MovieRecommender.R

santhoshthirumalai

2019-06-13

```
library(ggplot2)
library(tidyverse)

## ── Attaching packages ──────────────────────────────────────────────────────────────────────────────
tidyverse 1.2.1 ──

## ✔ tibble  2.1.2     ✔ purrr   0.3.2
## ✔ tidyr   0.8.3     ✔ dplyr   0.8.1
## ✔ readr   1.3.1     ✔ stringr 1.4.0
## ✔ tibble  2.1.2     ✔ forcats 0.4.0

## ── Conflicts ──────────────────────────────────────────────────────────────────────────────────────────
tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()

library(caret)

## Loading required package: lattice

##
## Attaching package: 'caret'

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

library(dslabs)
library(matrixStats)

##
## Attaching package: 'matrixStats'

## The following object is masked from 'package:dplyr':
##
##     count

library(lubridate)

##
## Attaching package: 'lubridate'
```

```
## The following object is masked from 'package:base':
##
##     date
```

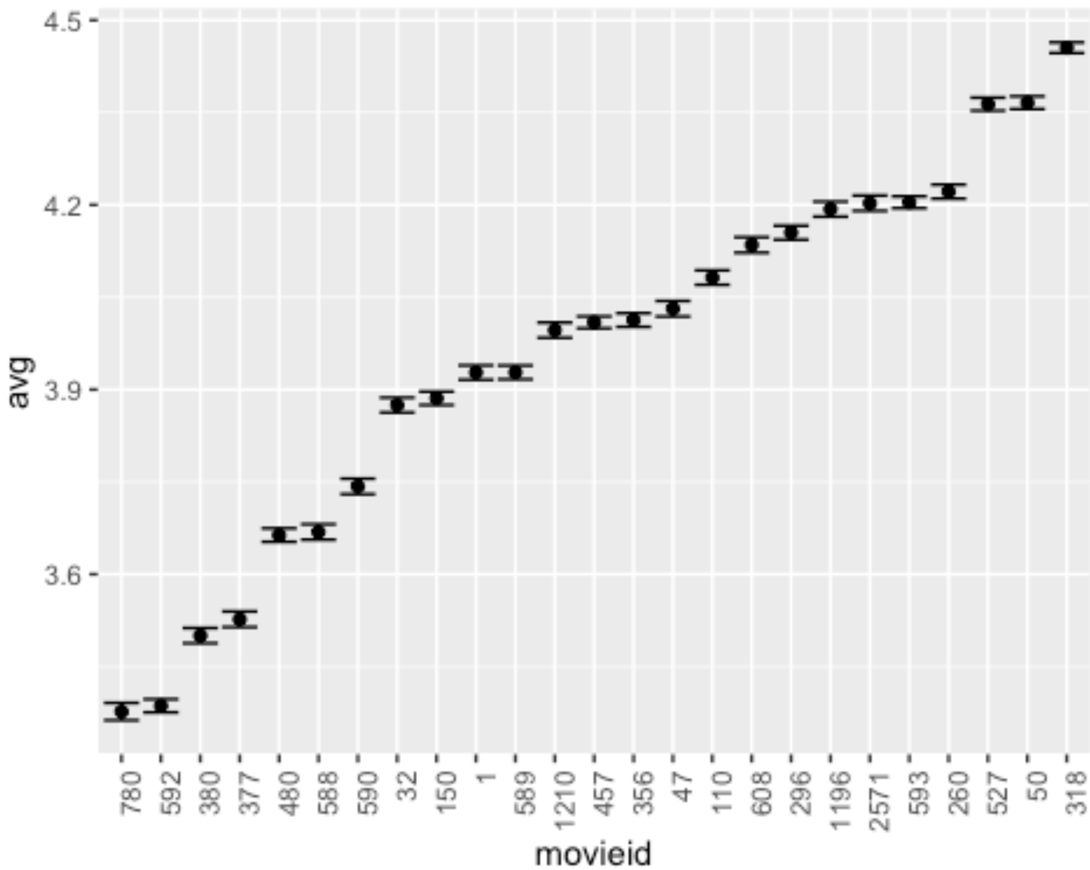
```
#################################################################
#  Attention: The R version used to run this algorithm is 3.5.3
#################################################################



###################################
# Create edx set and validation set
###################################

# Note: this process could take a couple of minutes

if(!require(tidyverse)) install.packages("tidyverse", repos =
"http://cran.us.r-project.org")
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-
project.org")

# MovieLens 10M dataset:
# https://grouplens.org/datasets/movielens/10m/
# http://files.grouplens.org/datasets/movielens/ml-10m.zip

dl <- tempfile()
download.file("http://files.grouplens.org/datasets/movielens/ml-10m.zip", dl)

ratings <- read.table(text = gsub("::", "\t", readLines(unzip(dl, "ml-
10M100K/ratings.dat"))),
                      col.names = c("userId", "movieId", "rating",
"timestamp"))

movies <- str_split_fixed(readLines(unzip(dl, "ml-10M100K/movies.dat")),
"\\::", 3)
colnames(movies) <- c("movieId", "title", "genres")
movies <- as.data.frame(movies) %>% mutate(movieId =
as.numeric(levels(movieId))[movieId],
                                           title = as.character(title),
                                           genres = as.character(genres))

movielens <- left_join(ratings, movies, by = "movieId")

# Validation set will be 10% of MovieLens data

set.seed(1) # if using R 3.6.0: set.seed(1, sample.kind = "Rounding")
test_index <- createDataPartition(y = movielens$rating, times = 1, p = 0.1,
list = FALSE)
edx <- movielens[-test_index,]
temp <- movielens[test_index,]
```

```r
# Make sure userId and movieId in validation set are also in edx set

validation <- temp %>%
  semi_join(edx, by = "movieId") %>%
  semi_join(edx, by = "userId")

# Add rows removed from validation set back into edx set

removed <- anti_join(temp, validation)

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

edx <- rbind(edx, removed)

rm(dl, ratings, movies, test_index, temp, movielens, removed)

#========================================================================#
#  Algorithm to predict the rating and compute RMSE
#========================================================================#

# Method to compute the RMSE

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

}

#========================================================================#
#                             INTRODUCTION
#========================================================================#
# The GOAL for this algorithm is to Recommend a movie based on predicting
# the rating using a training dataset called "edx" and test dataset called
# "Validation" which is scrapped from movielens dataset. The script
#  uses a linear model (explained below) to predict the outcomes.
# The Success of the algorithm will be measured based on the Root Mean
# Square Value (RMSE). That is to acheive an RMSE < 1 and most preferred
# RMSE would be < 0.87750
#========================================================================#



#========================================================================#
#  A Linear model with Average rating and different BIASES are used as
#  Predictors in this Algorithm
#========================================================================#
# Y_hat = mu + b_i + b_u + b_g + b_t
# mu = Average rating of all movies
```

```r
# b_i = Bias based on Movies
# b_u = Bias based on Users
# b_g = Bias based on Genres
# b_t = Bias based on Date the movie is rated
#==========================================================================#

# Compute mu - mean rating for all the movies
# which will be the best prediction with just the ratings
mu<-mean(edx$rating)

# Compute the RMSE for the model and store it in a Dataframe
naive_rmse<-RMSE(validation$rating,mu)

RMSE_Movie_var<-data_frame(method="RMSE for Average Rating",RMSE=naive_rmse)
```

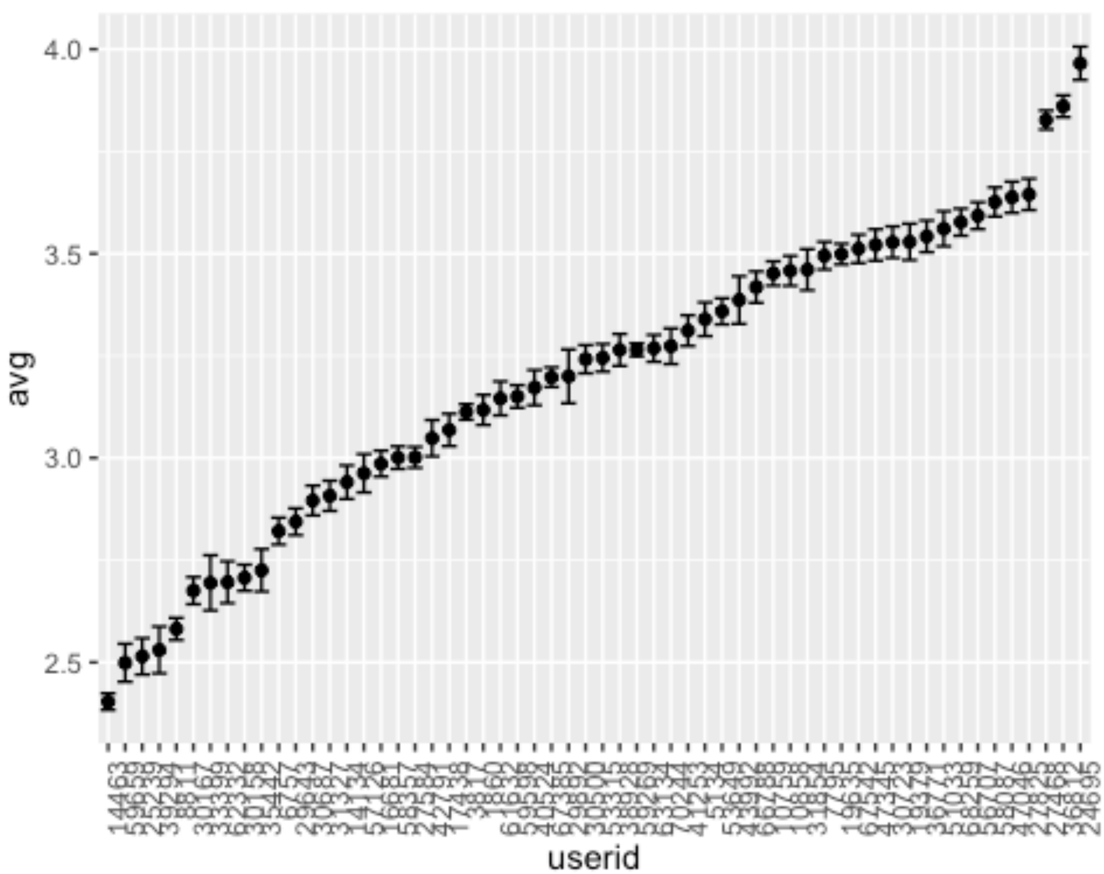
```
## Warning: `data_frame()` is deprecated, use `tibble()`.
## This warning is displayed once per session.
```

```r
#==========================================================================#
# Add a BIAS based on the movies / Users / Genres / Date
#==========================================================================#

#==========================================================================#
# Plot the Movie ID against the rating to prove the rating is
# dependent on MovieId
#==========================================================================#
# Get the mean rating of the movies whose count of ratings are greater
# than = 20000 and it is evident that not all the movies are rated
# equally and there is a bias identified and hence can be used in the
# model
#==========================================================================#

edx %>% group_by(movieId) %>%
  summarize(n = n(), avg = mean(rating), se = sd(rating)/sqrt(n())) %>%
  filter(n >= 20000) %>%
  mutate(movieid = reorder(movieId, avg)) %>%
  ggplot(aes(x = movieid, y = avg, ymin = avg - 2*se, ymax = avg + 2*se)) +
  geom_point() +
  geom_errorbar() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```

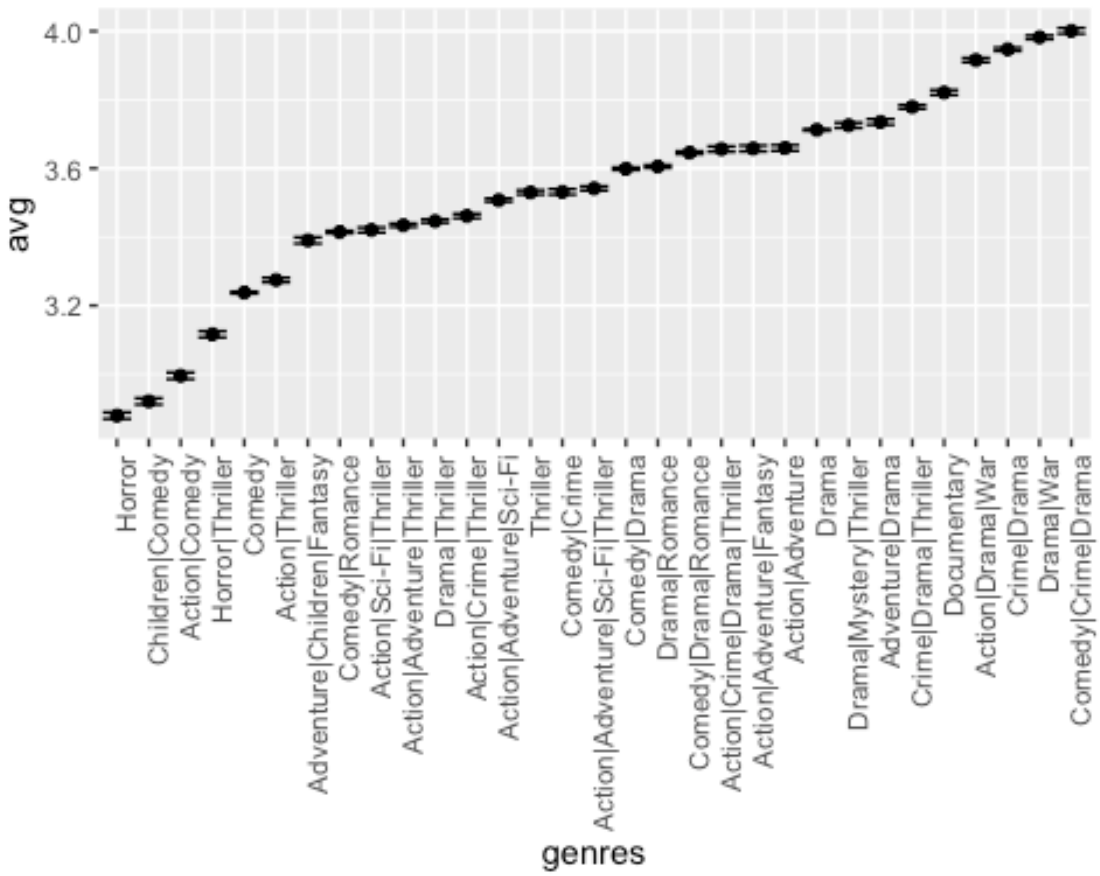



```
#=========================================================================#
# Compute the BIAS b_i based on the movies
#=========================================================================#
movie_avg<-edx %>% group_by(movieId) %>% summarize(b_i = mean(rating-mu))

#=========================================================================#
# Improve the predicted value mu by adding the BIAS based on the movies
#=========================================================================#
pred_with_movie_bias <- validation %>% left_join(movie_avg, by='movieId') %>%
.$b_i

pred_with_movie_bias <- pred_with_movie_bias + mu

#=============================================================================#
# Compute the RMSE for the model with Movie BIAS and store it in a Dataframe
#=============================================================================#

movie_bias_rmse<-RMSE(validation$rating,pred_with_movie_bias)

RMSE_Movie_var<-bind_rows(RMSE_Movie_var,data_frame(method="RMSE with Movie
BIAS",RMSE=movie_bias_rmse))
```

```
# @@@@@@@@@@@@@@@@@@@ End of Prediction based on Movie Bias @@@@@@@@@@@@#


# @@@@@@@@@@@@@@@@@@@ Start of Prediction based on User Bias @@@@@@@@@@@#

#=========================================================================#
# Plot the User ID against the rating to prove the rating is
# dependent on UserId
#=========================================================================#
#=========================================================================#
# Get the mean rating of the movies based on the users who has rated
# movies 2000 or more times it is evident that not all users rated
# the movies the same and hence a bias is identified to fit the model
#=========================================================================#

edx %>% group_by(userId) %>%
  summarize(n = n(), avg = mean(rating), se = sd(rating)/sqrt(n)) %>%
  filter(n >= 2000) %>%
  mutate(userid = reorder(userId, avg)) %>%
  ggplot(aes(x = userid, y = avg, ymin = avg - 2*se, ymax = avg + 2*se)) +
  geom_point() +
  geom_errorbar() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```

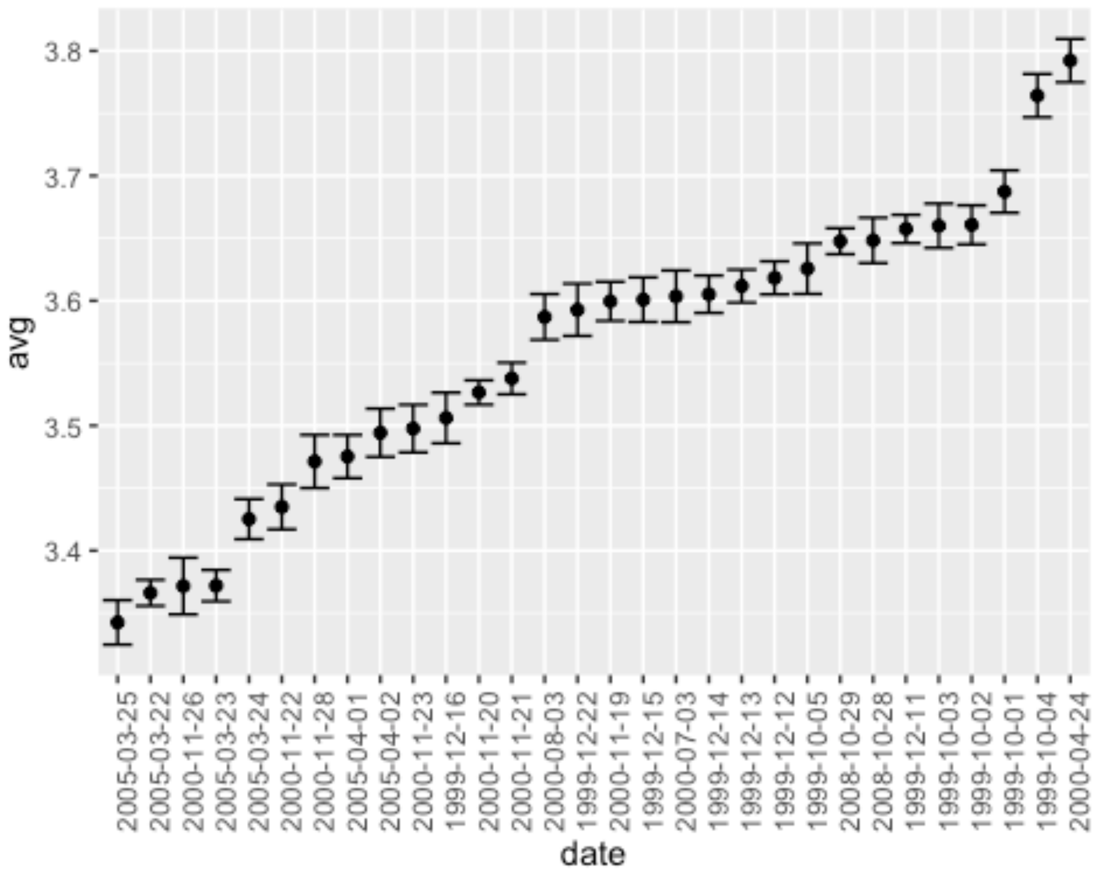


```r
#=======================================================================#
# Compute BIAS b_u on users rated the movies
#=======================================================================#

user_avgs<-edx %>% left_join(movie_avg,by="movieId") %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_i))
#=======================================================================#
# Improve the predicted value mu by adding the BIAS
# based on the movies and BIAS based on users
#=======================================================================#
pred_with_user_bias <- validation %>% left_join(movie_avg, by='movieId') %>%
  left_join(user_avgs,by='userId') %>% mutate(pred = mu + b_i + b_u) %>%
.$pred

#=========================================================================#
# Compute the RMSE for the model with Movie BIAS and store it in a Dataframe
#=========================================================================#
user_bias_rmse<-RMSE(validation$rating,pred_with_user_bias)

RMSE_Movie_var<-bind_rows(RMSE_Movie_var,data_frame(method="RMSE after USER
BIAS",RMSE=user_bias_rmse))

# @@@@@@@@@@@ End of Prediction based on User Bias @@@@@@@@@@@@@@@@@@@#

# @@@@@@@@@@ Start of Prediction based on Genre Bias @@@@@@@@@@@@@@@@@#

#======================================================================#
# Plot the Genre against the rating to prove the rating is
# dependent on Genre
#======================================================================#

#=======================================================================#
# Get the mean rating of the movies based on the genres which were rated
# 50000 or more times and it is evident that not all genres are rated
# the same and hence a bias is identified to fit the model.
# Note that the graph depicts the Genre effect is not that significant
# and the impact of this BIAS on the model is negligible.
#=======================================================================#

edx %>% group_by(genres) %>%
  summarize(n = n(), avg = mean(rating), se = sd(rating)/sqrt(n())) %>%
  filter(n >= 50000) %>%
  mutate(genres = reorder(genres, avg)) %>%
  ggplot(aes(x = genres, y = avg, ymin = avg - 2*se, ymax = avg + 2*se)) +
  geom_point() +
  geom_errorbar() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```




```
#=========================================================================#
# Compute BIAS b_g on genres of the movies
#=========================================================================#

genre_avgs<-edx %>% left_join(movie_avg,by="movieId") %>%
  left_join(user_avgs,by="userId")%>%group_by(genres) %>%
  summarize(b_g = mean(rating - mu - b_i - b_u))

#=========================================================================#
# Improve the predicted value mu by adding the
# BIAS based on the movies/users/Genres
#=========================================================================#
pred_with_genre_bias <- validation %>% left_join(movie_avg, by='movieId') %>%
  left_join(user_avgs,by='userId') %>% left_join(genre_avgs, by='genres') %>%
  mutate(pred_genre = mu + b_i + b_u + b_g) %>% .$pred_genre

#============================================================================#
# Compute the RMSE for the model with Genre BIAS and store it in a Dataframe
#============================================================================#
genre_bias_rmse<-RMSE(validation$rating,pred_with_genre_bias)

RMSE_Movie_var<-bind_rows(RMSE_Movie_var,data_frame(method="RMSE after GENRE
BIAS",RMSE=genre_bias_rmse))
```

```r
# @@@@@@@@@@@@ End  of Prediction based on Genre Bias @@@@@@@@@@@@@@@@@@@@@#

# @@@@@@@@@@@@ Start of Prediction based on Date Bias @@@@@@@@@@@@@@@@@@@@@#

# Transform the time stamp into date in edx and validation sets

edx_new <- edx %>% mutate(date = as_date(as_datetime(timestamp)))

validation_new <- validation %>% mutate(date =
as_date(as_datetime(timestamp)))

#=========================================================================#
# Plot the Genre against the rating to prove the rating is
# dependent on Date when a movie is rated
#=========================================================================#

#=========================================================================#
# Get the mean rating of the movies which were rated 10000 or more
# times on a given day and it is evident that there is a BIAS.
# Even though there is a BIAS it is not that significant compared to
# users and Movies. So the effect of this bias on the model is negligible
#=========================================================================#

edx_new %>% group_by(date) %>%
  summarize(n = n(), avg = mean(rating), se = sd(rating)/sqrt(n())) %>%
  filter(n >= 10000) %>%
  mutate(date = reorder(date, avg)) %>%
  ggplot(aes(x = date, y = avg, ymin = avg - 2*se, ymax = avg + 2*se)) +
  geom_point() +
  geom_errorbar() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```

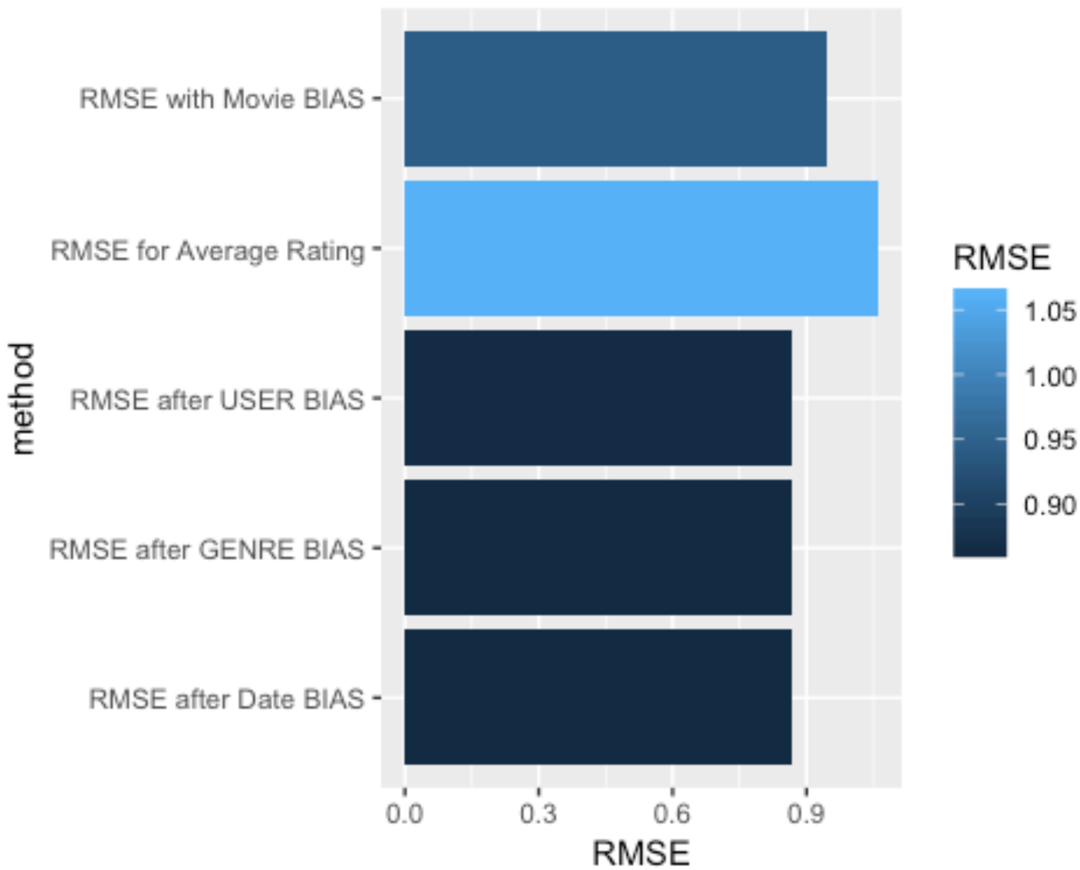



```
#========================================================================#
# Compute BIAS b_t on time_stamp of the ratings
#========================================================================#

ts_avgs<-edx_new %>% left_join(movie_avg,by="movieId") %>%
  left_join(user_avgs,by="userId")%>% left_join(genre_avgs,by="genres") %>%
  group_by(date) %>%
  summarize(b_t = mean(rating - mu - b_i - b_u - b_g))

#========================================================================#
# Improve the predicted value mu by adding the
# BIAS based on the movies/users/Genres/Date
#========================================================================#
pred_with_date_bias <- validation_new %>% left_join(movie_avg, by='movieId')
%>%
  left_join(user_avgs,by='userId') %>% left_join(genre_avgs, by='genres') %>%
  left_join(ts_avgs, by='date') %>%
  mutate(pred_date = mu + b_i + b_u + b_g + b_t) %>% .$pred_date

#===========================================================================#
# Compute the RMSE for the model with Genre BIAS and store it in a Dataframe
#===========================================================================#
ts_bias_rmse<-RMSE(validation$rating,pred_with_date_bias)
```

```
RMSE_Movie_var<-bind_rows(RMSE_Movie_var,data_frame(method="RMSE after Date
BIAS",RMSE=ts_bias_rmse))

# @@@@@@@@@@@@@@ End of Prediction based on Date Bias @@@@@@@@@@@@@@@@@@@@@@#


#===========================================================================#
#                      RESULTS
#===========================================================================#
#        DISPLAY THE RMSEs COMPUTED FOR DIFFERENT BIASES
#===========================================================================#

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

method	RMSE
RMSE for Average Rating	1.0612018
RMSE with Movie BIAS	0.9439087
RMSE after USER BIAS	0.8653488
RMSE after GENRE BIAS	0.8649469
RMSE after Date BIAS	0.8644285

```
#===========================================================================#
#                     CONCLUSION
#===========================================================================#
#  The plot below shows how the RMSE value decreases when the BIASES
#  were added to the mean which minimizes the errors and acheived the
#  GOAL of having RMSE < 0.8750
#===========================================================================#

RMSE_Movie_var %>%
  mutate(name = fct_reorder(method, desc(RMSE))) %>%
  ggplot( aes(x=method, y=RMSE, fill=RMSE)) +
  geom_bar(stat="identity") +
  coord_flip()
```



RMSE with Movie BIAS
RMSE for Average Rating
RMSE after USER BIAS
RMSE after GENRE BIAS
RMSE after Date BIAS
method
0.0
0.3
0.6
0.9
RMSE
RMSE
1.05
1.00
0.95
0.90


```
#==============================================================
```