Top 1000 Movies According to IMDB Users

Casey Freimund

library(knitr)  
library(ggplot2)  
library(fitdistrplus)

## Loading required package: MASS

## Loading required package: survival

## Loading required package: npsurv

## Loading required package: lsei

## Finding The Data

#DATA LOADER  
#A function that takes in an empty vector and reads in the html files that contain data on the top 1000 movies on IMDB. It then combines them into one character vector and outputs that vector, 'movies'  
#v = empty vector  
html\_loader=function(v){  
 imdb50=readLines("imdb50.html")  
 imdb100=readLines("imdb100.html")  
 imdb150=readLines("imdb150.html")  
 imdb200=readLines("imdb200.html")  
 imdb250=readLines("imdb250.html")  
 imdb300=readLines("imdb300.html")  
 imdb350=readLines("imdb350.html")  
 imdb400=readLines("imdb400.html")  
 imdb450=readLines("imdb450.html")  
 imdb500=readLines("imdb500.html")  
 imdb550=readLines("imdb550.html")  
 imdb600=readLines("imdb600.html")  
 imdb650=readLines("imdb650.html")  
 imdb700=readLines("imdb700.html")  
 imdb750=readLines("imdb750.html")  
 imdb800=readLines("imdb800.html")  
 imdb850=readLines("imdb850.html")  
 imdb900=readLines("imdb900.html")  
 imdb950=readLines("imdb950.html")  
 imdb1000=readLines("imdb1000.html")  
   
 movies=c(imdb50,imdb100,imdb150,imdb200,imdb250,imdb300,imdb350,imdb400,imdb450,imdb500,imdb550,imdb600,imdb650,imdb700,imdb750,imdb800,imdb850,imdb900,imdb950,imdb1000)  
   
 return(movies)  
}

#TITLE FINDER  
#A Function that takes in a character string and searches for a pattern that contains the movie title. It then 'gets rid' of any unnecessary symbols and/or punctuations that don't pertain to the movie and outputs a vector that contains the movie titles, 'titles'.  
#char.v = character string  
title.finder=function(char.v){  
   
 title.pattern=">\\s+<img\\s+alt.\*"  
 start.pat="([0-9]|[A-Z])"  
 stop.pat="\"$"  
  
 title.junk=regmatches(char.v,regexpr(title.pattern,char.v))  
 titles=substr(title.junk,start = regexpr(start.pat,title.junk),stop = regexpr(stop.pat,title.junk)-1)  
 titles=titles[titles!=""]  
 return(titles)  
}

#GENRE FINDER  
#A funtion that takes in a character string and searches for a pattern that contains the movies' genre(s). It then 'gets rid' of any unnecessary symbols and/or punctuations that don't pertain to the movie genre(s) and outputs a vector that contains the movie genres, 'genre'.  
#char.v = character string  
genre.finder=function(char.v){  
   
 genre.pattern="[A-Z].\*\\s+</span>"  
 start.pat="[A-Z]"  
 stop.pat="[a-z]\\s+</span>"  
   
 genre.junk=regmatches(char.v,regexpr(genre.pattern,char.v))  
 genre=substr(genre.junk,start = regexpr(start.pat,genre.junk),stop = regexpr(stop.pat,genre.junk))  
   
 return(genre)  
}

#ACTOR FINDER  
#A funtion that takes in a character string and searches for the location of the top 4 actors/actresses. It then 'gets rid' of any unnecessary symbols and/or punctuations that don't pertain to their names and outputs a vector that contains the top 4 paid actors and actresses from each movie, 'actor'.  
#char.v = character string  
actor.finder=function(char.v){  
   
 actors.pattern="Stars:"  
 stars.loc=grep(actors.pattern,char.v)  
   
 a1.loc=stars.loc+2  
 a2.loc=stars.loc+4  
 a3.loc=stars.loc+6  
 a4.loc=stars.loc+8  
  
 a1=char.v[a1.loc]  
 a2=char.v[a2.loc]  
 a3=char.v[a3.loc]  
 a4=char.v[a4.loc]  
  
 stop.pat="</a>"  
  
 a1=substr(a1,start = 2,stop = regexpr(stop.pat,a1)-1)  
 a2=substr(a2,start = 2,stop = regexpr(stop.pat,a2)-1)  
 a3=substr(a3,start = 2,stop = regexpr(stop.pat,a3)-1)  
 a4=substr(a4,start = 2,stop = regexpr(stop.pat,a4)-1)  
   
 actor=paste(a1,a2,a3,a4,sep = ", ")  
   
 return(actor)  
}

#REVENUE FINDER  
#A funtion that takes in a character string and searches for the location of the gross US revenue of a movie. It then 'gets rid' of any unnecessary symbols and/or punctuations that don't pertain to the gross US revenue and outputs a vector that contains the the gross US revenue for each movie, 'rev'. If gross US revenue isn't listed, gross revenue becomes NA.  
#char.v = character string  
rev.finder=function(char.v){  
   
 rev.pat="Votes:"  
 rev.loc=grep(rev.pat,char.v)  
 rev.loc=rev.loc+3  
 rev=char.v[rev.loc]  
 rev=substr(rev,start = regexpr("\\$[0-9]",rev)+1,stop = regexpr("M",rev)-1)  
 rev=as.numeric(rev)  
 return(rev)  
}

#RELEASE DATE FINDER  
#A funtion that takes in a character string and searches for the location of the year when the movie was released. It then gets rid of any unnecessary symbols and/or punctuations and outputs a vector that contains the year in which the movie was released, 'date'.  
#char.v = character string  
year.finder=function(char.v){  
   
 date.pat="lister-item-year.\*[0-9]{4}"  
 date.junk=regmatches(char.v,regexpr(date.pat,char.v))  
 date=substr(date.junk,start = regexpr("[0-9]",date.junk),stop = length(date.junk))  
 date=as.numeric(date)  
   
 return(date)  
}

#RATING/VOTES FINDER  
#A funtion that takes in a character string and searches for the location of the rating/voting statistics. It then isolates the average rating into 1 variable, 'rates', and the total number of votes, 'votes', into another variable. Then it gets rid of any unnecessary symbols and/or punctuations and outputs a list of 2 vectors that contain the average rating by IMDB users `$rating` and the total number of votes `$votes`.  
#char.v = character string  
vote.finder=function(char.v){  
   
 vote.pat="Users rated.\*votes"  
 vote.junk=regmatches(char.v,regexpr(vote.pat,char.v))  
   
 rate=substr(vote.junk,start = regexpr("[0-9]",vote.junk),stop = regexpr("[0-9]/[0-9]{2}",vote.junk))  
 rate=as.numeric(rate)  
   
 votes=substr(vote.junk,start = regexpr("\\([0-9]",vote.junk)+1,stop = regexpr("\\svotes",vote.junk))  
 votes=gsub("(\\,)+","",votes)  
 votes=as.numeric(votes)   
   
 return(list(rating=rate,votes=votes))   
}

#INFO FINDER  
#A function that takes in a character string and searches for the rating, total votes, title, year released, genre, top 4 paid actors, and gross US revenue for a list of movies and outputs a chart of the previously stated categories.  
#v = character string  
info.finder=function(v){  
   
 Rating=vote.finder(v)$rating  
 Total\_Votes=vote.finder(v)$votes  
 Title=title.finder(v)  
 Year=year.finder(v)  
 Genre=genre.finder(v)  
 Actors=actor.finder(v)  
 Revenue=rev.finder(v)  
   
 movie.chart=cbind.data.frame(Rating,Total\_Votes,Title,Year,Genre,Actors,Revenue)  
   
 movie.chart$Title=as.character(movie.chart$Title)  
 movie.chart$Genre=as.character(movie.chart$Genre)  
 movie.chart$Actors=as.character(movie.chart$Actors)  
   
 return(movie.chart)  
}

#Reads in the movie text for the top 1000 movies and puts the relevant information into a dataframe called 'movies'  
movie.text=0  
movie.text=html\_loader(movie.text)  
movies=info.finder(movie.text)

#Checks that data aligns properly by printing the first 5 and last 5 movies  
kable(movies[1:5,],caption = "Movies #1-5")

Movies #1-5

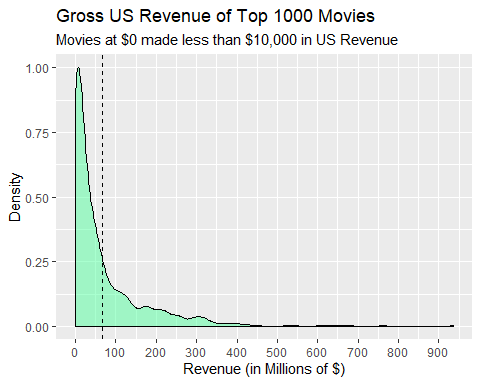
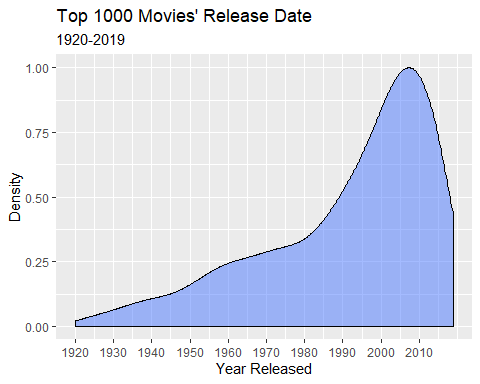
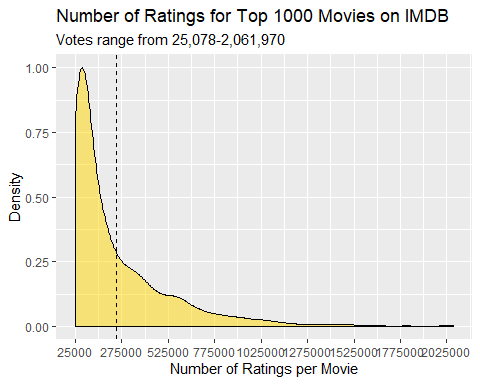
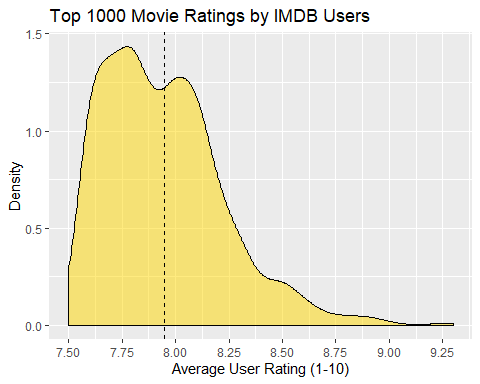
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Rating | Total\_Votes | Title | Year | Genre | Actors | Revenue |
| 9.3 | 2061970 | The Shawshank Redemption | 1994 | Drama | Tim Robbins, Morgan Freeman, Bob Gunton, William Sadler | 28.34 |
| 9.2 | 1414562 | The Godfather | 1972 | Crime, Drama | Marlon Brando, Al Pacino, James Caan, Diane Keaton | 134.97 |
| 9.0 | 2028211 | The Dark Knight | 2008 | Action, Crime, Drama | Christian Bale, Heath Ledger, Aaron Eckhart, Michael Caine | 534.86 |
| 9.0 | 981305 | The Godfather: Part II | 1974 | Crime, Drama | Al Pacino, Robert De Niro, Robert Duvall, Diane Keaton | 57.30 |
| 8.9 | 1468152 | The Lord of the Rings: The Return of the King | 2003 | Adventure, Drama, Fantasy | Elijah Wood, Viggo Mortensen, Ian McKellen, Orlando Bloom | 377.85 |

kable(movies[996:1000,],caption = "Movies #996-1000")

Movies #996-1000

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Rating | Total\_Votes | Title | Year | Genre | Actors | Revenue |
| 996 | 7.5 | 332546 | Collateral | 2004 | Crime, Drama, Thriller | Tom Cruise, Jamie Foxx, Jada Pinkett Smith, Mark Ruffalo | 101.01 |
| 997 | 7.5 | 395231 | Ice Age | 2002 | Animation, Adventure, Comedy | Denis Leary, John Leguizamo, Ray Romano, Goran Visnjic | 176.39 |
| 998 | 7.5 | 367291 | Gangs of New York | 2002 | Crime, Drama | Leonardo DiCaprio, Cameron Diaz, Daniel Day-Lewis, Jim Broadbent | 77.81 |
| 999 | 7.5 | 309201 | Batman | 1989 | Action, Adventure | Michael Keaton, Jack Nicholson, Kim Basinger, Robert Wuhl | 251.19 |
| 1000 | 7.5 | 142584 | The Fly | 1986 | Drama, Horror, Sci-Fi | Jeff Goldblum, Geena Davis, John Getz, Joy Boushel | 40.46 |

## Probability Distributions for Numerical Data

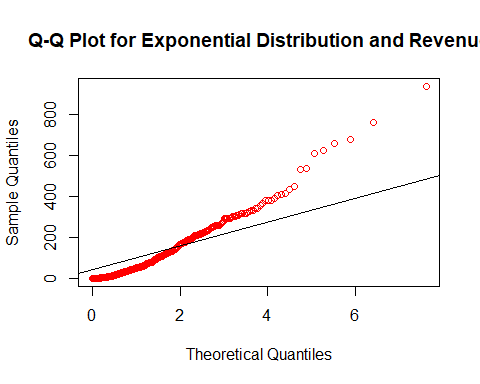
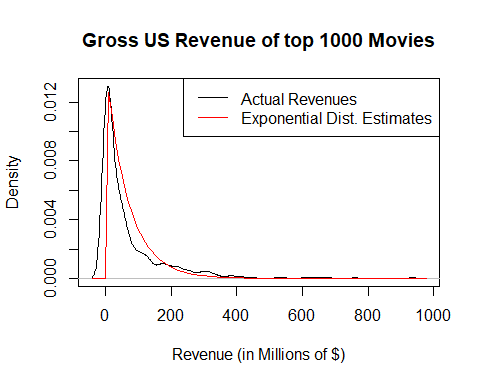


## Distribution Fitting to Numerical Data

#### Revenue

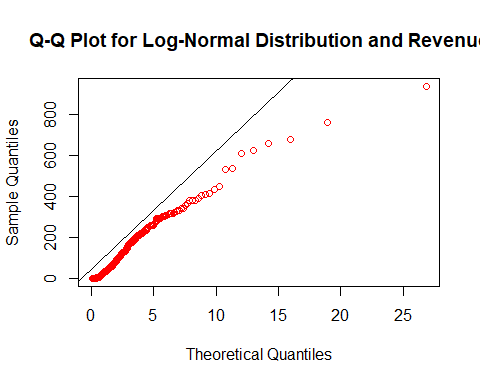
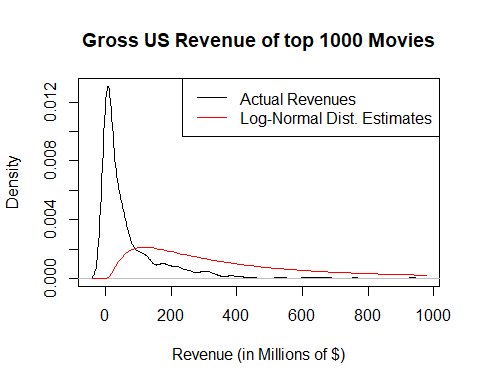
When fitting the exponential and log-normal distribution to the revenue data, I opted to omit the movies that didn’t have a gross US revenue reported. If I were to insert values in place of the NA values, it would have skewed my distributions with parameters that truly didn’t reflect the data set. When I was fitting the log-normal distribution to the revenue data, I had to omit values that were equal to 0, because log(0) is undefined and if I were to input a small decimal equal to about 0, 0.000001 for example, the log of my small decimal value would result in large negative numbers and potentially skew my parameter estimates.

##### -Exponential Distribution



##   
## One-sample Kolmogorov-Smirnov test  
##   
## data: movies$Revenue  
## D = 0.21355, p-value < 2.2e-16  
## alternative hypothesis: two-sided

##### -Log-Normal Distribution



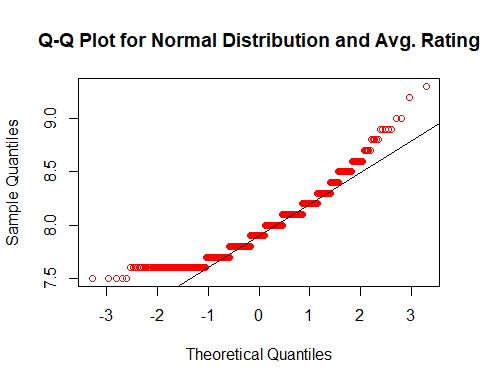
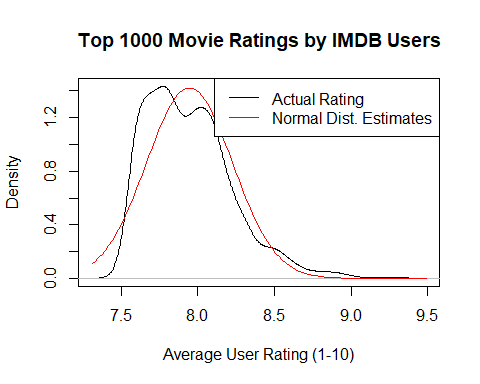
##   
## One-sample Kolmogorov-Smirnov test  
##   
## data: revs.gr8r.0  
## D = 0.15567, p-value < 2.2e-16  
## alternative hypothesis: two-sided

|  |  |  |
| --- | --- | --- |
| Distribution Fitted to Revenue | Parameters | Estimated Values |
| Exponential |  | 0.01474 |
|  |  |  |
| Log-Normal |  | 5.74328 |
|  |  | 0.96797 |

After fitting an exponential distribution and log-normal distribution to the revenue data, I ran a few tests to determine their good-ness of fit relative to the data. By creating a density plot of the revenue data and overlaying both distributions, I was able to see what distributions appeared to ‘fit’ the data. Through this test, it appeared that the exponential distribution best fit the revenue data, but to be sure I conducted two more tests. My next test was to create a Q-Q Plot for both distributions. For the exponential distribution, the plotted points fall closely around the identity line, but don’t fall on the line enough for us to say the exponential distribution fits our revenue data, especially as our Revenue increases. When looking at the Q-Q Plot for the log-normal distribution, we don’t have any points falling on our identity line, making the log-normal distribution a poor fit for the revenue data. My final test for the exponential and log-normal distributions was a Kolmogorov-Smirnov Test. Both tests resulted in p-values close to 0, therefore making both distributions ‘appear’ to fit better than the actually do. Finally, from these three tests, I’ve concluded that our exponential distribution best fits our revenue data.

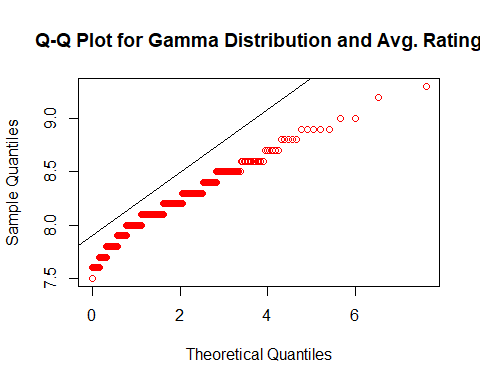
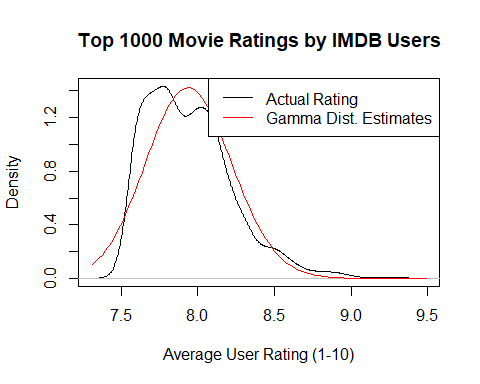
#### Average User Rating

##### -Gaussian (Normal) Distribution



##   
## One-sample Kolmogorov-Smirnov test  
##   
## data: movies$Rating  
## D = 0.13325, p-value = 7.772e-16  
## alternative hypothesis: two-sided

##### -Gamma Distribution



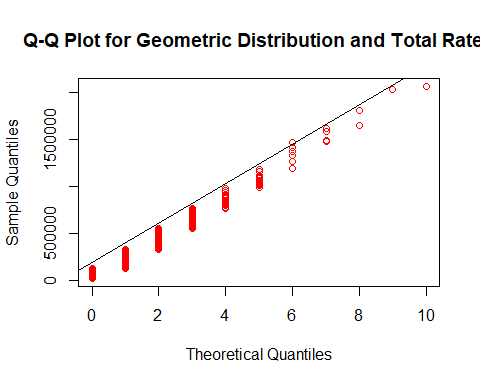
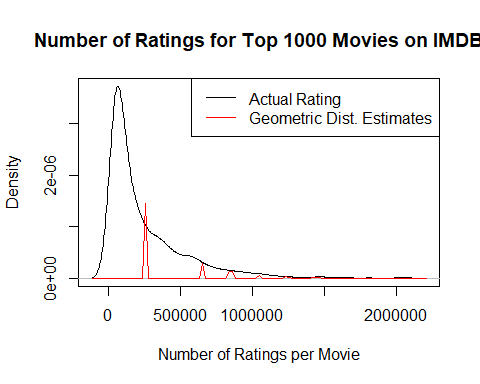
##   
## One-sample Kolmogorov-Smirnov test  
##   
## data: movies$Rating  
## D = 1, p-value < 2.2e-16  
## alternative hypothesis: two-sided

|  |  |  |
| --- | --- | --- |
| Distribution Fitted to Average User Rating | Parameters | Estimated Values |
| Gaussian |  | 7.9457 |
|  |  | 0.0787 |
|  |  |  |
| Gamma | (shape) | 802.2107 |
|  | (scale) | 0.0099 |

When I fit a normal and gamma distribution to the average user rating data, I found that there wasn’t significant evidence that either distribution fit better than the other or that either distribution fit the data well. When I overlayed my distributions with the average user rating density plot, I found that both distributions had a similar shape, relative to eachother, and that they both ‘fit’ the sample data in an ‘ok’ way. Comparing their Q-Q Plots, neither distributions had plotted points that fell mostly on the identity line and both showed signs of skewness. After running a KS test, both tests yield a p-value close to 0, making both distributions ‘appear’ to be a good fit. Based on these three tests, I’ve concluded that there wasn’t enough evidence to prove that either distribution is a good fit for the average user rating data.

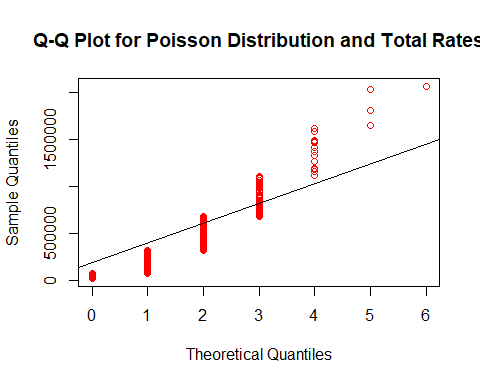
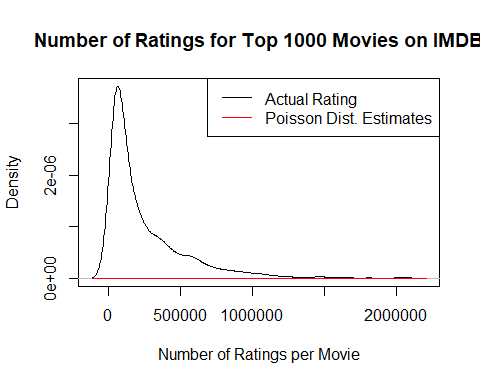
#### Number of Votes

##### -Geometric Distribution



##   
## One-sample Kolmogorov-Smirnov test  
##   
## data: movies$Total\_Votes  
## D = 0.10778, p-value = 1.626e-10  
## alternative hypothesis: two-sided

##### -Poisson Distribution



|  |  |  |
| --- | --- | --- |
| Distribution Fitted to Total Ratings per Movie | Parameters | Estimated Values |
| Geometric |  | 4.06095e-06 |
|  |  |  |
| Poisson |  | 246247.424 |

Finally, I fit a geometric and poisson distribution to the total number of votes per movie data set. While testing the geometric distribution with a Q-Q Plot, it appeared that the geometric distribution was not a good fit, as seen above. I also found that when I plotted the number of ratings density plot over my estimated geometric distribution, they were lacking in similar shape. Therefore it did not seem plausible to conclude that the geometric distribution was a good fit for this data set. Meanwhile, the KS test said otherwise, but due to the shape of both density plots side by side I concluded that the the geometric distribution did not fit ‘well’ for this data set. In my attempt to fit a poisson distribution to the total number of votes data, I plotted the density of my distribution with the actual data, and found that my distribution resulted in a line that did not resemble the actual density plot at all. Then by making a Q-Q Plot of the data I was also able to confirm that the poisson distribution was also not a good fit for the total number of ratings data.

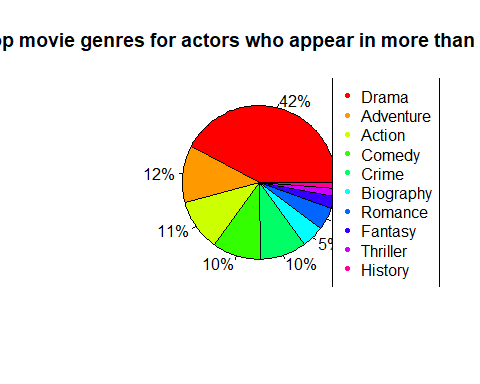
#### Good Acting or Good Movie Genre?

Under the “Stars” heading for each movie, IMDB lists the top 4 highest billed actors for each movie. Shown in the table below is a list of the top 5 actors who are one of the highest billed actors of a movie, in more than 5 of IMDB’s ‘Top 1000 Movies’ list. The final column in the table, Top2Genres, consists of the two genres that the actor would most likely appear in, if that movie made this top 1000 list. For example, of the 16 ‘Top 1000 Movies’ that Robert De Niro has starred in, if we were to draw one of them at random, they would most likely be listed as either a Drama and/or Crime movie.

kable(top.actor[1:5,])

|  |  |  |
| --- | --- | --- |
| Actors | NumberOfMovies | Top2Genres |
| Robert De Niro | 16 | Drama, Crime |
| Brad Pitt | 13 | Drama, Crime |
| Tom Hanks | 13 | Drama, Adventure |
| Al Pacino | 12 | Drama, Crime |
| Clint Eastwood | 12 | Western, Drama |

But, how does an actor appear as one of the highest billed actors in multiple movies from the ‘Top 1000 Movies’ list? Could it have something to do with their acting ability, or could it be the type of movie they chose to act in, that makes this movie a success. I decided to explore this hypothesis with the same data set. I started off by locating every actor who appeared in five or more movies on IMDB’s ‘Top 1000 Movies’ list. I then took all 86 ‘top actors’ and found the genre(s) of the movies they appeared in. Shown below is a pie chart of the top movie genres that an actor would appear in, if they were to appear in more than 5 movies on this list.



From this, I was able to find that the majority of these ‘top actors,’ have acted in a Drama movie. What I found to be most significant about this pie chart was the second most frequent movie genre, that these 86 ‘top actors’ may act in, was an Adventure movie. However, it’s more than three times smaller than the percentage of actors who have acted in Drama Movies. Raising the question, are these considered the ‘top actors’ due to they’re great at acting ability, or because they seem to land roles in a popular movie genre.