Report on MovieLens Project

Mario Träger

July 28, 2021

Introduction

Recommendation systems are information filtering systems that make recommendations to users for specific items based on user given ratings. These systems are widely used especially by internet companies. One approach to build a recommendation system is machine learning. Machine learning uses known data to train a mathematical model in order to make predictions on data with unknown outcomes. Items with a high predicted rating for specific users are then recommended to that users.

The task of this project is the development of a movie recommendation system based on the MovieLens data set. This data set consists of about ten million movie ratings released in January 2009. This task will be carried out by training a machine learning algorithm using the inputs in one subset to predict movie ratings in a validation subset. The ultimate goal of this effort is the development of an algorithm that predicts ratings in the validation set with an RMSE lower than 0.86490.

Methods and Analysis

Creating edx set and validation set

The data set was downloaded from the homepage of the GroupLens research lab in the Department of Computer Science and Engineering at the University of Minnesota. After the downloading, the data set was wrangled into two subsets, an edx set for training the machine learning algorithm and a validation set as the final hold-out test set:

```
library(tidyverse)
library(caret)
library(data.table)

dl <- tempfile()
download.file("http://files.grouplens.org/datasets/movielens/ml-10m.zip", dl)

ratings <- fread(text = gsub("::", "\t", readLines(unzip(dl, "ml-10M100K/ratings.dat"))),
                 col.names = c("userId", "movieId", "rating", "timestamp"))

movies <- str_split_fixed(readLines(unzip(dl, "ml-10M100K/movies.dat")), "\\::", 3)
colnames(movies) <- c("movieId", "title", "genres")

# if using R 4.0 or later:
movies <- as.data.frame(movies) %>% mutate(movieId = as.numeric(movieId),
                                           title = as.character(title),
                                           genres = as.character(genres))

movielens <- left_join(ratings, movies, by = "movieId")
```

```
# Validation set will be 10% of MovieLens data
set.seed(1, sample.kind="Rounding") # if using R 3.5 or earlier, use `set.seed(1)`
test_index <- createDataPartition(y = movielens$rating, times = 1, p = 0.1, list = FALSE)
edx <- movielens[-test_index,]
temp <- movielens[test_index,]

# Make sure userId and movieId in validation set are also in edx set
validation <- temp %>%
  semi_join(edx, by = "movieId") %>%
  semi_join(edx, by = "userId")

# Add rows removed from validation set back into edx set
removed <- anti_join(temp, validation)
edx <- rbind(edx, removed)

rm(dl, ratings, movies, test_index, temp, movielens, removed)
```

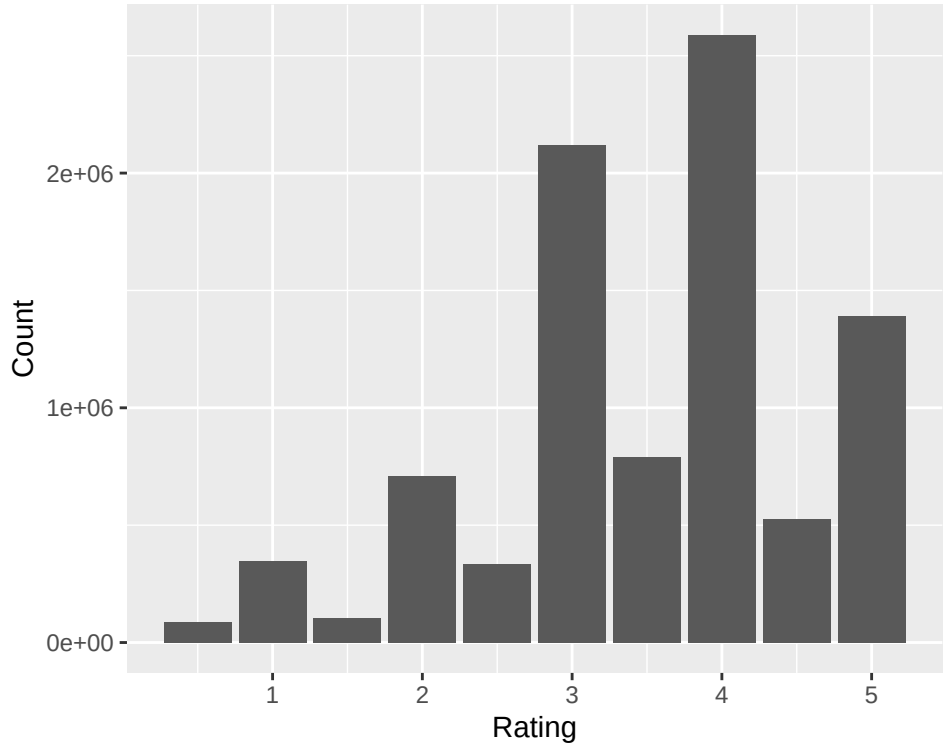
Data analysis

The the edx and validation sets consist of 9000055 and 999999 observations respectively and 6 features. In the edx data set 69878 unique users provide the ratings for 10677 unique movies. Besides the rating column the data sets contain columns for userId (unique for every user), movieId (unique for every movie), timestamp (point in time the rating was given), title (with release year) and genres associated with the movie:

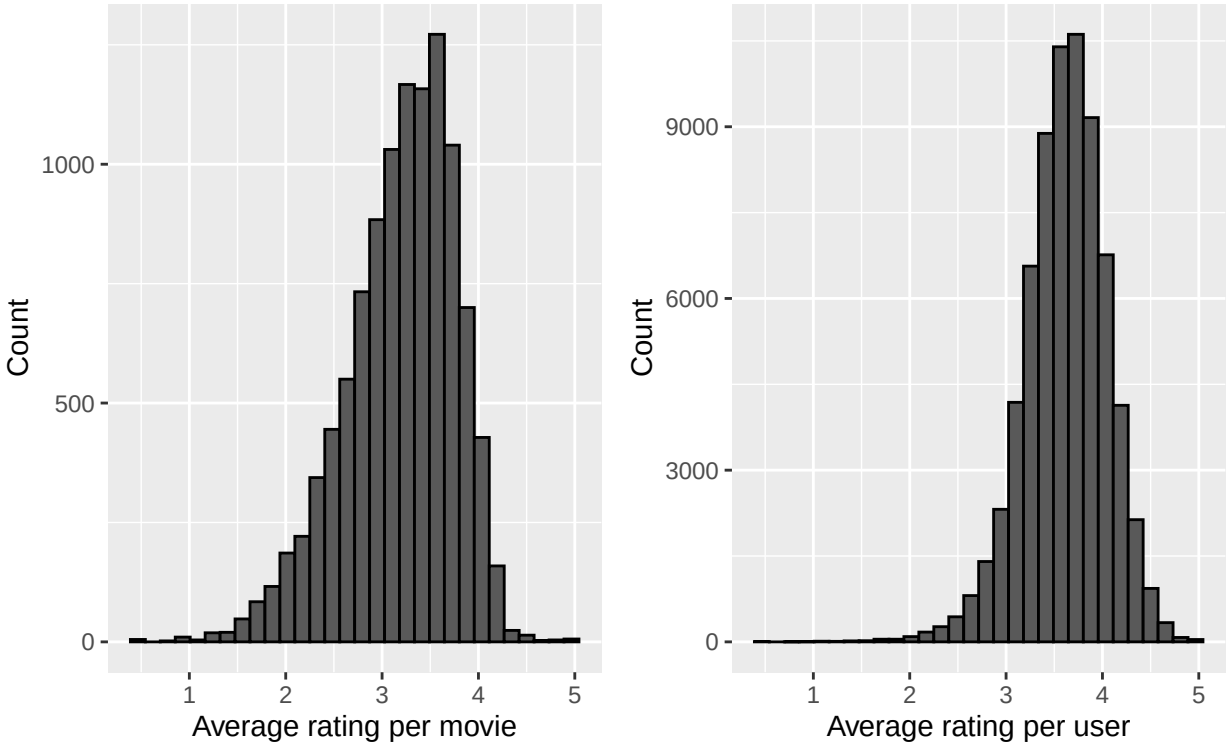
```
head(edx)
```

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

The ratings range from 0.5 to 5 stars with higher prevalence of whole star ratings than half star ratings. The most common ratings are 4, 3 and 5 stars, the least common ratings are 0.5 and 1.5 stars:



The overall mean rating of the edx set is 3.51. Looking at the average rating per movie shows a left skewed distribution with only a small fraction of movies rated below 1.5 and higher then 4.5. The distribution of the average rating per user is quite normal and only a very small fraction of users rated below 2 on average:



The mean of the average ratings per movie is 3.19 and the mean of the average ratings per user is 3.61. Thus,

in comparison to the overall mean there are clear movie and user biases.

Creating train and test sets

At first, the edx set was further split into a train and a test set with 20 % of the data going to test, in order to be able to train and test models independently from the validation set. To make sure both data sets contain the same users and movies the `semi_join` function was used:

```
set.seed(1994)

test_index <- createDataPartition(y = edx$rating, times = 1,
                                  p = 0.2, list = FALSE)
train_set <- edx[-test_index,]
test_set <- edx[test_index,]
test_set <- test_set %>%
  semi_join(train_set, by = "movieId") %>%
  semi_join(train_set, by = "userId")
```

These sets were only used for the naive model, the movie effect model and the movie + user effects model.

Function for RMSE calculation

In order to evaluate the goodness of the applied model, the root mean squared error (RMSE) between the true ratings $y_{u,i}$ and the predicted ratings $\hat{y}_{u,i}$ was calculated:

$$RMSE = \sqrt{\frac{1}{N}\sum_{u,i}(y_{u,i} - \hat{y}_{u,i})^2}$$

This value represents the typical deviation of predicted ratings from true ratings.

For this calculation, a predefined R function was used:

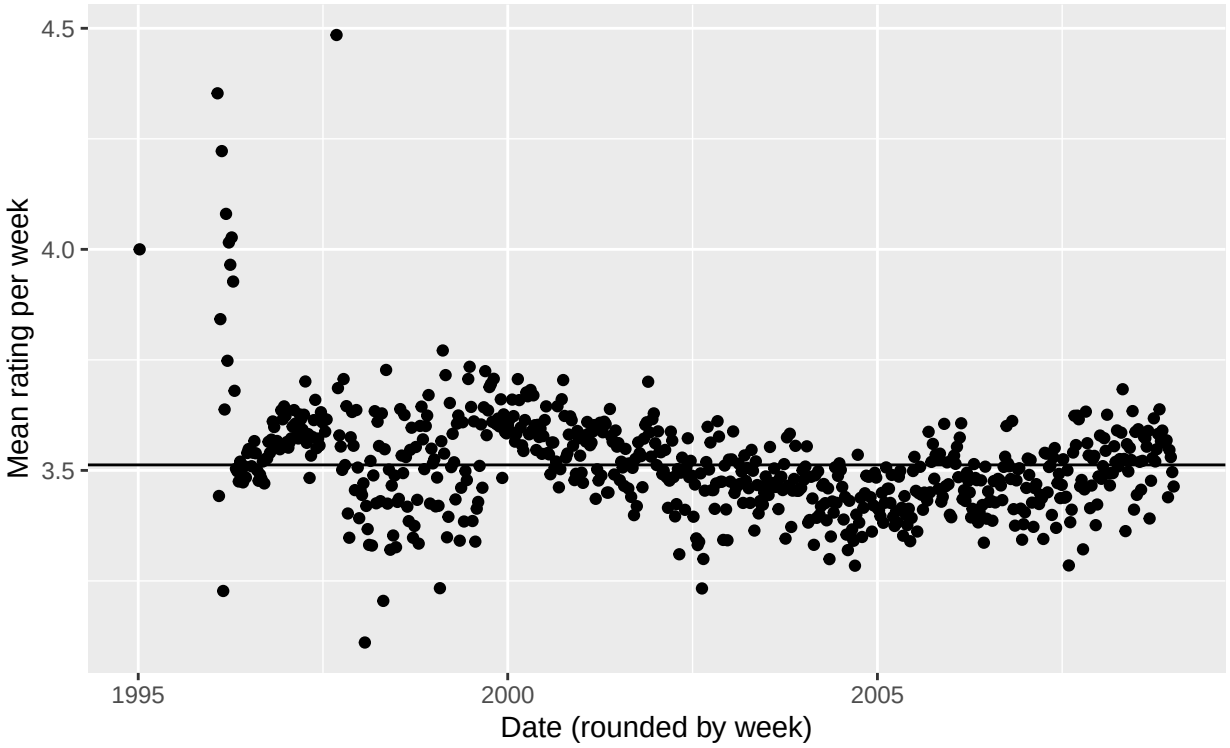
```
RMSE <- function(true_ratings, predicted_ratings){
  sqrt(mean((true_ratings - predicted_ratings)^2))
  }
```

Creating new columns for the usage as additional predictors

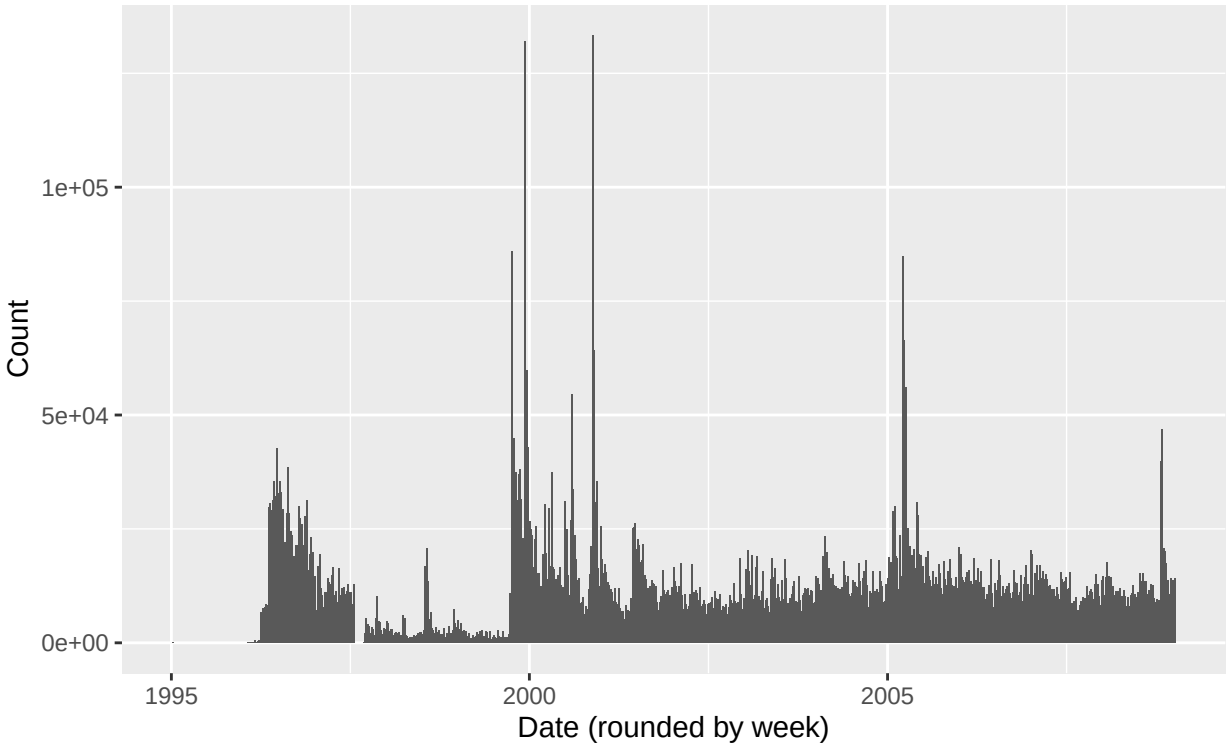
Date column: The timestamp column, representing the point in time the rating was given (in form of the number of seconds since the epoch 1970-01-01 00:00:00 UTC), was transformed into a date column in a year-month-day format and rounded by week by means of the `lubridate` package. The timestamp column itself was removed in order to keep the data set small:

```
library(lubridate)
edx <- edx %>%
  mutate(date = ymd(round_date(as_datetime(timestamp), unit = "week"))) %>%
  select(-timestamp)
```

The plot of the mean ratings per week versus the week shows a clear fluctuation in rating behavior over time (horizontal line represents the overall mean rating):



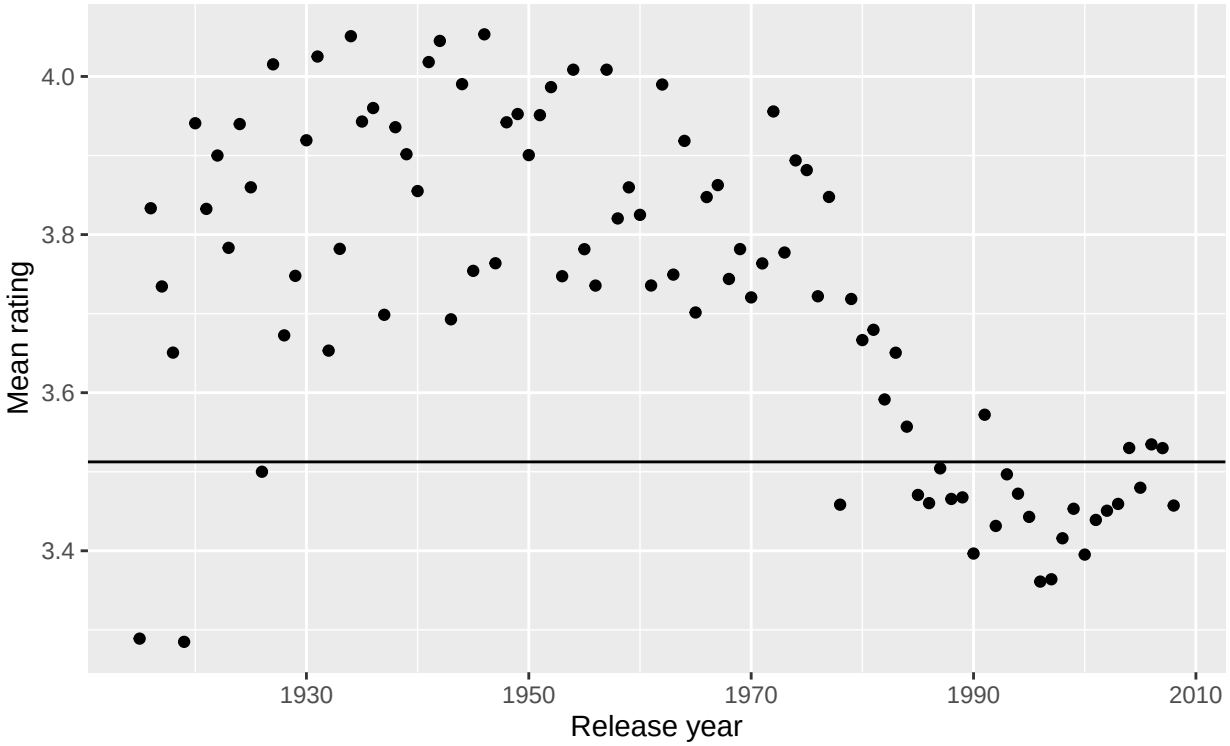
The plot of the counts per week shows an evenly distribution of the votes over the time range with exception of the time between 1997 and 2000 and some outliers:



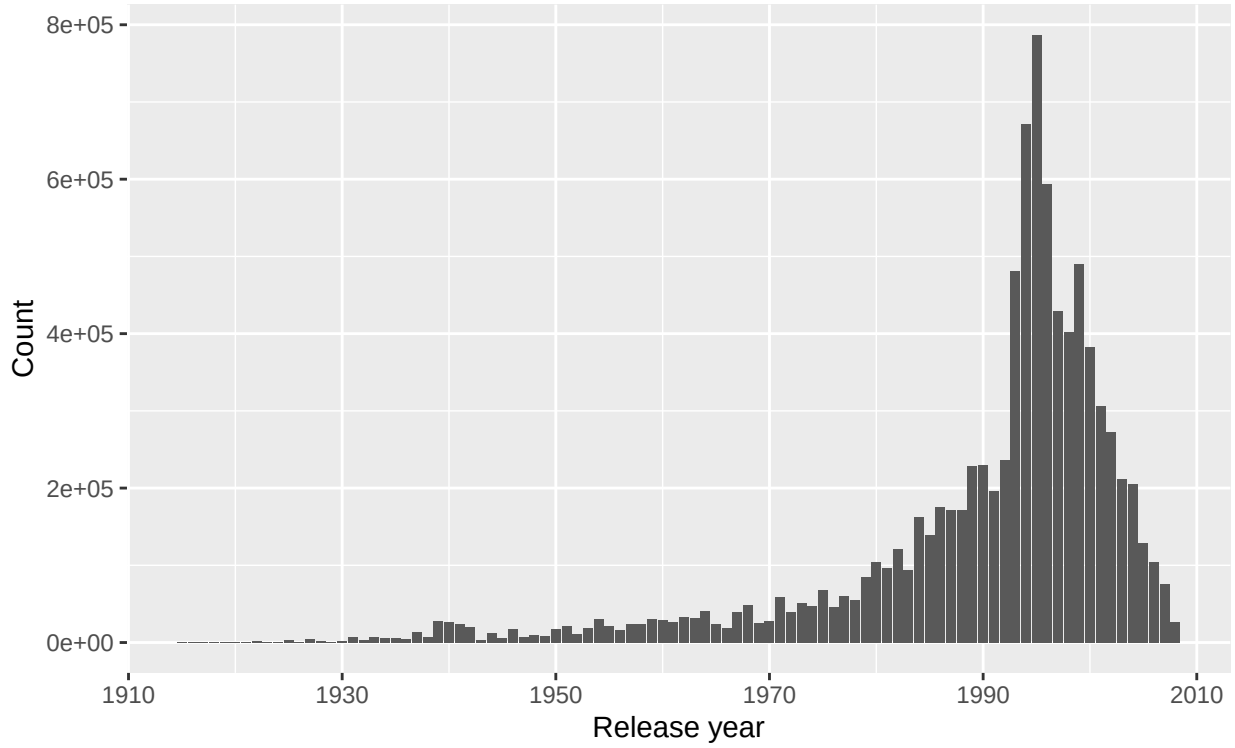
Year column: The release year of the movies was extracted from the title column into a newly generated year column by means of the `stringr` package, the title column was removed:

```
edx <- edx %>%
  mutate(year = str_extract(title, "\\s\\(\\d{4}\\)$"),
         year = as.numeric(str_extract(year, "\\d{4}"))) %>%
  select(-title)
```

The plot of the mean ratings versus the movie release year shows a clear trend with higher than average ratings for movies released between 1920 and 1980, about average ratings for movies of the eighties and lower than average ratings for movies released in the nineties and later (horizontal line represents the overall mean rating):



However, ratings for movies released in the eighties or later, especially in the nineties, are much more prevalent in the data set than ratings for older movies:



Validation data: In order to be able to predict movie ratings in the validation set, the validation set was transformed in the same way as the edx set before by generating date and year columns and removing timestamp and title columns:

```
validation <- validation %>%
  mutate(date = ymd(round_date(as_datetime(timestamp), unit = "week")),
         year = str_extract(title, "\\s\\(\\d{4}\\)$"),
         year = as.numeric(str_extract(year, "\\d{4}"))) %>%
  select(-timestamp, -title) %>%
  semi_join(edx, by = "date") %>%
  semi_join(edx, by = "year")
```

Creating 5 folds for cross-validation

Using only one train and one test set may lead to extraordinarily good or bad results just by chance. In order to calculate more stable estimates during the development process, the edx set was also split into five folds for use in cross-validation (cv):

```
set.seed(1994)

folds_index <- createFolds(y = edx$rating, k = 5)

fold1_train <-edx[-folds_index[[1]],]
fold1_test <- edx[folds_index[[1]],]
fold1_test <- fold1_test %>%
  semi_join(fold1_train, by = "movieId") %>%
  semi_join(fold1_train, by = "userId") %>%
  semi_join(fold1_train, by = "date") %>%
  semi_join(fold1_train, by = "year")
```

```
fold2_train <-edx[-folds_index[[2]],]
fold2_test <- edx[folds_index[[2]],]
fold2_test <- fold2_test  %>%
  semi_join(fold2_train, by = "movieId") %>%
  semi_join(fold2_train, by = "userId") %>%
  semi_join(fold2_train, by = "date") %>%
  semi_join(fold2_train, by = "year")

fold3_train <-edx[-folds_index[[3]],]
fold3_test <- edx[folds_index[[3]],]
fold3_test <- fold3_test  %>%
  semi_join(fold3_train, by = "movieId") %>%
  semi_join(fold3_train, by = "userId") %>%
  semi_join(fold3_train, by = "date") %>%
  semi_join(fold3_train, by = "year")

fold4_train <-edx[-folds_index[[4]],]
fold4_test <- edx[folds_index[[4]],]
fold4_test <- fold4_test  %>%
  semi_join(fold4_train, by = "movieId") %>%
  semi_join(fold4_train, by = "userId") %>%
  semi_join(fold4_train, by = "date") %>%
  semi_join(fold4_train, by = "year")

fold5_train <-edx[-folds_index[[5]],]
fold5_test <- edx[folds_index[[5]],]
fold5_test <- fold5_test  %>%
  semi_join(fold5_train, by = "movieId") %>%
  semi_join(fold5_train, by = "userId") %>%
  semi_join(fold5_train, by = "date") %>%
  semi_join(fold5_train, by = "year")
```

The cross-validation was used for all the regularized models.

Building a linear prediction model for movie ratings

The movie ratings prediction model was built according to the approach in the machine learning course. So the given code was adapted to the edx data set and subsequently improved by introducing cross-validation and adding the additional predictors rating date and release year.

Naive model: The simplest model for the prediction of rating outcomes is the assumption that the true ratings are just the population average of all ratings μ and the differences are explained by random variation. So the outcomes are a linear combination of the true mean μ and an error term ϵ with mean zero:

$$Y_{u,i} = \mu + \epsilon_{u,i}$$

The estimate for μ was determined by calculating the mean of all ratings in the training data $\hat{\mu}$.

Movie effect model: In the movie effect model a bias for movies b_i is added to the naive model:

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

The estimate for this movie-specific effect was determined by grouping by `movieId` and calculating the mean of the ratings for every movie minus $\hat{\mu}$:

$$\hat{b}_i = \frac{1}{n_i} \sum_{u=1}^{n_i} (Y_{u,i} - \hat{\mu})$$

Movie + User Effects Model: In the movie + user effects model a bias for users b_u is added to the movie effect model:

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

The estimate for this user-specific effect was determined by grouping by `userId` and calculating the mean of the ratings for every user minus $\hat{\mu}$ and the movie-specific effect, which was added initially to the dataset by `left_join` by `movieId`:

$$\hat{b}_u = \frac{1}{n_u} \sum_{i=1}^{n_u} (Y_{u,i} - \hat{\mu} - \hat{b}_i)$$

Movie + User + Date Effects Model: In the movie + user + date effects model a bias for the rating date (rounded by week) b_d is added to the movie + user effects model:

$$Y_{u,i} = \mu + b_i + b_u + f(b_d) + \epsilon_{u,i} \ , \text{ with } f \text{ a smooth function of } b_d$$

The estimate for this date-specific effect was determined by grouping by `date` and calculating the mean ratings for every week (minus $\hat{\mu}$) and subsequent smoothing of the time dependence. For this model the calculations were started with the $\hat{b}_d$ determination and addition of movie and user effects afterwards.

Movie + User + Date + Year Effects Model: In the movie + user + date + year effects model a bias for the release year b_y is added to the movie + user + date effects model:

$$Y_{u,i} = \mu + b_i + b_u + f(b_d) + g(b_y) + \epsilon_{u,i} \ , \text{ with } g \text{ a smooth function of } b_y$$

The estimate for this date-specific effect was determined by grouping by `year` and calculating the mean ratings for every release year (minus $\hat{\mu}$ and $\hat{b}_d$) and subsequent smoothing of the time dependence. For this model the calculations were started with the date effect and addition of year, movie and user effects afterwards.

Prediction of the ratings The prediction of the ratings was carried out by joining all biases to the test set by their corresponding predictors and subsequent summation of the effects according to the used model.

Smoothing of date and year data

For the smoothing of time dependent data a local weighted regression (loess) function was used. The loess function uses two tuning parameters, the span parameter controls the size of the neighborhood for the regression and the degree parameter decides whether the local fit is a line (degree = 1) or a parabola (degree = 2). Both tuning parameters were tuned to minimize the resulting overall RMSE of the model rather than fitting the data exactly.

Regularization

Estimating from small sample sizes may result in large estimates just by chance. Regularization is a technique that penalizes large estimates that originate from small sample sizes. This penalization was realized by adding a constant λ to the sample size while calculating the estimates of the biases. The regularization of the movie-specific effect looks like this:

$$\hat{b}_i(\lambda) = \frac{1}{n_i + \lambda} \sum_{u=1}^{n_i} (Y_{u,i} - \hat{\mu})$$

The λ value was tuned to minimize the resulting overall RMSE of the model.

Results

Naive model

The naive model assumes that the true ratings are just the population average ratings of all movies across all users plus a random error. The rating average of the train set, as an estimate for the population rating average, was calculated and then the RMSE function was used to determine the deviations from the test set ratings:

```
mu <- mean(train_set$rating)

naive_rmse <- RMSE(test_set$rating, mu)
```

The rating average in the training set $\hat{\mu}$ is 3.5124081. The prediction of the unknown ratings in the test set with this average yields an RMSE of 1.0600722. To compare this result with following results a table is used:

Method	RMSE
Just the average	1.060072

Movie effect model

The movie effect model adds a movie-specific effect to the naive model. The estimate for this effect was calculated and predictions were determined afterwards:

```
b_i <- train_set %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))
predicted_ratings <- test_set %>%
  left_join(b_i, by = "movieId") %>%
    mutate(pred = mu + b_i) %>%
  .$pred

model_1_rmse <- RMSE(predicted_ratings, test_set$rating)
```

The loss function on the test set rating yields an RMSE of 0.943604. In comparison to the naive model, the movie effect model provides a substantial improvement on the RMSE:

Method	RMSE
Just the average	1.060072
Movie Effect Model	0.943604

Movie + user effects model

The movie + user effects model adds an user-specific effect to the movie effect model. The estimate for this effect was calculated and predictions were determined afterwards:

```
b_u <- train_set %>%
  left_join(b_i, by="movieId") %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_i))

predicted_ratings <- test_set %>%
  left_join(b_i, by="movieId") %>%
  left_join(b_u, by="userId") %>%
  mutate(pred = mu + b_i + b_u) %>%
  .$pred

model_2_rmse <- RMSE(predicted_ratings, test_set$rating)
```

The loss function on the test set rating yields an RMSE of 0.8662377. In comparison to the movie effect model, the movie + user effects model provides another considerable improvement on the RMSE:

Method	**RMSE**
Just the average	1.0600722
Movie Effect Model	0.9436040
Movie + User Effects Model	0.8662377

Regularization on movie + user effects model

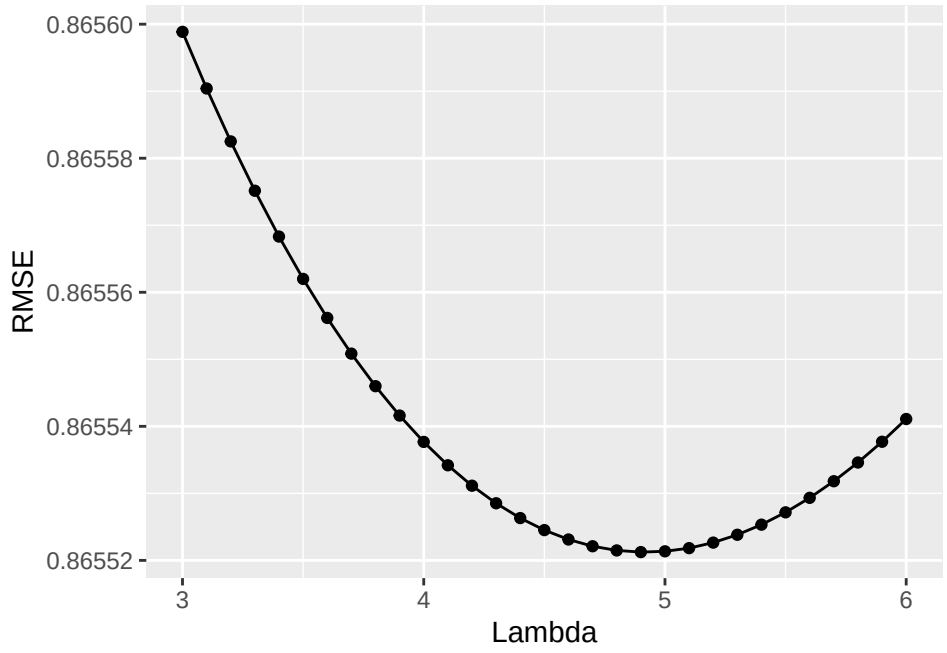
Regularization shrinks large estimates that originate from small sample sizes to circumvent overvaluation of these unstable estimates. The introduced parameter for this technique is the λ value, a measure for the degree of the shrinking. The λ value was varied between 3 and 6 to determine the value that minimizes the resulting RMSE of the algorithm. In order to obtain more stable estimates of the RMSE for every λ, a 5-fold cross-validation procedure was used instead of the former train and test set split:

```
lambdas <- seq(3, 6, 0.1)
rmses <- sapply(lambdas, function(l){
  #fold1
  mu <- mean(fold1_train$rating)
  b_i <- fold1_train %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n()+l))
  b_u <- fold1_train %>%
    left_join(b_i, by="movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - b_i - mu)/(n()+l))
  predicted_ratings <- fold1_test %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId")  %>%
    mutate(pred = mu + b_i + b_u) %>%
    .$pred
  rmse1 <- RMSE(predicted_ratings, fold1_test$rating)
  #fold2
  mu <- mean(fold2_train$rating)
  b_i <- fold2_train %>%
    group_by(movieId) %>%
```

```
  summarize(b_i = sum(rating - mu)/(n()+l))
b_u <- fold2_train %>%
  left_join(b_i, by="movieId") %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating - b_i - mu)/(n()+l))
predicted_ratings <- fold2_test %>%
  left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = "userId") %>%
  mutate(pred = mu + b_i + b_u) %>%
  .$pred
rmse2 <- RMSE(predicted_ratings, fold2_test$rating)
#fold3
mu <- mean(fold3_train$rating)
b_i <- fold3_train %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu)/(n()+l))
b_u <- fold3_train %>%
  left_join(b_i, by="movieId") %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating - b_i - mu)/(n()+l))
predicted_ratings <- fold3_test %>%
  left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = "userId") %>%
  mutate(pred = mu + b_i + b_u) %>%
  .$pred
rmse3 <- RMSE(predicted_ratings, fold3_test$rating)
#fold4
mu <- mean(fold4_train$rating)
b_i <- fold4_train %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu)/(n()+l))
b_u <- fold4_train %>%
  left_join(b_i, by="movieId") %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating - b_i - mu)/(n()+l))
predicted_ratings <- fold4_test %>%
  left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = "userId") %>%
  mutate(pred = mu + b_i + b_u) %>%
  .$pred
rmse4 <- RMSE(predicted_ratings, fold4_test$rating)
#fold5
mu <- mean(fold5_train$rating)
b_i <- fold5_train %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu)/(n()+l))
b_u <- fold5_train %>%
  left_join(b_i, by="movieId") %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating - b_i - mu)/(n()+l))
predicted_ratings <- fold5_test %>%
  left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = "userId") %>%
```

```
    mutate(pred = mu + b_i + b_u) %>%
    .$pred
  rmse5 <- RMSE(predicted_ratings, fold5_test$rating)
  #combination
  return((rmse1+rmse2+rmse3+rmse4+rmse5)/5)
})
```

The plot of the RMSE versus the λ values shows a parabolic dependence:




```
bestlambda <- lambdas[which.min(rmses)]
model_3_rmse <- min(rmses)
```

The minimum of λ is 4.9. The minimum value of the RMSE at this λ is 0.8655212. In comparison to the movie + user effects model, the regularized model provides a clear improvement on the RMSE:

Method	**RMSE**
Just the average	1.0600722
Movie Effect Model	0.9436040
Movie + User Effects Model	0.8662377
Regularized Movie + User Effects Model (CV)	0.8655212

Regularized movie + user + date effects model

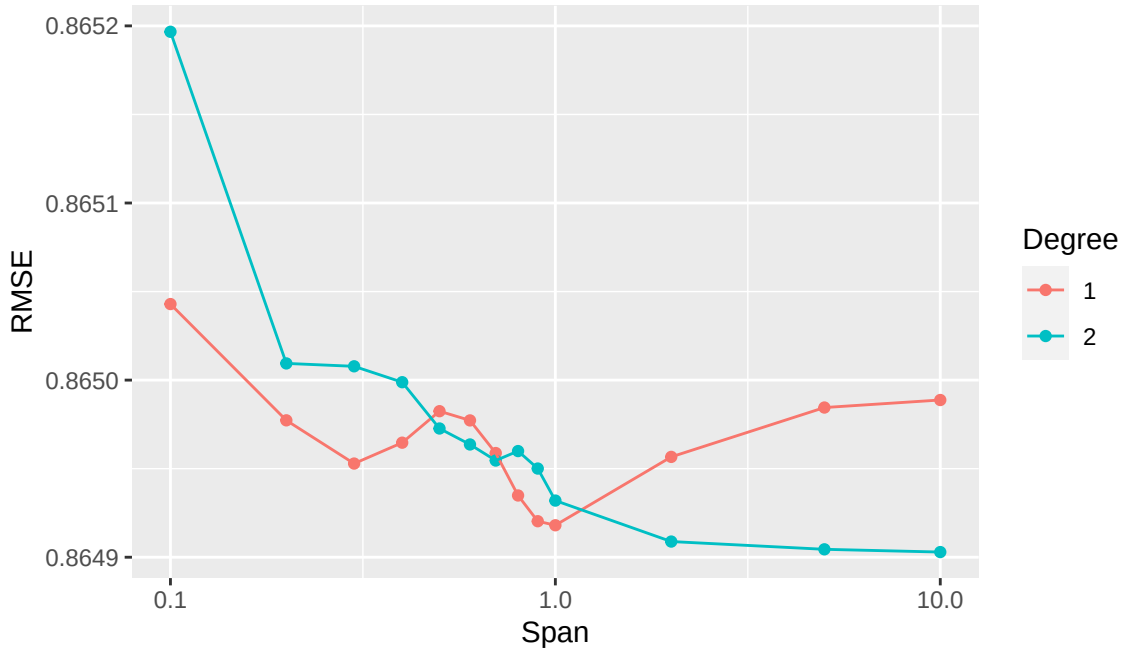
The regularized movie + user + date effects model adds a date-specific effect to the regularized movie + user effects model. The estimate for this effect was determined by calculating the mean rating for every week minus $\hat{\mu}$ and application of a loess smoothing on the time dependence. The calculation of the date effect was put in front of the calculation of the movie and user effects because this arrangement led to the best RMSE results. The regularization wasn't applied to the date effect due to the subsequent smoothing. In fact, the regularization before the smoothing increased the RMSE slightly (not shown). In order to determine the best loess parameter values, both parameters, span and degree, were varied in one apply function by using a parameter grid. As for the λ value, the best value of the regularized movie + user effects model was used:

```r
set.seed(1994)

l <- bestlambda
spans <- c(seq(0.1,1,0.1), 2, 5, 10)
degs <- 1:2
params <- expand.grid(s = spans, d = degs)

rmses <- apply(params,1,function(param){
  s <- param[1]
  d <- param[2]
  #fold1
  mu <- mean(fold1_train$rating)
  b_d <- fold1_train %>%
    group_by(date) %>%
    summarize(rating = mean(rating - mu)) %>%
    mutate(b_d = predict(loess(rating ~ as.numeric(date),
                               data = ., span = s, degree = d))) %>%
    select(-rating)
  b_i <- fold1_train %>%
    left_join(b_d, by = "date") %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu - b_d)/(n()+l))
  b_u <- fold1_train %>%
    left_join(b_d, by = "date") %>%
    left_join(b_i, by= "movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - mu - b_d - b_i)/(n()+l))
  predicted_ratings <- fold1_test %>%
    left_join(b_d, by = "date") %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId")  %>%
    mutate(pred = mu + b_d + b_i + b_u) %>%
    .$pred
  rmse1 <- RMSE(predicted_ratings, fold1_test$rating)
  #fold2
  mu <- mean(fold2_train$rating)
  b_d <- fold2_train %>%
    group_by(date) %>%
    summarize(rating = mean(rating - mu)) %>%
    mutate(b_d = predict(loess(rating ~ as.numeric(date),
                               data = ., span = s, degree = d))) %>%
    select(-rating)
  b_i <- fold2_train %>%
    left_join(b_d, by = "date") %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu - b_d)/(n()+l))
  b_u <- fold2_train %>%
    left_join(b_d, by = "date") %>%
    left_join(b_i, by= "movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - mu - b_d - b_i)/(n()+l))
  predicted_ratings <- fold2_test %>%
    left_join(b_d, by = "date") %>%
```

```
  left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = "userId")  %>%
  mutate(pred = mu + b_d + b_i + b_u) %>%
  .$pred
rmse2 <- RMSE(predicted_ratings, fold2_test$rating)
#fold3
mu <- mean(fold3_train$rating)
b_d <- fold3_train %>%
  group_by(date) %>%
  summarize(rating = mean(rating - mu)) %>%
  mutate(b_d = predict(loess(rating ~ as.numeric(date),
                             data = ., span = s, degree = d))) %>%
  select(-rating)
b_i <- fold3_train %>%
  left_join(b_d, by = "date") %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu - b_d)/(n()+l))
b_u <- fold3_train %>%
  left_join(b_d, by = "date") %>%
  left_join(b_i, by= "movieId") %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating - mu - b_d - b_i)/(n()+l))
predicted_ratings <- fold3_test %>%
  left_join(b_d, by = "date") %>%
  left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = "userId")  %>%
  mutate(pred = mu + b_d + b_i + b_u) %>%
  .$pred
rmse3 <- RMSE(predicted_ratings, fold3_test$rating)
#fold4
mu <- mean(fold4_train$rating)
b_d <- fold4_train %>%
  group_by(date) %>%
  summarize(rating = mean(rating - mu)) %>%
  mutate(b_d = predict(loess(rating ~ as.numeric(date),
                             data = ., span = s, degree = d))) %>%
  select(-rating)
b_i <- fold4_train %>%
  left_join(b_d, by = "date") %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu - b_d)/(n()+l))
b_u <- fold4_train %>%
  left_join(b_d, by = "date") %>%
  left_join(b_i, by= "movieId") %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating - mu - b_d - b_i)/(n()+l))
predicted_ratings <- fold4_test %>%
  left_join(b_d, by = "date") %>%
  left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = "userId")  %>%
  mutate(pred = mu + b_d + b_i + b_u) %>%
  .$pred
rmse4 <- RMSE(predicted_ratings, fold4_test$rating)
```

```
  #fold5
  mu <- mean(fold5_train$rating)
  b_d <- fold5_train %>%
    group_by(date) %>%
    summarize(rating = mean(rating - mu)) %>%
    mutate(b_d = predict(loess(rating ~ as.numeric(date),
                               data = ., span = s, degree = d))) %>%
    select(-rating)
  b_i <- fold5_train %>%
    left_join(b_d, by = "date") %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu - b_d)/(n()+l))
  b_u <- fold5_train %>%
    left_join(b_d, by = "date") %>%
    left_join(b_i, by= "movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - mu - b_d - b_i)/(n()+l))
  predicted_ratings <- fold5_test %>%
    left_join(b_d, by = "date") %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId")  %>%
    mutate(pred = mu + b_d + b_i + b_u) %>%
    .$pred
  rmse5 <- RMSE(predicted_ratings, fold5_test$rating)
  #combination
  return((rmse1+rmse2+rmse3+rmse4+rmse5)/5)
})
```

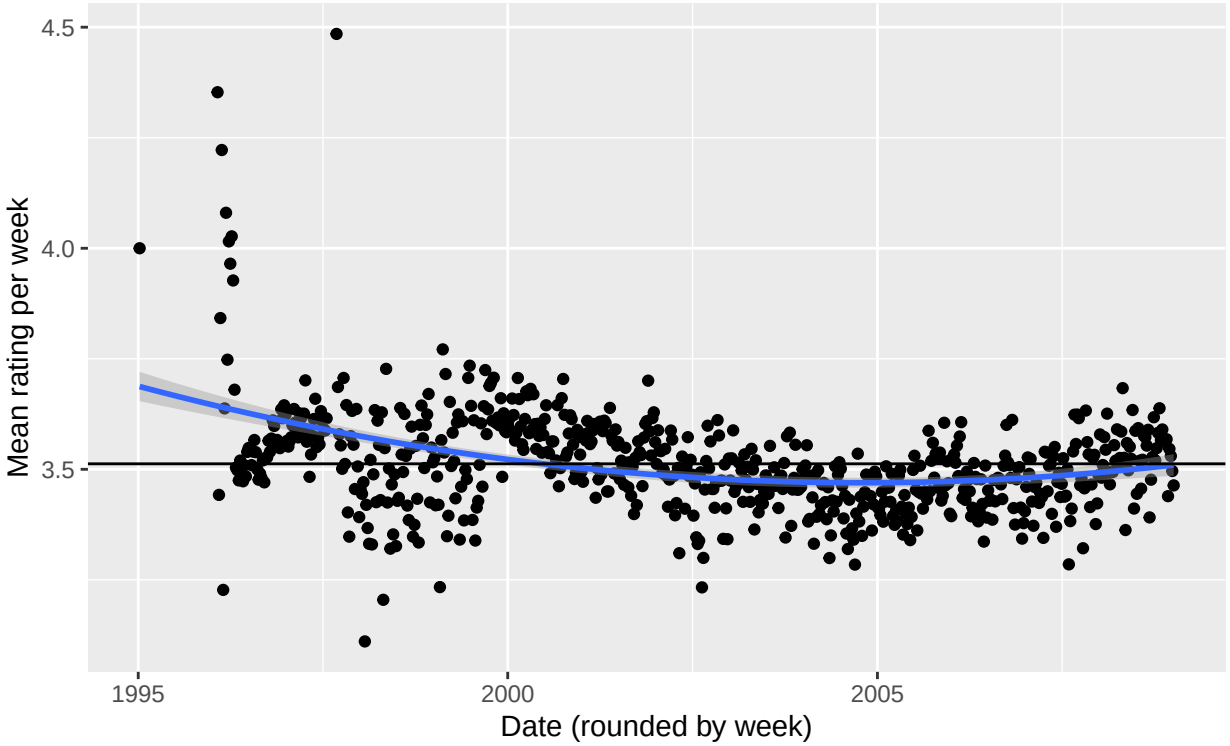
The RMSE was plotted against the span with degree 1 and 2 respectively:




```
params_date <- params[which.min(rmses),]
model_4_rmse <- min(rmses)
```

The plot shows a minimum at span = 10 and degree = 2. The minimum value of the RMSE at these parameters is 0.8649029.

The loess smoothing of rating average against date with these parameters on the edx data set does not follow the data exactly, but clearly represents the fluctuation:



In comparison to the regularized movie + user effects model, the regularized movie + user + date effects model clearly improves the RMSE, even though the date effect itself looks small:

Method	**RMSE**
Just the average	1.0600722
Movie Effect Model	0.9436040
Movie + User Effects Model	0.8662377
Regularized Movie + User Effects Model (CV)	0.8655212
Regularized Movie + User + Date Effects Model (CV)	0.8649029

Movie + user + date + year effects model

The regularized movie + user + date + year effects model adds a release year-specific effect to the preceding model. The estimate for this effect was determined by calculating the mean rating for every release year (minus $\hat{\mu}$ and $\hat{b}_d$) and a subsequent loess smoothing on the time dependence. The calculation of the year effect succeeded the calculation of the date effect and was followed by the movie and user effects. The regularization wasn't applied to the year effect due to the subsequent smoothing. In order to determine the best loess parameter values, both parameters, span and degree, were varied in one apply function by using a parameter grid. As for the λ value, the best value of the regularized movie + user effects model was used:

```
set.seed(1994)

l <- bestlambda
spans <- c(seq(0.1,0.9,0.1))
```

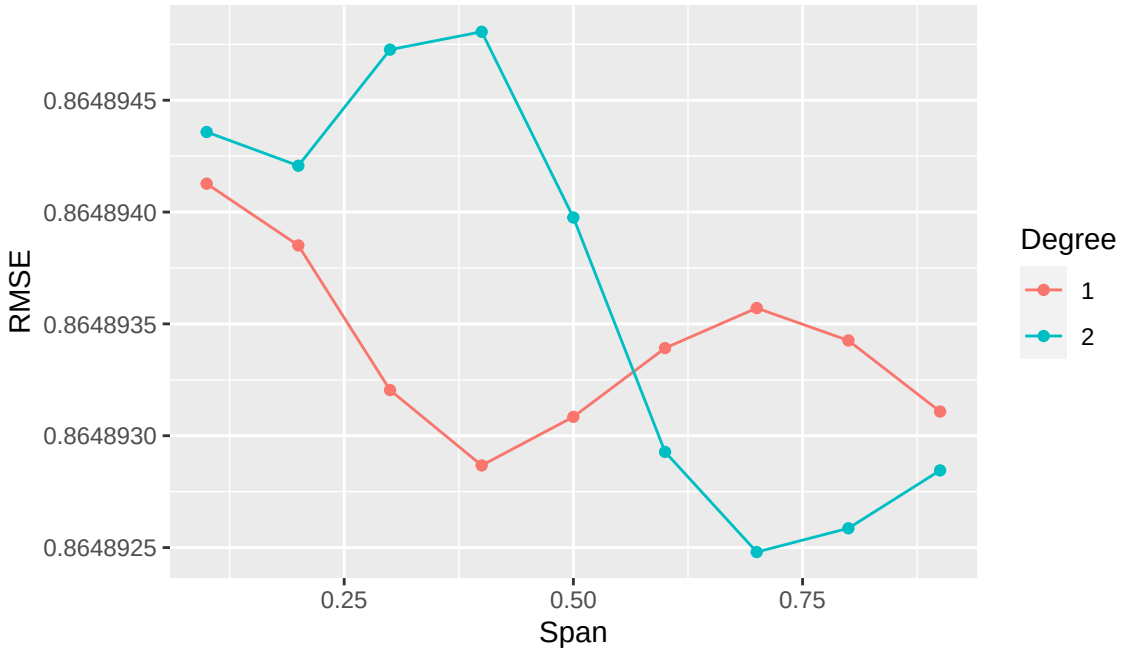
```
degs <- 1:2
params <- expand.grid(s = spans, d = degs)

rmses <- apply(params,1,function(param){
  s <- param[1]
  d <- param[2]
  #fold1
  mu <- mean(fold1_train$rating)
  b_d <- fold1_train %>%
    group_by(date) %>%
    summarize(rating = mean(rating - mu)) %>%
    mutate(b_d = predict(loess(rating ~ as.numeric(date), data = .,
                               span = params_date[1], degree = params_date[2]))) %>%
    select(-rating)
  b_y <- fold1_train %>%
    left_join(b_d, by = "date") %>%
    group_by(year) %>%
    summarize(rating = mean(rating - mu - b_d)) %>%
    mutate(b_y = predict(loess(rating ~ year, data = ., span = s, degree = d))) %>%
    select(-rating)
  b_i <- fold1_train %>%
    left_join(b_d, by = "date") %>%
    left_join(b_y, by = "year") %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu - b_d - b_y)/(n()+l))
  b_u <- fold1_train %>%
    left_join(b_d, by = "date") %>%
    left_join(b_y, by = "year") %>%
    left_join(b_i, by= "movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - mu - b_d - b_y - b_i)/(n()+l))
  predicted_ratings <- fold1_test %>%
    left_join(b_d, by = "date") %>%
    left_join(b_y, by = "year") %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId")  %>%
    mutate(pred = mu + b_i + b_u + b_d + b_y) %>%
    .$pred
  rmse1 <- RMSE(predicted_ratings, fold1_test$rating)
  #fold2
  mu <- mean(fold2_train$rating)
  b_d <- fold2_train %>%
    group_by(date) %>%
    summarize(rating = mean(rating - mu)) %>%
    mutate(b_d = predict(loess(rating ~ as.numeric(date), data = .,
                               span = params_date[1], degree = params_date[2]))) %>%
    select(-rating)
  b_y <- fold2_train %>%
    left_join(b_d, by = "date") %>%
    group_by(year) %>%
    summarize(rating = mean(rating - mu - b_d)) %>%
    mutate(b_y = predict(loess(rating ~ year, data = ., span = s, degree = d))) %>%
    select(-rating)
```

```
b_i <- fold2_train %>%
  left_join(b_d, by = "date") %>%
  left_join(b_y, by = "year") %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu - b_d - b_y)/(n()+l))
b_u <- fold2_train %>%
  left_join(b_d, by = "date") %>%
  left_join(b_y, by = "year") %>%
  left_join(b_i, by= "movieId") %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating - mu - b_d - b_y - b_i)/(n()+l))
predicted_ratings <- fold2_test %>%
  left_join(b_d, by = "date") %>%
  left_join(b_y, by = "year") %>%
  left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = "userId")  %>%
  mutate(pred = mu + b_i + b_u + b_d + b_y) %>%
  .$pred
rmse2 <- RMSE(predicted_ratings, fold2_test$rating)
#fold3
mu <- mean(fold3_train$rating)
b_d <- fold3_train %>%
  group_by(date) %>%
  summarize(rating = mean(rating - mu)) %>%
  mutate(b_d = predict(loess(rating ~ as.numeric(date), data = .,
                             span = params_date[1], degree = params_date[2]))) %>%
  select(-rating)
b_y <- fold3_train %>%
  left_join(b_d, by = "date") %>%
  group_by(year) %>%
  summarize(rating = mean(rating - mu - b_d)) %>%
  mutate(b_y = predict(loess(rating ~ year, data = ., span = s, degree = d))) %>%
  select(-rating)
b_i <- fold3_train %>%
  left_join(b_d, by = "date") %>%
  left_join(b_y, by = "year") %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu - b_d - b_y)/(n()+l))
b_u <- fold3_train %>%
  left_join(b_d, by = "date") %>%
  left_join(b_y, by = "year") %>%
  left_join(b_i, by= "movieId") %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating - mu - b_d - b_y - b_i)/(n()+l))
predicted_ratings <- fold3_test %>%
  left_join(b_d, by = "date") %>%
  left_join(b_y, by = "year") %>%
  left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = "userId")  %>%
  mutate(pred = mu + b_i + b_u + b_d + b_y) %>%
  .$pred
rmse3 <- RMSE(predicted_ratings, fold3_test$rating)
#fold4
```

```
mu <- mean(fold4_train$rating)
b_d <- fold4_train %>%
  group_by(date) %>%
  summarize(rating = mean(rating - mu)) %>%
  mutate(b_d = predict(loess(rating ~ as.numeric(date), data = .,
                             span = params_date[1], degree = params_date[2]))) %>%
  select(-rating)
b_y <- fold4_train %>%
  left_join(b_d, by = "date") %>%
  group_by(year) %>%
  summarize(rating = mean(rating - mu - b_d)) %>%
  mutate(b_y = predict(loess(rating ~ year, data = ., span = s, degree = d))) %>%
  select(-rating)
b_i <- fold4_train %>%
  left_join(b_d, by = "date") %>%
  left_join(b_y, by = "year") %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu - b_d - b_y)/(n()+l))
b_u <- fold4_train %>%
  left_join(b_d, by = "date") %>%
  left_join(b_y, by = "year") %>%
  left_join(b_i, by= "movieId") %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating - mu - b_d - b_y - b_i)/(n()+l))
predicted_ratings <- fold4_test %>%
  left_join(b_d, by = "date") %>%
  left_join(b_y, by = "year") %>%
  left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = "userId")  %>%
  mutate(pred = mu + b_i + b_u + b_d + b_y) %>%
  .$pred
rmse4 <- RMSE(predicted_ratings, fold4_test$rating)
#fold5
mu <- mean(fold5_train$rating)
b_d <- fold5_train %>%
  group_by(date) %>%
  summarize(rating = mean(rating - mu)) %>%
  mutate(b_d = predict(loess(rating ~ as.numeric(date), data = .,
                             span = params_date[1], degree = params_date[2]))) %>%
  select(-rating)
b_y <- fold5_train %>%
  left_join(b_d, by = "date") %>%
  group_by(year) %>%
  summarize(rating = mean(rating - mu - b_d)) %>%
  mutate(b_y = predict(loess(rating ~ year, data = ., span = s, degree = d))) %>%
  select(-rating)
b_i <- fold5_train %>%
  left_join(b_d, by = "date") %>%
  left_join(b_y, by = "year") %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu - b_d - b_y)/(n()+l))
b_u <- fold5_train %>%
  left_join(b_d, by = "date") %>%
```

```
    left_join(b_y, by = "year") %>%
    left_join(b_i, by= "movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - mu - b_d - b_y - b_i)/(n()+l))
  predicted_ratings <- fold5_test %>%
    left_join(b_d, by = "date") %>%
    left_join(b_y, by = "year") %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId")  %>%
    mutate(pred = mu + b_i + b_u + b_d + b_y) %>%
    .$pred
  rmse5 <- RMSE(predicted_ratings, fold5_test$rating)
  #combination
  return((rmse1+rmse2+rmse3+rmse4+rmse5)/5)
})
```

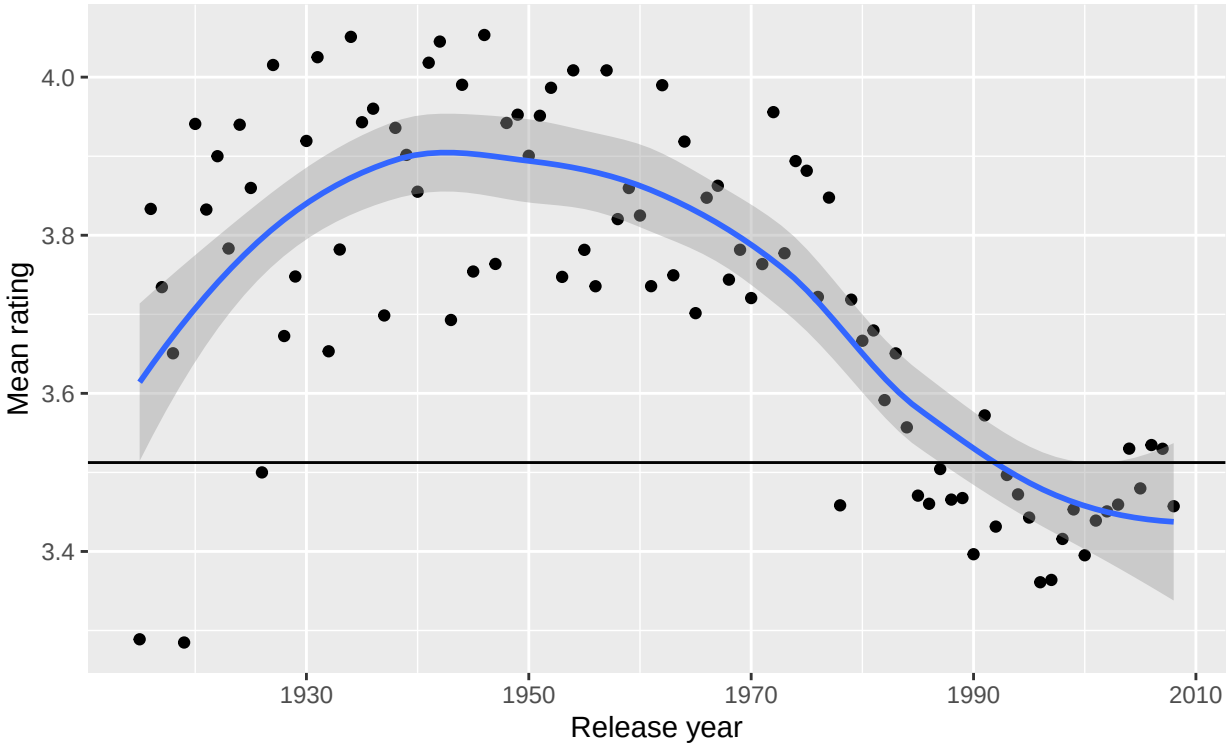
The RMSE was plotted against the span with degree 1 and 2 respectively:




```
params_year <- params[which.min(rmses),]
model_5_rmse <- min(rmses)
```

The plot shows a minimum at span = 0.7 and degree = 2. The minimum value of the RMSE at these parameters is 0.8648925.

The loess smoothing of rating average against year with these parameters on the edx data set follows the data quite well:



The release year effect looks large (in particular for movies released before 1980), but improves the RMSE only slightly in comparison to the previous model. This is maybe due to the fact that most of the movie ratings are for movies of the nineties and later, where the effect is small:

Method	**RMSE**
Just the average	1.0600722
Movie Effect Model	0.9436040
Movie + User Effects Model	0.8662377
Regularized Movie + User Effects Model (CV)	0.8655212
Regularized Movie + User + Date Effects Model (CV)	0.8649029
Regularized Movie + User + Date + Year Effects Model (CV)	0.8648925

To make sure the optimal λ value didn't change by adding date and year effects, even though regularization wasn't applied to them, a final λ tuning was performed:

```
set.seed(1994)

lambdas <- seq(4,6,0.1)
rmses <- sapply(lambdas, function(l){
  #fold1
  mu <- mean(fold1_train$rating)
  b_d <- fold1_train %>%
    group_by(date) %>%
    summarize(rating = mean(rating - mu)) %>%
    mutate(b_d = predict(loess(rating ~ as.numeric(date), data = .,
                               span = params_date[1], degree = params_date[2]))) %>%
    select(-rating)
  b_y <- fold1_train %>%
    left_join(b_d, by = "date") %>%
    group_by(year) %>%
```

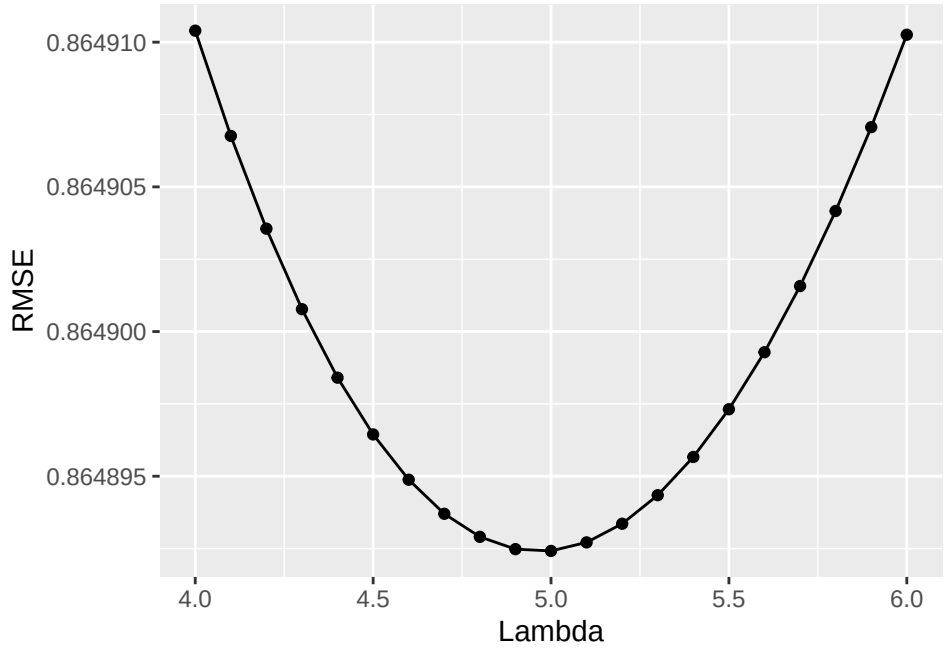
```
  summarize(rating = mean(rating - mu - b_d)) %>%
  mutate(b_y = predict(loess(rating ~ year, data = .,
                             span = params_year[1], degree = params_year[2]))) %>%
  select(-rating)
b_i <- fold1_train %>%
  left_join(b_d, by = "date") %>%
  left_join(b_y, by = "year") %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu - b_d - b_y)/(n()+l))
b_u <- fold1_train %>%
  left_join(b_d, by = "date") %>%
  left_join(b_y, by = "year") %>%
  left_join(b_i, by= "movieId") %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating - mu - b_d - b_y - b_i)/(n()+l))
predicted_ratings <- fold1_test %>%
  left_join(b_d, by = "date") %>%
  left_join(b_y, by = "year") %>%
  left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = "userId")  %>%
  mutate(pred = mu + b_d + b_y + b_i + b_u) %>%
  .$pred
rmse1 <- RMSE(predicted_ratings, fold1_test$rating)
#fold2
mu <- mean(fold2_train$rating)
b_d <- fold2_train %>%
  group_by(date) %>%
  summarize(rating = mean(rating - mu)) %>%
  mutate(b_d = predict(loess(rating ~ as.numeric(date), data = .,
                             span = params_date[1], degree = params_date[2]))) %>%
  select(-rating)
b_y <- fold2_train %>%
  left_join(b_d, by = "date") %>%
  group_by(year) %>%
  summarize(rating = mean(rating - mu - b_d)) %>%
  mutate(b_y = predict(loess(rating ~ year, data = .,
                             span = params_year[1], degree = params_year[2]))) %>%
  select(-rating)
b_i <- fold2_train %>%
  left_join(b_d, by = "date") %>%
  left_join(b_y, by = "year") %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu - b_d - b_y)/(n()+l))
b_u <- fold2_train %>%
  left_join(b_d, by = "date") %>%
  left_join(b_y, by = "year") %>%
  left_join(b_i, by= "movieId") %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating - mu - b_d - b_y - b_i)/(n()+l))
predicted_ratings <- fold2_test %>%
  left_join(b_d, by = "date") %>%
  left_join(b_y, by = "year") %>%
  left_join(b_i, by = "movieId") %>%
```

```
  left_join(b_u, by = "userId")  %>%
  mutate(pred = mu + b_d + b_y + b_i + b_u) %>%
  .$pred
rmse2 <- RMSE(predicted_ratings, fold2_test$rating)
#fold3
mu <- mean(fold3_train$rating)
b_d <- fold3_train %>%
  group_by(date) %>%
  summarize(rating = mean(rating - mu)) %>%
  mutate(b_d = predict(loess(rating ~ as.numeric(date), data = .,
                             span = params_date[1], degree = params_date[2]))) %>%
  select(-rating)
b_y <- fold3_train %>%
  left_join(b_d, by = "date") %>%
  group_by(year) %>%
  summarize(rating = mean(rating - mu - b_d)) %>%
  mutate(b_y = predict(loess(rating ~ year, data = .,
                             span = params_year[1], degree = params_year[2]))) %>%
  select(-rating)
b_i <- fold3_train %>%
  left_join(b_d, by = "date") %>%
  left_join(b_y, by = "year") %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu - b_d - b_y)/(n()+l))
b_u <- fold3_train %>%
  left_join(b_d, by = "date") %>%
  left_join(b_y, by = "year") %>%
  left_join(b_i, by= "movieId") %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating - mu - b_d - b_y - b_i)/(n()+l))
predicted_ratings <- fold3_test %>%
  left_join(b_d, by = "date") %>%
  left_join(b_y, by = "year") %>%
  left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = "userId")  %>%
  mutate(pred = mu + b_d + b_y + b_i + b_u) %>%
  .$pred
rmse3 <- RMSE(predicted_ratings, fold3_test$rating)
#fold4
mu <- mean(fold4_train$rating)
b_d <- fold4_train %>%
  group_by(date) %>%
  summarize(rating = mean(rating - mu)) %>%
  mutate(b_d = predict(loess(rating ~ as.numeric(date), data = .,
                             span = params_date[1], degree = params_date[2]))) %>%
  select(-rating)
b_y <- fold4_train %>%
  left_join(b_d, by = "date") %>%
  group_by(year) %>%
  summarize(rating = mean(rating - mu - b_d)) %>%
  mutate(b_y = predict(loess(rating ~ year, data = .,
                             span = params_year[1], degree = params_year[2]))) %>%
  select(-rating)
```

```
b_i <- fold4_train %>%
  left_join(b_d, by = "date") %>%
  left_join(b_y, by = "year") %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu - b_d - b_y)/(n()+l))
b_u <- fold4_train %>%
  left_join(b_d, by = "date") %>%
  left_join(b_y, by = "year") %>%
  left_join(b_i, by= "movieId") %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating - mu - b_d - b_y - b_i)/(n()+l))
predicted_ratings <- fold4_test %>%
  left_join(b_d, by = "date") %>%
  left_join(b_y, by = "year") %>%
  left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = "userId")  %>%
  mutate(pred = mu + b_d + b_y + b_i + b_u) %>%
  .$pred
rmse4 <- RMSE(predicted_ratings, fold4_test$rating)
#fold5
mu <- mean(fold5_train$rating)
b_d <- fold5_train %>%
  group_by(date) %>%
  summarize(rating = mean(rating - mu)) %>%
  mutate(b_d = predict(loess(rating ~ as.numeric(date), data = .,
                             span = params_date[1], degree = params_date[2]))) %>%
  select(-rating)
b_y <- fold5_train %>%
  left_join(b_d, by = "date") %>%
  group_by(year) %>%
  summarize(rating = mean(rating - mu - b_d)) %>%
  mutate(b_y = predict(loess(rating ~ year, data = .,
                             span = params_year[1], degree = params_year[2]))) %>%
  select(-rating)
b_i <- fold5_train %>%
  left_join(b_d, by = "date") %>%
  left_join(b_y, by = "year") %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu - b_d - b_y)/(n()+l))
b_u <- fold5_train %>%
  left_join(b_d, by = "date") %>%
  left_join(b_y, by = "year") %>%
  left_join(b_i, by= "movieId") %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating - mu - b_d - b_y - b_i)/(n()+l))
predicted_ratings <- fold5_test %>%
  left_join(b_d, by = "date") %>%
  left_join(b_y, by = "year") %>%
  left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = "userId")  %>%
  mutate(pred = mu + b_d + b_y + b_i + b_u) %>%
  .$pred
rmse5 <- RMSE(predicted_ratings, fold5_test$rating)
```

```
  #combination
  return((rmse1+rmse2+rmse3+rmse4+rmse5)/5)
})
```

The plot of the RMSE versus the λ values shows the same parabolic dependence as in the previous λ tuning:




```
final_bestlambda <- lambdas[which.min(rmses)]
model_6_rmse <- min(rmses)
```

The minimum of λ is 5, which means a slight shift occurred by addition of date and year effects. The minimum value of the RMSE at $\lambda = 5$ is 0.8648924. This result represents the RMSE of the final model of this project.

In comparison to the previous model, the final model provides only a tiny improvement on the RMSE:

Method	**RMSE**
Just the average	1.0600722
Movie Effect Model	0.9436040
Movie + User Effects Model	0.8662377
Regularized Movie + User Effects Model (CV)	0.8655212
Regularized Movie + User + Date Effects Model (CV)	0.8649029
Regularized Movie + User + Date + Year Effects Model (CV)	0.8648925
Final Regularized Movie + User + Date + Year Effects Model (CV)	0.8648924

Application of the final model on the validation set

The resulting final model with the optimized parameters (regularization: $\lambda = 5$; loess smoothing for `date`: span = 10 and degree = 2; loess smoothing for `year`: span = 0.7 and degree = 2) was trained on the edx set and tested on the validation set:

```
mu <- mean(edx$rating)
b_d <- edx %>%
  group_by(date) %>%
  summarize(rating = mean(rating - mu)) %>%
```

```
  mutate(b_d = predict(loess(rating ~ as.numeric(date), data = .,
                             span = params_date[1], degree = params_date[2]))) %>%
  select(-rating)
b_y <- edx %>%
  left_join(b_d, by = "date") %>%
  group_by(year) %>%
  summarize(rating = mean(rating - mu - b_d)) %>%
  mutate(b_y = predict(loess(rating ~ year, data = .,
                             span = params_year[1], degree = params_year[2]))) %>%
  select(-rating)
b_i <- edx %>%
  left_join(b_d, by = "date") %>%
  left_join(b_y, by = "year") %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu - b_d - b_y)/(n()+final_bestlambda))
b_u <- edx %>%
  left_join(b_d, by = "date") %>%
  left_join(b_y, by = "year") %>%
  left_join(b_i, by= "movieId") %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating - mu - b_d - b_y - b_i)/(n()+final_bestlambda))
predicted_ratings <- validation %>%
  left_join(b_d, by = "date") %>%
  left_join(b_y, by = "year") %>%
  left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = "userId")  %>%
  mutate(pred = mu + b_d + b_y + b_i + b_u) %>%
  .$pred
final_rmse <- RMSE(predicted_ratings, validation$rating)
```

The RMSE for the prediction of ratings in the validation set by means of the final model is 0.8641616. This result even surpasses the results of the development process:

Method	**RMSE**
Just the average	1.0600722
Movie Effect Model	0.9436040
Movie + User Effects Model	0.8662377
Regularized Movie + User Effects Model (CV)	0.8655212
Regularized Movie + User + Date Effects Model (CV)	0.8649029
Regularized Movie + User + Date + Year Effects Model (CV)	0.8648925
Final Regularized Movie + User + Date + Year Effects Model (CV)	0.8648924
Final Model on Validation Set	**0.8641616**

Conclusion

The goal of this project was the development of a machine learning algorithm that predicts outcomes in the validation subset of a data set with about ten million movie ratings. The predetermined threshold to beat was an RMSE of 0.86490. In this work several linear models based on each other were trained, resulting in improved RMSEs for every additional effect. For the final model on the validation set an RMSE of 0.8641616 was achieved.

The given code from the machine learning course uses a linear model approach to build an algorithm step by step by adding a movie-specific effect and an user-specific effect to a naive model based on the average of the

ratings. After regularization, this model already offers a good prediction of unknown outcomes. However, the modular structure of this approach allows for the addition of further effects to the model. Here, an effect for the rating date and an effect for the release year of the movies was added. Additionally, a cross-validation procedure was added to improve the stability of the estimates. These additions were sufficient to boost the RMSE under the required threshold.

Further improvements to this approach could be achieved by adding additional effects such as a genre effect or an effect of rating counts per movie for instance. There are also more complicated techniques possible such as matrix factorization, which takes similar rating patterns for groups of users and groups of movies into account. Due to hardware and time limitations, these additions were not considered in this work.