Basic MovieLens recommendation script

HarvardX Data Science Capstone Project

Codrin Kruijne

2019-05-31

```
# PLEASE NOTE
# THIS SCRIPT HAS ONLY BEEN TESTED ON A 6-CORE (12-THREAD), 64GB MEMORY MACHINE
# AND WILL PROBABLY NOT RUN WITH LOWER COMPUTER SPECIFICATIONS.
parallel::detectCores()
```

```
## [1] 12
```

```
memory.limit()
```

```
## [1] 65471
```

```
# Script settings
knitr::opts_chunk$set(
    message = FALSE,
    warning = FALSE,
    cache = TRUE,
    tidy.opts = list(width.cutoff = 100),
    tidy = TRUE
)
script_start <- Sys.time()

# Load required packages
library(tidyverse)
library(lubridate)
library(parallel)
library(doParallel)
```

Loading packages and data

The dataset was downloaded from the GroupLens website using the course provided script and stored locally. Information about the data was found on the [MovieLens README] (http://files.grouplens.org/datasets/movielens/ml-10m-README.html)

```
# Load locally stored data
edx_original <- readRDS("edx.RData")
print(str(edx_original))
```

```
## 'data.frame':    9000055 obs. of  6 variables:
##  $ userId   : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ movieId  : num  122 185 292 316 329 355 356 362 364 370 ...
```

```
##  $ rating   : num  5 5 5 5 5 5 5 5 5 5 ...
##  $ timestamp: int  838985046 838983525 838983421 838983392 838983392 838984474 838983653 838984885 83
##  $ title    : chr  "Boomerang (1992)" "Net, The (1995)" "Outbreak (1995)" "Stargate (1994)" ...
##  $ genres   : chr  "Comedy|Romance" "Action|Crime|Thriller" "Action|Drama|Sci-Fi|Thriller" "Action|Ad
## NULL
```

```r
validation_original <- readRDS("validation.RData")
print(str(validation_original))
```

```
## 'data.frame':    999999 obs. of  6 variables:
##  $ userId   : int  1 1 1 2 2 2 3 3 4 4 ...
##  $ movieId  : num  231 480 586 151 858 ...
##  $ rating   : num  5 5 5 3 2 3 3.5 4.5 5 3 ...
##  $ timestamp: int  838983392 838983653 838984068 868246450 868245645 868245920 1136075494 1133571200
##  $ title    : chr  "Dumb & Dumber (1994)" "Jurassic Park (1993)" "Home Alone (1990)" "Rob Roy (1995)
##  $ genres   : chr  "Comedy" "Action|Adventure|Sci-Fi|Thriller" "Children|Comedy" "Action|Drama|Romanc
## NULL
```

```r
# Make sure movies and users are in both train and test sets
train_data <- edx_original
print(str(train_data))
```

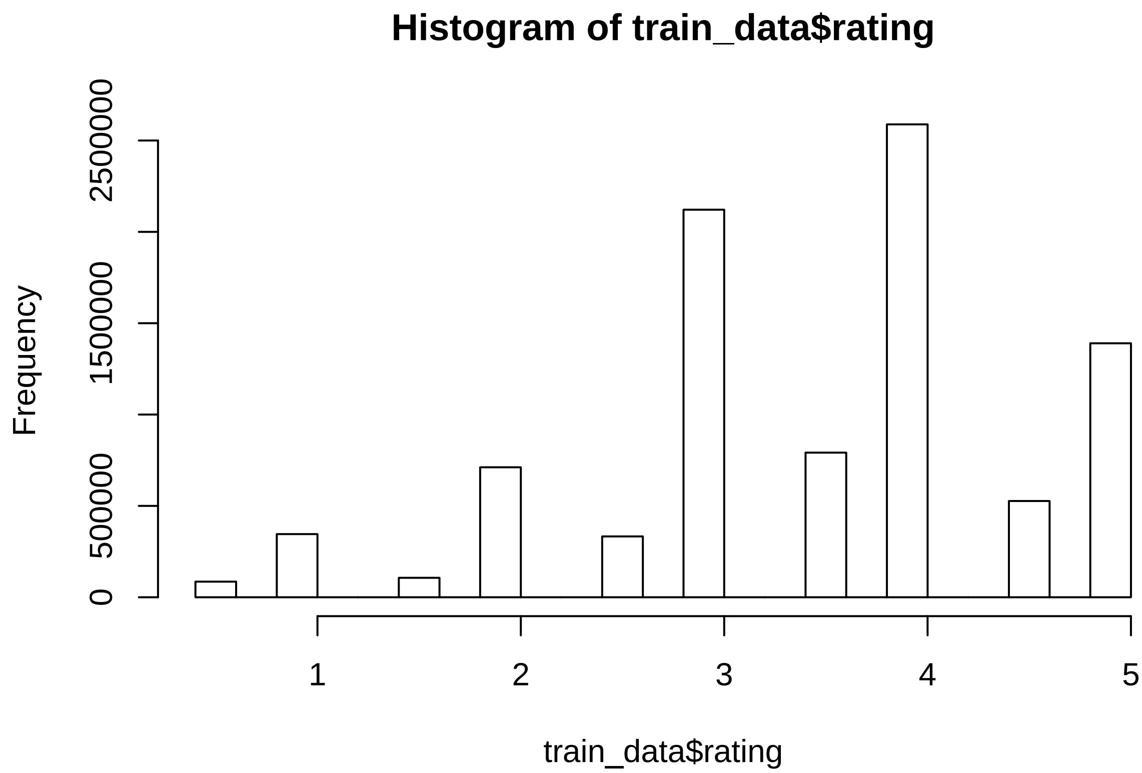
```
## 'data.frame':    9000055 obs. of  6 variables:
##  $ userId   : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ movieId  : num  122 185 292 316 329 355 356 362 364 370 ...
##  $ rating   : num  5 5 5 5 5 5 5 5 5 5 ...
##  $ timestamp: int  838985046 838983525 838983421 838983392 838983392 838984474 838983653 838984885 83
##  $ title    : chr  "Boomerang (1992)" "Net, The (1995)" "Outbreak (1995)" "Stargate (1994)" ...
##  $ genres   : chr  "Comedy|Romance" "Action|Crime|Thriller" "Action|Drama|Sci-Fi|Thriller" "Action|Ad
## NULL
```

```r
test_data <- validation_original %>%
     semi_join(train_data, by = "movieId") %>%
     semi_join(train_data, by = "userId")
print(str(test_data))
```

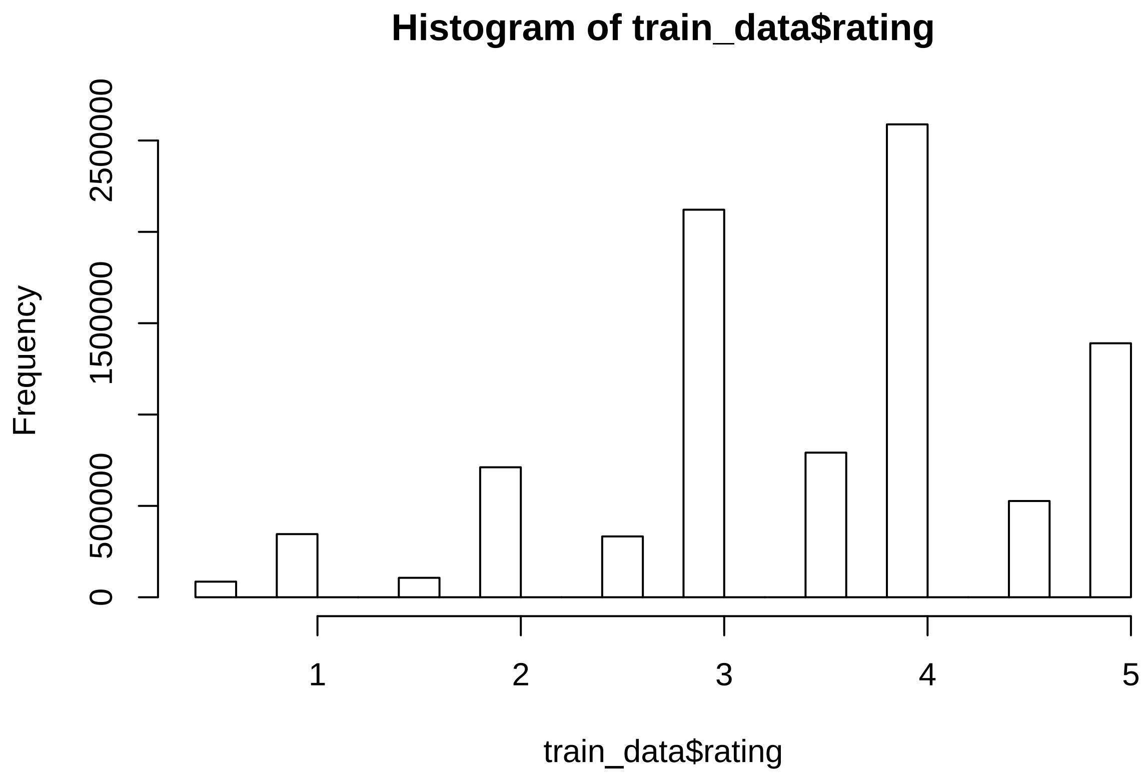
```
## 'data.frame':    999999 obs. of  6 variables:
##  $ userId   : int  1 1 1 2 2 2 3 3 4 4 ...
##  $ movieId  : num  231 480 586 151 858 ...
##  $ rating   : num  5 5 5 3 2 3 3.5 4.5 5 3 ...
##  $ timestamp: int  838983392 838983653 838984068 868246450 868245645 868245920 1136075494 1133571200
##  $ title    : chr  "Dumb & Dumber (1994)" "Jurassic Park (1993)" "Home Alone (1990)" "Rob Roy (1995)
##  $ genres   : chr  "Comedy" "Action|Adventure|Sci-Fi|Thriller" "Children|Comedy" "Action|Drama|Romanc
## NULL
```

```r
all_data <- setNames(list(train_data, test_data), c("train_data", "test_data"))

# Let's have a quick look at the rating training data
rating_hist <- hist(train_data$rating)
```




```r
saveRDS(rating_hist, "rating_hist.rds")
plot(rating_hist)
```



Preparing data

Calculating statistics

```r
# For all data entities we calculate some basic statistics

# User statistics
calc_user_stats <- function(rating_data){

  rating_data %>% group_by(userId) %>%
                  summarise(user_ratings = n(),
                            user_median = median(rating),
                            user_mean = mean(rating),
                            user_sd = sd(rating)) %>%
                  select(userId, user_median, user_mean, user_sd) %>%
                  unique()

}

user_stats <- parLapply(pcl, rating_data, calc_user_stats)

# Movie rating statistics
```

```r
calc_movie_stats <- function(rating_data){

  rating_data %>% group_by(movieId) %>%
                  summarise(movie_ratings = n(),
                            movie_median = median(rating),
                            movie_mean = mean(rating),
                            movie_sd = sd(rating)) %>%
                  select(movieId, movie_median, movie_mean, movie_sd) %>%
                  unique()

}

movie_stats <- parLapply(pcl, rating_data, calc_movie_stats)

# Merge statistics back
merge_data <- list(list(rating_data[[1]], user_stats[[1]], movie_stats[[1]]), # train data
                   list(rating_data[[2]], user_stats[[2]], movie_stats[[2]])) # test data

merge_stats <- function(data_stats){

  rating_data <- data_stats[[1]]
  user_stats <- data_stats[[2]]
  movie_stats <- data_stats[[3]]

  rating_data %>% inner_join(user_stats, by = "userId") %>%
                  inner_join(movie_stats, by = "movieId")

}

rating_data <- parLapply(pcl, merge_data, merge_stats)

## Rating characteristics
calc_rating_stats <- function(rating_data){
  rating_mean <- mean(rating_data$rating)

  rating_data <- rating_data %>% mutate(rating_mean_diff = rating - rating_mean) %>%
                                 mutate(movie_mean_diff = rating - movie_mean) %>%
                                 mutate(user_mean_diff = rating - user_mean)

}

rating_data <- parLapply(pcl, rating_data, calc_rating_stats)
stopCluster(pcl)
```

Derived models

```r
# Let's take the rating data with the added statistics
train_ratings <- rating_data[[1]]
test_ratings <- rating_data[[2]]

saveRDS(train_ratings, "train_ratings.rds")
```

```r
saveRDS(test_ratings, "test_ratings.rds")

# RSME loss function function to evaluate models as prescribed by assignment
RMSE <- function(predicted_ratings, true_ratings){
     sqrt(mean((true_ratings - predicted_ratings)^2))
}

# Deried models

# Naive model using average rating
train_mean <- mean(train_ratings$rating)
train_mean
```

```
## [1] 3.512465
```

```r
naive_rmse <- RMSE(test_ratings$rating, train_mean)

derived_results <- tibble(method = "Average training rating", RMSE = naive_rmse)

# Model using fixed number
fixed_number <- rep(3.5, nrow(test_ratings))

fixed_rmse <- RMSE(test_ratings$rating, fixed_number)

derived_results <- bind_rows(derived_results,
                             tibble(method = "Fixed number 3.5",
                                    RMSE = fixed_rmse ))

# Derived predictions

movie_pred <- test_ratings %>% group_by(movieId) %>%
               # Rating effect (effect of rating scale used; looking at differences from rating mean)
                               summarise(rating_mean_diff_avg = train_mean + mean(rating_mean_diff),
               # Movie effect (effect of general movie popularity; looking at differences between user
                                         movie_mean_diff_avg = train_mean + mean(movie_mean_diff),
               # User effect (effect of user rating behaviour; looking at differeces between rating and
                                         user_mean_diff_avg = train_mean + mean(user_mean_diff),
               # User-movie effect (effect of user rating behaviour; looking at differeces between user
                                         user_movie_mean_diff_avg = train_mean + mean(mean(movie_mean_di

der_pred <- test_ratings %>% inner_join(movie_pred, by = "movieId")

# Calculate RMSEs
rating_effect_rmse <- RMSE(test_ratings$rating,
                           der_pred$rating_mean_diff_avg)
movie_mean_rmse <- RMSE(test_ratings$rating,
                        test_ratings$movie_mean)
movie_effect_rmse <- RMSE(test_ratings$rating,
                          der_pred$movie_mean_diff_avg)
user_effect_rmse <- RMSE(test_ratings$rating,
                         der_pred$user_mean_diff_avg)
user_movie_effect_rmse <- RMSE(test_ratings$rating,
                               der_pred$user_movie_mean_diff_avg)
```

```r
# Print RMSEs

derived_results <- bind_rows(derived_results,
                             tibble(method = "Rating effect",
                                    RMSE = rating_effect_rmse))
derived_results <- bind_rows(derived_results,
                             tibble(method = "Movie mean",
                                    RMSE = movie_mean_rmse))
derived_results <- bind_rows(derived_results,
                             tibble(method = "Movie effect",
                                    RMSE = movie_effect_rmse))
derived_results <- bind_rows(derived_results,
                             tibble(method = "User effect",
                                    RMSE = user_effect_rmse))
derived_results <- bind_rows(derived_results,
                             tibble(method = "User-movie effect",
                                    RMSE = user_movie_effect_rmse))

saveRDS(derived_results, "derived_results.rds")
print(derived_results %>% arrange(RMSE))
```

```
## # A tibble: 7 x 2
##   method                   RMSE
##   <chr>                   <dbl>
## 1 Movie mean              0.938
## 2 Rating effect           0.938
## 3 User effect             0.945
## 4 Average training rating 1.06
## 5 Movie effect            1.06
## 6 User-movie effect       1.06
## 7 Fixed number 3.5        1.06
```

Train linear models

```r
# Load locally stored data
train_ratings <- readRDS("train_ratings.rds")
test_ratings <- readRDS("test_ratings.rds")

# Free up memory
rm(edx_original,
   validation_original,
   train_data,
   test_data,
   all_data,
   user_data,
   movie_data,
   rating_data,
   merge_data,
   der_pred)
invisible(gc()) # garbage collection
```

```r
# Set up parallel processing
no_cores <- detectCores() - 2
tcl <- makePSOCKcluster(no_cores)
registerDoParallel(tcl)
invisible(clusterEvalQ(tcl, library(caret)))

print(paste("Memory size before garbage collection: ", memory.size()))
```

```
## [1] "Memory size before garbage collection:  1006.56"
```

```r
invisible(gc())
print(paste("Memory size after garbage collection: ", memory.size()))
```

```
## [1] "Memory size after garbage collection:  1005.78"
```

```r
# Define models
user_formulas <- c("rating ~ user_median",
                   "rating ~ user_median + user_mean",
                   "rating ~ user_median + user_mean + user_sd")

movie_formulas <- c("rating ~ movie_median",
                   "rating ~ movie_median + movie_mean",
                   "rating ~ movie_median + movie_mean + movie_sd")

combined_formulas <- c("rating ~ movie_mean + user_mean",
                       "rating ~ movie_median + user_median")

full_formulas <- c("rating ~ movie_median + movie_mean + user_median + user_mean",
                   "rating ~ movie_median + movie_mean + movie_sd + user_median + user_mean + user_sd")

# Model training
e <- simpleError("Catch error")

train_lm <- function(lm_formula, lm_data){

  print(paste("Model formula: ", lm_formula))

  print(system.time({
    model <- tryCatch({
               train(as.formula(lm_formula),
                     data = lm_data, method = "lm",
                     na.action = na.omit)
             }, error = function(e) e, finally = print("Made it!"))
  }))

  # Full train objects are GBs large; save only the final model coefficients
  model_coef <- model$finalModel$coefficients

  # How's our memory?
  print(paste("Memory size before garbage collection: ", memory.size()))
  gc()
  print(paste("Memory size after garbage collection: ", memory.size()))
```

```r
  # Save model locally
  saveRDS(model_coef, paste0("models/", lm_formula, ".rds"))
  # Return results
  print(model_coef)
  model
}

# Train and save locally
system.time(user_fits <- parLapply(tcl, user_formulas, train_lm, lm_data = train_ratings))
```

```
##    user  system elapsed
##  233.59   56.56  880.96
```

```r
#saveRDS(user_fits, "user_fits.rds")
rm(user_fits)

system.time(movie_fits <- parLapply(tcl, movie_formulas, train_lm, lm_data = train_ratings))
```

```
##    user  system elapsed
##  185.73   69.69  857.55
```

```r
#saveRDS(movie_fits, "movie_fits.rds")
rm(movie_fits)

system.time(combined_fits <- parLapply(tcl, combined_formulas, train_lm, lm_data = train_ratings))
```

```
##    user  system elapsed
##  113.08   29.62  775.92
```

```r
#saveRDS(combined_fits, "combined_fits.rds")
rm(combined_fits)

system.time(full_fits <- parLapply(tcl, full_formulas, train_lm, lm_data = train_ratings))
```

```
##    user  system elapsed
##  147.49   50.88  943.49
```

```r
#saveRDS(full_fits, "full_fits.rds")
rm(full_fits)

# Stop parallel processing
stopImplicitCluster()

# Clean up before next step
invisible(gc())
```

Final model predictions and results

```r
# Read model coeffecients and calculate RMSE

calculate_results <- function(model_formulas, test_ratings){
  str(test_ratings)

  for(m in 1:length(model_formulas)){

    coefs <- readRDS(paste0("models/", model_formulas[[m]], ".rds"))

    prediction <- rep(coefs[[1]], nrow(test_ratings)) # start with the intercept

    for(c in 2:length(coefs)){ # then go through each coefficient

      ce <- rep(coefs[[c]], nrow(test_ratings))

      prediction <- prediction + (ce * test_ratings[, names(coefs[c])]) # and add coef * value to predi

    }

    RMSE <- function(predicted_ratings, true_ratings){

      rmse <- sqrt(mean((true_ratings - predicted_ratings)^2, na.rm = TRUE))

      rmse
    }

    model_rmse <- RMSE(prediction, test_ratings$rating)
    #print(paste("Model RMSE: ", model_rmse))

    #print(paste(model_formulas[[m]], model_rmse))

    results <- data.frame(model = model_formulas[m], rmse = model_rmse)
  }

  results
}

user_results <- lapply(user_formulas, calculate_results, test_ratings = test_ratings)
```

```
## 'data.frame':    999999 obs. of  13 variables:
##  $ userId          : Factor w/ 68534 levels "1","2","3","4",..: 1 1 1 2 2 2 3 3 4 4 ...
##  $ movieId         : Factor w/ 9809 levels "1","2","3","4",..: 228 475 579 149 834 1477 583 4777 34 4
##  $ rating          : num  5 5 5 3 2 3 3.5 4.5 5 3 ...
##  $ rating_year     : num  1996 1996 1996 1997 1997 ...
##  $ user_median     : num  5 5 5 3 3 3 4 4 3 3 ...
##  $ user_mean       : num  5 5 5 2.67 2.67 ...
##  $ user_sd         : num  0 0 0 0.577 0.577 ...
##  $ movie_median    : num  3 4 3 4 5 3 4 4 4 3 ...
##  $ movie_mean      : num  2.95 3.64 3.07 3.57 4.41 ...
##  $ movie_sd        : num  1.222 0.943 0.98 0.913 0.795 ...
##  $ rating_mean_diff: num  1.488 1.488 1.488 -0.512 -1.512 ...
##  $ movie_mean_diff : num  2.047 1.356 1.925 -0.572 -2.413 ...
##  $ user_mean_diff  : num  0 0 0 0.333 -0.667 ...
```

```
## 'data.frame':    999999 obs. of  13 variables:
##  $ userId          : Factor w/ 68534 levels "1","2","3","4",..: 1 1 1 2 2 2 3 3 4 4 ...
##  $ movieId         : Factor w/ 9809 levels "1","2","3","4",..: 228 475 579 149 834 1477 583 4777 34
##  $ rating          : num  5 5 5 3 2 3 3.5 4.5 5 3 ...
##  $ rating_year     : num  1996 1996 1996 1997 1997 ...
##  $ user_median     : num  5 5 5 3 3 3 4 4 3 3 ...
##  $ user_mean       : num  5 5 5 2.67 2.67 ...
##  $ user_sd         : num  0 0 0 0.577 0.577 ...
##  $ movie_median    : num  3 4 3 4 5 3 4 4 4 3 ...
##  $ movie_mean      : num  2.95 3.64 3.07 3.57 4.41 ...
##  $ movie_sd        : num  1.222 0.943 0.98 0.913 0.795 ...
##  $ rating_mean_diff: num  1.488 1.488 1.488 -0.512 -1.512 ...
##  $ movie_mean_diff : num  2.047 1.356 1.925 -0.572 -2.413 ...
##  $ user_mean_diff  : num  0 0 0 0.333 -0.667 ...
## 'data.frame':    999999 obs. of  13 variables:
##  $ userId          : Factor w/ 68534 levels "1","2","3","4",..: 1 1 1 2 2 2 3 3 4 4 ...
##  $ movieId         : Factor w/ 9809 levels "1","2","3","4",..: 228 475 579 149 834 1477 583 4777 34
##  $ rating          : num  5 5 5 3 2 3 3.5 4.5 5 3 ...
##  $ rating_year     : num  1996 1996 1996 1997 1997 ...
##  $ user_median     : num  5 5 5 3 3 3 4 4 3 3 ...
##  $ user_mean       : num  5 5 5 2.67 2.67 ...
##  $ user_sd         : num  0 0 0 0.577 0.577 ...
##  $ movie_median    : num  3 4 3 4 5 3 4 4 4 3 ...
##  $ movie_mean      : num  2.95 3.64 3.07 3.57 4.41 ...
##  $ movie_sd        : num  1.222 0.943 0.98 0.913 0.795 ...
##  $ rating_mean_diff: num  1.488 1.488 1.488 -0.512 -1.512 ...
##  $ movie_mean_diff : num  2.047 1.356 1.925 -0.572 -2.413 ...
##  $ user_mean_diff  : num  0 0 0 0.333 -0.667 ...
```

```r
movie_results <- lapply(movie_formulas, calculate_results, test_ratings = test_ratings)
```

```
## 'data.frame':    999999 obs. of  13 variables:
##  $ userId          : Factor w/ 68534 levels "1","2","3","4",..: 1 1 1 2 2 2 3 3 4 4 ...
##  $ movieId         : Factor w/ 9809 levels "1","2","3","4",..: 228 475 579 149 834 1477 583 4777 34
##  $ rating          : num  5 5 5 3 2 3 3.5 4.5 5 3 ...
##  $ rating_year     : num  1996 1996 1996 1997 1997 ...
##  $ user_median     : num  5 5 5 3 3 3 4 4 3 3 ...
##  $ user_mean       : num  5 5 5 2.67 2.67 ...
##  $ user_sd         : num  0 0 0 0.577 0.577 ...
##  $ movie_median    : num  3 4 3 4 5 3 4 4 4 3 ...
##  $ movie_mean      : num  2.95 3.64 3.07 3.57 4.41 ...
##  $ movie_sd        : num  1.222 0.943 0.98 0.913 0.795 ...
##  $ rating_mean_diff: num  1.488 1.488 1.488 -0.512 -1.512 ...
##  $ movie_mean_diff : num  2.047 1.356 1.925 -0.572 -2.413 ...
##  $ user_mean_diff  : num  0 0 0 0.333 -0.667 ...
## 'data.frame':    999999 obs. of  13 variables:
##  $ userId          : Factor w/ 68534 levels "1","2","3","4",..: 1 1 1 2 2 2 3 3 4 4 ...
##  $ movieId         : Factor w/ 9809 levels "1","2","3","4",..: 228 475 579 149 834 1477 583 4777 34
##  $ rating          : num  5 5 5 3 2 3 3.5 4.5 5 3 ...
##  $ rating_year     : num  1996 1996 1996 1997 1997 ...
##  $ user_median     : num  5 5 5 3 3 3 4 4 3 3 ...
##  $ user_mean       : num  5 5 5 2.67 2.67 ...
##  $ user_sd         : num  0 0 0 0.577 0.577 ...
##  $ movie_median    : num  3 4 3 4 5 3 4 4 4 3 ...
```

```
##  $ movie_mean      : num  2.95 3.64 3.07 3.57 4.41 ...
##  $ movie_sd        : num  1.222 0.943 0.98 0.913 0.795 ...
##  $ rating_mean_diff: num  1.488 1.488 1.488 -0.512 -1.512 ...
##  $ movie_mean_diff : num  2.047 1.356 1.925 -0.572 -2.413 ...
##  $ user_mean_diff  : num  0 0 0 0.333 -0.667 ...
## 'data.frame':    999999 obs. of  13 variables:
##  $ userId          : Factor w/ 68534 levels "1","2","3","4",..: 1 1 1 2 2 2 3 3 4 4 ...
##  $ movieId         : Factor w/ 9809 levels "1","2","3","4",..: 228 475 579 149 834 1477 583 4777 34
##  $ rating          : num  5 5 5 3 2 3 3.5 4.5 5 3 ...
##  $ rating_year     : num  1996 1996 1996 1997 1997 ...
##  $ user_median     : num  5 5 5 3 3 3 4 4 3 3 ...
##  $ user_mean       : num  5 5 5 2.67 2.67 ...
##  $ user_sd         : num  0 0 0 0.577 0.577 ...
##  $ movie_median    : num  3 4 3 4 5 3 4 4 4 3 ...
##  $ movie_mean      : num  2.95 3.64 3.07 3.57 4.41 ...
##  $ movie_sd        : num  1.222 0.943 0.98 0.913 0.795 ...
##  $ rating_mean_diff: num  1.488 1.488 1.488 -0.512 -1.512 ...
##  $ movie_mean_diff : num  2.047 1.356 1.925 -0.572 -2.413 ...
##  $ user_mean_diff  : num  0 0 0 0.333 -0.667 ...
```

```
combined_results <- lapply(combined_formulas, calculate_results, test_ratings = test_ratings)
```

```
## 'data.frame':    999999 obs. of  13 variables:
##  $ userId          : Factor w/ 68534 levels "1","2","3","4",..: 1 1 1 2 2 2 3 3 4 4 ...
##  $ movieId         : Factor w/ 9809 levels "1","2","3","4",..: 228 475 579 149 834 1477 583 4777 34
##  $ rating          : num  5 5 5 3 2 3 3.5 4.5 5 3 ...
##  $ rating_year     : num  1996 1996 1996 1997 1997 ...
##  $ user_median     : num  5 5 5 3 3 3 4 4 3 3 ...
##  $ user_mean       : num  5 5 5 2.67 2.67 ...
##  $ user_sd         : num  0 0 0 0.577 0.577 ...
##  $ movie_median    : num  3 4 3 4 5 3 4 4 4 3 ...
##  $ movie_mean      : num  2.95 3.64 3.07 3.57 4.41 ...
##  $ movie_sd        : num  1.222 0.943 0.98 0.913 0.795 ...
##  $ rating_mean_diff: num  1.488 1.488 1.488 -0.512 -1.512 ...
##  $ movie_mean_diff : num  2.047 1.356 1.925 -0.572 -2.413 ...
##  $ user_mean_diff  : num  0 0 0 0.333 -0.667 ...
## 'data.frame':    999999 obs. of  13 variables:
##  $ userId          : Factor w/ 68534 levels "1","2","3","4",..: 1 1 1 2 2 2 3 3 4 4 ...
##  $ movieId         : Factor w/ 9809 levels "1","2","3","4",..: 228 475 579 149 834 1477 583 4777 34
##  $ rating          : num  5 5 5 3 2 3 3.5 4.5 5 3 ...
##  $ rating_year     : num  1996 1996 1996 1997 1997 ...
##  $ user_median     : num  5 5 5 3 3 3 4 4 3 3 ...
##  $ user_mean       : num  5 5 5 2.67 2.67 ...
##  $ user_sd         : num  0 0 0 0.577 0.577 ...
##  $ movie_median    : num  3 4 3 4 5 3 4 4 4 3 ...
##  $ movie_mean      : num  2.95 3.64 3.07 3.57 4.41 ...
##  $ movie_sd        : num  1.222 0.943 0.98 0.913 0.795 ...
##  $ rating_mean_diff: num  1.488 1.488 1.488 -0.512 -1.512 ...
##  $ movie_mean_diff : num  2.047 1.356 1.925 -0.572 -2.413 ...
##  $ user_mean_diff  : num  0 0 0 0.333 -0.667 ...
```

```
full_results <- lapply(full_formulas, calculate_results, test_ratings = test_ratings)
```

```
## 'data.frame':    999999 obs. of  13 variables:
##  $ userId          : Factor w/ 68534 levels "1","2","3","4",..: 1 1 1 2 2 2 3 3 4 4 ...
##  $ movieId         : Factor w/ 9809 levels "1","2","3","4",..: 228 475 579 149 834 1477 583 4777 34
##  $ rating          : num  5 5 5 3 2 3 3.5 4.5 5 3 ...
##  $ rating_year     : num  1996 1996 1996 1997 1997 ...
##  $ user_median     : num  5 5 5 3 3 3 4 4 3 3 ...
##  $ user_mean       : num  5 5 5 2.67 2.67 ...
##  $ user_sd         : num  0 0 0 0.577 0.577 ...
##  $ movie_median    : num  3 4 3 4 5 3 4 4 4 3 ...
##  $ movie_mean      : num  2.95 3.64 3.07 3.57 4.41 ...
##  $ movie_sd        : num  1.222 0.943 0.98 0.913 0.795 ...
##  $ rating_mean_diff: num  1.488 1.488 1.488 -0.512 -1.512 ...
##  $ movie_mean_diff : num  2.047 1.356 1.925 -0.572 -2.413 ...
##  $ user_mean_diff  : num  0 0 0 0.333 -0.667 ...
## 'data.frame':    999999 obs. of  13 variables:
##  $ userId          : Factor w/ 68534 levels "1","2","3","4",..: 1 1 1 2 2 2 3 3 4 4 ...
##  $ movieId         : Factor w/ 9809 levels "1","2","3","4",..: 228 475 579 149 834 1477 583 4777 34
##  $ rating          : num  5 5 5 3 2 3 3.5 4.5 5 3 ...
##  $ rating_year     : num  1996 1996 1996 1997 1997 ...
##  $ user_median     : num  5 5 5 3 3 3 4 4 3 3 ...
##  $ user_mean       : num  5 5 5 2.67 2.67 ...
##  $ user_sd         : num  0 0 0 0.577 0.577 ...
##  $ movie_median    : num  3 4 3 4 5 3 4 4 4 3 ...
##  $ movie_mean      : num  2.95 3.64 3.07 3.57 4.41 ...
##  $ movie_sd        : num  1.222 0.943 0.98 0.913 0.795 ...
##  $ rating_mean_diff: num  1.488 1.488 1.488 -0.512 -1.512 ...
##  $ movie_mean_diff : num  2.047 1.356 1.925 -0.572 -2.413 ...
##  $ user_mean_diff  : num  0 0 0 0.333 -0.667 ...
```

```r
# Save results locally and print
saveRDS(user_results, "user_results.rds")
saveRDS(movie_results, "movie_results.rds")
saveRDS(combined_results, "combined_results.rds")
saveRDS(full_results, "full_results.rds")

# Lets have a look
lm_results <- bind_rows(c(user_results, movie_results, combined_results, full_results))
lm_results %>% arrange(rmse)
```

```
##                                                                                model
## 1                        rating ~ movie_median + movie_mean + user_median + user_mean
## 2                                                        rating ~ movie_mean + user_mean
## 3  rating ~ movie_median + movie_mean + movie_sd + user_median + user_mean + user_sd
## 4                                                    rating ~ movie_median + user_median
## 5                                           rating ~ movie_median + movie_mean + movie_sd
## 6                                                       rating ~ user_median + user_mean
## 7                                              rating ~ user_median + user_mean + user_sd
## 8                                                                  rating ~ movie_median
## 9                                                                   rating ~ user_median
## 10                                                    rating ~ movie_median + movie_mean
##         rmse
## 1  0.8452112
## 2  0.8452248
## 3  0.8467370
```

```
## 4  0.8843992
## 5  0.9386363
## 6  0.9395215
## 7  0.9413628
## 8  0.9578211
## 9  0.9679871
## 10        NA
```

```r
saveRDS(lm_results, "lm_results.rds")

# Clean up before next step
invisible(gc())
```

Automatic Machine Learning with H2O.ai

```r
# Shall we explore auto ML?

library(h2o)

h2o.init()
```

```
##
## H2O is not running yet, starting it now...
##
## Note:  In case of errors look at the following log files:
##     C:\Users\Codrin\AppData\Local\Temp\Rtmpiuz0Yl/h2o_Codrin_started_from_r.out
##     C:\Users\Codrin\AppData\Local\Temp\Rtmpiuz0Yl/h2o_Codrin_started_from_r.err
##
##
## Starting H2O JVM and connecting:  Connection successful!
##
## R is connected to the H2O cluster:
##     H2O cluster uptime:         1 seconds 527 milliseconds
##     H2O cluster timezone:       Europe/Berlin
##     H2O data parsing timezone:  UTC
##     H2O cluster version:        3.24.0.3
##     H2O cluster version age:    24 days
##     H2O cluster name:           H2O_started_from_R_Codrin_zsw042
##     H2O cluster total nodes:    1
##     H2O cluster total memory:   14.21 GB
##     H2O cluster total cores:    12
##     H2O cluster allowed cores:  12
##     H2O cluster healthy:        TRUE
##     H2O Connection ip:          localhost
##     H2O Connection port:        54321
##     H2O Connection proxy:       NA
##     H2O Internal Security:      FALSE
##     H2O API Extensions:         Amazon S3, Algos, AutoML, Core V3, Core V4
##     R Version:                  R version 3.5.0 (2018-04-23)
```

```r
h2o.no_progress()

# Import train/test set into H2O
train <- as.h2o(readRDS("edx.RData"))
test <- as.h2o(readRDS("validation.RData"))

# Identify predictors and response
y <- "rating"
x <- setdiff(names(train), y)

# Run AutoML for 10 base models (limited to 1 hour max runtime by default)
print(Sys.time())
```

```
## [1] "2019-05-31 22:12:39 CEST"
```

```r
aml <- h2o.automl(x = x, y = y,
                  training_frame = train,
                  validation_frame = test,
                  max_models = 10,
                  seed = 1,
                  stopping_metric = "RMSE",
                  sort_metric = "RMSE")

# View the AutoML Leaderboard
lb <- aml@leaderboard
autoML_results <- as.data.frame(lb) %>% select(model_id, rmse)

saveRDS(autoML_results, "autoML_results.rds")
print(autoML_results)
```

```
##                                                 model_id      rmse
## 1     StackedEnsemble_AllModels_AutoML_20190531_221239 0.9637109
## 2  StackedEnsemble_BestOfFamily_AutoML_20190531_221239 0.9638613
## 3                           GBM_5_AutoML_20190531_221239 0.9763070
## 4                           DRF_1_AutoML_20190531_221239 0.9889825
## 5                           XRT_1_AutoML_20190531_221239 0.9930478
## 6                           GBM_4_AutoML_20190531_221239 1.0087012
## 7                           GBM_3_AutoML_20190531_221239 1.0196827
## 8                           GBM_2_AutoML_20190531_221239 1.0240631
## 9                           GBM_1_AutoML_20190531_221239 1.0277414
## 10                 DeepLearning_1_AutoML_20190531_221239 1.0471039
## 11            GBM_grid_1_AutoML_20190531_221239_model_1 1.0569208
## 12            GLM_grid_1_AutoML_20190531_221239_model_1 1.0596654
```

```r
# How long did the whole script take?
script_end <- Sys.time()

print(paste("Total script running time: ", round(difftime(script_end, script_start, units = "mins"), 1)
```

```
## [1] "Total script running time:  118.8  minutes"
```

Extended automatic Machine Learning with H2O.ai

```r
# What will 10 hours of computation yield?

library(h2o)

h2o.init()
h2o.show_progress()

# Import train/test set into H2O
train <- as.h2o(readRDS("edx.RData"))
test <- as.h2o(readRDS("validation.RData"))

# Identify predictors and response
y <- "rating"
x <- setdiff(names(train), y)

# Run AutoML for 25 models (limited to 10 hour max runtime)
print(Sys.time())
xaml <- h2o.automl(x = x, y = y,
                  training_frame = train,
                  validation_frame = test,
                  max_models = 25,
                  seed = 1,
                  stopping_metric = "RMSE",
                  sort_metric = "RMSE",
                  max_runtime_secs = 36000)

# View the AutoML Leaderboard
xlb <- xaml@leaderboard
xautoML_results <- as.data.frame(xlb) %>% select(model_id, rmse)

saveRDS(xautoML_results, "xautoML_results.rds")
print(xautoML_results)
```