# title: "Movielens"

# author: "Sijuade"

# date: "Friday, September 04, 2015"

# output: html\_document

# Data Location: [linked phrase](http://grouplens.org/datasets/movielens/)

The data dictioary for this dataset is located at [linked phrase](http://files.grouplens.org/datasets/movielens/ml-100k-README.txt). The data originally downloaded was mostly in text format and had to be transformed to csv.

After much wrangling, I have 3 files imported into R.

The dataset has 100,000 ratings total and 943 unique users as well as 1682 movies, The average user rated about 106 movies and the average movie was rated about 59 times. There's a wide variance in the number ratings each movie has, some movies were rated just once and the movie with the most amount of ratings was rated 583 times.

occupation = read.csv("occupation.csv")  
ratings = read.csv("ratings.csv")  
films = read.csv("movieinfo.csv")  
  
occupation$X = NULL  
films$X = NULL  
ratings$X = NULL

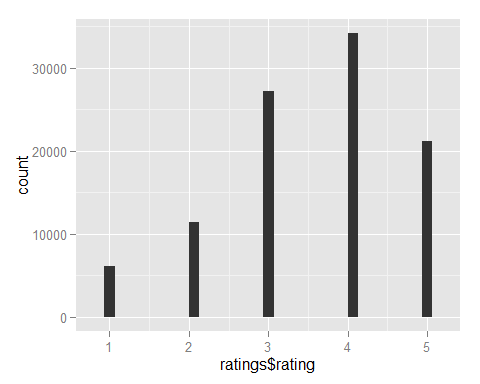
Below is a head of all three datasets

## userid age gender occupation zipcode  
## 1 1 24 M technician 85711  
## 2 2 53 F other 94043  
## 3 3 23 M writer 32067  
## 4 4 24 M technician 43537  
## 5 5 33 F other 15213  
## 6 6 42 M executive 98101

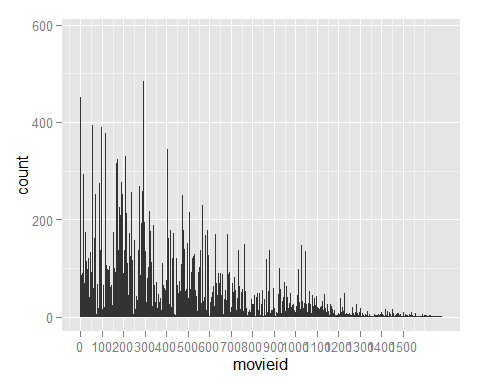
## movieid movietitle releasedate  
## 1 1 Toy Story (1995) 01-Jan-1995  
## 2 2 GoldenEye (1995) 01-Jan-1995  
## 3 3 Four Rooms (1995) 01-Jan-1995  
## 4 4 Get Shorty (1995) 01-Jan-1995  
## 5 5 Copycat (1995) 01-Jan-1995  
## 6 6 Shanghai Triad (Yao a yao yao dao waipo qiao) (1995) 01-Jan-1995  
## videoreleasedate  
## 1 NA  
## 2 NA  
## 3 NA  
## 4 NA  
## 5 NA  
## 6 NA  
## imdburl unknown  
## 1 http://us.imdb.com/M/title-exact?Toy%20Story%20(1995) 0  
## 2 http://us.imdb.com/M/title-exact?GoldenEye%20(1995) 0  
## 3 http://us.imdb.com/M/title-exact?Four%20Rooms%20(1995) 0  
## 4 http://us.imdb.com/M/title-exact?Get%20Shorty%20(1995) 0  
## 5 http://us.imdb.com/M/title-exact?Copycat%20(1995) 0  
## 6 http://us.imdb.com/Title?Yao+a+yao+yao+dao+waipo+qiao+(1995) 0  
## Action Adventure Animation Children.s Comedy Crime Documentary Drama  
## 1 0 0 1 1 1 0 0 0  
## 2 1 1 0 0 0 0 0 0  
## 3 0 0 0 0 0 0 0 0  
## 4 1 0 0 0 1 0 0 1  
## 5 0 0 0 0 0 1 0 1  
## 6 0 0 0 0 0 0 0 1  
## Fantasy Film.Noir Horror Musical Mystery Romance Sci.Fi Thriller War  
## 1 0 0 0 0 0 0 0 0 0  
## 2 0 0 0 0 0 0 0 1 0  
## 3 0 0 0 0 0 0 0 1 0  
## 4 0 0 0 0 0 0 0 0 0  
## 5 0 0 0 0 0 0 0 1 0  
## 6 0 0 0 0 0 0 0 0 0  
## Western  
## 1 0  
## 2 0  
## 3 0  
## 4 0  
## 5 0  
## 6 0

## userid movieid rating timestamp  
## 1 196 242 3 881250949  
## 2 186 302 3 891717742  
## 3 22 377 1 878887116  
## 4 244 51 2 880606923  
## 5 166 346 1 886397596  
## 6 298 474 4 884182806

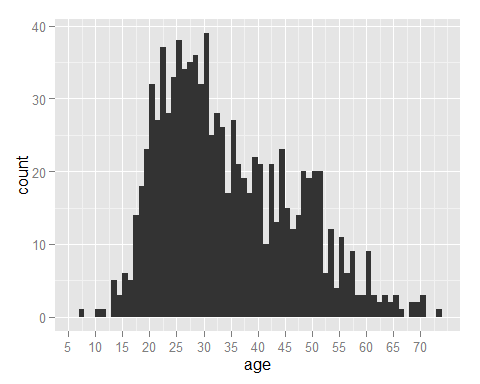
Since predicting ratings is the point of this exercise, it makes sense to combine all three into one dataset, or rather to merge the films and occupations dataset into the ratings dataset. But first we can explore the individual datasets in detail. First is a plot ratings just to get a feel of it's distribution



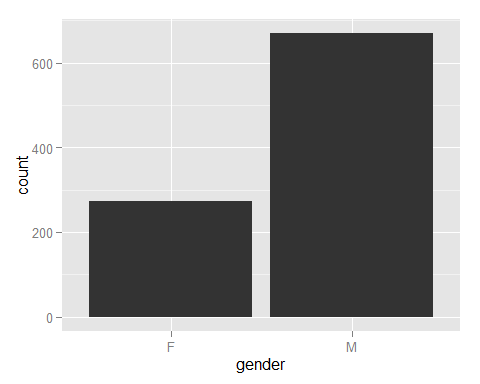
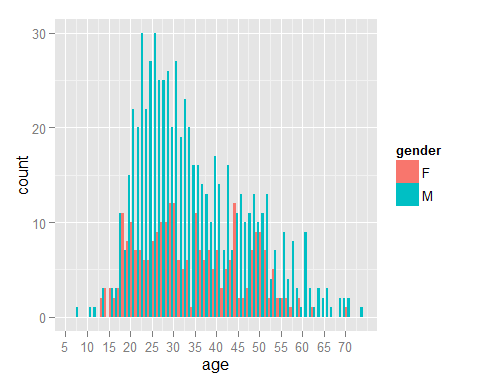
A rating of 4 looks to be the most common followed by 3 and then 5.



Histogram of movieid, this shows how much different ratings some movies are getting. Some movies are definitely being rated more often than others, one possible reason for this is that the movies are likely simply being viewed more times.

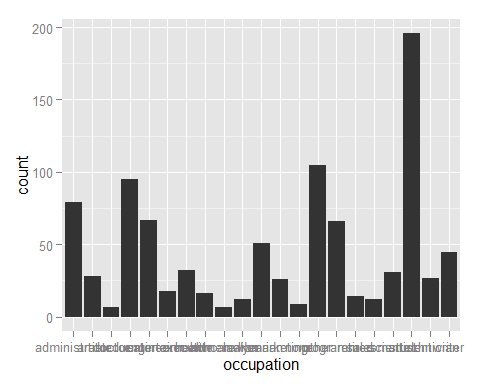
The following plots will look into the users 

This plot shows an almost normal distribution of age for the users, it shows that a majority of the users are between the ages of 18 and 30. The plot below shows there are more male than female users and the plot below that shows the same histogram of users' age but split by gender

Here we see Students are the most common of users, followed by educators, administrators and engineers.

qplot(occupation, data = occupation)



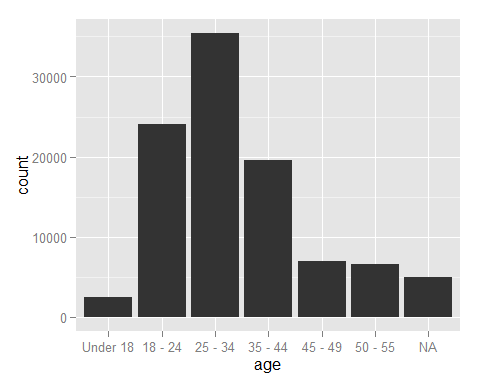
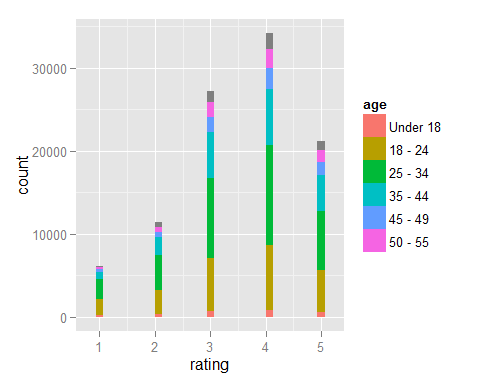
## Joining of the three datasets

Ratings and occupation are now joined, we see it's head

## userid movieid rating timestamp age gender occupation zipcode  
## 1 196 242 3 881250949 49 M writer 55105  
## 2 186 302 3 891717742 39 F executive 00000  
## 3 22 377 1 878887116 25 M writer 40206  
## 4 244 51 2 880606923 28 M technician 80525  
## 5 166 346 1 886397596 47 M educator 55113  
## 6 298 474 4 884182806 44 M executive 01581

We can now join with the last dataset films. And also it's head.

## userid movieid rating timestamp age gender occupation zipcode  
## 1 196 242 3 881250949 49 M writer 55105  
## 2 186 302 3 891717742 39 F executive 00000  
## 3 22 377 1 878887116 25 M writer 40206  
## 4 244 51 2 880606923 28 M technician 80525  
## 5 166 346 1 886397596 47 M educator 55113  
## 6 298 474 4 884182806 44 M executive 01581  
## movietitle  
## 1 Kolya (1996)  
## 2 L.A. Confidential (1997)  
## 3 Heavyweights (1994)  
## 4 Legends of the Fall (1994)  
## 5 Jackie Brown (1997)  
## 6 Dr. Strangelove or: How I Learned to Stop Worrying and Love the Bomb (1963)  
## releasedate videoreleasedate  
## 1 24-Jan-1997 NA  
## 2 01-Jan-1997 NA  
## 3 01-Jan-1994 NA  
## 4 01-Jan-1994 NA  
## 5 01-Jan-1997 NA  
## 6 01-Jan-1963 NA  
## imdburl  
## 1 http://us.imdb.com/M/title-exact?Kolya%20(1996)  
## 2 http://us.imdb.com/M/title-exact?L%2EA%2E+Confidential+(1997)  
## 3 http://us.imdb.com/M/title-exact?Heavyweights%20(1994)  
## 4 http://us.imdb.com/M/title-exact?Legends%20of%20the%20Fall%20(1994)  
## 5 http://us.imdb.com/M/title-exact?imdb-title-119396  
## 6 http://us.imdb.com/M/title-exact?Dr.%20Strangelove%20or:%20How%20I%20Learned%20to%20Stop%20Worrying%20and%20Love%20the%20Bomb%20(1963)  
## unknown Action Adventure Animation Children.s Comedy Crime Documentary  
## 1 0 0 0 0 0 1 0 0  
## 2 0 0 0 0 0 0 1 0  
## 3 0 0 0 0 1 1 0 0  
## 4 0 0 0 0 0 0 0 0  
## 5 0 0 0 0 0 0 1 0  
## 6 0 0 0 0 0 0 0 0  
## Drama Fantasy Film.Noir Horror Musical Mystery Romance Sci.Fi Thriller  
## 1 0 0 0 0 0 0 0 0 0  
## 2 0 0 1 0 0 1 0 0 1  
## 3 0 0 0 0 0 0 0 0 0  
## 4 1 0 0 0 0 0 1 0 0  
## 5 1 0 0 0 0 0 0 0 0  
## 6 0 0 0 0 0 0 0 1 0  
## War Western  
## 1 0 0  
## 2 0 0  
## 3 0 0  
## 4 1 1  
## 5 0 0  
## 6 1 0

It would make a lot more sense to split up the age into bins  

Information like imdburl would normally be very useful because from it we can gather more information through its API like the actors in the movie and the runtime, information like this can only help improve a predictive model, these all can be looked into at a later time and so can be removed for now, the id columns are also no longer needed and can also be removed. Timestamp refers to the time the users made the rating and so won't be required. While it's concievable that the release date of a movie could impact people wanting to see it (Some people prefering movies from a certain time period over others), it's unlikely the time a rating was given would and so the rating column can also be excluded

At this point, I am considering removing movietitle but will leave it in there for now. My reason is because I see two possible models that could be built. One model could be to learn how a particular user rates movies based on the characteristics of the movies they've rated and then predict how they'd rate new movies based on the new movie's characteristics with the end result being that the model recommends to them the movies the algorithm predicts they would have rated highly. In this case, the name of the movie the user rated is not important and thus can be removed, its characteristics however are important, and here getting more information on the movie from imdb would really help. This model is user-centric and wouldn't be useful until the user has seen and rated enough movies.

Another possible model would be to not be user-specific but to instead be userinfo-specific, what this means is to learn not on the particular user, but on the user's information like age, occupation e.t.c While this would be a much less restrictive algorithm in that it doesn't require that the user rate a lot of movies to begin with because it predicts based on the user's info as opposed to the particular user, the downside here is that we are making an assumption that users with a similar background would rate particular movies similarly, this kind of assumption would only really be justified if we had a lot of personal information. There however is no harm in trying and see how well it does.

A different albeit more difficult model would be to use an unsupervised learning algorithm to somehow divide up the users into specific classes(clusters) with each cluster representing movie tastes/preferences. This would be a more general implementation as it wouldn't require the ratings column to make it's predictions.

For the first implementation, I'm splitting the dataset into 3, the train set, a cross-validation set as well as a test set, the reason for this is to detect if the algorithm is overfitting. I will run this using a boosted decision tree and report the errors for all three. I'll be using two different metrics to measure the error, the mean absolute error and the root mean square error. Both these metrics are great for this kind of problem.

A first run gives really good errors:

## Train set

|  |  |
| --- | --- |
| Mean Absolute Error | Root Mean Squared Error |
| 0.651983 | 0.932193 |

## CV Set

|  |  |
| --- | --- |
| Mean Absolute Error | Root Mean Squared Error |
| 0.708 | 0.994736 |

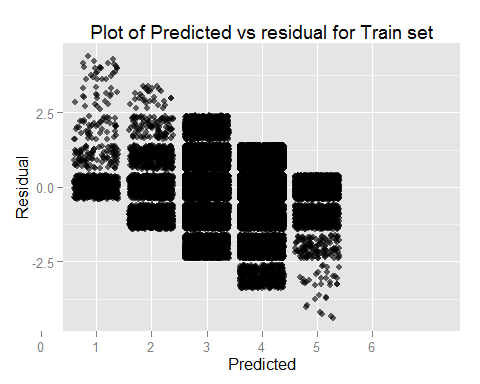
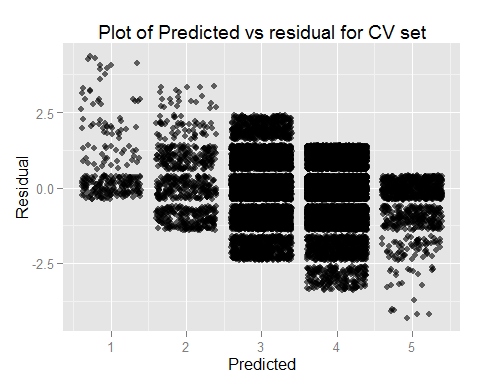
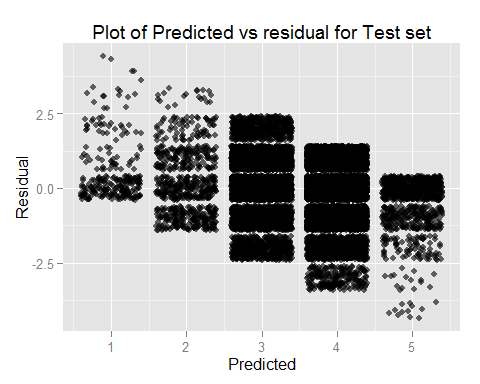
## Test Set

|  |  |
| --- | --- |
| Mean Absolute Error | Root Mean Squared Error |
| 0.7118 | 0.99955 |

There is a difference between the Train set errors and both the CV and test set errors but not enough to suggest overfitting.

While there is plenty of opportunity to tune the parameters and possibly get a lower error, the error given is good enough for what it's needed for.

Below is a residual vs fitted plot for each of the sets , this lets us see how the algorithm is performing on what it's predicting right and wrong. All the points at zero are the examples predicted correctly for that particular prediction with the predictions around zero how much wrong the predictions are.

These three plots are fairly identical, what this tells us is that there's not a lot of overfitting happening, also it shows the distribution of the errors, for ratings 1 and 2, our model sometimes predicts incorrectly but not by more than 2 ratings points above (meaning an actual rating of 1 sometimes gets predicted as 2 or 3), ratings of 3 sometimes gets predicted by up to two points in both directions and ratings 4/5 by not more than 2 points below. We also see that the errors closer to zero are darker than the others which suggests most of the errors are closer to zero. Below is a head of the predictions and actual ratings.

## actual predicted  
## 1 3 3  
## 2 2 3  
## 3 4 4  
## 4 3 4  
## 5 3 3  
## 6 2 4  
## 7 2 3  
## 8 3 4  
## 9 3 4  
## 10 2 3  
## 11 5 3  
## 12 3 3  
## 13 5 4  
## 14 5 4  
## 15 3 3  
## 16 4 4  
## 17 4 4  
## 18 4 4  
## 19 3 3  
## 20 4 4  
## 21 3 5  
## 22 3 4  
## 23 4 3  
## 24 4 4  
## 25 4 4

While it is possible to fine tune and boost the performance of this algorithm a bit more, the results we get aren't actually that bad which means given enough ratings by a particular customer, we can predict with a relatively high confidence, what other movies they likely would rate highly.

The second and third part of this project would be trying to implement the two other ideas discussed earlier.