R IMDB Webscraping

Guide: https://www.analyticsvidhya.com/blog/2017/03/beginners-guide-on-web-scraping-in-r-using-rvest-with-hands-on-knowledge/

I will be using the guide listed above on the 2020 IMDB movie list in order to learn the basics of webscraping with the rvest library. I will also be using the dpylr and tidyverse libraries.

Libraries

```
library(rvest)

## Warning: package 'rvest' was built under R version 4.0.5

library(dplyr)

## Warning: package 'dplyr' was built under R version 4.0.5

library(tidyverse)

## Warning: package 'tidyverse' was built under R version 4.0.5

## Warning: package 'ggplot2' was built under R version 4.0.5

## Warning: package 'tibble' was built under R version 4.0.5

## Warning: package 'tidyr' was built under R version 4.0.5

## Warning: package 'forcats' was built under R version 4.0.5

#EDA Libraries
library(reshape2)

#ML Libraries
library(neuralnet)

## Warning: package 'neuralnet' was built under R version 4.0.5

library(caret)
```

URL and Data Loading

The data for this project will come from IMDB, specifically the sites 2020 Feature Films list (sorted by popularity). We'll save the URL as a string variable and load the site to a list variable with the read_html() command.

```
url <- 'https://www.imdb.com/search/title/?count=100&release_date=2020,2020&t
itle_type=feature'
```

```r
#Reading HTML code from site
webpage <- read_html(url)
```

Scraping

Now that we have a copy of the site saved in R, we can start pulling the data we need from it. Since most theaters were closed in the year 2020 due to the COVID Pandemic, we won't be looking at Gross Earnings. We'll still be looking at other variables though. Specifically, we want to pull the data for: rankings, titles, descriptions, runtimes, genre, metascores, ratings, votes, directors, and actors. To pull the data, I used the SelectorGadget Chrome Extension tool to find out which bit of HTML code was needed to locate the data within the saved webpage.

For the rankings, it was as simple as loading in the data, seeing how the output looked, and converting that output to a numeric from a string.

```r
ranking_html <- html_nodes(webpage, '.text-primary')
rank_data <- html_text(ranking_html)
head(rank_data)

## [1] "1." "2." "3." "4." "5." "6."

ranking <- as.numeric(rank_data)
head(ranking)

## [1] 1 2 3 4 5 6
```

Titles were even easier to load, because they were already set as the correct data-type.

```r
titles_html <- html_nodes(webpage,'.lister-item-header a')
title_data <- html_text(titles_html)
head(title_data)

## [1] "After Love"             "Tenet"                  "365 Days"
## [4] "The Devil All the Time" "Sonic the Hedgehog"     "The Postcard Killin
gs"
```

For the movie descriptions, I used gsub to remove all the newline marks, and looked at the data again to make sure it looked right.

```r
descriptions_html <- html_nodes(webpage,'.ratings-bar+ .text-muted')
description_data <- html_text(descriptions_html)
head(description_data)

## [1] "\nSet in the port town of Dover, Mary Hussain suddenly finds herself
a widow following the unexpected death of her husband. A day after the burial
, she discovers he has a secret just twenty-one miles across the English Chan
nel in Calais."
## [2] "\nArmed with only one word, Tenet, and fighting for the survival of t
he entire world, a Protagonist journeys through a twilight world of internati
onal espionage on a mission that will unfold in something beyond real time."
```

```
## [3] "\nMassimo is a member of the Sicilian Mafia family and Laura is a sal
es director. She does not expect that on a trip to Sicily trying to save her
relationship, Massimo will kidnap her and give her 365 days to fall in love w
ith him."
## [4] "\nSinister characters converge around a young man devoted to protecti
ng those he loves in a postwar backwoods town teeming with corruption and bru
tality."
## [5] "\nAfter discovering a small, blue, fast hedgehog, a small-town police
officer must help him defeat an evil genius who wants to do experiments on hi
m."
## [6] "\nA New York detective investigates the death of his daughter who was
murdered while on her honeymoon in London; he recruits the help of a Scandina
vian journalist when other couples throughout Europe suffer a similar fate."
```

```r
description_data <- gsub("\n", "", description_data)
head(description_data)
```

```
## [1] "Set in the port town of Dover, Mary Hussain suddenly finds herself a
widow following the unexpected death of her husband. A day after the burial,
she discovers he has a secret just twenty-one miles across the English Channe
l in Calais."
## [2] "Armed with only one word, Tenet, and fighting for the survival of the
entire world, a Protagonist journeys through a twilight world of internationa
l espionage on a mission that will unfold in something beyond real time."
## [3] "Massimo is a member of the Sicilian Mafia family and Laura is a sales
director. She does not expect that on a trip to Sicily trying to save her rel
ationship, Massimo will kidnap her and give her 365 days to fall in love with
him."
## [4] "Sinister characters converge around a young man devoted to protecting
those he loves in a postwar backwoods town teeming with corruption and brutal
ity."
## [5] "After discovering a small, blue, fast hedgehog, a small-town police o
fficer must help him defeat an evil genius who wants to do experiments on him
."
## [6] "A New York detective investigates the death of his daughter who was m
urdered while on her honeymoon in London; he recruits the help of a Scandinav
ian journalist when other couples throughout Europe suffer a similar fate."
```

When scraping out the runtimes, I had to remove the “min” suffix and convert to a numeric.

```r
runtimes_html <- html_nodes(webpage,'.runtime')
runtime_data <- html_text(runtimes_html)
head(runtime_data)
```

```
## [1] "89 min"  "150 min" "114 min" "138 min" "99 min"  "104 min"
```

```r
runtime_data <- as.numeric(gsub(" min", "", runtime_data))
head(runtime_data)
```

```
## [1]  89 150 114 138  99 104
```

While scraping the genres, I realized that because most movies fall under multiple genres analysis would be difficult. To cope with this, I kept only the first genre listed as that would typically be its main category. I then converted the genres to a factor, to make later analysis easier.

```
genres_html <- html_nodes(webpage, '.genre')
genres_data <- html_text(genres_html)
head(genres_data)
```

```
## [1] "\nDrama            "
## [2] "\nAction, Sci-Fi, Thriller            "
## [3] "\nDrama, Romance            "
## [4] "\nCrime, Drama, Thriller            "
## [5] "\nAction, Adventure, Comedy            "
## [6] "\nCrime, Drama, Thriller            "
```

```
#Remove \n and spaces, keep only first genre, and convert to factor
genres_data <- gsub("\n","",genres_data)
genres_data <- gsub(" ", "", genres_data)
genres_data <- as.factor(gsub(",.*", "", genres_data))
head(genres_data)
```

```
## [1] Drama  Action Drama  Crime  Action Crime
## 10 Levels: Action Adventure Animation Biography Comedy Crime Drama ... Mys
tery
```

When pulling out the metascore data, there were some movies that didn't have any. After scrolling through the site to figure out which ones they were, I replaced their metascore data with NA values and made sure that I had the right amount of data.

```
metascore_html <- html_nodes(webpage, '.metascore')
metascore_data <- html_text(metascore_html)
head(metascore_data)
```

```
## [1] "82        " "69        " "55        " "47        " "29        "
## [6] "73        "
```

```
metascore_data <- gsub(" ", "", metascore_data)
head(metascore_data)
```

```
## [1] "82" "69" "55" "47" "29" "73"
```

```
#7 Movies don't have metascore data. numbers: 3, 27, 41, 45, 48, 69, 77
length(metascore_data)
```

```
## [1] 93
```

```
for(i in c(3, 27, 41, 45, 48, 69, 77)){
  a<-metascore_data[1:(i-1)]
  b<-metascore_data[i:length(metascore_data)]
  metascore_data<-append(a,list("NA"))
  metascore_data<-append(metascore_data,b)
```

```
}
metascore_data <- as.numeric(metascore_data)

## Warning: NAs introduced by coercion

## Warning: NAs introduced by coercion

## Warning: NAs introduced by coercion

## Warning: NAs introduced by coercion

## Warning: NAs introduced by coercion

## Warning: NAs introduced by coercion

## Warning: NAs introduced by coercion

head(metascore_data)

## [1] 82 69 NA 55 47 29

length(metascore_data)

## [1] 100
```

Movie ratings were another easy variable to scrape. I just had to convert to a numeric.

```
ratings_html <- html_nodes(webpage,'.ratings-imdb-rating strong')
rating_data <- html_text(ratings_html)
head(rating_data)

## [1] "7.3" "7.4" "3.4" "7.1" "6.5" "5.7"

rating_data <- as.numeric(rating_data)
head(rating_data)

## [1] 7.3 7.4 3.4 7.1 6.5 5.7
```

As with runtimes, I had to clean the data (by removing commas instead of suffixes) before converting to a numeric.

```
votes_html <- html_nodes(webpage,'.sort-num_votes-visible span:nth-child(2)')
vote_data <- html_text(votes_html)
head(vote_data)

## [1] "1,930"   "459,059" "74,577"  "124,374" "116,444" "9,972"

vote_data <- as.numeric(gsub(",", "", vote_data))
head(vote_data)

## [1]   1930 459059  74577 124374 116444   9972
```

Directors were saved as a factor as well.

```
director_html <- html_nodes(webpage, '.text-muted+ p a:nth-child(1)')
director_data <- html_text(director_html)
head(director_data)
```

```
## [1] "Aleem Khan"        "Christopher Nolan" "Barbara Bialowas"
## [4] "Antonio Campos"    "Jeff Fowler"       "Danis Tanovic"
```

```
director_data <- as.factor(director_data)
```

Actors were also saved as factors.

```
actor_html <- html_nodes(webpage, '.lister-item-content .ghost+ a')
actor_data <- html_text(actor_html)
head(actor_data)
```

```
## [1] "Joanna Scanlan"        "John David Washington" "Anna Maria Sieklucka"
## [4] "Bill Skarsgård"        "Ben Schwartz"          "Jeffrey Dean Morgan"
```

```
actor_data <- as.factor(actor_data)
```

Assembling Dataframe

Once all the data is saved as individual variables, we can put them all together into one dataframe and call the str() function to make sure we have eveything in the right format.

```
movies <- data.frame(Rank = rank_data, Title = title_data,
                     Description = description_data, Runtime = runtime_data,
                     Genre = genres_data, Rating = rating_data,
                     Metascore = metascore_data, Votes = vote_data,
                     Director = director_data, Actor = actor_data)
str(movies)
```

```
## 'data.frame':    100 obs. of  10 variables:
##  $ Rank       : chr  "1." "2." "3." "4." ...
##  $ Title      : chr  "After Love" "Tenet" "365 Days" "The Devil All the Ti
me" ...
##  $ Description: chr  "Set in the port town of Dover, Mary Hussain suddenly
finds herself a widow following the unexpected death of he"| __truncated__ "A
rmed with only one word, Tenet, and fighting for the survival of the entire w
orld, a Protagonist journeys thro"| __truncated__ "Massimo is a member of the
Sicilian Mafia family and Laura is a sales director. She does not expect that
on a t"| __truncated__ "Sinister characters converge around a young man devot
ed to protecting those he loves in a postwar backwoods tow"| __truncated__ ..
.
##  $ Runtime    : num  89 150 114 138 99 104 113 112 96 160 ...
##  $ Genre      : Factor w/ 10 levels "Action","Adventure",..: 7 1 7 6 1 6 6
7 4 4 ...
##  $ Rating     : num  7.3 7.4 3.4 7.1 6.5 5.7 7.5 7.2 6.9 8.4 ...
##  $ Metascore  : num  82 69 NA 55 47 29 73 65 77 90 ...
##  $ Votes      : num  1930 459059 74577 124374 116444 ...
##  $ Director   : Factor w/ 100 levels "Aaron Schneider",..: 4 20 10 8 49 23
```

```
36 34 85 92 ...
##  $ Actor      : Factor w/ 94 levels "Adrien Brody",..: 45 46 7 14 11 41 16
12 44 58 ...
```

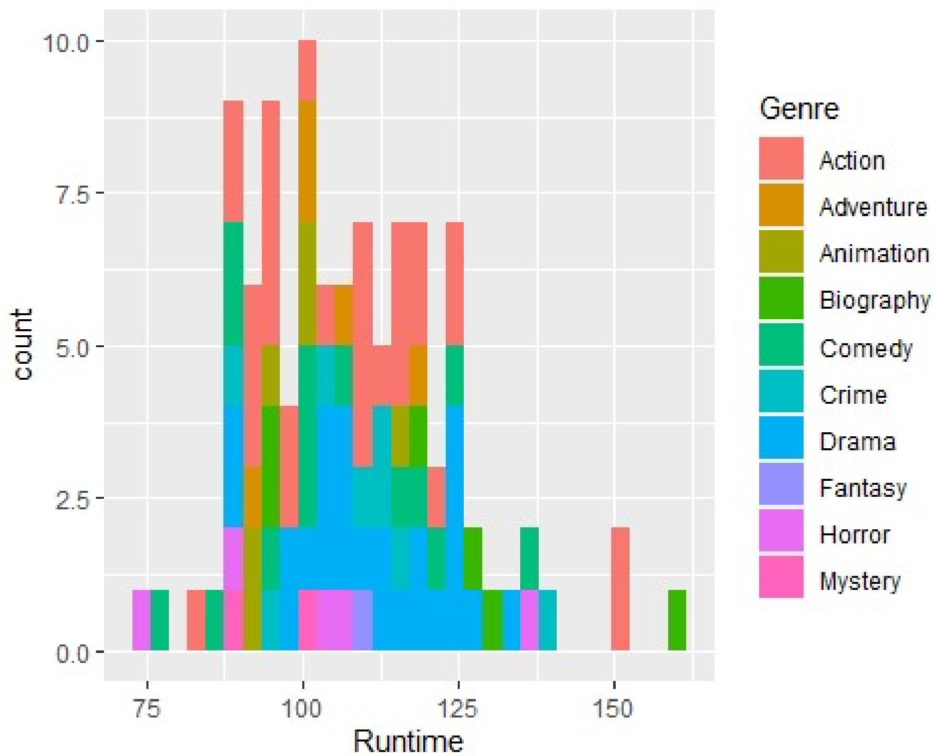
```
movies$Rank <- as.numeric(movies$Rank)
str(movies)
```

```
## 'data.frame':    100 obs. of  10 variables:
##  $ Rank       : num  1 2 3 4 5 6 7 8 9 10 ...
##  $ Title      : chr  "After Love" "Tenet" "365 Days" "The Devil All the Ti
me" ...
##  $ Description: chr  "Set in the port town of Dover, Mary Hussain suddenly
finds herself a widow following the unexpected death of he"| __truncated__ "A
rmed with only one word, Tenet, and fighting for the survival of the entire w
orld, a Protagonist journeys thro"| __truncated__ "Massimo is a member of the
Sicilian Mafia family and Laura is a sales director. She does not expect that
on a t"| __truncated__ "Sinister characters converge around a young man devot
ed to protecting those he loves in a postwar backwoods tow"| __truncated__ ..
.
##  $ Runtime    : num  89 150 114 138 99 104 113 112 96 160 ...
##  $ Genre      : Factor w/ 10 levels "Action","Adventure",..: 7 1 7 6 1 6 6
7 4 4 ...
##  $ Rating     : num  7.3 7.4 3.4 7.1 6.5 5.7 7.5 7.2 6.9 8.4 ...
##  $ Metascore  : num  82 69 NA 55 47 29 73 65 77 90 ...
##  $ Votes      : num  1930 459059 74577 124374 116444 ...
##  $ Director   : Factor w/ 100 levels "Aaron Schneider",..: 4 20 10 8 49 23
36 34 85 92 ...
##  $ Actor      : Factor w/ 94 levels "Adrien Brody",..: 45 46 7 14 11 41 16
12 44 58 ...
```

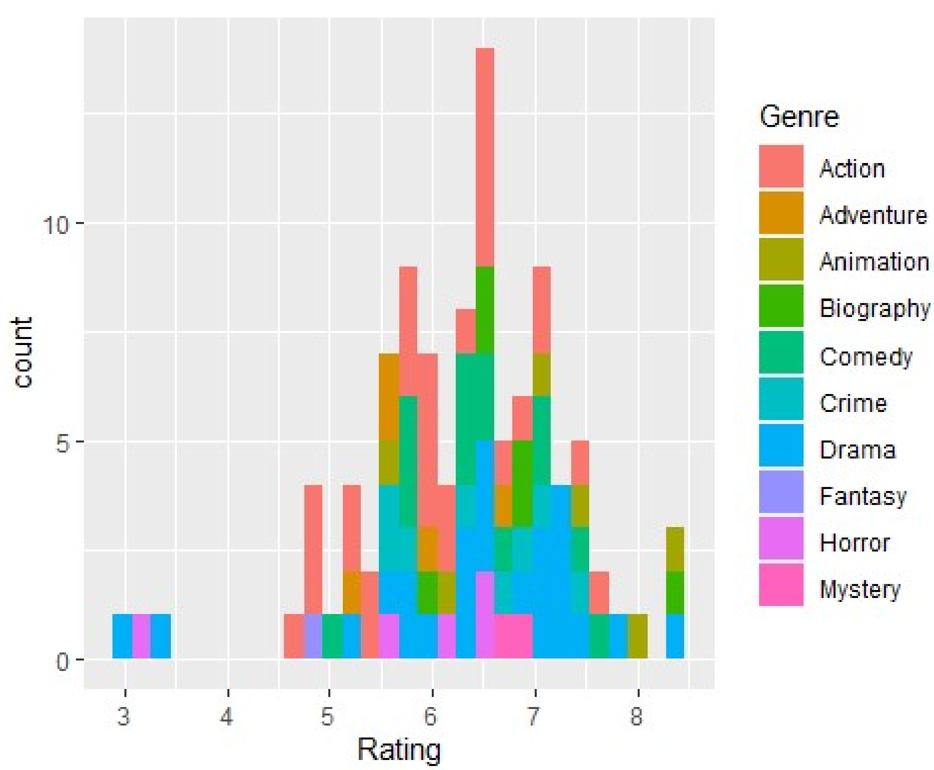
EDA and Plots

Now that we have a nice looking dataframe, it's time to make some simple graphs looking at the different relationships.

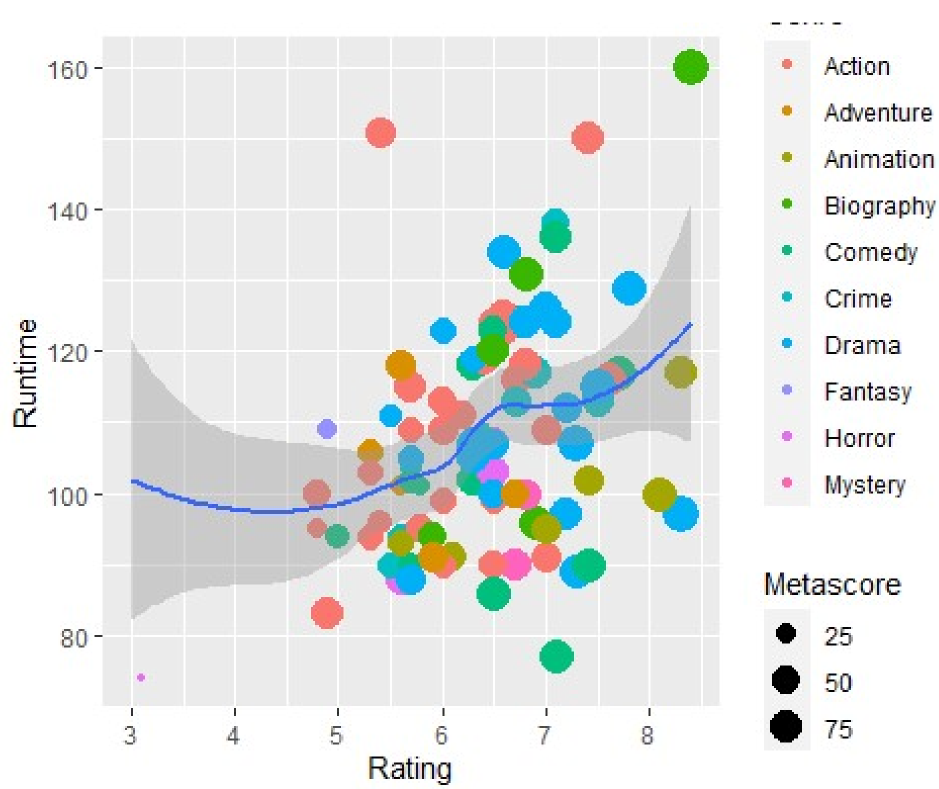
```
ggplot(movies, aes(x = Runtime, fill = Genre)) + geom_histogram(bins = 30)
```



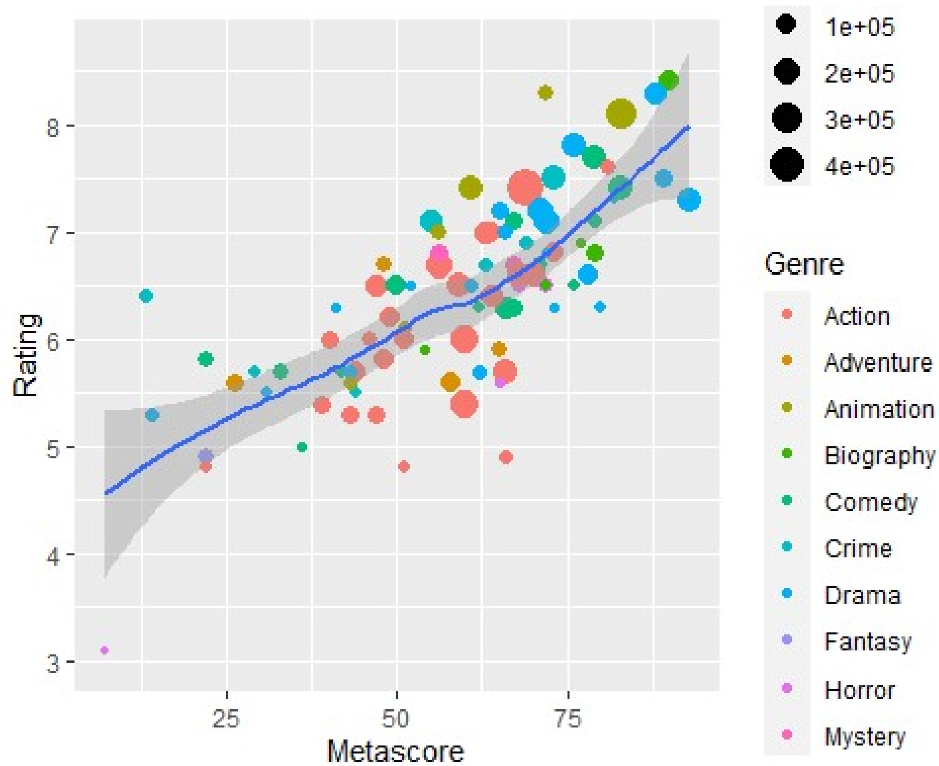

```r
ggplot(movies, aes(x = Rating, fill = Genre)) + geom_histogram(bins = 30)
```



```r
ggplot(movies, aes(x = Rating, y = Runtime)) +
  geom_point(aes(color = Genre, size = Metascore)) + geom_smooth()

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



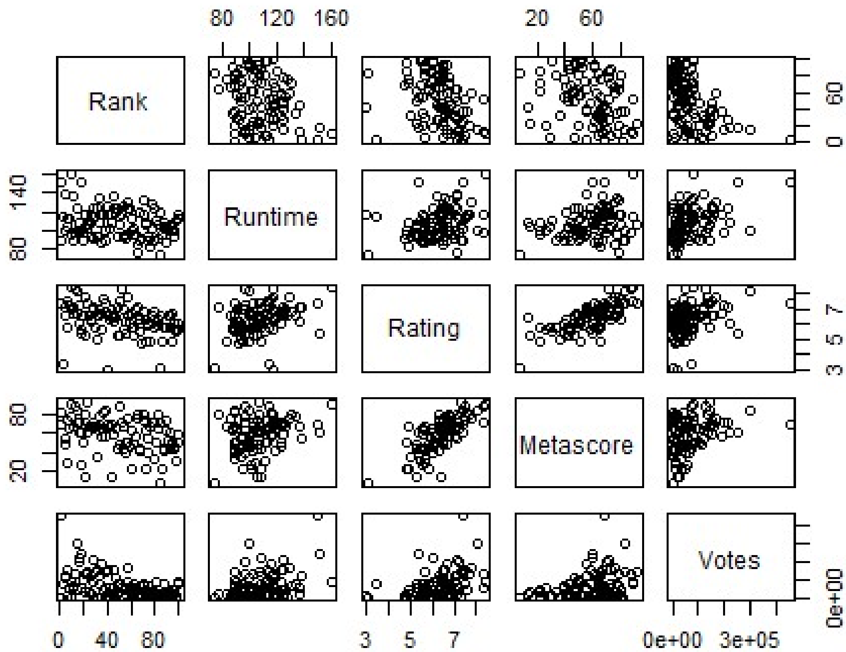

```r
ggplot(movies, aes(x = Metascore, y = Rating)) +
  geom_point(aes(color = Genre, size = Votes)) + geom_smooth()

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



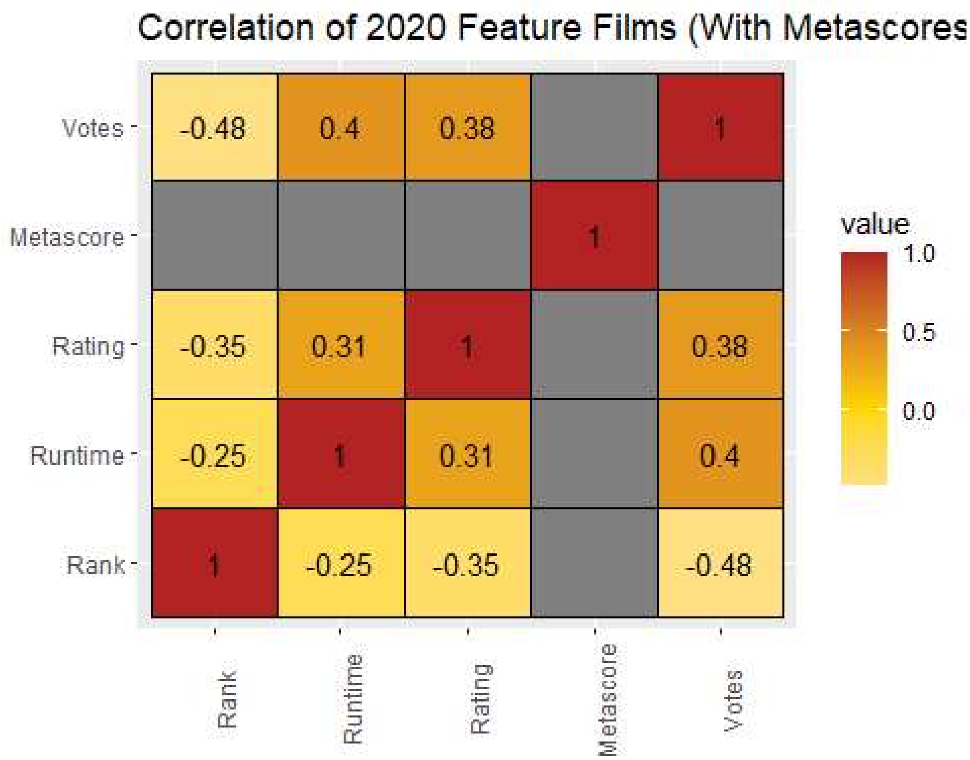
At a glance, we can see that most movies released in 2020 had a runtime of about 100 minutes and tended to be rated near 6.5. We can also see a slight positive correlation between rating and runtime, and a much stronger one between Metascore and Rating. To get a deeper look at these relationships, we can strip all non-numeric and variables with NAs from the data and plot the correlation values of each variable's interactions.

When making the tile plots, R removes all values for the correlations with an NA in one category, which means our Metascores are just blank correlations. To cope with this, I'll subset the data and remove the rows with an NA.

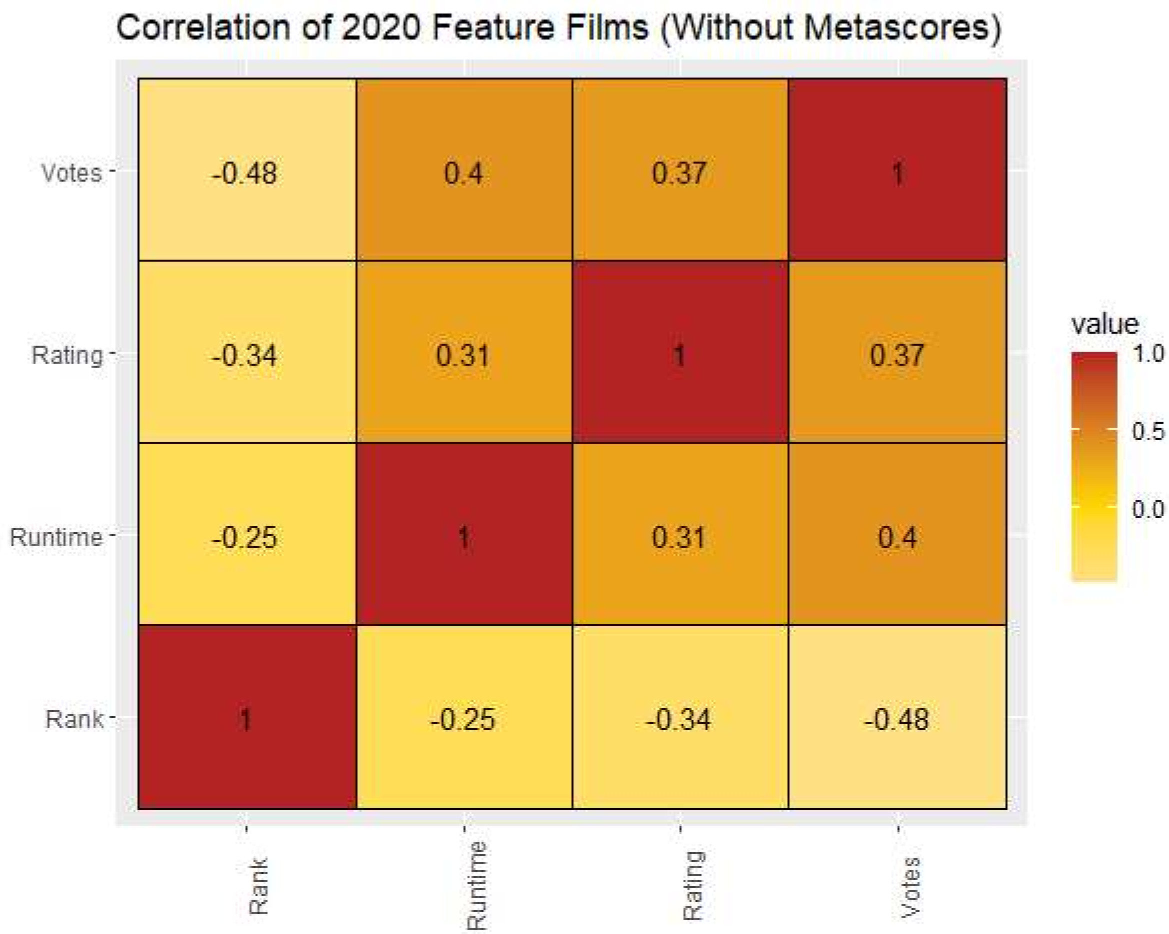
```
plot(movies[, -c(2, 3, 5, 9, 10)])
```



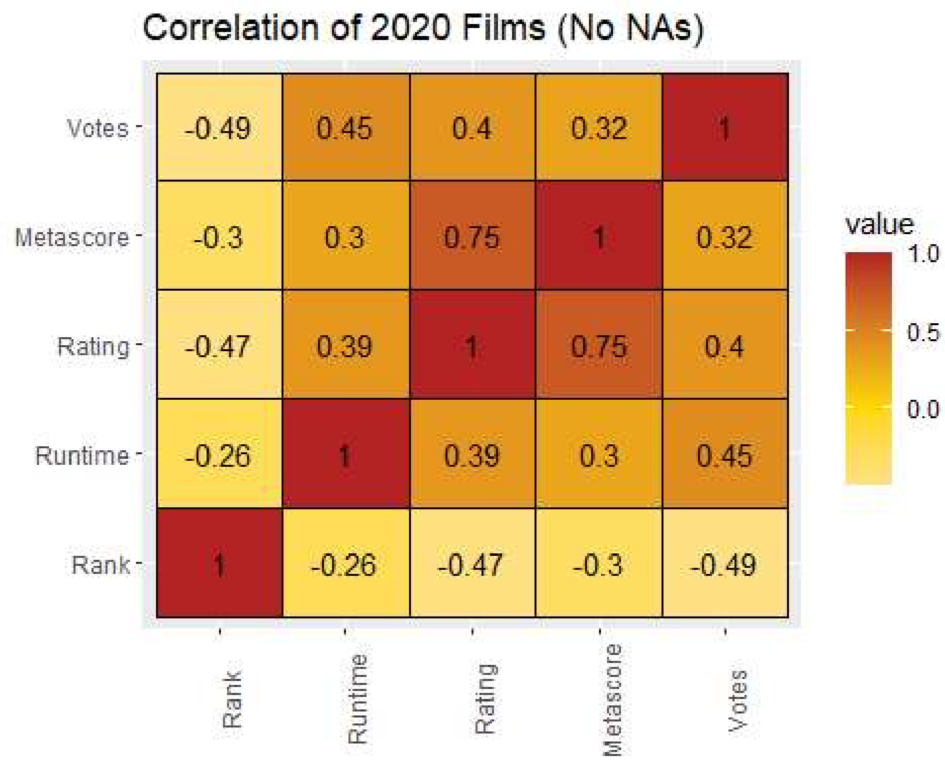

```
correlated <- movies[, -c(2, 3, 5, 9, 10)] %>%
  cor() %>%
  melt()

ggplot(correlated, aes(x = Var1, y = Var2, fill = value)) +
  scale_fill_gradient2(low = "antiquewhite", mid = "gold", high = "firebrick"
) +
  geom_tile(color = "black") +
  geom_text(aes(label = round(value, 2)), size = 4) +
  theme(axis.text.x = element_text(angle = 90), axis.title = element_blank())
+
  labs(title = "Correlation of 2020 Feature Films (With Metascores")

## Warning: Removed 8 rows containing missing values (geom_text).
```




```r
#Metascore is removed for the tile plot because the NA values prevent an accu
rate assessment of correlation.
correlated <- movies[, -c(2, 3, 5, 7, 9, 10)] %>%
  cor() %>%
  melt()

ggplot(correlated, aes(x = Var1, y = Var2, fill = value)) +
  scale_fill_gradient2(low = "antiquewhite", mid = "gold", high = "firebrick"
) +
  geom_tile(color = "black") +
  geom_text(aes(label = round(value, 2)), size = 4) +
  theme(axis.text.x = element_text(angle = 90), axis.title = element_blank())
+
  labs(title = "Correlation of 2020 Feature Films (Without Metascores")
```




```
correlated_noNA <- subset(movies, is.na(Metascore) == FALSE)
correlated_noNA <- correlated_noNA[, -c(2, 3, 5, 9, 10)] %>%
  cor() %>%
  melt()

ggplot(correlated_noNA, aes(x = Var1, y = Var2, fill = value)) +
  scale_fill_gradient2(low = "antiquewhite", mid = "gold", high = "firebrick"
) +
  geom_tile(color = "black") +
  geom_text(aes(label = round(value, 2)), size = 4) +
  theme(axis.text.x = element_text(angle = 90), axis.title = element_blank())
+
  labs(title = "Correlation of 2020 Films (No NAs)")
```



We can see through these plots that a movies rating and metascore correlate highly. Votes seem to have a consistent correlation with the other factors, although it's an inverse correlation with Rank. Rank actually only seems to have negative correlations, which is worrisome.

Deeper Analysis

The goal is to predict a movies rating based on generally applicable factors such as runtime, genre, metascore, and IMDB votes. To do this we'll use a neural network. We'll start by setting our seed so that the data will be reproducible, and removing the 7 rows with an NA for a metascore. Next, we use the model.matrix() function to convert our categorical variable (genre) into multiple binary columns. Then using a quick custom function to normalize the data, we split it into our test and train groups and double check the structure of the training group to ensure there won't be any problems.

```
set.seed(37) #Set the seed so that the data is reproducible
head(movies)
```

```
##   Rank                  Title
## 1    1             After Love
## 2    2                  Tenet
## 3    3               365 Days
## 4    4 The Devil All the Time
## 5    5      Sonic the Hedgehog
## 6    6  The Postcard Killings
##
```

```
Description
## 1 Set in the port town of Dover, Mary Hussain suddenly finds herself a wid
ow following the unexpected death of her husband. A day after the burial, she
discovers he has a secret just twenty-one miles across the English Channel in
Calais.
## 2                     Armed with only one word, Tenet, and fighting for the su
rvival of the entire world, a Protagonist journeys through a twilight world o
f international espionage on a mission that will unfold in something beyond r
eal time.
## 3          Massimo is a member of the Sicilian Mafia family and Laura is a sa
les director. She does not expect that on a trip to Sicily trying to save her
relationship, Massimo will kidnap her and give her 365 days to fall in love w
ith him.
## 4
Sinister characters converge around a young man devoted to protecting those h
e loves in a postwar backwoods town teeming with corruption and brutality.
## 5
After discovering a small, blue, fast hedgehog, a small-town police officer m
ust help him defeat an evil genius who wants to do experiments on him.
## 6                A New York detective investigates the death of his daught
er who was murdered while on her honeymoon in London; he recruits the help of
a Scandinavian journalist when other couples throughout Europe suffer a simil
ar fate.
##   Runtime  Genre Rating Metascore  Votes          Director
## 1      89  Drama    7.3        82   1930        Aleem Khan
## 2     150 Action    7.4        69 459059 Christopher Nolan
## 3     114  Drama    3.4        NA  74577  Barbara Bialowas
## 4     138  Crime    7.1        55 124374    Antonio Campos
## 5      99 Action    6.5        47 116444       Jeff Fowler
## 6     104  Crime    5.7        29   9972     Danis Tanovic
##                  Actor
## 1        Joanna Scanlan
## 2 John David Washington
## 3  Anna Maria Sieklucka
## 4        Bill Skarsgård
## 5          Ben Schwartz
## 6   Jeffrey Dean Morgan
```

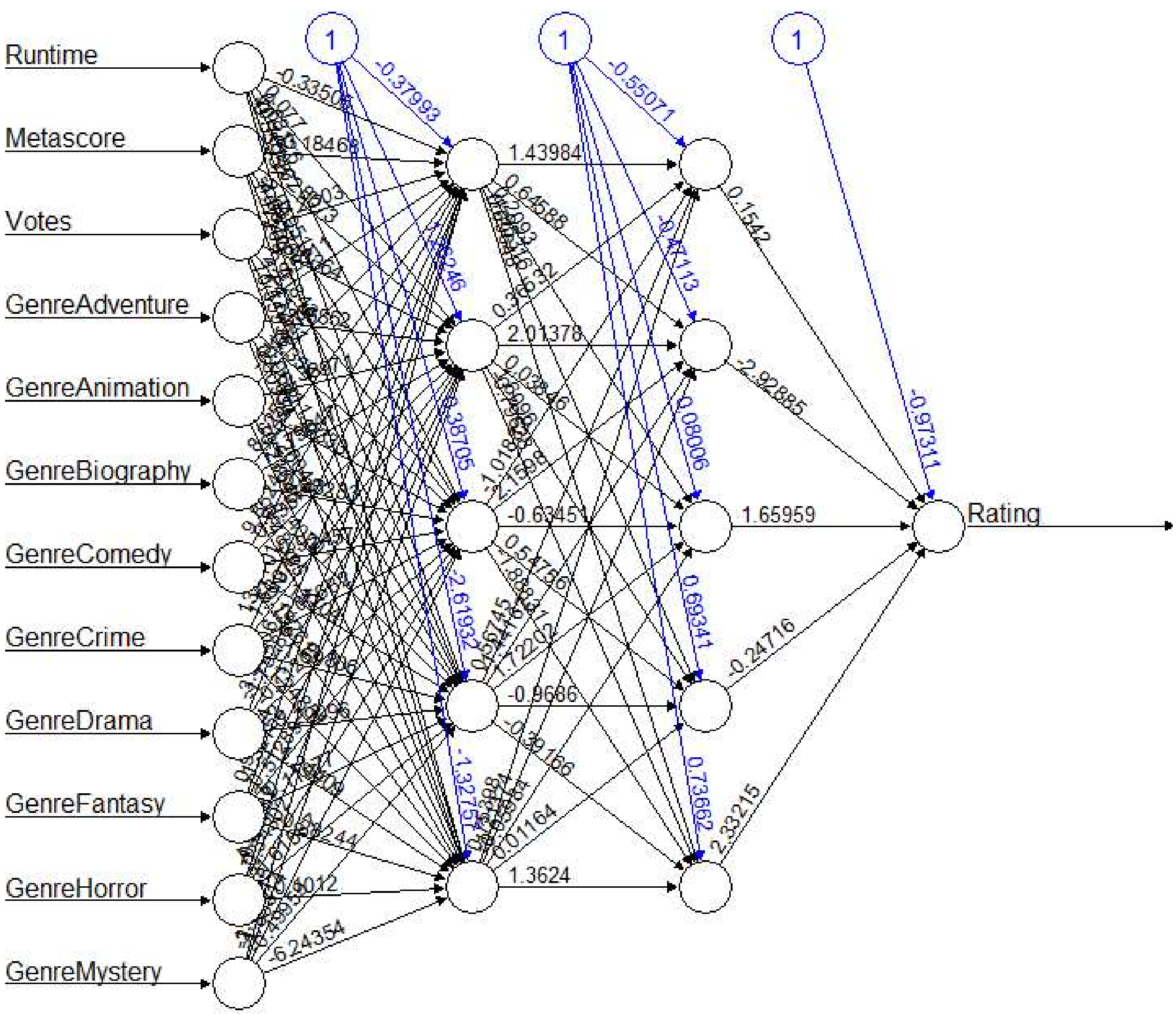
```
df <- subset(movies, is.na(Metascore) == FALSE)
str(df) #subset original frame and make sure it looks good
```

```
## 'data.frame':    93 obs. of  10 variables:
##  $ Rank       : num  1 2 4 5 6 7 8 9 10 11 ...
##  $ Title      : chr  "After Love" "Tenet" "The Devil All the Time" "Sonic
the Hedgehog" ...
##  $ Description: chr  "Set in the port town of Dover, Mary Hussain suddenly
finds herself a widow following the unexpected death of he"| __truncated__ "A
rmed with only one word, Tenet, and fighting for the survival of the entire w
orld, a Protagonist journeys thro"| __truncated__ "Sinister characters conver
ge around a young man devoted to protecting those he loves in a postwar backw
```

```
oods tow"| __truncated__ "After discovering a small, blue, fast hedgehog, a s
mall-town police officer must help him defeat an evil genius"| __truncated__
...
##  $ Runtime    : num  89 150 138 99 104 113 112 96 160 90 ...
##  $ Genre      : Factor w/ 10 levels "Action","Adventure",..: 7 1 6 1 6 6 7
4 4 10 ...
##  $ Rating     : num  7.3 7.4 7.1 6.5 5.7 7.5 7.2 6.9 8.4 6.7 ...
##  $ Metascore  : num  82 69 55 47 29 73 65 77 90 67 ...
##  $ Votes      : num  1930 459059 124374 116444 9972 ...
##  $ Director   : Factor w/ 100 levels "Aaron Schneider",..: 4 20 8 49 23 36
34 85 92 7 ...
##  $ Actor      : Factor w/ 94 levels "Adrien Brody",..: 45 46 14 11 41 16 1
2 44 58 80 ...

dummys <- model.matrix(~ Rating + Runtime + Metascore + Votes + Genre, data =
df)
dummys <- as.data.frame(dummys)

normalize <- function(x) {
  return ((x - min(x)) / (max(x) - min(x)))
}

maxmindf <- as.data.frame(lapply(dummys, normalize))

scaleddata <- as.data.frame(maxmindf)

ind <- sample(2, nrow(scaleddata), replace = TRUE, prob = c(0.7, 0.3)) #Rando
mly assign indexes with ~80% as 1 (training data)
train <- scaleddata[ind == 1,] #training data is everything with an index of
1
test <- scaleddata[ind == 2,] #testing data is everything with an index of 2
str(train)

## 'data.frame':    66 obs. of  14 variables:
##  $ X.Intercept.  : num  NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN ...
##  $ Rating        : num  0.792 0.811 0.755 0.642 0.774 ...
##  $ Runtime       : num  0.174 0.884 0.744 0.291 0.442 ...
##  $ Metascore     : num  0.872 0.721 0.558 0.465 0.674 ...
##  $ Votes         : num  0.00209 1 0.26939 0.25207 0.11154 ...
##  $ GenreAdventure: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ GenreAnimation: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ GenreBiography: num  0 0 0 0 0 1 1 0 0 0 ...
##  $ GenreComedy   : num  0 0 0 0 0 0 0 0 1 1 ...
##  $ GenreCrime    : num  0 0 1 0 0 0 0 0 0 0 ...
##  $ GenreDrama    : num  1 0 0 0 1 0 0 0 0 0 ...
##  $ GenreFantasy  : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ GenreHorror   : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ GenreMystery  : num  0 0 0 0 0 0 0 1 0 0 ...
```

Now that our data is ready to go, we create the neural network for our training data, and save it as variable nn. This will allow us to call it later for our testing data.

```
nn <- neuralnet(Rating ~ Runtime + Metascore + Votes + GenreAdventure +
                  GenreAnimation + GenreBiography + GenreComedy + GenreCrime
+
                  GenreDrama + GenreFantasy + GenreHorror + GenreMystery,
                data = train, hidden = c(5,5), linear.output = FALSE, thresho
ld = 0.01)
plot(nn)
```



Once our neural network is trained, we can run the testing data through and see how accurate it is. We use the compute() function to save the test results to a new variable which has its value rounded to make it easier to read before being converted from a temporary table to a new dataframe. We then use the table() function to look at the total number of true positive, false positives, true negatives, and false negatives.

```
nn_results <- compute(nn, test)
results <- data.frame(Actual = test$Rating, Predicted = nn_results$net.result
)
roundedresults<-sapply(results,round,digits=0)
roundedresultsdf=data.frame(roundedresults)
table(roundedresultsdf$Actual,roundedresultsdf$Predicted)

##
##      0  1
##   0  4  2
##   1  3 18

#Descaling
predicted=results$Predicted * abs(diff(range(dummys$Rating))) + min(dummys$Ra
ting)
actual=results$Actual * abs(diff(range(dummys$Rating))) + min(dummys$Rating)
comparison=data.frame(predicted,actual)
deviation=((actual-predicted)/actual)
comparison=data.frame(actual,predicted,deviation)
comparison

##    actual predicted    deviation
## 1     5.7  5.425794  0.048106257
## 2     7.5  7.405502  0.012599716
## 3     8.3  7.526811  0.093155287
## 4     7.2  7.223013 -0.003196318
## 5     6.0  7.089047 -0.181507825
## 6     5.4  6.999707 -0.296242019
## 7     7.7  7.293747  0.052760171
## 8     7.3  7.597033 -0.040689436
## 9     6.5  6.631472 -0.020226396
## 10    6.3  7.049993 -0.119046436
## 11    7.0  6.509881  0.070016974
## 12    5.7  7.030669 -0.233450653
## 13    6.4  5.402412  0.155873157
## 14    6.5  6.927053 -0.065700497
## 15    6.4  6.873036 -0.073911920
## 16    5.3  5.645688 -0.065224158
## 17    7.0  6.649156  0.050120635
## 18    5.4  5.207492  0.035649574
## 19    6.5  6.749431 -0.038373925
## 20    7.1  6.224134  0.123361371
## 21    6.3  5.714753  0.092896293
## 22    6.8  7.957075 -0.170158019
## 23    5.9  6.830103 -0.157644639
## 24    6.5  7.545498 -0.160845921
## 25    4.8  4.156292  0.134105831
```

```
## 26    6.7  5.663560  0.154692467
## 27    6.0  6.360481 -0.060080196
```

```
accuracy=1-abs(mean(deviation))
accuracy
```

```
## [1] 0.9754459
```

From our newest table, we can see that the neural net was correct in 20 of its 26 guesses (15 true positives and 5 true negatives). And if we compare the ratings to the predicted results after de-normalization, we get an accuracy of ~97.7%. With another dataset, results like this would be great. However, our data includes variables such as metascore and votes which cannot be decided in film production. This means that, as accurate as this model is, it's completely useless. Fun to make though.