CAPSTONE_Data_Science_Manfrini_Nicola

Nicola Manfrini

30/06/2022

INTRODUCTION/OVERVIEW/EXECUTIVE SUMMARY

This dataset “movielens” was previously used during the data Science professional certificate courses. It was generated from a dataset of the Grouplens research lab which was based on a larger NEtflix database, that was itself used to predict user rating for films in 2006 in the so-called Netflix Prize. Fortunately Movielens is a much smaller/filtered and thus user-friendly database.

The idea is to determine a model to predict ratings of a movie in order to obtain the lowest Residual Mean Squared Error (RMSE) possible. My approach was to apply a linear model starting from the mean

METHODS Ok,ready to start the movielens exercise. First thing I will load all that is needed for perfoming the analysis and exercise

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

```
## Loading required package: tidyverse

## Warning: package 'tidyverse' was built under R version 4.0.5

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

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.3     v dplyr   1.0.7
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   2.0.0     v forcats 0.5.1

## Warning: package 'ggplot2' was built under R version 4.0.5

## Warning: package 'tibble' was built under R version 4.0.5

## Warning: package 'tidyr' was built under R version 4.0.5

## Warning: package 'readr' was built under R version 4.0.5

## Warning: package 'dplyr' was built under R version 4.0.5

## Warning: package 'stringr' was built under R version 4.0.2

## Warning: package 'forcats' was built under R version 4.0.5

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

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

```
## Loading required package: caret

## Warning: package 'caret' was built under R version 4.0.5

## Loading required package: lattice

##
## Attaching package: 'caret'

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

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

```
## Loading required package: data.table

## Warning: package 'data.table' was built under R version 4.0.5

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

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

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

```r
library(tidyverse)
library(caret)
library(data.table)

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

ratings <- fread(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(movieId),
                                           title = as.character(title),
                                           genres = as.character(genres))

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

Next I will generate the final train set and test on which to evaluate the model on. The validation (test set) will be 10% of the entire Movielens dataset

```
set.seed(1, sample.kind="Rounding")
```

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

```
test_index <- createDataPartition(y = movielens$rating, times = 1, p = 0.1, list = FALSE)
edx <- movielens[-test_index,]
temp <- movielens[test_index,]
# following code is to test and confirm that userId and movieId in validation set are also present in t
validation <- temp %>%
  semi_join(edx, by = "movieId") %>%
  semi_join(edx, by = "userId")
# Next rows previosly removed from the validation set will be added 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)
```

Now i will start the project following the following instructions that were given:

#you will train a machine learning algorithm using the inputs in one subset to predict movie ratings in the validation set. #Your project itself will be assessed by peer grading.

#Predict movie ratings in the validation set (the final hold-out test set) as if they were unknown. #RMSE will be used to evaluate how close your predictions are to the true values in the validation set #(the final hold-out test set).

I will start by sub-partitioning edx into test and training set, I will use them for generating and testing my models: I will prepare models on the sub-training set and test them on the sub-test set

```
library(dslabs)
```

```
## Warning: package 'dslabs' was built under R version 4.0.5
```

```
test_index <- createDataPartition(y = edx$rating, times = 1,
                                  p = 0.2, list = FALSE)

train_set <- edx[-test_index,]
test_set <- edx[test_index,]
test_set <- test_set %>%
  semi_join(train_set, by = "movieId") %>%
  semi_join(train_set, by = "userId")
```

Before anything we need to define RMSE which will be the way to evaluate the efficay of our models. We will also define mu which is the mean of the values we want to predict. From mu and the RMSE of mu compared to true ratings we can only do better.

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

mu <- mean(train_set$rating)
# let's define our starting RMSE from which we can only do better #
model_1_rmse <- RMSE(test_set$rating, mu)
rmse_results <- (data_frame(method="Just the mean model",RMSE = model_1_rmse ))
```

```
## Warning: `data_frame()` was deprecated in tibble 1.1.0.
## Please use `tibble()` instead.
```

RMSE of mu relative to true rating values is really high: 1.0607, from here we can only improve, we can only improve Starting from the mean of rating values we will develop a linear regression model to predict ratings. Let's begin implementing our model starting from one simple assumption By experience for sure some genres are rated more than others First thing let's see if this holds true

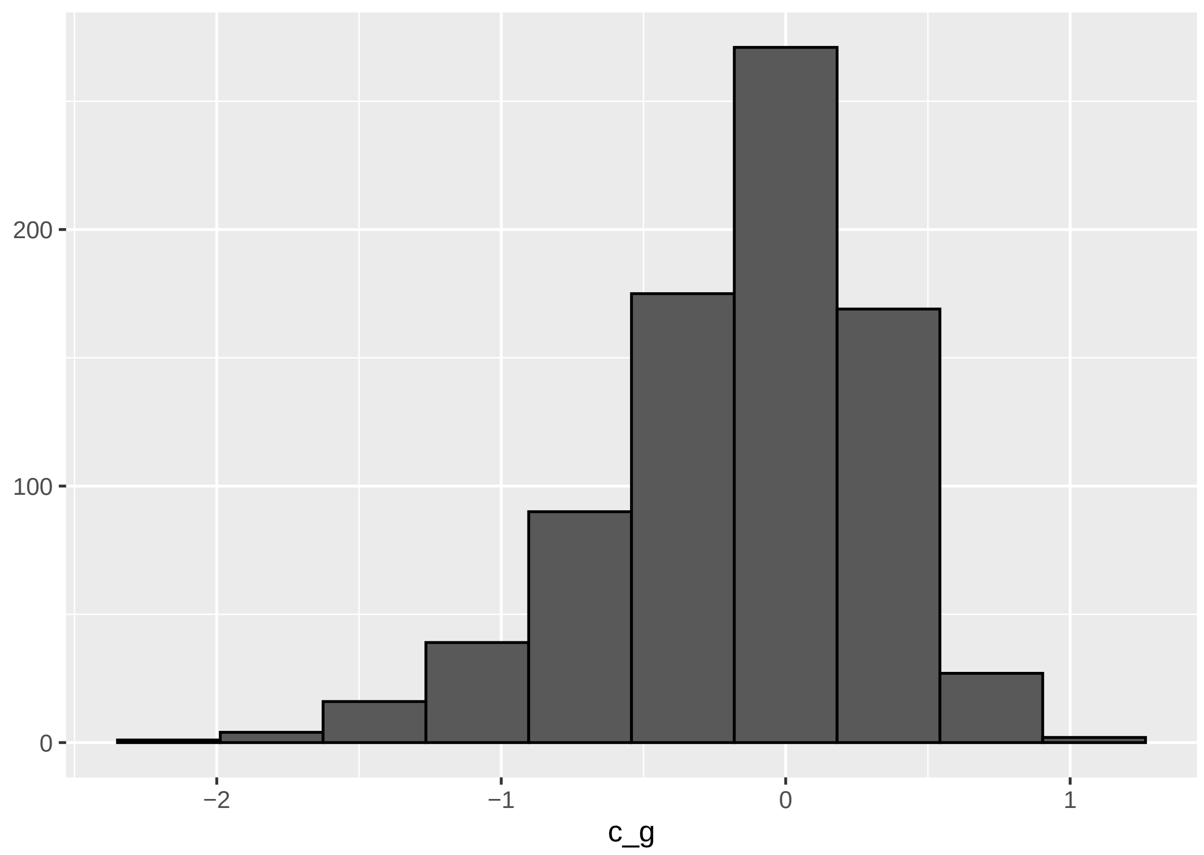
```
# let's have a look at what we are dealing with to have an idea of the predictors we can use to impleme
names(edx)
```

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

```
genres_c <- train_set %>%
  group_by(genres) %>%
  summarize(c_g = mean(rating - mu))
genres_c
```

```
## # A tibble: 794 x 2
##    genres                                                 c_g
##    <chr>                                                <dbl>
##  1 (no genres listed)                                  0.154
##  2 Action                                             -0.576
##  3 Action|Adventure                                    0.147
##  4 Action|Adventure|Animation|Children|Comedy          0.452
##  5 Action|Adventure|Animation|Children|Comedy|Fantasy -0.513
##  6 Action|Adventure|Animation|Children|Comedy|IMAX    -0.276
##  7 Action|Adventure|Animation|Children|Comedy|Sci-Fi  -0.444
##  8 Action|Adventure|Animation|Children|Fantasy        -0.811
##  9 Action|Adventure|Animation|Children|Sci-Fi         -0.535
## 10 Action|Adventure|Animation|Comedy|Drama             0.00202
## # ... with 784 more rows
```

```
qplot(c_g, data = genres_c, bins = 10, color = I("black"))
```



It seems ok, lets predict a simple model using genre bias

```
predicted_ratings <- test_set %>%
  left_join(genres_c, by='genres') %>%
  mutate(pred = mu + c_g) %>%
  .$pred
model_2_rmse <- RMSE(predicted_ratings, test_set$rating)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Genres Model",
                                     RMSE = model_2_rmse ))

# let's check if it is ok #
rmse_results
```

```
## # A tibble: 2 x 2
##   method               RMSE
##   <chr>               <dbl>
## 1 Just the mean model  1.06
## 2 Genres Model         1.02
```

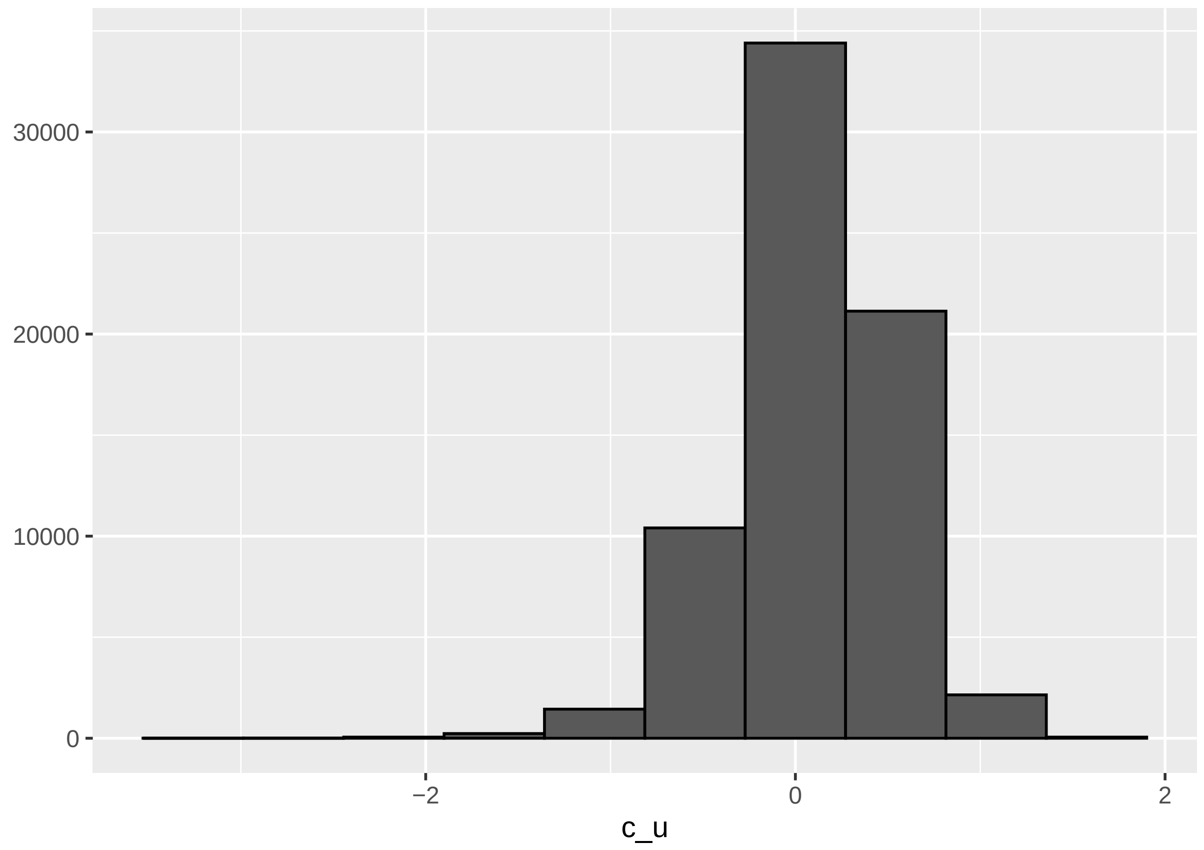
```
#we did a little better
```

Let's implement using the user predictor, as users for sure have an effect on rating value

```r
# let's see if users also have an effect on ratings, if affermative we will also use them to prepare a
user_c <- train_set %>%
  left_join(genres_c, by='genres') %>%
  group_by(userId) %>%
  summarize(c_u = mean(rating - mu - c_g))
user_c
```

```
## # A tibble: 69,878 x 2
##    userId      c_u
##     <int>    <dbl>
##  1      1  1.64
##  2      2 -0.195
##  3      3  0.437
##  4      4  0.729
##  5      5  0.260
##  6      6  0.423
##  7      7  0.200
##  8      8 -0.00580
##  9      9  0.438
## 10     10  0.209
## # ... with 69,868 more rows
```

```r
qplot(c_u, data = user_c, bins = 10, color = I("black"))
```



```r
# yes also used id inserts a great bias #
## let's implement using users #
# model number 3 #
predicted_ratings <- test_set %>%
  left_join(genres_c, by='genres') %>%
  left_join(user_c, by='userId') %>%
  mutate(pred = mu + c_g + c_u) %>%
  .$pred
model_3_rmse <- RMSE(predicted_ratings, test_set$rating)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Genres + User Model",
                                     RMSE = model_3_rmse ))
# Let's see if we actually improved#
rmse_results
```

```
## # A tibble: 3 x 2
##   method               RMSE
##   <chr>               <dbl>
## 1 Just the mean model 1.06
## 2 Genres Model        1.02
## 3 Genres + User Model 0.942
```

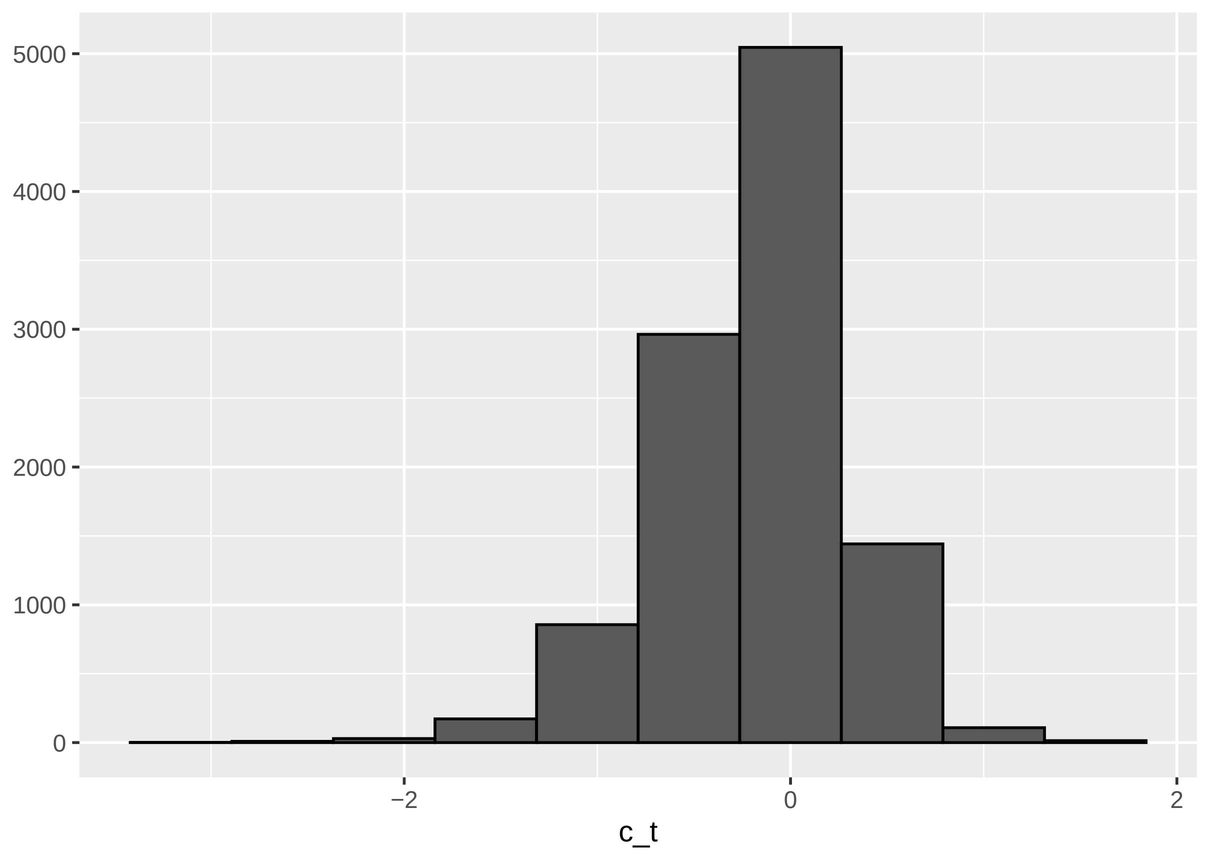
```r
# we did #
```

Let's implement using titles, as titles per se could affect movie rating. Let's see if it is true

```r
title_c <- train_set %>%
  left_join(genres_c, by='genres') %>%
  left_join(user_c, by='userId')%>%
  group_by(title) %>%
  summarize(c_t = mean(rating - mu - c_g - c_u))
title_c
```

```
## # A tibble: 10,640 x 2
##    title                                                              c_t
##    <chr>                                                            <dbl>
##  1 "'burbs, The (1989)"                                            -0.195
##  2 "'night Mother (1986)"                                          -0.223
##  3 "'Round Midnight (1986)"                                         0.335
##  4 "'Til There Was You (1997)"                                     -0.785
##  5 "\"Great Performances\" Cats (1998)"                            -0.674
##  6 "*batteries not included (1987)"                                 0.106
##  7 "...All the Marbles (a.k.a. The California Dolls) (1981)"        0.340
##  8 "...And God Created Woman (Et Dieu... crÃ©a la femme) (1956)" -0.501
##  9 "...And God Spoke (1993)"                                        0.182
## 10 "...And Justice for All (1979)"                                  0.171
## # ... with 10,630 more rows
```

```r
qplot(c_t, data = title_c, bins = 10, color = I("black"))
```



Titles seems to bring a bias on movie ratings. At this point let's use movie titles to implement the model

```
# model number 4 #

predicted_ratings <- test_set %>%
  left_join(genres_c, by='genres') %>%
  left_join(user_c, by='userId') %>%
  left_join(title_c, by='title') %>%
  mutate(pred = mu + c_g + c_u + c_t) %>%
  .$pred
model_4_rmse <- RMSE(predicted_ratings, test_set$rating)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Genres + User + title Model",
                                     RMSE = model_4_rmse ))
# let's see if we did better #
rmse_results
```

```
## # A tibble: 4 x 2
##   method                       RMSE
##   <chr>                       <dbl>
## 1 Just the mean model         1.06
## 2 Genres Model                1.02
## 3 Genres + User Model         0.942
## 4 Genres + User + title Model 0.872
```

```
# this fourth model seems to work really good#

#but let's try a little harder, let's try inserting time, maybe in different periods movies were rated
```

It indeed works good but let's try adding something to improve the model, i.e. something that has to do with the time at which ratings were made. Let's use the month unit

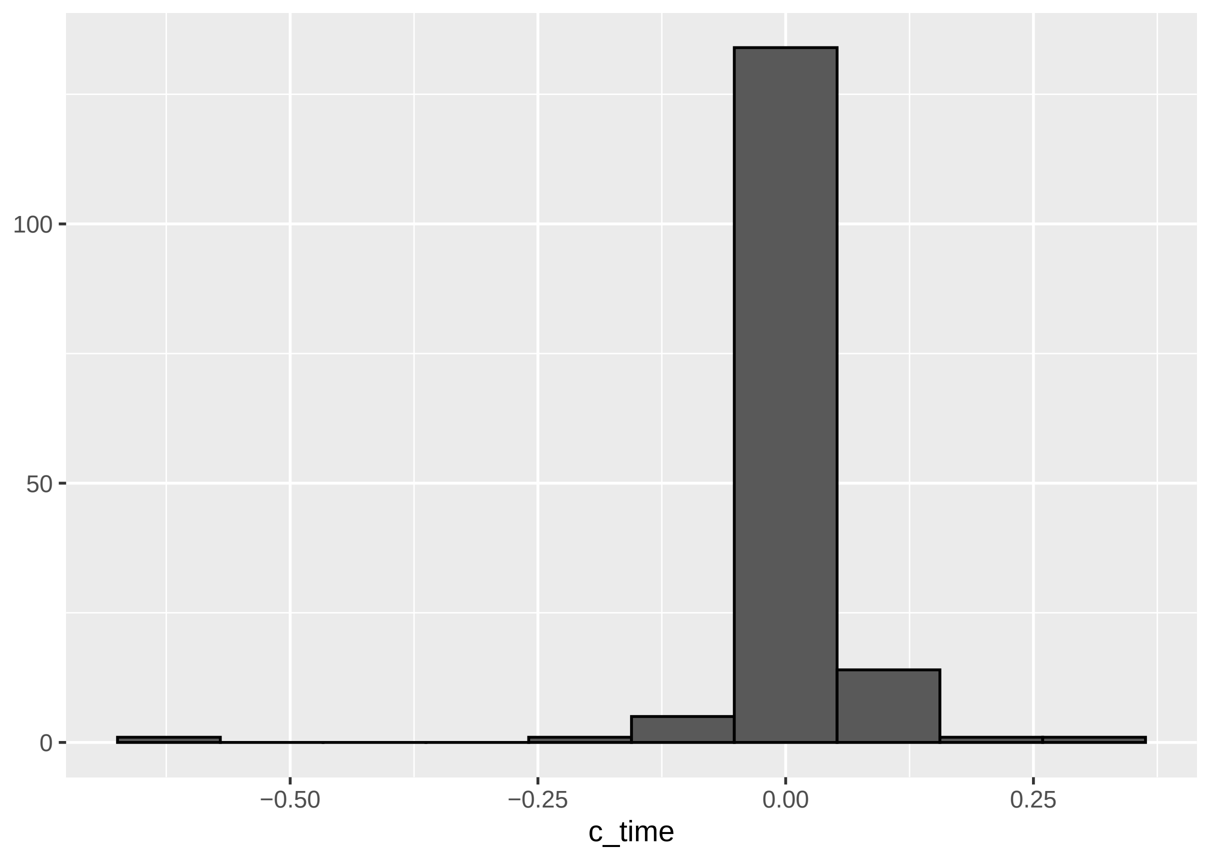
```
# model 5 #
# let's convert timestamp to dates using month as unit #
# to do so we need to install the lubridate package #
if(!require(lubridate)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: lubridate

## Warning: package 'lubridate' was built under R version 4.0.5

##
## 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
```

```
library(lubridate)

time_c <- train_set %>%
  left_join(genres_c, by='genres') %>%
  left_join(user_c, by='userId') %>%
  left_join(title_c, by='title') %>%
  mutate(date = round_date(as_datetime(timestamp), unit = "month")) %>%
  group_by(date) %>%
  summarize(c_time = mean(rating - mu - c_g - c_u - c_t))
time_c
```

```
## # A tibble: 157 x 2
##    date                 c_time
##    <dttm>                <dbl>
##  1 1995-01-01 00:00:00 -0.659
##  2 1996-02-01 00:00:00  0.274
##  3 1996-03-01 00:00:00  0.161
##  4 1996-04-01 00:00:00  0.155
##  5 1996-05-01 00:00:00  0.0968
##  6 1996-06-01 00:00:00  0.0696
##  7 1996-07-01 00:00:00  0.0723
##  8 1996-08-01 00:00:00  0.0645
##  9 1996-09-01 00:00:00  0.0556
## 10 1996-10-01 00:00:00  0.0556
## # ... with 147 more rows
```

```r
# Let's see if time gives any bias #
qplot(c_time, data = time_c, bins = 10, color = I("black"))
```



```r
# there's a really really small, if any, bias due to time let's test it anyway #
# let's add it to our model #
predicted_ratings <- test_set %>%
  left_join(genres_c, by='genres') %>%
  left_join(user_c, by='userId') %>%
  left_join(title_c, by='title') %>%
  mutate(date = round_date(as_datetime(timestamp), unit = "month")) %>%
  left_join(time_c, by='date') %>%
  mutate(pred = mu + c_g + c_u + c_t + c_time) %>%
  .$pred
model_5_rmse <- RMSE(predicted_ratings, test_set$rating)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Genres + User + title + time month Model",
                                     RMSE = model_5_rmse ))

rmse_results
```

```
## # A tibble: 5 x 2
##   method                                      RMSE
##   <chr>                                      <dbl>
## 1 Just the mean model                        1.06
## 2 Genres Model                               1.02
## 3 Genres + User Model                          0.942
## 4 Genres + User + title Model                  0.872
## 5 Genres + User + title + time month Model 0.872
```

```
# time model using month does not add anything ##
```

Adding time did not do any good to our model, not increasing the RMSE at all. Now let's check timestamp using the week unit

```
# let's try only time using week on original data to see if there's any bias
time_week_only <- train_set %>%
  mutate(date = round_date(as_datetime(timestamp), unit = "week")) %>%
  group_by(date) %>%
  summarize(only_time = mean(rating - mu))
time_week_only
```

```
## # A tibble: 671 x 2
##    date                only_time
##    <dttm>                  <dbl>
##  1 1995-01-08 00:00:00   -0.513
##  2 1996-01-28 00:00:00    0.760
##  3 1996-02-04 00:00:00    0.0219
##  4 1996-02-11 00:00:00    0.354
##  5 1996-02-18 00:00:00    0.757
##  6 1996-02-25 00:00:00   -0.272
##  7 1996-03-03 00:00:00    0.105
##  8 1996-03-10 00:00:00    0.582
##  9 1996-03-17 00:00:00    0.202
## 10 1996-03-24 00:00:00    0.524
## # ... with 661 more rows
```

```
predicted_ratings <- test_set %>%
  mutate(date = round_date(as_datetime(timestamp), unit = "week")) %>%
  left_join(time_week_only, by= 'date') %>%
  mutate(pred = mu + only_time) %>%
  .$pred
model_6_rmse <- RMSE(predicted_ratings, test_set$rating)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="time_week_only_model",
                                     RMSE = model_6_rmse ))
# Let's see how we did #
rmse_results
```

```
## # A tibble: 6 x 2
##   method                                      RMSE
##   <chr>                                      <dbl>
## 1 Just the mean model                         1.06
## 2 Genres Model                                1.02
## 3 Genres + User Model                         0.942
## 4 Genres + User + title Model                 0.872
## 5 Genres + User + title + time month Model 0.872
## 6 time_week_only_model                        1.06
```

```
# nothing good happened, really bad results. We will stick with model number 4 #
```

The best RMSE comes from model number 4. Let's try to create the same model from the final edx set, testing it to the final validation set. We know our model (trained on our train set) works in our test set, but will it work if we generate it on the edx set and test it on the validation set?

```
mu_edx <- mean(edx$rating)
genres_c_edx <- edx %>%
  group_by(genres) %>%
  summarize(c_g_edx = mean(rating - mu_edx))

user_c_edx <- edx %>%
  left_join(genres_c_edx, by='genres') %>%
  group_by(userId) %>%
  summarize(c_u_edx = mean(rating - mu_edx - c_g_edx))

title_c_edx <- edx %>%
  left_join(genres_c_edx, by='genres') %>%
  left_join(user_c_edx, by='userId')%>%
  group_by(title) %>%
  summarize(c_t_edx = mean(rating - mu_edx - c_g_edx - c_u_edx))

predicted_ratings_temp <- validation %>%
  left_join(genres_c_edx, by='genres') %>%
  left_join(user_c_edx, by='userId') %>%
  left_join(title_c_edx, by='title') %>%
  mutate(pred = mu_edx + c_g_edx + c_u_edx + c_t_edx) %>%
  .$pred
RMSE_final_model <- RMSE(predicted_ratings_temp, validation$rating)
RMSE_final_model
```

```
## [1] 0.8712429
```

```
## yes!!! we created a simple model that seems to work#
## cool! ##
# our final model RMSE is more or less 0.871, even better than expected, we did pretty good!!
```

We finally have established a model which predicts ratings based on "Genre, User and Title effects".

CONCLUSIONS The model we generated gives a final RMSE of 0.8712 which is much better than our starting RMSE of 1.06, which was based on tha mean of ratings from the train set. Of course implementations could be performed, i tried implementing using non linear models but they all failed. Regularization could be the next step in order to keep into account the number of ratings per movie for example.