Movie Recommender - HarvardX Capstone

David Edelman

5/3/2019

Executive Summary

People are always wanting to know if they should see a certain movie, whether it is new in the theaters, or an older movie available on other media. However, individual tastes widely vary when it comes to enjoyment of movies. Movie reviews and ratings often inform this decision, so by using a vast accumulation of movie reviews from individuals, we can create a algorithm to predict how a certain individual would rate the movie, and hence their level of enjoyment.

We will take almost 10 million movie reviews from individuals that logged a movie review on the site MovieLens (http://movielens.org (http://movielens.org)), explore the data to see what insights we can glean from it, and using those insights construct a prediction algorithm. We will test the predictive power against a validation set of 1 million records by using the Root Mean Square Error (RMSE) metric.

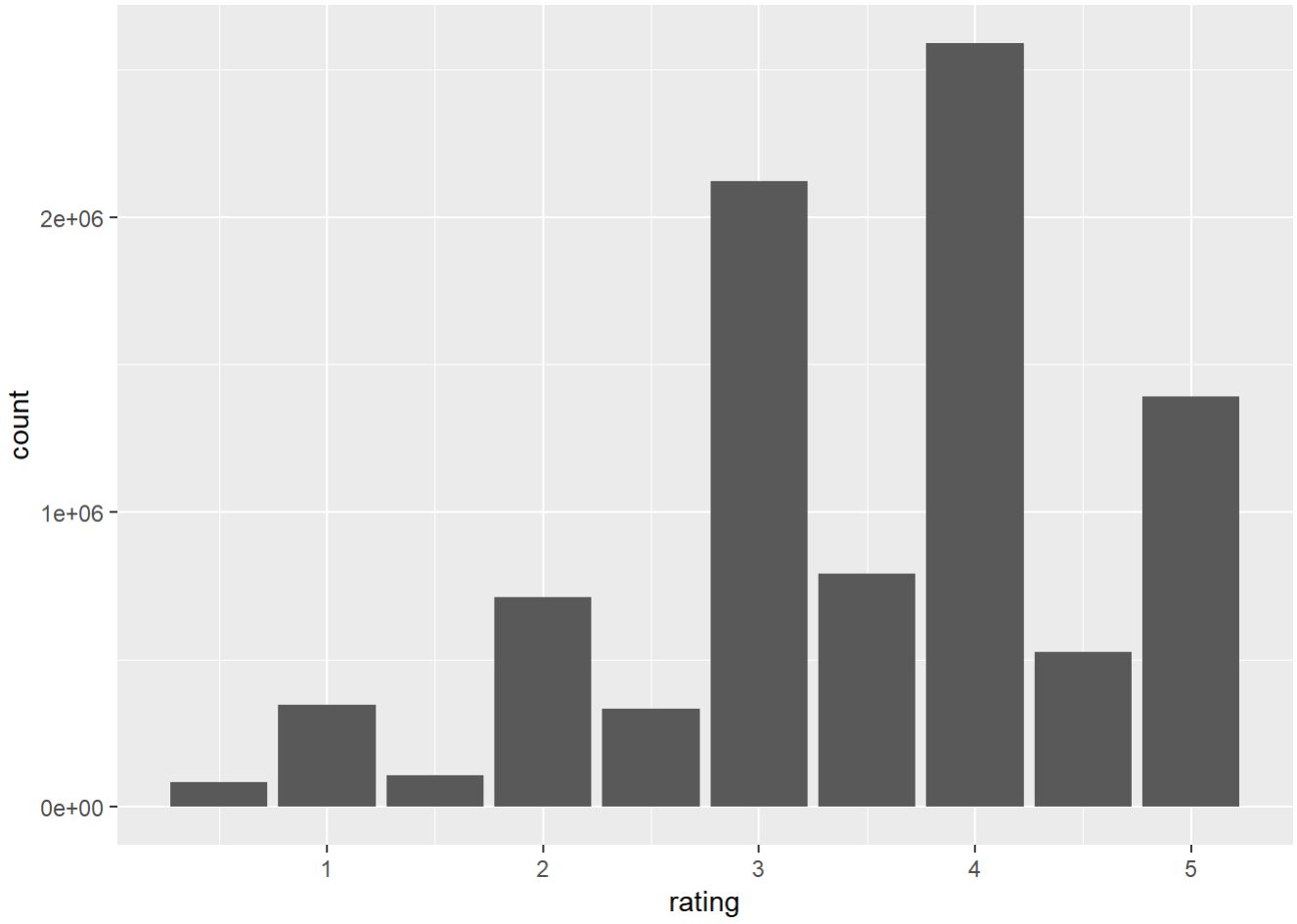
Data Exploration and Wrangling

The data set we are using is relatively simple. There are 5 predictors along with the “rating” dependent variable:

- userId - an integer identifier for each unique user; the value has no inherent meaning
- movieId - an integer identifier for each unique movie; the value has no inherent meaning
- timestamp - represents the date and time of the rating by storing the number of seconds since 1 January 1970
- title - the title of the movie, including the year of release
- genre - a list of all of the genres that MovieLens.org ascribed to the individual movie, listed alphabetically, concatenated by the pipe character (|)
- rating - the dependent variable; a set of numbers from 0.5 to 5, incrementing by 0.5

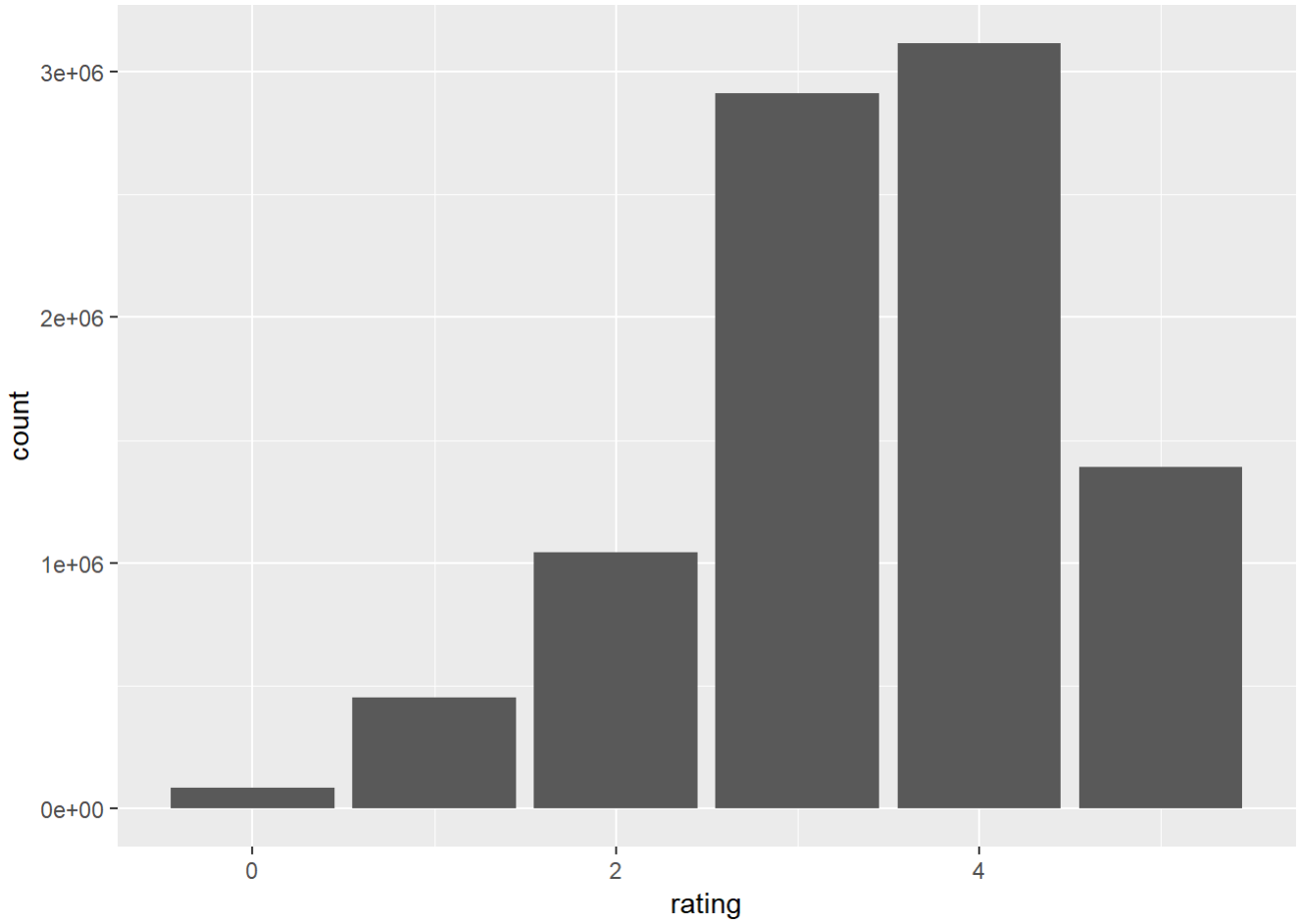
Ratings

```
edx %>% ggplot(aes(x=rating))+geom_histogram(stat="count")
```



A distribution of the number of reviews per rating shows that most ratings are 4 stars, followed by 3 stars. The “1/2 star” ratings are uncommon, so for the purposes of looking at a distribution, we will round the 1/2 stars ratings down.

```
edx %>% mutate(gp_rating = floor(rating)) %>%
  ggplot(aes(x=gp_rating))+geom_histogram(stat="count")+
  xlab("rating")
```



The distribution appears approximately normal with a negative skew. The mean rating ***mu*** is 3.51. Since the distribution is not continuous, median and standard deviation are not very meaningful.

Genres

There are 797 unique combinations of genres in the dataset:

```
dim(table(edx$genres))
```

```
## [1] 797
```

We will split the pipe-concatenated genres into individual genres, but to do this we need to know what the largest concatenated string is.

```
#Find the longest genre entry
g <- names(table(edx$genres))
g[which.max(str_length(g))]
```

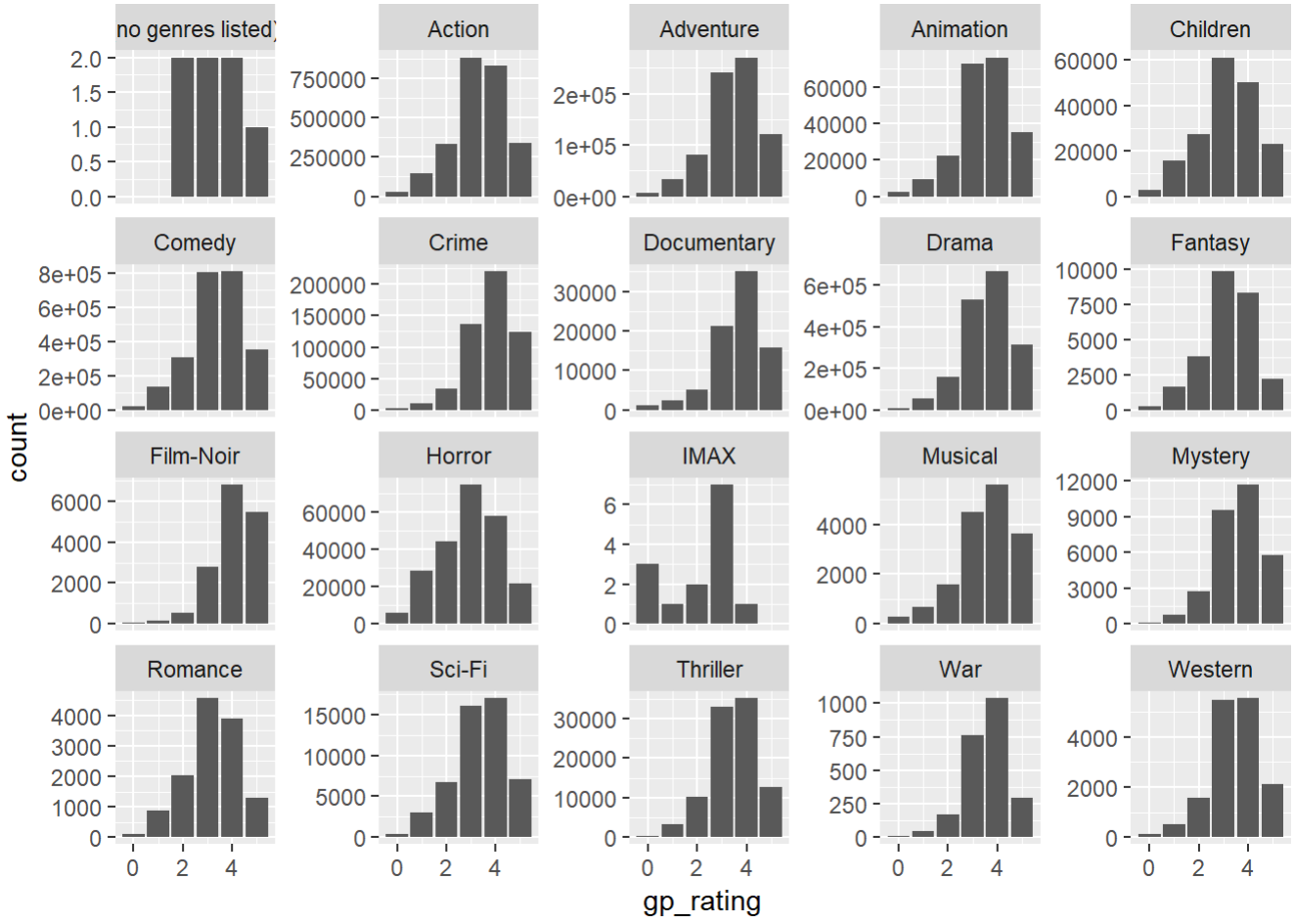
```
## [1] "Action|Adventure|Comedy|Drama|Fantasy|Horror|Sci-Fi|Thriller"
```

With a maximum of 8 genres concatenated together, we can now split the genre column into 8 separate columns, each with one genre (“NA” will be put in place for each missing column value).

```
#Since the longest has 8, we will split into 8 columns
#(NA will populate where there are not 8 genres)
genres <- c("genre1", "genre2", "genre3",
            "genre4", "genre5", "genre6",
            "genre7", "genre8")

edx <- edx %>% separate(genres, into=genres,
                        sep="[|]", fill="right", remove=FALSE, extra="merge")
```

To get an idea of ratings distribution across the genres, we plot histograms for each genre, using the rating of the first genre alphabetically (genre1) as a suitable approximation for the mean across all movies containing that genre. (rounding the 1/2 star ratings down):

```
edx %>% mutate(gp_rating = floor(rating)) %>%
  ggplot(aes(x=gp_rating))+geom_histogram(stat="count")+facet_wrap(~genre1, scales="free_y")
```



Just about all of the genres show a vast majority of ratings are 3 thru 4.5, so we may not see much benefit to this feature by itself in the prediction algorithm. However, there may be ways to use it in combination with other features.

Dates

Year of Review/Rating

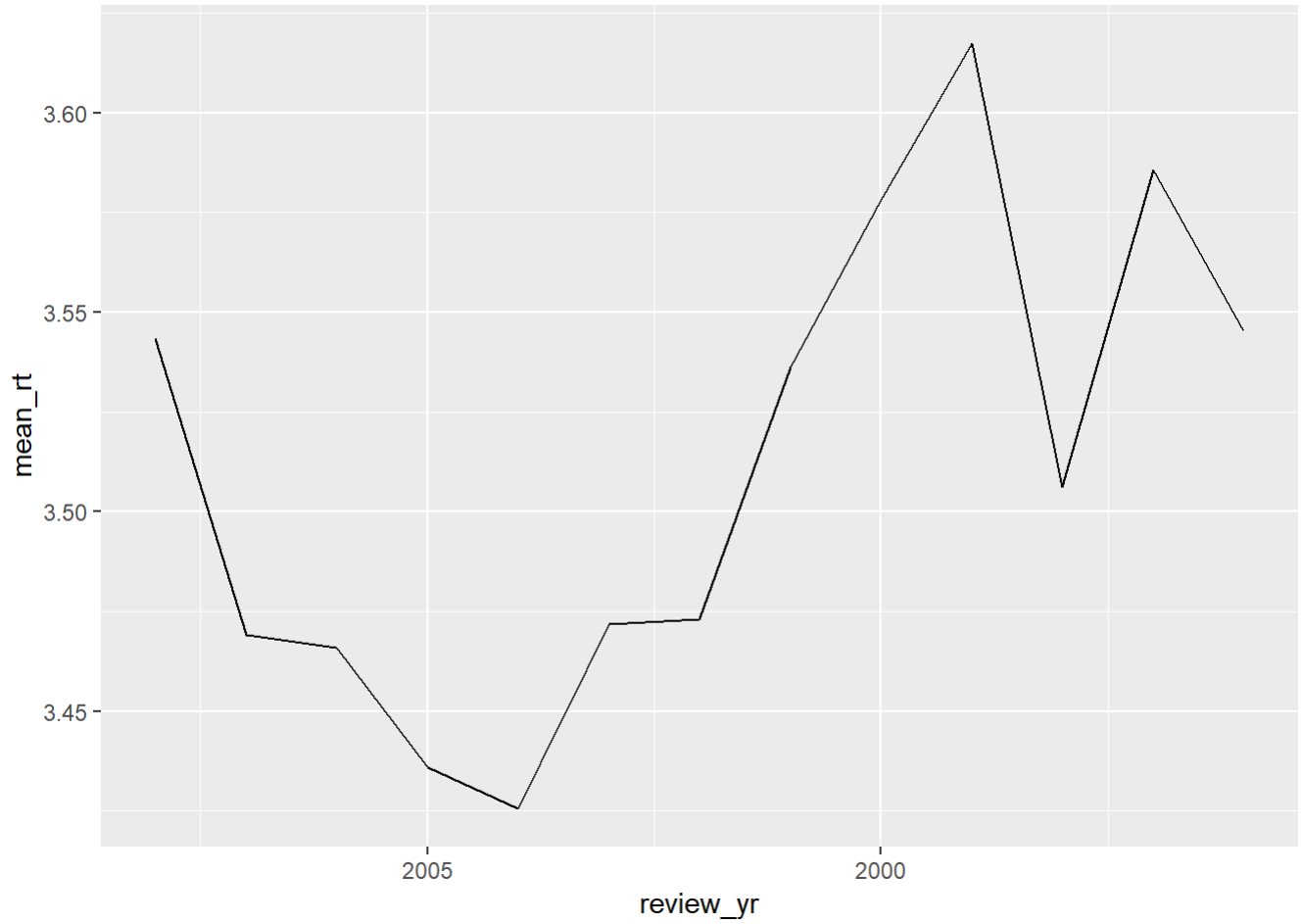
The year in which the user created the review may be of use for exploration and prediction. We can use the “timestamp” variable to extract a year. For this data set, “timestamp” represents the number of seconds elapsed since 1 Jan 1970, so we will convert to a date and extract the year.

```
#Add review year
edx <- edx %>%
  mutate(review_yr = year(as.Date(((edx$timestamp/60)/60)/24,
                                  origin="1970-1-1")))

table(edx$review_yr) %>% as.data.frame() %>% arrange(desc(Freq)) %>% grid.table()
```

	Var1	Freq
1	2000	1144349
2	2005	1059277
3	1996	942772
4	1999	709893
5	2008	696740
6	2004	691429
7	2006	689315
8	2001	683355
9	2007	629168
10	2003	619938
11	2002	524959
12	1997	414101
13	1998	181634
14	2009	13123
15	1995	2

As we see, all years have more than 100,000 ratings exccept for the first and last years of the range. Now lets look at the mean rating for each review year that is not 1995 or 2009. We plot as a line chart simply to make the trend more visible, and reverse the x-axis to show as the “age” of the review.

```
edx %>% filter(review_yr != 1995 & review_yr != 2009) %>%
  group_by(review_yr) %>% summarize(mean_rt = mean(rating)) %>%
  ggplot(aes(x=review_yr, y=mean_rt))+geom_line()+scale_x_reverse()
```



As we see from the plot, the mean rating is between approximately 3.425 and 3.620. If there is a trend, it appears to be a slightly increasing mean as the ratings get older.

Date of Movie Release

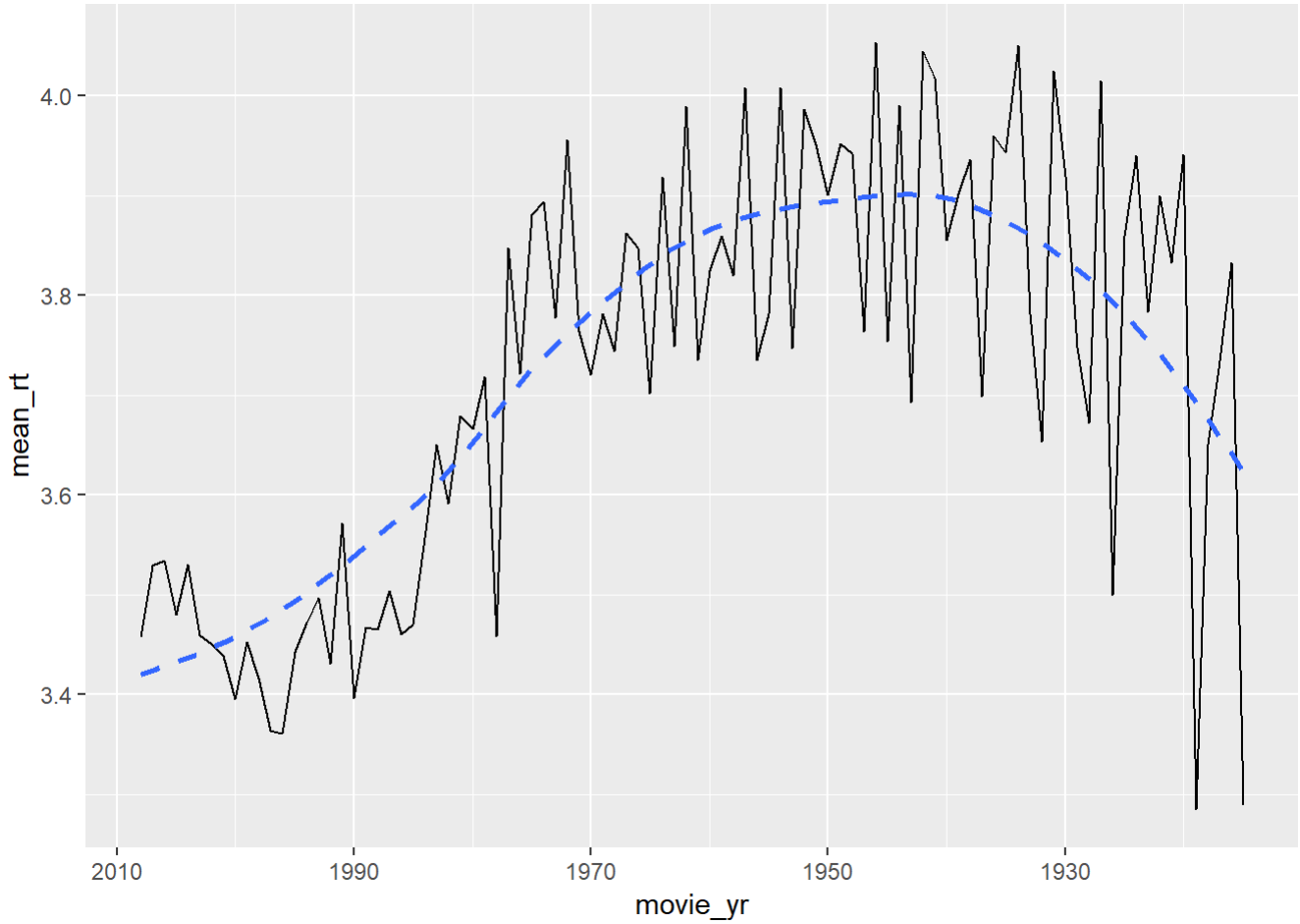
As with the year of the rating, the year the movie was released may have some exploratory or predictive value. The “title” variable contains the year, so we will need to extract it. Fortunately, the year of release is at the end of the title, enclosed in parenthesis. This allows for an easy extraction, although a regular expression technique could also be used. The year value starts at the 5th string element from the end. We then have to strip off the close parenthesis before it can be converted to a number. Since we still have the title in the data set, we do not need the title with the year stripped off.

```
edx <- edx %>% separate(title, into=c("ttl", "movie_yr"), sep= -5,
                        remove=FALSE, fill="right", extra="merge")

edx <- edx %>% mutate(movie_yr = as.numeric(sub(")", "", edx$movie_yr)))
edx <- edx %>% select(-ttl)

range(edx$movie_yr)
```

```
## [1] 1915 2008
```

We see that there are movies from as long ago as 1915, and as recent as 2008. Like the year of review, we will look at the mean rating per year of release, again with a “reversed” x-axis.

```
edx %>%
  group_by(movie_yr) %>% summarize(mean_rt = mean(rating)) %>%
  ggplot(aes(x=movie_yr, y=mean_rt))+geom_line()+scale_x_reverse()+
  geom_smooth(linetype = 2, se=FALSE)
```



There appears to be a kind of trend where newer movies are rated lower and the mean rating increases until it peaks around 1950 then falls off again. But since the actual line looks more like a seismograph readout (high variability year to year), with the mean bouncing all over the place, calling the deviations “trends” may be over-simplifying things.

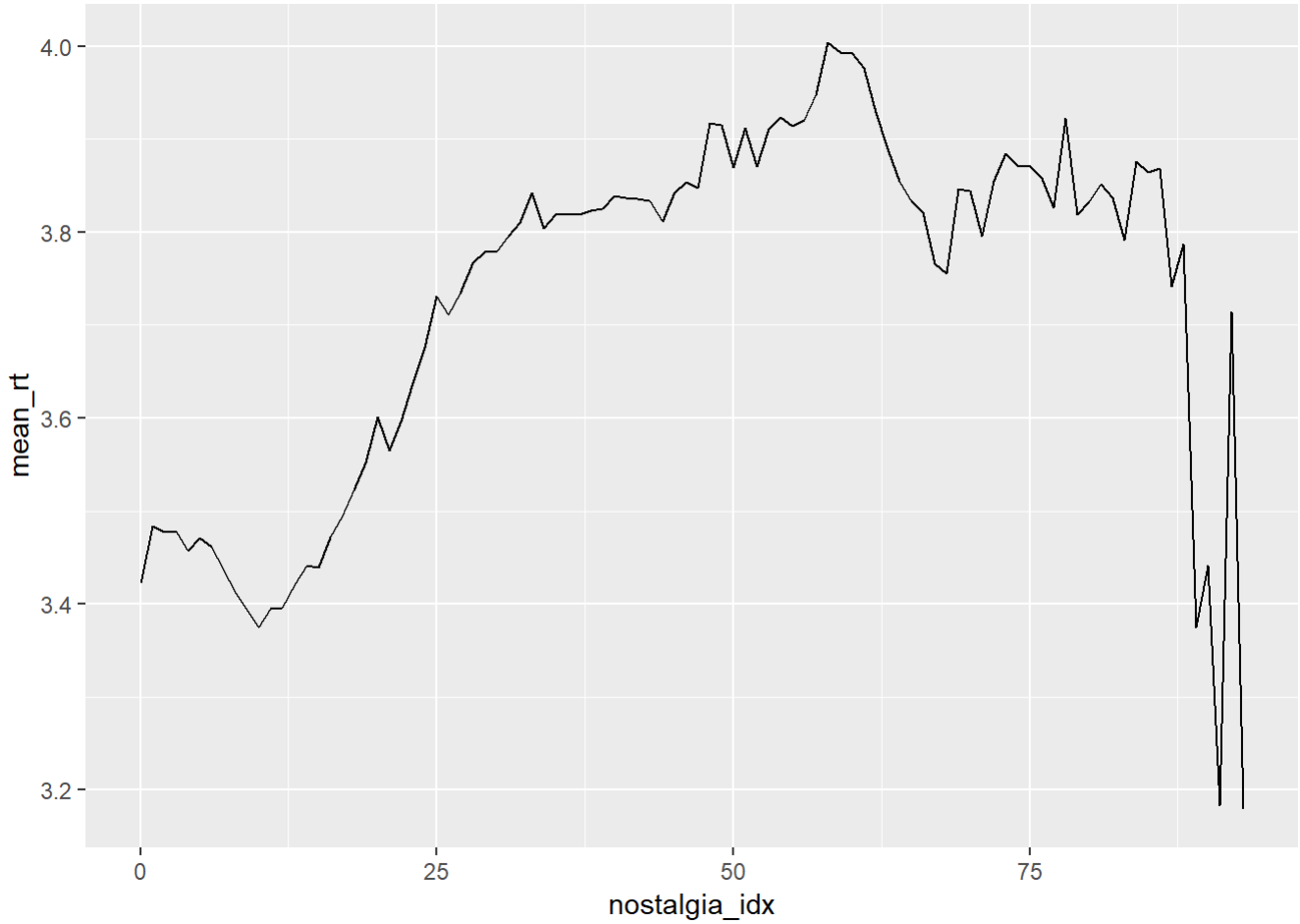
“Nostalgia” Index

Finally, we calculate a “nostalgia index”, defined as the time between release of the movie and the individual review. The thinking is that older movies, even ones within the range of the review years, may be rated differently than more recent movies.

```
#Calculate years between movie release and review; negative numbers are assumed
#to be data errors and will be set to 0
edx <- edx %>% mutate(nostalgia_idx = ifelse(review_yr < movie_yr,
                                             0,
                                             review_yr - movie_yr))
```

Like movie year and review year, we are interested in the mean rating for a film with a given “nostalgia index”

```
edx %>% group_by(nostalgia_idx) %>% summarize(mean_rt = mean(rating)) %>%
  ggplot(aes(x=nostalgia_idx, y=mean_rt))+geom_line()
```



The curve looks similar to the movie year, although with less overall variation year to year. This may be a good candidate to represent the age of a movie in the prediction algorithm.

User and Genre

We have split the single “genres” column into its constituent genres (genre1 thru genre8), and looked at a sample distribution, but we should explore further and look at the mean rating per genre, for every movie that is assigned to that genre. The genres are listed alphabetically, and not in some order indicating that a movie is more, for example, “Comedy” than it is “Horror”; because of this we need to comb through all of the movies and find which have been assigned to each genre. We previously stored the list of genres in *all_genres*:

```
all_genres
```

```
##  [1] "(no genres listed)" "Action"             "Adventure"
##  [4] "Animation"          "Children"           "Comedy"
##  [7] "Crime"              "Documentary"        "Drama"
## [10] "Fantasy"            "Film-Noir"          "Horror"
## [13] "IMAX"               "Musical"            "Mystery"
## [16] "Romance"            "Sci-Fi"             "Thriller"
## [19] "War"                "Western"
```

We define a function to find the mean across the population that has each genre as one of its listed genres, then apply that function for each genre.

```
find_genre_mean <- function(x) {
  tmp <- edx %>% mutate( gr = ifelse (genre1 == x | genre2 == x |
                              genre3 == x | genre4 == x |
                              genre5 == x | genre6 == x |
                              genre7 == x | genre8 == x, x, NA) )
  tmp %>% filter(! is.na(gr)) %>% group_by(gr) %>% summarize(mean_rt = mean(rating)) %>% .$mean_
rt
}

gr <- sapply(all_genres, find_genre_mean)
grd <- data.frame(genre=names(gr), mean_rt = gr, stringsAsFactors = FALSE)
grd %>% arrange(genre) %>% grid.table(theme=ttheme_default(base_size = 8))
```

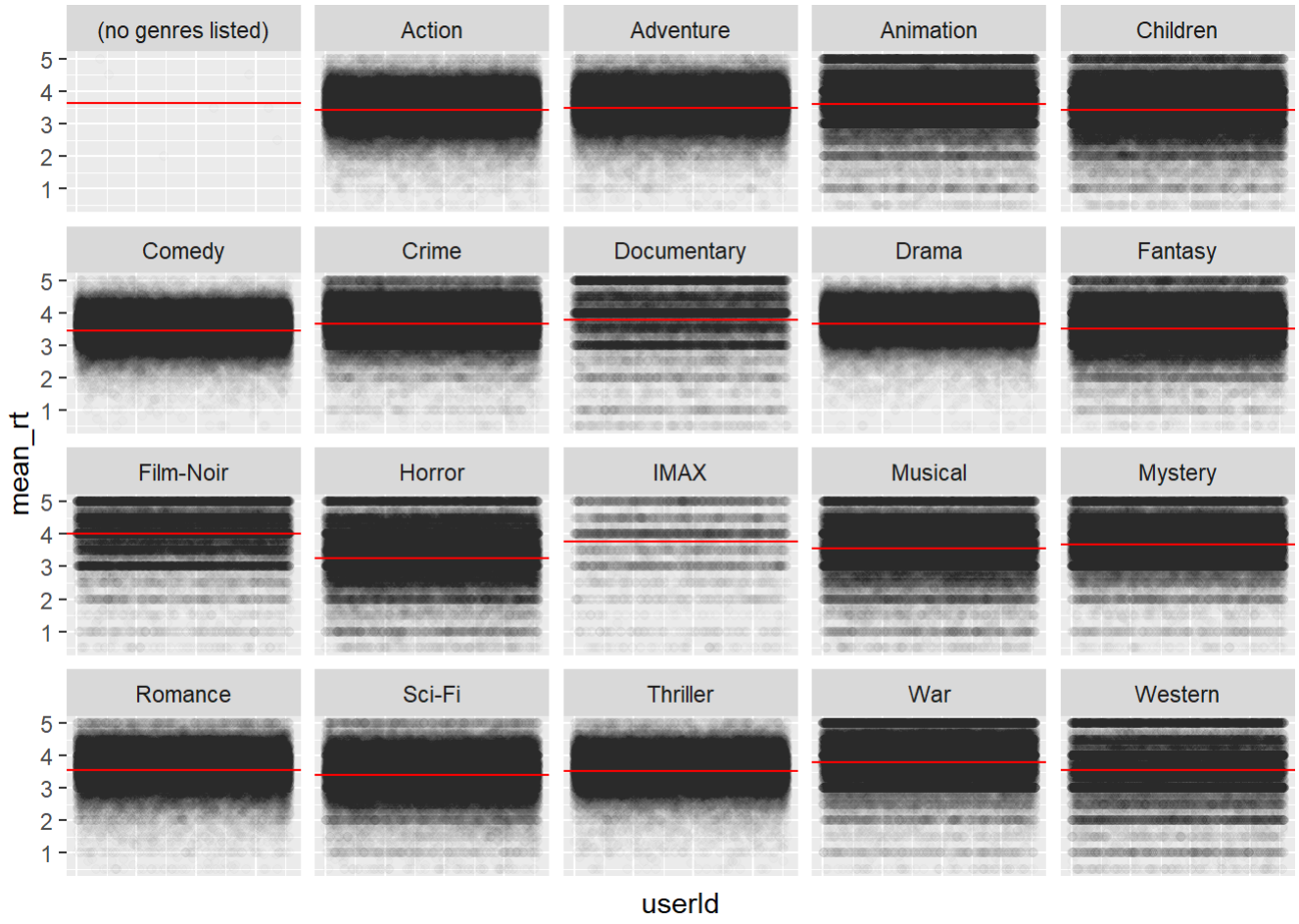
	genre	mean_rt
1	(no genres listed)	3.642857
2	Action	3.421405
3	Adventure	3.493544
4	Animation	3.600644
5	Children	3.418715
6	Comedy	3.436908
7	Crime	3.665925
8	Documentary	3.783487
9	Drama	3.673131
10	Fantasy	3.501946
11	Film-Noir	4.011625
12	Horror	3.269815
13	IMAX	3.767693
14	Musical	3.563305
15	Mystery	3.677001
16	Romance	3.553813
17	Sci-Fi	3.395743
18	Thriller	3.507676
19	War	3.780813
20	Western	3.555918

We have previously seen the distribution of ratings for the various genres (see **Genres** section, above), but we want to visualize the mean rating per genre *for each user*. We will first gather together the list of all users and all genres for which they have given ratings.

```
gu <- rbind(edx %>% select(userId, genre1, rating) %>% mutate(genre = genre1) %>% select(-genre
1),
            edx %>% filter(! is.na(genre2)) %>%
              select(userId, genre2, rating) %>% mutate(genre = genre2) %>% select(-genre2),
            edx %>% filter(! is.na(genre3)) %>%
              select(userId, genre3, rating) %>% mutate(genre = genre3) %>% select(-genre3),
            edx %>% filter(! is.na(genre4)) %>%
              select(userId, genre4, rating) %>% mutate(genre = genre4) %>% select(-genre4),
            edx %>% filter(! is.na(genre5)) %>%
              select(userId, genre5, rating) %>% mutate(genre = genre5) %>% select(-genre5),
            edx %>% filter(! is.na(genre6)) %>%
              select(userId, genre6, rating) %>% mutate(genre = genre6) %>% select(-genre6),
            edx %>% filter(! is.na(genre7)) %>%
              select(userId, genre7, rating) %>% mutate(genre = genre7) %>% select(-genre7),
            edx %>% filter(! is.na(genre8)) %>%
              select(userId, genre8, rating) %>% mutate(genre = genre8) %>% select(-genre8)
)

gu <- gu %>% group_by(userId, genre) %>% summarize(mean_rt = mean(rating))
```

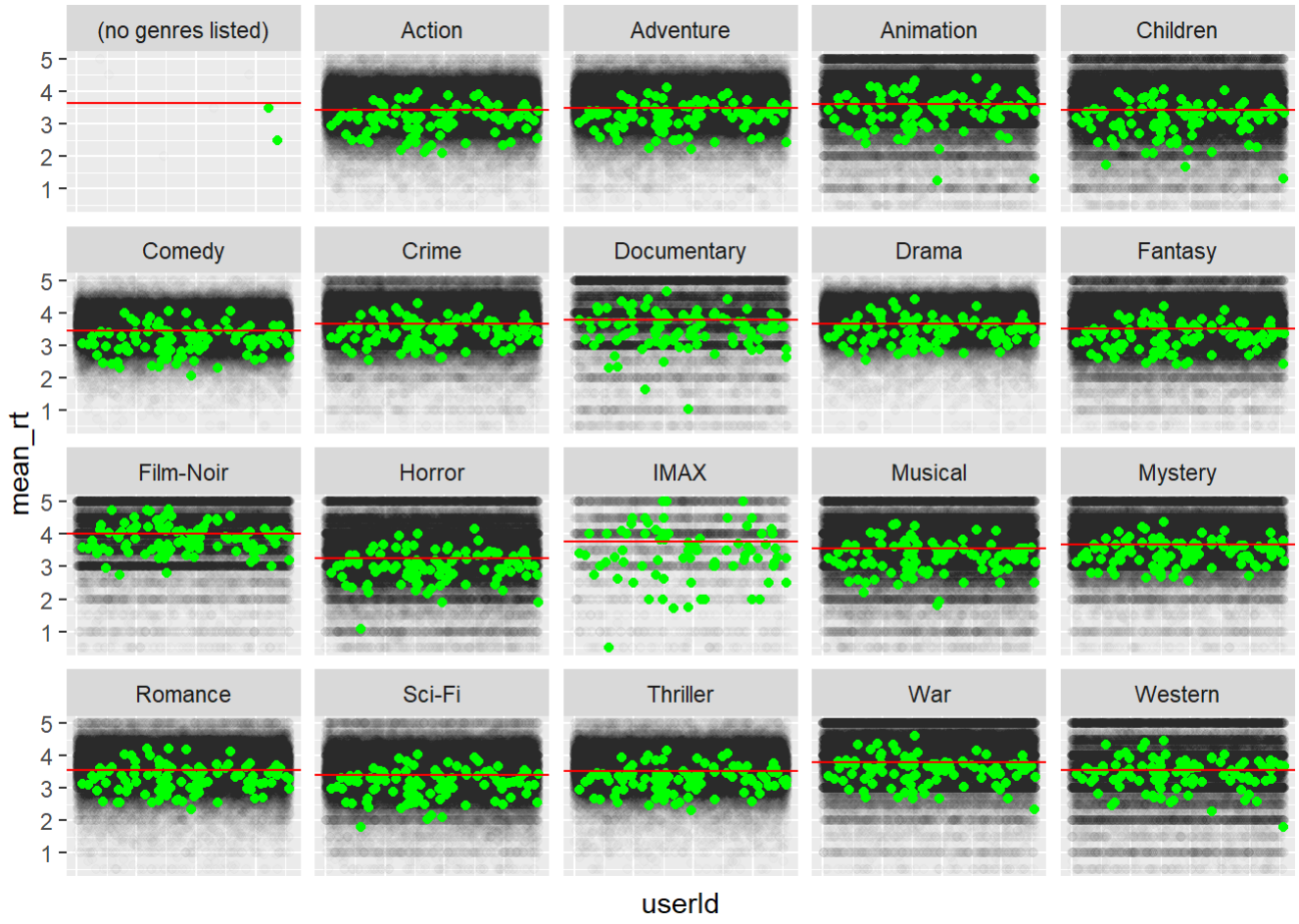
We plot the mean rating per genre against the user ID using a scatter plot (even though the user ID itself is meaningless), and overlay the overall genre mean (red line).

```
gu %>% ggplot(aes(x=userId, y=mean_rt))+geom_point(alpha=0.01)+
  geom_hline(aes(yintercept = mean_rt), data=grd, color = "red")+facet_wrap(~genre)+
  theme(axis.text.x=element_blank(), axis.ticks.x=element_blank())
```



From this we can infer from the density of the points a distribution that is relatively normal if left-skewed (as we saw with the histograms above). To see if the genre effect per user may be significant, we select the top 100 users by number of reviews, and overlay their individual genre average (green) over the entire population.

```
#Look at user+genre for users with most reviews (top 100)
top10_users <- edx %>% group_by(userId) %>% summarize(n=n()) %>% top_n(100, n)
top10_gu <- gu %>% inner_join(top10_users, by="userId")

gu %>% ggplot(aes(x=userId, y=mean_rt))+geom_point(alpha=0.01)+
  geom_point(data = top10_gu,
             aes(x=userId, y=mean_rt), color="green")+
  geom_hline(aes(yintercept = mean_rt), data=grd, color = "red")+facet_wrap(~genre)+
  theme(axis.text.x=element_blank(), axis.ticks.x=element_blank())
```



The distribution of the top 100 users shows a good bit of variation, so this is a promising candidate to use in the predictive model.

Modelling

We will do a relatively simple prediction algorithm based on the mean of the ratings in the training set, and determine the accuracy of the algorithm by using Root Mean Square Error (RMSE):

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - y)^2}$$

where y_i is the predicted value and y is the true value ("rating" in the validation set). For ease of use we define the population mean **μ** (mu) and a function for the RMSE

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

RMSE <- function(x) {sqrt(mean((x - validation$rating)^2))}

RMSE(mu)
```

```
## [1] 1.061202
```

If we predicted each movie to simply have the same rating, that being the population mean rating, our RMSE would be 1.061. We will now look at “biases” or “effects” that can be applied to the population mean to bring the prediction closer to the actual rating and hence reduce the RMSE.

NOTE: When adding in “effects”, we will confine the predicted rating to 0.0 thru 5.0, inclusively, since no ratings outside of that range were available to the users.

Movie effect

An individual movie may be more or less popular than average, so the prediction for that movie should be effected by the overall popularity (or lack thereof) of a given movie. To calculate this “movie effect”, we find, for each movie (represented by *movieId*) the mean, over all reviews, of the difference between each rating and the population mean.

```
#Find the mean effect per movie
movie <- edx %>% group_by(movieId) %>% summarize(bmov = mean(rating-mu))

#Predict by adding the movie effect to the population mean
pred <- validation %>% left_join(movie, by="movieId") %>%
  mutate(pred = ifelse(bmov+mu > 5, 5, bmov+mu)) %>% .$pred

RMSE(pred)
```

```
## [1] 0.9439087
```

We have reduced our RMSE considerably to 0.944.

User effect

Like the movie effect, individual users may tend to give higher or lower reviews than average. We will look at the prediction algorithm with just a user effect, and then add the user effect to the movie effect algorithm above.

```
#Find the mean effect per user
user <- edx %>% group_by(userId) %>% summarize(buser = mean(rating-mu))

#Predict by adding the user effect to the population mean
pred <- validation %>% left_join(user, by="userId") %>%
  mutate(pred = ifelse(mu + buser > 5, 5, mu + buser)) %>% .$pred

RMSE(pred)
```

```
## [1] 0.978336
```

```
#Predict by adding the user effect to the movie effect algorithm
pred <- validation %>% left_join(movie, by="movieId") %>%
  left_join(user, by="userId") %>% mutate(pred = ifelse(mu + bmov + buser > 5, 5,
                                                        mu + bmov + buser)) %>% .$pred

RMSE(pred)
```

```
## [1] 0.8847368
```

We see that user only (.978) was worse than movie only, but user effect added to the movie effect gave us another significant improvement to .885.

Dates

From the exploratory analysis, it did not appear that the year of the movie's release (movie_yr) nor the year of the review (review_yr) will have a considerable effect on RMSE, due to the high variability across the population. However, the nostalgia index (review_yr - movie_yr) had less variability, so we will see if has any significant effect on the prediction.

```
nidx <- edx %>% group_by(nostalgia_idx) %>% summarize(bnost = mean(rating-mu))

pred <- validation %>%
  left_join(nidx, by = "nostalgia_idx") %>%
  mutate(pred = ifelse(mu + bnost > 5, 5,
                       mu + bnost)) %>% .$pred

RMSE(pred)
```

```
## [1] 1.052393
```

The RMSE for just the nostalgia index (1.052) is not significantly better than just using the population mean as a prediction (1.061), so we would not think that adding it to any other algorithm will show any significant improvement. However, to confirm, we will add it to the movie and user effects to see what it does.

```
pred <- validation %>%
  left_join(movie, by="movieId") %>%
  left_join(user, by="userId") %>%
  left_join(nidx, by = "nostalgia_idx") %>%
  mutate(pred = ifelse(mu + bmov + buser + bnost > 5, 5,
                       mu + bmov + buser + bnost)) %>% .$pred

RMSE(pred)
```

```
## [1] 0.8961023
```

The RMSE of 0.896 is worse than just movie plus user, so we will not use nostalgia index (or any other date-related field) in the final model.

Genre effect

Since there are potentially multiple genres for a single movie, we can model the genre effect in two ways:

- Total of the genre effects
- Mean genre effect

Total genre effect

First we will need to find the effect for each genre, then join that result into the validation set.

```
find_genre_effect <- function(x) {
  tmp <- edx %>% mutate( gr = ifelse (genre1 == x | genre2 == x |
                                      genre3 == x | genre4 == x |
                                      genre5 == x | genre6 == x |
                                      genre7 == x | genre8 == x, x, NA) )
  tmp %>% filter(! is.na(gr)) %>% group_by(gr) %>% summarize(bgen = mean(rating-mu)) %>% .$bgen
}

gre <- sapply(all_genres, find_genre_effect)
gred <- data.frame(genre=names(gre), bgen = gre, stringsAsFactors = FALSE)
```

Since there are up to 8 genres per movie, we have to join the effects table 8 times. Each singular join looks like the following (the remaning 7 are done in the background):

```
validation <- validation %>%
  left_join(gred, by=c("genre1" = "genre")) %>%
  mutate(bgen1 = bgen) %>%
  select(-bgen)
```

We can now total all of the genre effects and add it to the movie and user effects to get a prediction.

```
validation <- validation %>%
  mutate(total_bgen = ifelse(is.na(bgen1),0,bgen1)+
           ifelse(is.na(bgen2),0,bgen2)+
           ifelse(is.na(bgen3),0,bgen3)+
           ifelse(is.na(bgen4),0,bgen4)+
           ifelse(is.na(bgen5),0,bgen5)+
           ifelse(is.na(bgen6),0,bgen6)+
           ifelse(is.na(bgen7),0,bgen7)+
           ifelse(is.na(bgen8),0,bgen8))

pred <- validation %>%
  left_join(movie, by="movieId") %>%
  left_join(user, by="userId") %>%
  mutate(pred = ifelse(mu + bmov + buser+total_bgen > 5, 5,
                       mu + bmov + buser+total_bgen)) %>% .$pred

RMSE(pred)
```

```
## [1] 0.910929
```

The RMSE adding in the total genre effect is worse than that of the movie and user alone, so we will not use that in the final model.

Mean genre effect

For the mean genre effect, instead of totalling the effects of the genres of a movie, we take the average of the effects of all the genres on a movie. We use an “unpivot” method to calculate the mean genre effect.

```
mean_genre <- validation %>% select(movieId,
                                    bgen1, bgen2, bgen3, bgen4,
                                    bgen5, bgen6, bgen7, bgen8) %>%
  gather(key=effect, value=genre_effect, -movieId)

mean_genre <- mean_genre %>%
  group_by(movieId) %>% summarize(mean_bgen = mean(genre_effect, na.rm=TRUE))

pred <- validation %>%
  left_join(movie, by="movieId") %>%
  left_join(user, by="userId") %>%
  left_join(mean_genre, by="movieId") %>%
  mutate(pred = ifelse(mu + bmov + buser + mean_bgen > 5, 5,
                       mu + bmov + buser + mean_bgen)) %>% .$pred

RMSE(pred)
```

```
## [1] 0.8916389
```

The results for the mean genre effect (0.892) are better than that of the total genre effect (0.911), but both are still not as good as movie and user alone. Genre by itself will not be used in the final model.

Genre per User effect

Using similar transformations while looking at just the genre effect, we will now look at the effect when faceting the genres of the movie by the user

```
gu <- rbind(edx %>% select(userId, genre1, rating) %>% mutate(genre = genre1) %>% select(-genre
1),
            edx %>% filter(! is.na(genre2)) %>%
              select(userId, genre2, rating) %>% mutate(genre = genre2) %>% select(-genre2),
            edx %>% filter(! is.na(genre3)) %>%
              select(userId, genre3, rating) %>% mutate(genre = genre3) %>% select(-genre3),
            edx %>% filter(! is.na(genre4)) %>%
              select(userId, genre4, rating) %>% mutate(genre = genre4) %>% select(-genre4),
            edx %>% filter(! is.na(genre5)) %>%
              select(userId, genre5, rating) %>% mutate(genre = genre5) %>% select(-genre5),
            edx %>% filter(! is.na(genre6)) %>%
              select(userId, genre6, rating) %>% mutate(genre = genre6) %>% select(-genre6),
            edx %>% filter(! is.na(genre7)) %>%
              select(userId, genre7, rating) %>% mutate(genre = genre7) %>% select(-genre7),
            edx %>% filter(! is.na(genre8)) %>%
              select(userId, genre8, rating) %>% mutate(genre = genre8) %>% select(-genre8)
)

gue <- gu %>% group_by(userId, genre) %>% summarize(bgu = mean(rating-mu))
```

We join to each of the possible 8 genres for each review

```
validation <- validation %>%
  left_join(gue, by=c("userId", "genre1" = "genre")) %>%
  mutate(bgu1 = bgu) %>%
  select(-bgu)

validation <- validation %>%
  left_join(gue, by=c("userId", "genre2" = "genre")) %>%
  mutate(bgu2 = bgu) %>%
  select(-bgu)

validation <- validation %>%
  left_join(gue, by=c("userId", "genre3" = "genre")) %>%
  mutate(bgu3 = bgu) %>%
  select(-bgu)

validation <- validation %>%
  left_join(gue, by=c("userId", "genre4" = "genre")) %>%
  mutate(bgu4 = bgu) %>%
  select(-bgu)

validation <- validation %>%
  left_join(gue, by=c("userId", "genre5" = "genre")) %>%
  mutate(bgu5 = bgu) %>%
  select(-bgu)

validation <- validation %>%
  left_join(gue, by=c("userId", "genre6" = "genre")) %>%
  mutate(bgu6 = bgu) %>%
  select(-bgu)

validation <- validation %>%
  left_join(gue, by=c("userId", "genre7" = "genre")) %>%
  mutate(bgu7 = bgu) %>%
  select(-bgu)

validation <- validation %>%
  left_join(gue, by=c("userId", "genre8" = "genre")) %>%
  mutate(bgu8 = bgu) %>%
  select(-bgu)
```

We can then look at the user-genre effect by taking the mean user-genre effect for the review

```
mean_gu <- validation %>% select(movieId, userId,
                                 bgu1, bgu2, bgu3, bgu4,
                                 bgu5, bgu6, bgu7, bgu8) %>%
  gather(key=effect, value=gu_effect, -c(movieId, userId))

mean_gu <- mean_gu %>%
  group_by(movieId, userId) %>%
  summarize(mean_bgu = ifelse(is.na(mean(gu_effect, na.rm = TRUE)), 0, mean(gu_effect, na.rm=TRU
E)))

pred <- validation %>%
  left_join(movie, by="movieId") %>%
  left_join(mean_gu, by=c("movieId", "userId")) %>%
  mutate(pred = ifelse(mu + bmov + mean_bgu > 5, 5,
                       mu + bmov + mean_bgu)) %>% .$pred

RMSE(pred)
```

```
## [1] 0.8730843
```

We see that this set of features significantly improved the RMSE (0.873) from the best we have seen so far (Movie + User effect, 0.885)

Conclusion

In order to create an accurate recommender system for movies, it is best to use the

- Population mean
- Movie effect (how much does a rating for a specific movie, on average, deviate from the population mean?)
- Mean user/genre effect (how much does a user's rating of a particular genre, averaged over all genres assigned to a movie, deviate from the population mean?)

Or mathematically

$$y_i = \mu + b_{movie} + b_{user/genre}$$

We obtain an RMSE of **0.873** by this method. This is the best RMSE obtained by attempting several different models.

```
results %>% arrange(RMSE) %>% grid.table()
```

	model	RMSE
1	Movie + Mean User/Genre	0.8730843
2	Movie Effect + User Effect	0.8847368
3	Movie + User + Mean Genre	0.8916389
4	Movie + User + Nostalgia	0.8961023
5	Movie + User + Total Genre	0.9109290
6	Movie Effect	0.9439087
7	User Effect only	0.9783360
8	Nostalgia Index Bias	1.0523934
9	Population Mean	1.0612018