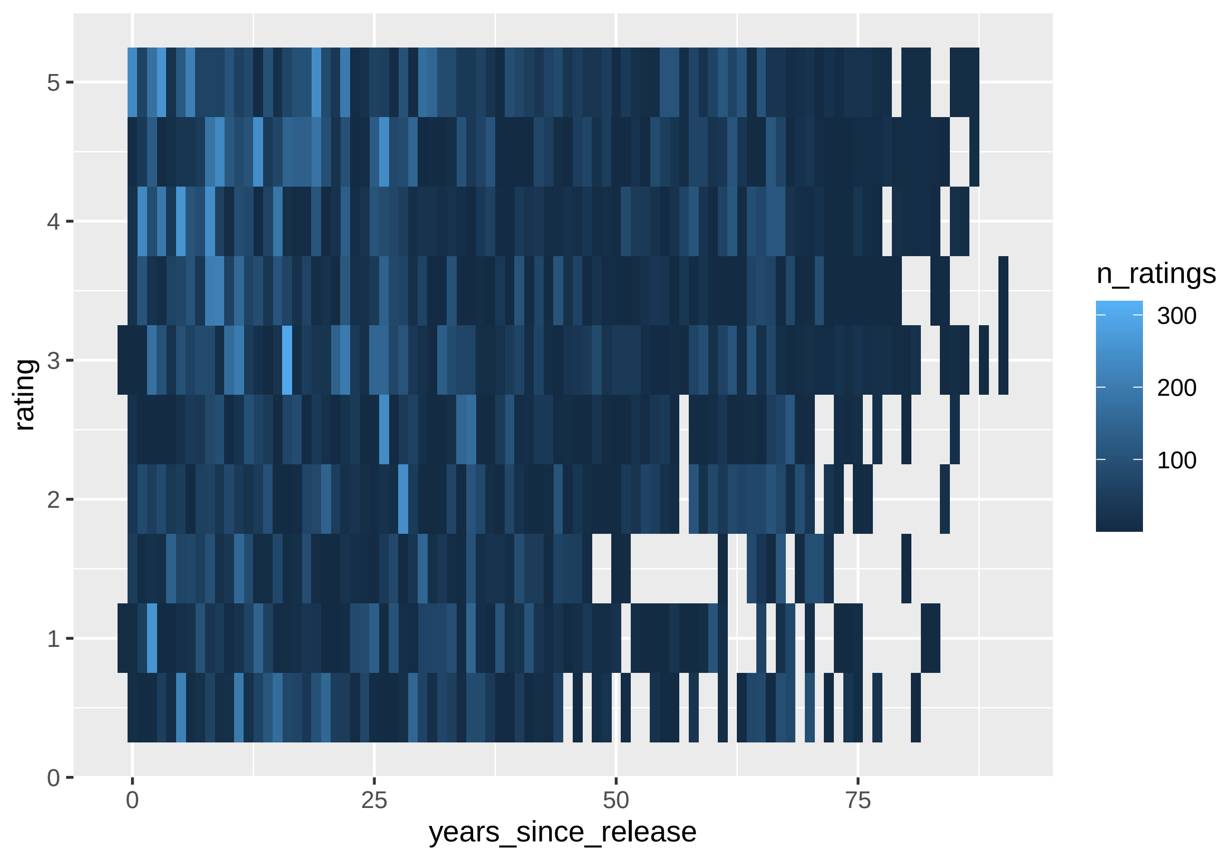
```
edx_small %>%
  select(-c(userId, movieId, rating, timestamp, title, genres, year, time, rating_year, rating_month, ra
  colSums() %>%
  sort(decreasing = T)
```

```
##              is_Drama             is_Comedy             is_Action
##                 39101                 35495                 25669
##           is_Thriller          is_Adventure            is_Romance
##                 23089                 18998                 17121
##              is_Crime             is_Sci-Fi            is_Fantasy
##                 13443                 13321                  9334
##           is_Children             is_Horror            is_Mystery
##                  7429                  6951                  5687
##                is_War          is_Animation            is_Musical
##                  5041                  4697                  4419
##            is_Western          is_Film-Noir        is_Documentary
##                  1993                  1144                   960
##               is_IMAX is_(no genres listed)
##                    59                     0
```

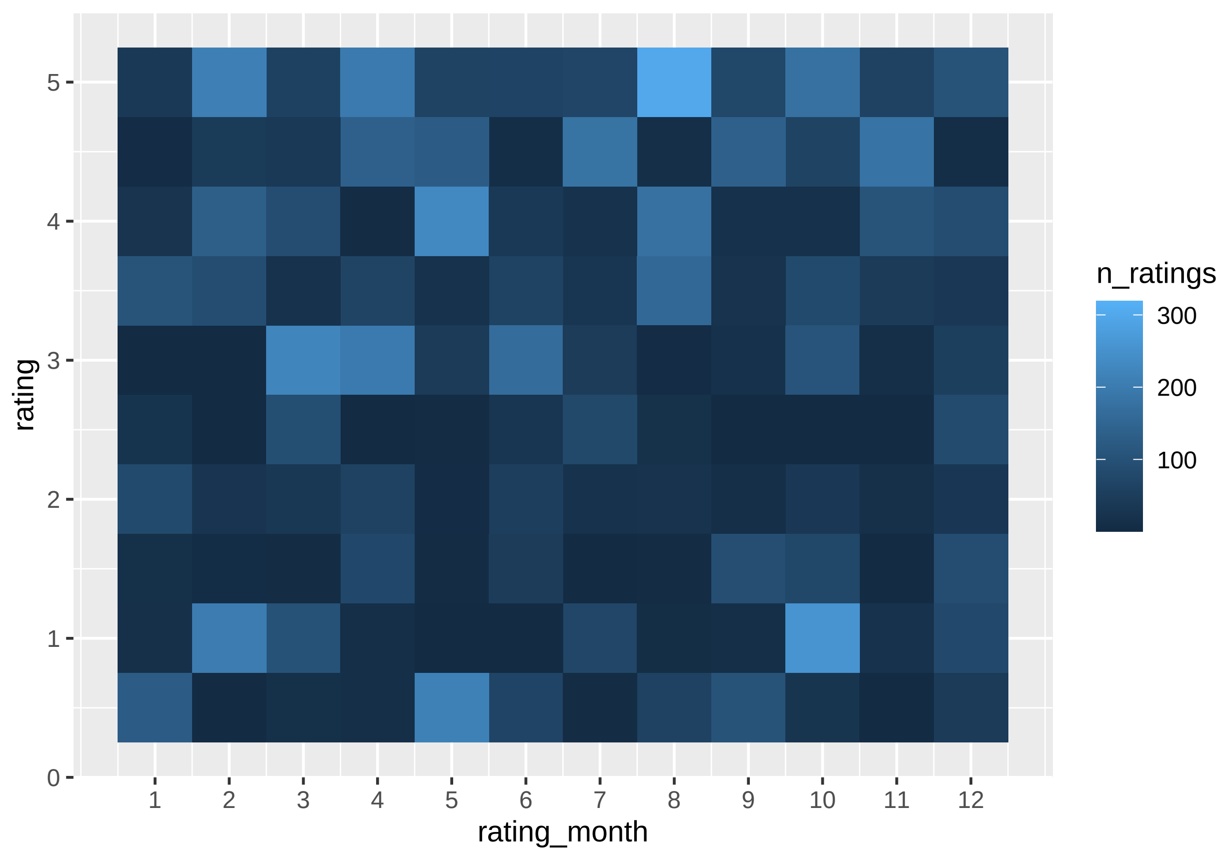
## Data exploration

[Ratings are apparently not affected by the month, day of the month or day of the week of the rating. However, ratings are clearly affected by years since release and seems to be slightly affected by the hour of rating, with a reduction of ratings around 9 hours. [Reminder: check if this affects the proportion of ratings]]

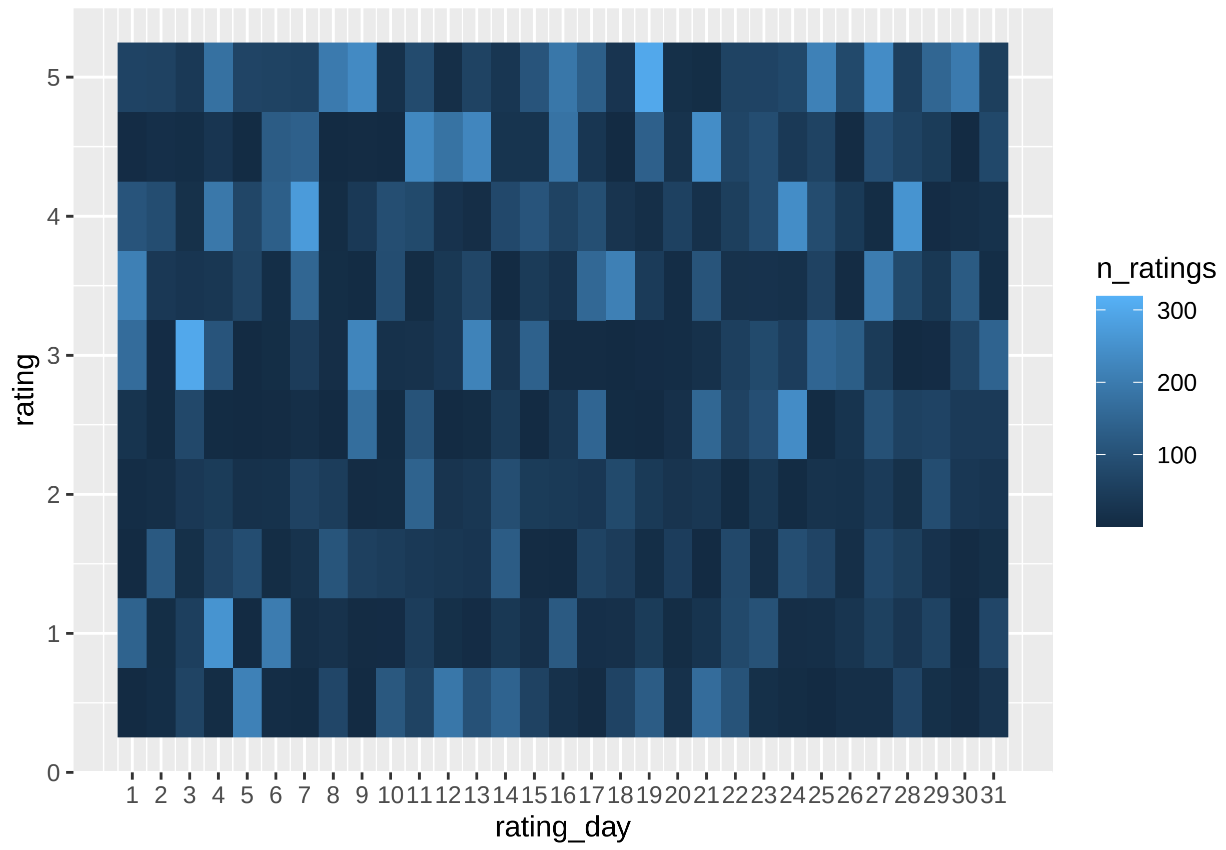
```
edx_small = edx_small %>%
  group_by(movieId) %>%
  mutate(n_ratings = n()) %>%
  ungroup()

edx_small %>%
  ggplot(aes(years_since_release, rating, fill = n_ratings)) +
  geom_tile()
```



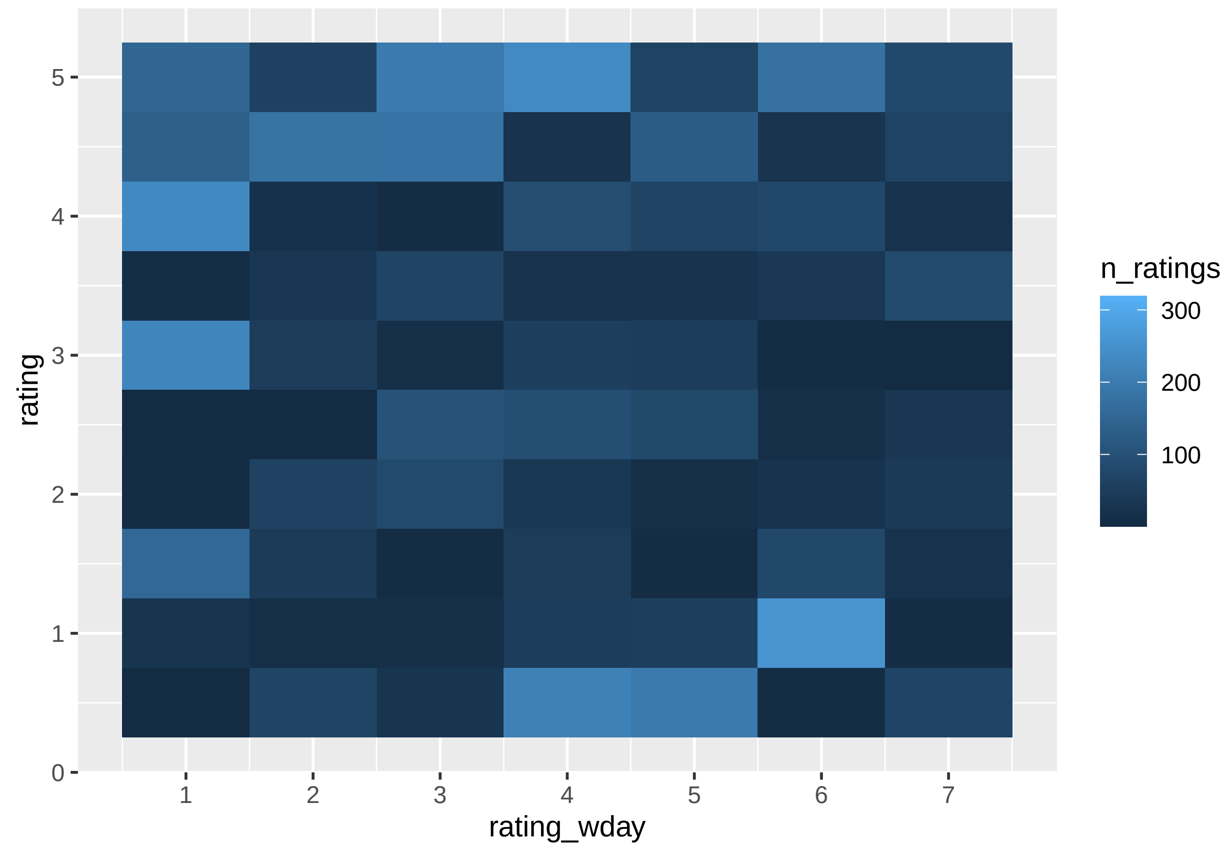
```r
edx_small %>%
  ggplot(aes(rating_month, rating, fill = n_ratings)) +
  geom_tile() +
  scale_x_continuous(breaks = 1:12)
```



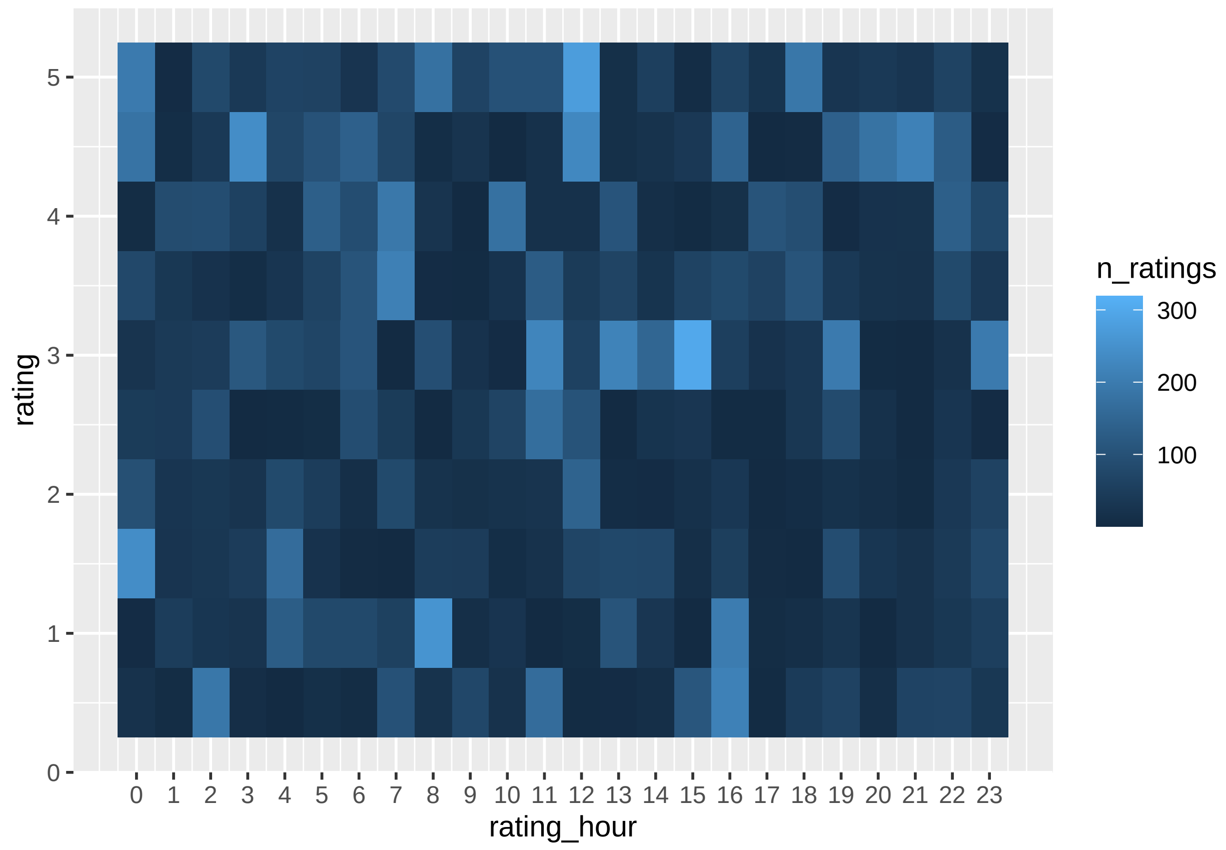
```
edx_small %>%
  ggplot(aes(rating_day, rating, fill = n_ratings)) +
  geom_tile() +
  scale_x_continuous(breaks = 1:31)
```



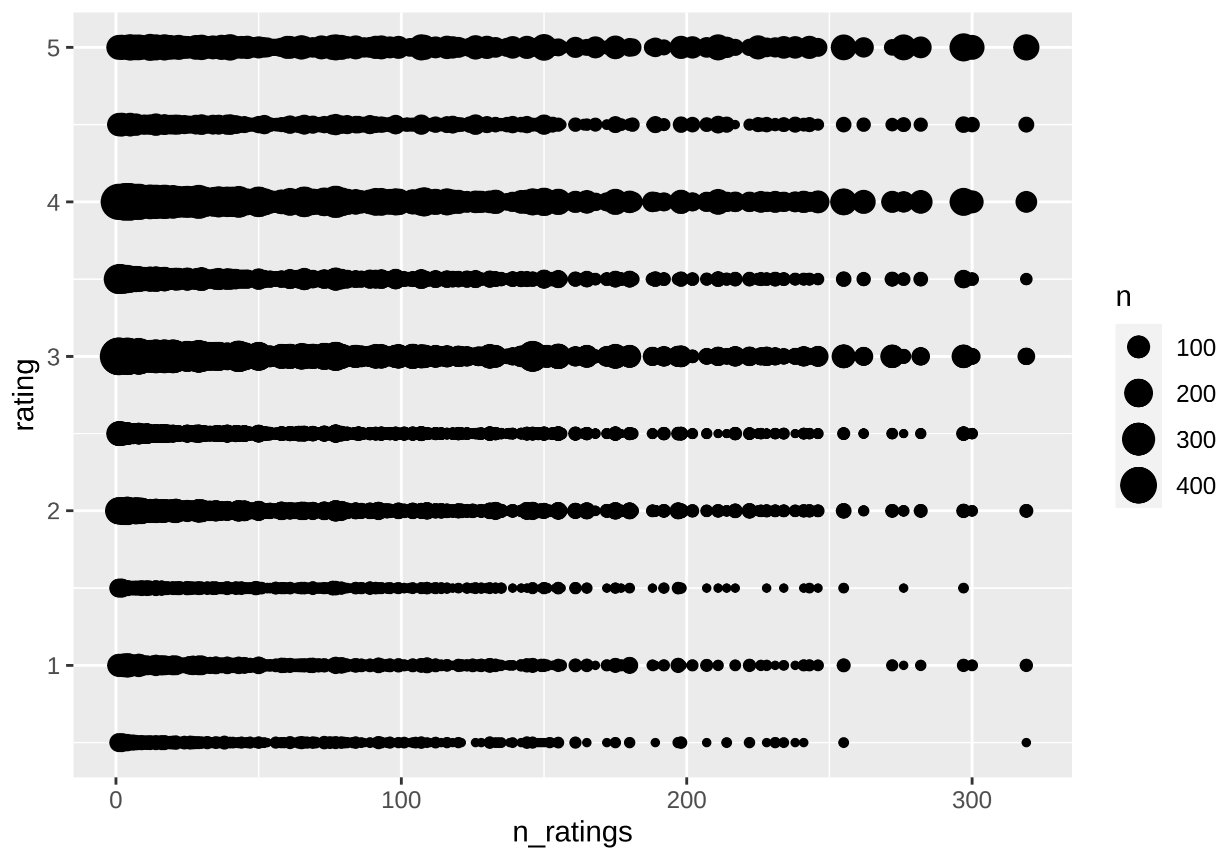
```
edx_small %>%
  ggplot(aes(rating_wday, rating, fill = n_ratings)) +
  geom_tile() +
  scale_x_continuous(breaks = 1:7)
```



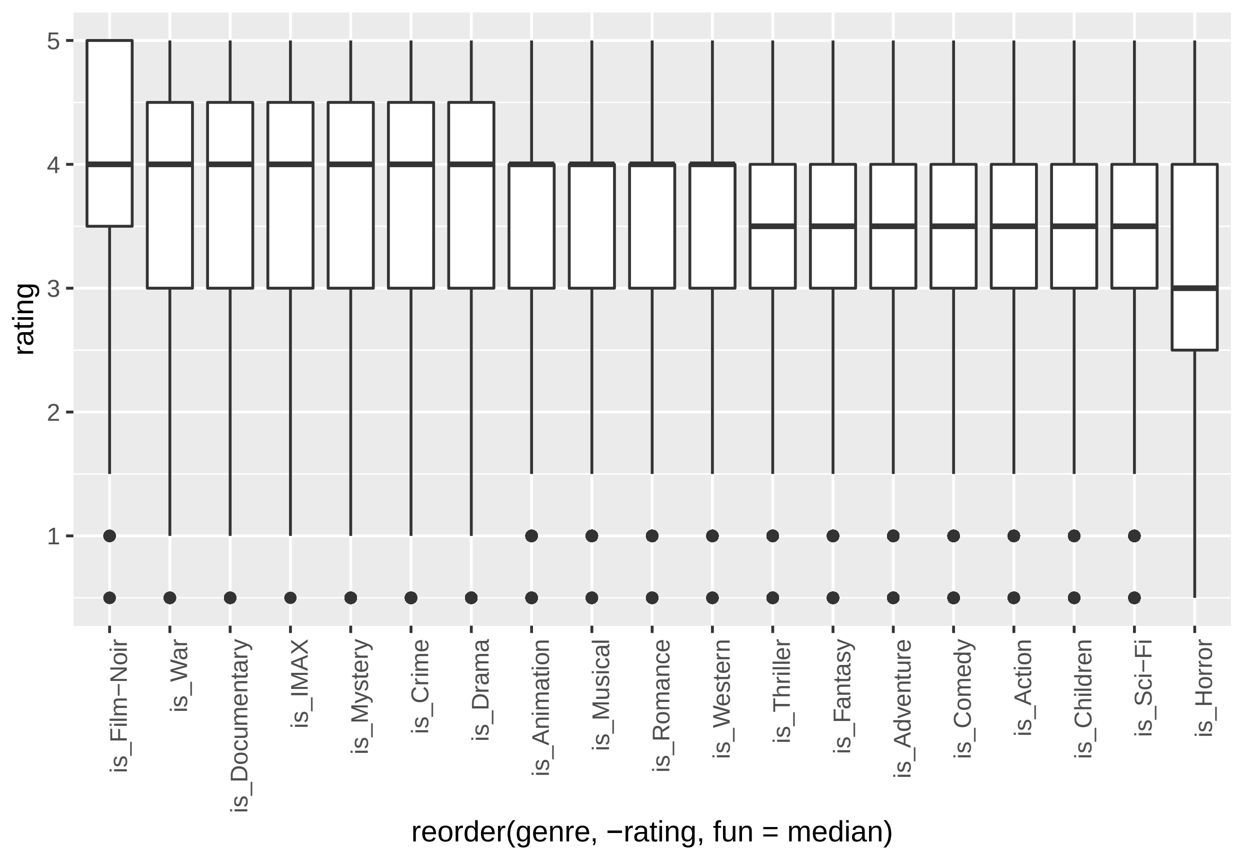
```r
edx_small %>%
  ggplot(aes(rating_hour, rating, fill = n_ratings)) +
  geom_tile() +
  scale_x_continuous(breaks = 0:23)
```



```
edx_small = edx_small %>%
  group_by(movieId) %>%
  mutate(n_ratings = n()) %>%
  ungroup()

edx_small %>%
  select(rating, n_ratings) %>%
  ggplot(aes(n_ratings, rating)) +
  geom_count()
```



```r
edx_small %>%
  select(rating, starts_with('is_')) %>%
  pivot_longer(-rating, names_to = 'genre', values_to = 'is_genre') %>%
  filter(is_genre == 1) %>%
  ggplot(aes(reorder(genre, -rating, fun = median), rating)) +
  geom_boxplot() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```



## Modeling

```r
set.seed(2020)

test_index = createDataPartition(y = edx_small$rating, times = 1, p = .1, list = FALSE) %>% as.vector()

train_set = edx_small[-test_index,]
test_set = edx_small[test_index,]

# Remove movies and users that do not appear in the training set from the test set
test_set = test_set %>%
  semi_join(train_set, by = 'movieId') %>%
  semi_join(train_set, by = 'userId')
```

*First model: average*: $Y_{u,i} = \mu + \epsilon_{u,i}$

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

# First model - the average of ratings
RMSE(test_set$rating, mean(train_set$rating))
```

```
## [1] 1.06
```

*Second model: movie effect*: $Y_{u,i} = \mu + m_i + \epsilon_{u,i}$ with $m_i = \frac{1}{N_i}\sum_i^{N_i} Y_{u,i} - \hat{\mu}$.

```
m_i = edx_small %>%
  group_by(movieId) %>%
  summarise(m_i = mean(rating - mean(edx_small$rating))) %>%
  .$m_i
```

## Results and Discussions

### Results

### Discussions

## Conclusion

### Limitations and suggestions for future work

## References

Irizarry, Rafael A. 2019. *Introduction to Data Science: Data Analysis and Prediction Algorithms with R*. Leanpub. https://leanpub.com/datasciencebook.

R Core Team. 2018. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. https://www.R-project.org.