What Makes a Good Scooby-Doo

April 30, 2021

Authored by: William C. Schooleman

Chapter 1

Preface

In the summer of 2020 during the lockdowns my sister and I, with our excessive free time went on a nostalgia trip and watched countless hours of *Scooby-Doo*. Our parents practically raised us on *Scooby-Doo* with DVD and VHS episodes and movies from Blockbusters ranging from episodes of the first series: '*Scooby-Doo, Where Are You?'* in the 60s, to the Warner Home Videos of the early 00s up to '*Scooby-Doo Mystery Incorporated'* from Cartoon Network.

During this time, I was also honing my data science skills in teaching myself python, R, and SQL. Not having a teacher and very little former computer science I was having issues with learning. Something particularly, I was having trouble with was finding good datasets to do analysis on. Searching online for a datasheet that was interesting, large, unstudied, many different types of variables (integers, factors, characters, Boolean, datetime, and numeric) was very difficult.

Then the idea popped in my head that I should create my own spreadsheet. Combined with the amount of *Scooby-Doo* I was watching it seemed perfect to create a spreadsheet on every *Scooby-Doo* episode. This was foolish in retrospect as I was not aware of how many installments of *Scooby-Doo* have been released over the past 50 years. I thought it would take me at most a month but has ended up taking me almost a year. Although due to being a fulltime student and being in the worst (lowest average IMDB score) *Scooby-Doo* series I took a long break during the fall semester of school.

In total I have watched just under 10 days' worth of *Scooby-Doo*, this includes every TV series, direct to DVD movie, cinematic movie and mashup cartoon starring *Scooby-Doo*, and crossover episodes in other TV shows that include *Scooby-Doo*. The amount of time I have spent storing the data and analyzing it has amounted to approximately 3 days (72 hours). The process of storing this data has been an experience that I will not soon forget. I made many rookie mistakes that I regret. However, the experience of making these mistakes has overall been good for my amateur skills and not all of them were uncorrectable. I will go further into detail on these mistakes in the next chapter.

Some of the things I considered a mistake have been needing to add or remove variables, change the way a variable was stored or formatted, or change the variable type. The number of times I have done this is easily in the 100s. This of course got harder and harder the deeper I was into the spreadsheet. It is easy when only a dozen rows were recorded / episodes watched to add a new variable or make a large-scale change. But when the project grew more and more making changes like that would be much harder. In the end the spreadsheet is 603 rows by 74 columns.

Purpose:

As previously stated, I, and possibly many Americans have a love of Scooby Doo. So, my question is: "What are the attributes that make a Scooby-Doo a good TV show or movie a good *Scooby-Doo*?" The variable I will use to track *good* is the public rating on the IMDB website. I will fit models on this variable with the other variables to determine what makes the episode or movie good.

Chapter 2

Data Collection and Variables

The method of data collection was clearly observational as it was nothing but me watching *Scooby-Doo* and jotting down all the variables I was tracking. I used *Google Sheets* to accomplish this task then downloading the spreadsheet as a csv file every time I completed a new 30 rows. After this I would perform basic analysis through graphs and correlation tables and basic summary functions. I would repeat this process until every *Scooby-Doo* iteration was accounted for, allowing me to analysis with complete data.

As mentioned previously there were many mistakes made while making this spreadsheet. I would like to go into depth on the most important ones I could think of and how I corrected each of them as well as those I could not fix and how it affected my study.

However, I believe that it would be better to have a brief overview on the variables that are stored in the spreadsheet to understand what they mean. I will display the variables, describe them, and discuss about issues in the following format:

Example Variable:

Column Name: Variable Type [column]

Index

index: Integer [0]

The index is the manual order I gave to the Scooby-Doo episodes with *1* being the first to air (in the United States) and *603* being the last to air (in the United States). There are however 2 issues. The first being that in

general ‘episode order’ and ‘date aired’ are sometimes not the same. The last 2 series of *Scooby-Doo: Be Cool, Scooby-Doo!* and *Scooby-Doo and Guess Who?* both exclusively online. There was somewhat of a strange schedule for both these series as they would air certain episodes regardless of the ‘official episode order’ which convoluted the index ordering. I took a simple solution to this by preferring official episode order over date aired. Because of this the date aired column is not perfectly constantly ascending and somewhat jumps around a bit near the end. The second issue with the index is that of disputed episode order in the very old *Scooby-Doo* episodes. Some early (70s/80s) shot 6-minute segmented episodes have debated airdates and debated orders (they had very little engagement). *IMDB* and *The Scoobypedia* have separate air dates and order. I again took the simple solution of just basing the date and order entirely on *The Scoobypedia*.

Series Name

series.name: Factor [1]

This is the name of the Series of the *Scooby-Doo* iteration. There were in total 29 unique values for this variable. You might be thinking “were there really 29 *Scooby-Doo* iterations?” The answer is “No”. Many of these include cross-over episodes which usually only have one or a few values in the column. Next, I ‘invented’ series names for some movies in order to better categorize them. None of the movies are particularly part of a series but some of them are easily grouped together. I would apply the network of the movies as series names in order to better group them for analysis which is better than just a NULL value.

Network

network: Factor [2]

In total there are 11 networks that have ran *Scooby-Doo* again however like ‘Series Name’ there sometimes is not a network for movies. In cases of this ‘network/producer’ would be a better fit as if the movie (or episode) does

not have a network I would instead resort to the movie/episode production company.

Season

season: Factor [3]

The longest running *Scooby-Doo* series is 4 seasons long and the series season number is tracked in this column. However there also are 3 other factors. The first being 'Movie', which is simply used when the row is a move. The next is 'Crossover' which is used when the row takes place in a non-*Scooby-Doo* series. This is best because appear on season 11 of *Supernatural* is likely not relevant and it could skew the data. Finally, the 'Special' value is for the few short specials which are episode length and fully *Scooby-Doo*.

Title

title: Character [4]

Goes without saying: title of the Scooby episode, movie, special, whatever. Interestingly there is one duplicate title name: *Wrestle Maniacs* which is the title for an episode from season 3 of *A Pup Named Scooby-Doo* from 1990 and an episode from season 3 of *What's New Scooby-Doo?* from 2005.

IMDB Score

imdb: Numeric [5]

This variable is from the website imdb of each episode, movie, etc. This will be one of the response variables I will use to test what makes a good *Scooby-Doo* iteration. For the latest season of *Scooby-Doo* I decided to record all values as NULL as there are not enough votes to make a reasonable conclusion.

IMDB Engagement

engagement: Integer [6]

Again from imdb this is the second part of the scoring variable on their website. It is a simple integer describing how many people voted to rank the episode, movie etc. Like the IMDB Score the last season is NULL.

Date Aired

date.aired: Date [7]

This is simply the date the episode, movie etc. aired in America according to Scoobypedia and not imdb.

Runtime

run.time: Integer [8]

Runtime is the value of how long the episode, movie, etc. is according to the play bar rounded to the nearest integer. imdb, Scoobypedia, and the runtime according to the player often would have different times and in cases of this I would choose the value of the video player and round it. In the case of 'TV Series (segmented)' in which there are 2-3 short sub-episodes contained in a usually 22-minute episode. It makes little sense to contain the whole episode as one row so I would divide it into 2 or 3 rows and likewise divide the total runtime by the number of segments and round usually giving me 6 or 11-minute segments.

Format

format: Integer [9]

There are 5 potential values for the format column decided by me. 'TV Series' and 'Movie' are the first 2 and pretty self-explanatory. Next up is Crossover which as previously discussed is simply when *Scooby-Doo* characters significantly appear in another Series' episode. TV Series (segmented) is almost entirely shows containing *Scrappy-Doo* and the series: *Laff-a-Lympics*. Finally, there is Movie (Theatrical) which were movies that made it to theater. This includes the 2 live action movies from

the 00s and the latest movie from 2020 *SCOOB!* (I know it did not make it to theaters due to covid but it was planned to so I would say that it fits)

Monster Name

monster.name: Character [10]

The 'Monster Name' is somewhat self-explanatory. It is the name in which the characters refer to the monster they are chasing. It should be noted that the meaning of 'monster' has been stretched in order to fit a wider range of villains. Some (of the more untraditional) Scooby series have a reoccurring mad doctor as the villain. He is not disguised as a monster nor are his ninja minions. However, he fits the same role as the monster in the rest of most of the series, so I feel he fits best here. In the many cases in where there are more than 1 monster they are separated with a comma without space. For example, in the case of one monster it may look like:

Black Knight

While multiple monsters would look like:

Werewolf,Dracula,Frankenstein's Monster

One last thing that should be mentioned about how I counted monsters was swarm monsters. In cases of swarm monsters i.e., dozens of zombies I did not count how many unique zombies there were. Instead, I would count a large indistinguishable swarm as one monster and a distinguished zombie like the leader as one monster as well. So, if a swarm of zombies were led by a leader it would be two monsters on the list.

Monster Gender

monster.gender: Character [11]

The gender of the monster is determined by how The Gang refers to it. She/Her: Female, He/Him: Male. This seems best as it gets around the whole debate if a feminine robot is no-gender or female. It is simple and gets the

job done. Like the past formatting style in cases of multiple monsters like the name variable genders are separated with a comma without space.

Monster Type

monster.type: Character [12]

Monster type is a column in which I tried to narrow the monsters down into a few selected types. The types I have narrowed it down to are:

Ghost, Possessed Object, Ancient, Animal, Undead, Magician, Extraterrestrial, Mechanical, Disguised, Mythical, Super-Villain, Plant

The formatting is same as both 'monster gender' and 'monster name'.

Monster Subtype

monster.subtype: Character [13]

Monster subtype is a much broader term than that of plain 'type'. This is a very flexible category in which anything goes. If the gang is against a ghost pirate this category will by ghost, if they are against a zombie this will be zombie. If the gang is against a pirate zombie, I had no rule and would pick one or the other. Because of this I will not be using this column likely.

Monster Species

monster.species: Character [14]

Like the subtype this variable is very flexible. Its goal is to say the species of the monster, it is usually human. In cases where the monster is not and wasn't a living being like a robot or possessed suit of armor I would say 'suit' or 'armor'. Same formatting.

Monster Real

monster.real: Boolean [15]

True or false value of whether the monster turns out to be real or not. Real as in the monster is not a man in a mask, mechanically controlled or

trained/hypnotized by a human. In cases with multiple monsters and potentially real and false monsters I make judgement on which is the 'main' monster and decide this value based on that. With this variable along with the past 5 in the many cases where no monster exists this along with them are labeled as NULL.

Monster Amount

monster.amount: Integer [16]

The number of monsters in the mystery. It should be noted that the only monsters that are counted are those who are part of the main story. If there is a movie and it decides to open at the end of catching a monster scene, I do not count that monster anywhere in the spreadsheet. This applies to the past 6 variables. As a rule, this number should be the count of commas in any of the past 6 variables + 1.

Caught

caught."member": Boolean [17:22]

These variables are Boolean values marking who the monster was caught by. For example, if Fred catches the monster caught.fred will be TRUE while the rest would be FALSE. But if Fred and Shaggy catch the monster caught.fred and caught.shaggy will be TRUE and caught.velma, caught.daphnie, and caught.scooby will be FALSE. In cases where certain members are not in the episode for whatever reason the values will be NULL instead of FALSE, this applies to any gang-member specific future variable.

Captured

capture."member": Boolean [22:27]

These variables are the number of times a member of the gang is captured by the monster. This can include being locked in a room, tied to a wall or

falling down a trapdoor. This only counts if they dedicate time to showing the character escape. It follows the same Boolean rules as the Caught variables including the NULL rule.

Unmasked

unmask."member": Boolean [27:32]

These variables are the Boolean values of when a member unmasks the monster. Like all the monster themed variables we are only talking about monsters that are part of the main story. Also, they do not have to do the traditional sort of unmasking, but if you thaw the Ice-Monster it counts as an unmask.

Snack

snack."member": Boolean [32:37]

This is when a member of the gang offers someone a *Scooby Snack* (usually given to Shaggy and Scooby). The person they offer does not have to eat it for this to count as TRUE. Same rules with other Boolean Gang values.

Unmask Other

unmask.other: Boolean [37]

This follows all the rules of the other unmask variables but only applies when a non-gang member unmasks the monster.

Caught Other

caught.other: Boolean [38]

Like unmask.other except for catching the monster. Follows all the same rules of the other caught variables and applies when a non-gang member unmasks the monster.

Not Caught

caught.not: Boolean [39]

TRUE when the monster is not caught. Not being caught means that the monster manages to get away without the gang trapping them. There are few cases in which this happens.

Trap Works First Time

trap.work.first: Boolean [40]

True when the monster is caught as intended by the trap. This value is often NULL and is NULL when there is no trap in the installment. Again, does not apply when the trap is on a monster that is not part of the main story.

Setting: Terrain

setting.terrain: Character [41]

The terrain setting that takes place in the plurality of the iteration. I have created 15 different potential settings and used judgement in cases where it was debatable. The 15 are:

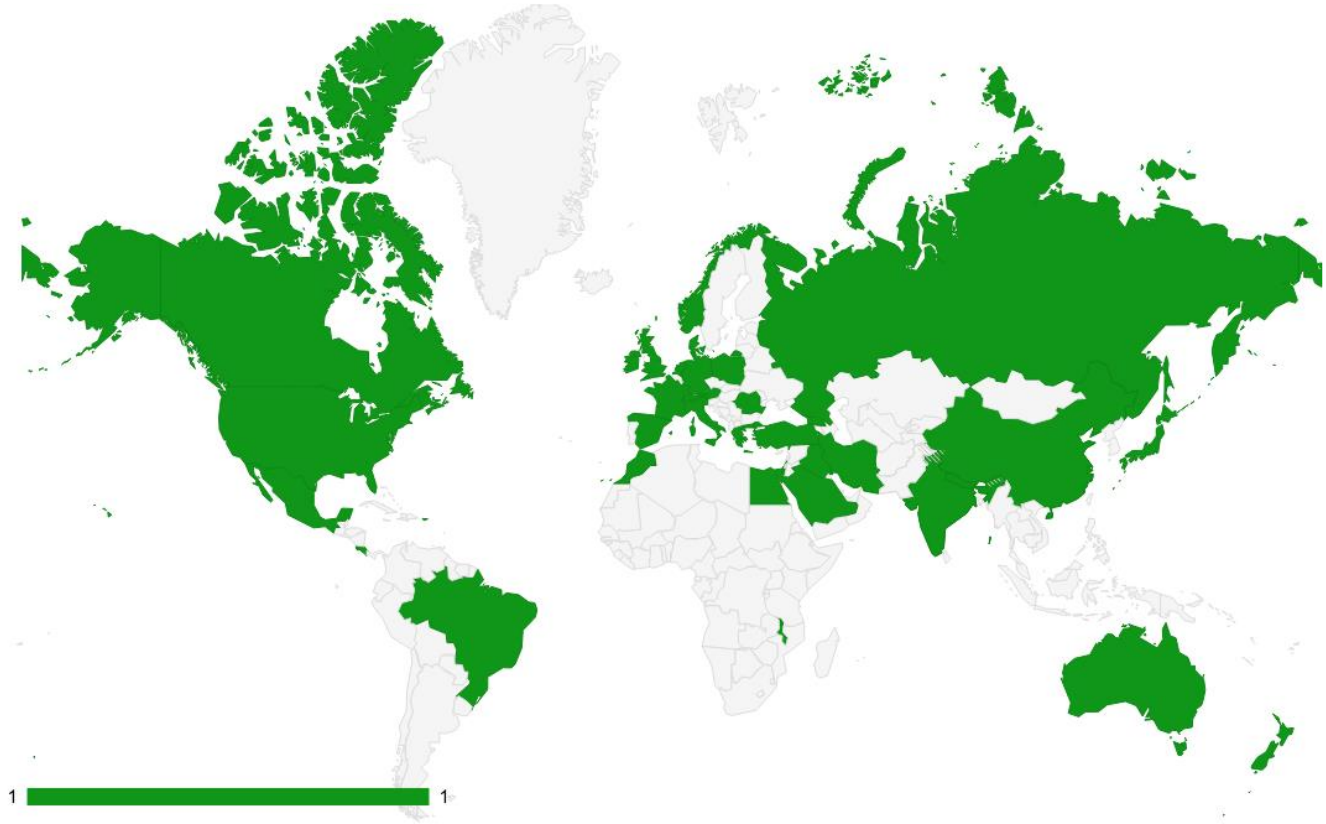
Urban, Rural, Swamp, Forest, Coast, Island, Cave, Desert, Ocean, Snow, Jungle, Mountain, Moon, Space, Air

Some of these descriptors can often be overlapping, for example if the location takes place on a snow mountain which would I pick, Snow or Mountain? I would again use judgement and decide if the iteration focuses more on the mountain or on the snow. If the gang is stuck inside a hotel on a mountain because it is snowing, I would choose 'Snow'. However, if they are on a snowy mountain for the purpose of finding a treasure at the peak I would choose 'Mountain'.

Setting: Country/State

setting.country/state: Character [42]

This column tracks which state or country the plurality of the iteration takes place. In the many cases in which the mention of which state or country they are in it is assumed that they are in the United States. In cases the country is not mentioned, however the continent is I would write down the continent. In cases ocean travel I would simply write ocean unless they are in a specific area in the ocean, like 'Bermuda Triangle'. In cases of time travel I would write which era of time they are in instead of country or state, for example: 'Prehistoric'. In total there are 79 separate countries, states, etc. they have visited. See *plot 1* for unique country visits.



Plot 1

Suspect Amount

suspect.amount: Integer [43]

The number of non-recurring characters that have a ‘decent’ amount of screen time. This rule is broken if there is an obvious attempt to make a non-recurring character have suspicion behind them in which case they are counted. Also includes suspects in backstories.

Non-Suspect

non-suspect: Boolean [44]

True or false value stating if the criminal was contained in the set of suspects from the ‘suspect amount’ variable. In rare cases where the culprit was disguising himself as a suspect, but the gang never realizes that the suspect who he says he is until the end it will count as false as I believe the main purpose of this variable was to show a gradual progression into having a concrete roster of suspects. Something the first series did not have down. Also, in cases there are no culprits the value is simply null.

Arrested

arrested: Boolean [45]

True or false value stating if the culprit is arrested. If there is no showing of the culprit being arrested however there is showing of him being caught by the gang it can be assumed he is arrested. This value is assumed True unless we see the culprit managing to get away from the gang, the gang letting him free, etc. In cases of multiple suspects, I make judgement on who the ‘main’ culprit is and set the values based of them.

Culprit Name

culprit.name: Character [46]

The name of the culprits separated by a comma. Number of names should be equal to the *culprit.amount* variable. NULL when no culprits.

Culprit Gender

culprit.gender: Character [47]

Gender of the culprits is ordered same as the *culprit.name* variable. Is separated by comma and contains either male or female. Number of genders should be equal to the *culprit.amount* variable. NULL when no culprits.

Culprit Amount

culprit.amount: Integer [48]

Integer value of the number of culprits. Along with *culprit.gender* and *cuprit.name* this variable only 'important' culprits. For example, a culprit who has no lines and is not given a name will not be included.

Motive

motive: Character [49]

The general motivation behind the monster attacks. In total I ended up with 27 motives including: Theft, Treasure, Natural Resources, Competition, Extortion, Safety, Counterfeit, Inheritance, Smuggling, Preservation, Experimentation, Food, Trespassing, Assistance, Abduction, Haunt, Anger, Imagination, Bully, Loneliness, Training, Conquer, Mistake, Automated, Production, Entertainment, Simulation. It should be noted that many of these have very low counts, sometimes only one and the strange ones take place in non-traditional series usually. In general, the 5 most popular motives are:

- Theft
- Treasure
- Natural Resources
- Competition
- Conquer

It should be noted that *monster.real* could be True and the motive is not NULL. This is because this variable too tracks the motive of the real monster if there happens to be no one behind the mask.

‘If it wasn’t for you meddling kids and that dog’

if.it.wasnt.for, and.that: Character [50:52]

A fill in the blank variable of the catchphrase “and I would have gotten away with it too, if it wasn’t for you meddling kids and that dog”. Many times, they switch up the iconic line by replacing the ‘meddling kids’ and ‘dog’ with other words. Sometimes the second part of the line is omitted, and the phrases is just ‘...if it wasn’t for you meddling kids’. In this case the variable is simply NULL. In cases of the phrase never being said both variables are NULL.

Door Gag

door.gag: Boolean [52]

A simple Boolean value of weather or not the iconic down the hall shot of the monster and the gang running in and out of doors is in the episode or movie.

Number of Snacks

number.of.snacks: Factor [53]

The number of snacks that are given. This is a factor because in many cases the number is not specific enough, so I result to some factor definitions. First if the number of snacks is countable, I simply set the variable as the number. Also, if the number of snacks is given in boxes I write ‘*x* box(es)’. In the remaining cases where the exact number is unknown the value of a small amount is *a couple, several, wheel barrel full, truck load, lifetime supply*.

Catchphrases

catchphrases: Integer [53:64]

A count of the number of times each of the character’s iconic catchphrases are said. For Fred I tracked:

- ‘Let’s split up and look for clues’

- ‘Looks like we got another mystery on our hands’
- ‘It’s time to set a trap’

For Daphne I tracked:

- ‘Jeepers!’

For Velma I tracked:

- ‘Jinkies!’
- ‘My glasses!’
- ‘Just about wrapped up this mystery’

For Shaggy I tracked:

- ‘Zoinks!’
- ‘Groovy’
- ‘Scooby-Doo where are you!’

For Scooby-Doo I tracked:

- ‘Rooby-rooby-roo!’

Some of these catchphrases I judged more liberally than others. For example, if Fred says ‘It’s trapping time’ I will still count it as a count. However, if Shaggy says ‘Doinks’ I would not count it. In general, I am more lenient with phrases and less so with exact words. The values are NULL in the cases said character does not appear in the episode or movie.

Recurring Guests

recurring guests: Boolean [64:69]

A True or False value for characters that show up throughout multiple series. This includes Batman, Scooby-Dum, Scrappy-Doo, Hex Girls, and The Blue Falcon. Containing means that the character(s) were present and had a role, not simply mentioned.

Voice Actors

character.va: Character [69:74]

The voice actor, or actor in cases of live action movies of each iteration. These variables are the source of these from the credits of each episode or movie.

Mistakes and Regrets

There are likely many mistakes in this dataset. I of course tried my hardest and corrected many after all the data was collected. Many of these were simple typos or missing values. However, there are likely many I missed. An issue that I likely had is values that relied on count. The main issue with these is that the error is not normally distributed as the way I collected data there is a much higher chance I missed a value instead of an overcounted one. I do not believe that this will be much of an issue however as many of the integer values likely will not be correlated at all with the IMDB score like catchphrases and the likes.

I have many regrets that I wish I could go back and tell former me to include. Most of these were me watching 200 episodes in and realizing something like "number of clues found" for each member of the gang would have been a great variable or realizing Scooby had the catchphrase "ruh-roh" I did not track. However, I do not regret it enough to go back and watch 10 days' worth of *Scooby* again.

Chapter 3

Model Plan

I plan to create the model in the form of a classic linear + ANOVA model. This will be to capture both categorical and numeric data as I predict both will be important to creating our model. I feel a linear model will be best as the response (*imdb score*) did not come from a count or time waiting, so a

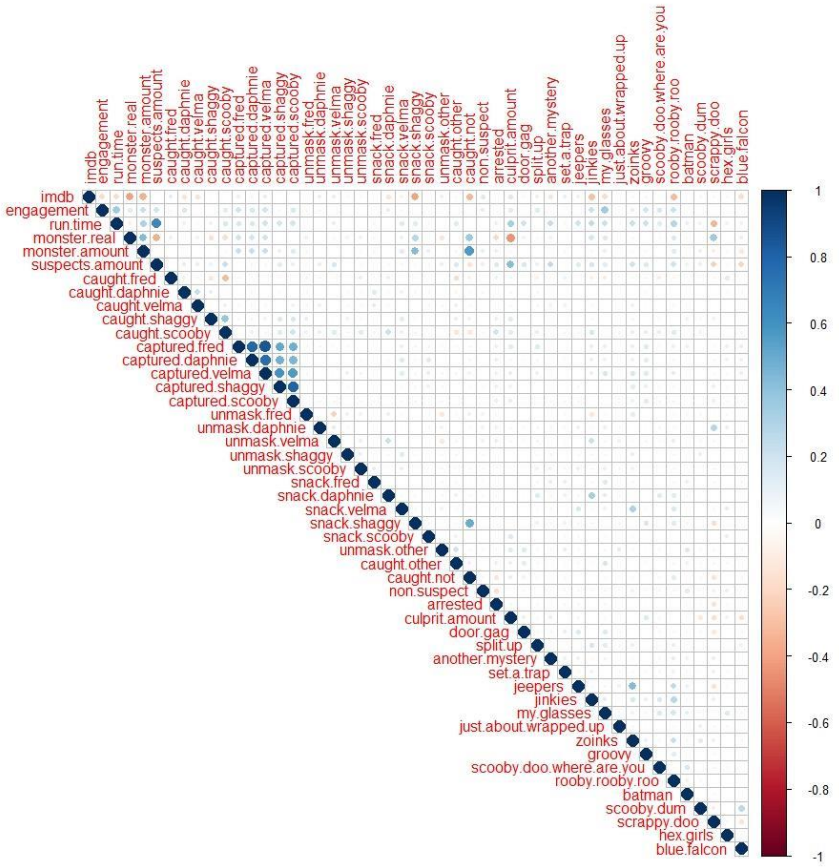
generalized linear model will not be of much use. Next, we do not want to use a non-parametric model as stated in our preface, but we will transform the explanatory variables if the cases arise where we need to. We want to *see* what makes a good Scooby-Doo and non-parametric models are rather hard to interpret. Now to build our linear model we will follow the steps we learned in class. First, we will pluck the values that show somewhat high correlation with our response variable, *imdb*. We will visualize this with a *corrplot*. For categorical data we will instead create a lone ANOVA model for each categorical independent variable to see which have strong predicting ability for *imdb*. An issue I predict will stem from the fact that many of the categorical variables are multi-level and can have up to double digit levels. This is because there are some strange "one-offs" in our factors which do not have a significant amount of data. I predict I may have to scrap some of these one-offs if they end up over-complicating the data. Once I have my ideal predictor variables, I will look for collinearity. I will use *kappa* to crack down on this by removing predictors and try to get a reasonable low *kappa* level. I will shoot for a somewhat low *kappa* level but won't expect to get below 30 as much of this data is co-linear. Finally, I will use the BIC scoring method on the remaining combination of variables to find the best model. BIC will be used because in general it is better for explanation. (My general *p-val* will be 0.05 for all p-values)

Data Exploration

Numeric

The first step to data analysis is to explore your data. Let us look at relationships across variables and make judgement on where to go from there. For starters let us draw a cross correlation function and note any higher correlated values. This step is important because in order to find out what makes a good Scooby-Doo we need to find what correlates with our IMDB variable. We do not want this model to have 74 predictors. First let us

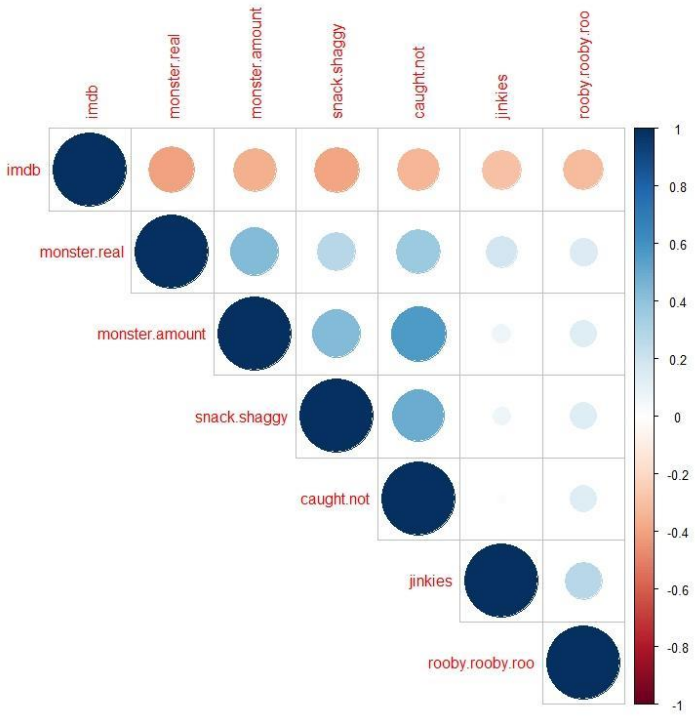
create a *corrplot* of all the integer and numeric values for this I will be using the *pairwise.complete.obs* method for my correlations. This method omits any values that are NULL when calculating the correlation. I definitely need this as NULL was a value I used a lot throughout creating the data frame. You can see the full correlation values in *plot 2*, I will now filter down to only values that have somewhat higher correlation with *imdb*.



Plot 2

The much cleaner matrix below in *plot 3* only shows values that have a .25 or higher or -.25 or lower correlation with *imdb*. These values are:

Monster Real, Monster Amount, Snack Shaggy, Caught Not, Jinkies, Rooby-Rooby-Roo



Plot 3

From a rudimentary look of this correlation matrix, we can say that the scores of shows are lower when there are many monsters, the monsters are real, and the monster is not caught. Another thing that we can see is that there is some obvious co-linearity as the correlation is high between monsters being real, there being many monsters and the monster not being caught. We will likely need to remove a few of these variables once we get to the collinearity section.

For fun let us look at our rudimentary linear model so far in *console 1* bellow:

```
Call:
lm(formula = imdb ~ monster.real + monster.amount + snack.shaggy +
    caught.not + jinkies + rooby.rooby.roo, data = scooby)

Residuals:
     Min       1Q   Median       3Q      Max
-2.14359 -0.33668 -0.01448  0.36332  1.92688

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)      7.92407    0.05115 154.920  < 2e-16 ***
monster.realTRUE  0.05363    0.14726   0.364 0.715948
monster.amount   -0.08739    0.02041  -4.281 2.39e-05 ***
snack.shaggyTRUE -0.34243    0.13693  -2.501 0.012839 *
caught.notTRUE    0.49028    0.20454   2.397 0.017041 *
jinkies          -0.06490    0.01680  -3.863 0.000133 ***
rooby.rooby.roo  -0.22220    0.03660  -6.071 3.24e-09 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.59 on 358 degrees of freedom
  (256 observations deleted due to missingness)
Multiple R-squared:  0.2493,    Adjusted R-squared:  0.2367
F-statistic: 19.81 on 6 and 358 DF,  p-value: < 2.2e-16
```

Console 1

It looks okay so far. The monster not real might be one of the first we cut off. The R^2 of 0.2367 is low, let us hope the categorical variables help increase this value. Also, we will look at the collinearity through the vif() function to see if there is any co-linearity in *console 2*:

```
> vif(lmod)
   monster.real  monster.amount    snack.shaggy      caught.not         jinkies rooby.rooby.roo
       1.232976        1.204111        1.066076        1.054978        1.104530        1.116121
```

Console 2

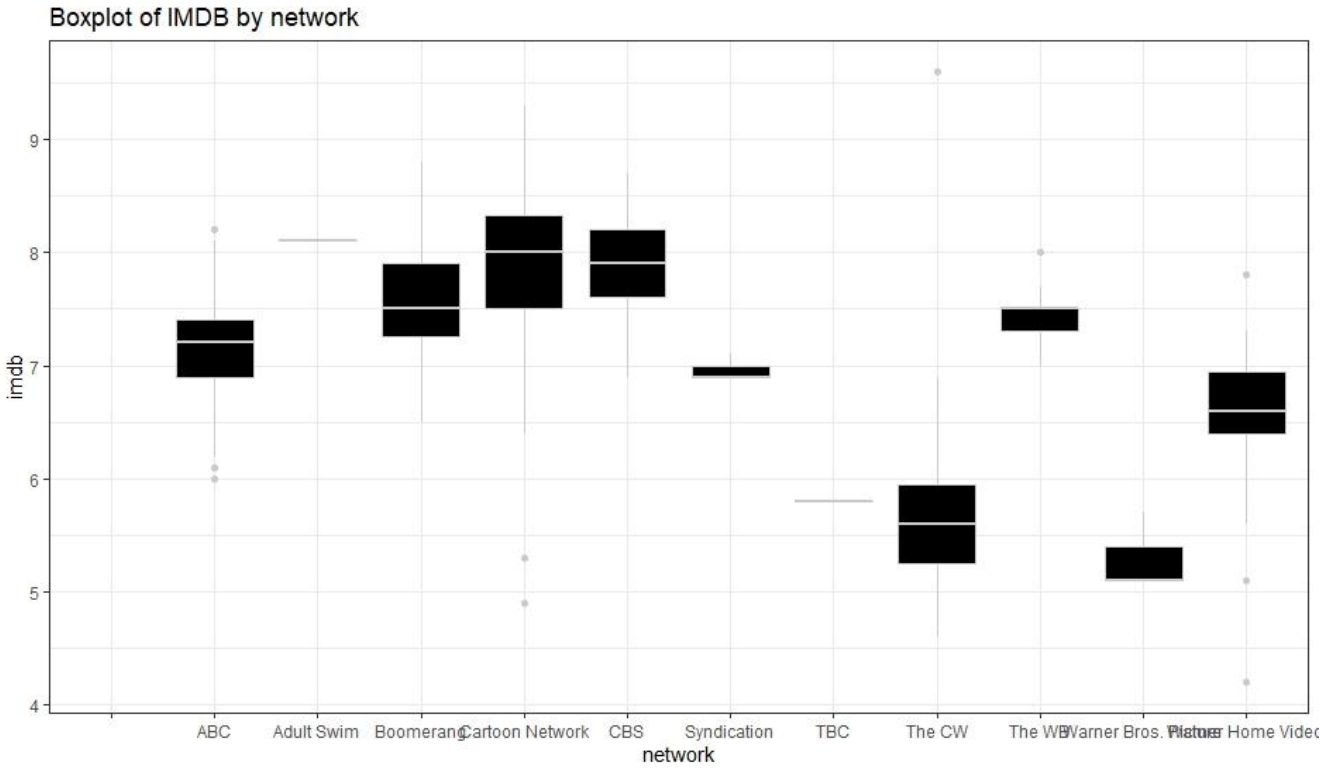
This is good. The collinearity we were worrying about earlier seems to be low enough below 5.

Categorical

Let us visualize how the categorical variables affect *imdb* score. We will make many graphs and if we see obvious differences, we will test an ANOVA model and decide where to go from there.

Network

Below is a boxplot of imdb by network in *plot 4*.



Plot 4

We can see Adult Swim and TBC network have only had one airing of *Scooby-Doo*. However, we see clear separations among the rest of the networks. We may need to cut Adult Swim and TBC as they might make the factor unusable. Let us see if we need to in *console 3* where we make our ANOVA model.

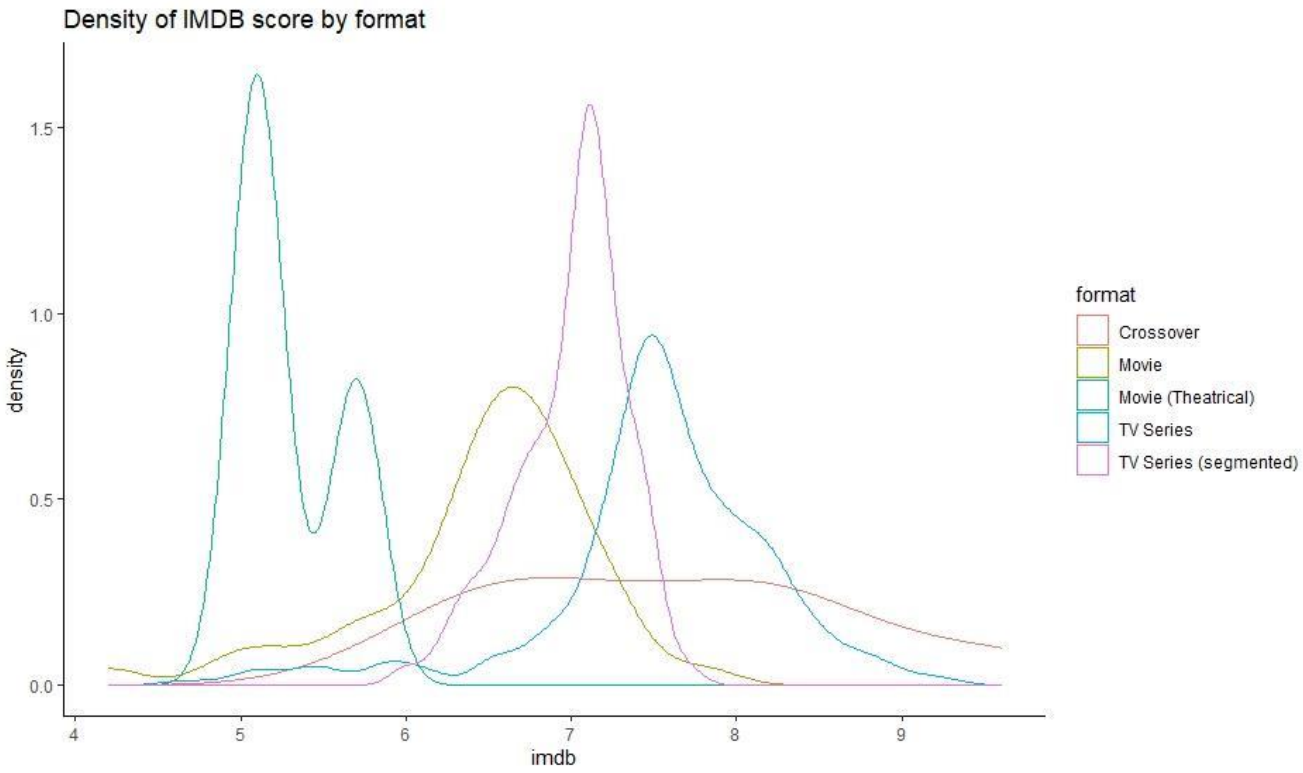
```
Call:
lm(formula = imdb ~ network, data = scooby)

Residuals:
    Min      1Q  Median      3Q     Max
-3.0119 -0.2662  0.0271  0.2881  3.8481

Coefficients:
                            Estimate Std. Error t value Pr(>|t|)
(Intercept)                  7.16619    0.03098 231.332  < 2e-16 ***
networkAdult Swim            0.93381    0.52021   1.795  0.07317 .
networkBoomerang             0.40669    0.07436   5.469 6.75e-08 ***
networkCartoon Network       0.74571    0.06457  11.548  < 2e-16 ***
networkCBS                   0.73585    0.08039   9.153  < 2e-16 ***
networkSyndication          -0.19953    0.30140  -0.662  0.50825
networkTBC                  -1.36619    0.52021  -2.626  0.00886 **
networkThe CW               -1.41434    0.10463 -13.518  < 2e-16 ***
networkThe WB                0.26308    0.08681   3.030  0.00255 **
networkWarner Bros. Picture -1.86619    0.30140  -6.192 1.13e-09 ***
networkWarner Home Video    -0.58927    0.08873  -6.641 7.22e-11 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.5193 on 577 degrees of freedom
  (33 observations deleted due to missingness)
Multiple R-squared:  0.506,     Adjusted R-squared:  0.4974
F-statistic:  59.1 on 10 and 577 DF,  p-value: < 2.2e-16
```

Console 3

This seems like a very strong categorical variable. Many of these levels are well significant while only 2 manage to fall above our 0.05 p-value. Syndication, one of the insignificant predictors had only produced 3 movies and Adult Swim had only 1. I do not feel we will lose much by getting rid of those 2 so we may filter them out in our final datasheet we plan to test on. We also might want to remove TBC as it too has only 1 value. The R^2 is 0.497 which seems great as well.

Format

For format we will create a density plot. This factor has 5 levels. See below *plot 5*.



Plot 5

This looks good, our data seems to be somewhat normally distributed, and the format truly seems to shift the means and variances. We can see that crossover episodes seem to have the widest range of imdb scores, and theatrical movies has the least normal distribution because there are only 3 values. Meanwhile TV Series preforms better than TV Series (segmented)

which preforms better than regular movies. Let us produce an ANOVA model in *console 4* below:

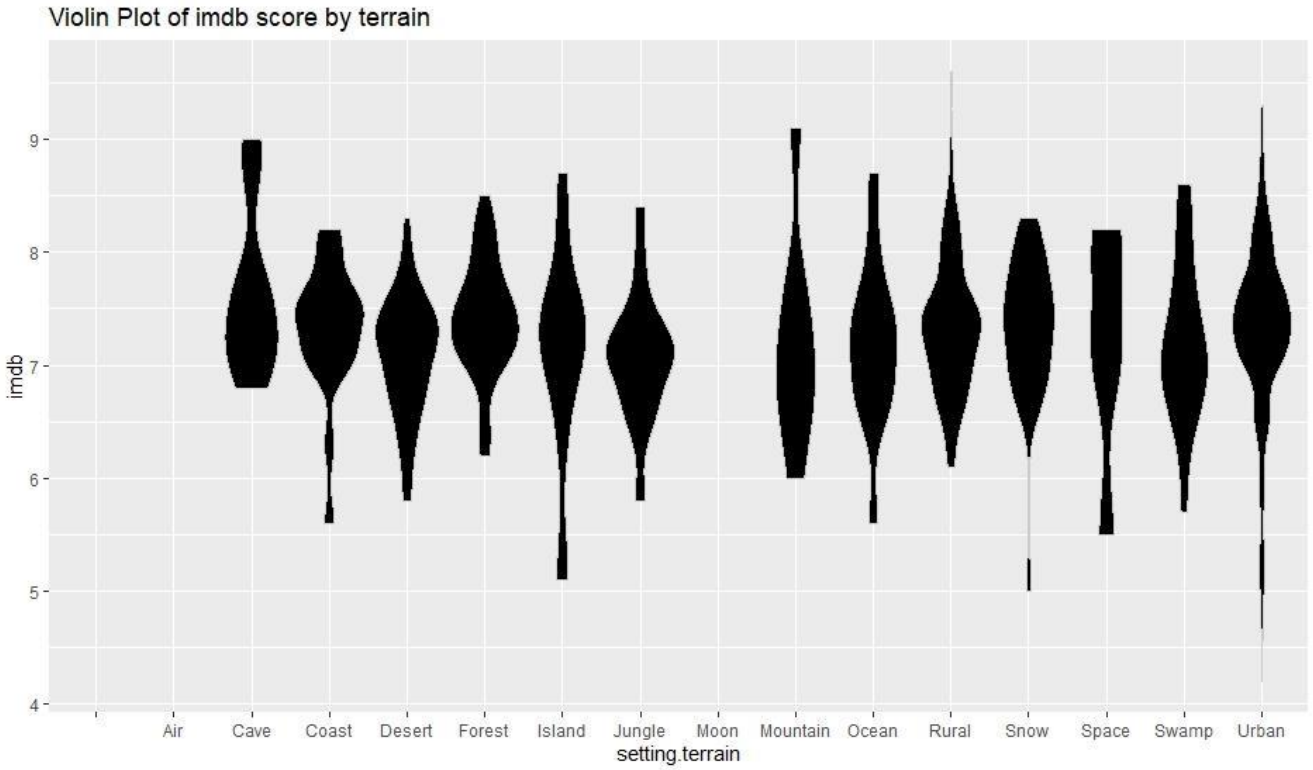
```
Call:
lm(formula = imdb ~ format, data = scooby)

Residuals:
     Min       1Q   Median       3Q      Max
-2.91671 -0.21671  0.08329  0.38329  1.97500

Coefficients:
                          Estimate Std. Error t value Pr(>|t|)
(Intercept)                 7.6250     0.2271  33.578  < 2e-16 ***
formatMovie                -1.1576     0.2473  -4.681 3.56e-06 ***
formatMovie (Theatrical)   -2.3250     0.4348  -5.347 1.29e-07 ***
formatTV Series            -0.1083     0.2296  -0.472  0.63737
formatTV Series (segmented) -0.6199     0.2322  -2.669  0.00781 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.6423 on 583 degrees of freedom
  (33 observations deleted due to missingness)
Multiple R-squared:  0.2363,    Adjusted R-squared:  0.2311
F-statistic: 45.11 on 4 and 583 DF,  p-value: < 2.2e-16
```

Console 4

We see strong correlation, even though some of these have quite a high p-value. The TV-Series is not significant in this however this is more likely due to it being closer to the intercept. There are tons of values (374) in the TV-Series, and it just turns out the TV-Series average was close to the intercept (Crossover). I think the p-value will change later if the intercept changes.

Setting Terrain

Now let us see if the terrain affects the goodness of *Scooby-Doo*. Below is a violin plot in *plot 6*:



Plot 6

This does not look very predictive; all the values look to have the same mean and variance. Let us see if a model shows us if there is somewhat of a correlation bellow in *consol 5:*

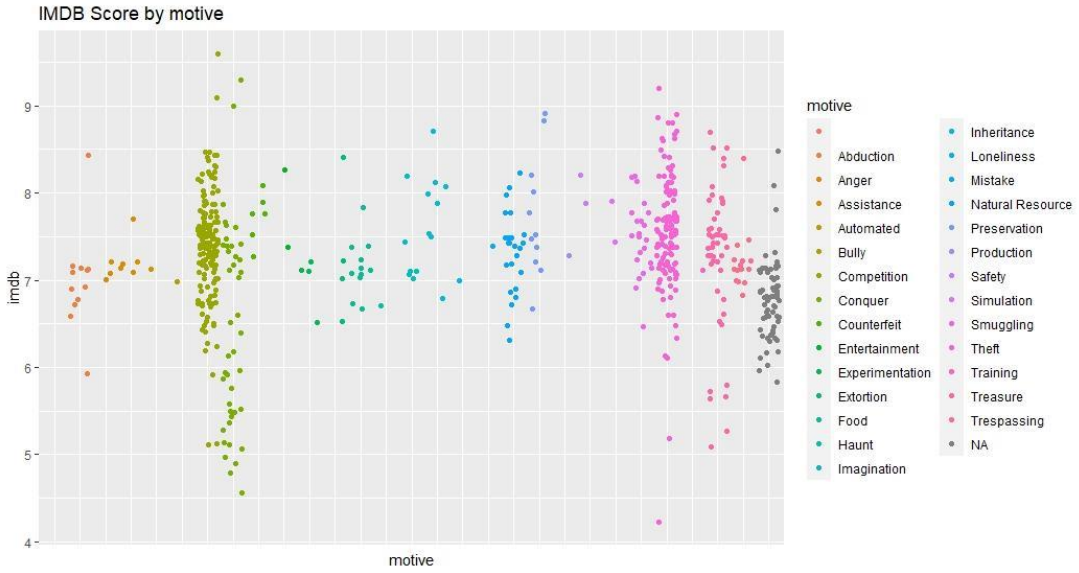
```
Call:
lm(formula = imdb ~ setting.terrain, data = scooby)

Residuals:
    Min      1Q  Median      3Q     Max
-3.0780 -0.3106  0.0500  0.4220  2.2500

Coefficients:
                      Estimate Std. Error t value Pr(>|t|)
(Intercept)             7.8000     0.7341  10.626   <2e-16 ***
setting.terrainCave    -0.1556     0.7738  -0.201    0.841
setting.terrainCoast   -0.5000     0.7553  -0.662    0.508
setting.terrainDesert  -0.7128     0.7434  -0.959    0.338
setting.terrainForest  -0.3894     0.7418  -0.525    0.600
setting.terrainIsland  -0.6960     0.7486  -0.930    0.353
setting.terrainJungle  -0.7200     0.7581  -0.950    0.343
setting.terrainMoon    -1.0000     1.0381  -0.963    0.336
setting.terrainMountain -0.6000    0.7786  -0.771    0.441
setting.terrainOcean   -0.6154     0.7618  -0.808    0.420
setting.terrainRural   -0.4500     0.7375  -0.610    0.542
setting.terrainSnow    -0.4875     0.7492  -0.651    0.516
setting.terrainSpace   -0.6200     0.8041  -0.771    0.441
setting.terrainSwamp   -0.5947     0.7531  -0.790    0.430
setting.terrainUrban   -0.5220     0.7355  -0.710    0.478
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.7341 on 573 degrees of freedom
  (33 observations deleted due to missingness)
Multiple R-squared:  0.01968,   Adjusted R-squared:  -0.004272
F-statistic: 0.8216 on 14 and 573 DF,  p-value: 0.6458
```

Console 5

As expected, none of these values are significant. Just looking at the R^2 of 0.0196 is pathetically small. We will not be using terrain in our model.

Motive

Let us see if the motive happens to affect the score, maybe monsters hoping for a good inheritance receive higher ratings than those drilling for oil. Bellow on *plot 7* is a swarm-plot of the imdb score by motivation:



Plot 7

If I were to guess there might be some significant values let us see bellow in *figure 6:*

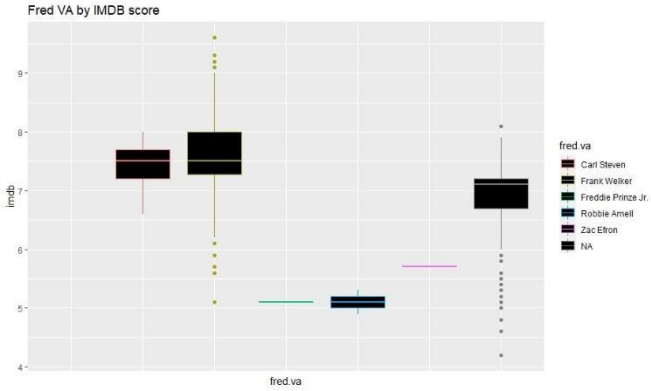
```
Call:
lm(formula = imdb ~ motive, data = scooby)

Residuals:
    Min      1Q  Median      3Q     Max
-3.3590 -0.3073  0.0000  0.3136  2.8595

Coefficients:
                          Estimate Std. Error t value Pr(>|t|)
(Intercept)               6.983333   0.197754  35.313  < 2e-16 ***
motiveAnger               0.116667   0.442191   0.264  0.79201
motiveAssistance          0.276667   0.364640   0.759  0.44837
motiveAutomated           0.116667   0.713011   0.164  0.87009
motiveBully               0.016667   0.713011   0.023  0.98136
motiveCompetition         0.423939   0.204819   2.070  0.03899 *
motiveConquer            -0.542857   0.224232  -2.421  0.01584 *
motiveCounterfeit         0.750000   0.342520   2.190  0.02902 *
motiveEntertainment       0.866667   0.523207   1.656  0.09827 .
motiveExperimentation    -0.008333   0.395508  -0.021  0.98320
motiveExtortion           0.291667   0.395508   0.737  0.46120
motiveFood                0.162121   0.285951   0.567  0.57100
motiveHaunt              -0.283333   0.713011  -0.397  0.69126
motiveImagination         0.333333   0.342520   0.973  0.33094
motiveInheritance         0.841667   0.312676   2.692  0.00735 **
motiveLoneliness          0.016667   0.713011   0.023  0.98136
motiveMistake             0.416667   0.713011   0.584  0.55923
motiveNatural Resource    0.341667   0.242198   1.411  0.15896
motivePreservation        0.753030   0.285951   2.633  0.00872 **
motiveProduction          0.316667   0.713011   0.444  0.65715
motiveSafety              1.066667   0.523207   2.039  0.04201 *
motiveSimulation          0.666667   0.523207   1.274  0.20319
motiveSmuggling           0.503030   0.245840   2.046  0.04127 *
motiveTheft               0.575683   0.207251   2.778  0.00568 **
motiveTraining            0.116667   0.713011   0.164  0.87009
motiveTreasure            0.297917   0.221095   1.347  0.17845
motiveTrespassing         0.236667   0.265315   0.892  0.37281
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.685 on 494 degrees of freedom
  (100 observations deleted due to missingness)
Multiple R-squared:  0.1808,    Adjusted R-squared:  0.1377
F-statistic: 4.193 on 26 and 494 DF,  p-value: 9.253e-11
```

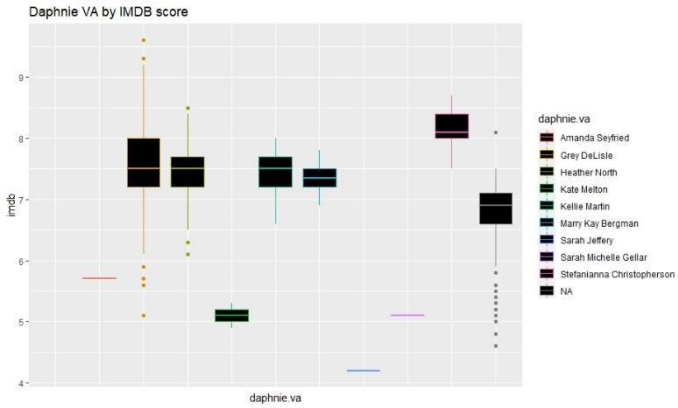
Console 6

This is surprising, the model's p-value is well significant with 9.253e-11. However, I know because there are so many levels here it will shoot up the BIC score. We will add it to our draft model, but I am sure it will be the first one to remove.

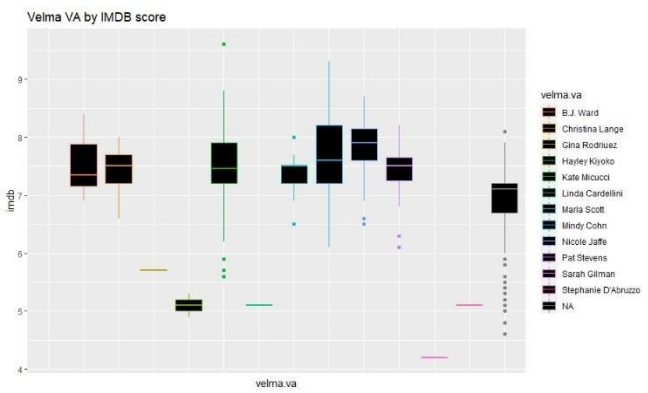
Voice Actors

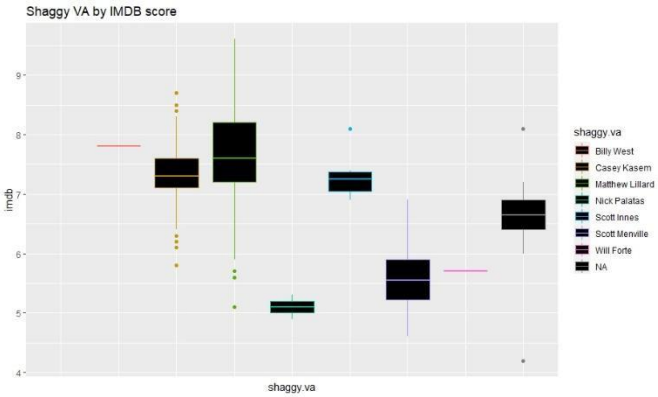
Up next are 5 factor variables. Instead of writing the same thing over and over I will get them all out of the way. Below *Plots 9-13* are all the models of all the gang's voice actors against IMDB score.



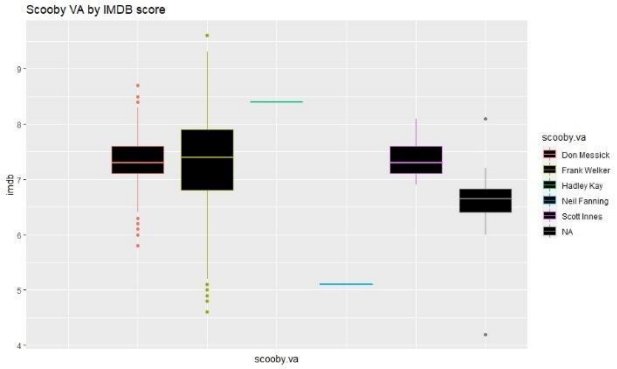


Plot 9 and 10





Plot 11 and 12



Plot 13

There is great difference in the means between the voice actors just from a quick glance of these graphs. Let again use the ANOVA model on these models and see if they are significant. I have a feeling they will. Bellow in *console 7-11*.

```
Call:
lm(formula = imdb ~ fred.va, data = scooby)

Residuals:
    Min      1Q  Median      3Q     Max
-2.4693 -0.2693  0.0000  0.4000  2.0307

Coefficients:
                       Estimate Std. Error t value Pr(>|t|)
(Intercept)              7.4000     0.1178  62.827  < 2e-16 ***
fred.vaFrank Welker      0.1693     0.1226   1.381  0.16802
fred.vaFreddie Prinze Jr.  -2.3000     0.4562  -5.042 7.28e-07 ***
fred.vaRobbie Amell        -2.3000     0.4562  -5.042 7.28e-07 ***
fred.vaZac Efron           -1.7000     0.6343  -2.680  0.00769 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.6233 on 364 degrees of freedom
  (252 observations deleted due to missingness)
Multiple R-squared:  0.1651,    Adjusted R-squared:  0.1559
F-statistic: 17.99 on 4 and 364 DF,  p-value: 1.713e-13
```

```
Call:
lm(formula = imdb ~ daphnie.va, data = scooby)

Residuals:
    Min      1Q   Median      3Q     Max
-2.44186 -0.25526 -0.02353  0.34474  2.05814

Coefficients:
                                Estimate Std. Error t value Pr(>|t|)
(Intercept)                       5.7000     0.5887   9.682  < 2e-16 ***
daphnie.vaGrey DeLisle            1.8419     0.5901   3.121  0.00193 **
daphnie.vaHeather North           1.7553     0.5906   2.972  0.00313 **
daphnie.vaKate Melton            -0.6000     0.7210  -0.832  0.40579
daphnie.vaKellie Martin           1.6724     0.5988   2.793  0.00546 **
daphnie.vaMarry Kay Bergman       1.6500     0.6582   2.507  0.01256 *
daphnie.vaSarah Jeffery          -1.5000     0.8325  -1.802  0.07232 .
daphnie.vaSarah Michelle Gellar  -0.6000     0.7210  -0.832  0.40579
daphnie.vaStefanianna Christopherson   2.4235     0.6058   4.001 7.48e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.5887 on 414 degrees of freedom
  (198 observations deleted due to missingness)
Multiple R-squared:  0.2385,    Adjusted R-squared:  0.2238
F-statistic: 16.21 on 8 and 414 DF,  p-value: < 2.2e-16
```

Console 7 and 8

```
Call:
lm(formula = imdb ~ velma.va, data = scooby)

Residuals:
    Min      1Q  Median      3Q     Max
-1.8359 -0.2490  0.0000  0.3529  2.1641

Coefficients:
                        Estimate Std. Error t value Pr(>|t|)
(Intercept)              7.51250    0.21154  35.513  < 2e-16 ***
velma.vaChristina Lange -0.14009    0.23894  -0.586 0.558060
velma.vaGina Rodriuez   -1.81250    0.63462  -2.856 0.004540 **
velma.vaHayley Kiyoko   -2.41250    0.47302  -5.100 5.51e-07 ***
velma.vaKate Micucci    -0.07663    0.22055  -0.347 0.728451
velma.vaLinda Cardellini -2.41250    0.47302  -5.100 5.51e-07 ***
velma.vaMarla Scott     -0.16806    0.29073  -0.578 0.563601
velma.vaMindy Cohn       0.13456    0.21853   0.616 0.538463
velma.vaNicole Jaffe     0.33652    0.22753   1.479 0.140012
velma.vaPat Stevens     -0.06523    0.22640  -0.288 0.773435
velma.vaSarah Gilman    -3.31250    0.63462  -5.220 3.04e-07 ***
velma.vaStephanie D'Abruzzo -2.41250    0.63462  -3.801 0.000169 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.5983 on 358 degrees of freedom
  (251 observations deleted due to missingness)
Multiple R-squared:  0.2921,    Adjusted R-squared:  0.2704
F-statistic: 13.43 on 11 and 358 DF,  p-value: < 2.2e-16
```

```
Call:
lm(formula = imdb ~ shaggy.va, data = scooby)

Residuals:
     Min       1Q   Median       3Q      Max
-2.50122 -0.30122 -0.00253  0.29878  1.99878

Coefficients:
                       Estimate Std. Error t value Pr(>|t|)
(Intercept)              7.8000     0.5830  13.379  < 2e-16 ***
shaggy.vaCasey Kasem    -0.4736     0.5838  -0.811 0.417588
shaggy.vaMatthew Lillard -0.1988     0.5848  -0.340 0.734035
shaggy.vaNick Palatas   -2.7000     0.7140  -3.781 0.000173 ***
shaggy.vaScott Innes    -0.4833     0.6297  -0.768 0.443075
shaggy.vaScott Menville -2.1962     0.5941  -3.697 0.000240 ***
shaggy.vaWill Forte     -2.1000     0.8245  -2.547 0.011134 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.583 on 549 degrees of freedom
  (65 observations deleted due to missingness)
Multiple R-squared:  0.354,     Adjusted R-squared:  0.347
F-statistic: 50.15 on 6 and 549 DF,  p-value: < 2.2e-16
```

Console 9 and 10

```
Call:
lm(formula = imdb ~ scooby.va, data = scooby)

Residuals:
     Min       1Q   Median       3Q      Max
-2.71074 -0.32045  0.07955  0.37955  2.28926

Coefficients:
                      Estimate Std. Error t value Pr(>|t|)
(Intercept)           7.320455   0.040539 180.579  < 2e-16 ***
scooby.vaFrank Welker -0.009711   0.061115  -0.159    0.874
scooby.vaHadley Kay    1.079545   0.712607   1.515    0.130
scooby.vaNeil Fanning -2.220455   0.504704  -4.400  1.3e-05 ***
scooby.vaScott Innes   0.065260   0.271943   0.240    0.810
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.7115 on 555 degrees of freedom
  (61 observations deleted due to missingness)
Multiple R-squared:  0.03778,   Adjusted R-squared:  0.03084
F-statistic: 5.448 on 4 and 555 DF,  p-value: 0.0002623
```

Console 11

All the p-values are greatly significant. This means we will use all the voice actor models. We again will likely get rid of these early because the factor levels are high.

Creating our Model

BIC

We just did a ton of data exploration to find significant variables that affect the IMDB score. Now let us mush it all together and get our draft of a model. This model will have a very high BIC score and co-variance *kappa* score. Our plan is to repeatedly remove variables, in order of greatest BIC reduction until the BIC score becomes positive with variable removal. So, without further ado, the messy model viewed below in *console 12:*

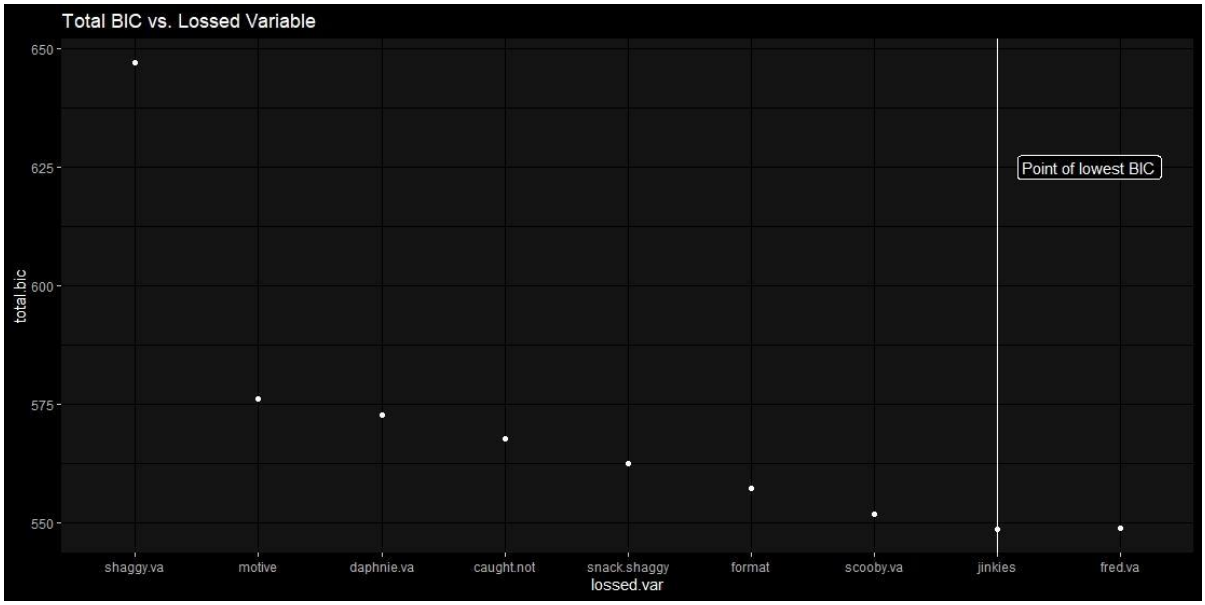
```
lm(formula = imdb ~ monster.real + monster.amount + snack.shaggy +
    caught.not + jinkies + rooby.rooby.roo + network + format +
    motive + fred.va + daphnie.va + velma.va + shaggy.va + scooby.va,
    data = scooby)

Residuals:
    Min      1Q  Median      3Q     Max
-0.9753 -0.2214  0.0000  0.1906  1.0930

Coefficients: (20 not defined because of singularities)
                                 Estimate Std. Error t value Pr(>|t|)
(Intercept)                       2.05863    0.79881   2.577  0.01042 *
monster.realTRUE                  0.35921    0.13304   2.700  0.00731 **
monster.amount                   -0.01450    0.01669  -0.869  0.38564
snack.shaggyTRUE                 -0.01012    0.10397  -0.097  0.92253
caught.notTRUE                    0.16938    0.18845   0.899  0.36943
jinkies                          -0.02235    0.01556  -1.436  0.15187
rooby.rooby.roo                  -0.04215    0.03094  -1.363  0.17399
networkAdult Swim                 1.90373    0.59970   3.174  0.00165 **
networkBoomerang                  0.52603    0.30754   1.710  0.08817 .
networkCartoon Network            0.47409    0.30193   1.570  0.11737
networkCBS                        0.24145    0.09090   2.656  0.00831 **
networkThe CW                     3.35143    0.52591   6.373 6.59e-10 ***
networkThe WB                     0.35669    0.33432   1.067  0.28682
networkWarner Bros. Picture       4.02691    0.79820   5.045 7.68e-07 ***
networkWarner Home Video         -0.54607    0.32559  -1.677  0.09450 .
formatMovie                       0.84671    0.26548   3.189  0.00157 **
formatMovie (Theatrical)               NA         NA      NA       NA
formatTV Series                   1.04992    0.20921   5.019 8.72e-07 ***
formatTV Series (segmented)       0.41124    0.35011   1.175  0.24105
motiveAssistance                 -0.11740    0.61287  -0.192  0.84822
motiveCompetition                -0.36443    0.38957  -0.935  0.35027
motiveConquer                    -0.11700    0.42436  -0.276  0.78295
motiveCounterfeit                -0.32499    0.41796  -0.778  0.43740
motiveEntertainment              -0.18823    0.49003  -0.384  0.70114
motiveExtortion                  -0.36566    0.48064  -0.761  0.44736
motiveFood                        0.67129    0.62935   1.067  0.28695
motiveImagination                -0.30666    0.54285  -0.565  0.57254
motiveInheritance                 0.10840    0.42107   0.257  0.79701
motiveNatural Resource           -0.41361    0.39949  -1.035  0.30130
motivePreservation               -0.30189    0.40512  -0.745  0.45672
motiveSafety                     -0.16206    0.48019  -0.337  0.73597
motiveSimulation                 -0.19808    0.48373  -0.409  0.68246
motiveSmuggling                  -0.24511    0.40072  -0.612  0.54120
motiveTheft                      -0.17431    0.39083  -0.446  0.65592
motiveTreasure                   -0.26303    0.39535  -0.665  0.50634
motiveTrespassing                 1.22451    0.64546   1.897  0.05873 .
fred.vaFrank Welker               0.25454    0.16658   1.528  0.12750
fred.vaFreddie Prinze Jr.        -6.07496    0.78577  -7.731 1.44e-13 ***
fred.vaRobbie Amell               1.75165    0.74859   2.340  0.01991 *
fred.vaZac Efron                       NA         NA      NA       NA
daphnie.vaGrey DeLisle            4.26203    0.71576   5.955 6.96e-09 ***
daphnie.vaHeather North          -0.31715    0.11839  -2.679  0.00777 **
daphnie.vaKate Melton                  NA         NA      NA       NA
daphnie.vaKellie Martin                NA         NA      NA       NA
daphnie.vaMarry Kay Bergman       4.45833    0.70243   6.347 7.64e-10 ***
```

```
daphnie.vaSarah Michelle Gellar             NA         NA      NA       NA
daphnie.vaStefanianna Christopherson        NA         NA      NA       NA
velma.vaChristina Lange                     NA         NA      NA       NA
velma.vaGina Rodriuez                       NA         NA      NA       NA
velma.vaHayley Kiyoko                       NA         NA      NA       NA
velma.vaKate Micucci                 -5.63018    0.63734  -8.834  < 2e-16 ***
velma.vaLinda Cardellini                    NA         NA      NA       NA
velma.vaMarla Scott                  -0.08636    0.15802  -0.546  0.58511
velma.vaMindy Cohn                   -4.96053    0.63101  -7.861 6.09e-14 ***
velma.vaNicole Jaffe                        NA         NA      NA       NA
velma.vaPat Stevens                         NA         NA      NA       NA
velma.vaStephanie D'Abruzzo          -6.51887    0.74358  -8.767  < 2e-16 ***
shaggy.vaCasey Kasem                  4.80673    0.58724   8.185 6.81e-15 ***
shaggy.vaMatthew Lillard              5.38247    0.56658   9.500  < 2e-16 ***
shaggy.vaNick Palatas                       NA         NA      NA       NA
shaggy.vaScott Innes                        NA         NA      NA       NA
shaggy.vaScott Menville                     NA         NA      NA       NA
shaggy.vaWill Forte                         NA         NA      NA       NA
scooby.vaFrank Welker                       NA         NA      NA       NA
scooby.vaHadley Kay                         NA         NA      NA       NA
scooby.vaNeil Fanning                       NA         NA      NA       NA
scooby.vaScott Innes                        NA         NA      NA       NA
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.3812 on 315 degrees of freedom
  (259 observations deleted due to missingness)
Multiple R-squared:  0.7226,    Adjusted R-squared:  0.6821
F-statistic: 17.84 on 46 and 315 DF,  p-value: < 2.2e-16
```

Console 12

And just like that, we created an ugly model. Alongside that many former p values are now insignificant. On top of that many values broke. I am not exactly sure why this is, but it makes removing certain variables easier. Instead of going through one at a time I will make a graph with the Y-axis being the new BIC value and the X-axis being the variable I remove.

After finishing this I am left with the graph below on *Plot 14*:



Plot 14

In the end we had to cut the variables:

Shaggy Voice Actor, Motive, Daphne Voice Actor, Not Caught, Shaggy Snack, Format, and Scooby Voice Actor

Our model now looks like:

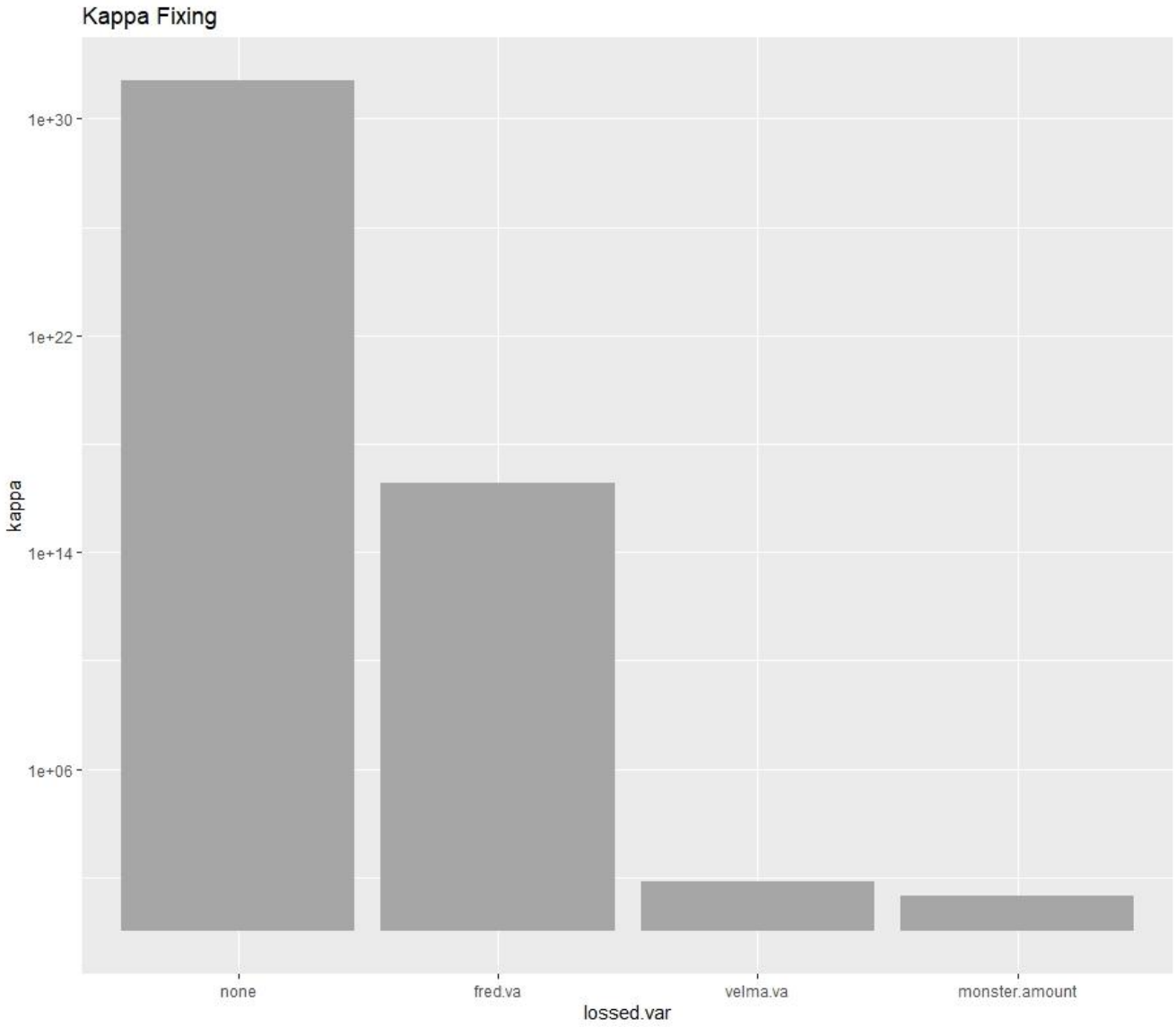
```
Residuals:
    Min      1Q  Median      3Q     Max
-1.3632 -0.2301  0.0000  0.2667  2.2910

Coefficients: (6 not defined because of singularities)
                           Estimate Std. Error t value Pr(>|t|)
(Intercept)                 7.56823    0.09247  81.842  < 2e-16 ***
monster.realTRUE            0.32228    0.13019   2.475 0.013786 *
monster.amount             -0.06002    0.01777  -3.377 0.000818 ***
rooby.rooby.roo            -0.10469    0.03043  -3.440 0.000653 ***
networkAdult Swim           0.51637    0.45109   1.145 0.253127
networkBoomerang            0.68270    0.21267   3.210 0.001452 **
networkCartoon Network      0.64146    0.19292   3.325 0.000979 ***
networkCBS                  0.54326    0.37623   1.444 0.149659
networkThe CW               0.09617    0.37887   0.254 0.799779
networkThe WB              -0.01934    0.20661  -0.094 0.925467
networkWarner Bros. Picture -1.52348    0.45741  -3.331 0.000960 ***
networkWarner Home Video   -0.53355    0.19465  -2.741 0.006442 **
fred.vaFrank Welker         0.07542    0.10569   0.714 0.475980
fred.vaFreddie Prinze Jr.  -0.27045    0.57437  -0.471 0.638038
fred.vaRobbie Amell        -3.17724    0.38557  -8.240 3.65e-15 ***
fred.vaZac Efron                NA         NA      NA       NA
velma.vaChristina Lange         NA         NA      NA       NA
velma.vaGina Rodriuez           NA         NA      NA       NA
velma.vaHayley Kiyoko           NA         NA      NA       NA
velma.vaKate Micucci       -0.63303    0.19689  -3.215 0.001427 **
velma.vaLinda Cardellini        NA         NA      NA       NA
velma.vaMarla Scott        -0.02532    0.17928  -0.141 0.887752
velma.vaMindy Cohn         -0.02941    0.18403  -0.160 0.873114
velma.vaNicole Jaffe       -0.12532    0.36740  -0.341 0.733235
velma.vaPat Stevens             NA         NA      NA       NA
velma.vaStephanie D'Abruzzo -1.47131    0.48819  -3.014 0.002771 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.4461 on 344 degrees of freedom
  (257 observations deleted due to missingness)
Multiple R-squared:  0.5876,    Adjusted R-squared:  0.5648
F-statistic:  25.8 on 19 and 344 DF,  p-value: < 2.2e-16
```

Console 13

There are both good and bad things from this model. For starters tons of significant values, next the F-statistic is well relevant. Our R^2 value turned out to be respectable at .5876. The bad things are that the Velma and Fred voice actor variables got shredded. If I were to guess, well rather I know, 257 observations were deleted due to missingness. This is something I feared with the data as "missing" was not particularly meant as "I forgot to put something in" but rather it was not there. I wish there was a way to just ignore it, but sadly linear models require nothing to be missing in the hat-matrix so will simply throw out the row if there is even one missing value. This does not mean however the non-missing data in the same variables were somehow corrupted, they are perfectly usable.

Collinearity

Next, we will check on the collinearity. I will do something very similar to the BIC however instead I will check the *kappa* value of the model instead of looking for the lowest value I will look for a reasonable value for *kappa*, which means under 30. Below is a graph of the decreasing *kappa* value per removed variable.



Plot 15

To fix our collinearity issue we dropped both the voice actors for Velma and Fred who both had heavy collinearity with each other and the network. Next, we removed the Monster Amount variable which it turns out had a

high collinearity with the Monster Real variable. Consulting with myself, a *Scooby-Doo* historian, I deemed that the monster being real was a more important variable to keep than the number of monsters variable. We managed to drop the *kappa* value from 1e32 to 20 which I consider a great win. Also, because the partial F-test of the model is so significant I do not see any purpose in doing extra F-testing. Now our final model bellow is:

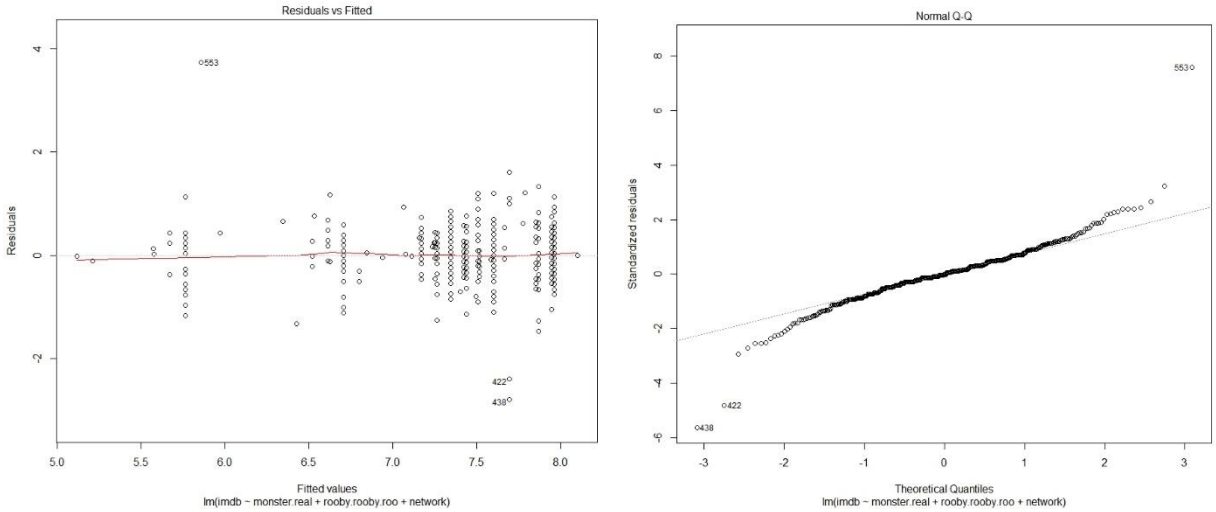
```
Call:
lm(formula = imdb ~ monster.real + rooby.rooby.roo + network,
    data = scooby)

Residuals:
    Min      1Q  Median      3Q     Max
-2.7959 -0.2447 -0.0092  0.2505  3.7409

Coefficients:
                           Estimate Std. Error t value Pr(>|t|)
(Intercept)                 7.43812    0.04875 152.562  < 2e-16 ***
monster.realTRUE           -0.17458    0.06735  -2.592  0.00982 **
rooby.rooby.roo            -0.09343    0.03096  -3.018  0.00268 **
networkAdult Swim           0.66188    0.50660   1.307  0.19200
networkBoomerang            0.16327    0.07950   2.054  0.04055 *
networkCartoon Network      0.52579    0.06914   7.604 1.50e-13 ***
networkCBS                  0.50753    0.08394   6.046 2.97e-09 ***
networkSyndication         -0.32392    0.29468  -1.099  0.27222
networkThe CW              -1.40441    0.11363 -12.359  < 2e-16 ***
networkThe WB               0.07942    0.08872   0.895  0.37113
networkWarner Bros. Picture -1.77258    0.29982  -5.912 6.39e-09 ***
networkWarner Home Video    -0.63553    0.09312  -6.825 2.64e-11 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.5042 on 484 degrees of freedom
  (125 observations deleted due to missingness)
Multiple R-squared:  0.5498,    Adjusted R-squared:  0.5396
F-statistic: 53.74 on 11 and 484 DF,  p-value: < 2.2e-16
```

Console 14

This is our final model. Of course, because we removed variables the R^2 value is lower but considering we managed to remove 3 variables and only decreased the 'Multiple R-squared' value by .03 from .5876 to our new value .5498 I will claim it a victory in terms of interpretation. Also, we no longer need to worry about those missing values from the voice actors.

Residuals

Let us take a glance at our residuals from this model. To do this we will simply apply the plot() function on the model and get the plots below:

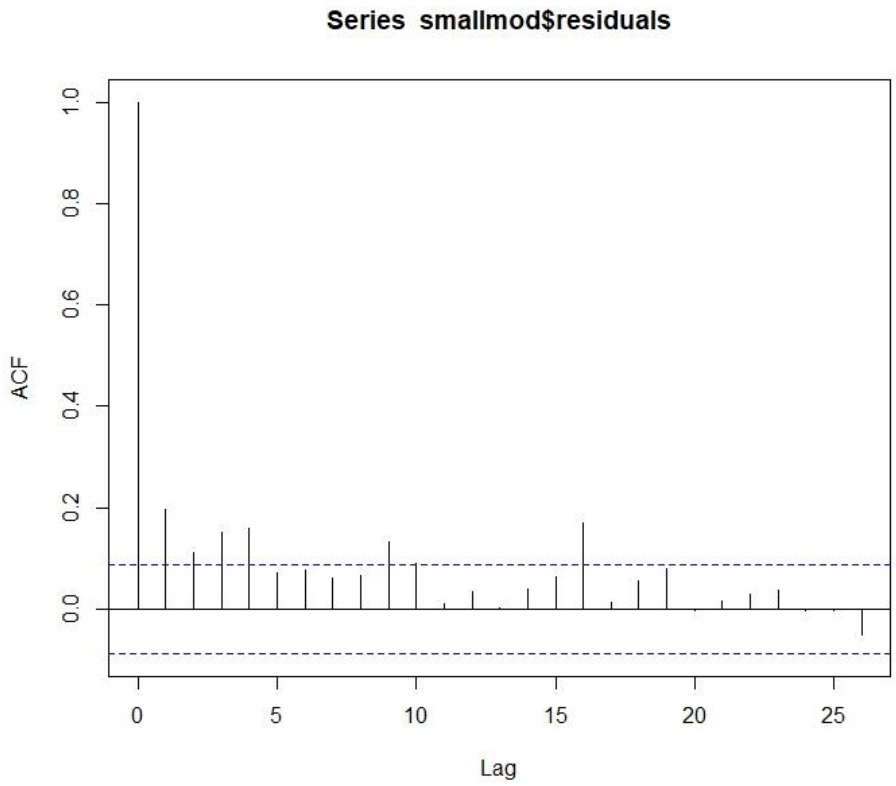


Plot 16 and 17

From a quick glance let us judge the fit of our model. In terms of homoscedasticity, or constant variance, the model seems a good fit. Looking at the residual vs fitted graph there are a few outliers, almost 4 z-scores above or below, but they are rare and somewhat evenly distributed.

In terms of linearity there is not much to say. Our residual vs. fitted plot lies almost perfectly horizontal on the x-axis at 0. We can assume that linearity is present in our model.

On normality we can look at our Q-Q plot. It does not seem the best, the dots do not perfectly fall across the lines and tend to tail off near the ends on both sides. Running a Shapiro-Test on the residual results in a p-value of 1.552e-15 which is well significant. This means that our residuals are not normally distributed.



Plot 18

In plot 18 we will check the residual independence using the autocorrelation function. This plot will check correlation between residuals of different lags (distances apart). If there is complete independence here, 95% of the bars (not including the first) should fall within the blue dashed lines. This is clearly violated as almost 1/4th the bars fall outside the 95% confidence interval. This really is not surprising. The residuals are heavily correlated with each other because residuals 1 apart are also almost always 1 episode apart so it makes perfect sense that there is a strong correlation for the lower values.

As we were attempting to explain, not predict what makes a good Scooby-Doo I feel these residual results are great. Homoscedasticity is something we do not want when explaining because it means that our model fails more or succeeds more depending on the *imdb* score. Next it is great that our linearity is on point which too is great for explanation. We do not want our

model to predict high for one set of fitted values or low for another. Normality is not met from our model. While this is not good it is also not the worst thing in terms of explanation I feel, this is because we have strong linearity, so the residual's means are still 0. Finally in terms of independence there is a definite correlation, however in this case because the data is already ordered in such a strange way as it is (mostly) chronologically ordered but also not evenly spaced. Like normality this isn't much of an issue in terms of explanation, in fact can help explain that nearby airing episodes/movies have a correlation. This means in general even though our residuals are not normal they are centered at 0 for all fitted values so their effect is significantly decreased.

Chapter 4

Interpreting our Model

At long last let us interpret our model. From the previous pages our model was:

```
Call:
lm(formula = imdb ~ monster.real + rooby.rooby.roo + network,
    data = scooby)

Residuals:
    Min      1Q  Median      3Q     Max
-2.7959 -0.2447 -0.0092  0.2505  3.7409

Coefficients:
                           Estimate Std. Error t value Pr(>|t|)
(Intercept)                 7.43812    0.04875 152.562  < 2e-16 ***
monster.realTRUE           -0.17458    0.06735  -2.592  0.00982 **
rooby.rooby.roo            -0.09343    0.03096  -3.018  0.00268 **
networkAdult Swim           0.66188    0.50660   1.307  0.19200
networkBoomerang            0.16327    0.07950   2.054  0.04055 *
networkCartoon Network      0.52579    0.06914   7.604 1.50e-13 ***
networkCBS                  0.50753    0.08394   6.046 2.97e-09 ***
networkSyndication         -0.32392    0.29468  -1.099  0.27222
networkThe CW              -1.40441    0.11363 -12.359  < 2e-16 ***
networkThe WB               0.07942    0.08872   0.895  0.37113
networkWarner Bros. Picture -1.77258    0.29982  -5.912 6.39e-09 ***
networkWarner Home Video    -0.63553    0.09312  -6.825 2.64e-11 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.5042 on 484 degrees of freedom
  (125 observations deleted due to missingness)
Multiple R-squared:  0.5498,    Adjusted R-squared:  0.5396
F-statistic: 53.74 on 11 and 484 DF,  p-value: < 2.2e-16
```

Console 14

We have 3 explanatory variables here predicting our one response variable. These are:

monster.real, *rooby.rooby.roo*, and network

monster.real is a Boolean true or false variable, *rooby.rooby.roo* is a numeric integer variable and finally network is a categorical variable. If we were to create a linear model equation from this it would be:

$$y_i = 7.438 + -0.174 \times x_1 + -0.093 \times x_2 +$$
$$(0.662 \times x_3 + 0.163 \times x_4 + 0.526 \times x_5 + 0.507 \times x_6 + -0.324 \times x_7$$
$$+ -1.404 \times x_8 + 0.079 \times x_9 + -1.773 \times x_{10} + -.636 \times + x_{11})$$

Where:

- y_i = *imdb score*
- x_1 = 0 if monster is fake, 1 if monster is real
- x_2 = number of times Scooby-Doo says "*Rooby-rooby-roo*"
- x_3 = 1 if network is *Adult Swim*
- x_4 = 1 if network is *Boomerang*
- x_5 = 1 if network is *Cartoon Network*
- x_6 = 1 if network is *CBS*
- x_7 = 1 if network is *Syndication*
- x_8 = 1 if network is *The CW*
- x_9 = 1 if network is *The WB*
- x_{10} = 1 if network is *Warner Bros. Picture*
- x_{11} = 1 if network is *Warner Home Video*

The intercept is the expected value on which all the explanatory variables are 0. This of course does not happen as at least one of the network values needs to be 1.

x_1, the monster being real. I believe the cause of this value being negative is when the monster is real the movie/episode tends to stray further away from the traditional format of the gang catching a monster in a mask. Scooby-Doo has many times tried to break away from their traditional format from the *Scooby Dark Ages* from the late 70s to early 90s which are the worst performing eras of *Scooby-Doo*. This could explain the negativity from this variable. Or maybe people in general just prefer the traditional format more.

x_2 is the amount of times Scooby-Doo says "*Rooby-rooby-roo*". This greatly surprised me as I did not think the catchphrases would have any effect on the *imdb* score. And personally, I do not have a great answer for why this might be. If I were to guess however it would be again during the dark ages,

they made the characters caricatures of themselves and reduced Scooby-Doo to a dog who just says *"Rooby-Rooby-Roo"*a lot. That would be my guess at least.

$x_3 - x_{11}$ are the response variables associated with the network. A large level factor variable like this I expected to make it into the final model as we all know just from common sense something like series, network, or format in general has a large impact on the *imdb* score.

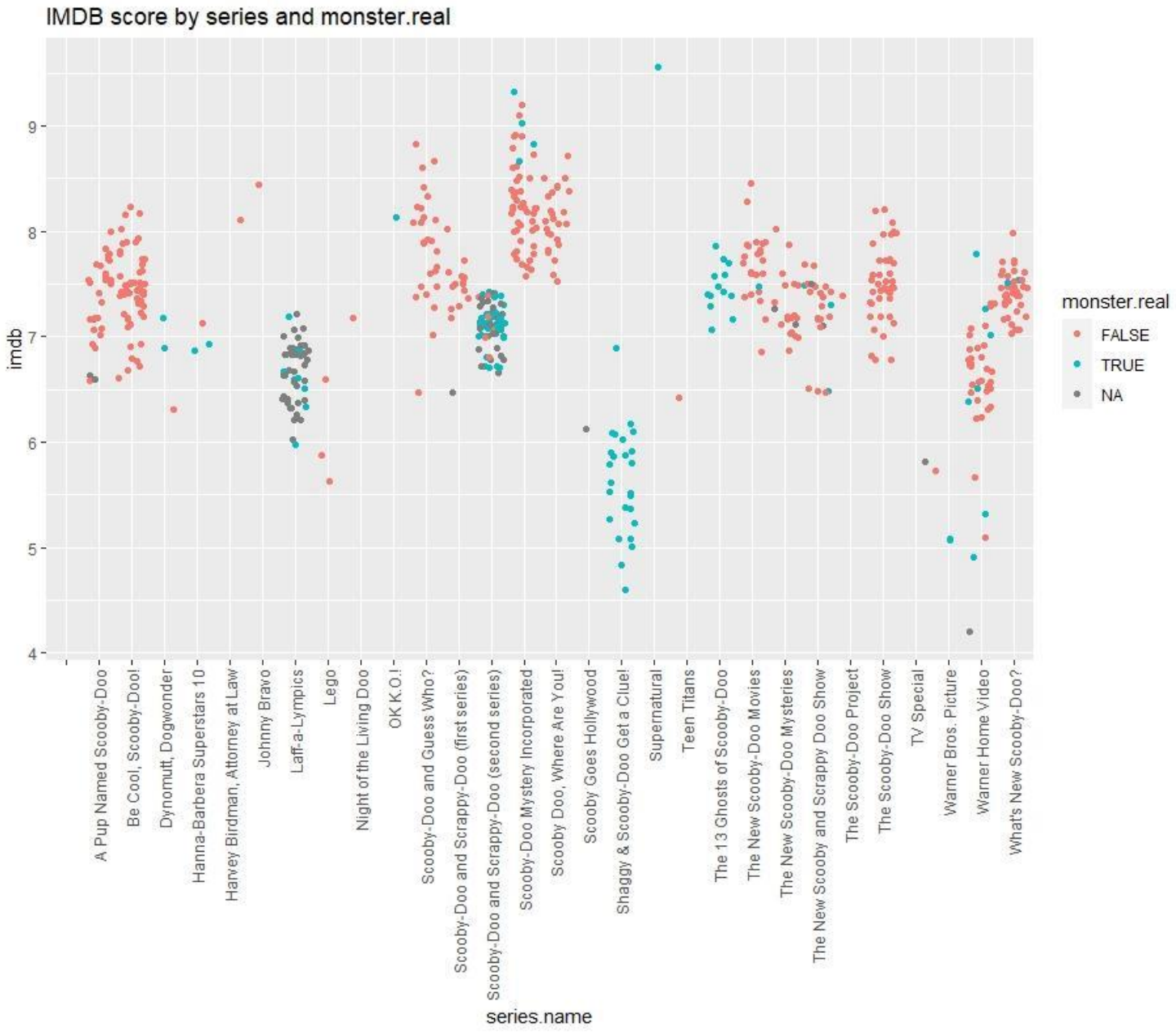
Our best networks turned out being: *Adult Swim*, *Cartoon Network*, and *CBS*. *Adult Swim* has had one *Scooby-Doo* iteration which was a crossover with *Johnny Bravo* and *Scooby-Doo!*. It came out in a time *Scooby-Doo* was recovering spectacularly from the 90s hiatus and the short stand-alone episode is funny even to this day. *Cartoon Network* created the highest rating *Scooby-Doo Series* which was *Scooby-Doo Monster Incorporated*. This series kept the traditional unmasking monster formula with an overarching mystery that is alluded to almost every episode. The series also delved into some forms of edginess with relationship drama, death, and cheating which it managed to pull off in a hilarious way in my opinion. Finally, *CBS* was the network that created the first *Scooby-Doo* series in the late 60s. They pioneered the concept of a gang of teens being chased by a monster and eventually finding out it is just some dude in a mask. They were responsible for only 2 series however the legacy of these series lives on in modern *Scooby-Doo* through emulating the exact same formula, callbacks, and re-using the original classic monsters.

Our worst networks were: *Warner Bros. Picture, The CW,* and *Warner Home Video*. To start off *Warner Bros. Picture* had only 3 movies released, and all were cinematic. In terms of engagement *Warner Bros. Picture* easily wins as their only 3 entries are the 3 highest engaged iterations however this does come at a cost in terms of *imdb* score. It is obvious that the more general

audience of a movie will have a more critical response than that of a small fanbase's response. Because of this *Warner Bros. Picture* is the worst ranked in terms of overall *imdb* score. Our second worst network is *The CW. The CW* is the only network in the bottom 3 to exclusively produce TV Series. They produced a 2-season series called *Shaggy & Scooby-Doo Get a Clue!* which is the worst rated series across all the *Scooby-Doo* series. The series gets rid of Fred, Daphne, and Velma, the villain is a repetitive Saturday morning cartoon villain, and there are no mysteries solved. It is not a mystery on why this show tanked and is considered the worst series to date. Finally, our third lowest network is *Warner Home Video. Warner Home Video* shocked me as it was here but the more, I thought about it the more I realized the reason. For starters *Warner Home Video* broke *Scooby-Doo* out of the 90s hiatus with huge hits such as *Scooby-Doo: Zombie Island, Scooby-Doo: Cyber-Chase, and Scooby-Doo: The Alien Invasion.* These were hits but since then *Warner Home Video* has been pumping out movie after movie, sometimes up to 3 a year. They are rather hit or miss. And as stated with the issue with *Warner Bros. Picture* movies tend to attract a more general and critical audience.

Conclusions

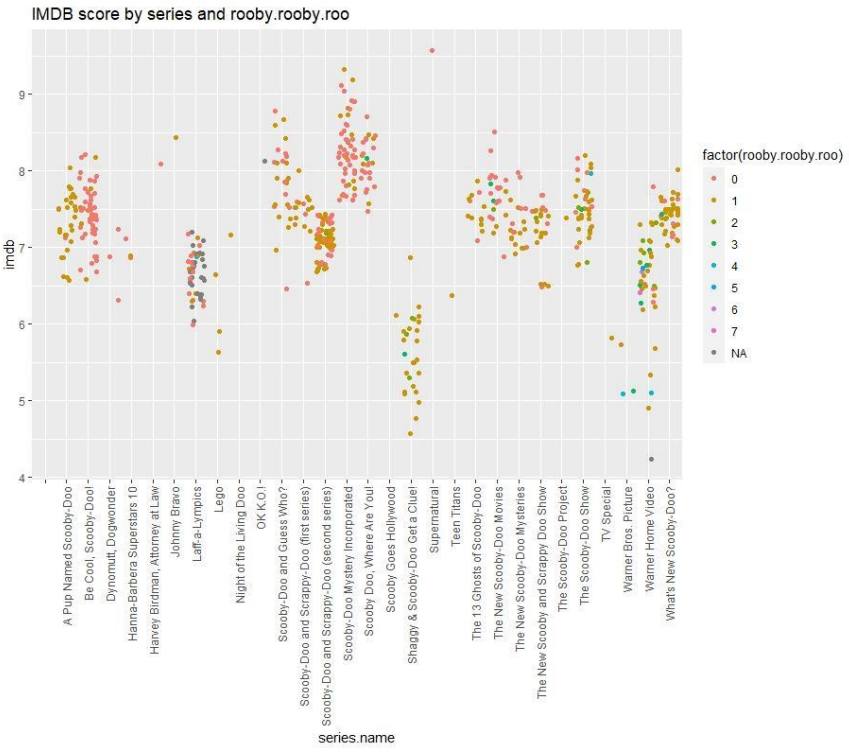
I could not be happier with our model. I was fearing the best model would have one or two obvious variables something like *network* and *series* which would have been wholly boring. Instead, we had *network* which was a rather obvious expected variable for predicting score, next we had a personally expected variable of *monster.real,* which showed us that the traditional format seems to be the preferred style. Finally, we had a "out of left field' variable in the form of *rooby.rooby.roo*. I did not expect such a strange variable to survive the model cleaning process, but it did. The model's residuals seem fine as well; besides normality everything is fine. Now at long last let us try to graph our model and see what R saw before us:



Plot 19

We can totally see more blue dots (monster is real) on the lower end of the y-axis as compared to red dots (monster is fake). I find it funny that the highest-ranking *Scooby-Doo* iteration: *Scoobynatural* had real monsters in it.

Now onto *rooby.rooby.roo*:



Plot 20

This graph is a bit messier to interpret. But I think we can see why the value is significant. Many iterations where *Scooby-Doo* did not say '*Rooby-Rooby-Roo*' once were among the higher-ranking series. This includes *Mystery Incorporated, 'Be Cool, Scooby-Doo!'* and *The New Scooby-Doo Movies*. While lower ranked values have somewhat of a lower ranked value.

As previously stated in my mistakes and regrets section there were many mistakes I made while collecting data. As I am happy with how my model turned out I do not think there is anything I would change.

I do not plan to stop building the spreadsheet. *Scooby-Doo* is still airing there are more rows to add. I would like to try the same thing but maybe an AIC or R^2 model instead of a BIC model or a lesser forward/backwards selection.

Chapter 5

References

[1] *Scooby-Doo* Spreadsheet:
https://docs.google.com/spreadsheets/d/1Jdu4PvQ4F9RhzSBCVcMYp6gVbg_UZVuLCiDffFhtFBI/edit?usp=sharing

[2] *Scooby-Doo* Wiki
https://scoobydoo.fandom.com/f

[3] IMDB
https://www.imdb.com/

Appendix

```
#library
library(car)
library(corrplot)
library(ggdark)
library(tidyverse)
library(dplyr)
library(ggplot2)
#df
scooby = `Scooby.Doo.Complete...Episode.List(6)`

#cor
scooby %>%
  select(imdb, engagement, run.time, monster.real, monster.amount, suspects.amount,
         caught.fred:caught.not, suspects.amount:arrested, culprit.amount, door.gag,
         split.up:blue.falcon) %>%
  cor(use="pairwise.complete.obs") %>%
  corrplot(type='upper')
```

```
cor(scooby$imdb, scooby$split.up, na.rm = TRUE)

scooby %>%
  select(imdb, monster.real, monster.amount, snack.shaggy, caught.not, jinkies,
rooby.rooby.roo) %>%
  cor(use="pairwise.complete.obs") %>%
  corrplot(type='upper')

lmod = lm(imdb ~ monster.real + monster.amount + snack.shaggy + caught.not + jinkies +
rooby.rooby.roo,
          data = scooby)
summary(lmod)
vif(lmod)
kappa(lmod)

#cat
scooby %>%
  ggplot(aes(x=network, y=imdb)) +
  geom_boxplot() +
  labs(title = 'Boxplot of IMDB by network') +
  theme_bw()

summary(lm(imdb ~ network, data=scooby))

sum(scooby$network == 'Syndication')
sum(scooby$network == 'Adult Swim')
sum(scooby$network == 'TBC')

scooby %>%
  ggplot(aes(x=imdb, col=format))+
  geom_density() +
  dark_mode() +
  labs(title = 'Density of IMDB score by format') +
  theme_classic()
```

```
sum(scooby$format == 'Movie (Theatrical)')

lm(imdb ~ format, data=scooby) %>% summary()
?lm()

sum(scooby$format == 'TV Series')

scooby %>%
  ggplot(aes(x = setting.terrain, y=imdb)) +
  geom_violin() +
  labs(title = 'Violin Plot of imdb score by terrain')

summary(lm(imdb ~ setting.terrain, data=scooby))

scooby%>%
  ggplot(aes(x=motive, y=imdb, col=motive)) +
  geom_jitter() +
  theme(axis.text.x = element_blank(),
        axis.ticks.x = element_blank()) +
  labs(title = 'IMDB Score by motive')

summary(lm(imdb ~ daphnie.va, data=scooby))

scooby %>% ggplot(aes(x=fred.va, y=imdb, col=fred.va))+
  geom_boxplot() +
  theme(axis.text.x = element_blank(),
        axis.ticks.x = element_blank()) +
  labs(title='Fred VA by IMDB score')

scooby %>% ggplot(aes(x=daphnie.va, y=imdb, col=daphnie.va))+
  geom_boxplot() +
  theme(axis.text.x = element_blank(),
        axis.ticks.x = element_blank()) +
  labs(title='Daphnie VA by IMDB score')
```

```
scooby %>% ggplot(aes(x=velma.va, y=imdb, col=velma.va))+
  geom_boxplot() +
  theme(axis.text.x = element_blank(),
        axis.ticks.x = element_blank()) +
  labs(title='Velma VA by IMDB score')

scooby %>% ggplot(aes(x=shaggy.va, y=imdb, col=shaggy.va))+
  geom_boxplot() +
  theme(axis.text.x = element_blank(),
        axis.ticks.x = element_blank()) +
  labs(title='Shaggy VA by IMDB score')

scooby %>% ggplot(aes(x=scooby.va, y=imdb, col=scooby.va))+
  geom_boxplot() +
  theme(axis.text.x = element_blank(),
        axis.ticks.x = element_blank()) +
  labs(title='Scooby VA by IMDB score')

summary(lm(imdb ~ fred.va, data=scooby))
summary(lm(imdb ~ daphnie.va, data=scooby))
summary(lm(imdb ~ velma.va, data=scooby))
summary(lm(imdb ~ shaggy.va, data=scooby))
summary(lm(imdb ~ scooby.va, data=scooby))

#model
bigmod = lm(imdb ~ monster.real + monster.amount + snack.shaggy + caught.not + jinkies
+
            rooby.rooby.roo + network + format + motive + fred.va +
            daphnie.va + velma.va + shaggy.va + scooby.va, data=scooby)
summary(bigmod)

#BIC
BIC(bigmod)

newmod1 = update(bigmod, .~.-  shaggy.va);BIC(bigmod) - BIC(newmod1)
```

```
newmod2 = update(newmod1, .~. -motive);BIC(newmod2) - BIC(newmod1)
newmod3 = update(newmod2, .~. -daphnie.va);BIC(newmod3) - BIC(newmod2)
newmod4 = update(newmod3, .~. -caught.not);BIC(newmod4) - BIC(newmod3)
newmod5 = update(newmod4, .~. -snack.shaggy);BIC(newmod5) - BIC(newmod4)
newmod6 = update(newmod5, .~. -format);BIC(newmod6) - BIC(newmod5)
newmod7 = update(newmod6, .~. -scooby.va);BIC(newmod7) - BIC(newmod6)
newmod8 = update(newmod7, .~. -jinkies);BIC(newmod8) - BIC(newmod7)
newmod9 = update(newmod8, .~. -fred.va);BIC(newmod9) - BIC(newmod8)

print('lolololololololololol')
BICdf = data.frame(lossed.var = c('shaggy.va', 'motive', 'daphnie.va', 'caught.not',
'snack.shaggy',
           'format', 'scooby.va', 'jinkies', 'fred.va'),
         total.bic = c(BIC(newmod1), BIC(newmod2), BIC(newmod3), BIC(newmod4),
BIC(newmod5),
           BIC(newmod6), BIC(newmod7), BIC(newmod8), BIC(newmod9)))
BICdf$lossed.var = factor(BICdf$lossed.var, levels=unique(BICdf$lossed.var))

BICdf %>%
  ggplot(aes(x=lossed.var, y=total.bic)) +
  geom_point() +
  dark_mode() +
  labs(title='Total BIC vs. Lossed Variable')+
  geom_vline(xintercept = 8) +
  geom_label(label='Point of lowest BIC',
           x=8.75, y = 625)

#Co-linear
medmod = newmod8
summary(medmod)
kappa(medmod);k0 = kappa(medmod)
medmod1 = update(medmod, .~.-fred.va);kappa(medmod1);k1=kappa(medmod1)
medmod2 = update(medmod1, .~.-velma.va);kappa(medmod2);k2=kappa(medmod2)
medmod3 = update(medmod2, .~.-monster.amount);kappa(medmod3);k3=kappa(medmod3)
```

```
kappadf = data.frame(kappa=c(k0,k1,k2,k3),
          lossed.var=c('none', 'fred.va', 'velma.va', 'monster.amount'))
kappadf$lossed.var = factor(kappadf$lossed.var, levels=unique(kappadf$lossed.var))

kappadf %>%
  ggplot(aes(x=lossed.var, y = kappa))+
  geom_col()+
  scale_y_log10()+
  labs(title='Kappa Fixing')

#Resid
smallmod = medmod3
summary(smallmod)
plot(smallmod)
shapiro.test(smallmod$residuals)

scooby %>%
  ggplot(aes(x = series.name, y = imdb, col = monster.real)) +
  geom_jitter() +
  labs(title='IMDB score by series and monster.real') +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))

scooby %>%
  ggplot(aes(x = series.name, y = imdb, col = factor(rooby.rooby.roo))) +
  geom_jitter()+
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))+
  labs(title='IMDB score by series and rooby.rooby.roo')
```