# Project4 - group3

Group 3

In this project, we are going to explore matrix factorization methods for recommender system. The goal is to match consumers with most appropriate products. Matrix factorization methods characterize both items and users by vectors of factors inferred from item rating patterns. High correspondence between item and user factors leads to a recommendation. Matrix factorization generally has 3 parts:

- factorization algorithm
- regularization
- postpocessing

It is highly recommended to read this review paper.

### Step 1: Load Data and Train-test Split

```r
#installed.packages("remotes")
#remotes::install_github("TimothyKBook/krr")
#installed.packages("krr")
library(krr)
library(dplyr)
library(tidyr)
library(ggplot2)
library(caret)
#installed.packages("remotes")
remotes::install_github("TimothyKBook/krr")
data <- read.csv("../data/ml-latest-small/ratings.csv")
set.seed(0)
test_idx <- sample(1:nrow(data), round(nrow(data)/5, 0))
train_idx <- setdiff(1:nrow(data), test_idx)
data_train <- data[train_idx,]
data_test <- data[test_idx,]
write.csv(data_train,file="train_set.csv",row.names = F)
write.csv(data_test, file = "test_set.csv", row.names = F)
```

### Step 2: Matrix Factorization without Regularization

### Step 2.1: Implementing ALS Algorithm

Here we perform alternating least squares (ALS) to do matrix factorization. Our algorithm considers the case that there are new users and movies adding to the dataset we used to train. In other words, the dimension of matrix R, q, p is dynamic.

The dataset contains 9724 ratings (I) from 610 users (U).

Input of ALS function: f = 10, lambda = 5, max.iter=20, data, train=data_train, test=data_test.

Output of ALS function: p (User), q (Movie), r (ratings), train_RMSE, test_RMSE.

```r
U <- length(unique(data$userId))
I <- length(unique(data$movieId))
source("../lib/ALS.R")
```

### Step 2.2: Parameter Tuning and Postprocessing(KRR) for ALS

#### 2.2.1 Preparations for Kernel Ridge Regression Input:

After matrix factorization, postporcessing will be performed to improve accuracy.We need to transform the results from ALS to the form that we can put into kernel ridge regression. * First, we should split rating data for 610 users since we should do krr for different users. * Second, each column of q matrix from ALS represents a movie. We should etract certain column of q matrix corresponding to the movie(movieid) users rating and then combine them to build 610 different transformed new q matrices. * Finally, we need to normalize each row.

**2.2.2Tuning Parameters for Kernel Ridge Regression:** We set the Gaussian kernel: $K(x_i^T, x_j^T) = exp(2(x_i^T x_j - 1))$ as the paper said and we'll use k-folds cross validation to get the value of parameter $\lambda$.

#### 2.2.3 Train Kernel Ridge Regression and Get Prediction*

We use the best $\lambda$ as the tuned parameter and train 610 krr models for each user. And we can get the prediction matrix by using the model.

## ALS + KRR with 10 factors

```r
#read output and data
train_set <- read.csv("../data/train_set.csv")
test_set<- read.csv("../data/test_set.csv")
q<-read.csv("../output/A3_q_f10.csv",header= FALSE)
for_colnames <- read.csv("../output/A3_q_f10.csv")

## Prepare for kernel ridge regression input
train_split <- split(train_set,train_set$userId)
n <- length(train_split)
movie <- as.vector(unlist(c(q[1,])))

### Define function to normalize each row
norm.row <- function(m){
  return(m/sqrt(sum(m^2)))
}
q <- as.matrix(q[-1,])
new_q_split <- list()
for (k in 1:n){
  new <- c()
  for (i in 1:dim(train_split[[k]])[1]){
    new<-cbind(new,q[,which(movie == train_split[[k]]$movieId[i])])}
    new_q_split[[k]]<-new
}
q_t <- apply(q,2,norm.row)
q_t[which(is.na(q_t))] <- 0
x_split<-list()
for (k in 1: n){
  x_split[[k]]<-apply(new_q_split[[k]],2,norm.row)
}
data_split<-list()
for (k in 1:n){
  data_split[[k]] <- cbind(train_split[[k]]$rating,t(x_split[[k]]))
}

### save data_split
save(data_split,file = "../output/data_split1.RData")

source("../lib/cv.krr.R")
#find a best lambda
#lambdas <- c(0.7, 0.8, 0.9)
#rmse_tune <- data.frame(lambdas=c(0.7,0.8,0.9),rmse=rep(0,length(lambdas)))
#for (i in 1:length(lambdas)){
#  m <- lapply(data_split, cv.krr, 5, lambdas[i])
#  rmse_tune[i,2] <-  sum(unlist(m))
#}
#rmse_tune

# Train kernel ridge regression and get prediction
# From the result above, we could see that 0.7 is the best value for $\lambda$ with the lowest RSME. As
n <- length(data_split)
### Each user has his KRR model.
train_model_f10 <- vector(mode="list",length=n)
for(i in 1:n){
   train_model_f10[[i]] <- krr(x = data_split[[i]][,-1],
                               y = data_split[[i]][,1],
                               lambda = 0.7)
}

pred.rating_f10<-matrix(0,nrow=length(data_split),ncol=dim(q)[2])

for (i in 1:n){
  pred.rating_f10[i,] <- predict(train_model_f10[[i]],t(q_t))
}
```

```
### Rename colname of pred.rating_f10

#for_colnames <- read.csv("../output/A3_q_f10.csv")
pred.rating_f10_new <- as_tibble(pred.rating_f10[,-1] )
colnames(pred.rating_f10_new) <- colnames(for_colnames[,-1])

RMSE_new <- function(rating, est_rating){
  error = 0
  for (i in 1:nrow(rating)){
      error = error + (rating[i,]$rating - est_rating[rating[i, ]$userId, paste0('X', as.character(ratin
  }
  return(sqrt(error / nrow(rating)))
}

## Evaluation on train set
train_rmse_f10 <- RMSE_new(train_set, pred.rating_f10_new)  ## 0.63
## Evaluation on test set
test_rmse_f10 <- RMSE_new(test_set, pred.rating_f10_new)  ## 1.076627
```

```r
cat("The Train RMSE for ALS after doing KRR with factor = 10 is ", train_rmse_f10)
```

```
## The Train RMSE for ALS after doing KRR with factor = 10 is  0.6383762
```

```r
cat("The Test RMSE for ALS after doing KRR with factor = 10 is ", test_rmse_f10)
```

```
## The Test RMSE for ALS after doing KRR with factor = 10 is  1.076627
```

## ALS + KRR with 50 factors

```r
q_50 <- read.csv("../output/A3_q_f50.csv",header= FALSE)

movie <- as.vector(unlist(c(q_50[1,])))

n <- length(train_split)
q_50 <- as.matrix(q_50[-1,])
new_q_split <- list()
for (k in 1:n){
  new <- c()
for (i in 1:dim(train_split[[k]])[1]){
  new<-cbind(new,q_50[,which(movie == train_split[[k]]$movieId[i])])}
  new_q_split[[k]]<-new
}

q_t_50 <- apply(q_50,2,norm.row)
q_t_50[which(is.na(q_t_50))] <- 0
x_split<-list()

for (k in 1: n){
  x_split[[k]]<-apply(new_q_split[[k]],2,norm.row)
}

data_split_50<-list()

for (k in 1:n){
  data_split_50[[k]] <- cbind(train_split[[k]]$rating,t(x_split[[k]]))
}

#rmse_tune <- data.frame(lambdas = c(0.55,0.6,0.65),rmse=rep(0,length(lambdas)))

#for (i in 1:length(lambdas)){
#  m <- lapply(data_split_50,
#              cv.krr,
#              5,
#              lambdas[i])
#  rmse_tune[i,2] <-  sum(unlist(m))
#}
#rmse_tune

# The best lambda is 0.55.
n <- length(data_split_50)

train_model_f50 <- vector(mode="list",length=n)
```

```r
for(i in 1:n){
   train_model_f50[[i]] <- krr(x = data_split_50[[i]][,-1],
                              y = data_split_50[[i]][,1],
                              lambda = 0.55)
}
pred.rating_f50<-matrix(0,nrow=length(data_split_50),ncol=dim(q_50)[2])
for (i in 1:n){
  pred.rating_f50[i,] <- predict(train_model_f50[[i]],t(q_t_50))
}

for_colnames_50 <- read.csv("../output/A3_q_f50.csv")
pred.rating_f50_new <- as_tibble(pred.rating_f50[,-1] )
colnames(pred.rating_f50_new) <- colnames(for_colnames_50[,-1])


## Evaluation by RMSE
train_rmse_f50 <- RMSE_new(train_set, pred.rating_f50_new)  # 0.422295
test_rmse_f50 <- RMSE_new(test_set, pred.rating_f50_new)  # 1.00481
cat("The Train RMSE for ALS after doing KRR with factor = 50 is ", train_rmse_f50)
```

```
## The Train RMSE for ALS after doing KRR with factor = 50 is  0.422295
```

```r
cat("The Test RMSE for ALS after doing KRR with factor = 50 is ", test_rmse_f50)
```

```
## The Test RMSE for ALS after doing KRR with factor = 50 is  1.00481
```

## ALS + KRR with 100 factors

```r
q_100 <- read.csv("../output/A3_q_f100.csv",header= FALSE)

movie <- as.vector(unlist(c(q_100[1,])))

n <- length(train_split)
q_100 <- as.matrix(q_100[-1,])
new_q_split <- list()
for (k in 1:n){
  new <- c()
for (i in 1:dim(train_split[[k]])[1]){
  new<-cbind(new,q_100[,which(movie == train_split[[k]]$movieId[i])])}
  new_q_split[[k]]<-new
}

q_t_100 <- apply(q_100,2,norm.row)
q_t_100[which(is.na(q_t_100))] <- 0
x_split<-list()

for (k in 1: n){
  x_split[[k]]<-apply(new_q_split[[k]],2,norm.row)
}

data_split_100<-list()

for (k in 1:n){
  data_split_100[[k]] <- cbind(train_split[[k]]$rating,t(x_split[[k]]))
}

#rmse_tune <- data.frame(lambdas = c(0.55,0.6,0.65),rmse=rep(0,length(lambdas)))

#for (i in 1:length(lambdas)){
#  m <- lapply(data_split_100,
#              cv.krr,
#              5,
#              lambdas[i])
#  rmse_tune[i,2] <-  sum(unlist(m))
#}
#rmse_tune

# The best lambda is 0.5
n <- length(data_split_100)


### Each user has his KRR model.
train_model_f100 <- vector(mode="list",length=n)
for(i in 1:n){
   train_model_f100[[i]] <- krr(x = data_split_100[[i]][,-1],
                            y = data_split_100[[i]][,1],
                            lambda = 0.5)
}

pred.rating_f100<-matrix(0,nrow=length(data_split_100),ncol=dim(q_100)[2])
for (i in 1:n){
  pred.rating_f100[i,] <- predict(train_model_f100[[i]],t(q_t_100))
}

for_colnames_100 <- read.csv("../output/A3_q_f100.csv")
pred.rating_f100_new <- as_tibble(pred.rating_f100[,-1] )
colnames(pred.rating_f100_new) <- colnames(for_colnames_100[,-1])


## Evaluation by RMSE
train_rmse_f100 <- RMSE_new(train_set, pred.rating_f100_new)  # 0.3900224
test_rmse_f100 <- RMSE_new(test_set, pred.rating_f100_new)  #  0.9883943

cat("The Train RMSE for ALS after doing KRR with factor = 100 is ", train_rmse_f100)
```

```
## The Train RMSE for ALS after doing KRR with factor = 100 is  0.3900224
```

```
cat("The Test RMSE for ALS after doing KRR with factor = 100 is ", test_rmse_f100)
```

```
## The Test RMSE for ALS after doing KRR with factor = 100 is  0.9883943
```

###Step 3 Matrix Factorization with Regularization

### Step 3.1 Algorithm

Here we performed two regularizations: R1=Penalty of Magitudes and R2=Bias and Intercepts. At the beginning of matrix factorization, we utilize cross validation to determine the best lambda for use. Note:

after doing cross validation,we choose lambda=10 And within each matrix factorization(factor=10, 50, 100) we Output of ALS function: p (User), q (Movie), r (ratings), train_RMSE, test_RMSE

**Step 3.2 Parameter Tuning and Postprocessing**

## ALS + KRR + Regularization with 10 factors

```
# find a best lambda
# ALS_R1R2.cv(data, 5, 10, 0.01)
# training RMSE: 0.4602281  test RMSE: 1.531161
# ALS_R1R2.cv(data, 5, 10, 0.1)
# training RMSE: 0.4610532  test RMSE: 1.290163
# ALS_R1R2.cv(data, 5, 10, 1)
# training RMSE: 0.5325315  test RMSE: 1.040177
# ALS_R1R2.cv(data, 5, 10, 10)
# training RMSE: 0.6823324  test RMSE: 0.8672043
# So we choose lambda=10
source("../lib/ALS_R1R2.R")
f10 <- ALS_R1R2(10, 1, 2, data, data_train, data_test)
```

```
## iter: 1  training RMSE: 0.9537328    test RMSE: 1.048169
## iter: 2  training RMSE: 0.6678604    test RMSE: 0.9823808
```

```
write.csv(as.data.frame(f10[[2]]), file="A3_rq_f10.csv", row.names = F)
#read output and data
q_r_10 <- read.csv("../doc/A3_rq_f10.csv",header= FALSE)

## Prepare for kernel ridge regression input
movie <- as.vector(unlist(c(q_r_10[1,])))

n <- length(train_split)
q_r_10 <- as.matrix(q_r_10[-1,])
new_q_r_split <- list()
for (k in 1:n){
  new <- c()
  for (i in 1:dim(train_split[[k]])[1]){
    new<-cbind(new,q_r_10[,which(movie == train_split[[k]]$movieId[i])])}
  new_q_r_split[[k]]<-new
}

q_r_t_10 <- apply(q_r_10,2, norm.row)
q_r_t_10[which(is.na(q_r_t_10))] <- 0
x_r_split<-list()

for (k in 1: n){
  x_r_split[[k]]<-apply(new_q_r_split[[k]],2,norm.row)
}

data_split_r_10<-list()

for (k in 1:n){
  data_split_r_10[[k]] <- cbind(train_split[[k]]$rating,t(x_r_split[[k]]))
```

```r
}

n <- length(data_split_r_10)

### Each user has his KRR model.
train_model_r_f10 <- vector(mode="list",length=n)
for(i in 1:n){
   train_model_r_f10[[i]] <- krr(x = data_split_r_10[[i]][,-1],
                                 y = data_split_r_10[[i]][,1],
                                 lambda=10)
}

pred.rating_r_f10<-matrix(0,nrow=length(data_split_r_10),ncol=dim(q_r_10)[2])
for (i in 1:n){
  pred.rating_r_f10[i,] <- predict(train_model_r_f10[[i]],t(q_r_t_10))
}
```

```r
for_colnames_rq_10 <- read.csv("../doc/A3_rq_f10.csv")
pred.rating_r_f10_new <- as_tibble(pred.rating_r_f10)
colnames(pred.rating_r_f10_new) <- colnames(for_colnames_rq_10)
```

```r
## Evaluation
train_rmse_reg_f10 <- RMSE_new(train_set, pred.rating_r_f10_new)  # 1.232865
test_rmse_reg_f10 <-RMSE_new(test_set, pred.rating_r_f10_new)  # 1.552325
cat("The Train RMSE for ALS+Regularzation after doing KRR with factor = 10 is ", train_rmse_f10)
```

```
## The Train RMSE for ALS+Regularzation after doing KRR with factor = 10 is  0.6383762
```

```r
cat("The Test RMSE for ALS+Regularzation after doing KRR with factor = 10 is ", test_rmse_f10)
```

```
## The Test RMSE for ALS+Regularzation after doing KRR with factor = 10 is  1.076627
```

## ALS + KRR + Regularization with 50 factors

```r
# find a best lambda
# ALS_R1R2.cv(data, 5, 50, 0.01)
# training RMSE: 0.1676079  test RMSE: 1.833671
# ALS_R1R2.cv(data, 5, 50, 0.1)
# training RMSE: 0.07037745     test RMSE: 1.3485
# ALS_R1R2.cv(data, 5, 50, 1)
# training RMSE: 0.1686628  test RMSE: 1.192529
# ALS_R1R2.cv(data, 5, 50, 10)
# training RMSE: 0.5636543  test RMSE: 0.8632474
# So we choose lambda=1
source("../lib/ALS_R1R2.cv.R")
f50 <- ALS_R1R2(50, 1, 2, data, data_train, data_test)
```

```
## iter: 1  training RMSE: 0.7611159    test RMSE: 1.06238
## iter: 2  training RMSE: 0.3243659    test RMSE: 1.010643
```

```r
write.csv(as.data.frame(f50[[2]]), file="A3_rq_f50.csv", row.names = F)
#read output and data
q_r_50 <- read.csv("../doc/A3_rq_f50.csv",header= FALSE)
```

```r
## Prepare for kernel ridge regression input
movie <- as.vector(unlist(c(q_r_50[1,])))

n <- length(train_split)
q_r_10 <- as.matrix(q_r_50[-1,])
new_q_r_split <- list()
for (k in 1:n){
  new <- c()
  for (i in 1:dim(train_split[[k]])[1]){
    new<-cbind(new,q_r_50[,which(movie == train_split[[k]]$movieId[i])])}
  new_q_r_split[[k]]<-new
}

q_r_t_50 <- apply(q_r_50,2, norm.row)
q_r_t_50[which(is.na(q_r_t_50))] <- 0
x_r_split<-list()

for (k in 1: n){
  x_r_split[[k]]<-apply(new_q_r_split[[k]],2,norm.row)
}

data_split_r_50<-list()

for (k in 1:n){
  data_split_r_50[[k]] <- cbind(train_split[[k]]$rating,t(x_r_split[[k]]))
}

n <- length(data_split_r_50)

### Each user has his KRR model.
train_model_r_f50 <- vector(mode="list",length=n)
for(i in 1:n){
   train_model_r_f50[[i]] <- krr(x = data_split_r_50[[i]][,-1],
                                 y = data_split_r_50[[i]][,1],
                                 lambda=1)
}

pred.rating_r_f50<-matrix(0,nrow=length(data_split_r_50),ncol=dim(q_r_50)[2])
for (i in 1:n){
  pred.rating_r_f50[i,] <- predict(train_model_r_f50[[i]],t(q_r_t_50))
}

for_colnames_rq_50 <- read.csv("../doc/A3_rq_f50.csv")
pred.rating_r_f50_new <- as_tibble(pred.rating_r_f50)
colnames(pred.rating_r_f50_new) <- colnames(for_colnames_rq_50)

## Evaluation
train_rmse_reg_f50 <- RMSE_new(train_set, pred.rating_r_f50_new)  # 0.9371947
test_rmse_reg_f50 <- RMSE_new(test_set, pred.rating_r_f50_new)  # 0.9516052

cat("The Train RMSE for ALS+Regularzation after doing KRR with factor = 50 is ", train_rmse_f50)
```

```
## The Train RMSE for ALS+Regularzation after doing KRR with factor = 50 is  0.422295
```

```r
cat("The Test RMSE for ALS+Regularzation after doing KRR with factor = 50 is ", test_rmse_f50)
```

```
## The Test RMSE for ALS+Regularzation after doing KRR with factor = 50 is  1.00481
```

## ALS + KRR + Regularization with 100 factors

```r
# find a best lambda
# ALS_R1R2.cv(data, 5, 100, 0.01)
# training RMSE: 0.2915573  test RMSE: 1.931979
# ALS_R1R2.cv(data, 5, 100, 0.1)
# training RMSE: 0.3204951  test RMSE: 1.233925
# ALS_R1R2.cv(data, 5, 100, 1)
# training RMSE: 0.1306793  test RMSE: 1.017521
# ALS_R1R2.cv(data, 5, 100, 10)
# training RMSE: 0.5572743  test RMSE: 0.8606678
# So we choose lambda=1
source("../lib/ALS_R1R2.cv.R")
f100 <- ALS_R1R2(100, 1, 2, data, data_train, data_test)
```

```
## iter: 1  training RMSE: 0.6388634    test RMSE: 1.039601
## iter: 2  training RMSE: 0.2216702    test RMSE: 0.9712735
```

```r
write.csv(as.data.frame(f100[[2]]), file="A3_rq_f100.csv", row.names = F)
#read output and data
q_r_100 <- read.csv("../doc/A3_rq_f100.csv",header= FALSE)

## Prepare for kernel ridge regression input
movie <- as.vector(unlist(c(q_r_100[1,])))

n <- length(train_split)
q_r_100 <- as.matrix(q_r_100[-1,])
new_q_r_split <- list()
for (k in 1:n){
  new <- c()
  for (i in 1:dim(train_split[[k]])[1]){
    new<-cbind(new,q_r_100[,which(movie == train_split[[k]]$movieId[i])])}
  new_q_r_split[[k]]<-new
}

q_r_t_100 <- apply(q_r_100,2, norm.row)
q_r_t_100[which(is.na(q_r_t_100))] <- 0
x_r_split<-list()

for (k in 1: n){
  x_r_split[[k]]<-apply(new_q_r_split[[k]],2,norm.row)
}

data_split_r_100<-list()

for (k in 1:n){
  data_split_r_100[[k]] <- cbind(train_split[[k]]$rating,t(x_r_split[[k]]))
}
```

```r
n <- length(data_split_r_100)

### Each user has his KRR model.
train_model_r_f100 <- vector(mode="list",length=n)
for(i in 1:n){
   train_model_r_f100[[i]] <- krr(x = data_split_r_100[[i]][,-1],
                                  y = data_split_r_100[[i]][,1],
                                  lambda=1)
}

pred.rating_r_f100<-matrix(0,nrow=length(data_split_r_100),ncol=dim(q_r_100)[2])
for (i in 1:n){
  pred.rating_r_f100[i,] <- predict(train_model_r_f100[[i]],t(q_r_t_100))
}


for_colnames_rq_100 <- read.csv("../doc/A3_rq_f100.csv")
pred.rating_r_f100_new <- as_tibble(pred.rating_r_f100)
colnames(pred.rating_r_f100_new) <- colnames(for_colnames_rq_100)


## Evaluation
train_rmse_reg_f100 <- RMSE_new(train_set, pred.rating_r_f100_new)  # 0.4127746
test_rmse_reg_f100 <- RMSE_new(test_set, pred.rating_r_f100_new)  # 1.459609

cat("The Train RMSE for ALS+Regularzation after doing KRR with factor = 100 is ", train_rmse_f100)
```

```
## The Train RMSE for ALS+Regularzation after doing KRR with factor = 100 is  0.3900224
```

```r
cat("The Test RMSE for ALS+Regularzation after doing KRR with factor = 100 is ", test_rmse_f100)
```

```
## The Test RMSE for ALS+Regularzation after doing KRR with factor = 100 is  0.9883943
```

### Step 4 Evaluation

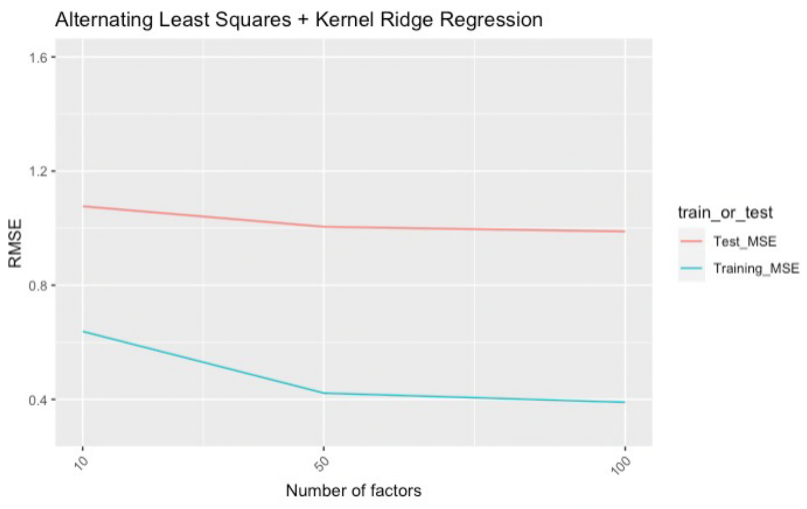
Visualization of training and testing RMSE by different dimension of factors and different model(ALS+KRR or ALS+Regularization+KRR)

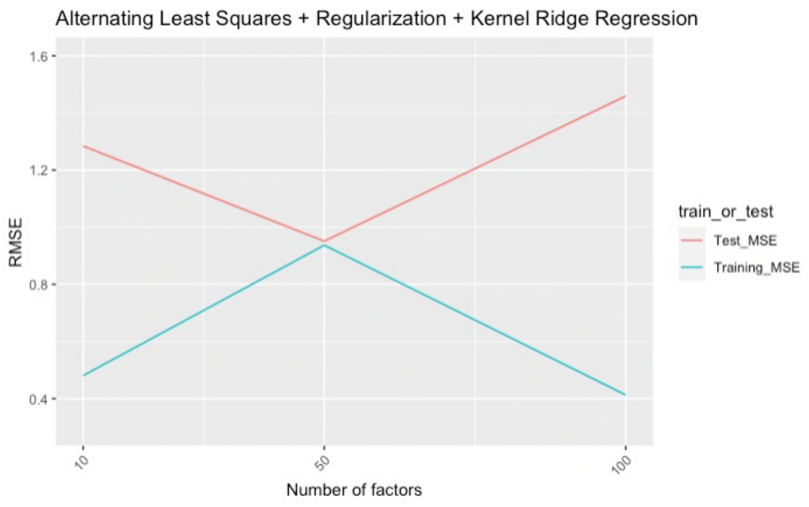
```r
train_rmse <- c(train_rmse_f10,train_rmse_f50,train_rmse_f100)
test_rmse <- c(test_rmse_f10,test_rmse_f50,test_rmse_f100)

train_rmse_reg <- c(train_rmse_reg_f10,train_rmse_reg_f50,train_rmse_reg_f100)
test_rmse_reg <- c(test_rmse_reg_f10,test_rmse_reg_f50,test_rmse_reg_f100)

rmse <- data.frame(epochs = c(10, 50, 100), Training_MSE = train_rmse, Test_MSE = test_rmse)%>%
  gather(key = train_or_test, value = RMSE, -epochs)

rmse_reg <- data.frame(epochs = c(10, 50, 100), Training_MSE = train_rmse_reg, Test_MSE = test_rmse_reg
  gather(key = train_or_test, value = RMSE, -epochs)

p1 <- ggplot(rmse, aes(x = epochs, y = RMSE, col = train_or_test)) +
  geom_line() +
  ggtitle("Alternating Least Squares + Kernel Ridge Regression ")+
  ylab("RMSE") +
  ylim(0.3,1.6)+
  scale_x_continuous("Number of factors", breaks=c(10, 50, 100))+
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

p2 <- ggplot(rmse_reg, aes(x = epochs, y = RMSE, col = train_or_test)) +
  geom_line() +
  ggtitle("Alternating Least Squares + Kernel Ridge + Regression ")+
  ylab("RMSE") +
  ylim(0.3,1.6)+
  scale_x_continuous("Number of factors", breaks=c(10, 50, 100))+
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

## Summary

From the RMSE of ALS with KRR, our result shows that the more latent factors we use, the lower the final RMSE is. From the RMSE of ALS+Regularzation with KRR, our plot shows that at the beginning,the more latent factors will reduce the test RMSE. However, if the latent factors become to large, the regularzations penalized the RMSE to aviod overfitting problem.

In conlcusion,our report shows that in general ALS with KRR have better performace than ALS+Regularzation with KRR. However, ALS with R1+R2 regularization has the lowest RMSE which is 0.9516052. The results are close so they may due to chances.It is hard to tell which method is superior to the other one.


Alternating Least Squares + Kernel Ridge Regression
RMSE
Number of factors
train_or_test
Test_MSE
Training_MSE


Alternating Least Squares + Regularization + Kernel Ridge Regression
RMSE
Number of factors
train_or_test
Test_MSE
Training_MSE