IS622-Final

Rohan Fray

Load packages and set constants

```
library(jsonlite)
library(Matrix)
library(irlba)

library(rmr2)
library(rhdfs)
hdfs.init()

#d here represents the other dimension of our U and V matrices
d=2
```

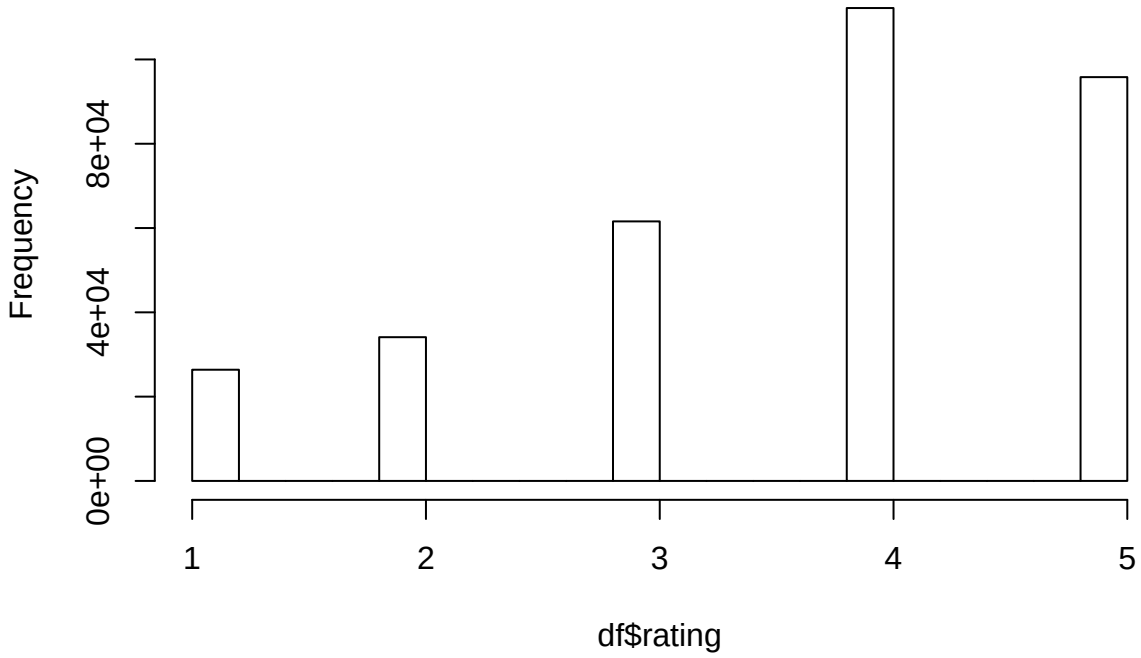
```
## NULL
```

Read Yelp data file and pull out only reviews

```
conn <- file("/home/tengig/tmpNFvucr",open="r")
linn <-readLines(conn)

j=0
x <- numeric(330071)
y <- character(330071)
y2 <- character(330071)
for (i in 1:length(linn)){
  tmp = fromJSON(linn[i])
  if (tmp$type == "review"){
    x[j] <- tmp$stars
    y[j] <- tmp$user_id
    y2[j] <- tmp$business_id
    j=j+1
  }
}
df <- data.frame(y,y2,x, stringsAsFactors=F)
colnames(df) <- c("userID","businessID","rating")

#removing blank rows
df[df==""]<-NA
df = na.omit(df)
hist(df$rating)
```



Get our sample and preprocess the data via RHadoop

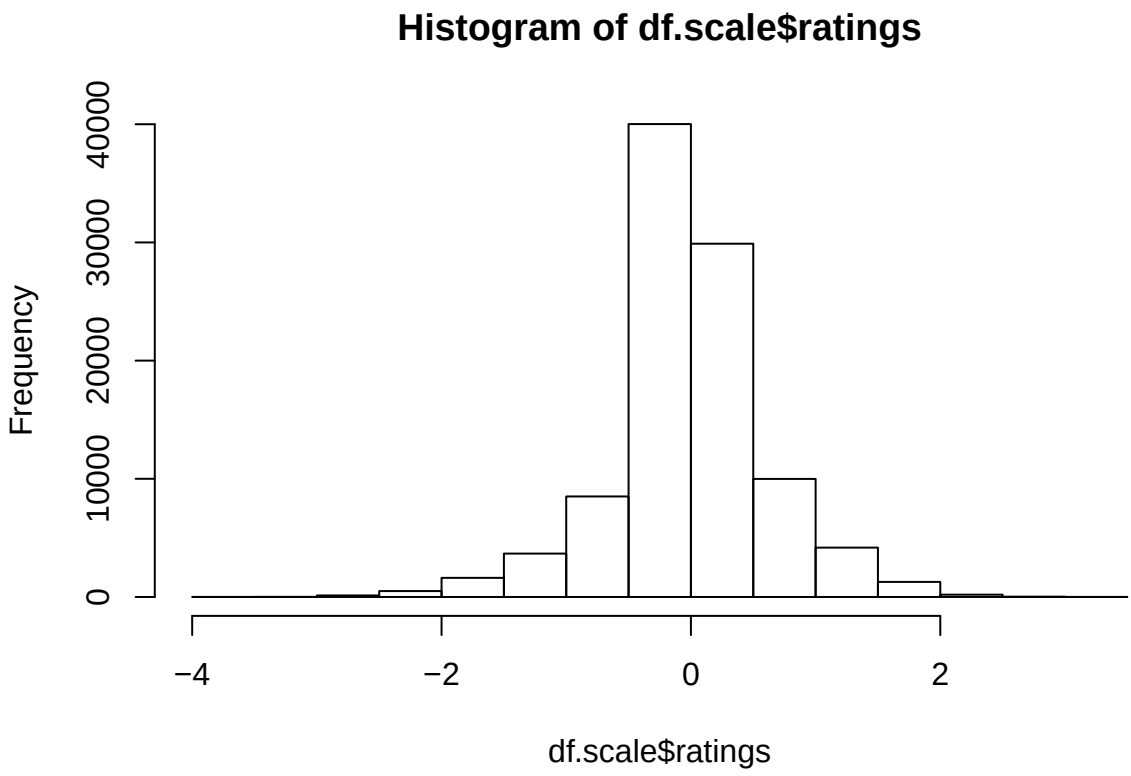
```
set.seed(622)
df.subset = df[sample(330070,100000),]

inp <- to.dfs(df.subset)
#scale by User
PreProcessUser = mapreduce(input = inp,
                           map = function(.,v){
                             keyval(v[,1],v[,c(2,3)])
                           },
                           reduce = function(k,v){
                             avg = scale(v[,2],scale = FALSE)
                             keyval(k,cbind(v[,1],avg))
                           }
)


ans <- from.dfs(PreProcessUser)
#recreate df by ans
ratings <- as.numeric(ans$val[,2])
businesses <- ans$val[,1]
users <- ans$key
df.scale <- data.frame(users,businesses,ratings, stringsAsFactors=F)
```

```
#scale by Business
inp = to.dfs(df.scale)
PreProcessBusiness = mapreduce(input = inp,
                               map = function(.,v){
                                 keyval(v[,2],v[,c(1,3)])
                               },
                               reduce = function(k,v){
                                 avg = scale(v[,2],scale = FALSE)
                                 keyval(k,cbind(v[,1],avg))
                               }
)

ans <- from.dfs(PreProcessBusiness)
#recreate df by ans
ratings <- as.numeric(ans$val[,2])
users <- ans$val[,1]
businesses <- ans$key
df.scale <- data.frame(users,businesses,ratings, stringsAsFactors=F)
hist(df.scale$ratings)
```



What is the RMSE of an initial U and V?

Since the mean of the ratings is 0, we will use the standard deviation. So, what is the RMSE if U and V had all the same value of $sqrt(a/d)$

```
a = sd(df.scale$ratings)
val = 2*(sqrt(a/d))^2
SE <- lapply(df.scale$ratings, function(x){(x-val)^2})
MSE <- Reduce("+", SE)/length(SE)
RMSE.ad <- sqrt(MSE)
RMSE.ad
```

```
## [1] 0.8706016
```

Create U and V

We will use jitter to create U and V

```
testdf <- df.scale
set.seed(622)
testU <- jitter(matrix(sqrt(a/d),
                       nrow = length(unique(testdf$users)),
                       ncol = d))
rownames(testU)<-unique(testdf$users)

set.seed(622)
testV <- jitter(matrix(sqrt(a/d),
                       ncol = length(unique(testdf$businesses)),
                       nrow = d))
colnames(testV) <- unique(testdf$businesses)

testdf2 <-df.scale
```

Calculate RMSE of this U and V

```
Uvals<- matrix(unlist(lapply(testdf2$users,
                             function(x){testU[x,]})),ncol=d,byrow = T)
```

```
## Warning: closing unused connection 5 (/home/tengig/tmpNFvucr)
```

```
Vvals<- t(matrix(unlist(lapply(testdf2$businesses,
                               function(x){testV[,x]})),nrow=d,byrow=F))
vals<-apply(cbind(Uvals,Vvals),1,function(x){
  newcalc<-0
  for(i in range(1:d)){
    newcalc = newcalc + x[i]*x[d+i]
  }
  newcalc
})
SE <- (vals-testdf$ratings)^2
MSE <- sum(SE)/nrow(df.scale)
RMSE.old <- sqrt(MSE)
RMSE.old
```

```
## [1] 0.8705821
```

Iterate two times for new U and V values

```
for (iter in (1:2)){
  #Change U values
  inpU <- to.dfs(testU)
  ChangeU <- mapreduce(input = inpU,
                       map = function(.,v){
                         keyval(rownames(v),v)
                       },
                       reduce = function(k,v){
                         #our values for the row in U and each of its column
                         vals <- numeric(d)

                         #values of M for the row in U
                         tmp.M <- testdf[testdf$users==k,]

                         for (i in range(1,d)){
                           denom <- sum((testV[i,unlist(tmp.M$businesses)])^2)
                           num <-0
                           for (j in tmp.M$businesses){
                             temp1 <- testV[i,j]
                             temp2 <- tmp.M[tmp.M$businesses==j,]$ratings
                             temp3 <- 0
                             for (kiter in range(1,d)){
                               if (kiter != i){
                                 temp3 = temp3 + (v[kiter]*testV[kiter,j])
                               }
                             }
                             num = temp1*(temp2-temp3)

                           }

                           vals[i] = (num/denom)
                         }

                         keyval(k,list(vals))
                       })

  Uout<-from.dfs(ChangeU)
  testU<-matrix(unlist(Uout$val),ncol = d,byrow = TRUE)
  rownames(testU)<-Uout$key

  #Change values in V
  inpV <- to.dfs(testV)
  ChangeV <- mapreduce(input = inpV,
                       map = function(.,v){
                         keyval(colnames(v),t(v))
                       },
                       reduce = function(k,v){
                         #our values for the column in V and each of its row
                         vals <- numeric(d)

                         #values of M for the row in U
```

```
                          tmp.M <- testdf[testdf$businesses==k,]

                          for (i in range(1,d)){
                            denom <- sum((testU[unlist(tmp.M$users),i])^2)

                            num <-0
                            for (j in tmp.M$users){
                              temp1 <- testU[j,i]
                              temp2 <- tmp.M[tmp.M$users==j,]$ratings
                              temp3 <- 0
                              for (kiter in range(1,d)){
                                if (kiter != i){
                                  temp3 = temp3 + (v[kiter]*testU[j,kiter])
                                }
                              }
                              num = temp1*(temp2-temp3)
                            }

                            vals[i] = (num/denom)
                          }

                          keyval(k,list(vals))
                        })

  Vout<-from.dfs(ChangeV)
  testV <- matrix(unlist(Vout$val),nrow = d)
  colnames(testV)<-Vout$key
}
```

Final RMSE after iterations

```
Uvals<- matrix(unlist(lapply(testdf2$users,
                             function(x){testU[x,]})),ncol=d,byrow = T)
Vvals<- t(matrix(unlist(lapply(testdf2$businesses,
                               function(x){testV[,x]})),nrow=d,byrow=F))

vals<-apply(cbind(Uvals,Vvals),1,function(x){
  newcalc<-0
  for(i in range(1:d)){
    newcalc = newcalc + x[i]*x[d+i]
  }
  newcalc
})
SE <- (vals-testdf$ratings)^2
MSE <- sum(SE)/nrow(df.scale)
RMSE <- sqrt(MSE)
RMSE
```

```
## [1] 0.6211447
```

Difference between original and new RMSE

```
RMSE.old - RMSE
```

```
## [1] 0.2494374
```

Using irlba

```
us<-as.factor(df.scale$users)
us.int <-as.integer(us)

bs<-as.factor(df.scale$businesses)
bs.int <-as.integer(bs)

mat<-with(df.scale,
          sparseMatrix(i=us.int,
                       j=bs.int,
                       x=ratings,
                       dimnames=list(levels(us), levels(bs))))


L <- irlba(mat,nv = 2,nu = 2)

L.d <- diag(L$d)
irlbaTestU<-L$u%*%L.d
rownames(irlbaTestU)<-levels(us)

irlbaTestV <- t(L$v)
colnames(irlbaTestV)<-levels(bs)
```

The RMSE for the irlba decomposition

```
Uvals<- matrix(unlist(lapply(testdf2$users,
                             function(x){irlbaTestU[x,]})),ncol=d,byrow = T)
Vvals<- t(matrix(unlist(lapply(testdf2$businesses,
                               function(x){irlbaTestV[,x]})),nrow=d,byrow=F))

vals<-apply(cbind(Uvals,Vvals),1,function(x){
  newcalc<-0
  for(i in range(1:d)){
    newcalc = newcalc + x[i]*x[d+i]
  }
  newcalc
})
SE <- (vals-testdf$ratings)^2
MSE <- sum(SE)/nrow(df.scale)
RMSE.irlba <- sqrt(MSE)
RMSE.irlba
```

```
## [1] 0.6120515
```

The three RMSEs

The original RMSE with our initial U and V

```
RMSE.old
```

```
## [1] 0.8705821
```

The RMSE after our mapreduce calls

```
RMSE
```

```
## [1] 0.6211447
```

The RMSE of the irlba decomp

```
RMSE.irlba
```

```
## [1] 0.6120515
```