Game recommender Project

Biswarup Mukherjee

28/04/2022

Steam Games User dataset: https://www.kaggle.com/tamber/steam-video-games (https://www.kaggle.com/tamber/steam-video-games)

BISWARUP MUKHERJEE, TAMOJIT ROY, AKASH PATRA

Including required packages

```
suppressMessages(library(dplyr))
suppressMessages(library(ggplot2))
suppressMessages(library(mixtools))
```

```
## Warning: package 'mixtools' was built under R version 4.1.3
```

```
suppressMessages(library(caTools))
```

```
## Warning: package 'caTools' was built under R version 4.1.3
```

```
suppressMessages(library(recommenderlab))
```

```
## Warning: package 'recommenderlab' was built under R version 4.1.3
```

```
## Warning: package 'arules' was built under R version 4.1.3
```

```
## Warning: package 'proxy' was built under R version 4.1.3
```

```
suppressMessages(library(reshape2))
```

```
steam <- read.csv('steam-200k.csv', header = FALSE)[-5]
```

```
head(steam)
```

```
##          V1                          V2       V3    V4
## 1 151603712 The Elder Scrolls V Skyrim purchase   1.0
## 2 151603712 The Elder Scrolls V Skyrim     play 273.0
## 3 151603712                  Fallout 4 purchase   1.0
## 4 151603712                  Fallout 4     play  87.0
## 5 151603712                      Spore purchase   1.0
## 6 151603712                      Spore     play  14.9
```

We give user specific column names to the dataset

```
colnames(steam) <- c('user', 'game', 'purchase_play', 'hrs')
str(steam)
```

```
## 'data.frame':    200000 obs. of  4 variables:
##  $ user         : int  151603712 151603712 151603712 151603712 151603712 151603712 151603712 151603712 151603712 15160371
2 ...
##  $ game         : chr  "The Elder Scrolls V Skyrim" "The Elder Scrolls V Skyrim" "Fallout 4" "Fallout 4" ...
##  $ purchase_play: chr  "purchase" "play" "purchase" "play" ...
##  $ hrs          : num  1 273 1 87 1 14.9 1 12.1 1 8.9 ...
```

```
steam_clean <- steam
```

```
head(steam)
```

```
##        user                         game purchase_play   hrs
## 1 151603712 The Elder Scrolls V Skyrim      purchase   1.0
## 2 151603712 The Elder Scrolls V Skyrim          play 273.0
## 3 151603712                   Fallout 4      purchase   1.0
## 4 151603712                   Fallout 4          play  87.0
## 5 151603712                       Spore      purchase   1.0
## 6 151603712                       Spore          play  14.9
```

check for any missing values

```
sum(is.na(steam))
```

```
## [1] 0
```

```
apply(steam_clean, 2, function(x) sum(is.na(x)))
```

```
##          user          game purchase_play           hrs
##             0             0             0             0
```

There are no missing values Also, hrs column, which means hours played is already in numeric, it is crucial for our recommender system.

Unless game is purchased, it cant be played.

We are splitting out the purchase_play column into two columns. The data collection essentially duplicates each record with one labelled as ‘purchase’ and one labelled as

‘play’. The ‘purchase’ row records 1 hour.

```
# split into purchase and play
steam_clean$purchase <- sapply(steam_clean$purchase_play, function(x) as.numeric(x == 'purchase'))
steam_clean$play <- sapply(steam_clean$purchase_play, function(x) as.numeric(x == 'play'))
steam_clean$hrs <- steam_clean$hrs-steam_clean$purchase
steam_clean <- steam_clean[,-3] # Keep hrs, purchase and play column.
steam_clean <- aggregate(. ~ user + game, data = steam_clean, FUN = 'sum') # add user and game
head(steam_clean)
```

```
##       user                                                    game hrs
## 1 46055854                                              007 Legends 0.7
## 2 11940338                                                0RBITALIS 0.6
## 3 86055705                                                0RBITALIS 0.3
## 4 93030550                                                0RBITALIS 0.3
## 5 11794760 1... 2... 3... KICK IT! (Drop That Beat Like an Ugly Baby) 0.0
## 6 35701646 1... 2... 3... KICK IT! (Drop That Beat Like an Ugly Baby) 0.0
##   purchase play
## 1        1    1
## 2        1    1
## 3        1    1
## 4        1    1
## 5        1    0
## 6        1    0
```

here, sapply() is used to apply the function(x) to every value of purchase_play column. Aggregate() Function using formula. 1st numerical then categorical.

Each row in the reformatted dataset represents then a unique interaction user-game.

Exploratory Data Analysis

```
# number of games
ngames <- length(unique(steam_clean$game))

# number of users
nusers <- length(unique(steam_clean$user))

cat("There are", ngames, "games purchased by", nusers, "users")
```

```
## There are 5155 games purchased by 12393 users
```

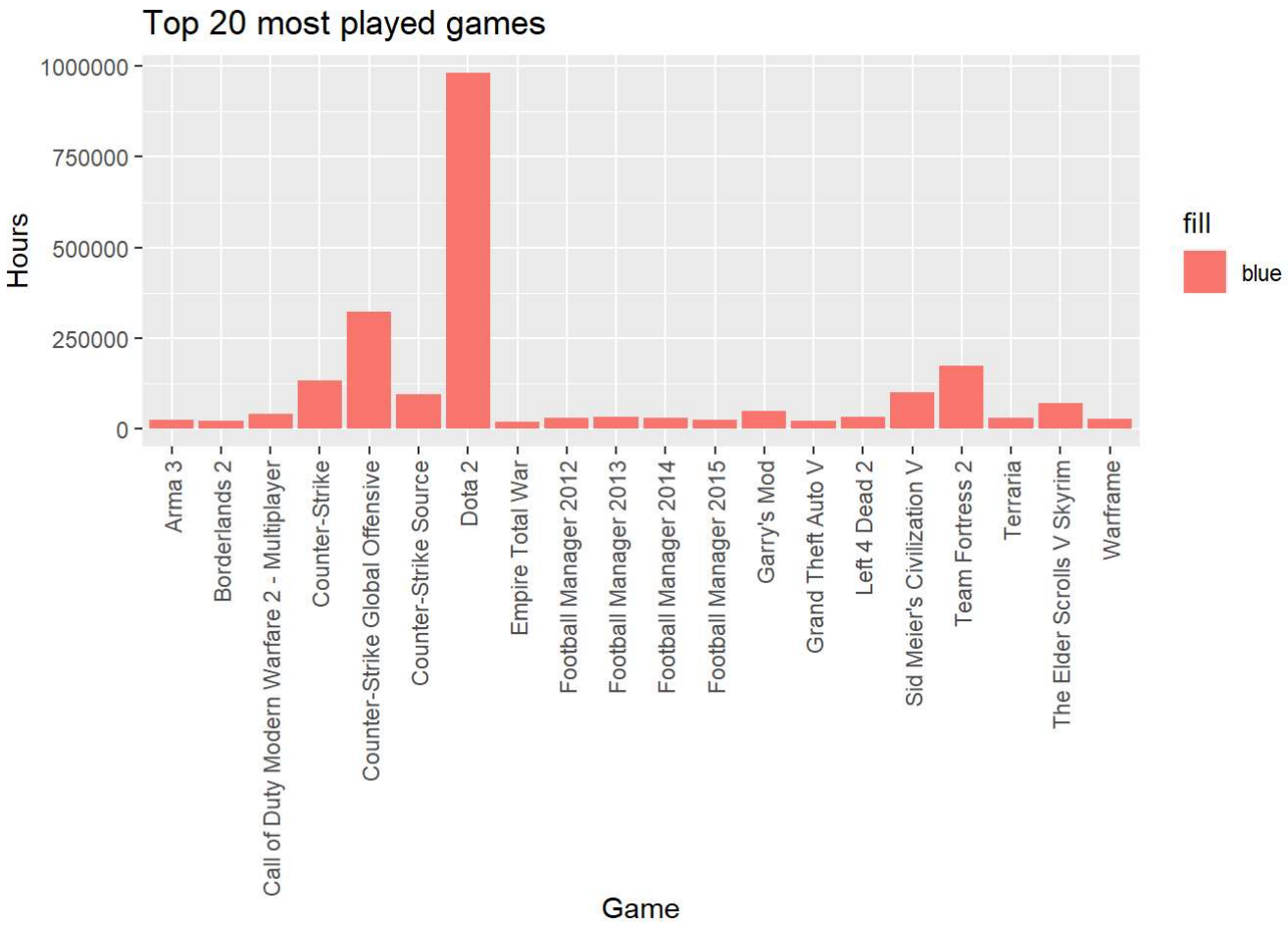
Top 20 games with the most users and hours played.

```
# most played game
game_total_hrs <- aggregate(hrs ~ game, data = steam_clean, FUN = 'sum')
game_total_hrs <- game_total_hrs[order(game_total_hrs$hrs, decreasing = TRUE),]
most_played_games <- head(game_total_hrs, 20)
most_played_games <- data.frame(game = most_played_games$game, hrs = most_played_games$hrs)
df=head(most_played_games, 20)
df
```

```
##                                         game      hrs
## 1                                    Dota 2 981684.6
## 2              Counter-Strike Global Offensive 322771.6
## 3                             Team Fortress 2 173673.3
## 4                              Counter-Strike 134261.1
## 5                   Sid Meier's Civilization V  99821.3
## 6                        Counter-Strike Source  96075.5
## 7                   The Elder Scrolls V Skyrim  70889.3
## 8                                 Garry's Mod  49725.3
## 9  Call of Duty Modern Warfare 2 - Multiplayer  42009.9
## 10                               Left 4 Dead 2  33596.7
## 11                      Football Manager 2013  32308.6
## 12                      Football Manager 2012  30845.8
## 13                      Football Manager 2014  30574.8
## 14                                   Terraria  29951.8
## 15                                   Warframe  27074.6
## 16                      Football Manager 2015  24283.1
## 17                                     Arma 3  24055.7
## 18                         Grand Theft Auto V  22956.7
## 19                              Borderlands 2  22667.9
## 20                        Empire Total War  21030.3
```

Plot of number of hours played for top 20 games.

```
ggplot(df, aes(x = game, y = hrs,fill="blue")) +
    geom_bar(stat = "identity") +
    theme(axis.text.x=element_text(angle=90,hjust=1,vjust=0.5)) +
    labs(title = "Top 20 most played games ", x = "Game", y = "Hours")
```

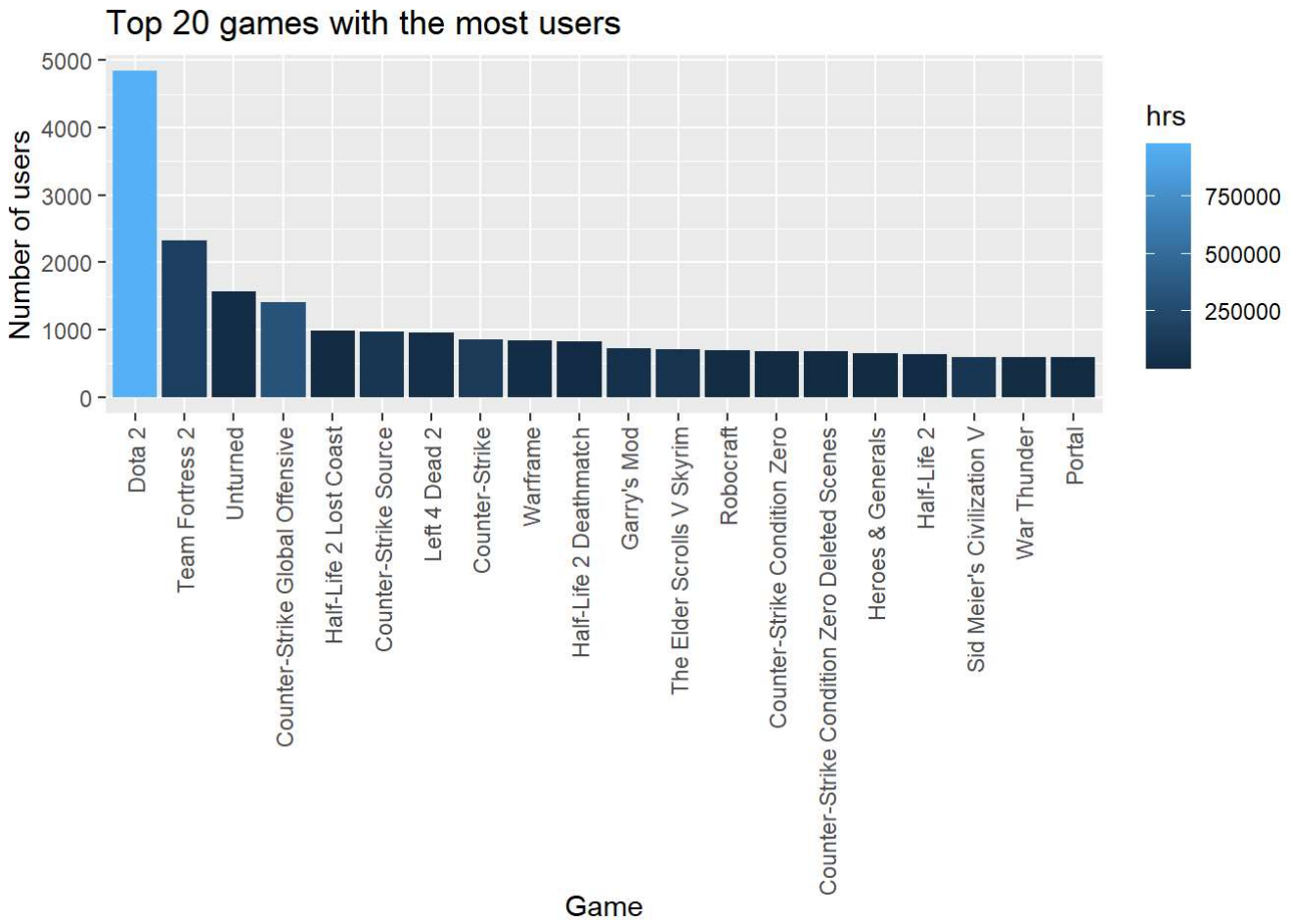


We try to assess if the most purchased games correspond to the most played games.

```
# game with the highest number of users
game_freq <- data.frame(sort(table(steam_clean$game), decreasing = TRUE))
colnames(game_freq) <- c("game", "nusers")
top20 <- merge(game_freq, game_total_hrs, by = 'game')
top20 <- head(top20[order(top20$nusers, decreasing = TRUE),], 20)
top20
```

```
##                                         game nusers      hrs
## 1333                                 Dota 2   4841 981684.6
## 4256                          Team Fortress 2   2323 173673.3
## 4822                                Unturned   1563  16096.4
## 971          Counter-Strike Global Offensive   1412 322771.6
## 2075                   Half-Life 2 Lost Coast    981    184.4
## 974                     Counter-Strike Source    978  96075.5
## 2475                           Left 4 Dead 2    951  33596.7
## 968                           Counter-Strike    856 134261.1
## 4922                                Warframe    847  27074.6
## 2072                  Half-Life 2 Deathmatch    823   3712.9
## 1888                              Garry's Mod    731  49725.3
## 4368                The Elder Scrolls V Skyrim    717  70889.3
## 3554                              Robocraft    689   9096.6
## 969              Counter-Strike Condition Zero    679   7950.0
## 970  Counter-Strike Condition Zero Deleted Scenes    679    418.2
## 2145                        Heroes & Generals    658   3299.5
## 2071                             Half-Life 2    639   4260.3
## 3833               Sid Meier's Civilization V    596  99821.3
## 4917                             War Thunder    590  14381.6
## 3239                                  Portal    588   2282.8
```

```
ggplot(top20, aes(x = game, y = nusers, fill = hrs)) +
    geom_bar(stat = "identity") +
    theme(axis.text.x=element_text(angle=90,hjust=1,vjust=0.5)) +
    labs(title = "Top 20 games with the most users", x = "Game", y = "Number of users")
```



*** INFERENCE *** Clearly Dota 2 has the highest number of players and the highest number of total hours played so undeniably the most popular game. Where as other games such as Ricochet have 524 users but a grand total of 21.2 hours played. an interesting example is ‘Half-Life 2 Lost Coast’ which has a high number of users (981 users), but the total of hours played is quite low (184.4 hours). A possible explanation for this could be that this game was purchased as part of a game bundle.

We can clearly see that for some cases there is no relation between the total number of users and the total of hours played, meaning that a high number of users does not represent an equivalent high total of hours played.

```
head(game_freq)
```

```
##                             game nusers
## 1                          Dota 2   4841
## 2                 Team Fortress 2   2323
## 3                         Unturned   1563
## 4 Counter-Strike Global Offensive   1412
## 5          Half-Life 2 Lost Coast    981
## 6           Counter-Strike Source    978
```

```
# how many purchased but did not play
purchased_not_played <- subset(steam_clean, purchase == 1 & play == 0)
nTransactions <-nrow(purchased_not_played)
nusers <- length(unique(purchased_not_played$user))
nPurchased <- nrow(subset(steam_clean, purchase == 1))
cat("There are", nTransactions, "games purchased out of", nPurchased, "that have not been played by", nusers, "users")
```

```
## There are 57904 games purchased out of 128097 that have not been played by 5949 users
```

lot of games haven’t been played.

```
rest = subset(steam_clean, play == 1)


game_total_hrs1 <- aggregate(hrs ~ game, data = rest, FUN = 'sum')
game_total_hrs1 <- game_total_hrs1[order(game_total_hrs1$hrs, decreasing = TRUE),]
most_played_games1 <- head(game_total_hrs1, 20)
most_played_games1 <- data.frame(game = most_played_games1$game, hrs = most_played_games1$hrs)
head(most_played_games1, 20)
```

```
##                                       game      hrs
## 1                                   Dota 2 981684.6
## 2             Counter-Strike Global Offensive 322771.6
## 3                           Team Fortress 2 173673.3
## 4                            Counter-Strike 134261.1
## 5                 Sid Meier's Civilization V  99821.3
## 6                      Counter-Strike Source  96075.5
## 7                The Elder Scrolls V Skyrim  70889.3
## 8                                 Garry's Mod  49725.3
## 9  Call of Duty Modern Warfare 2 - Multiplayer  42009.9
## 10                              Left 4 Dead 2  33596.7
## 11                     Football Manager 2013  32308.6
## 12                     Football Manager 2012  30845.8
## 13                     Football Manager 2014  30574.8
## 14                                  Terraria  29951.8
## 15                                  Warframe  27074.6
## 16                     Football Manager 2015  24283.1
## 17                                    Arma 3  24055.7
## 18                       Grand Theft Auto V  22956.7
## 19                            Borderlands 2  22667.9
## 20                        Empire Total War  21030.3
```

*** INFERENCE ***

It seems reasonable to turn the distribution of hours played for a game into a rating, at least it's a reasonable way to view the reception of the game. For example a bad game will have a large distribution around the 0-1 hours and a game which was well received will have a distribution of something greater.

We are going to use the EM algorithm to highlight 5 groups which can be considered 1-5 star ratings. Of course this might not be true, for example a user may only play the game for an average number of hours, but love it and would rate it a 5. The EM algorithm will be more appropriate than simply breaking the data into percentiles.

Removing space in between values of games column

```
# cleaning up the game columns.
steam_clean$game1 <- gsub("[^a-zA-Z0-9]", "", steam_clean$game)
head(steam_clean)
```

```
##       user                                                        game hrs
## 1 46055854                                                 007 Legends 0.7
## 2 11940338                                                   0RBITALIS 0.6
## 3 86055705                                                   0RBITALIS 0.3
## 4 93030550                                                   0RBITALIS 0.3
## 5 11794760 1... 2... 3... KICK IT! (Drop That Beat Like an Ugly Baby) 0.0
## 6 35701646 1... 2... 3... KICK IT! (Drop That Beat Like an Ugly Baby) 0.0
##   purchase play                              game1
## 1        1    1                          007Legends
## 2        1    1                           0RBITALIS
## 3        1    1                           0RBITALIS
## 4        1    1                           0RBITALIS
## 5        1    0 123KICKITDropThatBeatLikeanUglyBaby
## 6        1    0 123KICKITDropThatBeatLikeanUglyBaby
```

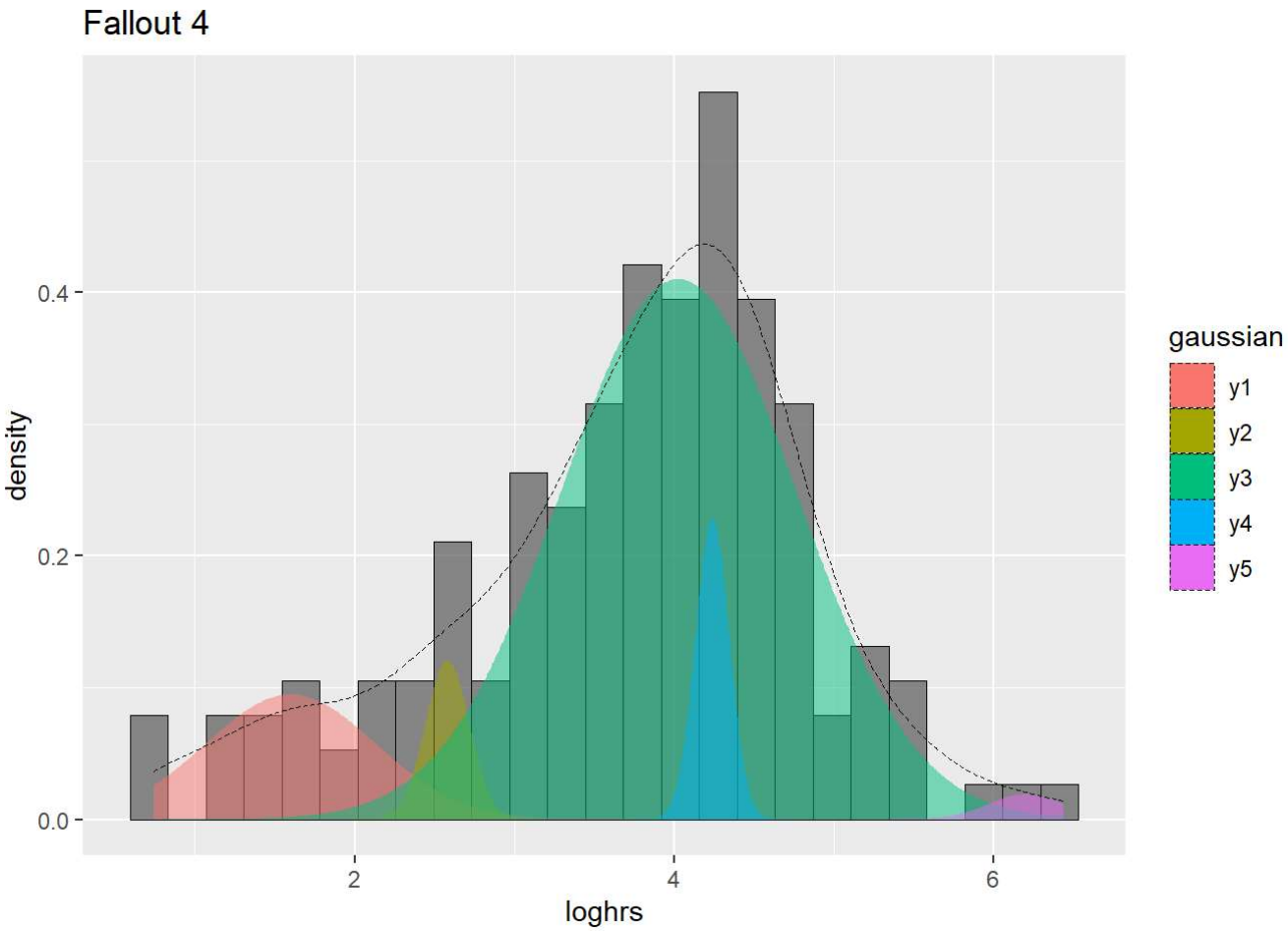
We create the rating system based on the distribution of hours played, this for each game available in the user dataset. We use 5 groups (equivalent to a 5 stars rating system) in order to define a rating users would give to a game they played based on

the hours each one played each game relative to that of everyone else.

```
# create a rating based on time played
game_hrs_density <- function(GAME, nclass, print_vals = TRUE){
  # subsetting data. Ignore the game hrs less than 2 hrs
  game_data <- subset(steam_clean, game1 == GAME & hrs > 2)
  game_data$loghrs <- log(game_data$hrs)

  # em algorithm
  mu.init <- seq(min(game_data$loghrs), max(game_data$loghrs), length = nclass)
  EM <- normalmixEM(game_data$loghrs, mu = mu.init, sigma=rep(1, nclass))

  # print results
  if(print_vals){
    cat(" lambda: ", EM$lambda, "\n mean  : ", EM$mu, "\n sigma : ", EM$sigma, "\n")
  }
  # building data frame for plotting
  x <- seq(min(game_data$loghrs), max(game_data$loghrs), 0.01)
  dens <- data.frame(x = x)
  for(k in 1:nclass){
    dens[,paste0('y', k)] <- EM$lambda[k]*dnorm(x, EM$mu[k], EM$sigma[k])
  }

  dens <- melt(dens, 'x', variable.name = 'gaussian')
  game_plt <- ggplot(game_data, aes(x = loghrs)) +
    geom_histogram(aes(y = ..density..), bins = 25, colour = "black", alpha = 0.7, size = 0.1) +
    geom_area(data = dens, aes(x = x, y = value, fill = gaussian), alpha = 0.5, position = position_dodge()) +
    geom_density(linetype = 2, size = 0.1) +
    ggtitle(game_data$game[1])

  return(game_plt)
}
game_hrs_density("Fallout4", 5, print_vals = TRUE)
```

```
## number of iterations= 840
##  lambda:  0.1266593 0.04165563 0.7589109 0.06148807 0.0112861
##  mean   :  1.601863 2.581101 4.028647 4.239826 6.208128
##  sigma  :  0.5330958 0.1375108 0.7381537 0.1072901 0.2357525
```

```
## Warning: Width not defined. Set with `position_dodge(width = ?)`
```



*** INFERENCE *** EM algorithm find groups (5) of people with similar gaming habits and that would potentially rate a game in a similar way.

We can see few users played ‘The Fallout 4’ game for very few hours. It’s possible some of these users lost their interest into the game shortly after starting playing it. The distribution is denser for groups 3 and 4. This shows that the majority of users are interested in this game. So the game like this would be highly rated.

As it is visible from graph, most of those who played the The Witcher 3 stuck with it and played 40+ hours. However there were a few users where The Witcher didn’t grab them and stopped playing after a few hours. The EM algorithm does a great job finding the groups of people with similar gaming habits and would potentially rate the game in a similar way.

The distribution is denser for some groups . This shows that the majority of users are interested in this game.

A user-item matrix is created with the users being the rows and games being the columns. The missing values are set to zero. The observed values are the log hours for each observed user-game combination. The data was subset to games which have greater than 50 users and users which played the game for greater than 2 hours. This was chosen as 2 hours is the limit in which Steam will offer a return if we did not like the purchased game.

To create a test set 10% of the observed values will be set to 0. The root mean squared error will be calculated to determine the accuracy.

```
# create user item matrix
set.seed(0910)
game_freq$game1 <- gsub("[^a-zA-Z0-9]", "", game_freq$game)
game_users <- subset(game_freq, game_freq$nusers > 50)
steam_clean_pos <- subset(steam_clean, steam_clean$hrs > 2 & (steam_clean$game1 %in% game_users$game1))
steam_clean_pos$loghrs <- log(steam_clean_pos$hrs)

# make matrix
games <- data.frame(game1 = sort(unique(steam_clean_pos$game1)), game_id = 1:length(unique(steam_clean_pos$game1)))
users <- data.frame(user = sort(unique(steam_clean_pos$user)), user_id = 1:length(unique(steam_clean_pos$user)))
steam_clean_pos <- merge(steam_clean_pos, games, by = 'game1')
steam_clean_pos <- merge(steam_clean_pos, users, by = 'user')
ui_mat <- matrix(0, nrow = nrow(users), ncol = nrow(games), dimnames = list(user = paste0("u", sort(unique(steam_clean_pos$u
ser))),
                                                                            game = sort(unique(steam_clean_pos$game1))))
for(k in 1:nrow(steam_clean_pos)){
  ui_mat[steam_clean_pos$user_id[k], steam_clean_pos$game_id[k]] <- steam_clean_pos$loghrs[k]
}

# create training set i.e. suppress a tenth of the actual ratings
index <- sample.split(steam_clean_pos$user, SplitRatio = 0.9)
train <- steam_clean_pos[index,]
test <- steam_clean_pos[!index,]
ui_train <- ui_mat
for(k in 1:nrow(test)){
  ui_train[test$user_id[k], test$game_id[k]] <- 0
}

# root mean squared error function
rmse <- function(pred, test, data_frame = FALSE){
  test_pred <- rep(NA, nrow(test))
  for(k in 1:nrow(test)){
    test_pred[k] <- pred[test$user_id[k], test$game_id[k]]
  }
  if(data_frame){
    return(data.frame(test_pred, test$loghrs))
  }
  return(sqrt(1/(nrow(test)-1)*sum((test_pred - test$loghrs)^2)))
```

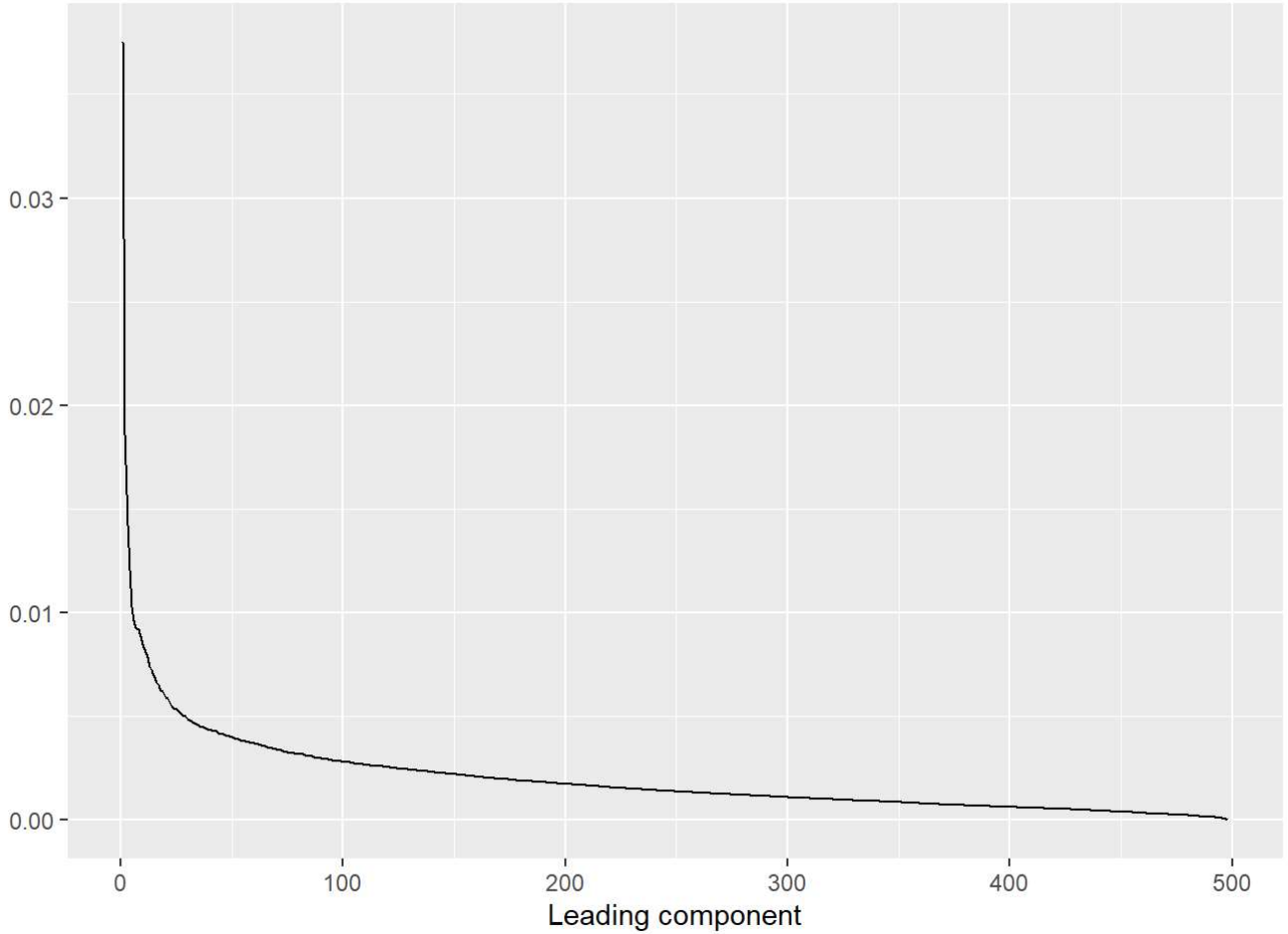
```
}

cat("Dimensions of training user-item matrix:", dim(ui_train))
```

```
## Dimensions of training user-item matrix: 8214 498
```

Basic SVD recommender

The basic SVD approach will perform matrix factorisation using the first 60 leading components(iterations). Since the missing values are set to 0 the factorisation will try and recreate them which is not quite what we want. For this example we will simply impute the missing observations with a mean value. Leading COMPONENTS means Latent factors.

```
Y <- ui_train

# mean impute
Y <- apply(Y, 2, function(x) ifelse(x == 0, mean(x), x))
Y_svd <- svd(Y)
U <- Y_svd$u
V <- Y_svd$v
D <- Y_svd$d
ggplot(data.frame(x = 1:length(D), y = D/sum(D)), aes(x = x, y = y)) +
    geom_line() +
    labs(x = "Leading component", y = "")
```




```
# take the leading components
lc <- 60
pred <- U[,1:lc] %*% diag(D[1:lc]) %*% t(V[,1:lc])

rmse(pred, test)
```

```
## [1] 2.93323
```

```
head(rmse(pred, test, TRUE))
```

```
##     test_pred test.loghrs
## 1 0.195877492    3.178054
## 2 0.263111494    2.797281
## 3 0.006601961    1.131402
## 4 0.665513329    2.501436
## 5 0.172080334    4.406719
## 6 1.467813492    3.688879
```

*** INFERENCE *** Does not give a good prediction

SVD via gradient descent

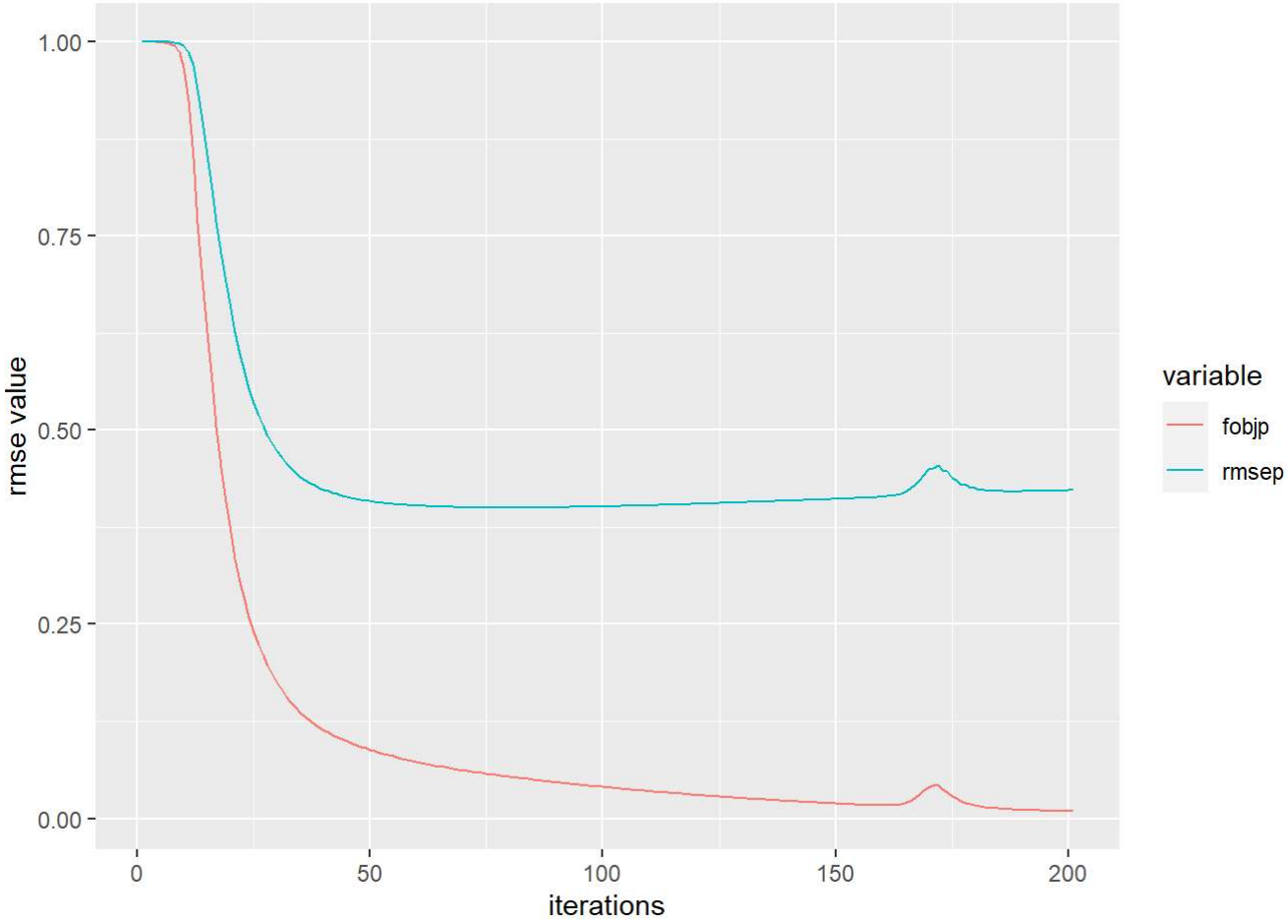
We will use a gradient descent approach to find optimal U and V matrices which retain the actual observations with predict the missing values by drawing on the information between similar users and games. We have chosen a learning rate of 0.001 and will run for 200 iterations tracking the RMSE. The objective function is the squared error between the actual observed values and the predicted values.

```
# svd via gradient descent
# setting matricies
leading_components <- 60
Y <- ui_train
I <- apply(Y, 2, function(x) ifelse(x>0, 1, 0))
U <- matrix(rnorm(nrow(Y)*leading_components, 0, 0.01), ncol = leading_components)
V <- matrix(rnorm(ncol(Y)*leading_components, 0, 0.01), ncol = leading_components)

# objective function
f <- function(U, V){
  return(sum(I*(U%*%t(V)-Y)^2))
}
dfu <- function(U){
  return((2*I*(U%*%t(V)-Y))%*%V)
}
dfv <- function(V){
  return(t(2*I*(U%*%t(V)-Y))%*%U)
}


# gradient descent
N <- 200
alpha <- 0.001
pred <- round(U%*%t(V), 2)
fobj <- f(U, V)
rmsej <- rmse(pred, test)
#pb <- txtProgressBar(min = 0, max = N, style = 3)
start <- Sys.time()
for(k in 1:N){
  U <- U - alpha*dfu(U)
  V <- V - alpha*dfv(V)
  fobj <- c(fobj, f(U, V))
  pred <- round(U%*%t(V), 2)
  rmsej <- c(rmsej, rmse(pred, test))
  #setTxtProgressBar(pb, k)
}
```

```
#close(pb)
Sys.time()-start
```

```
## Time difference of 3.538599 mins
```

```
path1 <- data.frame(itr = 1:(N+1), fobj, fobjp = fobj/max(fobj), rmse = rmsej, rmsep = rmsej/max(rmsej))
path1gg <- melt(path1[c("itr", "fobjp", "rmsep")], id.vars = "itr")
ggplot(path1gg, aes(itr, value, color = variable)) + geom_line()+labs(x = "iterations", y = "rmse value")
```



```
dimnames(pred) <- list(user = rownames(ui_train), game = colnames(ui_train))

# printing final iteration
tail(path1, 1)
```

```
##     itr     fobj       fobjp     rmse     rmsep
## 201 201 3279.008 0.009087652 1.369454 0.4225904
```

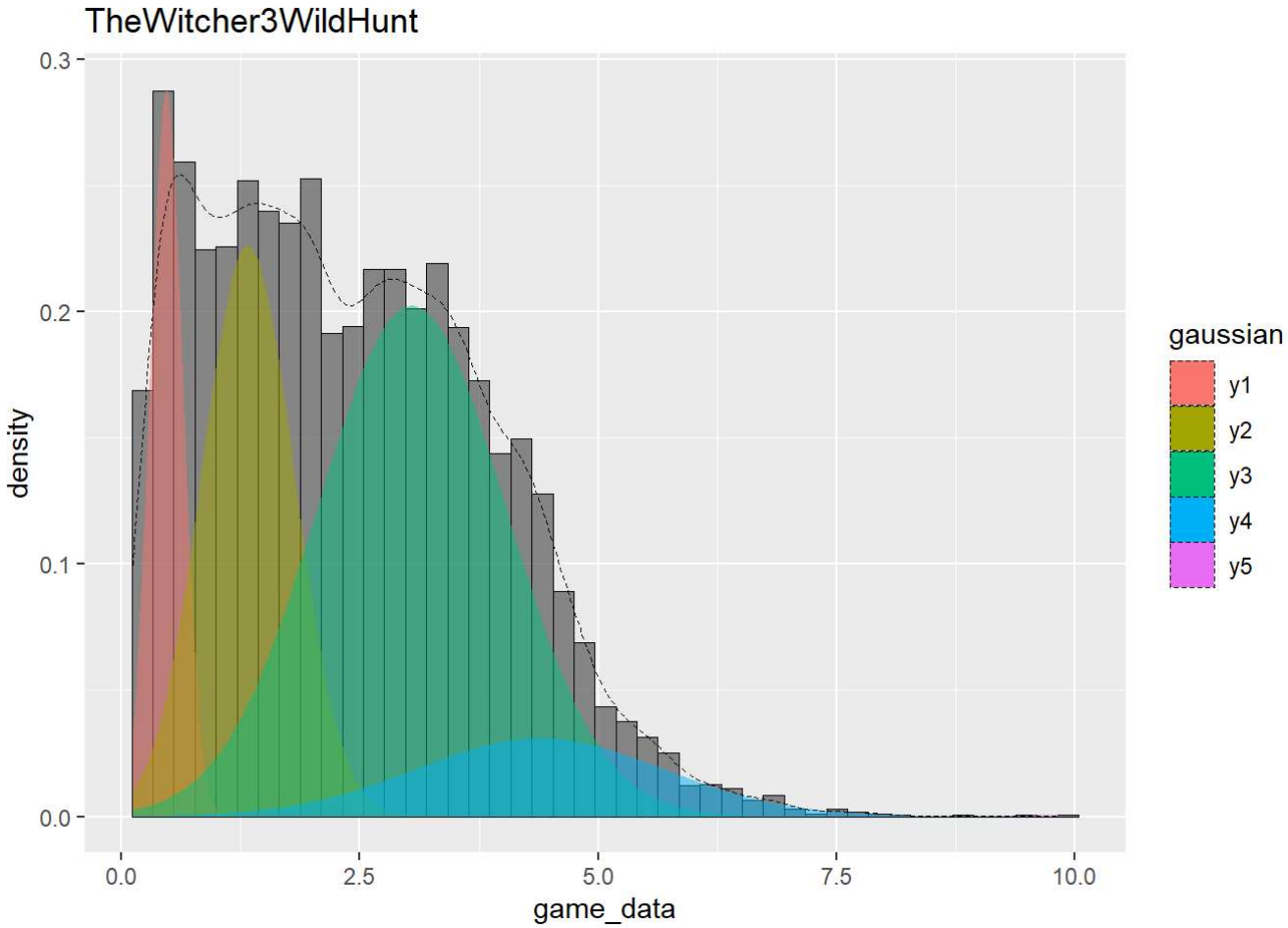
A large improvement on the basic SVD approach. The output shows the objective function converged to 0 on the training data, while the error in the test set essentially halved. Interestingly after the 75th iteration the accuracy in the test set decreased. This could be improved by using more leading components, the trade off being computation time. We can stop after 75 to 100 iterations for this data. It is the prediction of the unobserved which is the goal for test data.

Using the predicted user-item matrix, we will look at the distribution of hours and apply an EM algoritm to find a reasonable 1-5 star rating

```
# create a rating based on time played
# should consolodate this with the other one
game_hrs_density_p <- function(pred, GAME = NULL, nclass, print_vals = TRUE){

  if(is.null(GAME)){
    GAME <- sample(colnames(pred), 1)
  }

  # subsetting data
  game_data <- subset(pred[,GAME], pred[,GAME] > 0)

  # em algorithm
  mu.init <- seq(min(game_data), max(game_data), length = nclass)
  EM <- normalmixEM(game_data, mu = mu.init, sigma=rep(1, nclass), fast = TRUE)

  # print results
  if(print_vals){
    cat(" lambda: ", EM$lambda, "\n mean  : ", EM$mu, "\n sigma : ", EM$sigma, "\n")
  }

  # building data frame for plotting
  x <- seq(min(game_data), max(game_data), 0.01)
  dens <- data.frame(x = x)
  for(k in 1:nclass){
    dens[,paste0('y', k)] <- EM$lambda[k]*dnorm(x, EM$mu[k], EM$sigma[k])
  }

  dens <- melt(dens, 'x', variable.name = 'gaussian')
  game_plt <- ggplot(as.data.frame(game_data), aes(x = game_data)) +
    geom_histogram(aes(y = ..density..), bins = 45, colour = "black", alpha = 0.7, size = 0.1) +
    geom_area(data = dens, aes(x = x, y = value, fill = gaussian), alpha = 0.5, position = position_dodge()) +
    geom_density(linetype = 2, size = 0.1) +
    ggtitle(GAME)

  return(game_plt)
}
game_hrs_density_p(pred, "TheWitcher3WildHunt", 5)
```

```
## WARNING! NOT CONVERGENT!
## number of iterations= 1000
##  lambda:  0.124213 0.2746307 0.500309 0.100609 0.0002383166
##  mean  :  0.4732782 1.327967 3.047156 4.393495 9.686336
##  sigma :  0.1720453 0.484695 0.9869391 1.300674 0.144994
```

```
## Warning: Width not defined. Set with `position_dodge(width = ?)`
```



It is not quite as appropriate this time as all the new predictions create a dense distribution. The 2-4 distributions look like they fit fairly well. The 5 on the other hand is rather flat and only picks up the very end of the tail.

Now we use a percentile approach to recommend the top 10 games for a user for predicted user item matrix

```
# List the top games for a user that they haven't purcahsed
pred_percentile <- apply(pred, 2, percent_rank)
top <- function(n, user = NULL){
  if(is.null(user)){
    user <- sample(rownames(pred_percentile), 1)
  }
  not_purchased <- (I-1)%%2
  top_games <- names(sort((pred_percentile*not_purchased)[user,], decreasing = TRUE))[1:n]
  cat("top", n, "recommended games for user", user, ":\n")
  for(k in 1:n){
    cat(k, ")", top_games[k], "\n")
  }
}
df1=top(10)
```

```
## top 10 recommended games for user u302054444 :
## 1 ) Unturned
## 2 ) Arma2OperationArrowhead
## 3 ) CounterStrikeSource
## 4 ) AlienSwarm
## 5 ) PlanetSide2
## 6 ) GarrysMod
## 7 ) H1Z1
## 8 ) CallofDutyBlackOpsMultiplayer
## 9 ) KillingFloor
## 10 ) NapoleonTotalWar
```