Final EdX Project - Movielens

Daniel Verdon

2023-02-12

Objective/Introduction

The objective is to create a Movie recommendation system.

The data set contains **10000054 ratings** and **95580 tags** applied to **10681 movies** by **71567 users** of the online movie recommender service **MovieLens**.

Users were selected at random for inclusion.

Column names are

- **userID** = unique user
- **movieID** = unique movie Id
- **rating** = rating that was given by user for the movie
- **timestamp** = timestap when rating was given by user
- **title** = title of the movie
- **genres** = the various genres of the movie

We will try to predict the rating based on columns **userID**, **movieID** and **genres**.

First step is to create the two dataframes from the **ml-10M100K.zip** file.

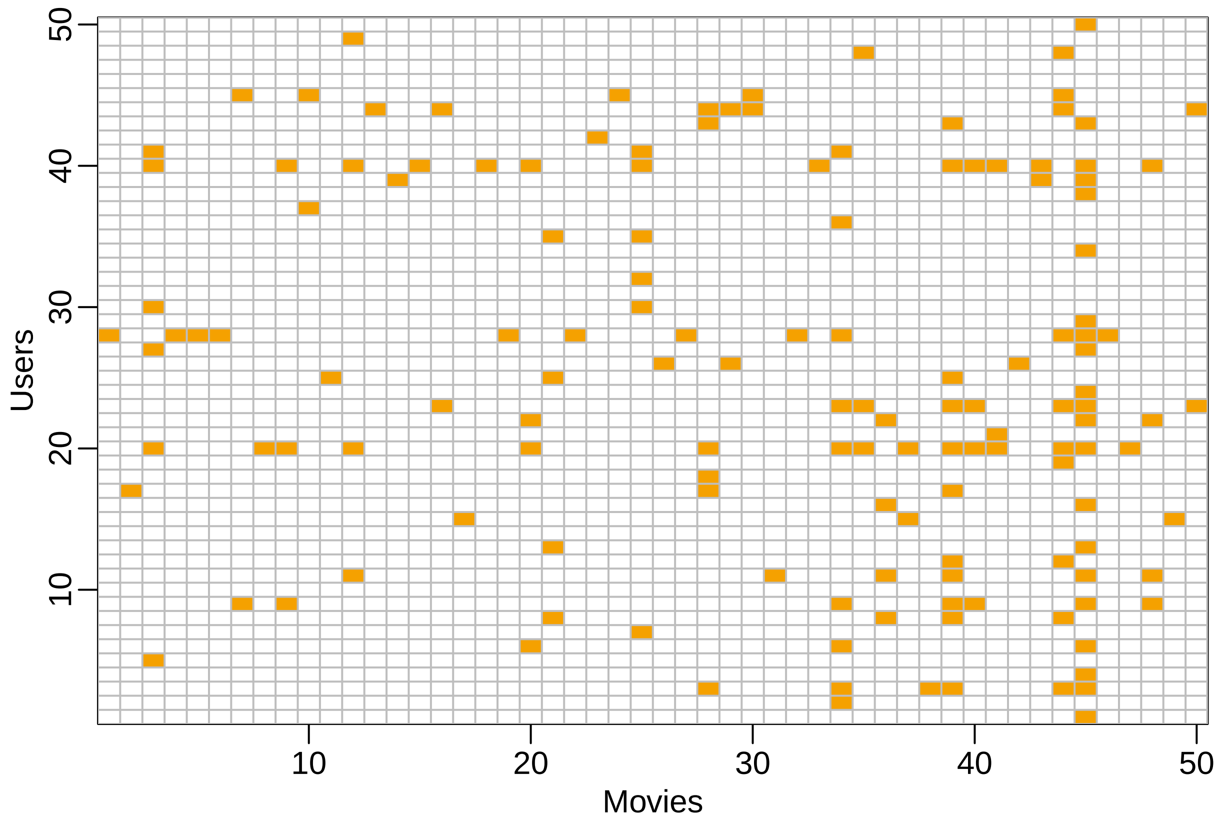
- **edx** (to be used to train our algorithm)
- **final_holdout_test** (to be used to test our algorithm)

We then train 4 models and determine which is the best predictor based on lowest RMSE.

1. **Movie Effect Model**
2. **Movie + User Effects Model**
3. **Movie + User + Genre Effects Model**
4. **Regularized Movie + User + Genre Effects Model**

Analysis

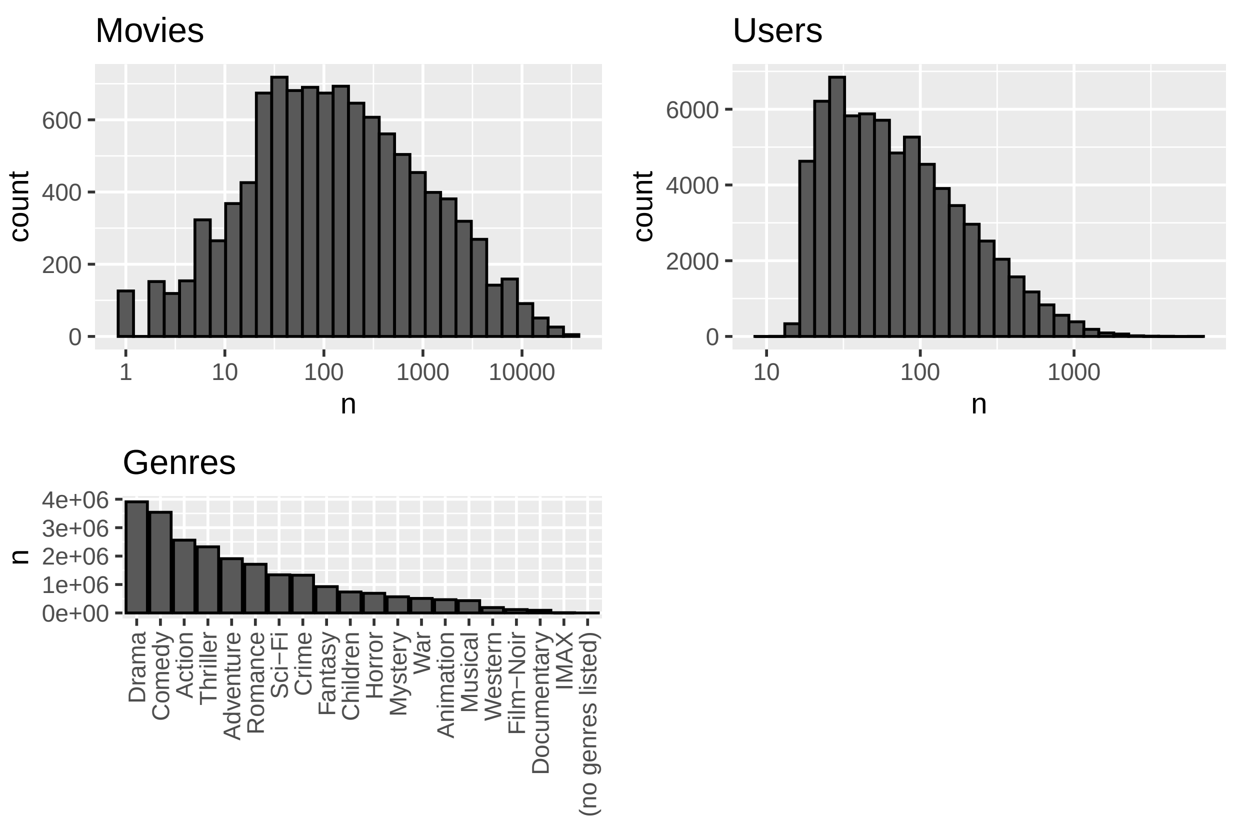
Lets visualise the objective. We create a matrix of a random sample of 50 movies and 50 users to see how sparse it is. The yellow dots represent the user/movie combination for which a rating is available



All the blank cells represent the ratings we are trying to predict.

Lets take a look at some some of the distributions

```
library(gridExtra)
p1 <- edx %>%
  dplyr::count(movieId) %>%
  ggplot(aes(n)) +
  geom_histogram(bins = 30, color = "black") +
  scale_x_log10() +
  ggtitle("Movies")

p2 <- edx %>%
  dplyr::count(userId) %>%
  ggplot(aes(n)) +
  geom_histogram(bins = 30, color = "black") +
  scale_x_log10() +
  ggtitle("Users")
p3 <- edx %>%
  separate_rows(genres, sep = "\\|") %>%
  dplyr::count(genres) %>%
  ggplot(aes(y=n, reorder(x=genres,-n,sum))) +
  geom_col( color = "black") +
  ggtitle("Genres") +
  xlab("") +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))
grid.arrange(p1, p2, p3, ncol = 2)
```



We see the number of **Movies** rated follows a normal distribution. The **Users** plot follows a *Poisson* distribution meaning we see a lot of users who have rated 50 movies then it tails off with few users rating 1000 or more movies. Looking at **Genres**, we see *Drama* and *Comedies* have the most ratings (note that films often have more than one genre)

Loss function: To compare different models or to see how well we're doing compared to some baseline, the typical error loss, the residual mean squared error (RMSE) is used on the test set. We can interpret RMSE similar to standard deviation.

If N is the number of user-movie-genre combinations, $y_{u,i,g}$ is the rating for movie i, movie genre g by user u , and $\hat{y}_{u,i,g}$ is our prediction, then RMSE is defined as follows:

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

we want to create a function to test the RMSE of our model

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

Model and results

The model we will create will be based on three predictors.

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

We start with assuming the same rating for all movies and all users. In this case, the *least squares estimate* of u — the estimate that minimizes the root mean squared error — is the average rating of all movies across all users.

- b_i represents the average rating for a movie i
- b_u represents the average rating a user has given u
- b_g represents the average rating for a genre g

Note that because there are thousands of b's, the lm() function will be very slow or cause R to crash, so we will not use linear regression to calculate these effects.

Movie effects

We know some movies are rated higher than others. Lets see the *least squares estimate* if we just take the same rating for all movies and the average rating b_i for movie i.

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

The least squares estimate $\hat{b}_i$ is just the average of $Y_{u,i} - \hat{\mu}$ for each movie i. So we can compute them as shown below (Note that to make the code look cleaner, we are not using the *hat* notation from this point onwards):

```r
mu <- mean(edx$rating)
movie_avgs <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))
```

Lets how this improves our RMSE by predicting the *final_holdout_test* ratings with the model that we just fit

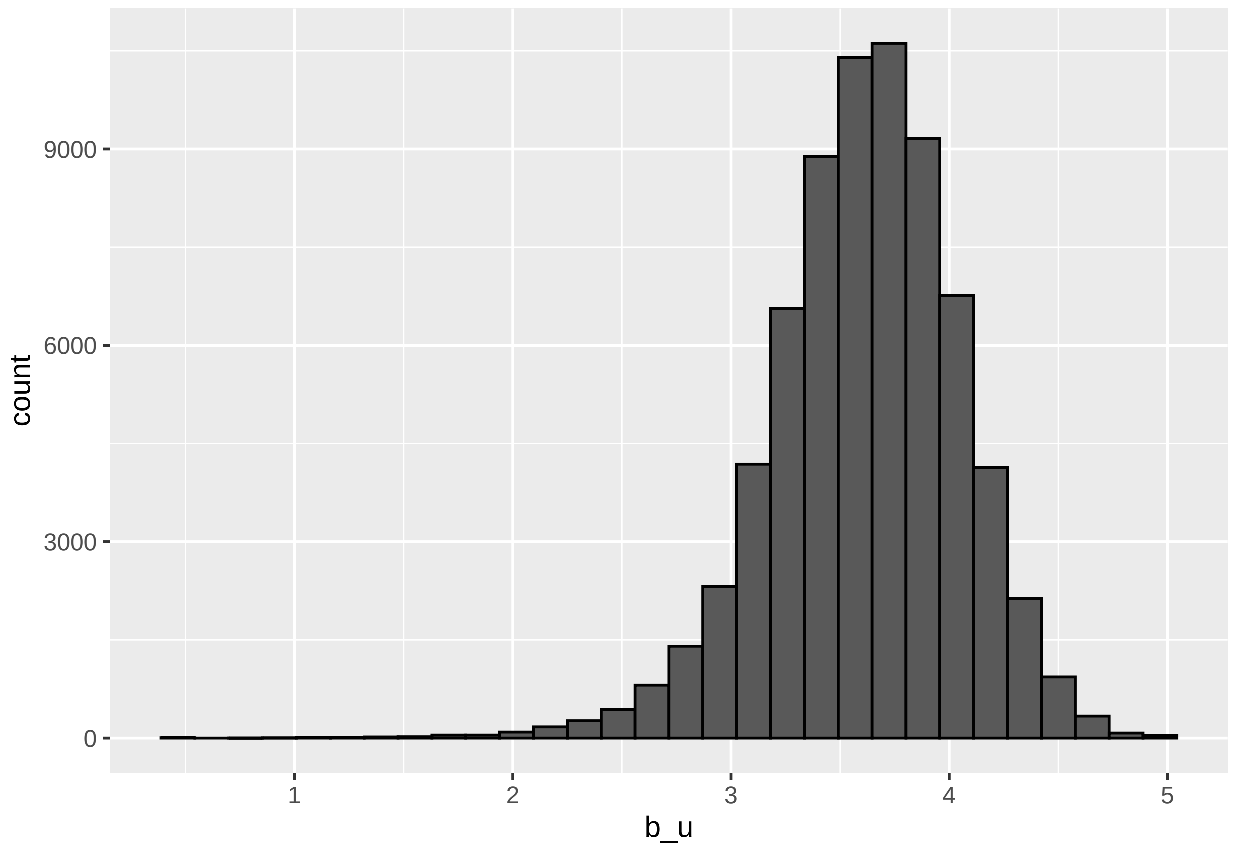
```r
predicted_ratings <- mu + final_holdout_test %>%
  left_join(movie_avgs, by='movieId') %>%
  pull(b_i)

model_1_rmse <- RMSE(predicted_ratings, final_holdout_test$rating)
rmse_results <- data_frame(method = "Movie Effect Model", RMSE = model_1_rmse)

rmse_results %>% knitr::kable()
```

method	RMSE
Movie Effect Model	0.9439087

Modeling user effects

Using the code below and looking at the plot we see there is *a lot of variability across users* which means we can add it to our model

```
edx  %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating)) %>%
  filter(n()>=100) %>%
  ggplot(aes(b_u)) +
  geom_histogram(bins = 30, color = "black")
```



Our new model is

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

and we follow the same method of calculating approximatation as average of $\hat{y}_{u,i} - \hat{u} - \hat{b}_i$

```
user_avgs <- edx %>%
  left_join(movie_avgs, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_i))

predicted_ratings <- final_holdout_test %>%
  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, final_holdout_test$rating)
```

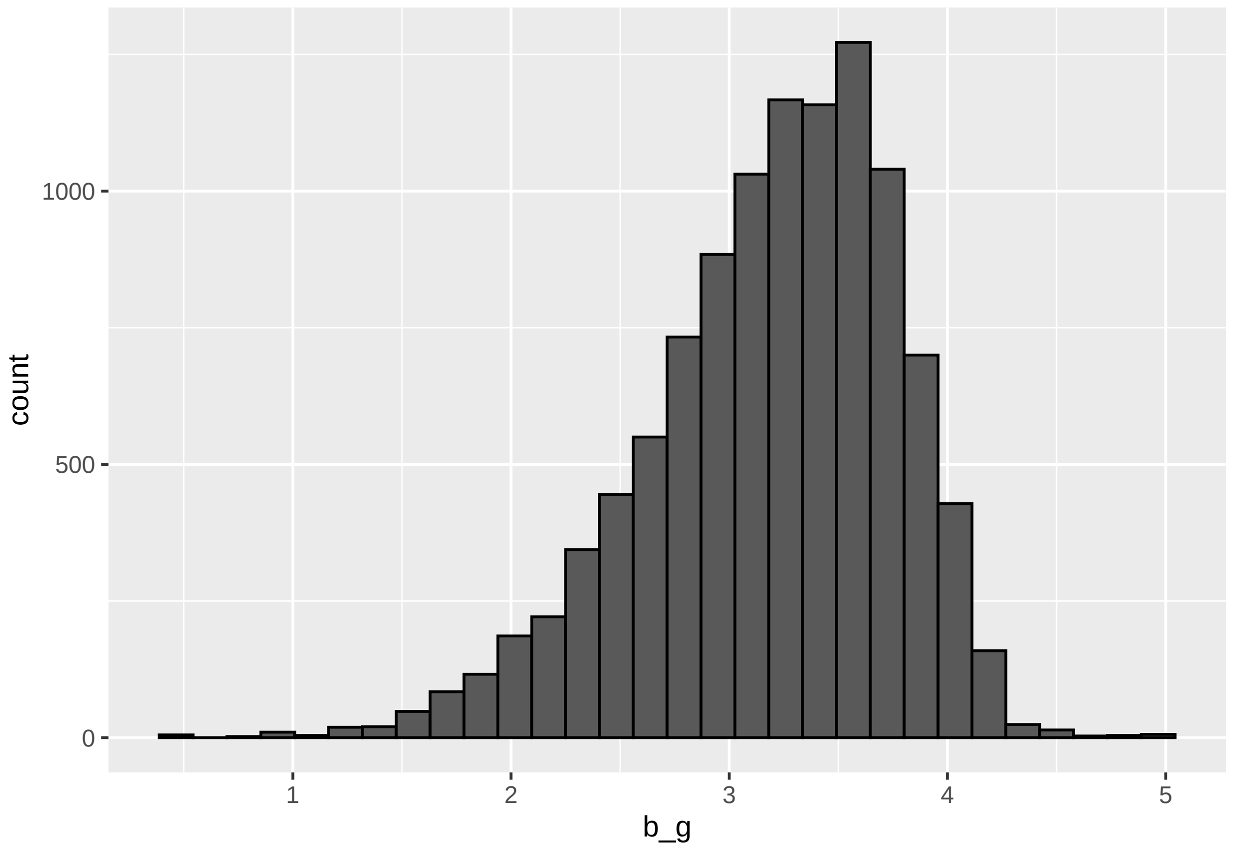
```r
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Movie + User Effects Model",
                                     RMSE = model_2_rmse ))
rmse_results %>% knitr::kable()
```

method	RMSE
Movie Effect Model	0.9439087
Movie + User Effects Model	0.8653488

Modeling user effects

As we see there is also a *lot of variability* across genres which means we can add it to our model

```r
edx %>% separate_rows(genres, sep = "\\|") %>%
  group_by(movieId) %>%
  summarize(b_g = mean(rating)) %>%
  filter(n()>=100) %>%
  ggplot(aes(b_g)) +
  geom_histogram(bins = 30, color = "black")
```



Our new model is

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

and we follow the same method of calculating approximatation as average of $\hat{y}_{u,i,g} - \hat{u} - \hat{b}_i - \hat{b}_g$

```r
genres_avgs <- edx %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  separate_rows(genres, sep = "\\|") %>%
  group_by(movieId) %>%
  summarize(b_g = mean(rating - mu - b_i - b_u))

predicted_ratings <- final_holdout_test %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  left_join(genres_avgs, by='movieId') %>%
  mutate(pred = mu + b_i + b_u + b_g) %>%
  pull(pred)

model_3_rmse <- RMSE(predicted_ratings, final_holdout_test$rating)

rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Movie + User + Genre Effects Model",
                                     RMSE = model_3_rmse ))
rmse_results %>% knitr::kable()
```

method	RMSE
Movie Effect Model	0.9439087
Movie + User Effects Model	0.8653488
Movie + User + Genre Effects Model	0.8640972

Consideration for Regularization

Regularization penalizes large estimates that are formed using small sample sizes, it also helps to reduce the possibility of overfitting and help us obtain an optimal model. We will see if our model improves. Using the code below I ran the code for several lambdas however it takes a very long time (provided it doesn't crash). I ran the code and can say the λ that minimises the RMSE is 5.25.

```r
#Advise not to run this code as it takes a very long time. Use code in next block.
lambdas <- seq(0,10, 0.25)

rmses <- sapply(lambdas, function(l){
  mu <- mean(edx$rating)
  b_i <- edx %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating-mu)/(n()+l))
  b_u <- edx %>%
    left_join(b_i, by="movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - b_i-mu)/(n()+l))
  b_g <- edx |>
    left_join(b_i, by="movieId") |>
    left_join(b_u, by="userId") |>
  separate_rows(genres, sep = "\\|") |>
    group_by(movieId) %>%
    summarize(b_g = mean(rating - mu - b_i - b_u)/(n()+1))
  predicted_ratings <-
    final_holdout_test %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    left_join(b_g, by = "movieId") |>
    mutate(pred = mu + b_i + b_u +b_g) %>%
    pull(pred)
  return(RMSE(predicted_ratings, final_holdout_test$rating))
})

qplot(lambdas, rmses)

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

So instead of running the code 40 times to find λ we run the below code

```
rmse <- sapply(5.25, function(l){
  mu <- mean(edx$rating)
  b_i <- edx %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating-mu)/(n()+l))
  b_u <- edx %>%
    left_join(b_i, by="movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - b_i-mu)/(n()+l))
  b_g <- edx %>%
    left_join(b_i, by="movieId") %>%
    left_join(b_u, by="userId") %>%
  separate_rows(genres, sep = "\\|") %>%
    group_by(movieId) %>%
    summarize(b_g = mean(rating - mu - b_i - b_u)/(n()+1))
  predicted_ratings <-
    final_holdout_test %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    left_join(b_g, by = "movieId") |>
    mutate(pred = mu + b_i + b_u +b_g) %>%
    pull(pred)
  return(RMSE(predicted_ratings, final_holdout_test$rating))
})

rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Regularized Movie + User + Genre Effect Model",
                                     RMSE = rmse))
rmse_results %>% knitr::kable()
```

method	RMSE
Movie Effect Model	0.9439087
Movie + User Effects Model	0.8653488
Movie + User + Genre Effects Model	0.8640972
Regularized Movie + User + Genre Effect Model	0.8648100

Conclusion

As we can see Regularization did not improve the model, most likely due to Genres having a Poisson distribution.However we may still want to use it to prevent risk of overfitting and **RMSE 0.86468100** is close to the best RMSE. The **Movie + User + Genres Effect Model** gave us the best **RMSE 0.8640972** but again, we may prefer to go with the regularized model.

There are a number of areas we could look into to better our model and improve the RMSE.

1. Although regularization did not produce the lowest RMSE, we could look at adapting our model to regularize just movies and not; movies, users and genres. This should improve our RMSE.
2. We were given a timestamp when the User submitted their rating and when converting to Date we see the range is from 1995 to 2009, so 14 years (see code below). A user preference often changes over time and movies can become very dated, this would lead to worsening reviews as time goes by. We could add this as a predictor where more recent predictions are given more weighting.

```
## Loading required package: anytime
```

```
edx |> mutate(Date=anydate(timestamp)) |>
       select(Date) |>
       summary() |>
       knitr::kable()
```

Date
Min. :1995-01-09
1st Qu.:2000-01-02
Median :2002-10-24
Mean :2002-09-21
3rd Qu.:2005-09-15
Max. :2009-01-05

Thank you for the attention. This was fun.