Project4 Group10

Step 1 Load Data and Train-test Split

```r
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(tidyr)
library(ggplot2)
library(purrr)
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,]
```

Step 2 Matrix Factorization

Step 2.1 load the function for A2

```r
source("../lib/Probalistic_ Matrix_Factorization_w_cv.R")
```

```
##  Attaching packages                                       tidyverse 1.3.0

##  tibble  2.1.3      stringr 1.4.0
##  readr   1.3.1      forcats 0.4.0

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

Step 2.2 pre Tunning for single A2 to shrink the possible parameters

We use pre-fine-tunning to shrink the number of possible parameters for our future model. And as we fixed the sigma, only parameter that would infect the model is D, sigma_U, sigma_V and lrate.

So we choose three candidate value for D, sigma_U, sigma_V and lrate.

```r
# tuned parameters
t_D <- c(5, 10, 15)
t_sigma_V <- c(.1, .5, 1)
t_sigma_U <- c(.1, .5, 1)
t_lrate <- c(.0001, .0005, .001)

# Do the first step of CV(on the A2) to select parameters that can be used
parameters <- expand.grid(D = t_D, sigma_V = t_sigma_V, sigma_U = t_sigma_U, lrate = t_lrate)
```

Do cross validation

```r
cv_train_rmse <- matrix(NA, dim(parameters)[1], 20)
cv_test_rmse <- matrix(NA, dim(parameters)[1], 20)

for(i in 1:dim(parameters)[1]){
  tmp_D <- parameters$D[i]
  tmp_sigma_V <- parameters$sigma_V[i]
  tmp_sigma_U <- parameters$sigma_U[i]
  tmp_lrate <- parameters$lrate[i]

  ret <- cv_A2.function(data, data.train, K = 5,
                       D = tmp_D, sigma_V = tmp_sigma_V, sigma_U = tmp_sigma_U, lrate = tmp_lrate)
  cat("--------------------------------------- No.",i,"Done --------------------------------------------
  cv_train_rmse[i,] <- ret$mean_train_rmse
  cv_test_rmse[i,] <- ret$mean_test_rmse
}
cv_A2_train_rmse <- cv_train_rmse
cv_A2_test_rmse <- cv_test_rmse
colnames(cv_A2_train_rmse) <- sapply(c(1:20), function(x) paste("train_",x,sep = ''))
colnames(cv_A2_test_rmse) <- sapply(c(1:20), function(x) paste("test_",x,sep = ''))

save(cv_A2_train_rmse, file = "../output/cv_A2_train.RData")
save(cv_A2_test_rmse, file = "../output/cv_A2_test.RData")
```

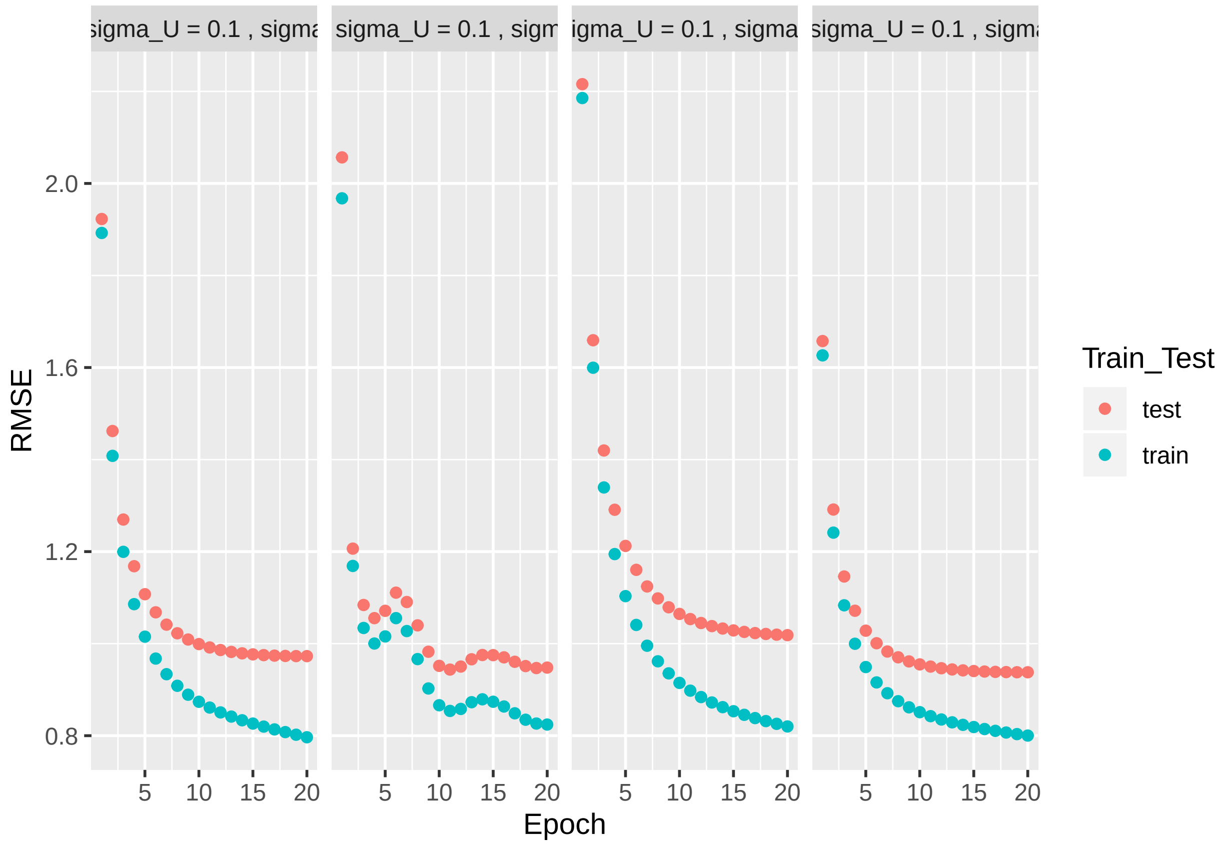
For each parameter, 2e select top two possible value which can make model to have the smallest mean(RMSE).

```r
load("../output/cv_A2_train.RData")
load("../output/cv_A2_test.RData")

cv.return <- parameters %>% cbind(cv_A2_train_rmse, cv_A2_test_rmse)

pre_ft_D <- (cv.return %>%
  group_by(D) %>%
  summarise(test_mean = mean(test_20)) %>%
  arrange(test_mean) %>%
  select(D) %>% as.matrix() %>% c())[1:2]

pre_ft_sigma_U <- (cv.return %>%
  group_by(sigma_U) %>%
  summarise(test_mean = mean(test_20)) %>%
  arrange(test_mean) %>%
  select(sigma_U) %>% as.matrix() %>% c())[1:2]
```

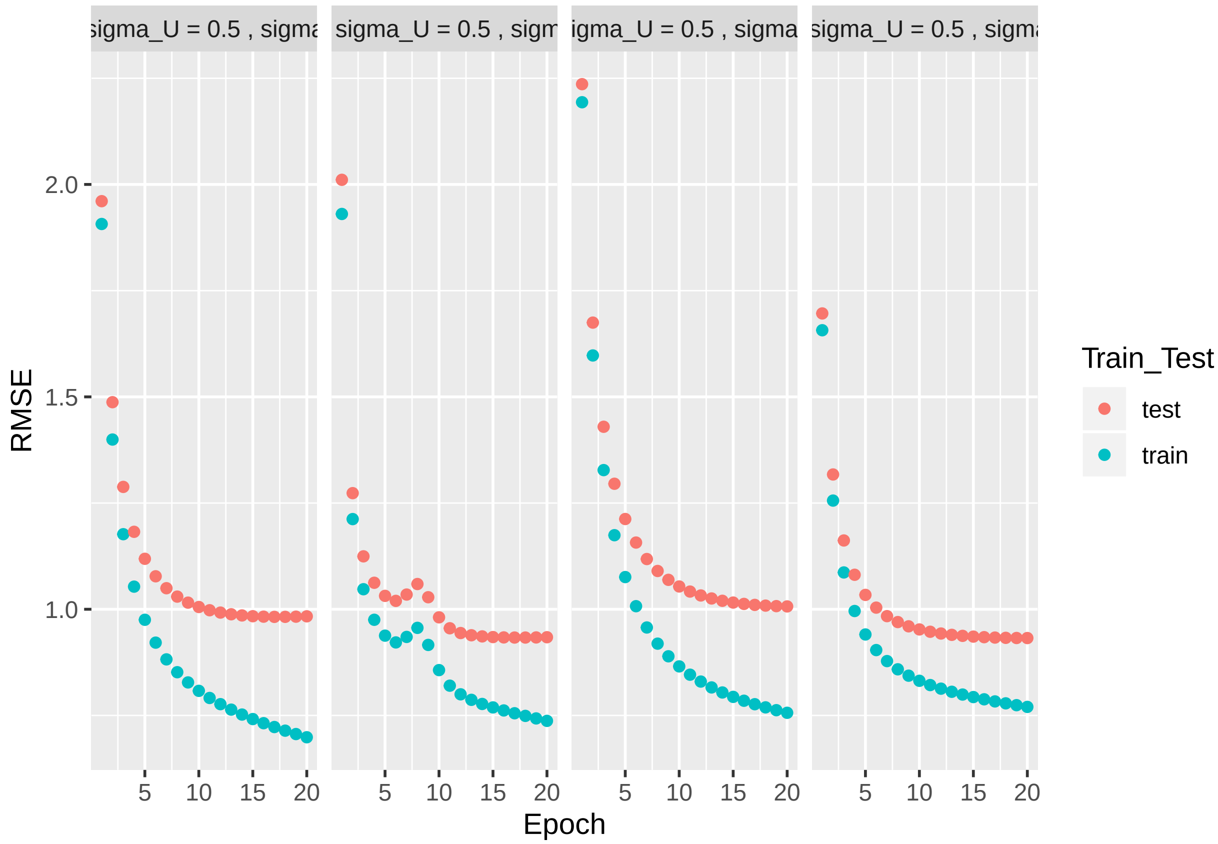
```r
pre_ft_sigma_V <- (cv.return %>%
  group_by(sigma_V) %>%
  summarise(test_mean = mean(test_20)) %>%
  arrange(test_mean) %>%
  select(sigma_V)%>% as.matrix() %>% c())[1:2]

pre_ft_lrate = (cv.return %>%
  group_by(lrate) %>%
  summarise(test_mean = mean(test_20)) %>%
  arrange(test_mean) %>%
  select(lrate)%>% as.matrix() %>% c())[1]
```

Then we draw the graph to see the convergency of these 8 model sets.

```r
cv.grap <- cv.return %>%
  filter(D %in% pre_ft_D,
         sigma_U %in% pre_ft_sigma_U,
         sigma_V %in% pre_ft_sigma_V,
         lrate == pre_ft_lrate) %>%
  mutate(param_set = paste("D =",D,", sigma_U =", sigma_U, ", sigma_V =", sigma_V)) %>%
  select(-c(D,sigma_V,sigma_U,lrate)) %>%
  pivot_longer(-c(param_set), names_to = "Train_Test", values_to = "RMSE") %>%
  separate(Train_Test, into = c("Train_Test", "Epoch"), sep = "_") %>%
  mutate(Epoch = as.integer(Epoch))


a <- unique(cv.grap$param_set)[1:4]
b <- unique(cv.grap$param_set)[5:8]

cv.grap %>%
  filter(param_set %in% a) %>%
  ggplot(aes(x = Epoch, y = RMSE, col = Train_Test)) + geom_point() + facet_grid(~param_set)
```




```r
cv.grap %>%
  filter(param_set %in% b) %>%
  ggplot(aes(x = Epoch, y = RMSE, col = Train_Test)) + geom_point() + facet_grid(~param_set)
```




```
## Promising value for D is 10 and 5.

## Promising value for sigma_U is 0.1 and 0.5.

## Promising value for sigma_V is 1 and 0.5.

## We fix the learning rate to be 5e-04.
```

Step 3 Postprocessing

Step 3.2 P3:Postprocessing with KNN

Step3.2.1 load the function for P2

```r
source("../lib/KNN.R")
```

Step 3.2.2 Tunning for A2+P2

During the former tunning, we have shrinked the parameters to 6 groups. In this part, we added an additional K to the combination, K also has two choices.

```r
#parameters
t_D <- c(5, 10)
t_sigma_V <- c(.1, .5)
t_sigma_U <- c(.5, 1)
t_K <- c(20, 30)

#build the data.frame to show all possible combination of parameters
par <- expand.grid(D = t_D, sigma_V = t_sigma_V, sigma_U = t_sigma_U,K = t_K)
```

Do cross validation

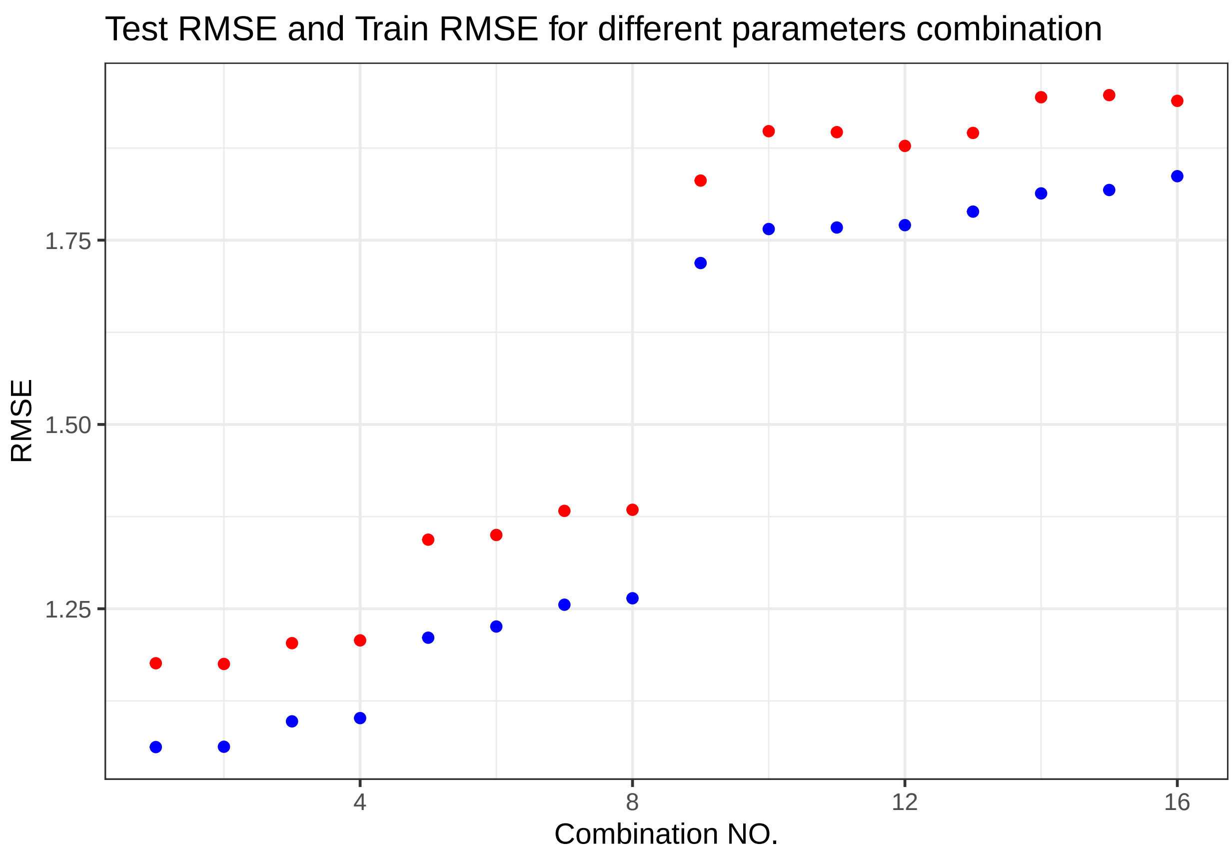
```r
cv_train_rmse <- c()
cv_test_rmse <- c()

for(i in dim(par)[1]){
  tmp_D <- par$D[i]
  tmp_sigma_V <- par$sigma_V[i]
  tmp_sigma_U <- par$sigma_U[i]
  tmp_k  <- par$K[i]

  ret <- cv_P2.function(data, data.train, K = 5, D = tmp_D,
                        sigma_V = tmp_sigma_V, sigma_U = tmp_sigma_U, k = tmp_k)

  cat("---------------------------------------- No.",i,"Done -----------------------------------------
  cv_train_rmse[i] <- ret$mean_train_rmse
  cv_test_rmse[i] <- ret$mean_test_rmse

}

cv_P2_train_rmse <- cv_train_rmse
cv_P2_test_rmse <- cv_test_rmse

rmse_P2<-cbind(par,cv_P2_train_rmse,cv_P2_test_rmse)

save(rmse_P2, file = "../output/RMSE_A2_P2.RData")
```

Then we use these result to choose the best model for A2+P2 model based on the train RMSE and test RMSE we have.

```r
load("../output/RMSE_A2_P2.RData")

best_rmse_P2<-RMSE_A2_P2%>%data.frame()%>%
  arrange(train_RMSE_A2_P2,test_RMSE_A2_P2)
head(best_rmse_P2)
```

```
##   D sigma_V sigma_U  K train_RMSE_A2_P2 test_RMSE_A2_P2
## 1 5     0.1     1.0 30         1.062353        1.176098
## 2 5     0.5     1.0 30         1.062805        1.175113
## 3 5     0.1     1.0 20         1.097260        1.203354
## 4 5     0.5     1.0 20         1.101682        1.207118
## 5 5     0.1     0.5 30         1.210724        1.343710
## 6 5     0.1     0.5 20         1.225879        1.350066
```

```r
g<-ggplot(best_rmse_P2)+
  geom_point(aes(x = 1:nrow(best_rmse_P2),y = test_RMSE_A2_P2),col = "red")+
  geom_point(aes(x = 1:nrow(best_rmse_P2),y = train_RMSE_A2_P2),col = "blue")+
  labs(x = "Combination NO.",
       y = "RMSE",
       title = "Test RMSE and Train RMSE for different parameters combination")+
  theme_bw()
g
```



Step 3.2.3 Output the model, the test RMSE and the rating matrix

The best group of parameters is $D = 5$, $\sigma_v = 0.5$, $\sigma_u = 1$, $K = 30$. Then we use these to Calculate the user-movie prediction matrix and see the RMSE on the train and test set.

```r
t0<-Sys.time()

r <- P2_KNN(data = data, data.train, data.test, K=30, D=5, sigma_V = 0.5, sigma_U = 1)

t1<-Sys.time()

ratings_A2_P2<-r$pred
test_RMSE_A2_P2 <- r$test_RMSE
train_RMSE_A2_P2 <- r$train_RMSE
##Run time for KNN model to build the rating matrix and get the RMSEs is about 7.615576 mins
t1-t0
```

```r
#part of the rating matrix
ratings_A2_P2[1:5,1:5]

save(train_RMSE_A2_P2,file = "../output/train_RMSE_A2_P2.RData")
save(test_RMSE_A2_P2,file = "../output/test_RMSE_A2_P2.RData")
save(ratings_A2_P2,file = "../output/ratings_A2_P2.RData")
```

Step 3.3 P3:Postprocessing with kernel ridge regression

Step 3.3.1 load the function for P3

```r
source("../lib/krr_cv.R")
```

Step 3.3.2 Tunning for A2+P3

During the former tunning, we have shrinked the parameters to 6 groups. In this part, we added two tunning parameters "λ" and "Kernel Type" to the combination, each one has two possible choices.

```r
#tunes parameters
t_D <- c(5, 10)
t_sigma_V <- c(.1, .5)
t_sigma_U <- c(.5, 1)
t_lambda <- c(.5,1)
t_kernel <- c("Gaussian","Linear")

#build the data.frame to show all possible combination of parameters
par <- expand.grid(D = t_D, sigma_V = t_sigma_V, sigma_U = t_sigma_U,lambda = t_lambda,kernel = t_kerne
```

Do cross validation

```r
cv_train_rmse <- c()
cv_test_rmse <- c()

for(i in dim(par)[1]){
  tmp_D <- par$D[i]
  tmp_sigma_V <- par$sigma_V[i]
  tmp_sigma_U <- par$sigma_U[i]
  tmp_lambda  <- par$lambda[i]
  tmp_kernel  <- par$kernel[i]

  ret <- cv_P3.function(data, data.train, K = 5,D = tmp_D, sigma_V = tmp_sigma_V, sigma_U = tmp_sigma_U
  cat("---------------------------------------- No.",i,"Done -----------------------------------------
  cv_train_rmse[i] <- ret$mean_train_rmse
  cv_test_rmse[i] <- ret$mean_test_rmse

}

cv_P3_train_rmse <- cv_train_rmse
cv_P3_test_rmse <- cv_test_rmse
```

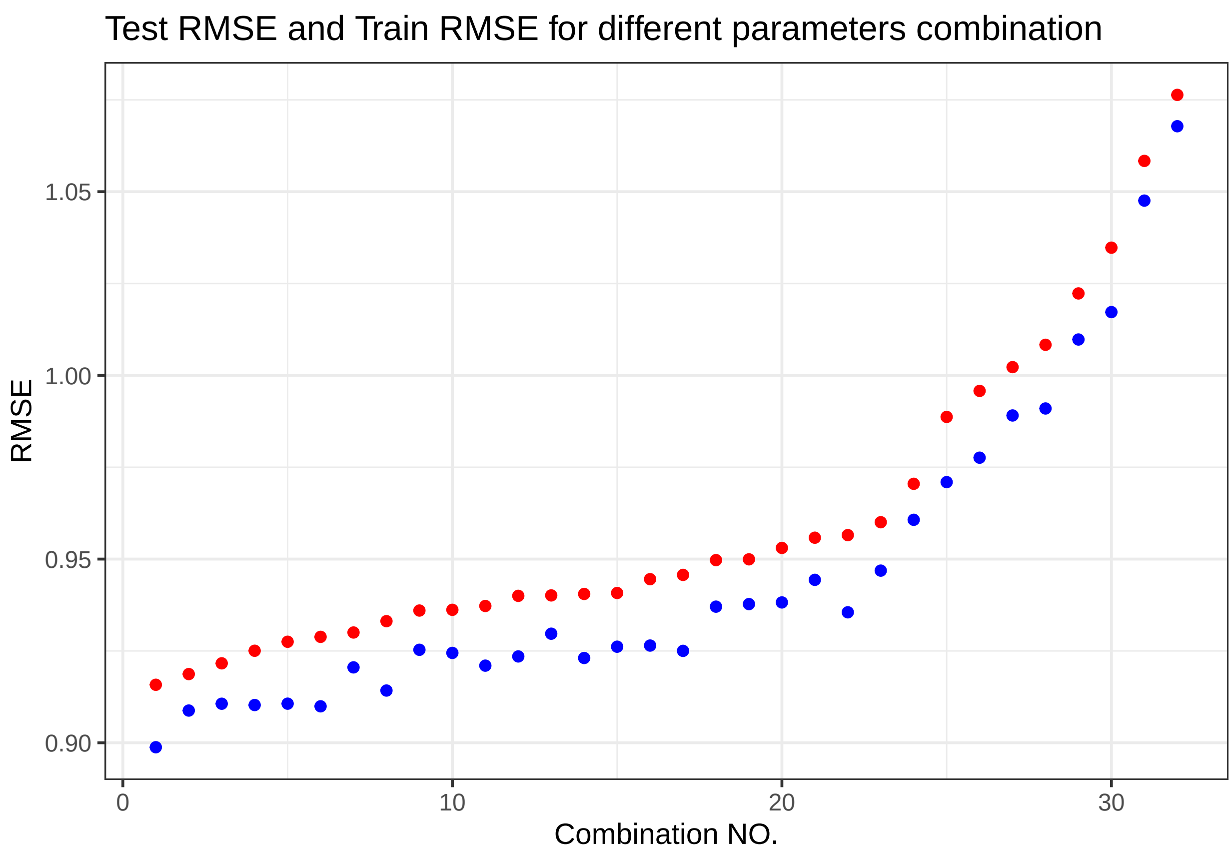
```r
rmse_P3<-par%>%
  mutate("train rmse" = cv_P3_train_rmse,
         "test rmse" = cv_P3_test_rmse)

save(rmse_P3, file = "../output/RMSE_A2_P3.RData")
```

Then we use these result to choose the best model for A2+P3 model based on the train RMSE and test RMSE we have.

```r
load("../output/RMSE_A2_P3.RData")

best_rmse_P3<-rmse_P3%>%
  arrange(`test rmse`,`train rmse`)
head(best_rmse_P3)
```

```
##   D sigma_V sigma_U lambda   kernel train rmse test rmse
## 1 5     0.1     0.5    1.0 Gaussian  0.8987834 0.9157871
## 2 5     0.1     1.0    0.5 Gaussian  0.9087701 0.9187002
## 3 5     0.1     0.5    0.5 Gaussian  0.9106254 0.9216217
## 4 5     0.1     1.0    1.0 Gaussian  0.9102728 0.9250427
## 5 5     0.1     1.0    1.0   Linear  0.9106466 0.9274937
## 6 5     0.1     0.5    1.0   Linear  0.9099221 0.9288295
```

```r
g<-ggplot(best_rmse_P3)+
  geom_point(aes(x = 1:nrow(best_rmse_P3),y = `test rmse`),col = "red")+
  geom_point(aes(x = 1:nrow(best_rmse_P3),y = `train rmse`),col = "blue")+
  labs(x = "Combination NO.",
       y = "RMSE",
       title = "Test RMSE and Train RMSE for different parameters combination")+
  theme_bw()
g
```



Step 3.3.3 Output the model, the test RMSE and the rating matrix

The best group of parameters is $D = 5,\ \sigma_v = 0.1,\ \sigma_u = 0.5,\ \lambda = 1$ and use the Gaussian kernel. Then we use these to build the model and to see the RMSE on the test set.

```
t0<-Sys.time()

output<-gradPMF(D = 5, data, data.train, data.test, sigma = .5,sigma_V = .1, sigma_U = .5, max.iter = 20
model_A2_p3<-P3(data.train, V = output$V, kernel = "Gaussian",lambda = 1)

t1<-Sys.time()
prediction_A2_P3<-predict.P3(data.new = data.test, model_A2_p3, V = output$V)
t2<-Sys.time()

prediction_A2_P3_train<-predict.P3(data.new = data.train, model_A2_p3, V = output$V)
t3<-Sys.time()

test_RMSE_A2_P3<-sqrt(mean((data.test$rating-prediction_A2_P3)^2))

train_RMSE_A2_P3<-sqrt(mean((data.train$rating-prediction_A2_P3_train)^2))

##Time for build the model is about 3.124177 mins
t1-t0
##Time for modeling and output the prediction of test set is about 3.150982 mins
t2-t0
```

```
##Time for modeling and output the prediction of training set is about 196.4251 secs
t3-t0-(t2-t1)

save(train_RMSE_A2_P3,file = "../output/train_RMSE_A2_P3.RData")
save(test_RMSE_A2_P3,file = "../output/test_RMSE_A2_P3.RData")
```

Calculate the user-movie prediction matrix.

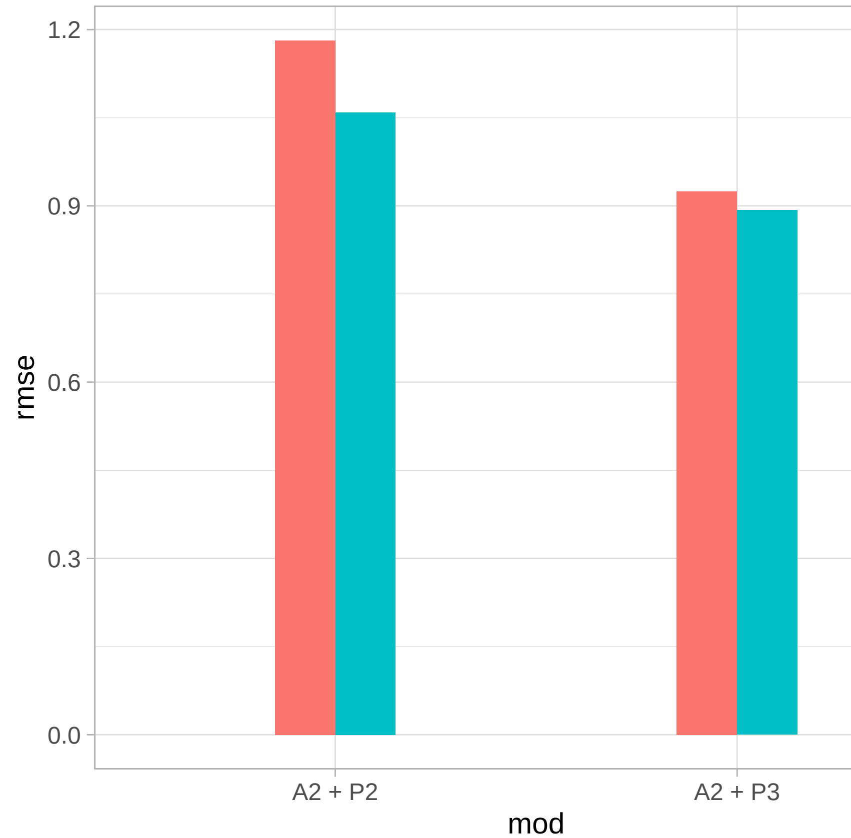
```
data.pred<-expand.grid(userId = c(1:610),movieId = sort(unique(data$movieId)))

t4<-Sys.time()
prediction<-predict.P3(data.new = data.pred, model_A2_p3, V = output$V)
t5<-Sys.time()

ratings_A2_P3<-cbind(data.pred,prediction)%>%
  pivot_wider(1:3,names_from = "movieId",values_from = "prediction")%>%
  select(-userId)

#Time for building the rating matrix is 5.124815 mins
t4-t5

#The total run time for krr function is 8.444076 mins, including modeling, outputing the prediction for
t<-t3-t0+t5-t4

#part of the rating matrix
ratings_A2_P3[1:5,1:5]

save(ratings_A2_P3,file = "../output/ratings_A2_P3.RData")
save(t,file = "../output/run_time_A2_P3.RData")
```

Step 4 Comparation

We test these two model with data.test, the rmse is being showed below.

So we can conclude that:

From RMSE perspective, Probabilistic Matrix Factorization with Kernel Ridge Regression performs better



than Probabilistic Matrix Factorization with KNN.

We also caculate the run time of training + predicting + Rmse calculating, where for A2 + P2 is 7.615576 mins and for A2 + P3 is 8.444076 mins. But we think that comparing this time is meaningless, because we use different matrices to implement the given algorithm.