Homework 5

Problem 1:

a) Load the data into R

There are **807 songs** in Songs.csv, **2421 users** in Users.csv and the range of ratings goes from **1** to **3.43**. After setting the seed (set.seed(345)), we split the dataset with the dimensions:

- Training set with 84% of observations (Train)
- Validation set A for tuning the collaborative filtering model with 4% of observations (Val1)
- Validation set B to be used for blending with 4% of observations (Val2)
- Test set with 8% of observation (Test)

b) Let's create the ratings matrix

$$X_{i,j} = \alpha_i + \beta_j + \epsilon_{i,j}$$

i) For this model:

Number of parameters: 2421 + 807 = 3228

Number of observations: 2421 * 807 = 243103

ii) We use the **biscale** standardize a matrix to have optionally row means zero and variances one, and/or column means zero and variances one.

After removing the user affinity for rating songs highly (or lowly), we use **mutate** to join alpha and beta respectively to their dataset users and songs, then we use **inner_join** to join them by userID and songID to the test set. We sum up the two new columns then we have Xij.

Three most popular songs:

We simply look for the songs with the highest Beta since it's the rating without the bias of the user $\beta_j = X_{i,j} - \alpha_i$

Rank	SongID	Song Name	Artist Name	Beta
1	54	You're The One	Dwight Yoakam	1.71
2	26	Undo	Bjork	1.69
3	439	Secrets	One Republic	1.64

Here is the code

```r
```{r}
users <- users %>% mutate(alpha = alpha)%>% arrange(desc(alpha))
songs <- songs %>% mutate(beta = beta) %>% arrange(desc(beta))
test <- inner_join(x=test,y=users,by="userID")
test <- inner_join(x=test,y=songs[,c("songID","beta")],by="songID
")
test <- test %>% mutate(X = alpha + beta) %>% arrange(desc(X))
```
```

iii) The three users that are most enthused about songs after removing the bias due to the effect of the popularity of songs are:

Rank	usersID	alpha
1	1540	0.59
2	838	0.49
3	1569	0.47

iv) Performances on the test set:

We will use the metrics OSR, RMSE and MAE to assess the performances of the model on the test set

	Collaborative Filtering
OSR	$\mathbf{0.28}$
RMSE	$\mathbf{0.098}$
MAE	$\mathbf{0.0748}$

c) Let's consider the following model

$$X_{i,j} = Z_{i,j} + \alpha_i + \beta_j + \epsilon_{i,j} \ with \ Z_{i,j} = \sum_{t=1}^{k} w_{i,j} * S$$

i) Number of parameters

$$N = (2421 + 807) * k + (2421 + 807) = 3228(k+1)$$

We will train the model on the same training set, thus we still have 243103 observations.

ii) Number of archetypes

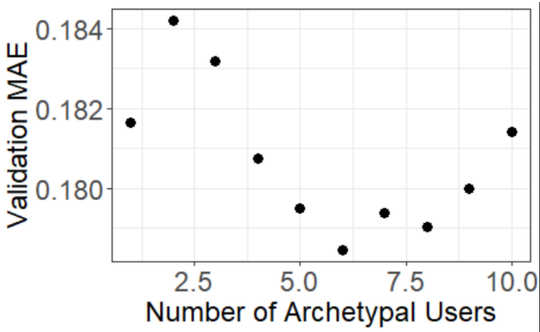
The performances would be measured by the metrics

$$\mathrm{MAE} = \frac{1}{N} \sum_{(i,j) \in \mathrm{OBS}} |X_{i,j} - Z_{i,j}| \qquad \mathrm{RMSE} = \sqrt{\frac{1}{N} \sum_{(i,j) \in \mathrm{OBS}} (X_{i,j} - Z_{i,j})^2}$$

Remark: Here we normalize MAE and RMSE by the scale of the ratings which is 3.43 – 1 = 2.43

We will find the number of archetypes that minimizes the MAE

We choose k=6



iii) Final collaborative model

	CF	CF (k=6)
OSR	$\mathbf{0.28}$	$\mathbf{0.27}$
RMSE	$\mathbf{0.098}$	$\mathbf{0.097}$
MAE	$\mathbf{0.0748}$	$\mathbf{0.073}$

Comments:

We see that we sligthly decrease the MAE while the coefficient of determination sligthly decrease too.

d) Add others features associated with songs

i) In this part, we will fit Random Forest and a Linear Regression model based on the independent variables: "genre" and "year". The goal is to build a final ensemble model that will also catch the specificities of each song for the predicition.

Remark: Before trainning our algorithm we set those two independent variables as factor.

- Random Forest: Mtry = 1; num.trees = 500
- Linear Regression:

	Linear Regression	Random Forest
OSR	$\mathbf{0.031}$	$\mathbf{0.043}$
RMSE	$\mathbf{0.114}$	$\mathbf{0.113}$
MAE	$\mathbf{0.093}$	$\mathbf{0.092}$

ii) We use the validation set B to perfom blending of the collaborative filtering model, Linear Regression and Random Forest. Here are the results

$$Blended\ Model = 0.74 * CF + 0.03 * LR + 0.22 * RF$$

	CF (k=6)	Blending	Increase
OSR	$\mathbf{0.27}$	$\mathbf{0.304}$	12.6%
RMSE	$\mathbf{0.097}$	$\mathbf{0.096}$	-1%
MAE	$\mathbf{0.073}$	$\mathbf{0.0744}$	-1.2%

Interpretation:

We observe that the MAE is almost the same despite a small increase of the OSR. Thus, blending the collaborative filtering model with other model doesn't add a lot of predictive power on top of the collaborative model. However, we can imagine tuning the parameters (features selection for Linear Regression and cross validation for Random Forest) to sharp our added models.

```{r}
library(softImpute)
library(randomForest)
library(ranger)
library(dplyr)
library(tidyverse)
library(reshape2)
```

```{r}
OSR2 <- function(predictions, train, test) {
  SSE <- sum((test - predictions)^2)
  SST <- sum((test - mean(train))^2)
  r2 <- 1 - SSE/SST
  return(r2)
}
```

```{r}
set.seed(345)

ratings = read.csv("MusicRatings.csv")
songs = read.csv("Songs.csv")
users = read.csv("Users.csv")

print(str_c("Number of users is ",nrow(users)))
print(str_c("Number of songs is ",nrow(songs)))
print(str_c("Range of the ratings is ",range(ratings$rating)[1]," and ", range(ratings$rating)[2]))
N = range(ratings$rating)[2] - range(ratings$rating)[1]
```

```{r}
set.seed(345)

train.ids <- sample(nrow(ratings), 0.92*nrow(ratings))
train <- ratings[train.ids,]
test <- ratings[-train.ids,]

# split training into real training and validation set
val1.ids <- sample(nrow(train), (4/92)*nrow(train))
val1 <- train[val1.ids,]
train <- train[-val1.ids,]

val2.ids <- sample(nrow(train), (4/88)*nrow(train))
val2 <- train[val2.ids,]
train <- train[-val2.ids,]
train <- inner_join(x=train,y=songs[,c("songID","genre","year")],by="songID")
train$genre = as.factor(train$genre)
train$year = as.factor(train$year)

val2 <- inner_join(x=val2,y=songs[,c("songID","genre","year")],by="songID")
val2$genre = as.factor(val2$genre)
val2$year = as.factor(val2$year)
```

```{r}
mat.train <- Incomplete(train$userID, train$songID, train$rating)

"number of parameters = 2421 * 807 = 1953747"
"number of observations = 2421"
```

```
```{r}
# Instruction of biscale()
set.seed(345)
mat.train.centered <- biScale(mat.train, maxit = 1000, row.scale = FALSE, col.scale = FALSE)
# mat.train.centered is X_ij - alpha_i - beta_j
alpha <- attr(mat.train.centered, "biScale:row")$center
beta <- attr(mat.train.centered, "biScale:column")$center
#center take the mean of the column

```

```{r}
users <- users %>% mutate(alpha = alpha)%>% arrange(desc(alpha))
songs <- songs %>% mutate(beta = beta) %>% arrange(desc(beta))
test <- inner_join(x=test,y=users,by="userID")
test <- inner_join(x=test,y=songs[,c("songID","beta","genre","year")],by="songID")
test <- test %>% mutate(X = alpha + beta) %>% arrange(desc(X))
test$genre = as.factor(test$genre)
test$year = as.factor(test$year)
```

```{r}
MAE = mean(abs(test$X - test$rating))/N
RMSE = sqrt(mean((test$X - test$rating)^2))/N
OSR = OSR2(test$X, train$rating, test$rating)
print(str_c("OSR is ",OSR, " RMSE is ",RMSE, " MAE is ",MAE))
```
```

```
```{r}
set.seed(345)
mae.vals = rep(NA, 10)
for (rnk in seq_len(10)) {
  print(str_c("Trying rank.max = ", rnk))
  mod <- softImpute(mat.train, rank.max = rnk, lambda = 0, maxit = 1000)
  #Impute missing values for a matrix via norm regularization
  preds <- impute(mod, val1$userID, val1$songID)
  #impute make prediction from an svd object
  mae.vals[rnk] <- mean(abs(preds - val1$rating))
}

mae.val.df <- data.frame(rnk = seq_len(10), mae = mae.vals)
ggplot(mae.val.df, aes(x = rnk, y = mae)) + geom_point(size = 3) +
  ylab("Validation MAE") + xlab("Number of Archetypal Users") +
  theme_bw() + theme(axis.title=element_text(size=18), axis.text=element_text(size=18))

```

```{r}
# choose k = 6
set.seed(345)
mod.final <- softImpute(mat.train, rank.max = 6, lambda = 0, maxit = 1000)
#Impute missing values for a matrix via norm regularization
preds <- impute(mod.final, test$userID, test$songID)
#impute make prediction from an svd object
MAE_cf = mean(abs(preds - test$rating))/N
RMSE_cf = sqrt(mean((preds - test$rating)^2))/N
OSR_cf = OSR2(preds, train$rating, test$rating)

print(str_c("OSR is ",OSR_cf, " RMSE is ",RMSE_cf, " MAE is ",MAE_cf))
```
```

```{r}
#Linear regression
|
lin.mod <- lm(rating ~ genre + year, data = train)
summary(lin.mod)

preds.lm <- predict(lin.mod, newdata = test)
MAE_lr = mean(abs(preds.lm - test$rating))/N
RMSE_lr = sqrt(mean((preds.lm - test$rating)^2))/N
OSR_lr = OSR2(preds.lm, train$rating, test$rating)

print(str_c("OSR is ",OSR_lr, " RMSE is ",RMSE_lr, " MAE is ",MAE_lr))
```

```{r}
#Random FOrest
set.seed(345)
rf.mod <- ranger(rating ~ genre + year,
                 data = train,
                 mtry = 2,
                 num.trees = 500,
                 verbose = TRUE)

preds.rf <- predict(rf.mod, data = test)
preds.rf <- preds.rf$predictions
MAE_rf = mean(abs(preds.rf - test$rating))/N
RMSE_rf = sqrt(mean((preds.rf - test$rating)^2))/N
OSR_rf = OSR2(preds.rf, train$rating, test$rating)
print(str_c("OSR is ",OSR_rf, " RMSE is ",RMSE_rf, " MAE is ",MAE_rf))
```

```{r}
# Blending

val.preds.cf <- impute(mod.final, val2$userID, val2$songID)
val.preds.lm <- predict(lin.mod, newdata = val2)
val.preds.rf <- predict(rf.mod, data = val2)$predictions

# Build validation set data frame
val.blending_df = data.frame(rating = val2$rating, cf_preds = val.preds.cf,
                             lm_preds = val.preds.lm, rf_preds = val.preds.rf)

# Train blended model
blend.mod = lm(rating ~ . -1, data = val.blending_df)
summary(blend.mod)

# Get predictions on test set
test.preds.cf <- impute(mod.final, test$userID, test$songID)
test.preds.lm <- predict(lin.mod, newdata = test)
test.preds.rf <- predict(rf.mod, data = test)$predictions

test.blending_df = data.frame(rating = test$rating, cf_preds = test.preds.cf,
                              lm_preds = test.preds.lm, rf_preds = test.preds.rf)

test.preds.blend <- predict(blend.mod, newdata = test.blending_df)

MAE_bl = mean(abs(test.preds.blend - test$rating))/N
RMSE_bl = sqrt(mean((test.preds.blend - test$rating)^2))/N
OSR_bl = OSR2(test.preds.blend, train$rating, test$rating)
print(str_c("OSR is ",OSR_bl, " RMSE is ",RMSE_bl, " MAE is ",MAE_bl))
```