Data Science: Capstone - Movielens project

Introduction

The Movielens 10M dataset is a dataset of movie recommendations provided by the grouplens lab of the University of Minnesota. According to its readme file it consists of 10000054 ratings of 10681 movies by 71567 users from the online movie recommender service MovieLens. User reviews are anonymized. The user is represented by an identifier (ID) and review submission is time-stamped. Movies are represented by a unique movieID, their title together with the year of publication, and one or more different genre tags. Ratings are given on a scale from 0 to 5 stars in 0.5 star increments. In the present work, a generalized linear model combined with elastic net regularization was used to predict movie ratings. The final model was trained on 90 % of the data with 5-fold cross-validation. It uses bias factors taking characteristics of the movie (year of publication, genre and individual movie bias) and of the user (numbers of years the review was submitted, individual user bias, submission week of review) into account, as well as the total number of ratings a movie has received and the temporal order in which a user submits the reviews. The proposed model yields a RMSE of 0.86452 on an independent validation dataset.

Analysis

This section describes the preparation of the dataset and initial exploratory analysis to identify baseline predictors for the modeling approach.

Dataset preparation

The Movielens 10M dataset is downloaded and partitioned into evaluation and validation data. The prediction method is developed based only on the evaluation data. To optimize the prediction method, the training data is further partitioned into a training and training-test data set (80 % training, 20 % test). The final prediction method is re-trained on the full evaluation dataset prior to validation.

Download and partition of Movielens 10M Dataset

The Movielens 10M dataset is partioned into a dataset for evaluation *edx* (90 % of the data) and a final hold-out dataset for validation *validation* (10 % of the data). The procedure is performed as defined in the course instructions. *edx* and *valiation* are exported to csv files.

Data pre-processing

Inspection of the evaluation data shows that the publication year of the movie can be extracted from the title using the *stringr* package. Extraction is realized in two parts to safely distinguish the publication year given in brackets from any other four-digit number in the movie title. The publication year is stored in *pub_year*. The timestamp of the user review is converted to a date and stored in *review_date*. The new dataset is called *edx_work*. All operations need to be applied to the *validation* data set prior to validation of the final model.

Prior to pre-processing the dataset looks like this:

```
## # A tibble: 6 x 6
##   userId movieId rating timestamp title                    genres
##    <dbl>   <dbl>  <dbl>     <dbl> <chr>                    <chr>
## 1      1     122      5 838985046 Boomerang (1992)         Comedy|Romance
## 2      1     185      5 838983525 Net, The (1995)          Action|Crime|Thriller
## 3      1     292      5 838983421 Outbreak (1995)          Action|Drama|Sci-Fi|T~
## 4      1     316      5 838983392 Stargate (1994)          Action|Adventure|Sci-~
## 5      1     329      5 838983392 Star Trek: Generations~ Action|Adventure|Dram~
## 6      1     355      5 838984474 Flintstones, The (1994) Children|Comedy|Fanta~
```

After pre-processing it looks like this:

```
## # A tibble: 6 x 8
##   userId movieId rating timestamp title    genres   review_date         pub_year
##    <dbl>   <dbl>  <dbl>     <dbl> <chr>    <chr>    <dttm>                 <dbl>
## 1      1     122      5 838985046 Boomera~ Comedy|~ 1996-08-02 11:24:06     1992
## 2      1     185      5 838983525 Net, Th~ Action|~ 1996-08-02 10:58:45     1995
## 3      1     292      5 838983421 Outbrea~ Action|~ 1996-08-02 10:57:01     1995
## 4      1     316      5 838983392 Stargat~ Action|~ 1996-08-02 10:56:32     1994
## 5      1     329      5 838983392 Star Tr~ Action|~ 1996-08-02 10:56:32     1994
## 6      1     355      5 838984474 Flintst~ Childre~ 1996-08-02 11:14:34     1994
```

Partition of evaluation dataset

The *edx_work* dataset is further divided into sets *edx_work_test* (20%) and *edx_work_train* (80%) using the procedure provided for defining *edx* and *validation* data sets. The following data exploration and intermediate model training is based on *edx_work_train*.

Data exploration

In the *edx_work_train* data set, there are 10677 different movies reviewed by 69878 users. The average rating is approx. 3.51 with a standard deviation of approx. 1.06. Average and standard deviation are used as reference to assess the effect of different variables on the rating of a movie.

Movie effects

In this section, base line predictors are explored that are associated with a specific movie.

Publication year of movie The mean movie rating per publication year is calculated.

The dataset *edx_work train* includes movies published from 1915 to 2008. Fig. 1 shows the number of ratings per publication year. While the number of ratings for movies published before 1950 is low, it steadily increases afterwards with a maximum in 1995 indicating a need for regularization when this bias is included into a model. The average rating of a movie with a publication year earlier than 1984 is above average, while afterwards it approaches the overall average (Fig. 2, red line).

```
fig_1
```

```
fig_2
```

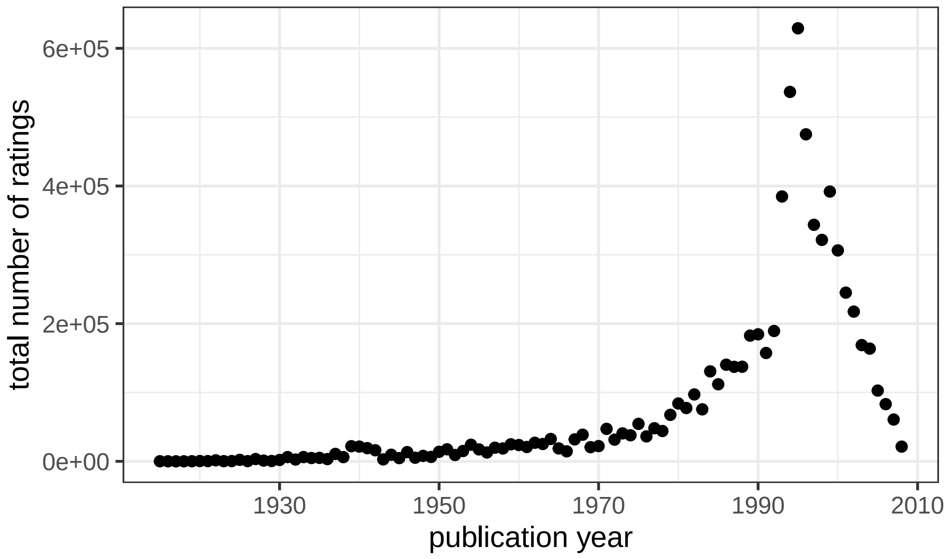

Figure 1: number of ratings per publication year

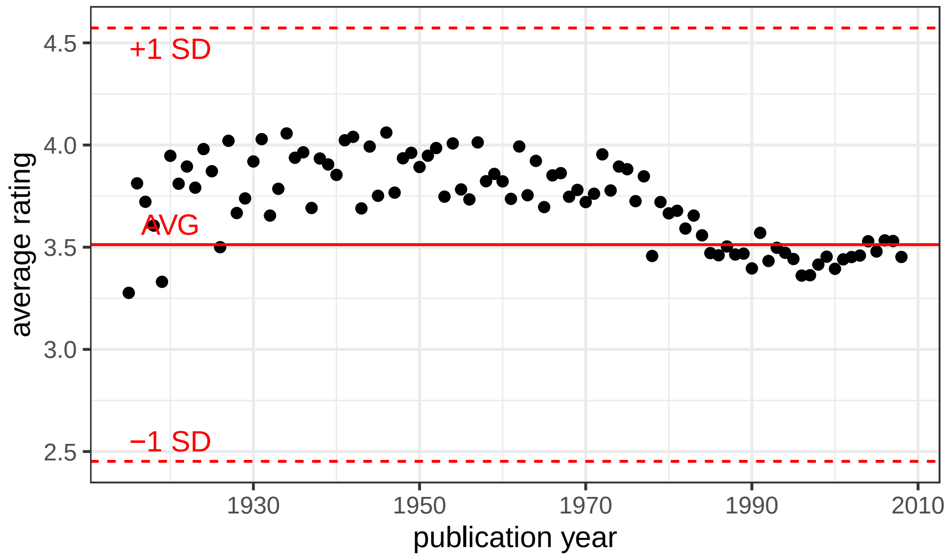

Figure 2: average rating per publication year

Movie genre The dataset *edx_work train* contains 10677 movies of 797 genres and sub-genres. The number of ratings are not evenly distributed (Fig. 3) indicating a need for regularization when this bias is included in to a model. The average rating per genre/sub-genre is shown in Fig. 4 demonstrating an effect on the movie rating.

```
fig_3
```

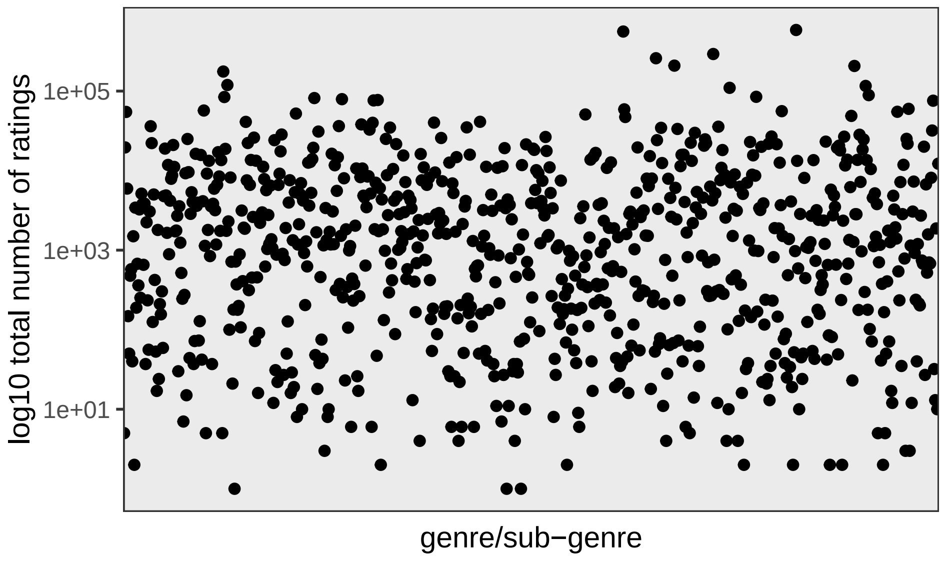

Figure 3: number of ratings per genre/sub-genre

```
fig_4
```

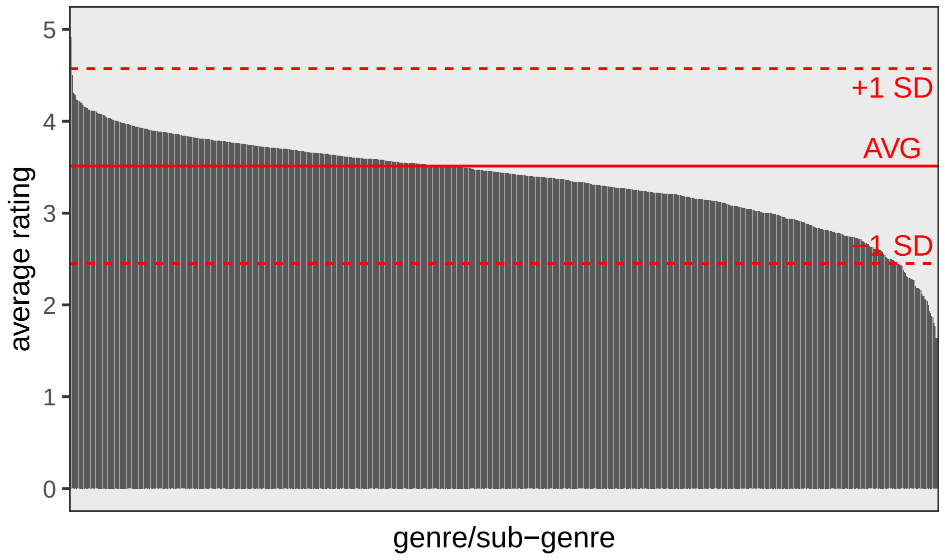

Figure 4: average rating by genre/sub-genre

Number of ratings Next, the number of ratings per movie was investigated whether it affects the movie rating. For this, the number of distinct user ID associated to a movie ID was counted and stratified to multiples of hundred. The number of ratings per group are shown in Fig. 5. Fig. 6 demonstrates that the average rating generally increases with an increasing number of ratings.

```
fig_5
```

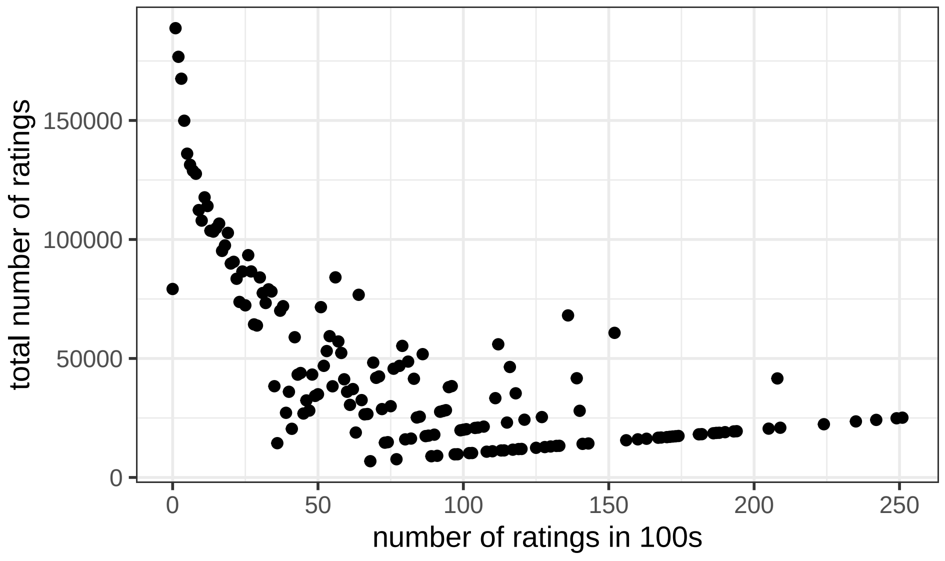

Figure 5: number of ratings per number of ratings in 100s

```
fig_6
```

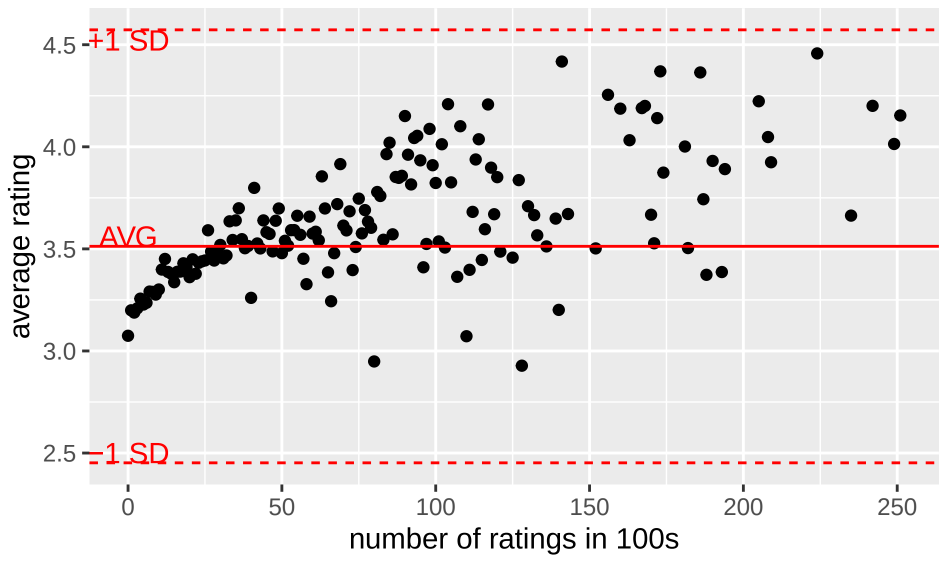

Figure 6: average rating by number of ratings in 100s

User effects

In this section, base line predictors are explored that are associated with a specific user or user review.

Number of reviews per user First, the number of ratings per user was investigated whether it affects the movie rating. For this, the number of distinct movie IDs associated to a user ID was counted and stratified to multiples of ten. The number of individual ratings per group are shown in Fig. 7. The biggest user group has only rated between 15 to 24 individual movies. The size of user groups with more ratings is rapidly declining. Fig. 8 demonstrates that the average rating generally decreases with an increasing number of ratings.

```
fig_7
```

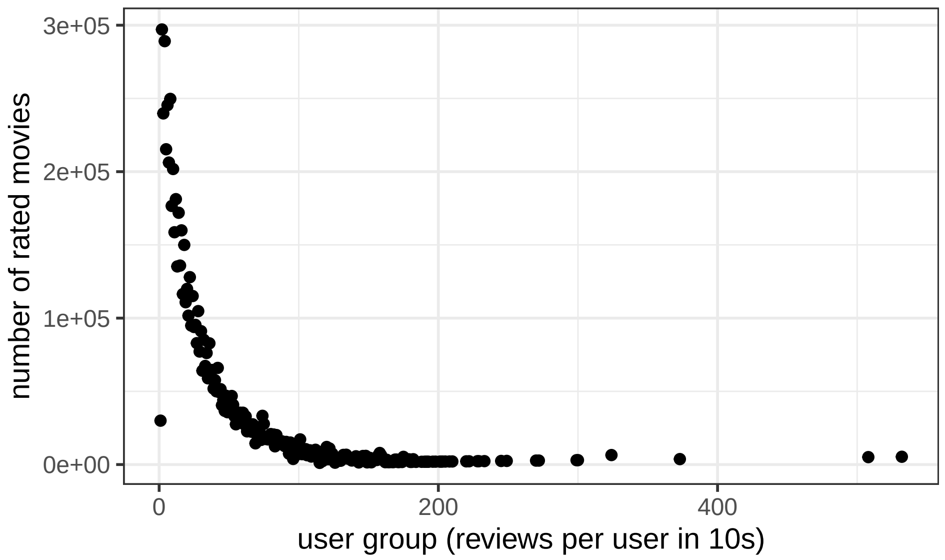

Figure 7: number of reviewed movies per user group (number of reviews in 10s)

```
fig_8
```

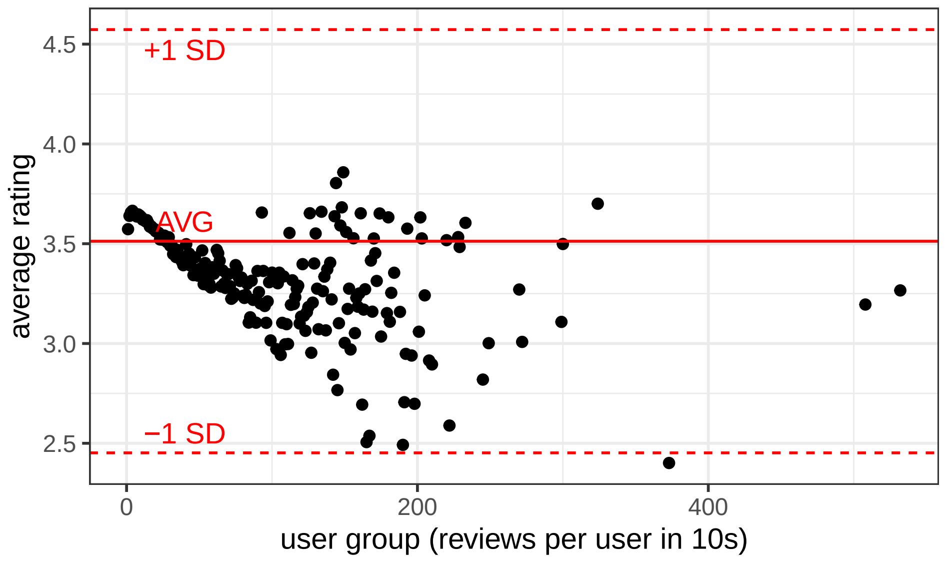

Figure 8: average rating per per user group (number of reviews in 10s)

The number of reviews increases over time as the user submits more and more reviews which means that a possible bias changes. This is investigated further. To do this, the review_date is stratified to weeks and assigned an order for a specific user. The average rating is calculated for a review week (Fig. 9) and the review week order counted from the first week when a review was submitted by a specific user (Fig. 10). While the average for the review week is fluctuating around the overall mean rating, the average rating is slightly below the overall average rating with increasing review order.

```
fig_9
```

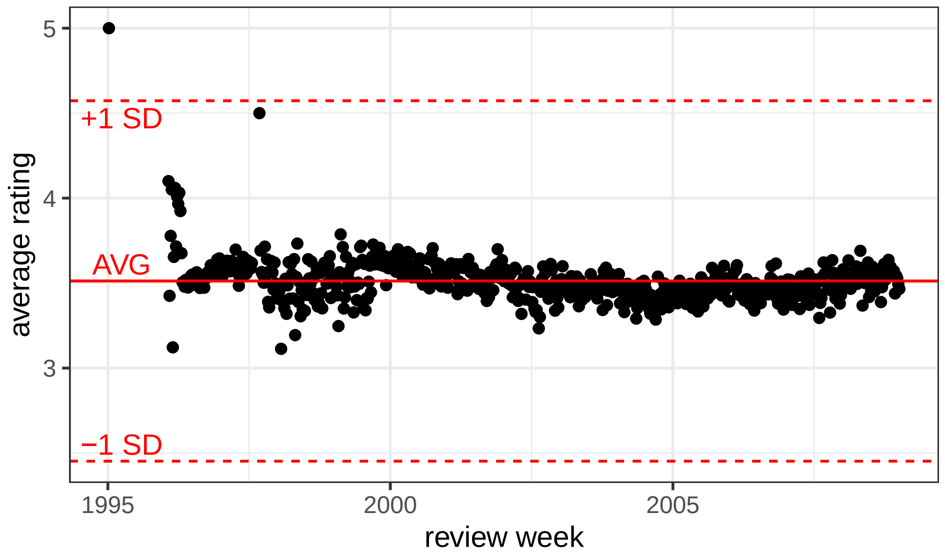

Figure 9: time course in weeks of average rating

```
fig_10
```

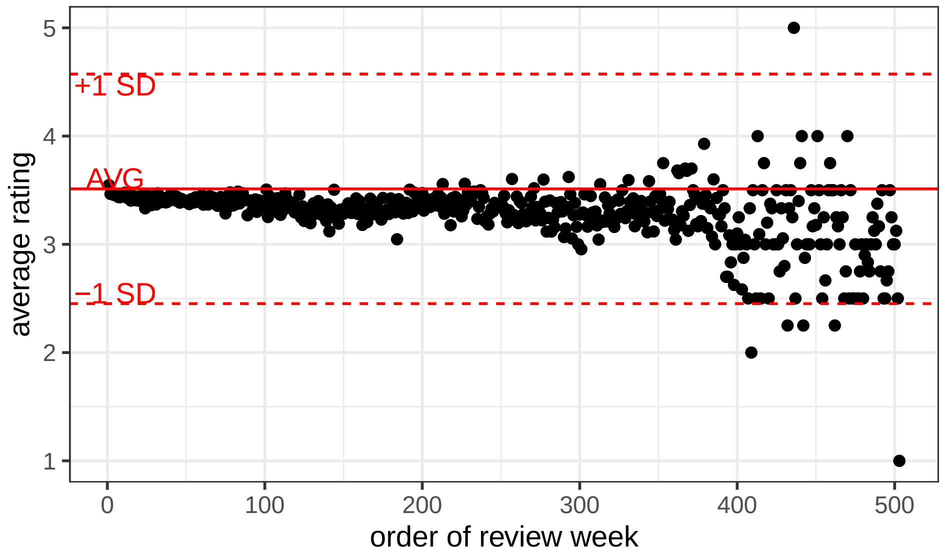

Figure 10: average rating in ordered review week

Timespan in years between review_date and the year of publication Then, the number of years t after the publication year was investigated whether it affects the movie rating. For this, the publication year was subtracted from the review year. The number of ratings per t are shown in Fig. 11. Fig. 12 demonstrates that the average rating generally increases to a plateau (t approx. 30 years) with a little bump at around 10 years after movie publication.

```
fig_11
```

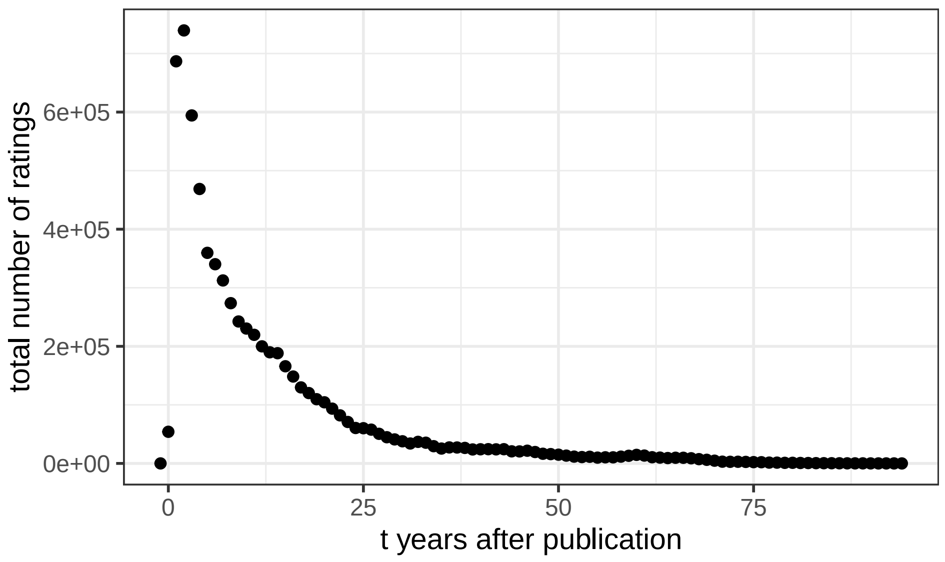

Figure 11: number of ratings t years after publication

```
fig_12
```

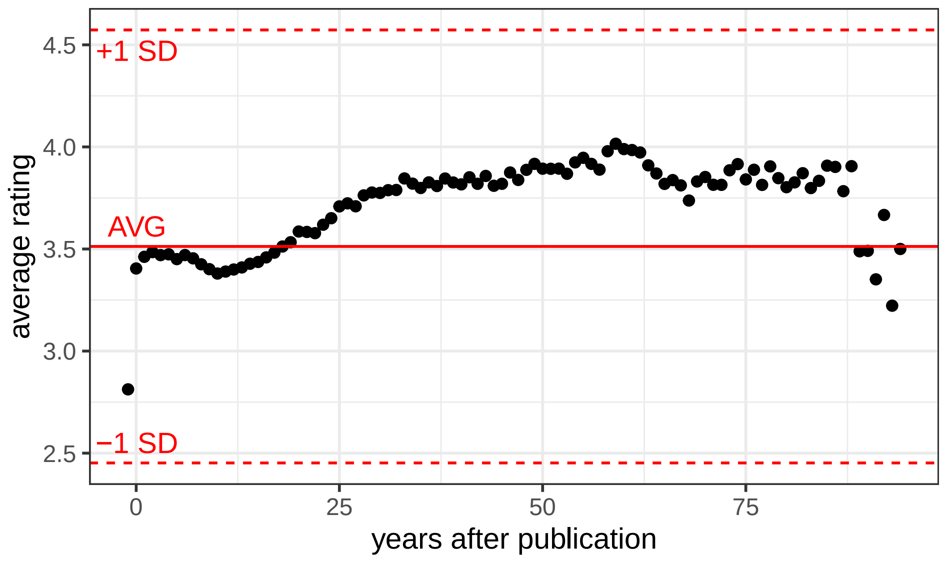

Figure 12: average rating t years after publication

Modelling

In this section, the creation of the model is described.

Definition of bias parameters

In this section, the process to the final prediction model is described. In a first step, bias parameters associated to a specific movie are calculated: the effect of the publication year (b_{py}), the effect of the genre (b_g), and the movie bias (b_i).

```
#Modelling of movie effects

# genre bias
  genre_bias <- edx_work_train %>%
    group_by(genres) %>%
    summarize(b_g=mean(rating-mu)) %>%
    ungroup()

# pub year bias
  pub_year_bias <- edx_work_train %>%
    left_join(x.,y=genre_bias,by="genres") %>%
    group_by(pub_year) %>%
    summarize(b_py=mean(rating-mu-b_g)) %>%
    ungroup()

# movie bias
  movie_bias <- edx_work_train %>%
    left_join(x.,y=genre_bias,by="genres") %>%
    left_join(x=.,y=pub_year_bias,by="pub_year") %>%
    group_by(movieId) %>%
    summarize(b_i=mean(rating-mu-b_g-b_py),n_rating=n_distinct(userId)) %>%
    ungroup()

fig_13 <- ggplot(data=genre_bias,aes(b_g))+geom_histogram(binwidth = 0.1) +
  xlab("genre bias") +
  ylab("frequency") +
  theme_bw()
fig_14 <- ggplot(data=pub_year_bias,aes(b_py))+geom_histogram(binwidth = 0.1)  +
  xlab("publication year bias") +
  ylab("frequency") +
  theme_bw()
fig_15 <- ggplot(data=movie_bias,aes(b_i))+geom_histogram(binwidth = 0.1)  +
  xlab("true movie bias") +
  ylab("frequency") +
  theme_bw()
```

The distribution of these parameters are shown in Fig. 13, Fig. 14, and Fig. 15.

```
fig_13
```

```
fig_14
```

```
fig_15
```

Next, bias parameters associated to a specific user are calculated: the effect of t, the number of years after the publication year a review was submitted (b_t), the bias of the week the review was submitted (b_w), and the effect of the user (b_u). They were substracted from the residual rating which could not be explained by the movie-associated effects as defined above.

```
#Modelling of user effects

# user bias
  user_bias <- edx_work_train %>%
    left_join(x.,y=genre_bias,by="genres") %>%
    left_join(x=.,y=pub_year_bias,by="pub_year") %>%
```

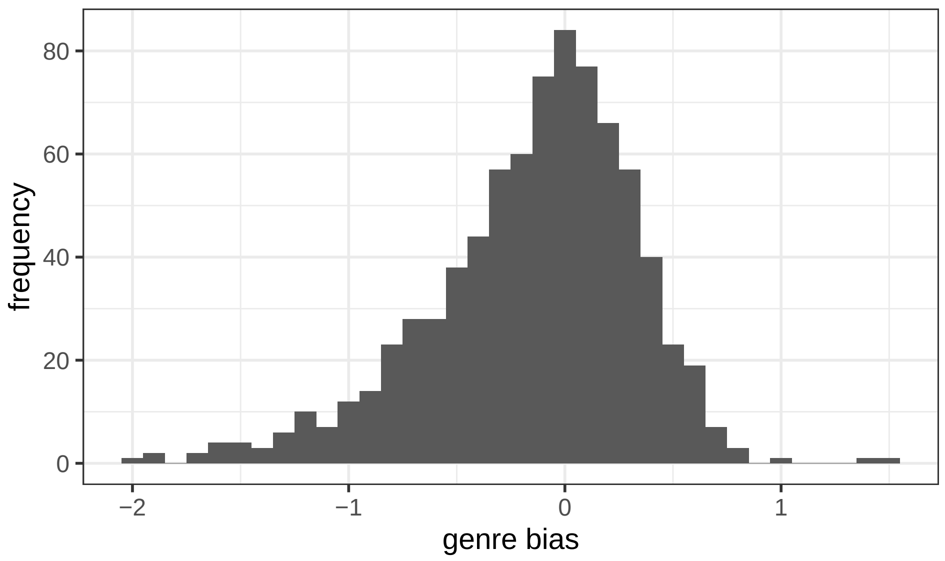

Figure 13: distribution of genre bias

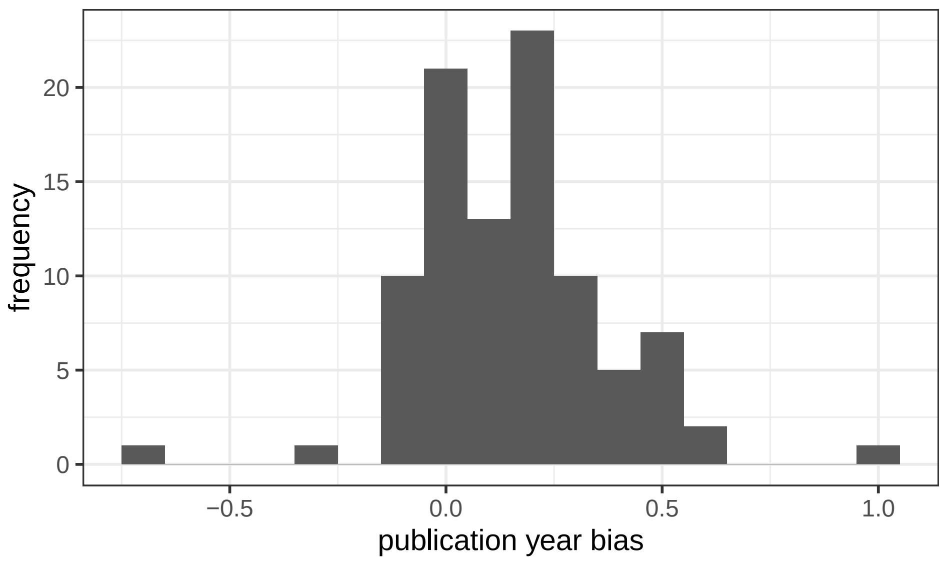

Figure 14: distribution of publication year bias

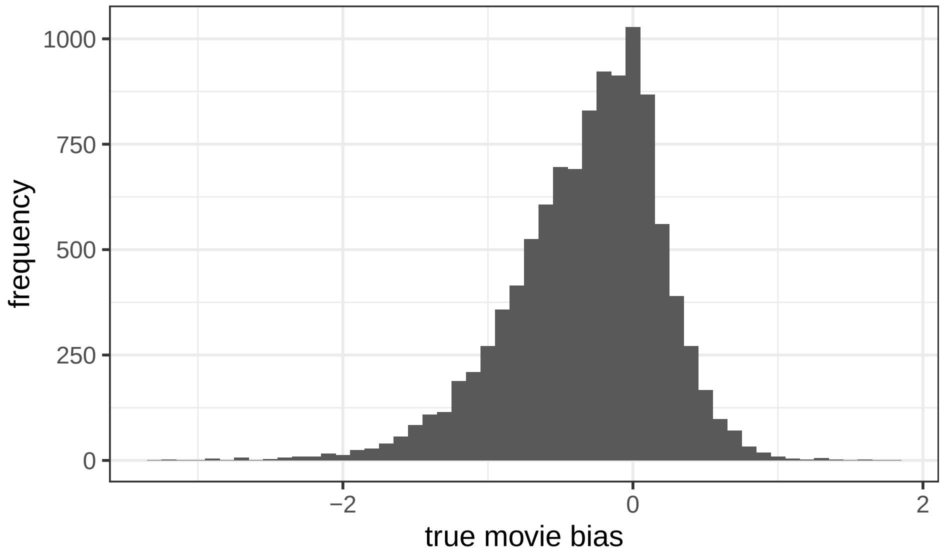

Figure 15: distribution of movie bias

```
    left_join(x=.,y=movie_bias,by="movieId") %>%
    group_by(userId) %>%
    summarize(b_u=mean(rating-mu-b_g-b_py-b_i)) %>%
    ungroup()

# review week bias
  review_week_bias <- edx_work_train %>%
    mutate(t= year(round_date(review_date, "year")) -
             pub_year,review_week=round_date(review_date,"week")) %>%
    left_join(x.,y=genre_bias,by="genres") %>%
    left_join(x=.,y=pub_year_bias,by="pub_year") %>%
    left_join(x=.,y=movie_bias,by="movieId") %>%
    left_join(x=.,y=user_bias,by="userId") %>%
    group_by(review_week) %>%
    summarize(b_w=mean(rating-mu-b_g-b_py-b_i-b_u)) %>%
    ungroup()

# t bias
  t_bias <- edx_work_train %>%
    mutate(t= year(round_date(review_date, "year")) -
             pub_year,review_week=round_date(review_date,"week")) %>%
    left_join(x.,y=genre_bias,by="genres") %>%
    left_join(x=.,y=pub_year_bias,by="pub_year") %>%
    left_join(x=.,y=movie_bias,by="movieId") %>%
    left_join(x=.,y=user_bias,by="userId") %>%
    left_join(x=.,y=review_week_bias,by="review_week") %>%
    group_by(t) %>%
    summarize(b_t=mean(rating-mu-b_g-b_py-b_i-b_u-b_w)) %>%
    ungroup()

fig_16 <- ggplot(data=t_bias,aes(b_t))+geom_histogram(binwidth = 0.1)  +
  xlab("t bias") +
  ylab("frequency") +
  theme_bw()

fig_17 <- ggplot(data=user_bias,aes(b_u))+geom_histogram(binwidth = 0.1)  +
  xlab("user bias") +
  ylab("frequency") +
  theme_bw()

fig_18 <- ggplot(data=review_week_bias,aes(b_w))+geom_histogram(binwidth = 0.1)  +
  xlab("review week bias") +
  ylab("frequency") +
  theme_bw()
```

The distributions of user-associated parameters are shown in Fig. 16, Fig. 17, and Fig. 18.

```
fig_16
```

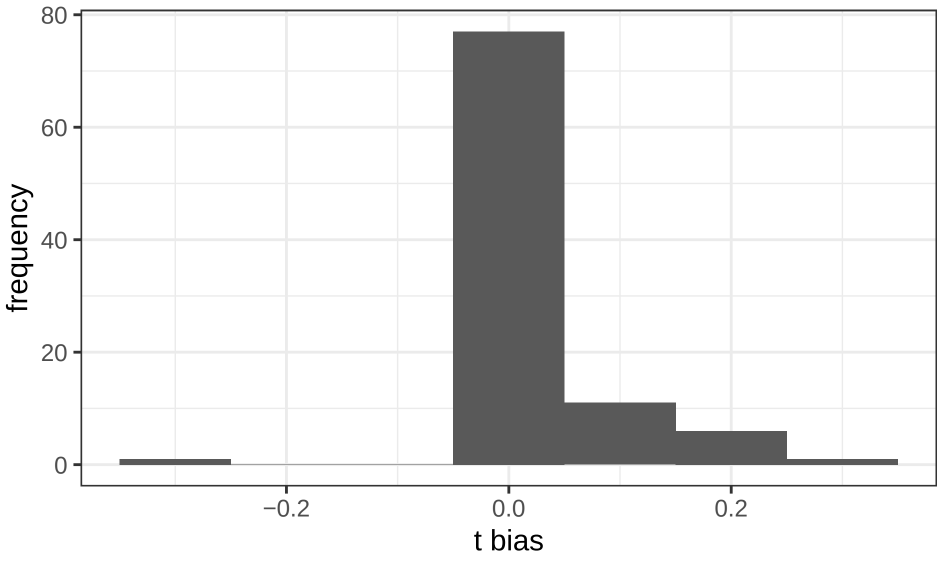

Figure 16: distribution of t bias

```
fig_17
```

```
fig_18
```

Preliminary models

To combine the parameters and predict ratings generalized linear models with elastic net regularization were chosen. Elastic net regularization is a combination of Lasso (least absolute shrinkage and selection operator) and Ridge regularization (glmnet). In Lasso regularization the absolute value of estimated weights multiplied by a tuning parameter *lambda* is added to the residual sum squares term to evaluate a fit, while in Ridge regression regression it is the squared weights multiplied by *lambda*. The mixture between Lasso and rigde is controlled by the tuning parameter *alpha* ranging between 0 and 1. Tuning parameters are selected by 5-fold cross-validation. Biases are used as defined above, addtionally the user specific review order and the number of ratings a movie has received are also included. Four models are compared, *eln_model_1* only contains the bias as defined previously, while *eln_model_2* also takes additional information into account: the order of reviews (stratified by weeks) a user has submitted, the number of ratings a movie has received in total is

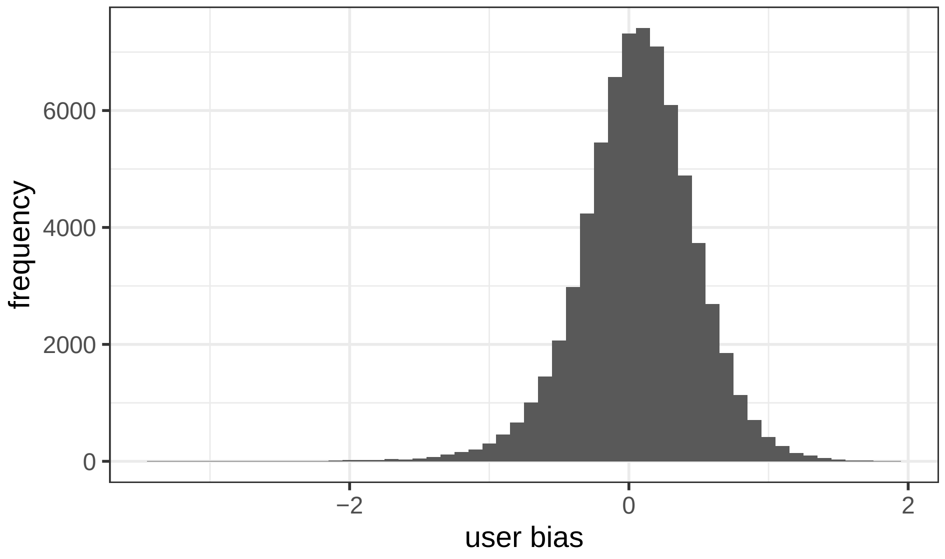

Figure 17: distribution of user bias

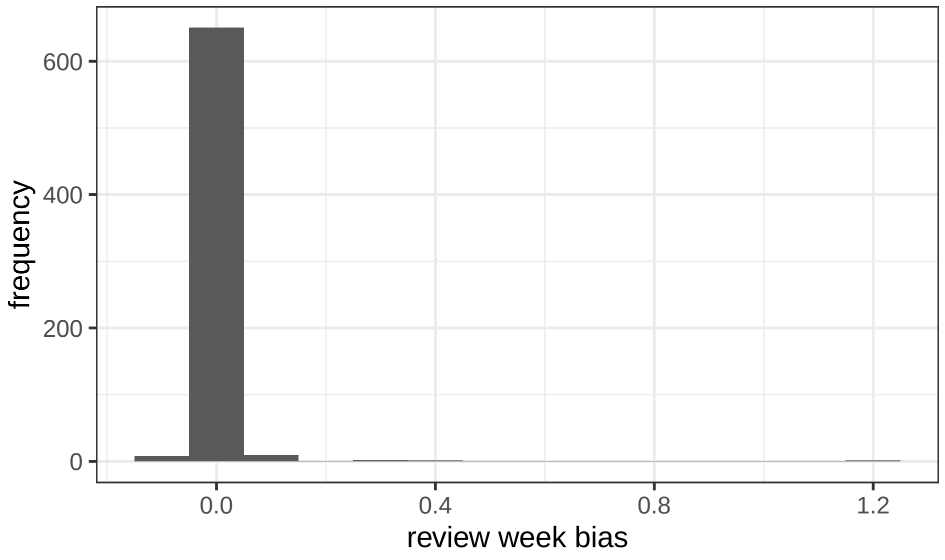

Figure 18: distribution of review week bias

included. *eln_model_3* excludes the bias of review week and *eln_model_4* uses the review week directly as an additional variable.

```r
## define the settings for cross-validation.
ctrl_pars <- trainControl(method="cv", number=5)

#define the training dataset:

# define the user-specific review order
  review_order_train <- edx_work_train %>%
    mutate(review_week=round_date(review_date,"week")) %>%
    select(userId,review_week) %>%
    distinct() %>%
    group_by(userId) %>%
    summarize(review_week,review_order=order(review_week)) %>%
    ungroup()

# join to data set
  edx_work_train_order <- edx_work_train %>%
    mutate(t= year(round_date(review_date, "year")) -
             pub_year,review_week=round_date(review_date,"week")) %>%
    mutate(review_week=round_date(review_date,"week")) %>%
    left_join(x=.,y=review_order_train, by=c("userId","review_week"))

# complete the dataset
  edx_work_train_eln <- edx_work_train_order %>%
    left_join(x=.,y=movie_bias,by="movieId") %>%
    left_join(x=.,y=genre_bias,by="genres") %>%
    left_join(x=.,y=pub_year_bias,by="pub_year") %>%
    left_join(x=.,y=user_bias,by="userId") %>%
    left_join(x=.,y=review_week_bias,by="review_week") %>%
    left_join(x=.,y=t_bias,by="t") %>%
    select(rating,movieId,userId,b_g,b_py,b_i,b_u,b_w,b_t,
           n_rating,review_order,review_week)


#define the test data set

# define the user-specific review order
  review_order_test <- edx_work_test %>%
    mutate(review_week=round_date(review_date,"week")) %>%
    select(userId,review_week) %>%
    distinct() %>%
    group_by(userId) %>%
    summarize(review_week,review_order=order(review_week)) %>%
    ungroup()

# join to data set
  edx_work_test_order <- edx_work_test %>%
    mutate(t= year(round_date(review_date, "year")) -
             pub_year,review_week=round_date(review_date,"week")) %>%
    mutate(review_week=round_date(review_date,"week")) %>%
    left_join(x=.,y=review_order_test, by=c("userId","review_week"))
```

```r
# complete the dataset
  edx_work_test_eln <- edx_work_test_order %>%
    left_join(x=.,y=movie_bias,by="movieId") %>%
    left_join(x=.,y=genre_bias,by="genres") %>%
    left_join(x=.,y=pub_year_bias,by="pub_year") %>%
    left_join(x=.,y=user_bias,by="userId") %>%
    left_join(x=.,y=review_week_bias,by="review_week") %>%
    left_join(x=.,y=t_bias,by="t") %>%
    select(rating,movieId,userId,b_g,b_py,b_i,b_u,b_w,b_t,
           n_rating,review_order,review_week)

#building the models

set.seed(1, sample.kind="Rounding")
eln_model_1 <- train(rating ~ b_py + b_g + b_i + b_t + b_u + b_w,
                   data = edx_work_train_eln,
                   method = "glmnet",
                   trControl = ctrl_pars,
                   metric = "RMSE",
                   family="gaussian",
                   tuneGrid = expand.grid(alpha =seq(0,1,0.33),
                                          lambda = seq(0.0001,1,length =100)))

set.seed(1, sample.kind="Rounding")
eln_model_2 <- train(rating ~ b_g + b_py + b_i + b_u + b_t + b_w + n_rating + review_order,
                   data = edx_work_train_eln,
                   method = "glmnet",
                   trControl = ctrl_pars,
                   metric = "RMSE",
                   family="gaussian",
                   tuneGrid = expand.grid(alpha =seq(0,1,0.33),
                                          lambda = seq(0.0001,1,length =100)))

set.seed(1, sample.kind="Rounding")
eln_model_3 <- train(rating ~ b_g + b_py + b_i + b_u + b_t + n_rating + review_order,
                   data = edx_work_train_eln,
                   method = "glmnet",
                   trControl = ctrl_pars,
                   metric = "RMSE",
                   family="gaussian",
                   tuneGrid = expand.grid(alpha =seq(0,1,0.33),
                                          lambda = seq(0.0001,1,length =100)))

set.seed(1, sample.kind="Rounding")
eln_model_4 <- train(rating ~ b_g + b_py + b_i + b_u + b_t + n_rating + review_order + review_week,
                   data = edx_work_train_eln,
                   method = "glmnet",
                   trControl = ctrl_pars,
                   metric = "RMSE",
                   family="gaussian",
                   tuneGrid = expand.grid(alpha =seq(0,1,0.33),
                                          lambda = seq(0.0001,1,length =100)))
```

```r
#making the predictions

eln_predict_1 <- predict(eln_model_1,newdata = edx_work_test_eln, na.action=na.pass)

eln_predict_2 <- predict(eln_model_2,newdata = edx_work_test_eln, na.action=na.pass)

eln_predict_3 <- predict(eln_model_3,newdata = edx_work_test_eln, na.action=na.pass)

eln_predict_4 <- predict(eln_model_4,newdata = edx_work_test_eln, na.action=na.pass)


# replace 'NA' in prediction with mu

eln_predict_1 <- tibble(eln_predict_1) %>%
  mutate(eln_predict_1 = ifelse(is.na(eln_predict_1) == TRUE, mu, eln_predict_1))

eln_predict_2 <- tibble(eln_predict_2) %>%
  mutate(eln_predict_2 = ifelse(is.na(eln_predict_2) == TRUE, mu, eln_predict_2))

eln_predict_3 <- tibble(eln_predict_3) %>%
  mutate(eln_predict_3 = ifelse(is.na(eln_predict_3) == TRUE, mu, eln_predict_3))

eln_predict_4 <- tibble(eln_predict_4) %>%
  mutate(eln_predict_4 = ifelse(is.na(eln_predict_4) == TRUE, mu, eln_predict_4))

# calculate RMSE

rmse_1 <- RMSE(edx_work_test_eln$rating,eln_predict_1$eln_predict_1)

rmse_2 <- RMSE(edx_work_test_eln$rating,eln_predict_2$eln_predict_2)

rmse_3 <- RMSE(edx_work_test_eln$rating,eln_predict_3$eln_predict_3)

rmse_4 <- RMSE(edx_work_test_eln$rating,eln_predict_4$eln_predict_4)
```

The final model is selected based on the performance as measured by comparing the RMSE tested on an independent test set *edx_work_test*. R-squared in all models indicates that just about one third of the variance in the data is explained by the models. *eln_model_2* is selected, demonstrating the lowest mean RMSE, highest mean R-squared, as well as the lowest RMSE with respect to the independent test data set *edx_work_test*.

```r
# print rmses
rmse_1
```

```
## [1] 0.8653047
```

```r
rmse_2
```

```
## [1] 0.8651449
```

```r
rmse_3
```

```
## [1] 0.8652332
```

```r
rmse_4
```

```
## [1] 0.8652332
```

```
# comparison of model performance (cross-validation)
model_list <- list(eln_model_1 = eln_model_1,
                   eln_model_2 = eln_model_2,
                   eln_model_3 = eln_model_3,
                   eln_model_4 = eln_model_4)

res <- resamples(model_list)
summary(res)
```

```
##
## Call:
## summary.resamples(object = res)
##
## Models: eln_model_1, eln_model_2, eln_model_3, eln_model_4
## Number of resamples: 5
##
## MAE
##                  Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## eln_model_1 0.6609570 0.6612051 0.6615365 0.6614994 0.6617952 0.6620031    0
## eln_model_2 0.6606288 0.6608757 0.6612258 0.6611838 0.6614870 0.6617017    0
## eln_model_3 0.6607338 0.6609974 0.6613361 0.6612919 0.6615917 0.6618005    0
## eln_model_4 0.6607338 0.6609974 0.6613361 0.6612919 0.6615917 0.6618005    0
##
## RMSE
##                  Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## eln_model_1 0.8543894 0.8548887 0.8550088 0.8551649 0.8555231 0.8560146    0
## eln_model_2 0.8541663 0.8546795 0.8547968 0.8549511 0.8553214 0.8557917    0
## eln_model_3 0.8542639 0.8547889 0.8548987 0.8550499 0.8554173 0.8558806    0
## eln_model_4 0.8542639 0.8547889 0.8548987 0.8550499 0.8554173 0.8558806    0
##
## Rsquared
##                  Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## eln_model_1 0.3491374 0.3493132 0.3493302 0.3496912 0.3499288 0.3507461    0
## eln_model_2 0.3494705 0.3496265 0.3496492 0.3500103 0.3502274 0.3510780    0
## eln_model_3 0.3493379 0.3494620 0.3494955 0.3498625 0.3500847 0.3509326    0
## eln_model_4 0.3493379 0.3494620 0.3494955 0.3498625 0.3500847 0.3509326    0
```

A final model based on *eln_model_2* is trained on the complete *edx* dataset and tested on the *validation* dataset.

The final model yields an RMSE of 0.86462 on the *validation* dataset. R-squared is around 0.35 and all bias parameters are of approximately equal immportance for the model, while *n_rating* and *review_order* do not contribute very much to model prediction.

```
# print rmse
rmse_final
```

```
## [1] 0.8645202
```

```
#summarize final model and compare to eln_model_2
res_final <- resamples(list(eln_model_final=eln_model_final,eln_model_2=eln_model_2))
summary(res_final)
```

```
##
## Call:
## summary.resamples(object = res_final)
```

```
##
## Models: eln_model_final, eln_model_2
## Number of resamples: 5
##
## MAE
##                      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## eln_model_final 0.6612723 0.6618938 0.6620359 0.6619043 0.6620696 0.6622497
## eln_model_2     0.6606288 0.6608757 0.6612258 0.6611838 0.6614870 0.6617017
##                 NA's
## eln_model_final    0
## eln_model_2        0
##
## RMSE
##                      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## eln_model_final 0.8554117 0.8554161 0.8562055 0.8559044 0.8562154 0.8562731
## eln_model_2     0.8541663 0.8546795 0.8547968 0.8549511 0.8553214 0.8557917
##                 NA's
## eln_model_final    0
## eln_model_2        0
##
## Rsquared
##                      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## eln_model_final 0.3475119 0.3481026 0.3487004 0.3484284 0.3487233 0.3491038
## eln_model_2     0.3494705 0.3496265 0.3496492 0.3500103 0.3502274 0.3510780
##                 NA's
## eln_model_final    0
## eln_model_2        0
```

```
eln_model_final$bestTune
```

```
##     alpha lambda
## 301 0.498  1e-04
```

```
#give variable importance
varImp(eln_model_final)
```

```
## glmnet variable importance
##
##                Overall
## b_g          100.00000
## b_i           98.22668
## b_u           96.23003
## b_py          94.85530
## b_t           86.67700
## b_w           86.35910
## review_order   0.02071
## n_rating       0.00000
```

Conclusion

Exploratory data analysis on 70 % of the initial dataset yielded effects of the individual movie, its publication year and its genre. Especially older movies with a publication year preceding the creation of the database or their release for movie rental services seem to have on average a higher rating than the younger movies. This can be interpreted as such, that only movies that were still popular (could get a high rating) at the time were actually included in movie rental or streaming services, while movies which were released at this time or later had not to go past this additional "quality check". Additionally it was observed, that the number of years a review was submitted after movie publication and the individual taste of the user should as well be taken into account. This temporal effect which is not completely independent from the effect of the publication year described above can be explained similarly, only movies with high ratings will actually be viewed (and reviewed) years after their publication. A bias on the week the review was submitted was also included. Furthermore, the number of ratings a movie receives in total shows some effect on the rating, and the individual order (stratified by weeks) in which a user submits the reviews (reflecting an individual development of preference or rating scheme). Since it was not clear whether individual parameters were fully independent and regularization was described as an essential recipe for success in the initial movielens competition in the blog article referenced in the text book, elastic net regularized generalized linear regression was selected as modelling framework. Generalized linear regression had the additional advantage that calculations were actually feasible with the available hardware given the size of the dataset. In the end the final model yields an RMSE of 0.86452 on the validation data set (10 % of the original data). A tuning hyperparameter $alpha = 0.498$ shows that an equal mixture of lasso and ridge regularization with a low penalty (*lambda*= 1e-04) is suitable for the model. The very low R-Square of around 0.35 shows that the majority of variance in the data is not explained by the model. Therefore, next steps which are not part of the present work would be to explore also other modelling frameworks and to also explore classification in contrast to regression as the rating scheme in the movie lens data is not continuous and therefore might be better represented as a classification task.