Movielens data project

Jose Arizpe

31 December 2019

Executive Summary

The purpose of this project is to create a machine learning algorithm using the inputs of the "edx" dataset to predict the movie ratings of the "validation" dataset. The first part of the code is copied from the project instructions in order to create these two datasets.

The movielens data contains movie ratings issued by specific users on specific movies. In fact, each record contains the following information about a movie rating: "userId", "movieId", "rating", "timestamp", "title" and "genres".

The algorithm implemented here starts from the algorithm implemented for Movielens in the Data Science course, which defines prediction estimates as calculated through the following formula:

$\mathrm{Y}_{\mathrm{u,i}} = \mu + \mathrm{b}_{\mathrm{i}} + \mathrm{b}_{\mathrm{u}} + \varepsilon_{\mathrm{u,i}}$

where

- $\mathrm{Y}_{\mathrm{u,i}}$ - is the prediction for movie i and user u
- μ - is the mean of ratings
- b_{i} - is the movie effect
- b_{u} - is the user effect
- $\varepsilon_{\mathrm{u,i}}$ - is the residual error after taking into account the effects above

Apart from the method implemented in the course materials, this algorithm adds regularisation through penalised least squares and an additional specific model for fitting residuals using "rpart", which yields an RMSE of 0.42525.

Method / Analysis

The model used for fitting the Movielens data follows the methodology used in the course for the same purpose, based on producing predictions based on the following formula:

$\mathrm{Y}_{\mathrm{u,i}} = \mu + \mathrm{b}_{\mathrm{i}} + \mathrm{b}_{\mathrm{u}} + \varepsilon_{\mathrm{u,i}}$

where

- $\mathrm{Y}_{\mathrm{u,i}}$ - is the prediction for movie i and user u
- μ - is the mean of ratings
- b_{i} - is the movie effect
- b_{u} - is the user effect
- $\varepsilon_{\mathrm{u,i}}$ - is the residual error after taking into account the effects above

The course methodology stops with the calculation of movie and user effects (b_{i} and b_{u}), and gives an overview of regularisation. The present process adds the following enhancements & tests:

1. Introduction of regularisation based on penalised least squares based on the methodology introduced in the course material.
2. Use of the "rpart" [1] tree-basaed model for fitting the data from various angles:
   i) Estimate ratings
   ii) Estimate residuals $\varepsilon_{\mathrm{u,i}}$ using the original data
   iii) Estimate residuals $\varepsilon_{\mathrm{u,i}}$ using an augmented version of the trainig data that includes the addition of the followong columns (which were added on the rationale that additional columns can enhance the result of a tree-based approach, premise that was not confirmed by the results):
      A. Break-up of the movielens "genres" field into columns, one column per genre containing logical values
      B. Extract the movie year data from the movie title and put it in an independent column

The highest improvement on RMSE is obtained with the fitting of the residuals $\varepsilon_{\mathrm{u,i}}$ using "rpart". The application of regularisation based on penalised least squares also improves RMSE but in a much more modest way than the fitting of residuals.

`The resulting RMSE (0.42525) can be seen at the bottom of the report.`

Below are the steps in the process followed in detail:

1. Load and prepare data

Load data using the code provided in the project materials

Create edx and validation sets

. . . .

Extract the year from the movie title

```r
# ----------------------------------------------------------------------
# Augment data - Create date field
#
edx <- edx %>% mutate(date = as_datetime(timestamp))
# EXTRACT YEAR AND TITLE FROM ORIGINAL "title" COLUMN
#    Extract the year into vector "y"
y <- str_match(edx$title, "^(.*)\\s\\((\\d{4})\\)$")[,3]
edx$year <- y ## Attach "y" as a new column to "edx"
#    Eliminate year data from title
movieTitles <- str_match(edx$title, "^(.*)\\s\\((\\d{4})\\)$")[,2]
edx$title <- movieTitles ## Replace title with the stripped down new title
```

Break-up the "Genres" column

Into as many columns as there are defined genres with logical values for each movie highligthing the genres which those movies have been classsified against.

```r
# ----------------------------------------------------------------------
# Augment data - split genres into columns
#
#   Create new data frame including only "movieId" and "genres"
movieGenres <- edx %>% select(movieId, genres) %>% unique()
```

[1] With regards to the use of "rpart" and not other fitting methods, I have tried various but all of them end up running out of memory except "rpart". I have sticked with "rpart" because of the good results produced and have left experimentation with other methods to a later stage when I have access to more computing resources.

```
movieGenres <- suppressWarnings(movieGenres %>% separate("genres", c("g1", "g2", "g3",
                                                                       "g4", "g5", "g6",
                                                                       "g7", "g8", "g9",
                                                                       "g10"), "\\|"))
#   separate genres into different columns, assume no more than 10 genres per movie.
#   Create 1-n table for "movieId" and "genre". Use intermediate table t
t <- gather(movieGenres, "key", "genre", -movieId, na.rm = TRUE)
allGenres <- t %>% select(genre) %>% unique() ## List all different genres found
#   Delete records with text in parenthesis, assuming these are not valid genre names
allGenres <- allGenres %>% filter(!str_detect(allGenres$genre, "^\\(.*\\)$"))
#   Create colums with genres
for (g in allGenres[,1]) {
  edx <- cbind(edx, data.frame(nom = str_detect(edx$genres, g)))
  names(edx)[names(edx) == "nom"] <- str_replace(tolower(g), "-", "_")
}
```

(Note that this processing makes the assumption that no movie is identified with more than 10 genres. Also it assumes that genres in parentheses are not genre names but rather it means no genre has been identified.)

Repeat the previous steps for the "validation" set

. . . .

2. Create the model

Create the model based on the following formula:

$Y_{u,i} = \mu + b_i + b_u + \varepsilon_{u,i}$

Estimate μ, Naive approach

Start with naive approach as initial benchmark to measure progress against.

```
# ----------------------------------------------------------------------
# Naïve RSME
#
mu_hat <- mean(edx$rating)
naive_rmse <- RMSE(validation$rating, mu_hat)
```

```
## RMSE naive method = 1.06120181029262
```

Try rpart as overarching method for predicting ratings

Before estimating the remaining variables in the prediction formula (b_i, b_u and $\varepsilon_{u,i}$) try fitting the ratings using "rpart", again to use as a benchmark against the method that we are following here.

```
# ----------------------------------------------------------------------
# Predict ratings with rpart
#
set.seed(1)
library(rpart) # Fit ratings using "rpart"
fit <- rpart(rating~., data = edx)
predicted_ratings <- predict(fit, validation)
model_0_rmse <- RMSE(predicted_ratings, validation$rating)
```

```
## RMSE fit ratings using rpart = 0.957880109409465
```

As can be seen, initial results with rpart for directly predicting the end-ratings are not great and do not achieve many points, so we will continue estimating the elements of the formula:

$(\mathrm{Y}_{\mathrm{u,i}} = \mu + \mathrm{b}_{\mathrm{i}} + \mathrm{b}_{\mathrm{u}} + \varepsilon_{\mathrm{u,i}})$

Add movie effects ($\mathbf{b_i}$) to naive estimate (μ):

```r
# ----------------------------------------------------------------------
# Movie effects
#
mu <- mean(edx$rating)
#    Calculate movie averages
movie_avgs <- edx %>% group_by(movieId) %>% summarize(b_i = mean(rating - mu))
#    Apply movie averages to estimate
predicted_ratings <- mu + validation %>% left_join(movie_avgs, by='movieId') %>%
  pull(b_i)
model_1_rmse <- RMSE(predicted_ratings, validation$rating)
```

```
## RMSE from adding movie effect to naive = 0.943908662806309
```

Add user effects ($\mathbf{b_u}$) to naive estimate (μ) + movie effects ($\mathbf{b_i}$):

```r
# ----------------------------------------------------------------------
# Add user effects
#
#    Calculate user averages
user_avgs <- edx %>% left_join(movie_avgs, by='movieId') %>% group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_i))
#    Apply user averages to estimate
predicted_ratings <- validation %>% left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>% mutate(pred = mu + b_i + b_u) %>% pull(pred)
model_2_rmse <- RMSE(predicted_ratings, validation$rating)
```

```
## RMSE from adding movie effect and user effect to naive = 0.865348824577316
```

3. Enhance the model

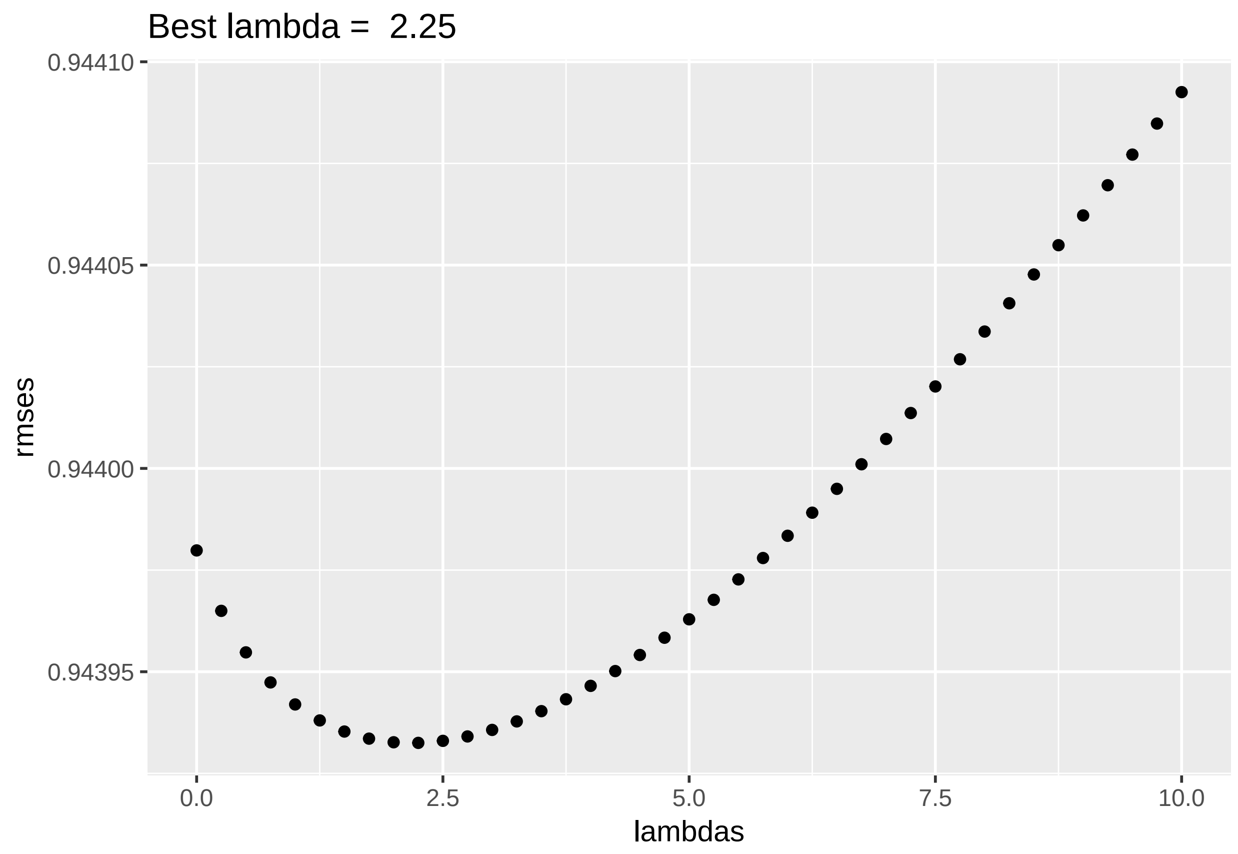
Apply regularisation to the naive (μ) + movie effects ($\mathbf{b_i}$) estimate

Apply regularisation based on penalised least squares, and see if this improves the results. For this the data has to be partitioned in order to fin the lambda value that yields the lowest RMSE:

```r
# ----------------------------------------------------------------------
# Movie effect with regularisation
#
# Partition data
set.seed(1)
test_index <- createDataPartition(edx$rating, times=1, p=0.9, list=FALSE)
edx_test <- edx[-test_index,]
edx_train <- edx[test_index,]
edx_test <- edx_test %>%
  semi_join(edx_train, by = "movieId") %>%
  semi_join(edx_train, by = "userId")

# Determine the better lambda based on lowest RMSE
lambdas <- seq(0, 10, 0.25)
mu <- mean(edx_train$rating)
```

```r
just_the_sum <- edx_train %>% group_by(movieId) %>%
  summarize(s = sum(rating - mu), n_i = n())
rmses <- sapply(lambdas, function(l1){
  predicted_ratings <- edx_test %>% left_join(just_the_sum, by='movieId') %>%
    mutate(b_i = s/(n_i+l1)) %>% mutate(pred = mu + b_i) %>% pull(pred)
  return(RMSE(predicted_ratings, edx_test$rating))
})
l1 <- lambdas[which.min(rmses)]
qplot(lambdas, rmses, main = paste("Best lambda = ", l1))
```



```r
# Calculate RMSE for better lambda
mu <- mean(edx$rating)
just_the_sum <- edx %>% group_by(movieId) %>%
  summarize(s = sum(rating - mu), n_i = n())
predicted_ratings <- validation %>%
  left_join(just_the_sum, by='movieId') %>% mutate(b_i = s/(n_i+l1)) %>%
  mutate(pred = mu + b_i) %>% pull(pred)
model_6_rmse <- RMSE(predicted_ratings, validation$rating)
```

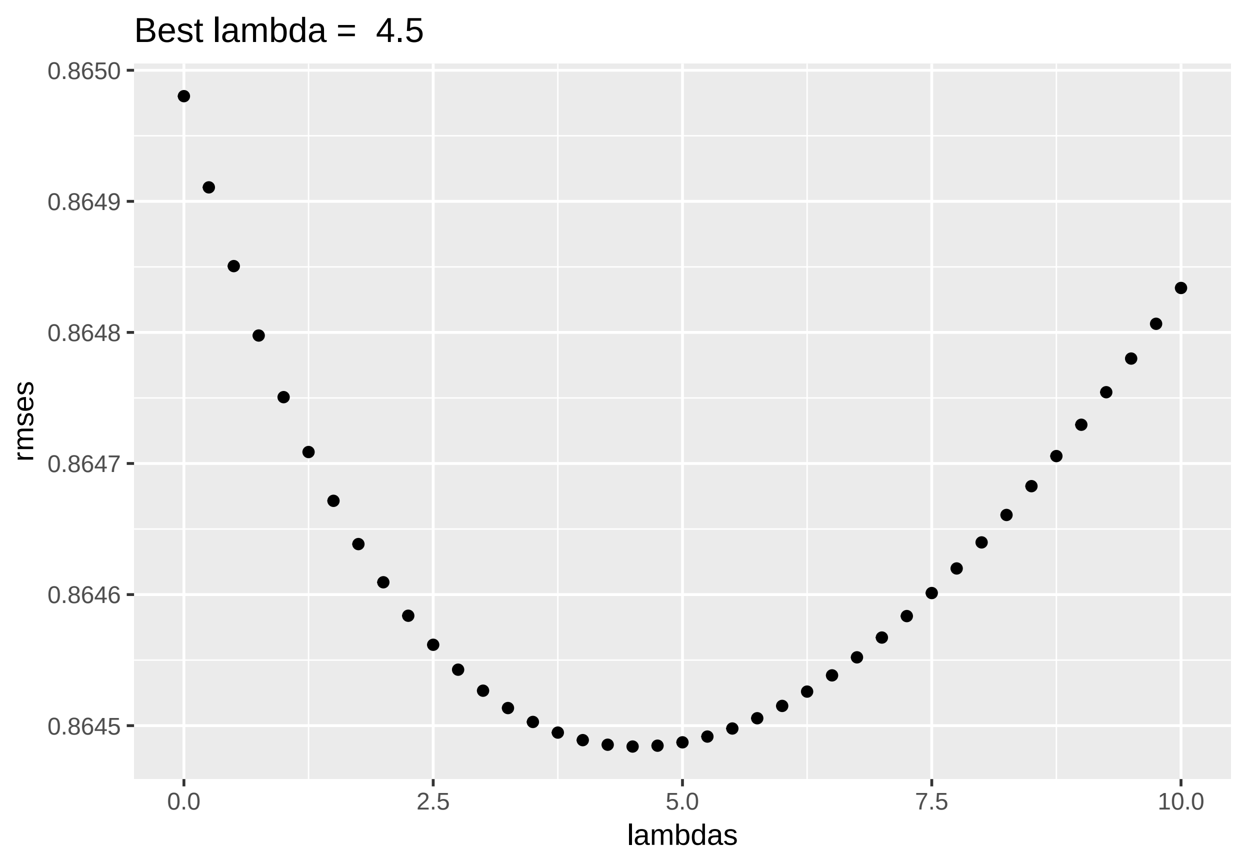
```
## RMSE from naive + reg'd movie effect = 0.943852139888902
```

Apply regularisation to the naive (μ) + movie effects ($\mathbf{b_i}$) + user effects ($\mathbf{b_u}$) estimate

Apply regularisation based on penalised least squares, and see if this improves the results. For this the data has to be partitioned in order to fin the lambda value that yields the lowest RMSE:

```r
# ------------------------------------------------------------------------
# Movie and user effect with regularisation
#
# Partition data
set.seed(1)
test_index <- createDataPartition(edx$rating, times=1, p=0.9, list=FALSE)
edx_test <- edx[-test_index,]
edx_train <- edx[test_index,]
edx_test <- edx_test %>%
  semi_join(edx_train, by = "movieId") %>%
  semi_join(edx_train, by = "userId")

# Determine the better lambda based on lowest RMSE
lambdas <- seq(0, 10, 0.25)
rmses <- sapply(lambdas, function(l2){
  mu <- mean(edx_train$rating)
  b_i <- edx_train %>% group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n()+l2))
  b_u <- edx_train %>% left_join(b_i, by="movieId") %>%
    group_by(userId) %>% summarize(b_u = sum(rating - b_i - mu)/(n()+l2))
  predicted_ratings <- edx_test %>% left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>% mutate(pred = mu + b_i + b_u) %>%
    pull(pred)
  return(RMSE(predicted_ratings, edx_test$rating))
})
l2 <- lambdas[which.min(rmses)]
qplot(lambdas, rmses, main = paste("Best lambda = ", l2))
```



```
# Calculate RMSE for better lambda
mu <- mean(edx$rating)
b_i <- edx %>% group_by(movieId) %>%
  summarize(b_i = sum(rating - mu)/(n()+l2))
b_u <- edx %>% left_join(b_i, by="movieId") %>%
  group_by(userId) %>% summarize(b_u = sum(rating - b_i - mu)/(n()+l2))
predicted_ratings <- validation %>% left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = "userId") %>% mutate(pred = mu + b_i + b_u) %>%
  pull(pred)
model_3_rmse <- RMSE(predicted_ratings, validation$rating)
```

```
## RMSE from naive + reg'd movie effect + user effect = 0.8648242289282
```

Model the residuals ($\varepsilon_{\mathbf{u,i}}$) using "rpart"

In order to understand the effect of the new columns added.

Code and RMSE results follow:

```
# ----------------------------------------------------------------------
# MODEL RESIDUALS USING rpart
#
# First identify the winning parameters for all calculations to follow
#
mu <- mean(edxCopy$rating)
b_i <- edxCopy %>% group_by(movieId) %>% summarize(b_i = sum(rating - mu)/(n()+l2))
b_u <- edxCopy %>% left_join(b_i, by="movieId") %>% group_by(userId) %>%
  summarize(b_u = sum(rating - b_i - mu)/(n()+l2))
```

```
# Create residuals field by subtracting mu, b_i and b_u from each rating
residuals <- edxCopy %>% left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = "userId") %>% mutate(residual = rating - mu - b_i - b_u) %>%
  pull(residual)
edxCopy <- bind_cols(edxCopy, data.frame(residuals = residuals))
library(rpart)
fit <- rpart(residuals~., data = edxCopy) # Fit residuals using "rpart"
r_hat <- predict(fit, validationCopy) # Estimate residuals for validation set
validationCopy <- bind_cols(validationCopy, data.frame(residuals = r_hat))

# Calculate predicted ratings using mu + b_i + b_u + residuals, i.e. add residuals
predicted_ratings <- validationCopy %>% left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = "userId") %>% mutate(pred = mu + b_i + b_u + residuals) %>%
  pull(pred)
model_7_rmse <- RMSE(predicted_ratings, validationCopy$rating)
```

```
## RMSE from naive + reg'd movie & user effects + rpart residuals fit = 0.425251409677095
```

Model the residuals ($\varepsilon_{\mathbf{u,i}}$) using "rpart" on the augmented dataset

Use the complete data breadth, initial columns + calculated year + additional genre columns created above.

Code and RMSE results follow:

```
# ----------------------------------------------------------------------
# MODEL RESIDUALS USING rpart ON AUGMENTED DATASET
#
# First identify the winning parameters for all calculations to follow
#
mu <- mean(edx$rating)
b_i <- edx %>% group_by(movieId) %>% summarize(b_i = sum(rating - mu)/(n()+l2))
b_u <- edx %>% left_join(b_i, by="movieId") %>% group_by(userId) %>%
  summarize(b_u = sum(rating - b_i - mu)/(n()+l2))
# Create residuals field by subtracting mu, b_i and b_u from each rating
residuals <- edx %>% left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = "userId") %>% mutate(residual = rating - mu - b_i - b_u) %>%
  pull(residual)
edx <- bind_cols(edx, data.frame(residuals = residuals))
library(rpart)
fit <- rpart(residuals~., data = edx) # Fit residuals using "rpart"
r_hat <- predict(fit, validation) # Estimate residuals for validation set
validation <- bind_cols(validation, data.frame(residuals = r_hat))

# Calculate predicted ratings using mu + b_i + b_u + residuals, i.e. add residuals
predicted_ratings <- validation %>% left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = "userId") %>% mutate(pred = mu + b_i + b_u + residuals) %>%
  pull(pred)
model_4_rmse <- RMSE(predicted_ratings, validation$rating)
```

```
## RMSE from naive + reg'd movie & user effects + rpart residuals fit on augmented data =

## 0.426187656223384
```

Model the residuals ($\varepsilon_{\mathbf{u,i}}$) without previous regularisation on the augmented dataset

Just to be sure, and taking into account that the RMSE improves very little with Regularisation, model the residuals same as in the previus step but using the non-regularised naive + movie + user effects results.

```r
# ----------------------------------------------------------------------
# MODEL RESIDUALS USING rpart ON AUGMENTED DATASET
#
# First identify the winning parameters for all calculations to follow
#
mu <- mean(edx$rating)
b_i <- edx %>% group_by(movieId) %>% summarize(b_i = mean(rating - mu))
b_u <- edx %>% left_join(b_i, by="movieId") %>% group_by(userId) %>%
  summarize(b_u = mean(rating - b_i - mu))
# Create residuals field by subtracting mu, b_i and b_u from each rating
residuals <- edx %>% left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = "userId") %>% mutate(residual = rating - mu - b_i - b_u) %>%
  pull(residual)
edx <- edx[ ,-which(names(edx)=="residuals")]
edx <- bind_cols(edx, data.frame(residuals = residuals))
fit <- rpart(residuals~., data = edx) # Fit residuals using "rpart"
r_hat <- predict(fit, validation) # Estimate residuals for validation set
validation <- validation[ ,-which(names(validation)=="residuals")]
validation <- bind_cols(validation, data.frame(residuals = r_hat))

# Calculate predicted ratings using mu + b_i + b_u + residuals, i.e. add residuals
predicted_ratings <- validation %>% left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = "userId") %>% mutate(pred = mu + b_i + b_u + residuals) %>%
  pull(pred)
model_5_rmse <- RMSE(predicted_ratings, validation$rating)
```

```
## RMSE from naive + movie & user effects + rpart residuals fit on augmented data =
```

```
## 0.436926766613497
```

Results

The following table shows the results (RMSEs) from executing the steps shown above:

```
##                                                         method      rmse
## 1            Naive+reg'd movie&user effects+rpart residuals fit 0.4252514
## 2 Naive+reg'd movie&user effects+rpart residuals fit augm.data 0.4261877
## 3       Naive+movie&user effects+rpart residuals fit augm.data 0.4369268
## 4                               Naive+reg'd movie&user effects 0.8648242
## 5                                     Naive+movie&user effects 0.8653488
## 6                                    Naive+reg'd movie effects 0.9438521
## 7                                          Naive+movie effects 0.9439087
## 8                                            rpart ratings fit 0.9578801
## 9                                                        Naive 1.0612018
```

The results show that the step improvement on the RSME comes with the specific modelling of the residuals with "rpart". Regularisation provides a small improvement and the genres variable break-up actually worsens the results. As part of the analysis, I have also tested eliminating the "genres" column after break-up but there is no improvement in doing that.

On this note, the final and lowest RMSE is 0.4252514, which is delivered by Naive+reg'd movie&user effects+rpart residuals fit.

Conclusion

This report has presented a model for predicting the movie ratings of the "validation" dataset using the inputs of the "edx" dataset, both of which are extracts from the 10m record Movielens data.

The model is based on the calculation of predictions using the formula $\mathrm{Y_{u,i}} = \mu + \mathrm{b_i} + \mathrm{b_u} + \varepsilon_{\mathrm{u,i}}$. It covers procedures for estimating each of the variables in this formula, some presented in the Data Science course material and some enhancements introduced on top of that.

Of the enhancements introduced, the one that provides the step change in RMSE is the modelling of the residuals with "rpart", which delivers an overall RMSE of 0.4252514.

The method presented here is limited by the computing resources that are available for this project. This limitation has meant that the only model that has not run into memory problems is "rpart", which is the reason it is use extensively in this project, without prejudice to the good performance boost it provides. There remains a future effort to take this code to a more powerful environment where I can use and test other methods.