MovieLens Project HarvardX PH125.9X

Herbert Holeman

12/1/2021

SUMMARY

The movie provider Netflix Inc. uses a computerized recommendation system. It predicts user movie ratings based on a massive collection of user viewing data

To that end, this script generates a movie ratings predictor and calculates a root mean squared error (RMSE) for evaluating prediction performance. These are the results of this algorithm:

method	RMSE
Average movie rating model	1.0603313
Movie effect model	0.9423475
Regularized movie and user effect model	0.8648170

METHODS AND ANALYSIS

While the data from the Netflix system are not publicly available, the GroupLens Lab generated its own such database with over 20 million ratings for over 27,000 movies viewed by more than 138,000 users. The edx dataset, used in this project is a subset of the GroupLens database. It contains some 69,878 movies and 10,677 user ratings.

In developing this script's predictor, the caret (acronm for Classification And REgression Training) family of packages is used to perform the following tasks.

SETUP

Load required R coding packages and download MovieLens 10M data. Run setup code to separate the data into the edx dataset, for training, and the validation dataset, for testing.

```
## Loading required package: caret
```

```
## Warning: package 'caret' was built under R version 4.1.2
```

```
## Loading required package: ggplot2
```

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

```
## Loading required package: tidyverse
```

```
## Warning: package 'tidyverse' was built under R version 4.1.2

## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v tibble  3.1.5     v dplyr   1.0.7
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.0.2     v forcats 0.5.1
## v purrr   0.3.4

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## x purrr::lift()   masks caret::lift()

## Loading required package: data.table

##
## Attaching package: 'data.table'

## The following objects are masked from 'package:dplyr':
##
##     between, first, last

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

## Loading required package: sqldf

## Loading required package: gsubfn

## Loading required package: proto

## Loading required package: RSQLite

## Loading required package: lubridate

##
## Attaching package: 'lubridate'

## The following objects are masked from 'package:data.table':
##
##     hour, isoweek, mday, minute, month, quarter, second, wday, week,
##     yday, year

## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union

## Loading required package: ggthemes
```

```
## Warning: `as.tibble()` was deprecated in tibble 2.0.0.
## Please use `as_tibble()` instead.
## The signature and semantics have changed, see `?as_tibble`.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was generated.

## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used

## Joining, by = c("userId", "movieId", "rating", "timestamp", "title", "genres")
```

EXPLORE DATA

Probe the dataset to examine its structure and key prediction features: userId, movieId, and rating.

```
str(edx)
```

```
## Classes 'data.table' and '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 83
##  $ title    : chr  "Boomerang (1992)" "Net, The (1995)" "Outbreak (1995)" "Stargate (1994)" ...
##  $ genres   : chr  "Comedy|Romance" "Action|Crime|Thriller" "Action|Drama|Sci-Fi|Thriller" "Action|Ad
##  - attr(*, ".internal.selfref")=<externalptr>
```

```
summary(edx)
```

```
##      userId         movieId          rating        timestamp
##  Min.   :    1   Min.   :    1   Min.   :0.500   Min.   :7.897e+08
##  1st Qu.:18124   1st Qu.:  648   1st Qu.:3.000   1st Qu.:9.468e+08
##  Median :35738   Median : 1834   Median :4.000   Median :1.035e+09
##  Mean   :35870   Mean   : 4122   Mean   :3.512   Mean   :1.033e+09
##  3rd Qu.:53607   3rd Qu.: 3626   3rd Qu.:4.000   3rd Qu.:1.127e+09
##  Max.   :71567   Max.   :65133   Max.   :5.000   Max.   :1.231e+09
##     title              genres
##  Length:9000055     Length:9000055
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##
```

```
anyNA(edx)
```

```
## [1] FALSE
```

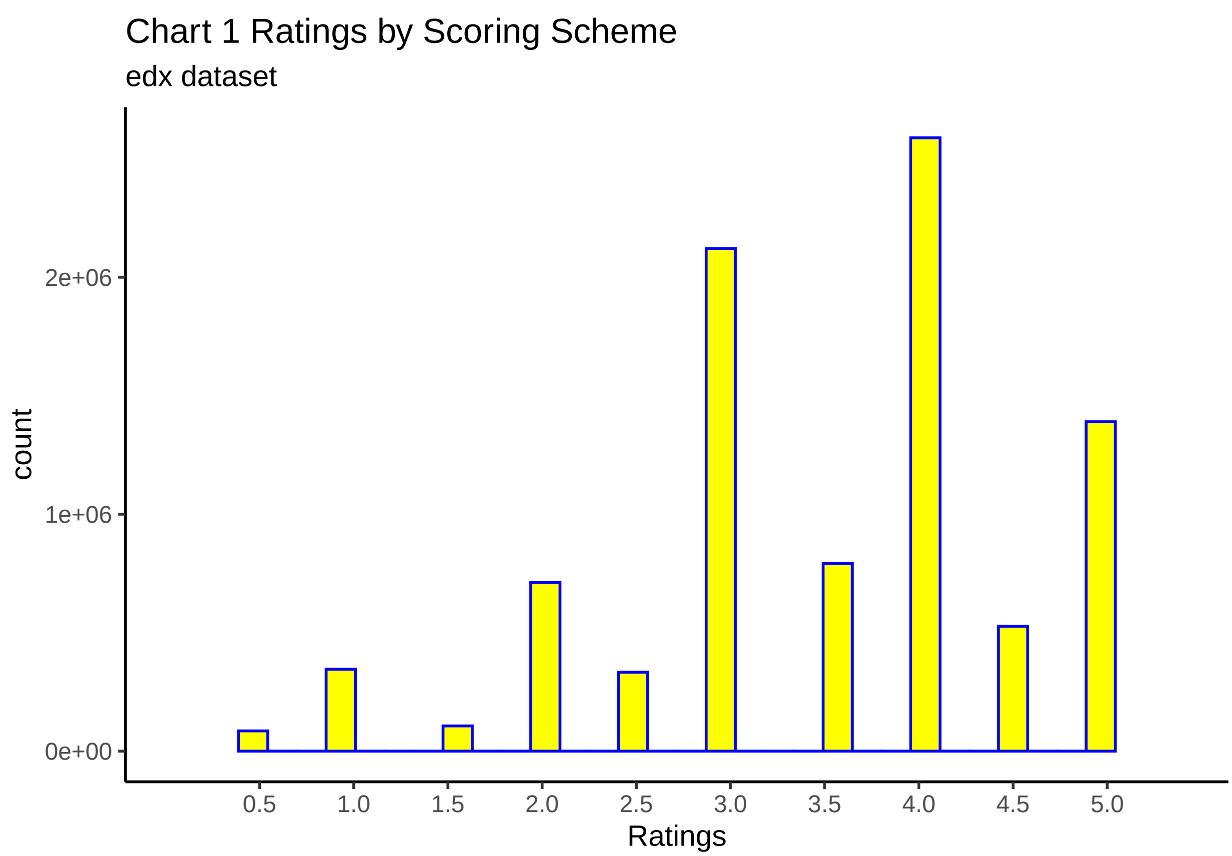
```
edx%>%
  summarize(n_users = n_distinct(userId),
            n_movies = n_distinct(movieId))
```

```
##   n_users n_movies
## 1   69878    10677
```

Create Chart 1 to show movie viewer preference based on the time the Netflix dataset was compiled. A five star rating scheme (1 to 5 stars) was then used for scoring user preference and movie popularity. Follow up the chart with a tabulation of viewer scores by rating given.

```
edx %>%
  ggplot() +
  geom_histogram(aes(x= rating), bins = 30,
  fill = "yellow", color="blue")+
  scale_x_discrete(limits = c(seq(0.5,5,0.5))) +
  labs(title = "Chart 1 Ratings by Scoring Scheme",
  subtitle = "edx dataset", x = "Ratings" )+
  theme_classic()
```

```
## Warning: Continuous limits supplied to discrete scale.
## Did you mean 'limits = factor(...)' or 'scale_*_continuous()'?
```




```
table(edx$rating)
```

```
##
##     0.5       1     1.5       2     2.5       3     3.5       4     4.5       5
##   85374  345679  106426  711422  333010 2121240  791624 2588430  526736 1390114
```

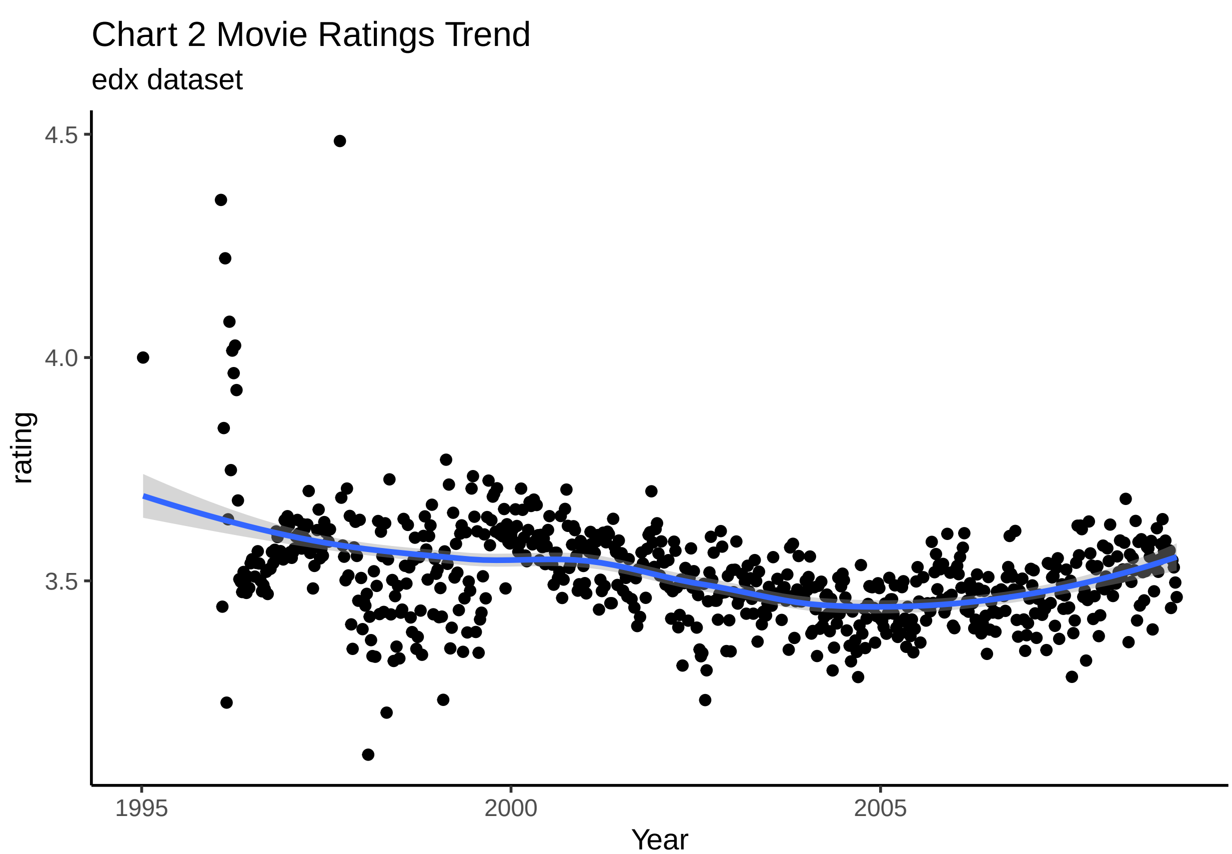
```r
summary(edx$rating)
```

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

Add a date feature to the dataset to evaluate ratings trend over the project's timeframe.

```r
edx <- mutate(edx, date = as_datetime(timestamp))
edx %>% mutate(date = round_date(date, unit = "week")) %>%
  group_by(date) %>%
  summarize(rating = mean(rating)) %>%
  ggplot(aes(date,  rating))  +
  geom_point() +
  geom_smooth()+
  labs(title="Chart 2 Movie Ratings Trend",
       subtitle = "edx dataset", x = "Year") +
  theme_classic()
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



Create charts 3 and 4 to depict a big picture perspective of the users who view the movies and of the movies themselves.

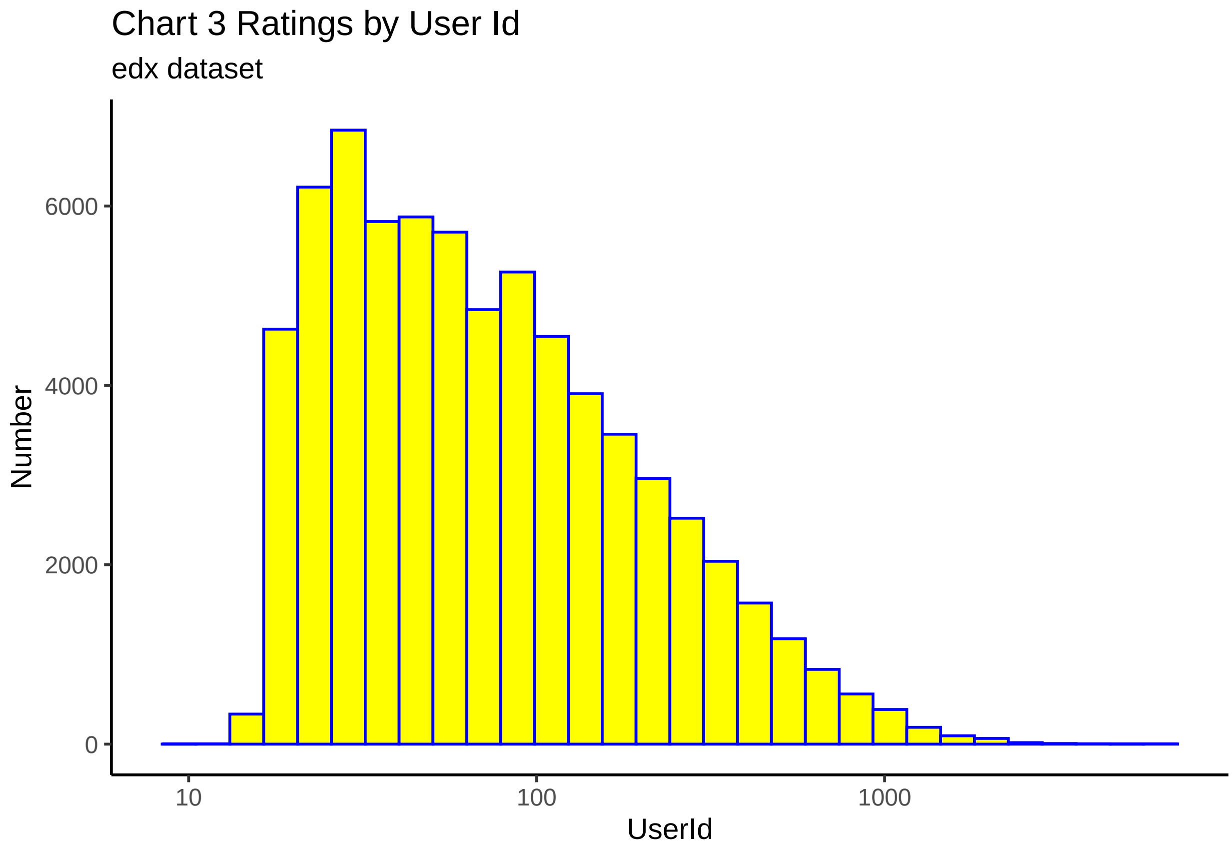
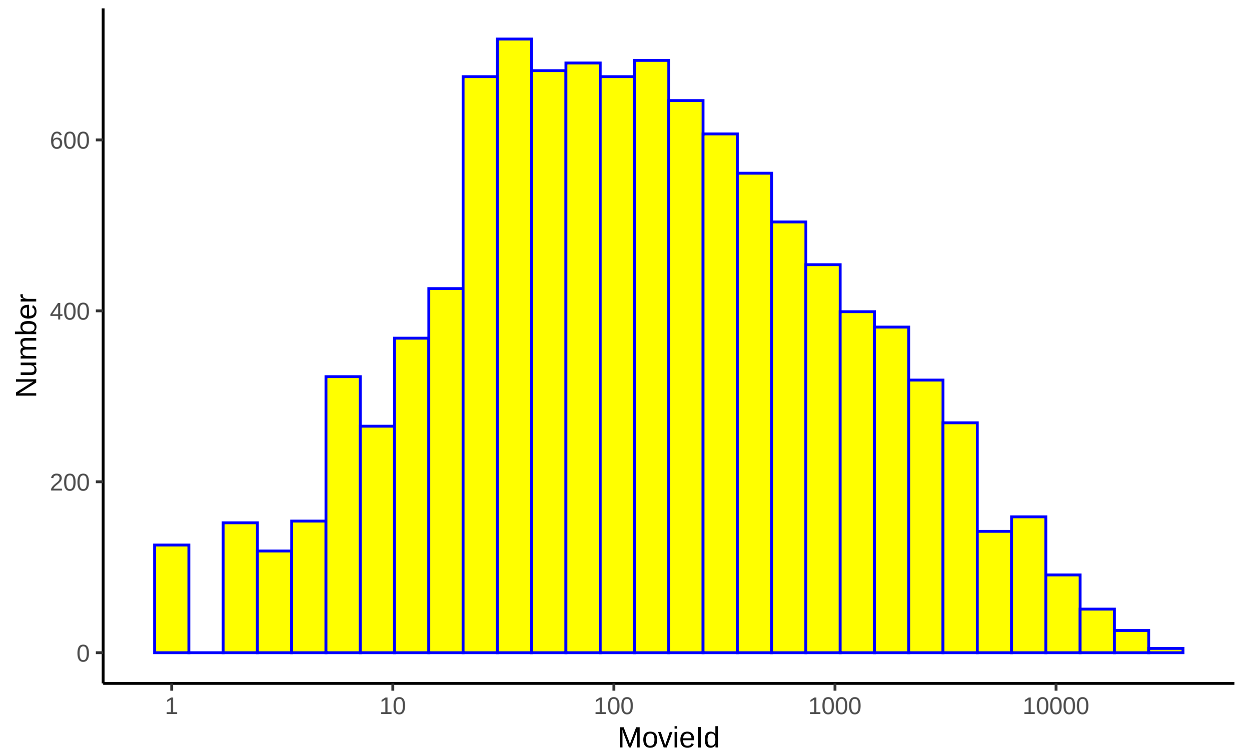


Chart 3 Ratings by User Id
edx dataset
Number
6000
4000
2000
0
10
100
1000
UserId

Chart 4 Ratings by movie Id

edx dataset



BUILD THE RECOMMENDATION SYSTEM

Make a simple baseline prediction of RMSE.

- Get edx dataset mean rating

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

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

```
naive_rmse <- RMSE(edx$rating, mu)
naive_rmse
```

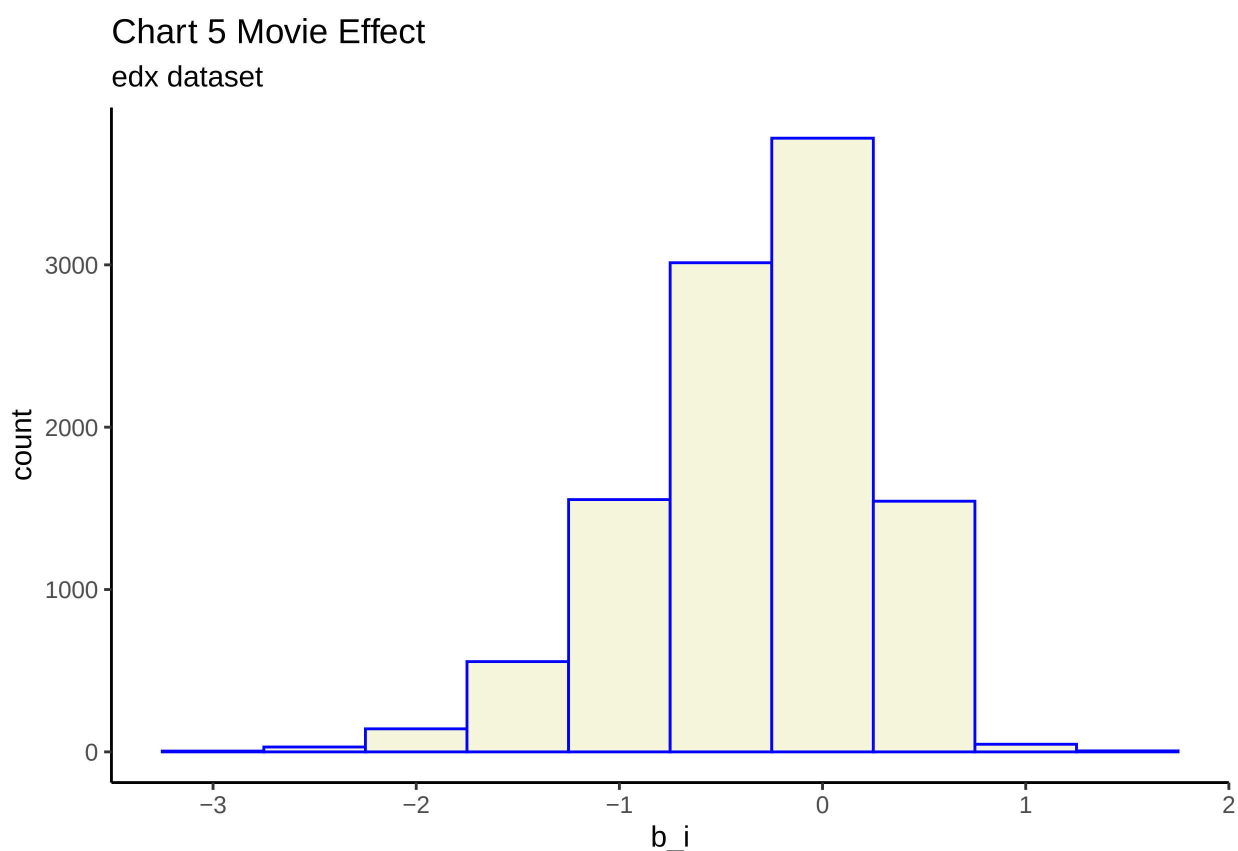
```
## [1] 1.060331
```

```
rmse_results <- data.frame(method = "Average movie rating model",
RMSE = naive_rmse)

print(rmse_results %>% knitr::kable())
```

```
##
##
## |method                     |     RMSE|
## |:--------------------------|--------:|
## |Average movie rating model | 1.060331|
```

Improve baseline RMSE through analysis of bias as an error influencing both movie and user ratings. Show such in Chart 5 as movie effect and in Chart 6 as user effect.

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

# Chart 5 movie effect
movie_avgs  %>%
  ggplot(aes(b_i)) +
  geom_histogram(bins = 10, fill="beige", color= "blue")+
  labs(title="Chart 5 Movie Effect", subtitle="edx dataset")+
  theme_classic()
```

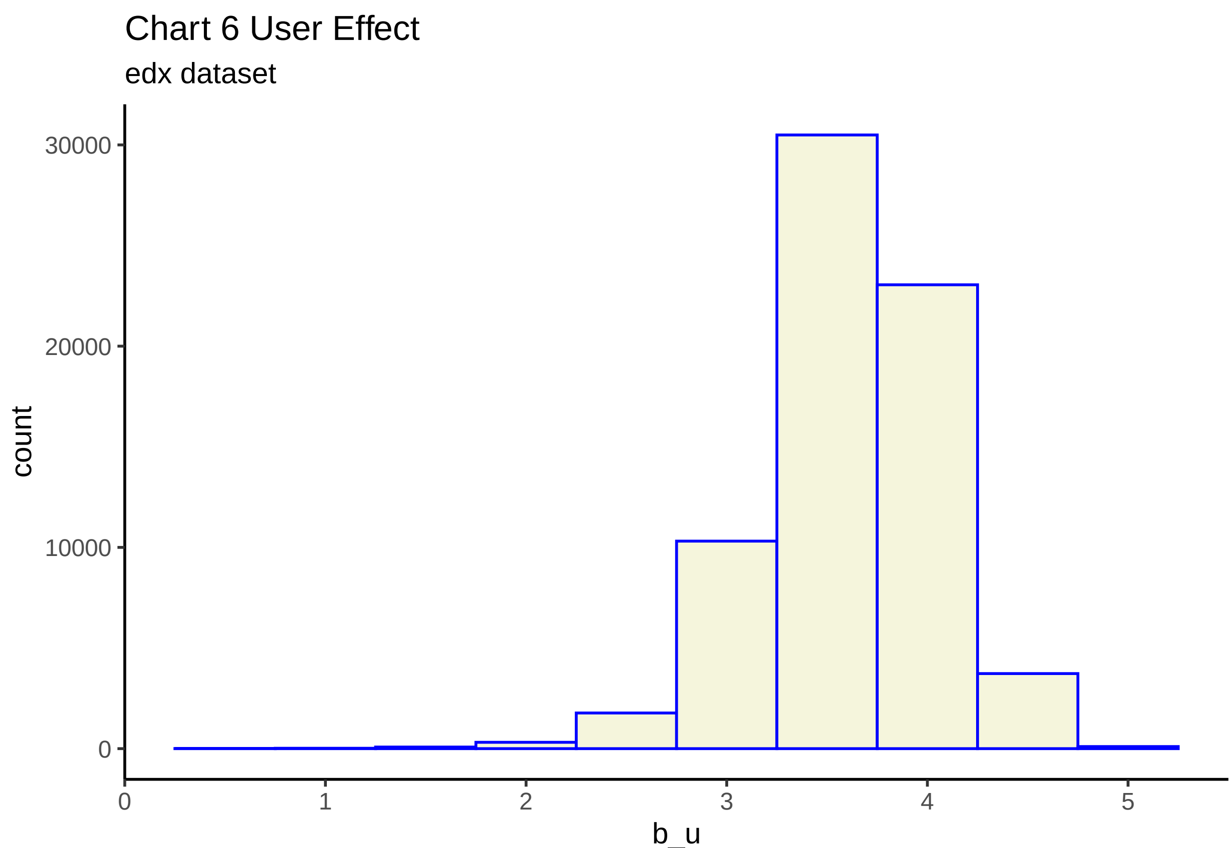



```
predicted_ratings <- mu +  edx %>%
  left_join(movie_avgs, by='movieId') %>%
  pull(b_i)

model_1_rmse <- RMSE(predicted_ratings, edx$rating)

rmse_results <- bind_rows(rmse_results,
                          tibble(method="Movie effect model",
                                 RMSE = model_1_rmse ))
print(rmse_results %>% knitr::kable())
```

```
##
```

```
##
## |method                     |      RMSE|
## |:--------------------------|---------:|
## |Average movie rating model | 1.0603313|
## |Movie effect model         | 0.9423475|
```

```
edx %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating)) %>%
  filter(n()>=100) %>%
  ggplot(aes(b_u)) +
  geom_histogram(bins = 10, fill="beige", color= "blue")+
  labs(title="Chart 6 User Effect", subtitle="edx dataset")+
  theme_classic()
```



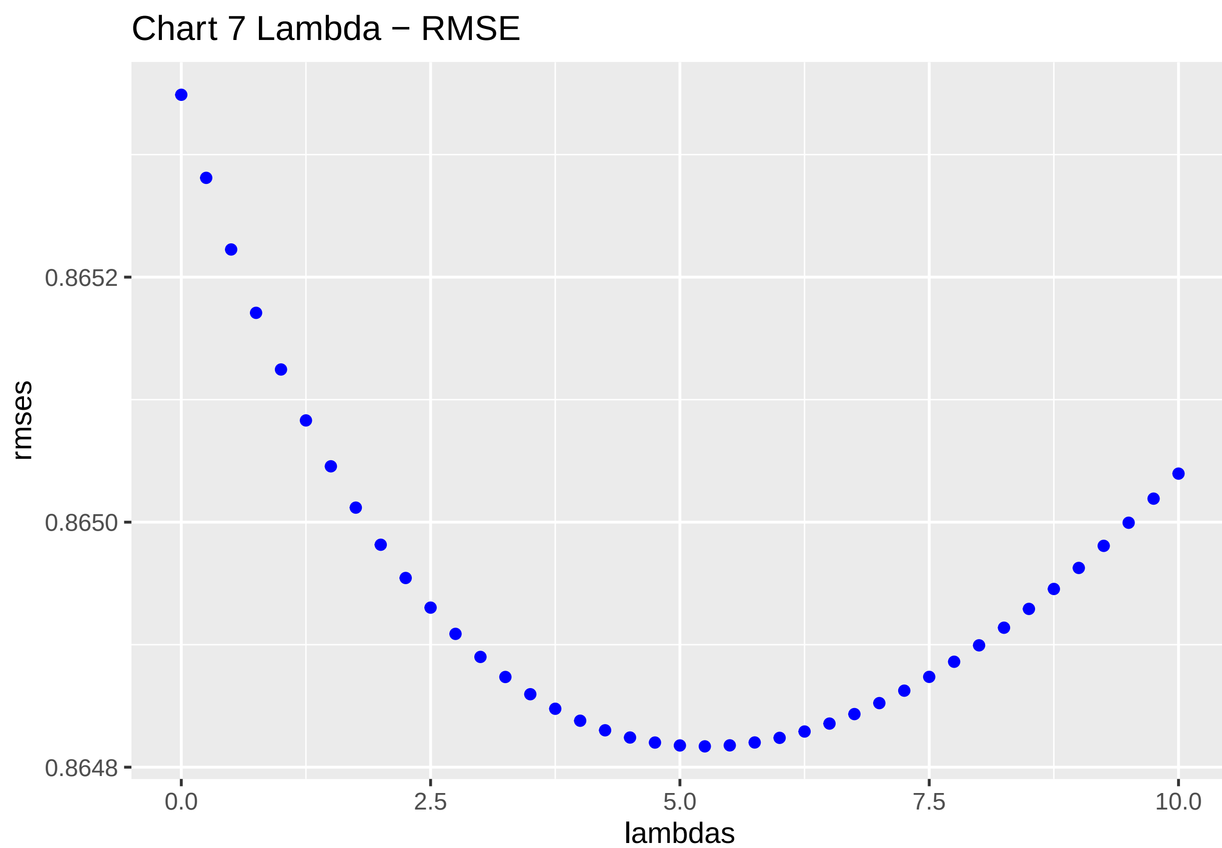

```
user_avgs <- edx %>%
  left_join(movie_avgs, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_i))

predicted_ratings <- edx %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  mutate(pred = mu + b_i + b_u) %>%
  pull(pred)
```

```
model_2_rmse <- RMSE(predicted_ratings, edx$rating)

mse_results <- bind_rows(rmse_results,
                         data.frame(method="Movie and user effect model",
                                    RMSE = model_2_rmse))

print(rmse_results %>% knitr::kable())
```

```
##
##
## |method                     |      RMSE|
## |:--------------------------|---------:|
## |Average movie rating model | 1.0603313|
## |Movie effect model         | 0.9423475|
```

Use regularization to get RMSE, a single number for prediction which minimizes sum of squares while penalizing for large values. Plot rmses vs lambdas to select the optimal lambda

```
lambdas <- seq(0, 10, 0.25)

rmses <- sapply(lambdas, function(l){

  mu <- mean(edx$rating)

  b_i <- edx %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n()+l))

  b_u <- edx %>%
    left_join(b_i, by="movieId") %>%
    group_by(userId) %>%

  summarize(b_u = sum(rating - b_i - mu)/(n()+l))

  predicted_ratings <- validation %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    mutate(pred = mu + b_i + b_u) %>%
    pull(pred)
  RMSE(predicted_ratings, validation$rating)
})

qplot(lambdas, rmses, col = I("blue"),
      main = "Chart 7 Lambda - RMSE")
```




```
lambda <- lambdas[which.min(rmses)]
lambda
```

```
## [1] 5.25
```

```
rmse_results <- bind_rows(rmse_results,
                          data.frame(method="Regularized movie and user effect model",
RMSE = min(rmses)))
```

RESULTS

```
rmse_results %>% knitr::kable()
```

method	RMSE
Average movie rating model	1.0603313
Movie effect model	0.9423475
Regularized movie and user effect model	0.8648170

CONCLUSION

The caret packages proves dependable for machine language modeling, which in this case, builds a basic recommendation system using two predictors, movies and users.