HarvardX: PH125.9x Data Science: Capstone Project

Movielens Recommender System Project Report

Wilfredo A. de Vera

June 12, 2020

1. Introduction

The objective of this project is to build recommender system models using the movielens dataset in partial fulfilment of the Harvardx Data Science Capstone course PH125.9x.

Subsequent to the wrangling and cleaning of the edx and validation sets, models were developed and trained using selected variables from the train_set, which comprises 70% of the wrangled edx dataset. These were then tested on the test_set (comprising 30% of wrangled edx) and eventually on the validation set, with the following dimensions indicated below:

train_set: 16,359,990 obs. 11 variables

test_set : 7,011,381 obs. 11 variables

validation: 2,595,763 obs. 11 variables

The goal is to determine the algorithm that yields the least root mean squared error (RMSE) following the equation below:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{t=1}^{n}(\hat{y}_{u,i} - y_{u,i})^2}$$

Eighteen models were built and the RMSE assessed on both the test and validation sets. These models were similar to those indicated in Section 33.7 of the Introduction to Data Science Book (https://rafalab.github.io/dsbook/); albeit, with the addition of other independent variables such as genres, weekday_rated, year_released, and year_rated. Note that these are aside from userId and movieId already used as variables in the book. In addition, stats's linear regression (lm) model was included. Furthermore and out of curiousity, random forests, generalized linear model (glm), deep neural network, and gradient boosting machine (gbm) from the h2o package were explored and tested on the datasets.

Finally, the "Regularized movie + user model" at lambda = 27.5 was determined to be the best algorithm that yielded the least RMSE of 0.8571358019 on the validation set.

2. Analysis

2.1 Data wrangling

The original edx and validation datasets were first generated containing 6 variables with 9,000,055 obs. and 999,999 obs., respectively. This was the result of running the script provided by Harvardx. These datasets were then wrangled and cleaned by:

a.) extracting year_rated, month_rated, day_rated, weekday_rated, and wday_rated from original timestamp variable;

b.) separating genres from original genres variable which were separated by pipe (|) to create new rows;

c.) separating year_released from original title which is of the format "title (yyyy)"; and

d.) removing specific observations for genres that indicated "(no genres listed)". There were only about 7 observations in edx and none in validation.

After wrangling and cleaning, the edx and validation datasets grew to 11 variables containing 23,371,416 obs. and 2,595,763 obs., respectively.

2.2 Exploratory data analysis (EDA)

The EDA was performed on both the edx and validation datasets in terms of generating summary statistics, visualization, checking correlation, principal components, and variable importance:

2.2.1 Generate numerical and character summary statistics

```
## [1] "edx statistical summary"
```

```
##                     n     mean       sd   max    min   range nunique nzeros
## userId       23371416 35885.68 20588.42 71567    1.0 71566.0   69878      0
## movieId      23371416  4277.29  9331.20 65133    1.0 65132.0   10676      0
## rating       23371416     3.53     1.05     5    0.5     4.5      10      0
## year_rated   23371416  2002.28     3.75  2009 1995.0    14.0      15      0
## month_rated  23371416     6.79     3.53    12    1.0    11.0      12      0
## day_rated    23371416    15.61     8.80    31    1.0    30.0      31      0
## wday_rated   23371416     3.91     1.95     7    1.0     6.0       7      0
## year_released 23371416 1990.43    13.61  2008 1915.0    93.0      94      0
##                 iqr lowerbound upperbound noutlier kurtosis skewness  mode miss
## userId        35498   -35107.0   106885.0        0  -1.1936  0.00747 59269    0
## movieId        3019    -3912.5     8163.5  1619908  17.7460  4.22158   356    0
## rating            1        1.5        5.5  1060268   0.0405 -0.60245     4    0
## year_rated        5     1992.5     2012.5        0  -1.1256 -0.15637  2000    0
## month_rated       6       -5.0       19.0        0  -1.2470 -0.09512    11    0
## day_rated        15      -14.5       45.5        0  -1.1884  0.01544    20    0
## wday_rated        4       -4.0       12.0        0  -1.1965  0.08068     3    0
## year_released    11     1970.5     2014.5  2008273   4.4115 -2.00068  1995    0
##               miss%   1%     5%   25%   50%   75%   95%   99%
## userId            0  762 3798.0 18140 35784 53638 68087 70904
## movieId           0   10  107.0   616  1748  3635  8984 53129
## rating            0    1    1.5     3     4     4     5     5
## year_rated        0 1996 1996.0  2000  2003  2005  2008  2008
## month_rated       0    1    1.0     4     7    10    12    12
## day_rated         0    1    2.0     8    16    23    29    31
## wday_rated        0    1    1.0     2     4     6     7     7
## year_released     0 1939 1960.0  1987  1995  1998  2004  2007
```

```
## [1] "edx character summary"
```

```
##                     n miss miss% unique
## title        23371416    0     0  10406
```

```
## genres        23371416    0     0     19
## weekday_rated 23371416    0     0      7
```

```
## [1] "validation statistical summary"
```

```
##                     n     mean       sd   max    min   range nunique nzeros
## userId        2595771 35899.41 20585.25 71567    1.0 71566.0   68534      0
## movieId       2595771  4269.67  9307.36 65133    1.0 65132.0    9809      0
## rating        2595771     3.53     1.05     5    0.5     4.5      10      0
## year_rated    2595771  2002.28     3.74  2009 1995.0    14.0      15      0
## month_rated   2595771     6.78     3.53    12    1.0    11.0      12      0
## day_rated     2595771    15.61     8.79    31    1.0    30.0      31      0
## wday_rated    2595771     3.90     1.95     7    1.0     6.0       7      0
## year_released 2595771  1990.41    13.63  2008 1915.0    93.0      94      0
##                 iqr lowerbound upperbound noutlier kurtosis skewness  mode miss
## userId        35513   -35132.5   106919.5        0   -1.193  0.00572 59269    0
## movieId        3024    -3925.0     8171.0   179641   17.788  4.22487   356    0
## rating            1        1.5        5.5   118062    0.043 -0.60364     4    0
## year_rated        5     1992.5     2012.5        0   -1.125 -0.15585  2000    0
## month_rated       6       -5.0       19.0        0   -1.246 -0.09499    11    0
## day_rated        15      -14.5       45.5        0   -1.186  0.01462    11    0
## wday_rated        4       -4.0       12.0        0   -1.198  0.08042     3    0
## year_released    11     1970.5     2014.5   223978    4.394 -1.99850  1995    0
##               miss%   1%     5%   25%   50%   75%   95%   99%
## userId            0  782 3795.0 18137 35828 53650 68085 70905
## movieId           0   10  107.0   611  1734  3635  8984 53125
## rating            0    1    1.5     3     4     4     5     5
## year_rated        0 1996 1996.0  2000  2003  2005  2008  2008
## month_rated       0    1    1.0     4     7    10    12    12
## day_rated         0    1    2.0     8    16    23    29    31
## wday_rated        0    1    1.0     2     4     6     7     7
## year_released     0 1939 1959.0  1987  1995  1998  2004  2007
```

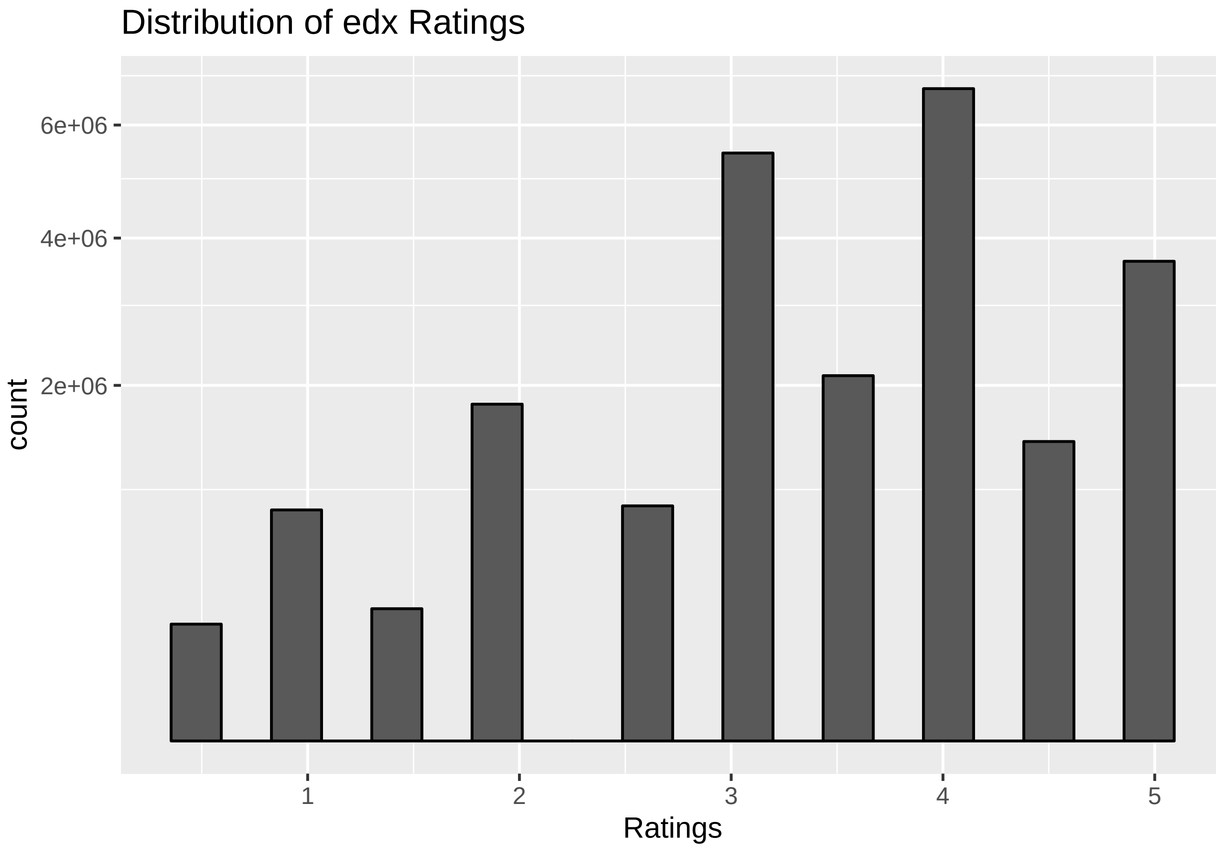
```
## [1] "validation character summary"
```

```
##                     n miss miss% unique
## title         2595771    0     0   9557
## genres        2595771    0     0     19
## weekday_rated 2595771    0     0      7
```

2.2.2 Visualization

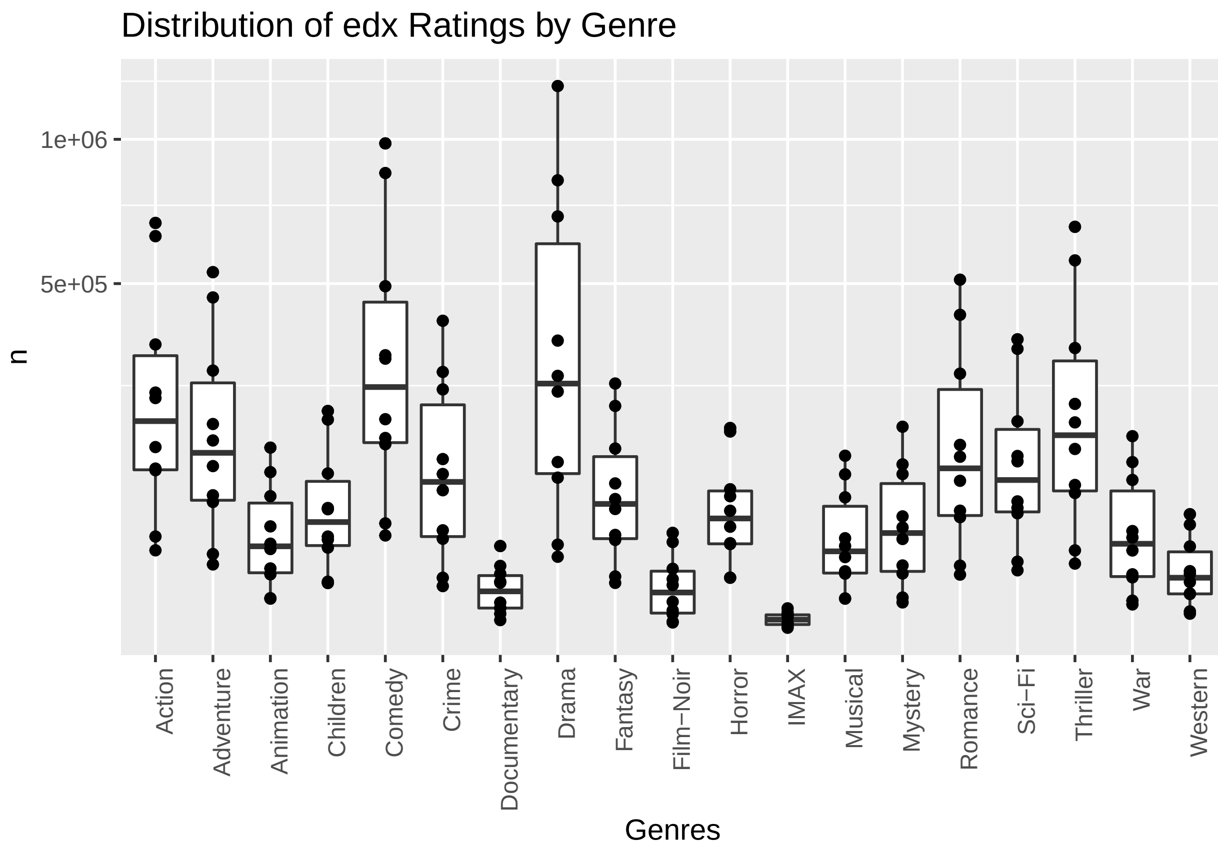
The following charts pertain to the distribution of rating vs. genres, year_released, year_rated, month_rated, day_rated, weekday_rated, movieId, and userId:

2.2.2.1 Distribution of edx ratings



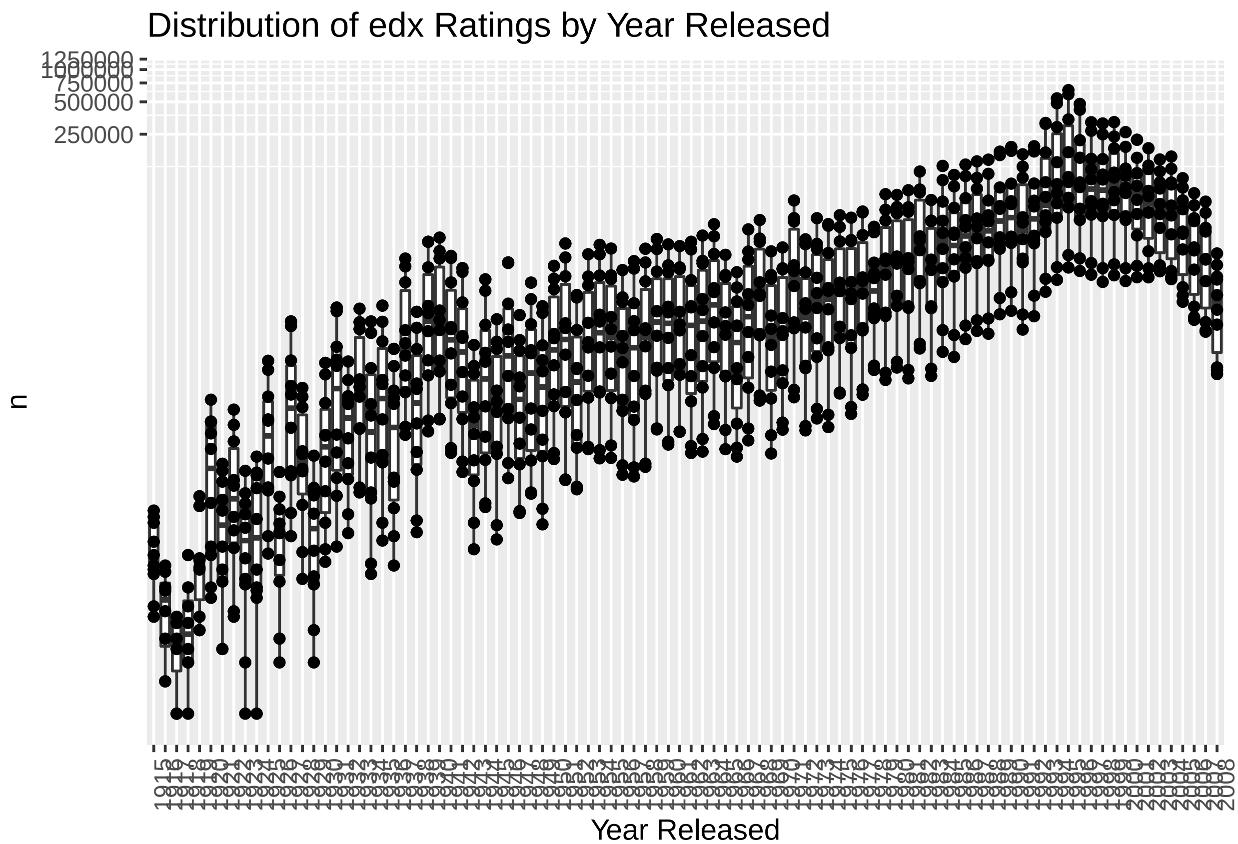
The dependent variable, ratings, on the edx dataset does not follow a normal distribution.

2.2.2.2 Distribution of edx ratings by genre



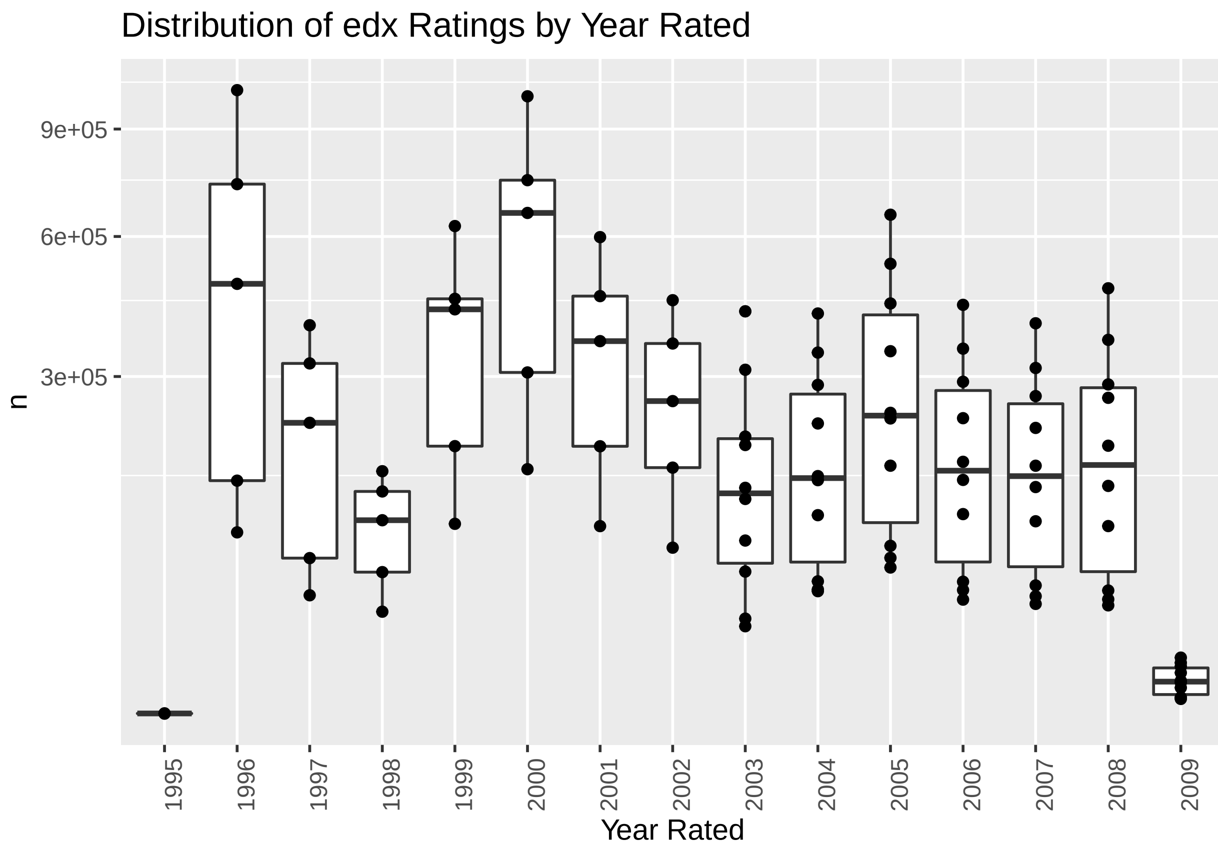
Note that Drama, Comedy, Action, Thriller, and Adventure top the genres.

2.2.2.3 Distribution of edx ratings by year released



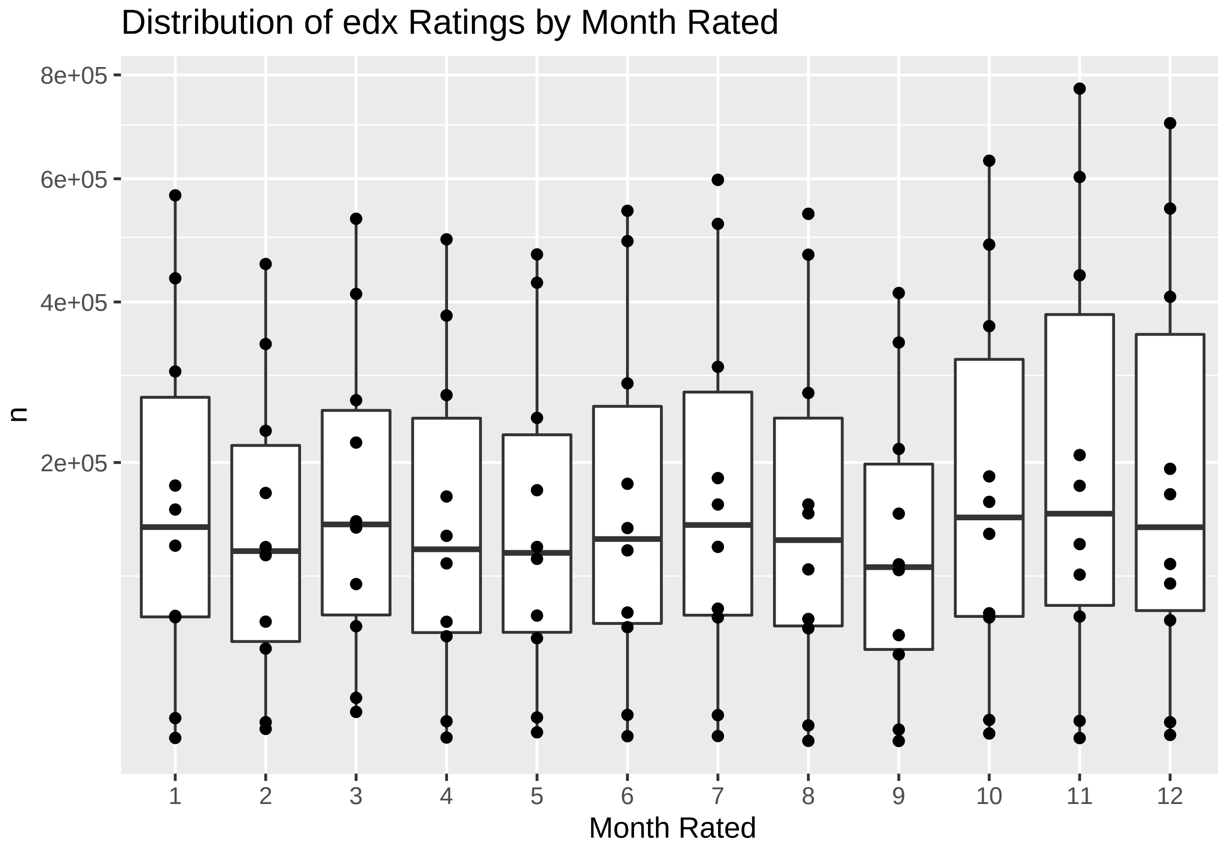
There were many ratings made for movies released between 1994 and 1999, with peak at 1995.

2.2.2.4 Distribution of edx ratings by year rated



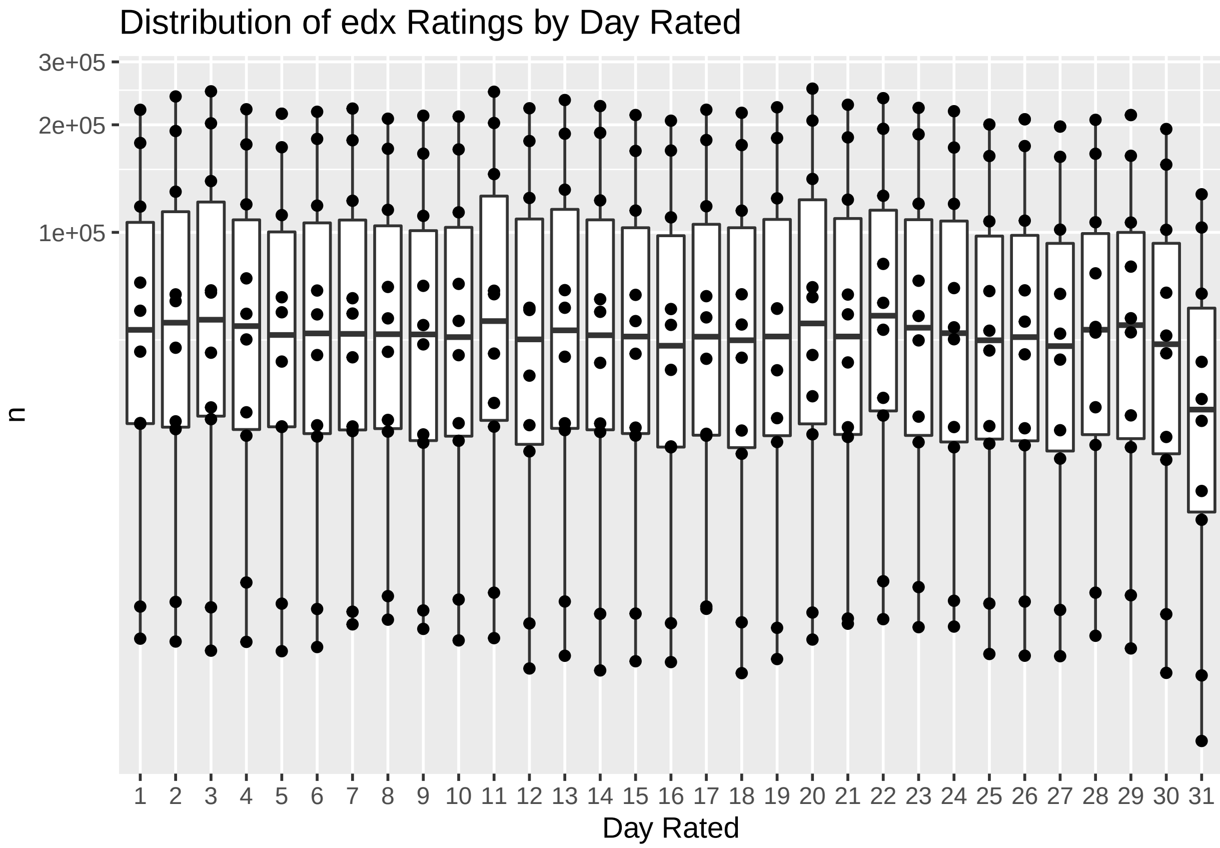
The year 2000 was the year when there were many ratings made.

2.2.2.5 Distribution of edx ratings by month rated



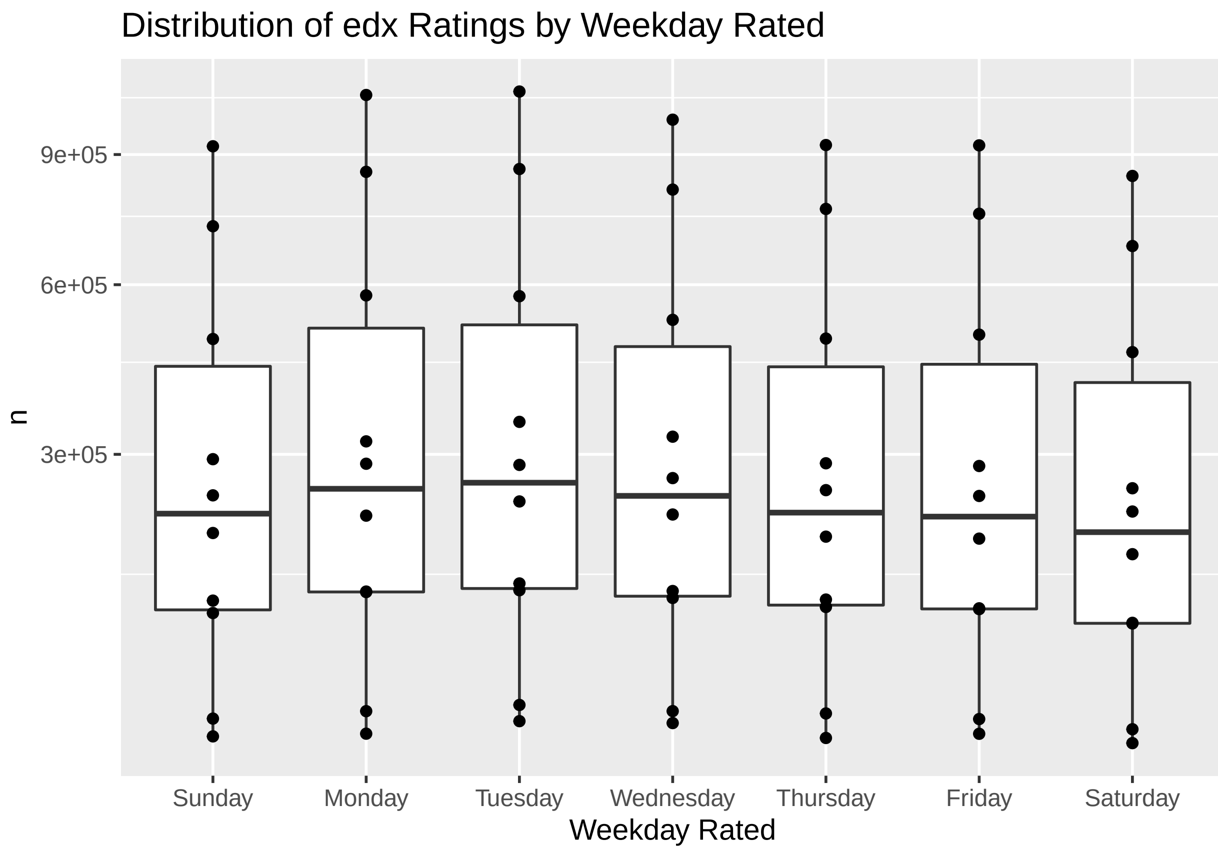
It may be observed here that there were many ratings made between October to December, with peak at around November.

2.2.2.6 Distribution of edx ratings by day rated



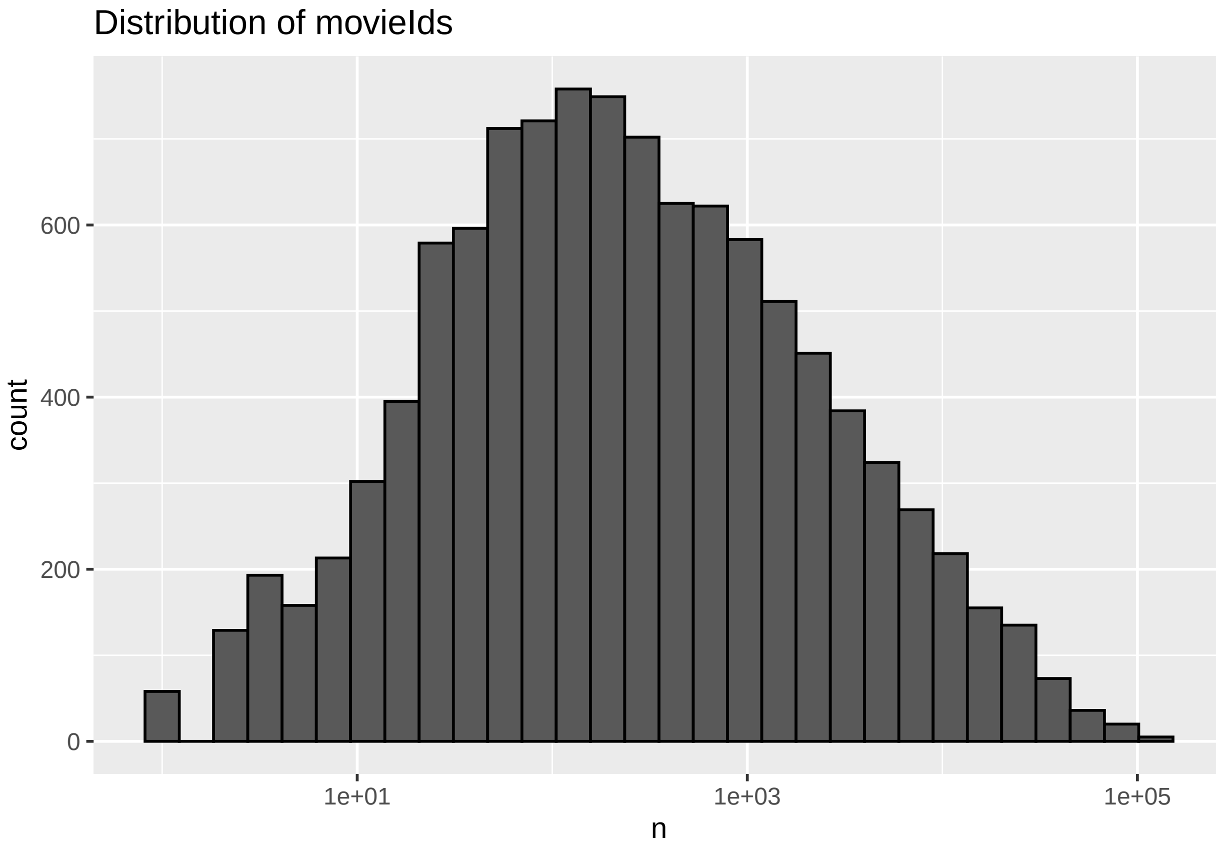
It appears that the number of ratings peaked around the 20th day of the month.

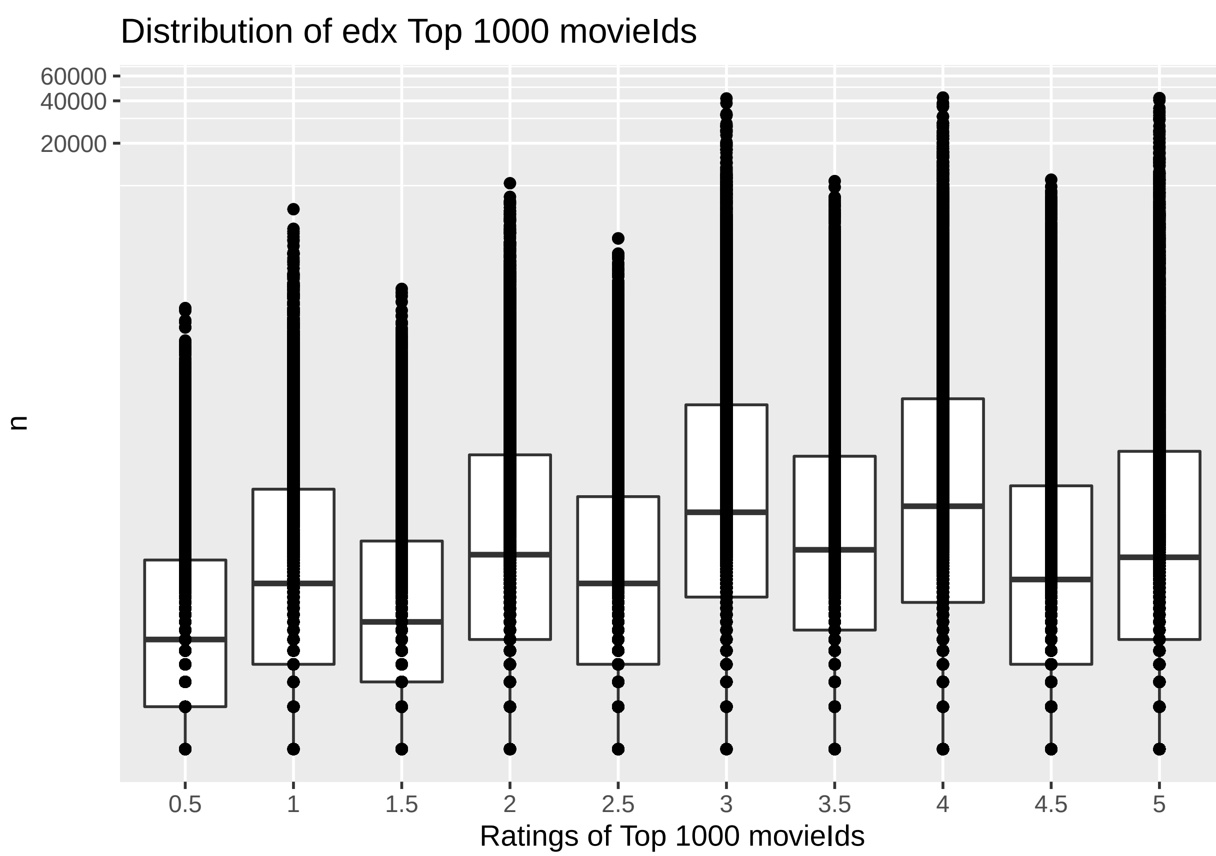
2.2.2.7 Distribution of edx ratings by weekday rated



It appears that there are more ratings made around Tuesday of the week.

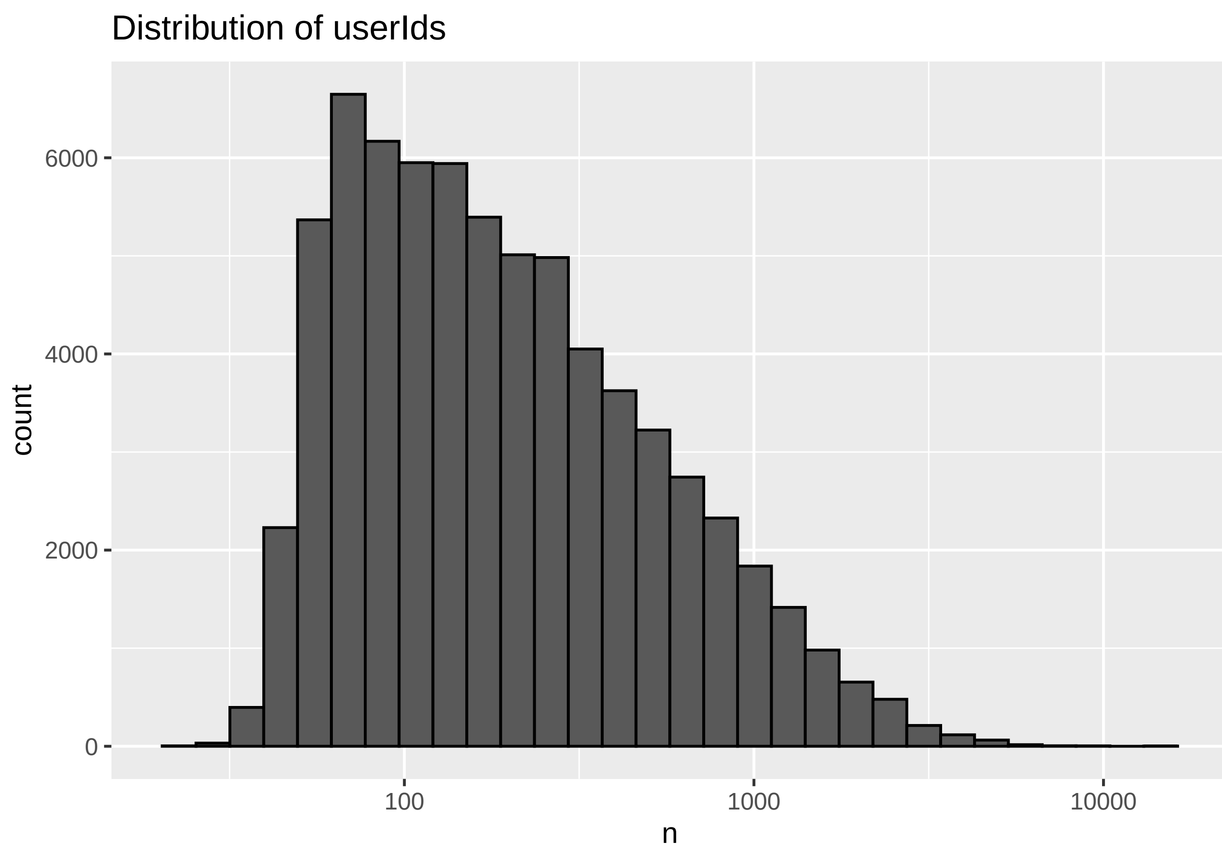
2.2.2.8 Distribution of edx ratings of Top 1000 movieIds





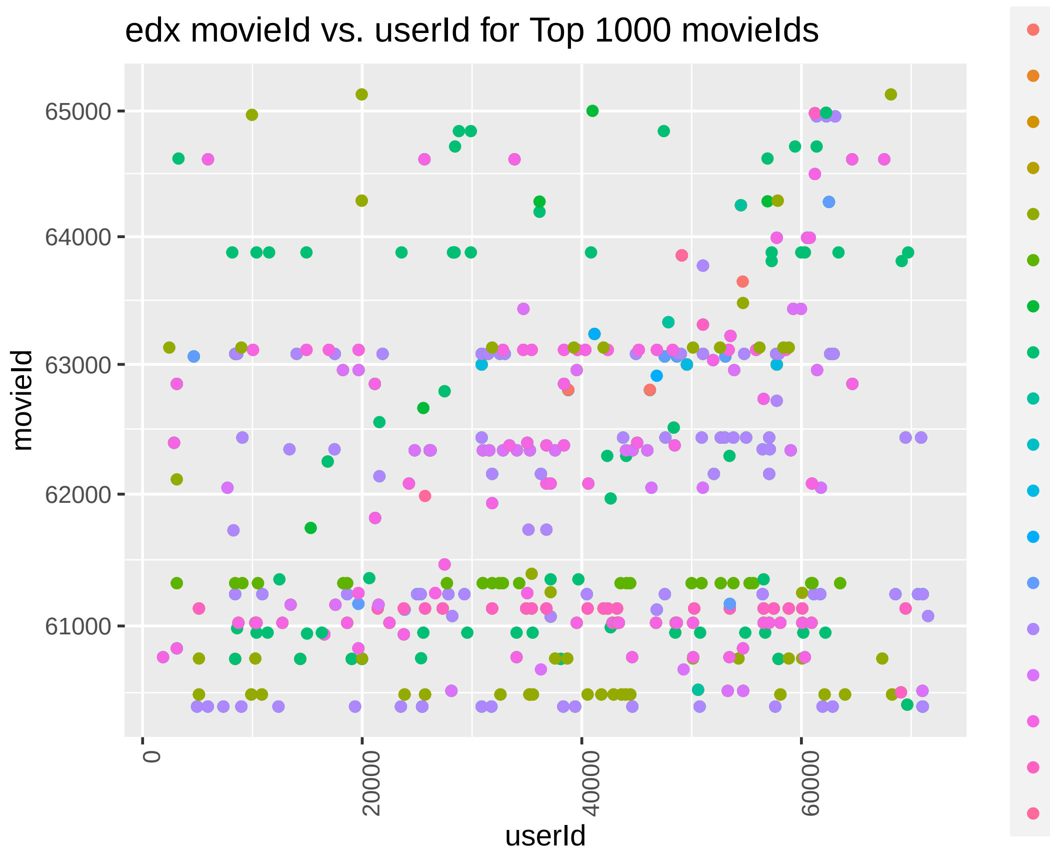
As depicted in the histogram, edx movieId somehow follows a normal distribution, and rating of 3 and 4 are prevalent in top 1000 movies.

2.2.2.9 Distribution of edx ratings of Top 1000 userIds



As depicted in the histogram, edx userId is skewed to the right.

2.2.2.10 Plot of movieId vs. userId for Top 1000 movieIds rated 5



There seems to be no obvious trend here, but it appears that movieIds ranging from 1 to 61500 were rated 5 by most users.

2.2.3 Correlation

This is to check the edx dataset if correlation exists between the independent variables: userId, movieId, genres, year_released, year_rated, month_rated, day_rated, and weekday_rated.

```
##                   userId   movieId    genres year_rated month_rated day_rated
## userId          1.000000  0.004413 -0.000564     0.0159    -0.02905   0.02318
## movieId         0.004413  1.000000 -0.006925     0.3740    -0.00609   0.00963
## genres         -0.000564 -0.006925  1.000000    -0.0140     0.00299  -0.00140
## year_rated      0.015904  0.374036 -0.013970     1.0000    -0.16044   0.01656
## month_rated    -0.029053 -0.006093  0.002986    -0.1604     1.00000   0.01833
## day_rated       0.023182  0.009634 -0.001400     0.0166     0.01833   1.00000
## weekday_rated   0.019726  0.000641 -0.002218     0.0223    -0.00355   0.02626
## wday_rated     -0.008260 -0.011325  0.000346    -0.0207    -0.00907  -0.01313
## year_released   0.000150  0.257266 -0.040684     0.1101    -0.02284   0.00831
##                weekday_rated wday_rated year_released
## userId              0.019726  -0.008260       0.00015
## movieId             0.000641  -0.011325       0.25727
## genres             -0.002218   0.000346      -0.04068
## year_rated          0.022295  -0.020732       0.11007
```

```
## month_rated       -0.003549  -0.009067      -0.02284
## day_rated          0.026258  -0.013130       0.00831
## weekday_rated      1.000000  -0.188748       0.00708
## wday_rated        -0.188748   1.000000      -0.00625
## year_released      0.007082  -0.006252       1.00000
```

It is apparent that a very slight positive correlation exists between movieId vs. year_rated and year_released at 0.374036 and 0.257266, respectively.

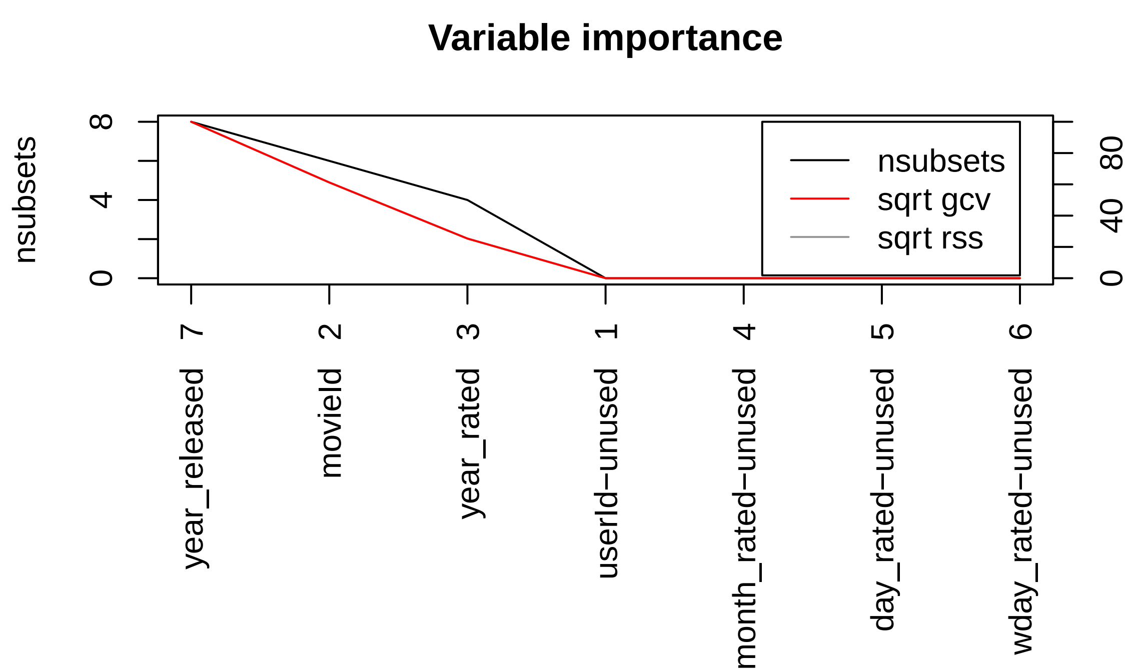
2.2.4 Principal components analysis: edx

```
## Importance of components:
##                          PC1   PC2   PC3   PC4   PC5   PC6    PC7   PC8   PC9
## Standard deviation     1.240 1.092 1.025 1.008 1.001 0.981 0.9280 0.900 0.753
## Proportion of Variance 0.171 0.133 0.117 0.113 0.111 0.107 0.0957 0.090 0.063
## Cumulative Proportion  0.171 0.304 0.420 0.533 0.644 0.751 0.8470 0.937 1.000
```

The first 7 components in edx dataset account for 84.702% of the variability.

2.2.5 Variable importance

Using the earth package, the edx variables year_released, movieId, and year_rated were determined to be important. (Note: running varimp may take a longer while esp. in ordinary 8 GB machines.)



```
## Selected 9 of 9 terms, and 3 of 7 predictors
## Termination condition: RSq changed by less than 0.001 at 9 terms
## Importance: year_released, movieId, year_rated, userId-unused, ...
## Number of terms at each degree of interaction: 1 8 (additive model)
## GCV 0.975    RSS 22789392    GRSq 0.0249    RSq 0.0249
```

2.3 Generate train_set and test_set from edx

The wrangled edx dataset contains 23,371,416 obs. of 11 variables. This was then split into 70%-30% proportions corresponding to train_set and test_set, respectively.

```
## [1] "train_set"
```

```
## 'data.frame':    16359990 obs. of  11 variables:
##  $ userId       : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ movieId      : num  122 122 185 185 185 292 292 292 316 316 ...
##  $ title        : chr  "Boomerang" "Boomerang" "Net, The" "Net, The" ...
##  $ rating       : num  5 5 5 5 5 5 5 5 5 5 ...
##  $ genres       : Factor w/ 19 levels "Action","Adventure",..: 5 15 1 6 17 1 8 16 1 2 ...
##  $ year_rated   : num  1996 1996 1996 1996 1996 ...
##  $ month_rated  : num  8 8 8 8 8 8 8 8 8 8 ...
##  $ day_rated    : num  2 2 2 2 2 2 2 2 2 2 ...
##  $ weekday_rated: Factor w/ 7 levels "Friday","Monday",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ wday_rated   : num  6 6 6 6 6 6 6 6 6 6 ...
##  $ year_released: num  1992 1992 1995 1995 1995 ...
```

```
## [1] "test_set"
```

```
## 'data.frame':    7011381 obs. of  11 variables:
##  $ userId       : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ movieId      : num  292 362 362 364 364 377 377 466 466 588 ...
##  $ title        : chr  "Outbreak" "Jungle Book, The" "Jungle Book, The" "Lion King, The" ...
##  $ rating       : num  5 5 5 5 5 5 5 5 5 5 ...
##  $ genres       : Factor w/ 19 levels "Action","Adventure",..: 17 2 4 2 3 15 17 1 5 3 ...
##  $ year_rated   : num  1996 1996 1996 1996 1996 ...
##  $ month_rated  : num  8 8 8 8 8 8 8 8 8 8 ...
##  $ day_rated    : num  2 2 2 2 2 2 2 2 2 2 ...
##  $ weekday_rated: Factor w/ 7 levels "Friday","Monday",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ wday_rated   : num  6 6 6 6 6 6 6 6 6 6 ...
##  $ year_released: num  1995 1994 1994 1994 1994 ...
```

```
## [1] "validation"
```

```
## 'data.frame':    2595763 obs. of  11 variables:
##  $ userId       : num  1 1 1 1 1 1 1 2 2 2 ...
##  $ movieId      : num  231 480 480 480 480 586 586 151 151 151 ...
##  $ title        : chr  "Dumb & Dumber" "Jurassic Park" "Jurassic Park" "Jurassic Park" ...
##  $ rating       : num  5 5 5 5 5 5 5 3 3 3 ...
##  $ genres       : Factor w/ 19 levels "Action","Adventure",..: 5 1 2 16 17 4 5 1 8 15 ...
##  $ year_rated   : num  1996 1996 1996 1996 1996 ...
##  $ month_rated  : num  8 8 8 8 8 8 8 7 7 7 ...
##  $ day_rated    : num  2 2 2 2 2 2 2 7 7 7 ...
```

```
##  $ weekday_rated: Factor w/ 7 levels "Friday","Monday",..: 1 1 1 1 1 1 1 2 2 2 ...
##  $ wday_rated   : num  6 6 6 6 6 6 6 2 2 2 ...
##  $ year_released: num  1994 1993 1993 1993 1993 ...
```

3. Methods

With rating as the dependent variable, models will be built using the following independent variables:

a.) movieId
b.) userId
c.) genres
d.) weekday_rated
e.) year_rated
f.) year_released

These independent variables will be gradually added to the models starting with 'movieId', then 'movieId + userId', then 'movieId + userId + genres', … and so on and so forth, on both the non-regularized and regularized models.

The following eighteen models will be built and trained on the train_set with 16,359,990 obs. of 11 variables:

a.) Just the mean (naive) model
b.) Non-regularized movie effect
c.) Non-regularized movie + user effect
d.) Non-regularized movie + user + genres effect
e.) Non-regularized movie + user + genres + weekday_rated effect
f.) Non-regularized movie + user + genres + weekday_rated + year_released effect
g.) Non-regularized movie + user + genres + weekday_rated + year_released + year_rated effect
h.) Regularized movie effect
i.) Regularized movie + user effect
j.) Regularized movie + user + genres effect
k.) Regularized movie + user + genres + weekday_rated effect
l.) Regularized movie + user + genres + weekday_rated + year_released effect
m.) Regularized movie + user + genres + weekday_rated + year_released + year_rated effect
n.) Linear regression (lm) method from stats package
o.) Random forest model from h2o package
p.) Generalized linear model (glm) from h2o package
q.) Deep learning with (7,3) hidden neurons from h2o package
r.) Gradient boosting machine (gbm) from h2o package

Note that for simplicity, all the six independent variables will be applied outrightly, not gradually, on the last five models, which use the stats and h2o packages.

The respective RMSEs of the models will then be calculated on both the test_set with 7,011,381 obs. of 11 variables and validation dataset with 2,595,763 obs. However, only the RMSE of the validation set will be reported as the basis of determining the best model and grade.

3.1 Just the average (naive)

This is the simplest model and it assumes the same rating μ for all movies for all users under all circumstances. As discussed in Section 33.7.4 of the book (https://rafalab.github.io/dsbook/), the equation is:

$$Y_{u,i} = \mu + \epsilon_{u,i}$$

where:
μ = true rating for all movies
$\epsilon_{u,i}$ = independent errors sampled from the same distribution centered at 0

method	RMSE_validation
Just the mean	1.052557167

The average from all ratings in the train_set mu was calculated to be **3.5269723576** and this represents the predicted rating that any user will most likely provide for any movie. The RMSEs on the test_set and validation sets are **1.0518982335** and **1.052557167**, respectively. The validation set's RMSE of **1.052557167** should be the maximum value, and anything above this should be worse.

3.2 Non-regularized models

3.2.1 Movie effect

The naive model could be improved by adding the movie effect b_i. As discussed in Section 33.7.5 of the book (https://rafalab.github.io/dsbook/), this value likewise is referred to as "bias", with the intuition that different movies are rated differently - meaning that certain movies are rated higher than others. The formula that considers the effect of movie can be defined as:

$$Y_{u,i} = \mu + b_i + \epsilon_{u,i}$$

where: b_i = bias for $movie_i$ and is just the average of ($Y_{u,i}$ - μ) for each movie

method	RMSE_validation
Just the mean	1.0525571670
Movie effect	0.9411804404

The RMSE of adding movie effect on the test_set = **0.9409529753**, while the RMSE on validation set is **0.9411804404**, with the naive model likewise indicated for comparison.

3.2.2 Movie + user effect

The effect of user is added to the movie effect as some users have the tendency of giving higher ratings to certain movies than other users. Some users love every movie but some are even peeky, choosy, or hard-to-please. Hence, there is considerable variability across users which could be represented by the user-specific effect or bias b_u. This simply means, as discussed in Section 33.7.6 of the book (https://rafalab.github.io/dsbook/), that if a cranky user (-b_u) gives a rating to a great movie (+b_i), the effects would counter each other so we could say that such user gave this great movie a 3 rather than a 5.

The formula to add the user effect can be defined as:

$$Y_{u,i} = \mu + b_i + b_u + \epsilon_{u,i}$$

where:
b_i = bias for $movie_i$
b_u = user-specific effect = average of ($Y_{u,i}$ - μ - b_i) for each user

method	RMSE_validation
Just the mean	1.0525571670
Movie effect	0.9411804404
Movie + user effect	0.8641412163

The RMSEs on the test_set and validation datasets as a result of adding user effect are **0.8577063124** and **0.8641412163**, respectively. RMSEs from previous models are likewise indicated for comparison.

3.2.3 Movie + user + genres effect

The effect of genres is added to the movie and user effects as certain users may be biased to give high ratings to movies of specific genres than other users. For instance, some users may give higher ratings to thriller movies than documentary. This is evident during the exploratory data analysis in Section 2.2.2.2 of this material in that drama, comedy, action, and thriller movies were rated higher than for example, documentary or film-noir. The formula to add the genre effect can be defined as:

$$Y_{u,i} = \mu + b_i + b_u + b_g + \epsilon_{u,i}$$

where:
b_i = bias for $movie_i$
b_u = user-specific effect
b_g = genre-specific effect = average of ($Y_{u,i}$ - μ - b_i - b_u) for each genre

method	RMSE_validation
Just the mean	1.0525571670
Movie effect	0.9411804404
Movie + user effect	0.8641412163
Movie + user + genres effect	0.8640504507

The RMSEs on the test_set and validation datasets as a result of adding genre effect are **0.8576231022** and **0.8640504507**, respectively.

3.2.4 Movie + user + genres + weekday_rated effect

Based of the exploratory data analysis in Section 2.2.2.7 of this material, the count of ratings were comparatively higher around Monday or Tuesday of the week. There seems to be no clear explanation for this but perhaps users who watched movies during the weekend may have reflected the ratings on either Monday or Tuesday the following week as they were prompted and/or were available to rate. The formula to add the weekday effect could be defined as:

$$Y_{u,i} = \mu + b_i + b_u + b_g + b_d + \epsilon_{u,i}$$

where:
b_i = bias for $movie_i$
b_u = user effect
b_g = genre effect
b_d = weekday effect = average of ($Y_{u,i}$ - μ - b_i - b_u - b_g) for each weekday

method	RMSE_validation
Just the mean	1.0525571670
Movie effect	0.9411804404
Movie + user effect	0.8641412163
Movie + user + genres effect	0.8640504507
Movie + user + genres + weekday_rated effect	0.8640488429

The RMSEs on the test_set and validation datasets as a result of adding weekday effect are **0.8576227528** and **0.8640488429**, respectively.

3.2.5 Movie + user + genres + weekday_rated + year_released effect

The year_released variable was one of the important variables identified after running the earth package in the main R code and as indicated in Section 2.2.5 Variable importance of this material. As well, the distribution of edx ratings by year_released in Section 2.2.2.3 indicate that there were many ratings made sometime in mid 1990's, specifically 1994. Perhaps, there were many great movies that were released during this year. The formula to add the year_released effect could be written as:

$$Y_{u,i} = \mu + b_i + b_u + b_g + b_d + y_r + \epsilon_{u,i}$$

where:
b_i = bias for $movie_i$
b_u = user effect
b_g = genre effect
b_d = weekday effect
y_r = year released effect = average of ($Y_{u,i}$ - μ - b_i - b_u - b_g - b_d) for each year_released

The calculated RMSE on the validation set is shown below:

method	RMSE_validation
Just the mean	1.0525571670
Movie effect	0.9411804404
Movie + user effect	0.8641412163
Movie + user + genres effect	0.8640504507
Movie + user + genres + weekday_rated effect	0.8640488429
Movie + user + genres + weekday_rated + year_released effect	0.8636784216

The RMSEs on the test_set and validation datasets as a result of adding year released effect are **0.8572703631** and **0.8636784216**, respectively.

3.2.6 Movie + user + genres + weekday_rated + year_released + year_rated effect

In addition to year_released, the year_rated variable was likewise one of the important variables determined after running the earth package in the main R code. This was discussed in indicated in Section 2.2.5 Variable importance of this material. The distribution of edx ratings by year_rated in Section 2.2.2.4 indicate that there were many ratings made sometime in during the years 1996 and 2000. The formula to add the year_rated effect could be written as:

$$Y_{u,i} = \mu + b_i + b_u + b_g + b_d + y_r + y_a + \epsilon_{u,i}$$

where:
b_i = bias for $movie_i$
b_u = user effect
b_g = genre effect
b_d = weekday effect
y_r = year released effect
y_a = year rated effect = average of ($Y_{u,i}$ - μ - b_i - b_u - b_g - b_d - y_r) for each year rated

method	RMSE_validation
Just the mean	1.0525571670
Movie effect	0.9411804404
Movie + user effect	0.8641412163
Movie + user + genres effect	0.8640504507
Movie + user + genres + weekday_rated effect	0.8640488429
Movie + user + genres + weekday_rated + year_released effect	0.8636784216
Movie + user + genres + weekday_rated + year_released + year_rated effect	0.8635987481

The RMSEs on the test_set and validation datasets as a result of adding year rated effect are **0.8571846922** and **0.8635987481**, respectively.

3.3 Regularized models

The precision of the movie bias b_i is dependent on the number of occurrences (samples) that such movies were rated - i.e., the more ratings, the more precise b_i would be. However, while there are certain movies that were rated many times, there are likewise certain movies that were rated only once or a few times. Hence, there is a need to put more weight on the movies that have many ratings, and lesser weight on those that have less.

As discussed in Section 33.9 of the book (https://rafalab.github.io/dsbook/), regularization penalizes large estimates that are formed using small sample sizes. This is solved by introducing the penalty parameter λ, and the idea is to constrain the total variability of the effect sizes. Hence, in the formula below:

$$b_i(\lambda) = \frac{1}{\lambda + n_i} \sum_{u=1}^{n_i} (Y_{u,i} - \mu)$$

a.) If the number of ratings made for $movie_i$ is very large, then the penalty λ is effectively ignored because the term (λ + n_i) will approximate n_i. In this case we have a stable estimate of $b_i(\lambda)$.

b.) On the contrary, if the number of ratings made for $movie_i$ is very small, then the penalty λ becomes large because the term (λ + n_i) will approximate λ. In this case, the estimate $b_i(\lambda)$ is shrunk to 0; thus the larger the λ, the more we shrink.

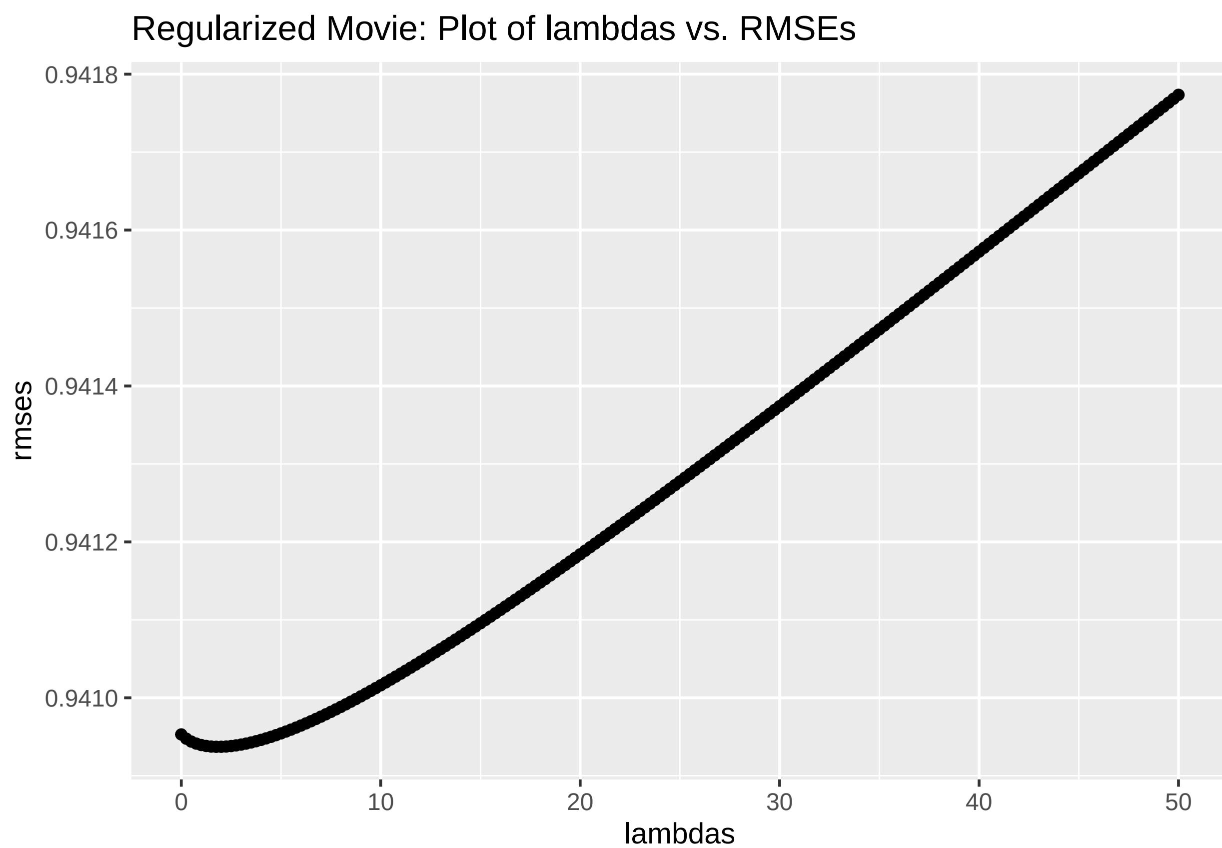
In the succeeding sections for regularized models, the penalty λ will become the tuning parameter to reduce the RMSEs. It will first be calculated from the train_set and then the λ with the least RMSE will be introduced into the models to predict the ratings and estimate the RMSE on both the test_set and validation datasets. And aside from the formulas and RMSE tables, graphs of lambdas vs. RMSEs will be presented along with with the values of the best-tuned lambda for each of the models.

3.3.1 Movie effect

The formula used to regularize movie effect b_i is:

$$Y_{u,i} = \mu + b_i + \epsilon_{u,i}$$

$$where : b_i(\lambda) = \frac{1}{\lambda + n_i} \sum_{u=1}^{n_i} (Y_{u,i} - \mu)$$



method	RMSE_validation
Just the mean	1.0525571670
Movie effect	0.9411804404
Movie + user effect	0.8641412163
Movie + user + genres effect	0.8640504507
Movie + user + genres + weekday_rated effect	0.8640488429
Movie + user + genres + weekday_rated + year_released effect	0.8636784216
Movie + user + genres + weekday_rated + year_released + year_rated effect	0.8635987481
Regularized movie model	0.9369663126

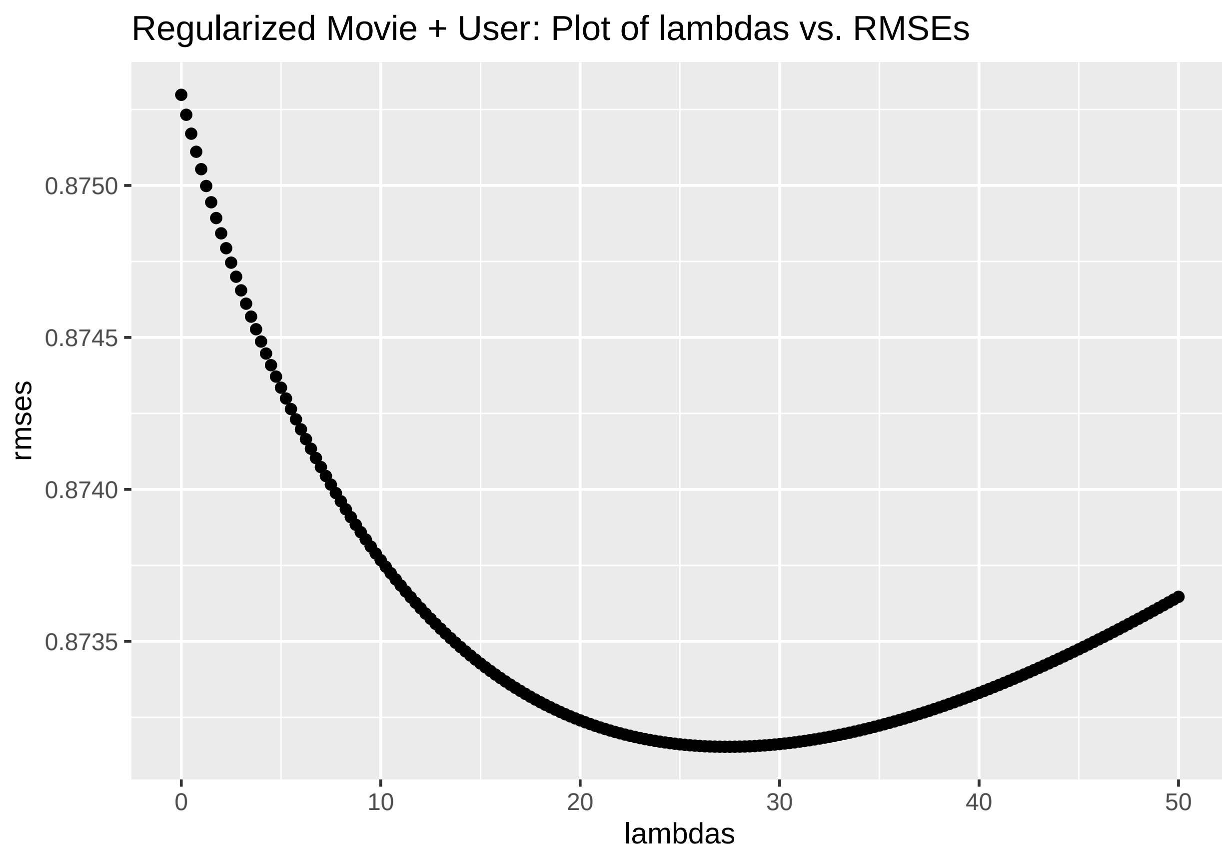
The best-tuned λ with the least RMSE to regularize movie effect b_i = **1.75**. The RMSEs on the test_set and validation datasets as a result of regularizing movie effect b_i are **0.9399571015** and **0.9369663126**, respectively.

3.3.2 Movie + user effect

The formula used to regularize user effect b_u is:

$$Y_{u,i} = \mu + b_i + b_u + \epsilon_{u,i}$$

$$where: b_u(\lambda) = \frac{1}{\lambda + n_i} \sum_{u=1}^{n_i} (Y_{u,i} - \mu - b_i)$$



method	RMSE_validation
Just the mean	1.0525571670
Movie effect	0.9411804404
Movie + user effect	0.8641412163
Movie + user + genres effect	0.8640504507
Movie + user + genres + weekday_rated effect	0.8640488429
Movie + user + genres + weekday_rated + year_released effect	0.8636784216
Movie + user + genres + weekday_rated + year_released + year_rated effect	0.8635987481
Regularized movie model	0.9369663126
Regularized movie + user model	0.8571358019

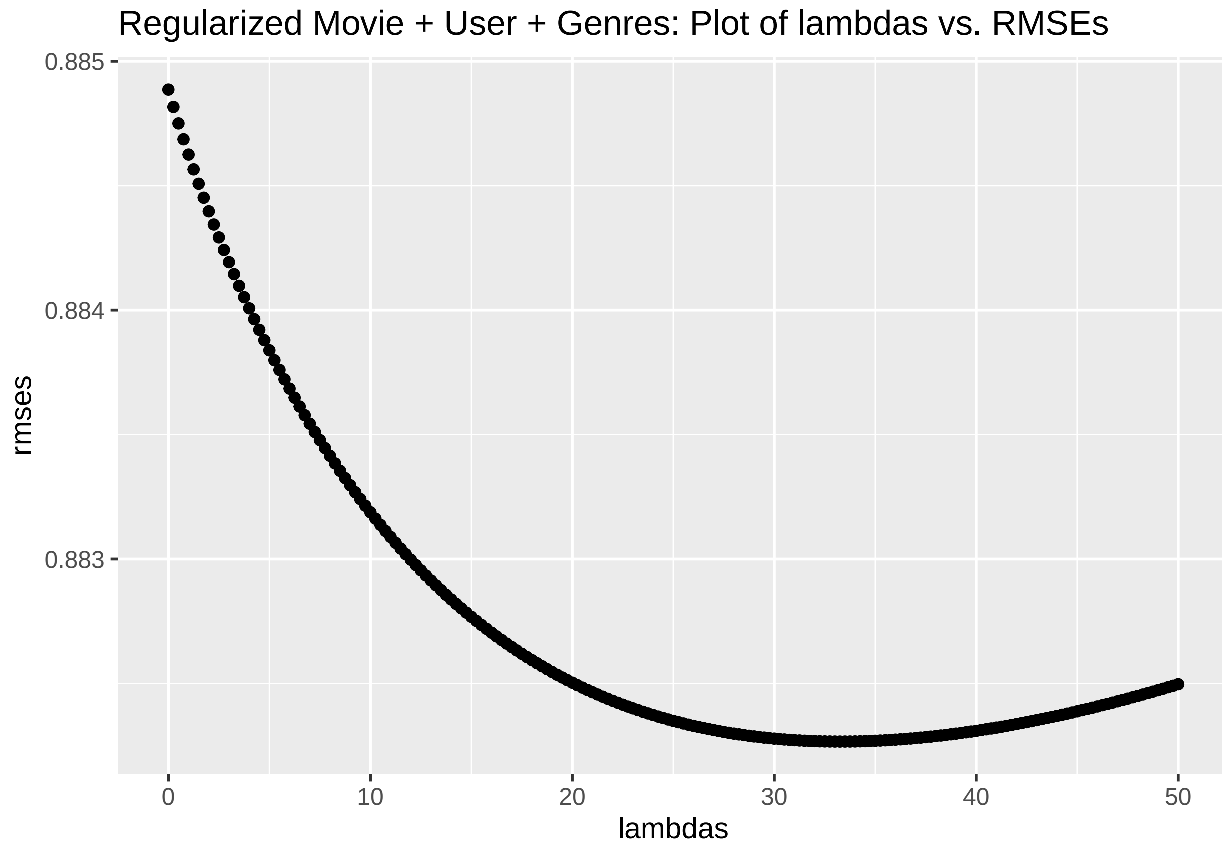
The best-tuned λ with the least RMSE to regularize user effect b_u = **27.5**. The RMSEs on the test_set and validation datasets as a result of regularizing user effect b_u are **0.8692830213** and **0.8571358019**, respectively.

3.3.3 Movie + user + genres effect

The formula used to regularize genre effect b_g is:

$$Y_{u,i} = \mu + b_i + b_u + b_g + \epsilon_{u,i}$$

$$where: b_g(\lambda) = \frac{1}{\lambda + n_i} \sum_{u=1}^{n_i} (Y_{u,i} - \mu - b_i - b_u)$$



method	RMSE_validation
Just the mean	1.0525571670
Movie effect	0.9411804404
Movie + user effect	0.8641412163
Movie + user + genres effect	0.8640504507
Movie + user + genres + weekday_rated effect	0.8640488429
Movie + user + genres + weekday_rated + year_released effect	0.8636784216
Movie + user + genres + weekday_rated + year_released + year_rated effect	0.8635987481
Regularized movie model	0.9369663126
Regularized movie + user model	0.8571358019
Regularized movie + user + genres model	0.8684473123

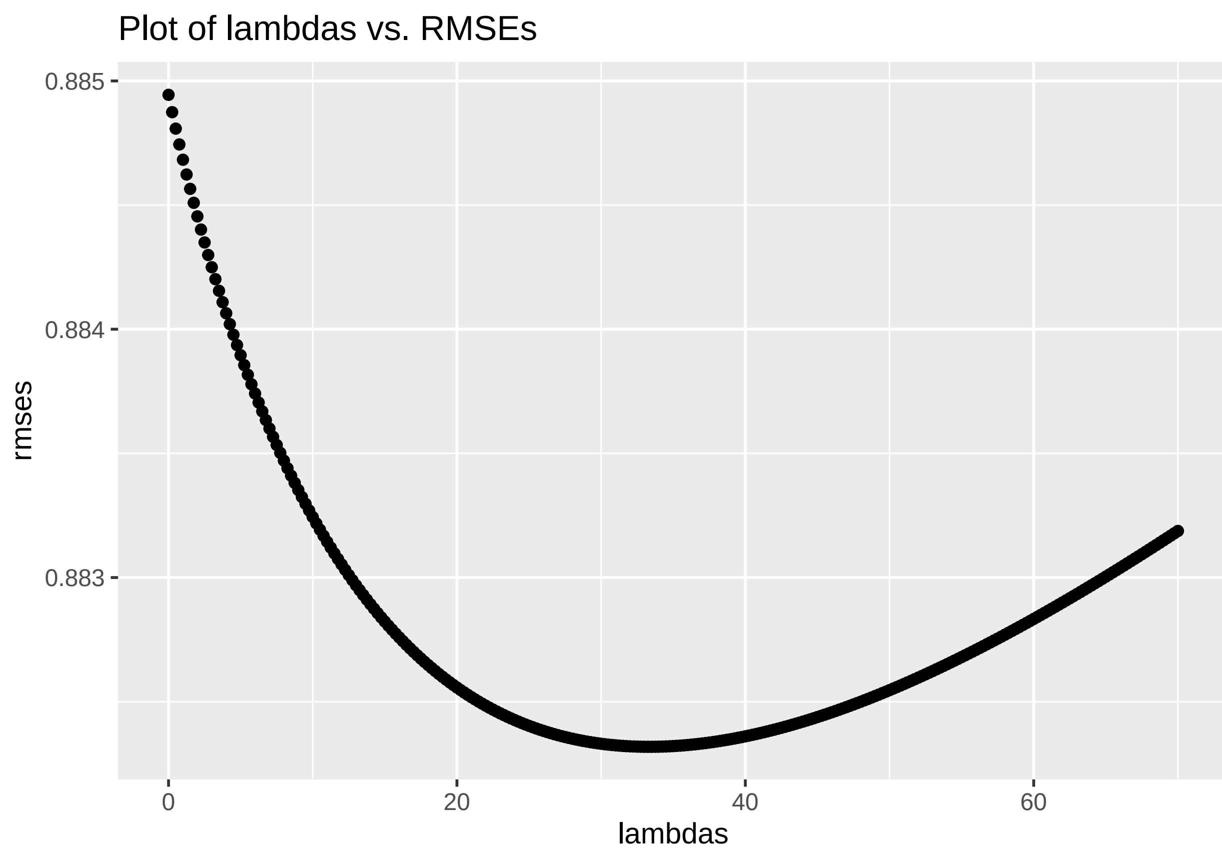
The best-tuned λ with the least RMSE to regularize genre effect b_g = **33.25**. The RMSEs on the test_set and validation datasets as a result of regularizing genre effect b_g are **0.8788499509** and **0.8684473123**, respectively.

3.3.4 Movie + user + genres + weekday_rated effect

The formula used to regularize weekday effect b_d is:

$$Y_{u,i} = \mu + b_i + b_u + b_g + b_d + \epsilon_{u,i}$$

$$where: b_d(\lambda) = \frac{1}{\lambda + n_i} \sum_{u=1}^{n_i} (Y_{u,i} - \mu - b_i - b_u - b_g)$$



method	RMSE_validation
Just the mean	1.0525571670
Movie effect	0.9411804404
Movie + user effect	0.8641412163
Movie + user + genres effect	0.8640504507
Movie + user + genres + weekday_rated effect	0.8640488429
Movie + user + genres + weekday_rated + year_released effect	0.8636784216
Movie + user + genres + weekday_rated + year_released + year_rated effect	0.8635987481
Regularized movie model	0.9369663126
Regularized movie + user model	0.8571358019
Regularized movie + user + genres model	0.8684473123
Regularized movie + user + genres + weekday_rated model	0.8684984877

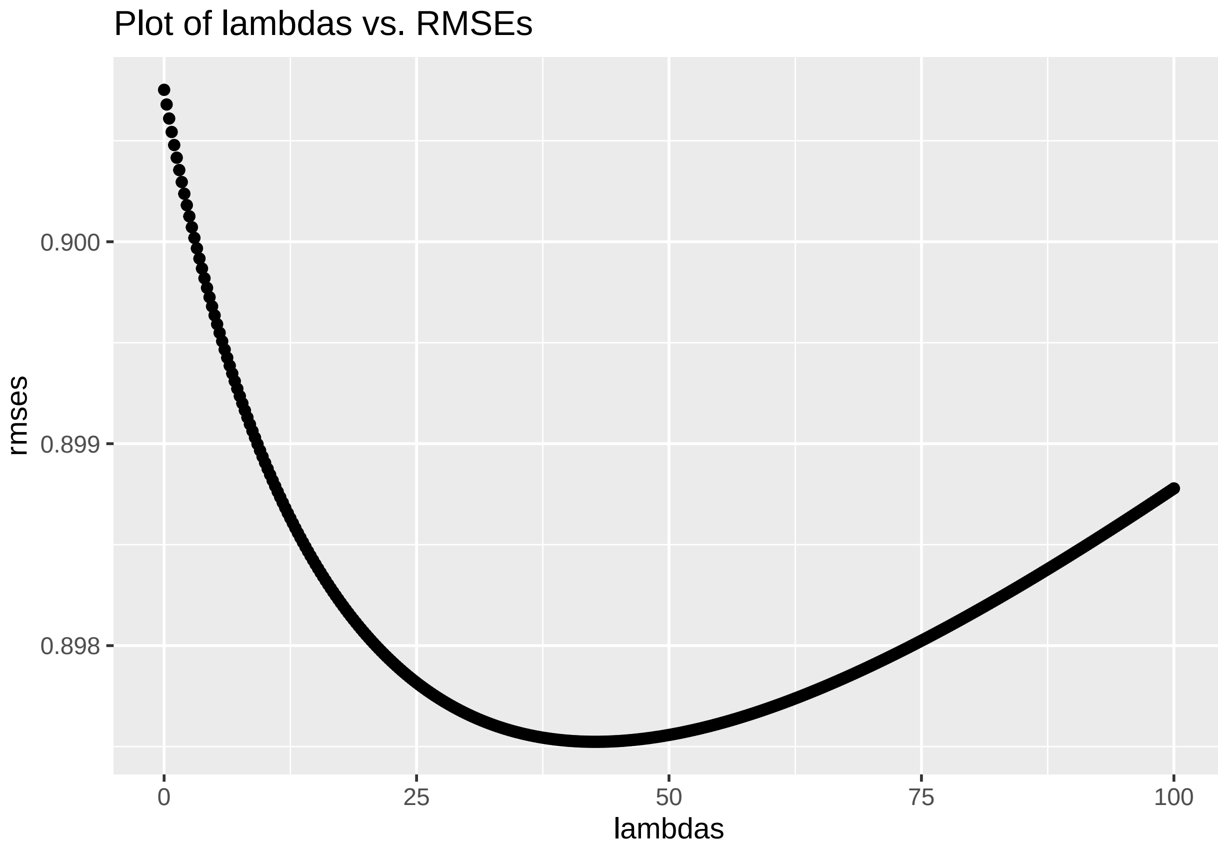
The best-tuned λ with the least RMSE to regularize weekday effect b_d = **33.25**. The RMSEs on the test_set and validation datasets as a result of regularizing weekday effect b_d are **0.8788909578** and **0.8684984877**, respectively.

3.3.5 Movie + user + genres + weekday_rated + year_released effect

The formula used to regularize year released effect y_r is:

$$Y_{u,i} = \mu + b_i + b_u + b_g + b_d + y_r + \epsilon_{u,i}$$

$$where: y_r(\lambda) = \frac{1}{\lambda + n_i} \sum_{u=1}^{n_i} (Y_{u,i} - \mu - b_i - b_u - b_g - b_d)$$



method	RMSE_validation
Just the mean	1.0525571670
Movie effect	0.9411804404
Movie + user effect	0.8641412163
Movie + user + genres effect	0.8640504507
Movie + user + genres + weekday_rated effect	0.8640488429
Movie + user + genres + weekday_rated + year_released effect	0.8636784216
Movie + user + genres + weekday_rated + year_released + year_rated effect	0.8635987481
Regularized movie model	0.9369663126
Regularized movie + user model	0.8571358019
Regularized movie + user + genres model	0.8684473123
Regularized movie + user + genres + weekday_rated model	0.8684984877
Regularized movie + user + genres + weekday_rated + year_released model	0.8866161287

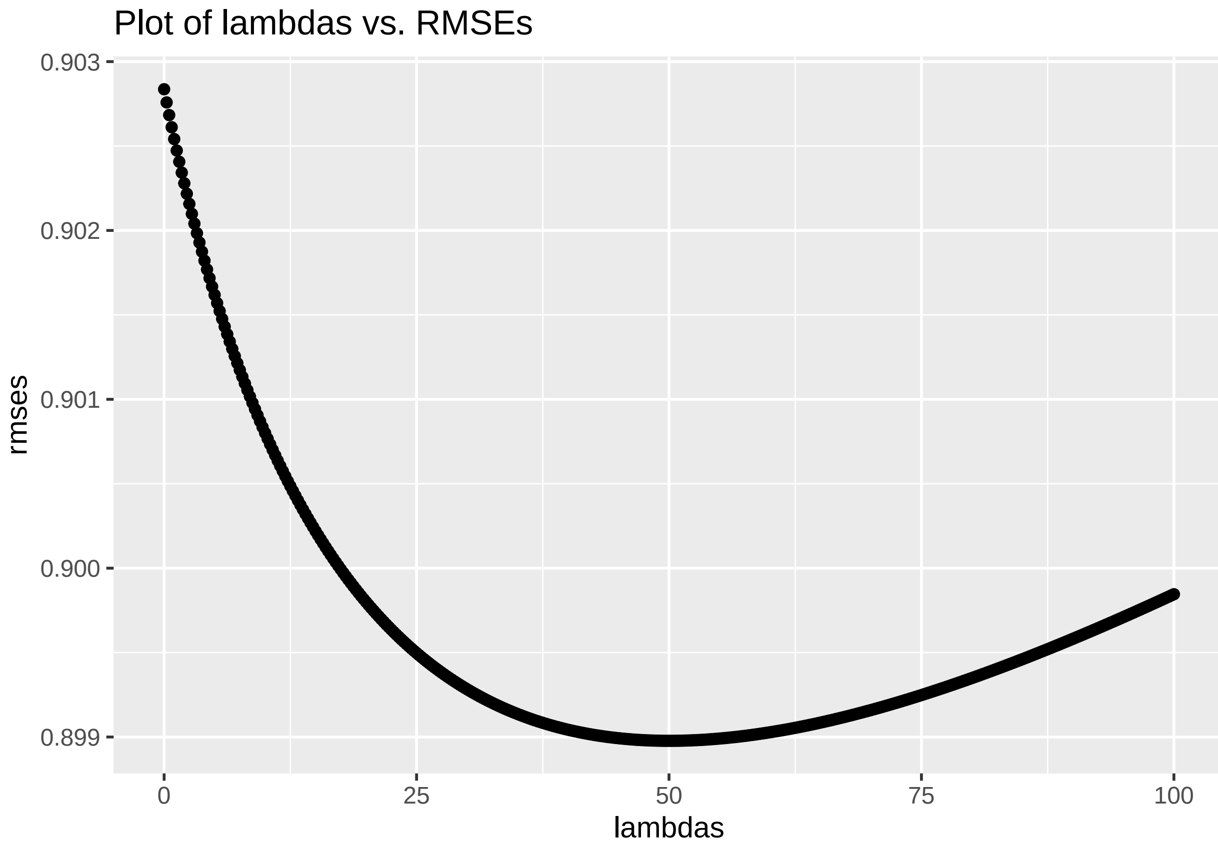
The best-tuned λ with the least RMSE to regularize year released effect y_r = **42.75**. The RMSEs on the test_set and validation datasets as a result of regularizing year released effect y_r are **0.8947160951** and **0.8866161287**, respectively.

3.3.6 Movie + user + genres + weekday_rated + year_released + year_rated effect

The formula used to regularize year rated effect y_a is:

$$Y_{u,i} = \mu + b_i + b_u + b_g + b_d + y_r + y_a + \epsilon_{u,i}$$

$$where: y_a(\lambda) = \frac{1}{\lambda + n_i} \sum_{u=1}^{n_i} (Y_{u,i} - \mu - b_i - b_u - b_g - b_d - y_r)$$



method	RMSE_validation
Just the mean	1.0525571670
Movie effect	0.9411804404
Movie + user effect	0.8641412163
Movie + user + genres effect	0.8640504507
Movie + user + genres + weekday_rated effect	0.8640488429
Movie + user + genres + weekday_rated + year_released effect	0.8636784216
Movie + user + genres + weekday_rated + year_released + year_rated effect	0.8635987481
Regularized movie model	0.9369663126
Regularized movie + user model	0.8571358019
Regularized movie + user + genres model	0.8684473123
Regularized movie + user + genres + weekday_rated model	0.8684984877
Regularized movie + user + genres + weekday_rated + year_released model	0.8866161287
Regularized movie + user + genres + weekday_rated + year_released + year_rated	0.8897930635

The best-tuned λ with the least RMSE to regularize year rated effect $y_a = \mathbf{50}$. The RMSEs on the test_set and validation datasets as a result of regularizing year rated effect y_a are **0.8964442194** and **0.8897930635**, respectively.

3.4 linear regression (lm) method

The formula to implement linear model with rating $Y_{u,i}$ as the dependent variable and userId, movieId, genres, weekday_rated, year_released, and year_rated as independent variables is given by:

$$Y_{u,i} = b_0 + b_1 movieId + b_2 userId + b_3 genres + b_4 weekday_{rated} + b_5 year_{released} + b_6 year_{rated}$$

where:
b_0 = y-intercept
b_1 = coefficient for movie effect
b_2 = coefficient for user effect
b_3 = coefficient for genre effect
b_4 = coefficient for weekday effect
b_5 = coefficient for year released effect
b_6 = coefficient for year rated effect

method	RMSE_validation
Just the mean	1.0525571670
Movie effect	0.9411804404
Movie + user effect	0.8641412163
Movie + user + genres effect	0.8640504507
Movie + user + genres + weekday_rated effect	0.8640488429
Movie + user + genres + weekday_rated + year_released effect	0.8636784216
Movie + user + genres + weekday_rated + year_released + year_rated effect	0.8635987481
Regularized movie model	0.9369663126
Regularized movie + user model	0.8571358019
Regularized movie + user + genres model	0.8684473123
Regularized movie + user + genres + weekday_rated model	0.8684984877
Regularized movie + user + genres + weekday_rated + year_released model	0.8866161287
Regularized movie + user + genres + weekday_rated + year_released + year_rated	0.8897930635
stats linear regression (lm) method	1.0384922169

The RMSEs on the test_set and validation datasets as a result of running the linear model from the stats package based of the above formula are **1.0379411513** and **1.0384922169**, respectively.

Brief Introduction to h2o Library:

The remaining four sections attempt to predict the rating $Y_{u,i}$ using the h2o library.

The h2o library is a scalable open-source machine learning library that features AutoML. This article from R-bloggers (https://www.r-bloggers.com/5-reasons-to-learn-h2o-for-high-performance-machine-learning/) caught my attention, and this is the reason h2o models are included in this project. According to the author, there are 5 reasons for using h2o:

a.) h2o AutoML automates the machine learning workflow, which includes automatic training and tuning of many models.

b.) Scalable on Local Compute: distributed, in-memory processing speeds up computations

c.) Spark integration & GPU support: the result is 100x faster training than traditional ML.

d.) Superior performance: best algorithms, optimized and ensembled: The most popular algorithms are incorporated including GLM, random forest, GBM and more.

e.) Production ready, e.g. docker containers

Similar to lm() model, the formula that will be used in modeling using the h2o library is:

$$Y_{u,i} = movieId + userId + genres + weekday_{rated} + year_{released} + year_{rated} + \epsilon$$

Finally, the hyper-parameters from the following h2o models below have not yet been fine-tuned due to time constraints in learning how to use the library and so determining the best-tuned parameters is beyond the scope of this project. As well, there is no guarantee that the models created here using h2o will significantly reduce the RMSE on the movielens dataset. But then out of curiousity, it may be worth exploring and trying it out in this dataset and project in particular, and for other projects in general. Nevertheless for more information and other details, h2o tutorials are available in http://docs.h2o.ai/h2o-tutorials/latest-stable/.

3.5 h2o random forest implementation

The h2o library need to first be loaded and initialized. As well, the train_set, test_set, and validation datasets should be converted to h2o instance using the folowing code:

```
library(h2o)
h2o.init()
```

```
##
## H2O is not running yet, starting it now...
##
## Note:  In case of errors look at the following log files:
##     C:\Users\willy\AppData\Local\Temp\RtmpamATPp\file125833523566/h2o_willy_started_from_r.out
##     C:\Users\willy\AppData\Local\Temp\RtmpamATPp\file12587b6358d6/h2o_willy_started_from_r.err
##
##
## Starting H2O JVM and connecting: . Connection successful!
##
## R is connected to the H2O cluster:
##     H2O cluster uptime:         3 seconds 223 milliseconds
##     H2O cluster timezone:       America/Denver
##     H2O data parsing timezone:  UTC
##     H2O cluster version:        3.30.0.1
##     H2O cluster version age:    2 months and 8 days
##     H2O cluster name:           H2O_started_from_R_willy_kaq408
##     H2O cluster total nodes:    1
##     H2O cluster total memory:   7.10 GB
##     H2O cluster total cores:    8
##     H2O cluster allowed cores:  8
##     H2O cluster healthy:        TRUE
##     H2O Connection ip:          localhost
##     H2O Connection port:        54321
##     H2O Connection proxy:       NA
##     H2O Internal Security:      FALSE
##     H2O API Extensions:         Amazon S3, Algos, AutoML, Core V3, TargetEncoder, Core V4
##     R Version:                  R version 3.5.1 (2018-07-02)
```

```
train.h2o <- as.h2o(train_set)
```

```
##
  |
  |                                                                      |   0%
  |
  |======================================================================| 100%
```

```
test.h2o <- as.h2o(test_set)
```

```
##
  |
  |                                                                      |   0%
  |
  |======================================================================| 100%
```

```r
validation.h2o <- as.h2o(validation)
```

```
##
  |
  |                                                                      |   0%
  |
  |======================================================================| 100%
```

method	RMSE_validation
Just the mean	1.0525571670
Movie effect	0.9411804404
Movie + user effect	0.8641412163
Movie + user + genres effect	0.8640504507
Movie + user + genres + weekday_rated effect	0.8640488429
Movie + user + genres + weekday_rated + year_released effect	0.8636784216
Movie + user + genres + weekday_rated + year_released + year_rated effect	0.8635987481
Regularized movie model	0.9369663126
Regularized movie + user model	0.8571358019
Regularized movie + user + genres model	0.8684473123
Regularized movie + user + genres + weekday_rated model	0.8684984877
Regularized movie + user + genres + weekday_rated + year_released model	0.8866161287
Regularized movie + user + genres + weekday_rated + year_released + year_rated	0.8897930635
stats linear regression (lm) method	1.0384922169
h2o random forest model	1.0290810245

The RMSEs on the test_set and validation datasets as a result of running the h2o random forests model are **1.028565994** and **1.0290810245**, respectively.

3.6 h2o generalized linear model (glm) implementation

method	RMSE_validation
Just the mean	1.0525571670
Movie effect	0.9411804404
Movie + user effect	0.8641412163
Movie + user + genres effect	0.8640504507
Movie + user + genres + weekday_rated effect	0.8640488429
Movie + user + genres + weekday_rated + year_released effect	0.8636784216
Movie + user + genres + weekday_rated + year_released + year_rated effect	0.8635987481
Regularized movie model	0.9369663126
Regularized movie + user model	0.8571358019
Regularized movie + user + genres model	0.8684473123
Regularized movie + user + genres + weekday_rated model	0.8684984877
Regularized movie + user + genres + weekday_rated + year_released model	0.8866161287
Regularized movie + user + genres + weekday_rated + year_released + year_rated	0.8897930635
stats linear regression (lm) method	1.0384922169
h2o random forest model	1.0290810245
h2o glm model	1.0385139067

The RMSEs on the test_set and validation datasets as a result of running the h2o generalized linear model (glm) are **1.0379629524** and **1.0385139067**, respectively.

3.7 h2o deep neural network implementation

method	RMSE_validation
Just the mean	1.0525571670
Movie effect	0.9411804404
Movie + user effect	0.8641412163
Movie + user + genres effect	0.8640504507
Movie + user + genres + weekday_rated effect	0.8640488429
Movie + user + genres + weekday_rated + year_released effect	0.8636784216
Movie + user + genres + weekday_rated + year_released + year_rated effect	0.8635987481
Regularized movie model	0.9369663126
Regularized movie + user model	0.8571358019
Regularized movie + user + genres model	0.8684473123
Regularized movie + user + genres + weekday_rated model	0.8684984877
Regularized movie + user + genres + weekday_rated + year_released model	0.8866161287
Regularized movie + user + genres + weekday_rated + year_released + year_rated	0.8897930635
stats linear regression (lm) method	1.0384922169
h2o random forest model	1.0290810245
h2o glm model	1.0385139067
h2o deep learning: (7,3) hidden layers	1.0294192697

The RMSEs on the test_set and validation datasets as a result of running the h2o deep learning model, with (7,3) hidden layers, and using rectifier activation function are **1.0285261996** and **1.0294192697**, respectively.

3.8 h2o gradient boosting machine (gbm) implementation

method	RMSE_validation
Just the mean	1.0525571670
Movie effect	0.9411804404
Movie + user effect	0.8641412163
Movie + user + genres effect	0.8640504507
Movie + user + genres + weekday_rated effect	0.8640488429
Movie + user + genres + weekday_rated + year_released effect	0.8636784216
Movie + user + genres + weekday_rated + year_released + year_rated effect	0.8635987481
Regularized movie model	0.9369663126
Regularized movie + user model	0.8571358019
Regularized movie + user + genres model	0.8684473123
Regularized movie + user + genres + weekday_rated model	0.8684984877
Regularized movie + user + genres + weekday_rated + year_released model	0.8866161287
Regularized movie + user + genres + weekday_rated + year_released + year_rated	0.8897930635
stats linear regression (lm) method	1.0384922169
h2o random forest model	1.0290810245
h2o glm model	1.0385139067
h2o deep learning: (7,3) hidden layers	1.0294192697
h2o gradient boosting machine (gbm)	1.0095082035

The RMSEs on the test_set and validation datasets as a result of running the h2o gradient boosting machine (gbm) model are **1.0091901316** and **1.0095082035**, respectively.

Finally, the h2o library needs to be shutdown using the code below:

```
h2o.shutdown()
```

```
## Are you sure you want to shutdown the H2O instance running at http://localhost:54321/ (Y/N)?
```

4. Results

Herewith is a summary of RMSEs as well as the corresponding grades from the 18 models that were built:

method	RMSE_validation	grade
Just the mean	1.0525571670	5
Movie effect	0.9411804404	5
Movie + user effect	0.8641412163	25
Movie + user + genres effect	0.8640504507	25
Movie + user + genres + weekday_rated effect	0.8640488429	25
Movie + user + genres + weekday_rated + year_released effect	0.8636784216	25
Movie + user + genres + weekday_rated + year_released + year_rated effect	0.8635987481	25
Regularized movie model	0.9369663126	5
Regularized movie + user model	0.8571358019	25
Regularized movie + user + genres model	0.8684473123	10
Regularized movie + user + genres + weekday_rated model	0.8684984877	10
Regularized movie + user + genres + weekday_rated + year_released model	0.8866161287	10
Regularized movie + user + genres + weekday_rated + year_released + year_rated	0.8897930635	10
stats linear regression (lm) method	1.0384922169	5
h2o random forest model	1.0290810245	5
h2o glm model	1.0385139067	5
h2o deep learning: (7,3) hidden layers	1.0294192697	5
h2o gradient boosting machine (gbm)	1.0095082035	5

After iterating on the independent variables and determining the best-tuned lambda of 27.5, it could be observed that the "Regularized movie + user model" provides the least RMSE of 0.8571358019 on the validation set.

Without Regularization:

For the first six models without regularization, the addition of the variables genres, weekday_rated, year_released, and year_rated to movieId and userId failed to effectively reduce the RMSE. Accordingly, the RMSE was pegged to around 0.864 on the validation set as indicated above.

With Regularization:

Referring to the next six models that were regularized, the addition of both the genres and weekday_rated variables to movieId and userId increased the RMSE from 0.857 to 0.868. To make matters worse, the RMSE increased from 0.857 to around 0.887 when the year_released and year_rated variables were added to the above-mentioned variables.

Linear Models (lm) and h2o models:

The RMSEs of four h2o models as well as the stats lm() model were in the range of 1.010 to 1.038, which were better than the RMSE of "Just the mean" (naive) model of 1.053. However, these RMSEs were obviously not better than the RMSEs of either the non-regularized or regularized models which were in the range of 0.857 to 0.941. This is not to mention the limitation in terms of the intensive compute time required to run the random forests, generalized linear model (glm), deep learning, and gradient boosting machine (gbm) models from the h2o package in processing over 16 million observations. In fact, the time to knit this rmd file would require about 7-8 hours on R 3.5.x Windows 10, 64-bit running on Intel Core i7-7700 with 32 GB RAM.

5. Conclusion

Based of the above results, it is evident that the 'Regularized movie + user' model provides the least RMSE of 0.8571358019 on the validation set. This means that the variables movieId and userId are sufficient to predict the ratings of movies with the least acceptable RMSE. As a limitation, it is likewise evident that the training and implementation of linear model, random forests, gradient boosting, and deep learning models may not be expedient on this type and magnitude of dataset. This is due to the compute-intensiveness and larger memory required to run these models. Perhaps future work could explore and focus on implementing matrix factorization, singular value decomposition (SVD), or principal components analysis (PCA) as described in Section 33.11 of the book (https://rafalab.github.io/dsbook/large-datasets.html#matrix-factorization).

To conclude, it is very possible to reach an RMSE of 0.857 using the regularized movie and user effects.