MovieLens.R

rharidas

2021-11-11

```
# Execute the given source code for the project
source("ProjGivenCode.R")
```

```
## Loading required package: tidyverse

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

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.2     v dplyr   1.0.7
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.1

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

## Loading required package: caret

## Loading required package: lattice

##
## Attaching package: 'caret'

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

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

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

```
library(caret)
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
library(kableExtra)
```

```
##
## Attaching package: 'kableExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     group_rows
```

```
set.seed(1996,sample.kind="Rounding")
```

```
## Warning in set.seed(1996, sample.kind = "Rounding"): non-uniform 'Rounding'
## sampler used
```

```
# Split the development dataset 90% - training set and 10% test set
test_index <- createDataPartition(y = edx$rating, times = 1, p = 0.1,
                                  list = FALSE)
train_set <- edx[-test_index,]

# create a test set to assess the accuracy of the models implemented during development.
temp <- edx[test_index,]

# Exclude users and movies in the test set that do not appear in the training set using the semi_join
test_set <- temp %>%
  semi_join(train_set, by = "movieId") %>%
  semi_join(train_set, by = "userId")

removed <- anti_join(temp, test_set)
```

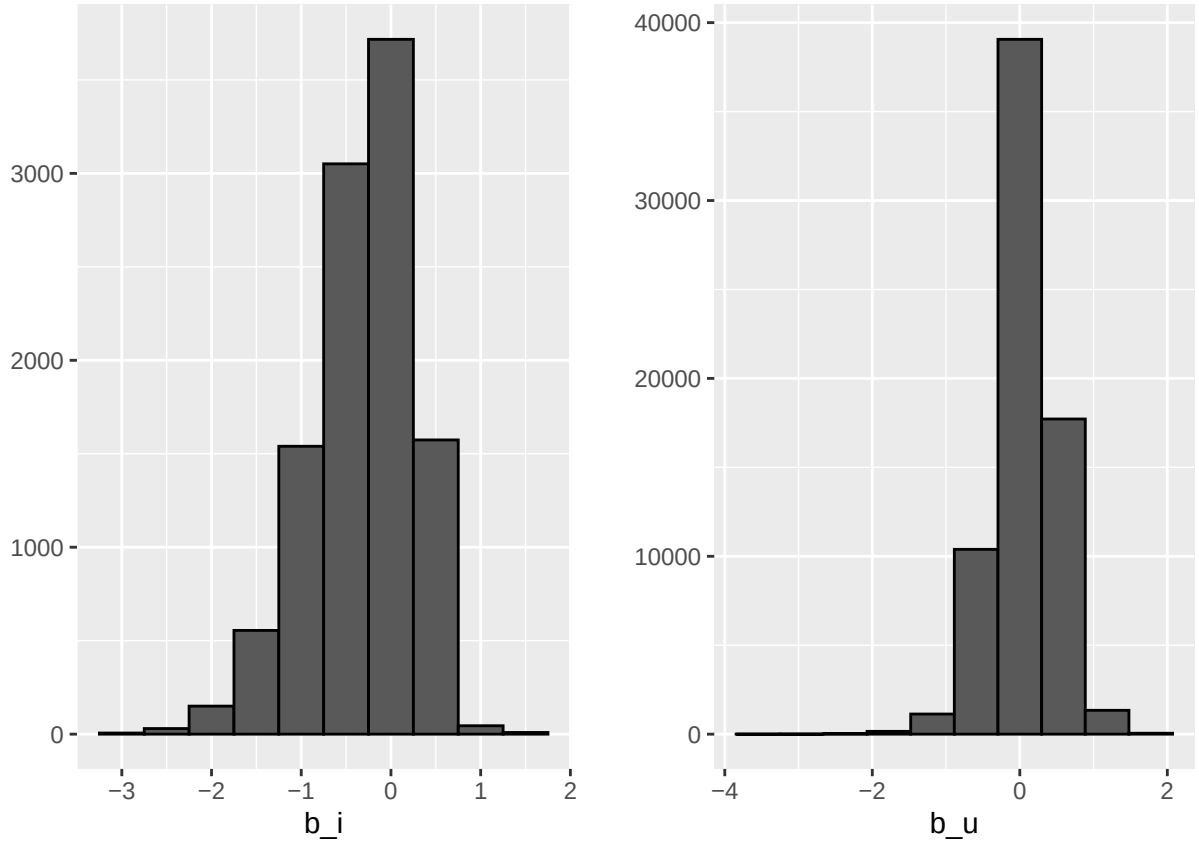
```
## Joining, by = c("userId", "movieId", "rating", "timestamp", "title", "genres")
```

```
train_set <- rbind(train_set, removed) #add back removed items

#remove temporary data to save space
rm(removed, temp)
```

```r
# create a function that computes the RMSE for vectors of ratings and their corresponding predictors:
RMSE <- function(true_ratings, predicted_ratings){
  sqrt(mean((true_ratings - predicted_ratings)^2))
}

# A model with just rating and movie explained by random variation would look like this:

# mean rating
mu_hat <- mean(train_set$rating)

naive_rmse <- RMSE(test_set$rating,mu_hat)

# RMSE for plain model with same rating for all movies
naive_rmse
```

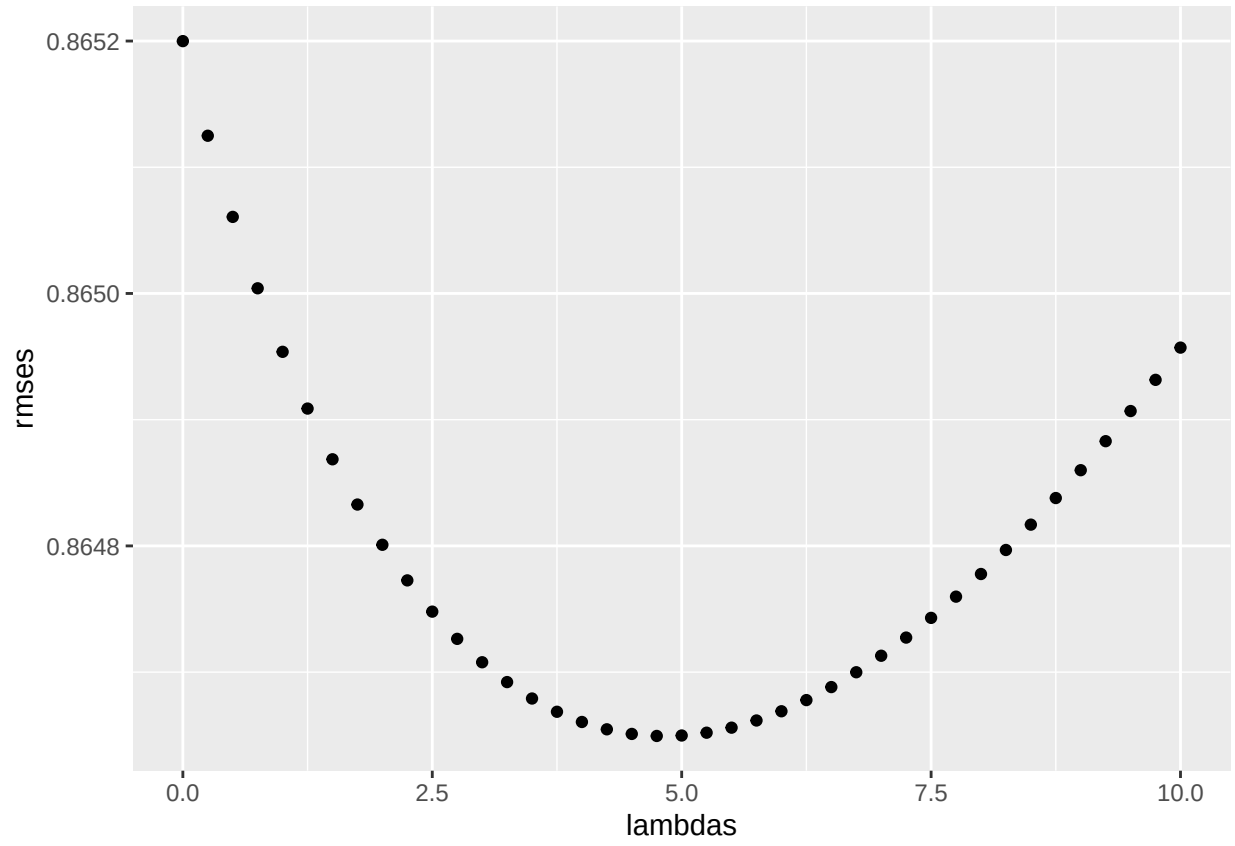
```
## [1] 1.059691
```

```r
# the least squares estimate b_i_hat is just the average of mean of rating - avg. rating
b_i_hat <- train_set %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu_hat))

# quick plot to review least square estimates for movie bias
movie_plot <- b_i_hat %>% qplot(b_i, geom ="histogram", bins = 10, data = ., color = I("black"))

# Calculate the predicted ratings and RMSE for just the movieid
predicted_ratings <- mu_hat + test_set %>%
  left_join(b_i_hat, by='movieId') %>%
  pull(b_i)

rmse_movie <- RMSE(predicted_ratings, test_set$rating)

# now include user id in the calculation of b_u_hat
b_u_hat <- train_set %>%
  left_join(b_i_hat, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu_hat - b_i))

# quick plot to review least square estimates for user bias
user_plot <- b_u_hat %>% qplot(b_u, geom ="histogram", bins = 10, data = ., color = I("black"))

#review the two bias effect in a side-by-side plot
grid.arrange(movie_plot, user_plot,  ncol=2)
```




```
# Calculate the predicted ratings and RMSE for the movieid and user id together
predicted_ratings_w_user <- test_set %>%
  left_join(b_i_hat, by='movieId') %>%
  left_join(b_u_hat, by='userId') %>%
  mutate(pred = mu_hat + b_i + b_u) %>%
  pull(pred)

rmse_movie_user <- RMSE(predicted_ratings_w_user, test_set$rating)

# Regularization
# Choosing the penalty term (lambda)
lambdas <- seq(0, 10, 0.25)

# iterate through the sequence of lambdas and compute the ratings and RMSEs
rmses <- sapply(lambdas, function(l){

  mu_hat <- mean(train_set$rating)

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

  b_u <- train_set %>%
    left_join(b_i, by="movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - b_i - mu_hat)/(n()+l))
```

```
  predicted_ratings <-
    test_set %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    mutate(pred = mu_hat + b_i + b_u) %>%
    pull(pred)

  return(RMSE(predicted_ratings, test_set$rating))
})

# verify the lambda penalty term
qplot(lambdas, rmses)
```




```
# find the lambda with the lowest RMSE
lambda <- lambdas[which.min(rmses)]

#optimal lambda
lambda
```

```
## [1] 4.75
```

```
# compute these regularized estimates for movie and user effect
mu_hat <- mean(train_set$rating)
b_i_hat <- train_set %>%
```

	method	RMSE
a.	Just the movie	0.943035
b.	Movie and User	0.8652
c.	Movie and User regularized	0.864649
d.	Regularized RMSE final test	0.86466

```
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu_hat)/(n()+lambda), n_i = n())

b_u_hat <- train_set %>%
  left_join(b_i_hat, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating - mu_hat - b_i)/(n()+lambda), n_i = n())

# Compute the predictions and RMSE using regularized estimates
predicted_ratings_regularized <- test_set %>%
  left_join(b_i_hat, by = "movieId") %>%
  left_join(b_u_hat, by = "userId") %>%
  mutate(pred = mu_hat + b_i  + b_u) %>%
  pull(pred)

rmse_movie_user_regularized <- RMSE(predicted_ratings_regularized, test_set$rating)

# Get the validation set
validation_test_index <- createDataPartition(y = validation$rating, times = 1, p = 0.1,
                                             list = FALSE)
validation_test_set <- validation[validation_test_index, ]

# test the regularized estimates on the validation set
validation_ratings_regularized <- validation_test_set %>%
  left_join(b_i_hat, by = "movieId") %>%
  left_join(b_u_hat, by = "userId") %>%
  mutate(pred = mu_hat + b_i  + b_u) %>%
  pull(pred)

# compute the RMSE for the validation set
rmse_validation_test <- RMSE(validation_ratings_regularized,validation_test_set$rating)

# tabulate all the RMSE results
rmse_results <- matrix( c("Just the movie", round(rmse_movie,6),
                          "Movie and User", round(rmse_movie_user,6),
                          "Movie and User regularized", round(rmse_movie_user_regularized,6),
                          "Regularized RMSE final test", round(rmse_validation_test,6)
                        ),
                        nrow = 4, ncol=2, byrow=TRUE,
                        dimnames=list(c("a.","b.","c.","d."),c("method","RMSE")))


rmse_results %>% knitr::kable() %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed"))
```