null

Introduction:

Read the data

library(tidyverse)

library(caret)

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(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)  
test\_index <- createDataPartition(y = movielens$rating, times = 1, p = 0.1, list = FALSE)  
edx <- movielens[-test\_index,]  
temp <- movielens[test\_index,]  
  
# 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")

DATA EXPLORATION

summary(edx)

## userId movieId rating timestamp   
## Min. : 1 Min. : 1 Min. :0.500 Min. :7.897e+08   
## 1st Qu.:18122 1st Qu.: 648 1st Qu.:3.000 1st Qu.:9.468e+08   
## Median :35743 Median : 1834 Median :4.000 Median :1.035e+09   
## Mean :35869 Mean : 4120 Mean :3.512 Mean :1.033e+09   
## 3rd Qu.:53602 3rd Qu.: 3624 3rd Qu.:4.000 3rd Qu.:1.127e+09   
## Max. :71567 Max. :65133 Max. :5.000 Max. :1.231e+09   
## title genres   
## Length:9000047 Length:9000047   
## Class :character Class :character   
## Mode :character Mode :character   
##   
##   
##

#A new dataframe "explore\_ratings" is created which contains half star and whole star ratings from the edx set :   
  
group <- ifelse((edx$rating == 1 |edx$rating == 2 | edx$rating == 3 |   
 edx$rating == 4 | edx$rating == 5) ,  
 "whole\_star",   
 "half\_star")   
  
explore\_ratings <- data.frame(edx$rating, group)

# Plot the explore\_ratings dataframe via histogram   
  
ggplot(explore\_ratings, aes(x= edx.rating, fill = group)) +  
 geom\_histogram( binwidth = 0.2) +  
 scale\_x\_continuous(breaks=seq(0, 5, by= 0.5)) +  
 scale\_fill\_manual(values = c("half\_star"="purple", "whole\_star"="brown")) +  
 labs(x="rating", y="number of ratings", caption = "source data: edx set") +  
 ggtitle("histogram : number of ratings for each rating")

![](data:image/png;base64;base64,) Exploring ratings of the edx set , we notice the following facts:

1.The average user’s activity reveals that no user gives 0 as rating 2.The top 5 ratings from most to least are : 4, 3, 5, 3.5 and 2. 3.The histogram shows that the half star ratings are less common than whole star ratings.

Exploring the features “genres” and “title” of our edx set.

#bar chart of top\_title  
  
top\_title <- edx %>%  
 group\_by(title) %>%  
 summarize(count=n()) %>%  
 top\_n(20,count) %>%  
 arrange(desc(count))

## `summarise()` ungrouping output (override with `.groups` argument)

top\_title %>%   
 ggplot(aes(x=reorder(title, count), y=count)) +  
 geom\_bar(stat='identity', fill="blue") + coord\_flip(y=c(0, 40000)) +  
 labs(x="", y="Number of ratings") +  
 geom\_text(aes(label= count), hjust=-0.1, size=3) +  
 labs(title="Top 20 movies title based \n on number of ratings" , caption = "source data: edx set")

![](data:image/png;base64;base64,) The movies which have the highest number of ratings are in the top genres categories : movies like Pulp fiction (1994), Forrest Gump(1994) or Jurrasic Park(1993) which are in the top 5 of movie’s ratings number , are part of the Drama, Comedy or Action genres.

#Computing the average and standard error for each "genre" , plotting the effect of genre   
edx %>% group\_by(genres) %>%  
 summarize(n = n(), avg = mean(rating), se = sd(rating)/sqrt(n())) %>%  
 filter(n >= 100000) %>%   
 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 = 45, hjust = 1)) +  
 labs(title = "error bar plots by genres" , caption = "source data : edx set") +  
 theme(  
 panel.background = element\_rect(fill = "lightblue",  
 colour = "lightblue",  
 size = 0.5, linetype = "solid"),  
 panel.grid.major = element\_line(size = 0.5, linetype = 'solid',  
 colour = "white"),   
 panel.grid.minor = element\_line(size = 0.25, linetype = 'solid',  
 colour = "white")  
 )

## `summarise()` ungrouping output (override with `.groups` argument)

![](data:image/png;base64;base64,) We observe that the generated plot shows strong evidence of a genre effect .

edx %>%  
 summarize(n\_users = n\_distinct(userId),  
 n\_movies = n\_distinct(movieId))

Even if each row represents a rating given by one user to one movie, the number of uniques values for the userId is 69878 and for the movieId 10664 : Both usersId and movieId which are presented as integer should be presumably treat as factors for some analysis purposes. Also, this means that there are less movies provided for ratings than users that rated them.If we think in terms of a large matrix, with user on the rows and movies on the columns, a challenge we face is the sparsity of our matrix. This large matrix will contain many empty cells. More over, we face a curse of dimensionality problem .These issues should be treat in our further analysis.

# histogram of number of ratings by movieId  
  
edx %>%   
 count(movieId) %>%   
 ggplot(aes(n)) +   
 geom\_histogram( bins=30, color = "red") +  
 scale\_x\_log10() +   
 ggtitle("Movies") +  
 labs(subtitle ="number of ratings by movieId",   
 x="movieId" ,   
 y="number of ratings",   
 caption ="source data : edx set") +  
 theme(panel.border = element\_rect(colour="black", fill=NA))

![](data:image/png;base64;base64,)

edx %>%   
 count(userId) %>%   
 ggplot(aes(n)) +   
 geom\_histogram( bins=30, color = "red") +  
 scale\_x\_log10() +   
 ggtitle("Users") +  
 labs(subtitle ="number of ratings by UserId",   
 x="userId" ,   
 y="number of ratings") +  
 theme(panel.border = element\_rect(colour="black", fill=NA))

![](data:image/png;base64;base64,)

DATA PREPROCESSING Data typically needs to be preprocessed (e.g. cleansed, filtered, transformed) in order to be used by the machine learning techniques in the analysis step. 1.Data transformation: Building a rating matrix

#Using SparseMatrix function to get the rating matrix from Matrix package  
library(Matrix)

##   
## Attaching package: 'Matrix'

## The following objects are masked from 'package:tidyr':  
##   
## expand, pack, unpack

edx\_1 <- edx  
  
edx\_1$userId <- as.factor(edx\_1$userId)  
edx\_1$movieId <- as.factor(edx\_1$movieId)  
  
edx\_1$userId <- as.numeric(edx\_1$userId)  
edx\_1$movieId <- as.numeric(edx\_1$movieId)  
sparse\_ratings <- sparseMatrix(i = edx\_1$userId,  
 j = edx\_1$movieId ,  
 x = edx\_1$rating,   
 dims = c(length(unique(edx\_1$userId)),  
 length(unique(edx\_1$movieId))),   
 dimnames = list(paste("u", 1:length(unique(edx\_1$userId)), sep = ""),   
 paste("m", 1:length(unique(edx\_1$movieId)), sep = "")))  
  
  
# remove the copy created  
rm(edx\_1)  
  
#give a look on the first 10 users  
sparse\_ratings[1:10,1:10]

## 10 x 10 sparse Matrix of class "dgCMatrix"

## [[ suppressing 10 column names 'm1', 'm2', 'm3' ... ]]

##   
## u1 . . . . . . . . . .  
## u2 . . . . . . . . . .  
## u3 . . . . . . . . . .  
## u4 . . . . . . . . . .  
## u5 1 . . . . . 3 . . .  
## u6 . . . . . . . . . .  
## u7 . . . . . . . . . .  
## u8 . 2.5 . . 3 4 . . . .  
## u9 . . . . . . . . . .  
## u10 . . . . . . 3 . . .

class(sparse\_ratings)

## [1] "dgCMatrix"  
## attr(,"package")  
## [1] "Matrix"

#Convert rating matrix into a recommenderlab sparse matrix via recommenderlab package  
library(recommenderlab)

ratingMat <- new("realRatingMatrix", data = sparse\_ratings)  
ratingMat

## 69878 x 10664 rating matrix of class 'realRatingMatrix' with 9000047 ratings.

2. Relevant Data

We know that some users saw more movies than the others. So, instead of displaying some random users and movies, we should select the most relevant users and movies. Thus we visualize only the users who have seen many movies and the movies that have been seen by many users.To identify and select the most relevant users and movies, we follow these steps:

1.Determine the minimum number of movies per user. 2.Determine the minimum number of users per movie. 3.Select the users and movies matching these criteria.

min\_n\_movies <- quantile(rowCounts(ratingMat), 0.9)  
  
min\_n\_users <- quantile(colCounts(ratingMat), 0.9)  
  
  
ratings\_movies <- ratingMat[rowCounts(ratingMat) > min\_n\_movies,  
 colCounts(ratingMat) > min\_n\_users]

we can notice that now, we have a rating matrix of 6976 distinct users (rows) x 1067 distinct movies(columns) , with 2311476 ratings .

#before to proceed with regularization, i just remove the object copy of validation, "valid"  
rm(valid)

## Warning in rm(valid): object 'valid' not found

#e. regularization   
  
# remembering (5), $\lambda$ is a tuning parameter. We can use cross-validation to choose it  
  
  
lambdas <- seq(0, 10, 0.25)  
   
 rmses <- sapply(lambdas, function(l){  
   
 mu\_reg <- mean(edx$rating)  
   
 b\_i\_reg <- edx %>%   
 group\_by(movieId) %>%  
 summarize(b\_i\_reg = sum(rating - mu\_reg)/(n()+l))  
   
 b\_u\_reg <- edx %>%   
 left\_join(b\_i\_reg, by="movieId") %>%  
 group\_by(userId) %>%  
 summarize(b\_u\_reg = sum(rating - b\_i\_reg - mu\_reg)/(n()+l))  
   
 predicted\_ratings\_b\_i\_u <-   
 validation %>%   
 left\_join(b\_i\_reg, by = "movieId") %>%  
 left\_join(b\_u\_reg, by = "userId") %>%  
 mutate(pred = mu\_reg + b\_i\_reg + b\_u\_reg) %>%  
 .$pred  
   
 return(RMSE(validation$rating,predicted\_ratings\_b\_i\_u))  
 })

qplot(lambdas, rmses)

![](data:image/png;base64;base64,)

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

## [1] 5.5

#valid\_set  
mu <- mean(edx$rating)  
b\_i\_reg <- edx %>%   
 group\_by(movieId) %>%  
 summarize(b\_i = sum(rating - mu)/(n()+lambda))

## `summarise()` ungrouping output (override with `.groups` argument)

b\_u\_reg <- edx %>%   
 left\_join(b\_i\_reg, by="movieId") %>%  
 group\_by(userId) %>%  
 summarize(b\_u = sum(rating - b\_i - mu)/(n()+lambda))

## `summarise()` ungrouping output (override with `.groups` argument)

predicted\_ratings\_6 <-   
 validation %>%   
 left\_join(b\_i\_reg, by = "movieId") %>%  
 left\_join(b\_u\_reg, by = "userId") %>%  
 mutate(pred = mu + b\_i + b\_u) %>%  
 pull(pred)  
View(predicted\_ratings\_6)  
model\_6\_rmse <- RMSE(predicted\_ratings\_6, validation$rating) # 0.864818

Methods and Analysis Recommender Engines

# a. POPULAR , UBCF and IBCF algorithms of the recommenderlab package  
library(recommenderlab)  
model\_pop <- Recommender(ratings\_movies, method = "POPULAR",   
 param=list(normalize = "center"))  
  
#prediction example on the first 10 users  
pred\_pop <- predict(model\_pop, ratings\_movies[1:10], type="ratings")  
as(pred\_pop, "matrix")[,1:10]

## m1 m2 m3 m5 m6 m7 m9 m10  
## u8 3.845709 NA 2.908521 NA NA 3.091285 2.524774 3.315059  
## u17 NA NA 3.012617 NA 3.860392 NA 2.628869 NA  
## u28 NA NA NA NA 3.167586 2.502576 NA 2.726350  
## u30 NA NA NA 2.800736 NA 3.118312 2.551801 NA  
## u43 4.693015 3.793565 3.755826 3.621014 4.603602 NA 3.372079 4.162365  
## u48 NA NA NA 3.505851 4.488439 3.823428 3.256917 NA  
## u57 NA NA 2.646955 2.512143 3.494730 2.829720 2.263208 3.053493  
## u70 4.426355 3.526905 3.489166 3.354354 4.336941 3.671931 3.105419 3.895704  
## u88 NA 3.040854 3.003116 2.868303 NA 3.185880 2.619368 3.409654  
## u103 NA 2.819650 2.781912 2.647099 NA 2.964676 2.398164 3.188450  
## m11 m14  
## u8 3.452896 3.411770  
## u17 NA 3.515866  
## u28 2.864186 2.823060  
## u30 NA 3.438797  
## u43 NA 4.259075  
## u48 4.185039 4.143913  
## u57 NA 3.150204  
## u70 4.033541 3.992415  
## u88 3.547490 3.506365  
## u103 3.326286 NA

#Calculation of rmse for popular method   
e <- evaluationScheme(ratings\_movies, method="split", train=0.7, given=-5)  
#5 ratings of 30% of users are excluded for testing  
  
model\_pop <- Recommender(getData(e, "train"), "POPULAR")  
  
prediction\_pop <- predict(model\_pop, getData(e, "known"), type="ratings")  
  
rmse\_popular <- calcPredictionAccuracy(prediction\_pop, getData(e, "unknown"))[1]  
rmse\_popular

## RMSE   
## 0.8470042

#Estimating rmse for UBCF using Cosine similarity and selected n as 50 based on cross-validation  
set.seed(1)  
model <- Recommender(getData(e, "train"), method = "UBCF",   
 param=list(normalize = "center", method="Cosine", nn=50))  
  
prediction <- predict(model, getData(e, "known"), type="ratings")  
  
rmse\_ubcf <- calcPredictionAccuracy(prediction, getData(e, "unknown"))[1]  
rmse\_ubcf

## RMSE   
## 0.8320826

#Estimating rmse for IBCF using Cosine similarity and selected n as 350 based on cross-validation  
set.seed(1)  
  
model\_ibcf <- Recommender(getData(e, "train"), method = "IBCF",   
 param=list(normalize = "center", method="Cosine", k=350))  
  
prediction\_ibcf <- predict(model\_ibcf, getData(e, "known"), type="ratings")  
  
rmse\_ibcf <- calcPredictionAccuracy(prediction\_ibcf, getData(e, "unknown"))[1]  
rmse\_ibcf

## RMSE   
## 0.9569868

#summarize all the rmse for recommender algorithms  
library(kableExtra)

rmse\_results <- data.frame(methods=c("Regularized Movie + User Effect Model","Recommender Popular Model" , "Recommender UBCF" ,"Recommender IBCF"),rmse = c(model\_6\_rmse, rmse\_popular,rmse\_ubcf,rmse\_ibcf))  
  
kable(rmse\_results) %>%  
 kable\_styling(bootstrap\_options = "striped" , full\_width = F , position = "center") %>%  
 kable\_styling(bootstrap\_options = "bordered", full\_width = F , position ="center") %>%  
 column\_spec(1,bold = T ) %>%  
 column\_spec(2,bold = T ,color = "white" , background ="#D7261E")

|  |  |
| --- | --- |
| methods | rmse |
| Regularized Movie + User Effect Model | 0.8649855 |
| Recommender Popular Model | 0.8470042 |
| Recommender UBCF | 0.8320826 |
| Recommender IBCF | 0.9569868 |