Movie Lens Project

Lindsay Lee

2024-06-07

edX Harvard Data Science Capstone: MovieLens Project

##**Introduction/Overview**

The MovieLens dataset (10 million version; Harper & Konstan, 2015) was originally collected by GroupLens Research to help understand people's preferences for movies across time (GroupLens, 2024). According to GroupLens (2024), this was a random sample of users that rated at least 20 movies (i.e., Action, Adventure, Animation, Children, Comedy, Crime, Documentary, Drama, Fantasy, Film-Noir, Horror, IMAX, Musical, Mystery, Romance, Sci-Fi, Thriller, War, Western). The current report sought to train a machine learning algorithm using inputs from a subset of the Movielens dataset (i.e., 10 Million version; Harper & Konstan, 2015) to predict movie ratings in the validation set. Specifically, I aimed to build a recommendation system based on how specific movie ID, user input, and genres predict movie ratings. The following research questions guided my inquiry: 1. To what extent does genres and/or years relate to ratings for all movies included? 2. To what extent does genres predict movie ratings? 3. To what extent does genres, user Id, and movie Id predict movie ratings?

##**Methods/Analysis** The MovieLens 10 Million dataset was imported from GroupLens (2024) with provided by the edX Harvard R for Data Science Course (Irizarry, 2024). All analyses were conducted in RStudio R version 4.3.1 (R Core Team, 2023).

Prior to the splitting of the datasets, I explored the movie lens dataset by genres and ratings alone, then again by year. This required me to clean the movielens dataset prior to splitting the dataset to build a recommendation system. The genres variable specifically had over 700+ levels of and several levels were seperated by a space. Cleaning the genres variable reduced genres to 20 levels.

This helped to answer the first research question on exploring the relationship of genres, year, and ratings using descriptive statistics and data visualization.

For the second and third research questions, we used simple models with user ID, movie ID, and genre respectively. The outcome of interest was movie ratings by users (i.e., rating). The features of interest (predictors) were the specific users (userId), 20 different genres (genres), and the specific movies (movieId) that is also linked to the title of the movie (see documentation at Harper & Konstan, 2015). To build a recommendation system, there needs to be users who have provided ratings to specific items of interest (e.g., Netflix, Amazon; Irizarry, 2024). To assess ratings provided, I built a recommendation system with simple machine learning algorithms to assess the features (i.e., userId, movieId, genres) with the outcome (rating). Specifically, I was interested in how genres predicted movie ratings. Also, I was interested in how genres, user ID, and movie ID predicted movie ratings.

Initial data were split from the movielens dataset into a training set(edx) and a validation set (final_holdout_test). Then, we assessed a model to look at the effect of combination of movie ID and user ID, and then another model to assess movie, user ID, and genres. Finally, to assess the robustness of our final model we used regularization to assess a final model of movie ID, user ID, and genres.

Due to the nature of our outcome variable being "continuous", using a loss function was more appropriate for evaluation. Root Mean Square Error (RMSE) was used to evaluate model fit. RMSE measures the mean

difference between the predicted values (training set) and the actual values (test set). It is calculated by computing by taking the average of the difference of the mean of a variable in the training set subtracted from the mean of same variable in the test set. Then, the square root is taken for the average difference squared (see Chapter 27.4.8: The loss function; Irizarry, 2024).

```
knitr::opts_chunk$set(echo = TRUE)
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: tidyverse
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.2     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.5.0     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: caret
## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(tidyverse)
library(caret)
library(tidyr)
library(ggplot2)
library(dplyr)


# MovieLens 10M dataset:
# https://grouplens.org/datasets/movielens/10m/
# http://files.grouplens.org/datasets/movielens/ml-10m.zip

options(timeout = 120)

dl <- "ml-10M100K.zip"
if(!file.exists(dl))
  download.file("https://files.grouplens.org/datasets/movielens/ml-10m.zip", dl)

ratings_file <- "ml-10M100K/ratings.dat"
```

```r
if(!file.exists(ratings_file))
  unzip(dl, ratings_file)

movies_file <- "ml-10M100K/movies.dat"
if(!file.exists(movies_file))
  unzip(dl, movies_file)

ratings <- as.data.frame(str_split(read_lines(ratings_file), fixed("::"), simplify = TRUE),
                         stringsAsFactors = FALSE)

colnames(ratings) <- c("userId", "movieId", "rating", "timestamp")
ratings <- ratings |>
  mutate(userId = as.integer(userId),
         movieId = as.integer(movieId),
         rating = as.numeric(rating),
         timestamp = as.integer(timestamp))

movies <- as.data.frame(str_split(read_lines(movies_file), fixed("::"), simplify = TRUE),
                        stringsAsFactors = FALSE)
colnames(movies) <- c("movieId", "title", "genres")
movies <- movies |>mutate(movieId = as.integer(movieId))

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

#convert timestamp to date then to year format
movielens <- mutate(movielens, date = as_datetime(timestamp))
movielens$year <- as.numeric(format(movielens$date, "%Y"))

#remove title, timestamp, and date
movielens <- movielens |>
  select(-timestamp, -date)

# Convert factors
movielens$userId <- as.factor(movielens$userId)
movielens$movieId <- as.factor(movielens$movieId)

movielens <- movielens |>
  separate_rows(genres, sep = "\\|")

movielens = as.data.frame(movielens)
movielens$genres = as.factor(movielens$genres)
```

##**Results**

##*Research Question 1* We can make a approximation that the accuracy of the ratings for movies is 3.53.

```r
#Overall estimated Accuracy of ratings
overallmean = mean(movielens$rating, na.rm = TRUE)
print(overallmean)
```

```
## [1] 3.526945
```

To assess the first research question, I initially analyzed descriptive statistics and visualizations of genres,

year, and rating data in the movielens dataset. From the total MovieLens dataset, Drama (n = 4344198), Comedy(n = 3934068), & Action (n = 2845349) were the top 3 in the sheer quantity of included genres.

```
#Summarize mean and sd of rating by genres
genres_n = movielens |>group_by(genres) |>
  summarise(n = n(), mean = mean(rating, na.rm = TRUE),
            sd = sd(rating, na.rm = TRUE)) |> arrange(desc(n))

#kable can create clean tables in R Markdown
knitr::kable(genres_n, summary=TRUE, rownames=TRUE)
```

genres	n	mean	sd
Drama	4344198	3.673263	0.9954426
Comedy	3934068	3.436946	1.0748707
Action	2845349	3.421331	1.0663432
Thriller	2584435	3.507186	1.0310767
Adventure	2121074	3.493621	1.0529115
Romance	1901883	3.553776	1.0304104
Sci-Fi	1490489	3.396193	1.0925896
Crime	1474957	3.665655	1.0118693
Fantasy	1028482	3.502019	1.0654116
Children	820149	3.418474	1.0926587
Horror	768225	3.269243	1.1503833
Mystery	630944	3.677631	0.9998899
War	568063	3.780173	1.0123122
Animation	519112	3.599988	1.0198116
Musical	481174	3.562478	1.0570704
Western	210459	3.555657	1.0237554
Film-Noir	131592	4.012151	0.8864959
Documentary	103454	3.783459	1.0040711
IMAX	9080	3.764537	1.0312114
(no genres listed)	7	3.642857	1.1073349

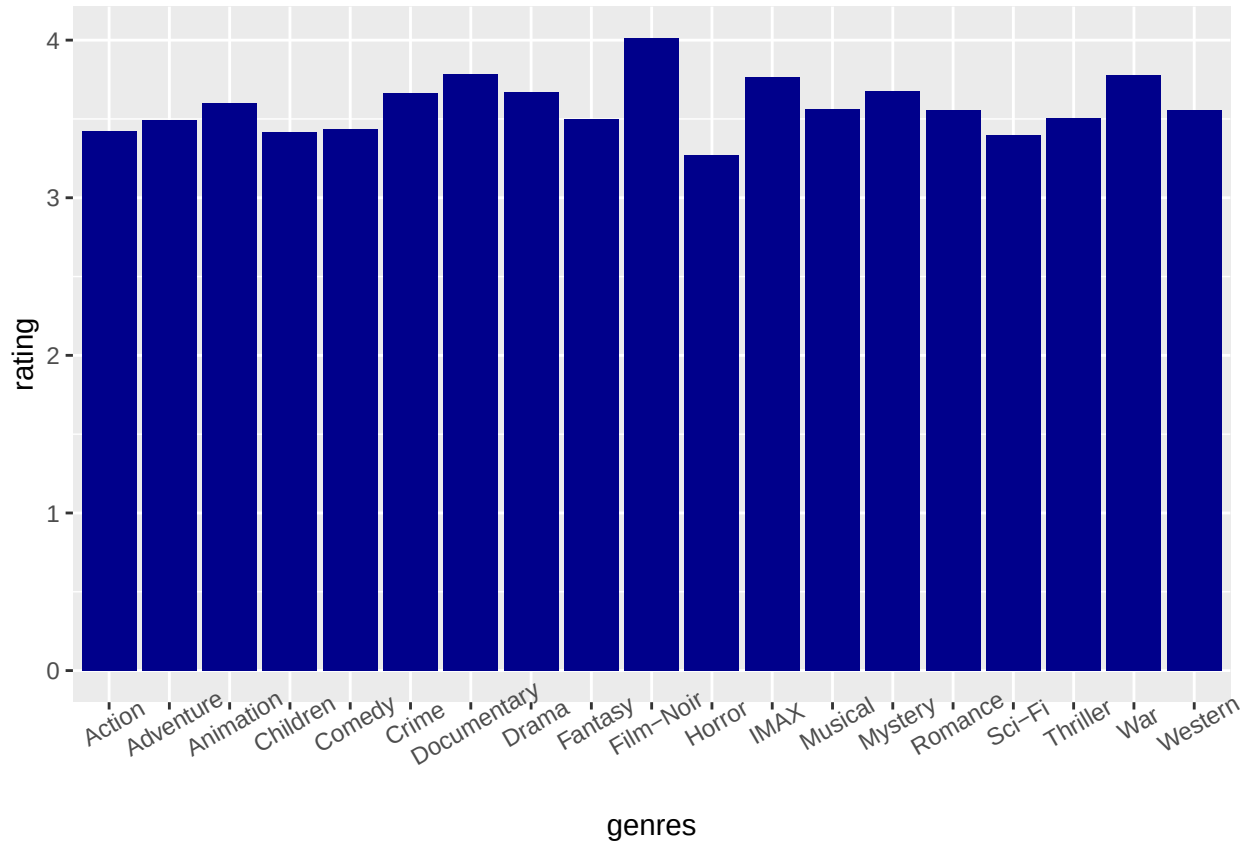
```
#drop no genres listed as a category, only 7
movielens = subset(movielens, genres != "(no genres listed)")
movielens  = droplevels(movielens)
```

After filtering by the mean rating from the MovieLens dataset, we find that Film-Noir (m = 4.01, sd = .89), Documentary (m = 3.78, sd = 1), and War (m = 3.78, sd = 1.01) have the highest mean ratings. When we visually inspect genres and ratings, we do not see large differences in ratings, yet it does verify the same findings from the summary table. This led me to question how genres were impacting the ratings, how time (years) impacted ratings, and more specifically, how users (userId) were rating specific movies (movieId) based on genres (see Research Question 2 & 3 for building my recommendation system).

```
#Summarize mean and sd of rating by genres
genres_m = movielens |>group_by(genres) |>
  summarise(n = n(), mean = mean(rating, na.rm = TRUE),
            sd = sd(rating, na.rm = TRUE)) |> arrange(desc(mean))

#kable can create clean tables in R Markdown
knitr::kable(genres_m, summary=TRUE, rownames=TRUE)
```

genres	n	mean	sd
Film-Noir	131592	4.012151	0.8864959
Documentary	103454	3.783459	1.0040711
War	568063	3.780173	1.0123122
IMAX	9080	3.764537	1.0312114
Mystery	630944	3.677631	0.9998899
Drama	4344198	3.673263	0.9954426
Crime	1474957	3.665655	1.0118693
Animation	519112	3.599988	1.0198116
Musical	481174	3.562478	1.0570704
Western	210459	3.555657	1.0237554
Romance	1901883	3.553776	1.0304104
Thriller	2584435	3.507186	1.0310767
Fantasy	1028482	3.502019	1.0654116
Adventure	2121074	3.493621	1.0529115
Comedy	3934068	3.436946	1.0748707
Action	2845349	3.421331	1.0663432
Children	820149	3.418474	1.0926587
Sci-Fi	1490489	3.396193	1.0925896
Horror	768225	3.269243	1.1503833

```r
#Explore bar plots of Genres x Rating
movielens |>
  ggplot(aes(x = genres, y = rating, fill = genres)) +
  geom_bar(stat = "summary", fun = "mean", fill = "darkblue") +
  theme(axis.text.x = element_text(angle = 30))
```



When filtering by year and rating, we see that Drama, Comedy, Action, and Thriller seem to be frequent. It does not appear that there is a clear pattern of how year is influencing ratings. To visualize the relationship between genres, year, and mean rating, see Figure below. Across time, genres did not consistently predict mean ratings.

```
#Summarize mean, sd, and count by year and genres
#make year a factor type to run in ggplot plot
movielens$year = as.factor(movielens$year)
year_meanrating = movielens |>
  group_by(genres, year) |>
  summarise(n = n(), meanrating = mean(rating, na.rm = TRUE,
                                       sd = sd(rating, na.rm = TRUE)))
```

```
## `summarise()` has grouped output by 'genres'. You can override using the
## `.groups` argument.
```

```
knitr::kable(year_meanrating, summary=TRUE, rownames=TRUE)
```

genres	year	n	meanrating
Action	1995	1	3.000000
Action	1996	331387	3.440509
Action	1997	129587	3.539915
Action	1998	55068	3.440092
Action	1999	194594	3.447994

genres	year	n	meanrating
Action	2000	330156	3.472286
Action	2001	201279	3.464649
Action	2002	148081	3.402948
Action	2003	180619	3.382518
Action	2004	221961	3.340720
Action	2005	351725	3.342110
Action	2006	233169	3.382176
Action	2007	216363	3.393406
Action	2008	247138	3.490604
Action	2009	4221	3.434257
Adventure	1996	233698	3.516534
Adventure	1997	102435	3.614077
Adventure	1998	39081	3.526087
Adventure	1999	146001	3.514277
Adventure	2000	245717	3.554805
Adventure	2001	147270	3.552509
Adventure	2002	109662	3.493389
Adventure	2003	134395	3.488199
Adventure	2004	167080	3.429743
Adventure	2005	266164	3.419356
Adventure	2006	176014	3.447453
Adventure	2007	163719	3.442371
Adventure	2008	186444	3.504712
Adventure	2009	3394	3.481291
Animation	1996	56977	3.688559
Animation	1997	20088	3.651185
Animation	1998	8432	3.582187
Animation	1999	34327	3.641798
Animation	2000	56081	3.674667
Animation	2001	34498	3.631718
Animation	2002	27082	3.591721
Animation	2003	34379	3.593909
Animation	2004	39900	3.537794
Animation	2005	67452	3.549739
Animation	2006	45767	3.543241
Animation	2007	41875	3.527188
Animation	2008	51121	3.587039
Animation	2009	1133	3.572374
Children	1996	102193	3.531240
Children	1997	38675	3.524680
Children	1998	17973	3.346075
Children	1999	59744	3.475981
Children	2000	98287	3.464934
Children	2001	56504	3.457861
Children	2002	43889	3.389938
Children	2003	53581	3.403081
Children	2004	61652	3.337353
Children	2005	100363	3.331028
Children	2006	64477	3.359772
Children	2007	56755	3.336710
Children	2008	64541	3.414992
Children	2009	1515	3.383498

genres	year	n	meanrating
Comedy	1995	2	3.000000
Comedy	1996	400989	3.434176
Comedy	1997	173379	3.502425
Comedy	1998	73717	3.442015
Comedy	1999	314308	3.573278
Comedy	2000	501623	3.531168
Comedy	2001	304439	3.478877
Comedy	2002	235807	3.407142
Comedy	2003	282849	3.415313
Comedy	2004	313937	3.356180
Comedy	2005	474980	3.354715
Comedy	2006	299677	3.378723
Comedy	2007	264018	3.365947
Comedy	2008	288419	3.440165
Comedy	2009	5924	3.324105
Crime	1995	2	4.000000
Crime	1996	164957	3.586129
Crime	1997	60062	3.713529
Crime	1998	27149	3.611514
Crime	1999	107759	3.767593
Crime	2000	162781	3.735817
Crime	2001	102679	3.727500
Crime	2002	77487	3.674307
Crime	2003	99997	3.631329
Crime	2004	112906	3.579482
Crime	2005	181604	3.610768
Crime	2006	125212	3.639751
Crime	2007	115035	3.652345
Crime	2008	135031	3.741904
Crime	2009	2296	3.619338
Documentary	1996	4736	3.867188
Documentary	1997	2750	3.745455
Documentary	1998	1220	3.606557
Documentary	1999	8244	3.969190
Documentary	2000	10652	3.924803
Documentary	2001	4866	3.754213
Documentary	2002	4783	3.727995
Documentary	2003	7235	3.787906
Documentary	2004	10129	3.833004
Documentary	2005	16104	3.760525
Documentary	2006	11484	3.712992
Documentary	2007	10244	3.679666
Documentary	2008	10812	3.693026
Documentary	2009	195	3.682051
Drama	1995	1	3.000000
Drama	1996	442224	3.710923
Drama	1997	195337	3.721282
Drama	1998	89122	3.671989
Drama	1999	362268	3.773469
Drama	2000	561500	3.741425
Drama	2001	326233	3.699837
Drama	2002	248062	3.643408

genres	year	n	meanrating
Drama	2003	297750	3.627938
Drama	2004	326632	3.591291
Drama	2005	500338	3.603201
Drama	2006	331637	3.624627
Drama	2007	311067	3.629540
Drama	2008	345638	3.683832
Drama	2009	6389	3.617546
Fantasy	1996	87029	3.531490
Fantasy	1997	39347	3.542964
Fantasy	1998	16890	3.448905
Fantasy	1999	71722	3.533253
Fantasy	2000	116212	3.508390
Fantasy	2001	67876	3.544301
Fantasy	2002	55537	3.513441
Fantasy	2003	69123	3.528485
Fantasy	2004	84506	3.478611
Fantasy	2005	134021	3.460879
Fantasy	2006	93225	3.474020
Fantasy	2007	90959	3.466210
Fantasy	2008	100133	3.519549
Fantasy	2009	1902	3.507886
Film-Noir	1996	3446	3.677307
Film-Noir	1997	4620	3.901299
Film-Noir	1998	3182	3.896606
Film-Noir	1999	15634	4.169950
Film-Noir	2000	21394	4.138403
Film-Noir	2001	10263	4.089935
Film-Noir	2002	8084	4.024369
Film-Noir	2003	8519	4.017021
Film-Noir	2004	8390	3.950060
Film-Noir	2005	15342	3.969398
Film-Noir	2006	11637	3.939804
Film-Noir	2007	10176	3.906348
Film-Noir	2008	10750	3.923395
Film-Noir	2009	155	3.861290
Horror	1995	1	5.000000
Horror	1996	51680	3.599110
Horror	1997	24637	3.464586
Horror	1998	17319	3.206305
Horror	1999	59855	3.290836
Horror	2000	108956	3.262152
Horror	2001	67610	3.239979
Horror	2002	50392	3.169094
Horror	2003	56107	3.186403
Horror	2004	61163	3.175343
Horror	2005	92800	3.232128
Horror	2006	60069	3.219914
Horror	2007	55817	3.227287
Horror	2008	60595	3.348849
Horror	2009	1224	3.310458
IMAX	1996	8	3.625000
IMAX	1997	11	2.363636

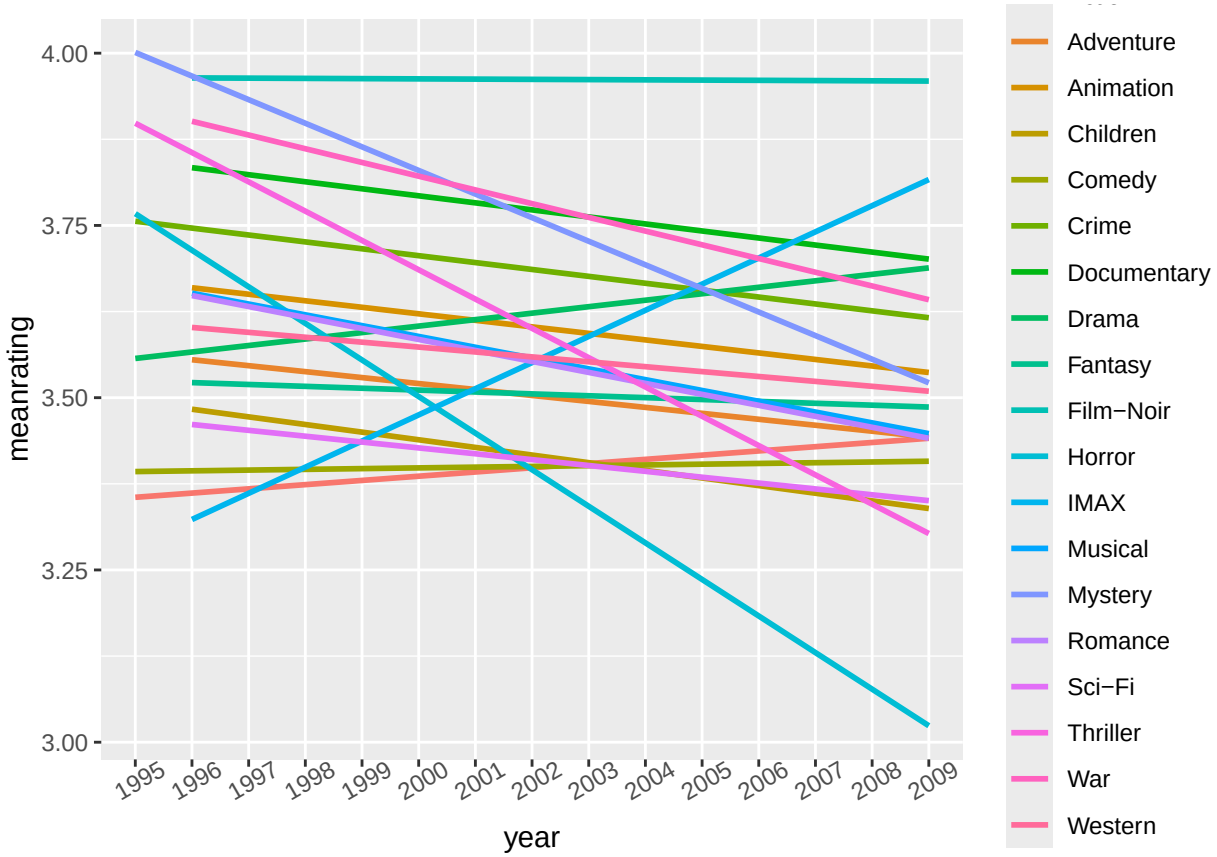
genres	year	n	meanrating
IMAX	1998	46	3.782609
IMAX	1999	320	3.759375
IMAX	2000	818	3.797066
IMAX	2001	585	3.538461
IMAX	2002	485	3.635051
IMAX	2003	442	3.588235
IMAX	2004	431	3.548724
IMAX	2005	624	3.280449
IMAX	2006	421	3.334917
IMAX	2007	1052	3.592205
IMAX	2008	3732	4.028537
IMAX	2009	105	4.104762
Musical	1996	55378	3.631460
Musical	1997	24160	3.588742
Musical	1998	10519	3.574864
Musical	1999	42458	3.668920
Musical	2000	63796	3.660590
Musical	2001	34367	3.642739
Musical	2002	26612	3.564783
Musical	2003	32911	3.539850
Musical	2004	36433	3.469259
Musical	2005	56878	3.465866
Musical	2006	34845	3.471531
Musical	2007	29431	3.447284
Musical	2008	32706	3.503715
Musical	2009	680	3.466176
Mystery	1995	1	5.000000
Mystery	1996	45186	3.625105
Mystery	1997	20185	3.650681
Mystery	1998	12480	3.624920
Mystery	1999	50818	3.777205
Mystery	2000	79940	3.731611
Mystery	2001	45443	3.722708
Mystery	2002	39172	3.727484
Mystery	2003	44499	3.670184
Mystery	2004	49539	3.608793
Mystery	2005	81429	3.625821
Mystery	2006	54504	3.638082
Mystery	2007	50577	3.645926
Mystery	2008	56181	3.712073
Mystery	2009	990	3.659091
Romance	1996	231765	3.617932
Romance	1997	98495	3.624935
Romance	1998	45034	3.617844
Romance	1999	166514	3.685606
Romance	2000	240840	3.633873
Romance	2001	136936	3.574998
Romance	2002	105305	3.525350
Romance	2003	124022	3.512627
Romance	2004	142148	3.452141
Romance	2005	215191	3.439168
Romance	2006	138897	3.477552

genres	year	n	meanrating
Romance	2007	121177	3.476600
Romance	2008	132800	3.542771
Romance	2009	2759	3.443820
Sci-Fi	1996	134417	3.405455
Sci-Fi	1997	62832	3.618172
Sci-Fi	1998	30270	3.431615
Sci-Fi	1999	116304	3.445221
Sci-Fi	2000	195667	3.448027
Sci-Fi	2001	110565	3.407733
Sci-Fi	2002	87876	3.372183
Sci-Fi	2003	103338	3.350849
Sci-Fi	2004	117365	3.297487
Sci-Fi	2005	177311	3.309876
Sci-Fi	2006	114562	3.346271
Sci-Fi	2007	109439	3.373386
Sci-Fi	2008	128254	3.462286
Sci-Fi	2009	2289	3.413499
Thriller	1995	1	5.000000
Thriller	1996	313634	3.578301
Thriller	1997	131650	3.528264
Thriller	1998	54794	3.464595
Thriller	1999	178832	3.591181
Thriller	2000	301736	3.549706
Thriller	2001	184932	3.528951
Thriller	2002	138337	3.491409
Thriller	2003	163347	3.467147
Thriller	2004	188267	3.415702
Thriller	2005	297764	3.425540
Thriller	2006	205104	3.455825
Thriller	2007	199203	3.482970
Thriller	2008	223020	3.557073
Thriller	2009	3814	3.472470
War	1996	59973	3.943008
War	1997	26847	3.848735
War	1998	12643	3.842047
War	1999	50483	3.914961
War	2000	75246	3.889562
War	2001	42657	3.825585
War	2002	30486	3.744768
War	2003	34716	3.701982
War	2004	41389	3.639445
War	2005	66095	3.642409
War	2006	42219	3.657394
War	2007	40108	3.692081
War	2008	44436	3.737926
War	2009	765	3.723529
Western	1996	35233	3.547668
Western	1997	7142	3.663820
Western	1998	3993	3.577511
Western	1999	15638	3.546937
Western	2000	28540	3.637877
Western	2001	16250	3.641600

genres	year	n	meanrating
Western	2002	11185	3.534555
Western	2003	13399	3.521158
Western	2004	15661	3.491859
Western	2005	23753	3.460447
Western	2006	14940	3.525736
Western	2007	11845	3.529675
Western	2008	12660	3.598341
Western	2009	220	3.502273

```
#Explore genres and year by ratings
ggplot(movielens, aes(x = year, y = meanrating, group = genres, color = genres)) +
  geom_smooth(data = year_meanrating, se = FALSE, method = "lm") +
  theme(axis.text.x = element_text(angle = 30))
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



#Research Question 2 & 3 To build a recommendation system, we used the code provided in the Introduction to Data Science textbook (Irizarry, 2024) and the edX Harvard R for Data Science Course, however provided some modifications to explore the data visually and to add additional models that included genres (which was not used in the course).

```
library(caret)
# Final hold-out test set will be 10% of MovieLens data
set.seed(1, sample.kind="Rounding") # if using R 3.6 or later
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding'
## sampler used
```

```
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 final hold-out test set are also in edx set
#use this at the end to test the model
final_holdout_test <- temp |>
  semi_join(edx, by = "movieId") |>
  semi_join(edx, by = "userId")
```

Then, prior to analysis, we removed the temp and final_holdout_test set from the edx dataset and removed the other objects related to the entire dataset.

```
#Add rows removed from final hold-out test set back into edx set
removed <- anti_join(temp, final_holdout_test)
```

```
## Joining with `by = join_by(userId, movieId, rating, title, genres, year)`
```

```
edx <- rbind(edx, removed)
rm(dl, ratings, movies, test_index, temp, movielens, removed)
```

The estimated accuracy of the training and test set resemble the movielens dataset around 3.52.

```
#What is the estimated Accuracy of each set?
mean(edx$rating, na.rm = TRUE)
```

```
## [1] 3.526893
```

```
mean(final_holdout_test$rating, na.rm = TRUE)
```

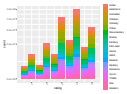
```
## [1] 3.527416
```

I ran summary statistics and visually inspected both of the training and test set to verify that their distributions of genres and ratings look similar.

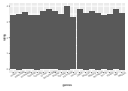
```
library(ggplot2)
#Summarize count, mean, sd
genres_edx = edx |>group_by(genres) |>
  summarise(n = n(), mean = mean(rating, na.rm = TRUE),
            sd = sd(rating, na.rm = TRUE)) |>
          arrange(desc(mean))

#kable can create clean tables in R Markdown
knitr::kable(genres_edx, summary=TRUE, rownames=TRUE)
```

genres	n	mean	sd
Film-Noir	118352	4.012328	0.8861690
Documentary	93043	3.783562	1.0047333
War	511103	3.780417	1.0123765
IMAX	8172	3.767193	1.0351097
Mystery	567740	3.678030	0.9995640
Drama	3910305	3.673055	0.9955599
Crime	1327378	3.665633	1.0117831
Animation	467431	3.599723	1.0204306
Musical	432829	3.562098	1.0573319
Western	189536	3.554599	1.0244182
Romance	1711450	3.554206	1.0302638
Thriller	2325999	3.507204	1.0312135
Fantasy	925727	3.501997	1.0655289
Adventure	1909216	3.493382	1.0530817
Comedy	3540355	3.436880	1.0750450
Action	2560658	3.421494	1.0663144
Children	738186	3.418262	1.0925953
Sci-Fi	1341475	3.395803	1.0924939
Horror	691526	3.269169	1.1504853

```
edx |>group_by(genres, rating) |>
  ggplot(aes(x = rating, fill = genres)) +
  scale_y_continuous(trans='sqrt') +
  geom_bar() +
  theme(axis.text.x = element_text(angle = 30))
```



```
ggplot(edx, aes(x = genres, y = rating)) +
  geom_bar(stat = "summary", fun = "mean") + theme(axis.text.x = element_text(angle = 30))
```

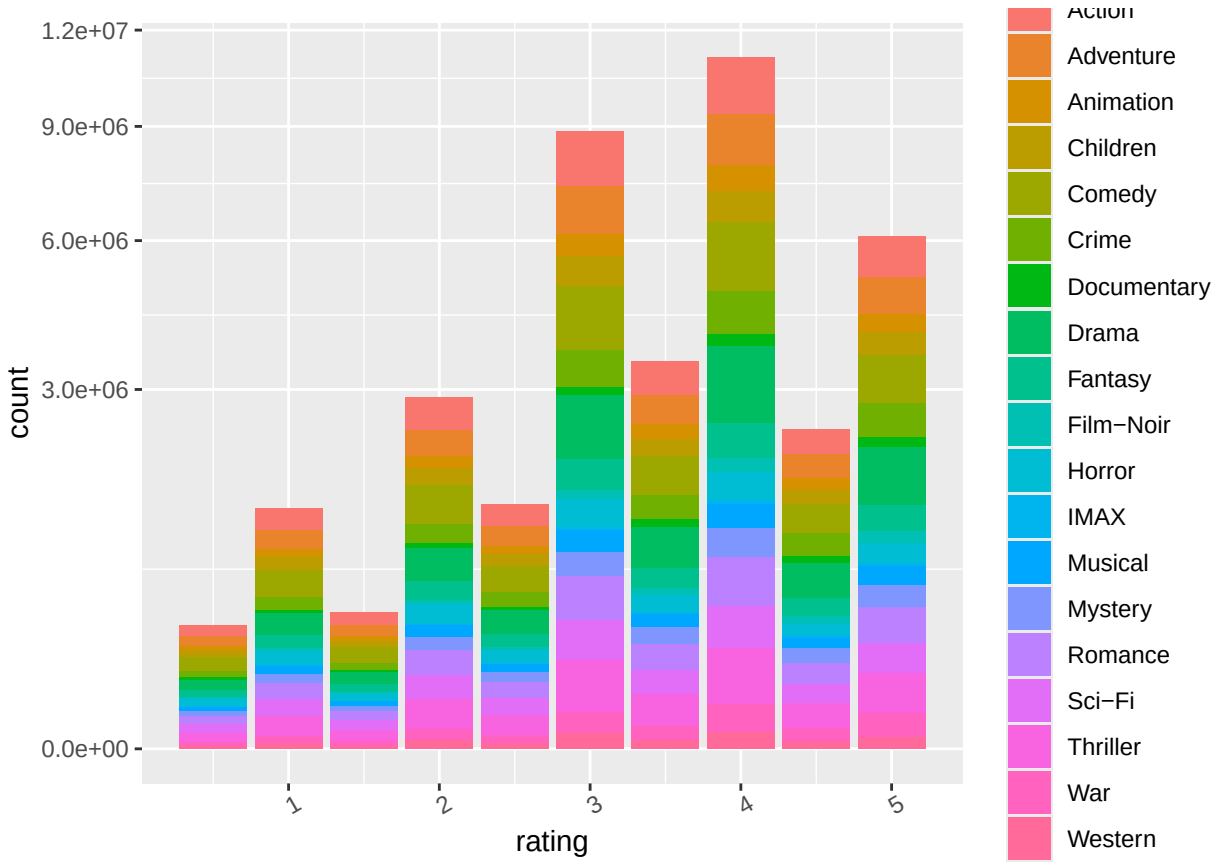


```
#Visually, the training and final hold out sets look similar
#plots are based on count of ratings, did a square root transformation on ratings on both the edx and f
genres_final_holdout_test = final_holdout_test |>group_by(genres) |>
  summarise(n = n(), mean = mean(rating, na.rm = TRUE),
            sd = sd(rating, na.rm = TRUE)) |> arrange(desc(mean))
#kable can create clean tables in R Markdown
knitr::kable(genres_final_holdout_test, summary=TRUE, rownames=TRUE)
```

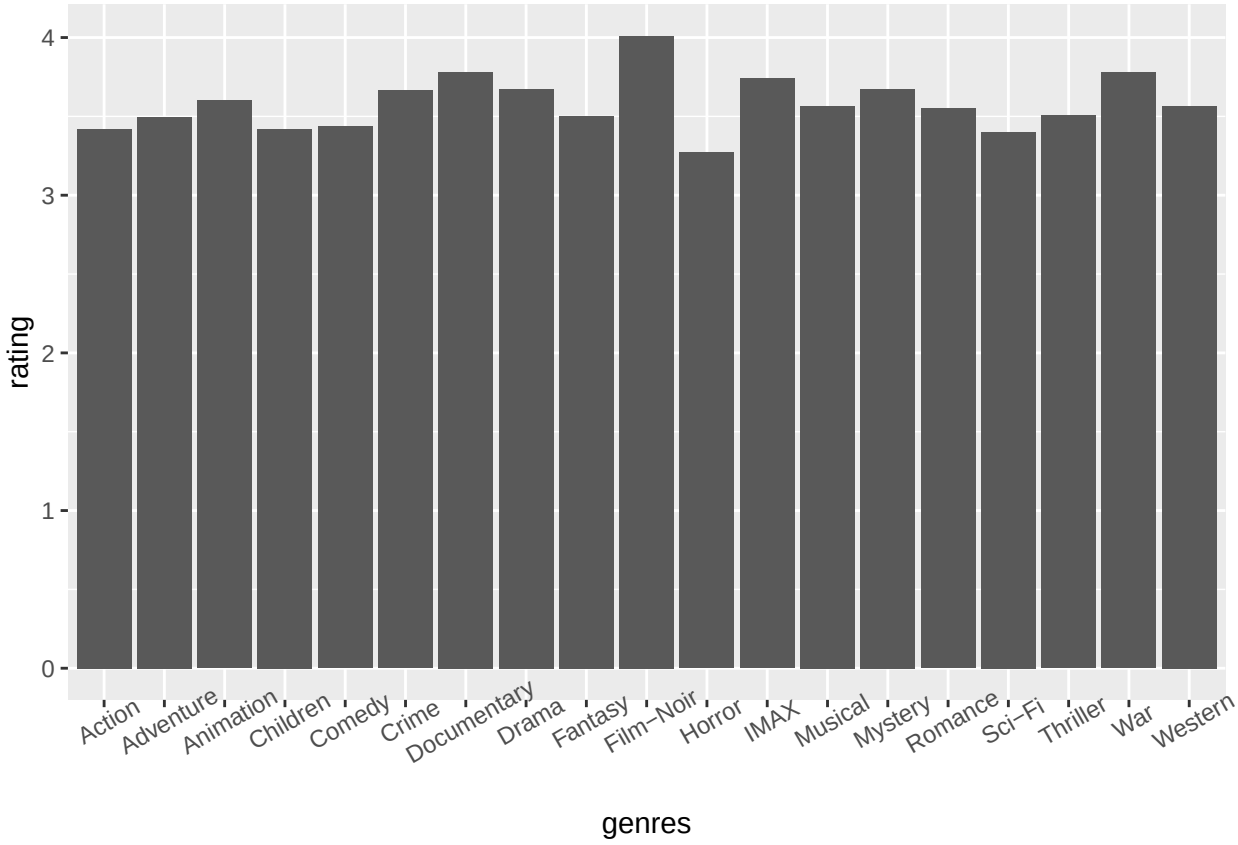
genres	n	mean	sd
Film-Noir	13240	4.010574	0.8894443
Documentary	10411	3.782538	0.9981811
War	56960	3.777985	1.0117413

genres	n	mean	sd
IMAX	908	3.740639	0.9956745
Drama	433893	3.675134	0.9943842
Mystery	63204	3.674040	1.0028129
Crime	147579	3.665853	1.0126480
Animation	51681	3.602388	1.0142032
Musical	48345	3.565881	1.0547313
Western	20923	3.565239	1.0177060
Romance	190433	3.549915	1.0317217
Thriller	258436	3.507023	1.0298470
Fantasy	102755	3.502219	1.0643594
Adventure	211858	3.495773	1.0513772
Comedy	393713	3.437542	1.0733039
Children	81963	3.420385	1.0932338
Action	284691	3.419862	1.0666024
Sci-Fi	149014	3.399707	1.0934478
Horror	76699	3.269919	1.1494707

```
final_holdout_test |>group_by(genres, rating) |>
  ggplot(aes(x = rating, fill = genres)) +
  scale_y_continuous(trans='sqrt') +
  geom_bar() +  theme(axis.text.x = element_text(angle = 30))
```



```
ggplot(final_holdout_test, aes(x = genres, y = rating)) +
  geom_bar(stat = "summary", fun = "mean") + theme(axis.text.x = element_text(angle = 30))
```

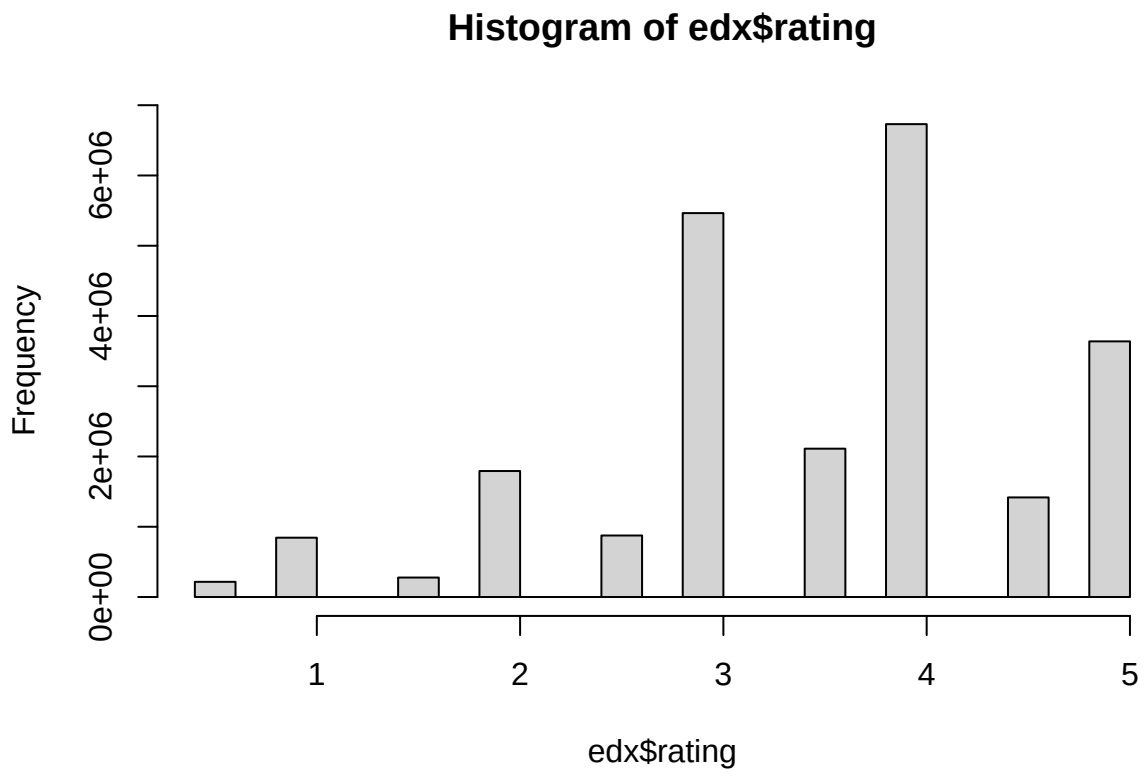


To assess the recommendation system, we will use RMSE to evaluate performance of the each algorithm. We will build the recommendation system hierarchically, first with the Average, then each variable individually Movie, then User, then Genre. Then, we will run the Movie & User Effects Model and then finally the interaction of the Movie, User, & Genre Effects Model. The following code was adapted from the edX Harvard R for Data Science course.

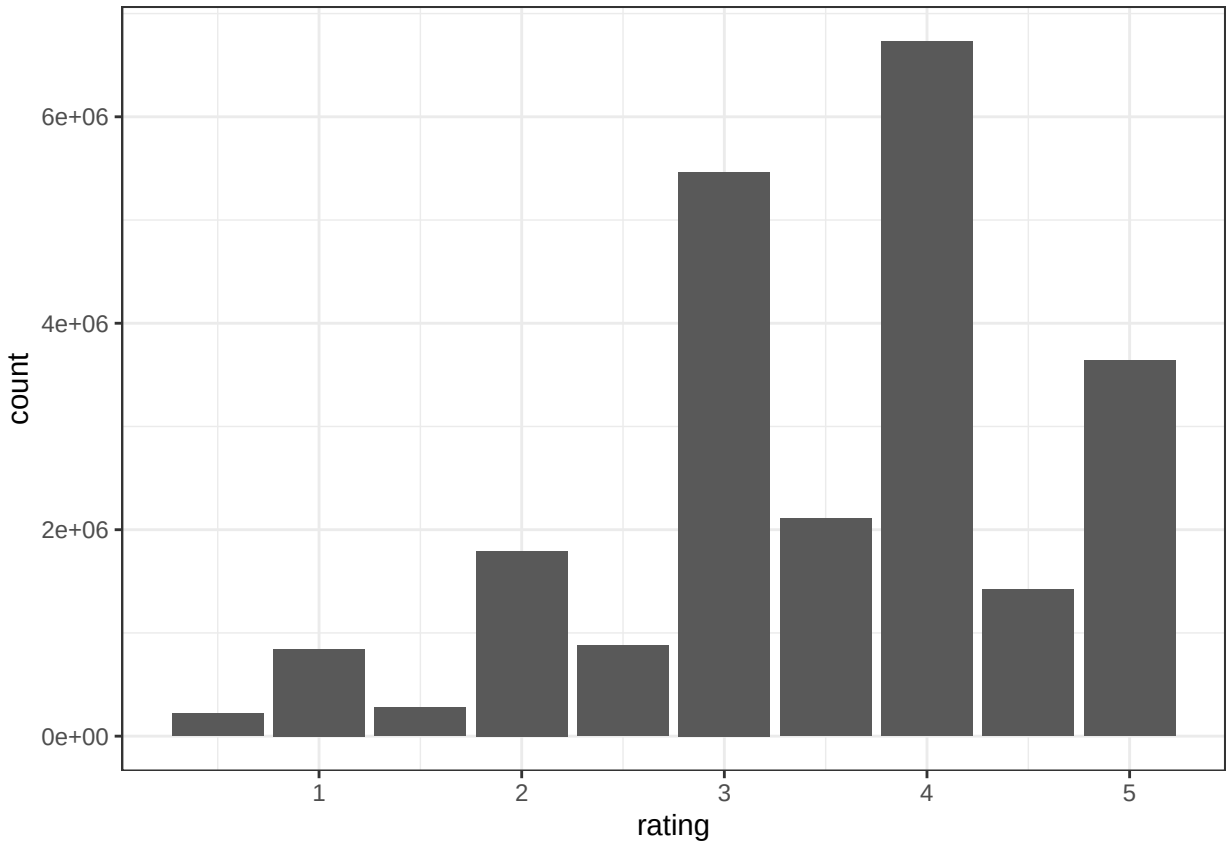
```
#Building the Recommendation System-----------
#Define RMSE function -----------
#Use the RMSE function defined in the course
#https://learning.edx.org/course/course-v1:HarvardX+PH125.8x+1T2023/block-v1:HarvardX+PH125.8x+1T2023+t
#https://learning.edx.org/course/course-v1:HarvardX+PH125.8x+1T2023/block-v1:HarvardX+PH125.8x+1T2023+t
#https://rafalab.dfci.harvard.edu/dsbook/large-datasets.html#modeling-movie-effects

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

mu_hat = mean(edx$rating, na.rm = TRUE)
mu_hat
```

```
## [1] 3.526893
```

```r
hist(edx$rating)
```




```r
#same plot but in ggplot
edx |>
  ggplot(aes(x = rating))  +
  geom_bar() + theme_bw()
```




```
#use this for testing later
naive_rmse <- RMSE(final_holdout_test$rating, mu_hat)
naive_rmse
```

```
## [1] 1.051619
```

```
predictions <- rep(2.5, nrow(final_holdout_test))
RMSE(final_holdout_test$rating, predictions)
```

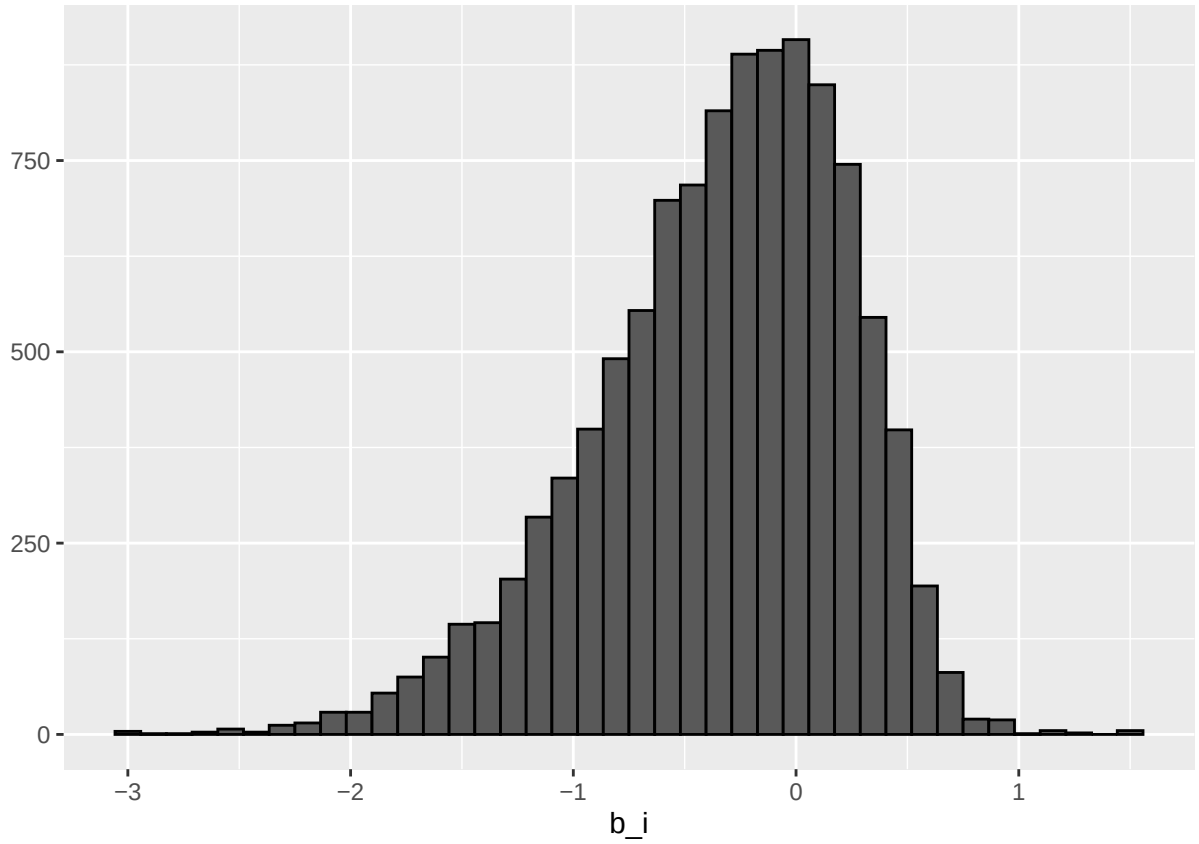
```
## [1] 1.470199
```

```
rmse_results <- tibble(method = "Average", RMSE = naive_rmse)
```

The movie effect model yielded improved performance compared to the average effect model with RMSE at .94.

```
#Model ONLY Movie effects -----------
mu <- mean(edx$rating)
movie_avgs <- edx |>
  group_by(movieId) |>
  summarize(b_i = mean(rating - mu))
movie_avgs %>%
  qplot(b_i, geom ="histogram", bins = 40, data = ., color = I("black"))
```

```
## Warning: `qplot()` was deprecated in ggplot2 3.4.0.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```




```
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 <- bind_rows(rmse_results,
                          data_frame(method="Movie Effect Model",
                                     RMSE = model_1_rmse ))
```

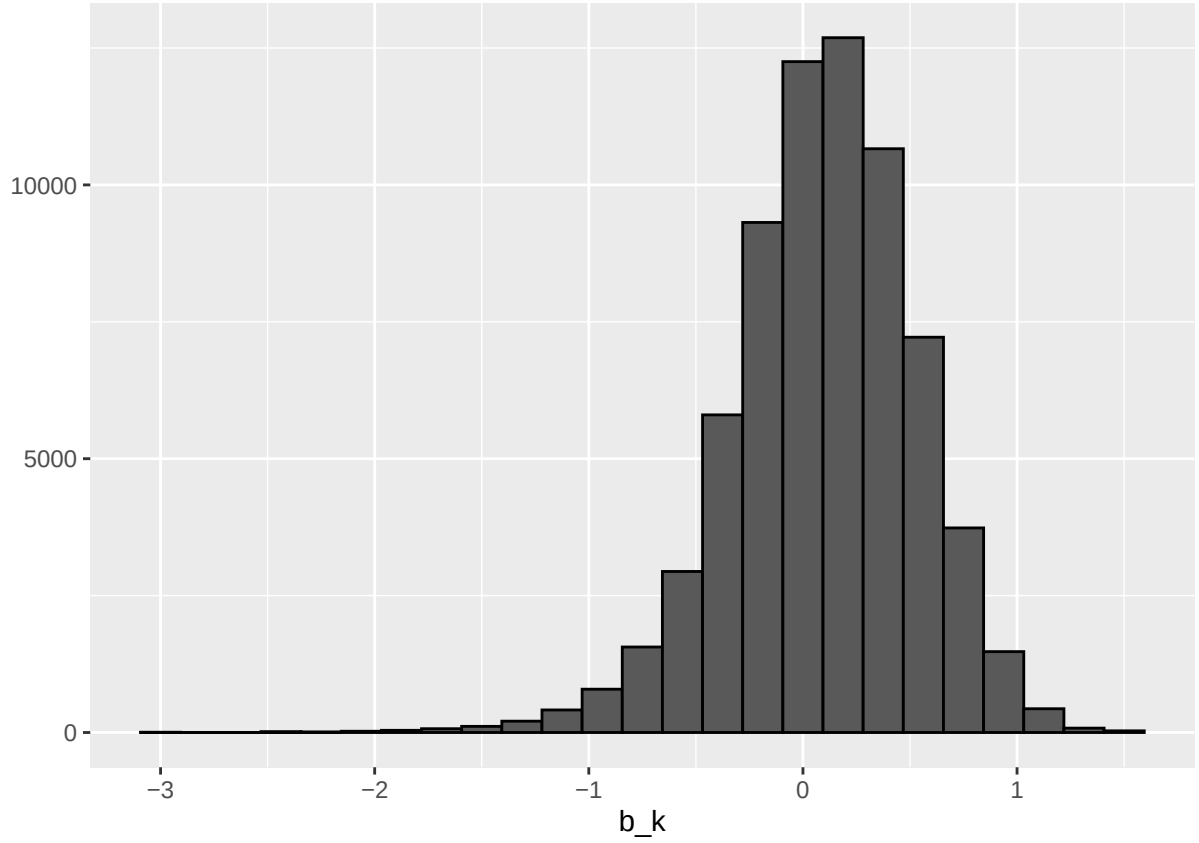
```
## Warning: `data_frame()` was deprecated in tibble 1.1.0.
## i Please use `tibble()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

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

method	RMSE
Average	1.051619
Movie Effect Model	0.940324

The user effect model yielded an EMSE of .966, which showed worse performance than the movie effect model.

```
#Modeling ONLY Users Effects--------
#This modeled all users who rated all movies
user_avgs <- edx |>
  group_by(userId) |>
  summarize(b_k = mean(rating - mu))
user_avgs %>%
  qplot(b_k, geom ="histogram", bins = 25, data = ., color = I("black"))
```




```
predicted_ratings <- mu + final_holdout_test |>
  left_join(user_avgs, by='userId') |>
  pull(b_k)

model_2_rmse <- RMSE(predicted_ratings, final_holdout_test$rating)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="User Effect Model",
                                     RMSE = model_2_rmse ))
rmse_results |>knitr::kable()
```

method	RMSE
Average	1.0516186
Movie Effect Model	0.9403240
User Effect Model	0.9662205

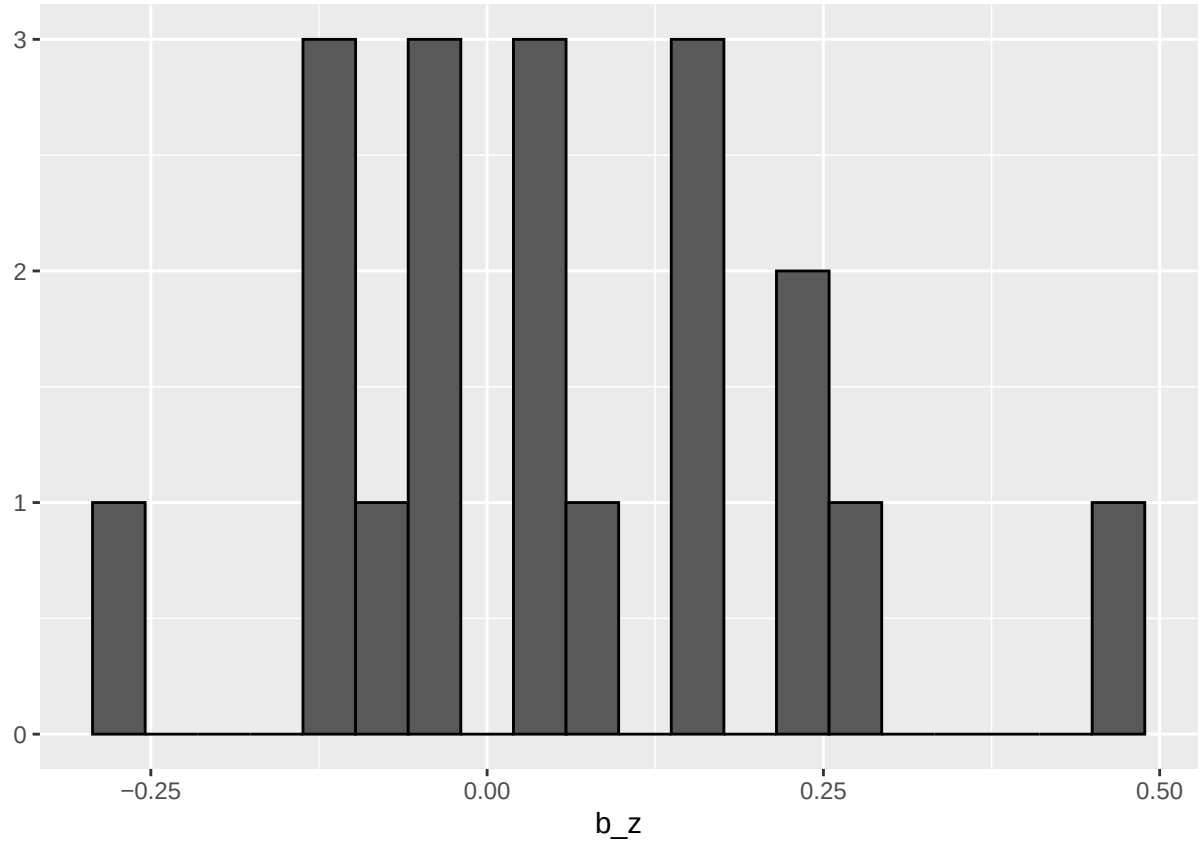
The genre effect model yielded poor performance on its own. With an RMSE at around 1.04.

```
#Modeling ONLY Genre Effects-------
#group by all genres and count in the training set
edx |>group_by(genres, rating) |>
  summarize(genres_count = n())
```

```
## `summarise()` has grouped output by 'genres'. You can override using the
## `.groups` argument.
```

```
## # A tibble: 190 x 3
## # Groups:   genres [19]
##    genres rating genres_count
##    <fct>   <dbl>        <int>
##  1 Action    0.5        27381
##  2 Action    1         107598
##  3 Action    1.5        37605
##  4 Action    2         225411
##  5 Action    2.5       110188
##  6 Action    3         645401
##  7 Action    3.5       236221
##  8 Action    4         689463
##  9 Action    4.5       140981
## 10 Action    5         340409
## # i 180 more rows
```

```
genre_avgs <- edx |>
  group_by(genres) |>
  summarize(b_z = mean(rating - mu))

genre_avgs %>%
  qplot(b_z, geom ="histogram", bins = 20, data = ., color = I("black"))
```



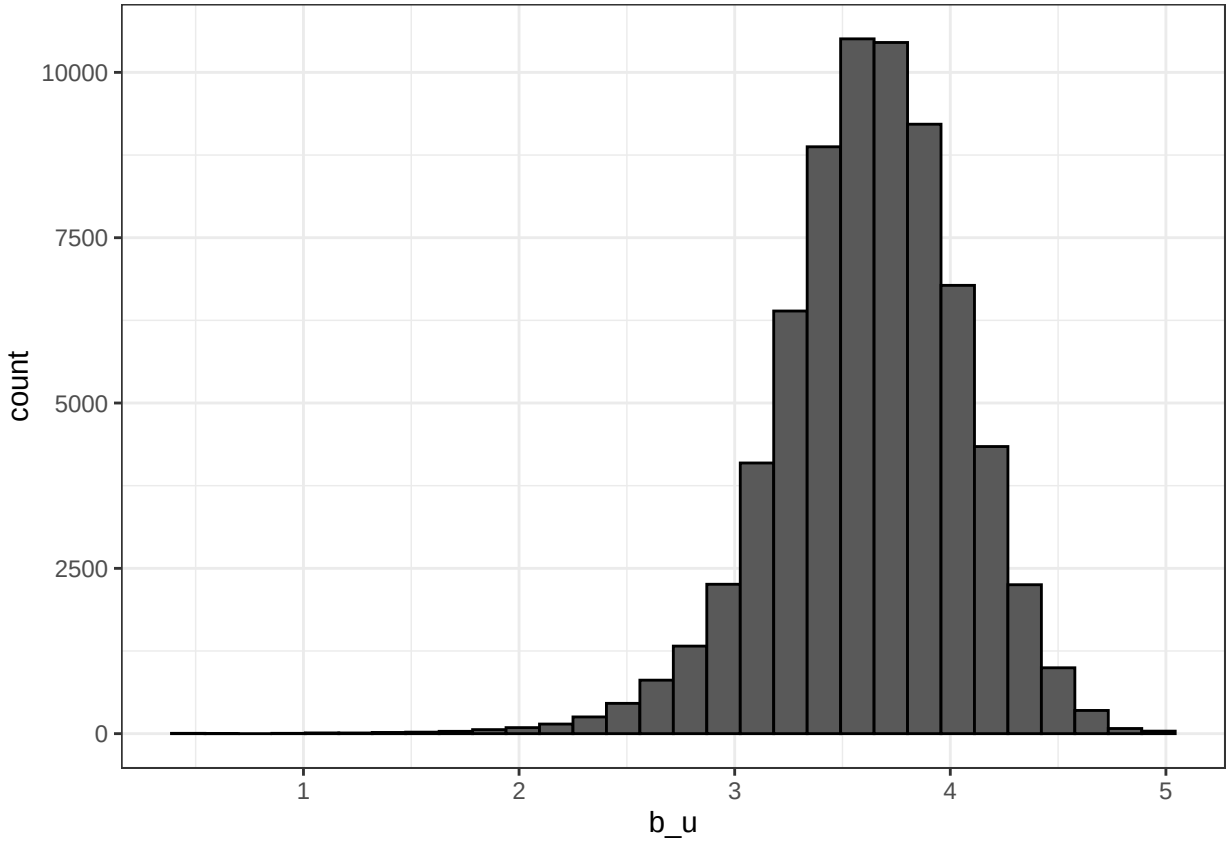

```
predicted_ratings_genre <- mu + final_holdout_test |>
  left_join(genre_avgs, by='genres') |>
  pull(b_z)

model_3_rmse <- RMSE(predicted_ratings_genre, final_holdout_test$rating)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Genre Effect Model",
                                     RMSE = model_3_rmse ))
rmse_results |>knitr::kable()
```

method	RMSE
Average	1.0516186
Movie Effect Model	0.9403240
User Effect Model	0.9662205
Genre Effect Model	1.0448537

This model assessed movie and user effects and yielded an improved RMSE at .857

```
#Hierarchically Add Movie & User Effect -----------
# Modeling Movie & User Effects ----------------
edx |>
  group_by(userId) |>
  summarize(b_u = mean(rating)) |>
```

```
  ggplot(aes(b_u)) +
  geom_histogram(bins = 30, color = "black")+ theme_bw()
```



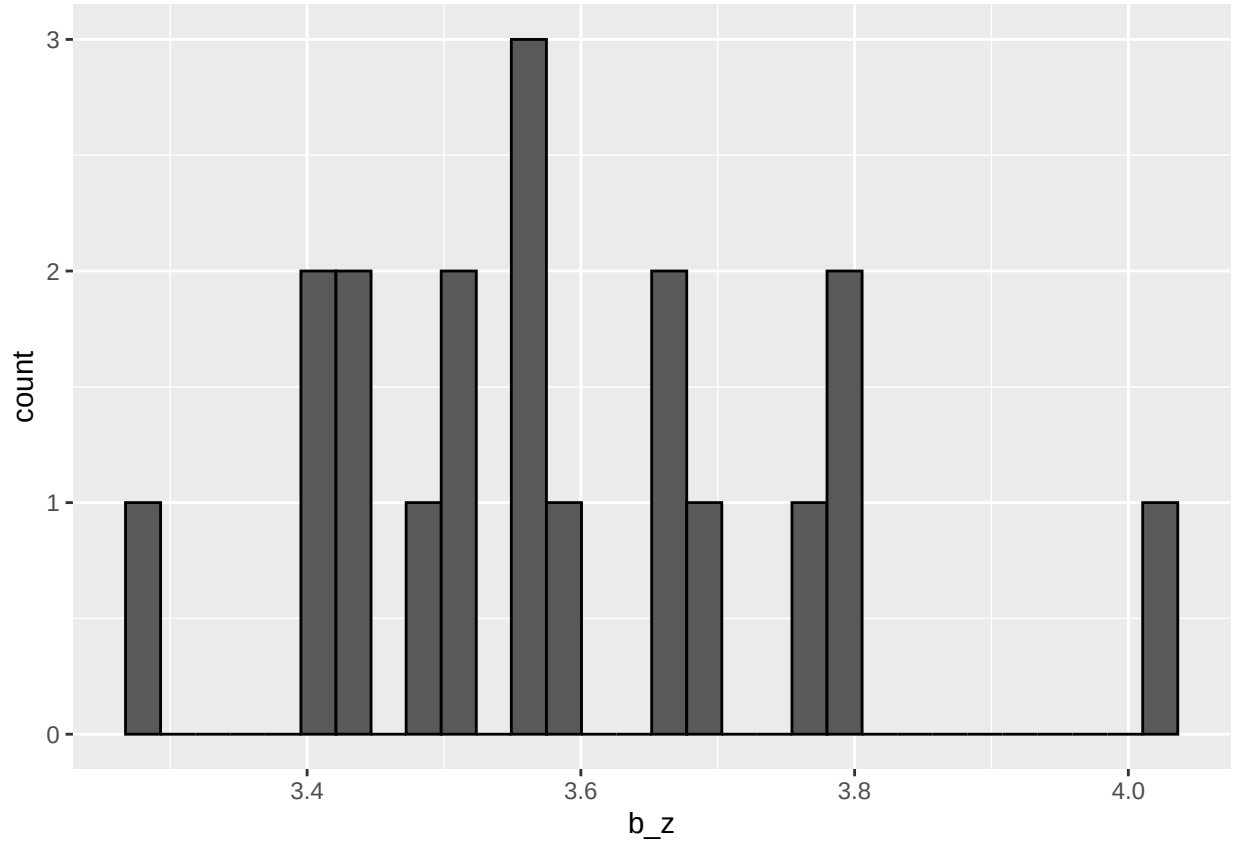

```
user_avgs <- edx |>
  left_join(movie_avgs, by='movieId') |>
  group_by(userId) |>
  summarize(b_u = mean(rating - mu - b_i))

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_4_rmse <- RMSE(predicted_ratings, final_holdout_test$rating)

rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Movie + User Effects Model",
                                     RMSE = model_4_rmse ))
rmse_results |>knitr::kable()
```

method	RMSE
Average	1.0516186

method	RMSE
Movie Effect Model	0.9403240
User Effect Model	0.9662205
Genre Effect Model	1.0448537
Movie + User Effects Model	0.8570215

The final model explored genre, movie, and user effects. Similar to the course (Irizarry, 2024), we filtered by users who rated over 100. This yielded the best performance with RMSE = .856

```
#Model Genre x Movie x User Effects --------------------
edx |>
  group_by(genres) |>
  summarize(b_z = mean(rating)) |>
  ggplot(aes(b_z)) +
  geom_histogram(bins = 30, color = "black")
```




```
genre_avgs <- edx |>
  left_join(movie_avgs, by='movieId') |>
  left_join(user_avgs, by ='userId') |>
  group_by(genres) |>
  summarize(b_final = 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(genre_avgs, by = 'genres') |>
  mutate(pred = mu + b_i + b_u + b_final) |>
  pull(pred)

model_5_rmse <- RMSE(predicted_ratings, final_holdout_test$rating)

rmse_results <- bind_rows(rmse_results,
                          tibble(method="Movie + User + Genre Effects Model",
                                 RMSE = model_5_rmse ))
rmse_results |>knitr::kable()
```

method	RMSE
Average	1.0516186
Movie Effect Model	0.9403240
User Effect Model	0.9662205
Genre Effect Model	1.0448537
Movie + User Effects Model	0.8570215
Movie + User + Genre Effects Model	0.8569437

#Conclusion Exploring the MovieLens dataset provided insight that genres was a predictor to continue to explore in relation to ratings, and that year was justifiably excluded in the dataset between genres and ratings. After building a recommendation system, we found that both the User & Movie Effect Model and the Genre, User, & Movie Effect yielded models with the lowest RMSE < 0.8649. The final model that yielded slightly better performance was the Genre, User, & Movie Effect Model (RMSE = .856) than the User & Movie Effect model with (RMSE = .857).

#Limitations & Future Directions The following report was limited to only user ID, movie ID, genres, titles, and year, however other model effects should take into account specific user characteristics that could impact ratings. There is also the risk of overfitting the model, which would need to be compared to the larger MovieLens dataset for verification.

We also used simple machine learning algorithms to develop a recommendation system. Future research should explore different machine learning algorithms to improve prediction and performance (e.g., random forest algorithms or neural networks). Also, I'd like to explore specific genres as predictors in a different machine learning algorithm (e.g., kNN). To further explore trends across time, I'd like to also use multilevel modeling to look at different clusters of data based on user characteristics.

##References Grouplens. (2024). MovieLens. https://grouplens.org/datasets/movielens/

Harper, M. F., & Konstan, J. A. (December, 2015). The MovieLens datasets: History and context. ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4, Article 19. http://dx.doi.org/10.1145/2827872

Irizarry, R. A. (2024). Introduction to data science: Data analysis and prediction algorithms with R. https://rafalab.dfci.harvard.edu/dsbook/large-datasets.html#recommendation-systems

Kuhn, M. et al. (2023). caret: Classification and regression training. R package version 6.0-94. https://cran.r-project.org/web/packages/caret/index.html

Wickham, H., Vaughan, D., & Girlich, M. (2024). tidyr: Tidy messy data. R package version 1.3.1, https://github.com/tidyverse/tidyr, https://tidyr.tidyverse.org

Wickham et al. (2014). dplyr. https://cran.r-project.org/web/packages/dplyr/vignettes/dplyr.html