

**Picking the Low Hanging Fruits: Cleaning and Exploring Netflix Data using R**

By Michael Hoang

What’s up! It’s your boy! Back at it again with another article. So last time, I gave some ideas on how to build an unguided project, along with some advice. Now’s a good time to put all of that into action. The only question is where to begin? Well, considering that I spent the last 14 months of my life during the global lockdown binging on the various different streaming services, I figured it might be a good idea to look at some data from the world’s most prolific streaming service, [Netflix](http://www.netflix.com).

OK, let’s face it. Netflix is basically in the ranks as [one of the world’s titan tech companies with Amazon, Facebook and Google](https://www.fool.com/investing/stock-market/market-sectors/information-technology/faang-stocks/#:~:text=FAANG%20is%20an%20acronym%20used,turning%20the%20acronym%20into%20FAANG.). They figured out the psychology of how we consume content and what we’re willing to watch through their ingenious use of machine learning and made huge profits. However, if you’ve ever just taken a step back and look at what’s actually on Netflix, you’ll realize that it’s a giant mix bag of random content that really isn’t catered to anyone that varies in terms of content quality. Like for every A-list cast member with an award-winning performance there are lots of no-named actors/actresses credited with hot garbage.

Nevertheless, we’ve never seemed to mind what we’re watching, which if you think about it is really odd. You would think that we would be more critical of its content, but rarely have you heard anything negative about it. Is there something about the content that make it so, like the cast, director or genre? Maybe it has to do with the content rating where there’s a looser restrictive? Perhaps the content is really skewed to favor a particular cohort over others in terms of available quality content (i.e., those not borrowing a password VS. those that are).



“*When you’ve been putting this off to tomorrow for the past 5 years.*”

Since I have no idea why this may be the case, I figured it would be a good idea to look at what sort of content Netflix has to offer to find out. Specifically, it’ll be great to answer some of the following questions:

1. What’s the breakdown between TV series and movies? How does the breakdown differ in regards to genres?
2. What’s the breakdown between English-Speaking and non-English-speaking content?
3. What’s the distribution of Netflix content in terms of content ratings?
4. Which actors/actresses/directors are credited with the most headlining English-speaking or Non-English-speaking roles on Netflix?
5. What are some of the most common terms used to describe English-speaking and Non-English-Speaking Netflix content?

**THE DATA**

So, in order to get a clear idea of what Netflix has to offer, I first need a data set to work off of. With a quick Google search, I was able to find this [data set from Kaggle](https://www.kaggle.com/shivamb/netflix-shows) that contains about 7787 different titles. It contains a list of Netflix content dating back to 2010. Some of the variables contained in this data set include:

|  |  |
| --- | --- |
| **VARIABLE** | **DESCRIPTION** |
| show\_id | Netflix ID code |
| type | Is the content a TV Show or Movie |
| title | Title of the Netflix content |
| director\* | All credited directors for the content (lead + assistant) |
| cast\*\* | All credited cast member for content (lead + supporting) |
| country | All countries where the content was made |
| date\_added | Date when it was added on Netflix |
| release\_year | Year of release with content was released |
| rating | Content rating for audience |
| duration | Length of duration for the given content |
| listed\_in | All of the different genres applicable for the content (max. 3) |
| description | Content description |

\* The order of the names corresponds to the hierarchy in directing roles

\*\* The order of names corresponds to the casting hierarchy starting with headlining roles

**THE PROCESS**

As the title suggest, this project will be a simple project that serves to gain some understanding of the data that you are working with. While this isn’t the sexiest of things to do, it is nevertheless very important as it would serve as a foundation for other projects to be built off of it (*HINT*: there’s more articles to come). However, the first step before all of this is to do the prep work to be able to perform this data wrangling and exploration.

Some of the libraries that I would be using over the next little while as I work with this data set:

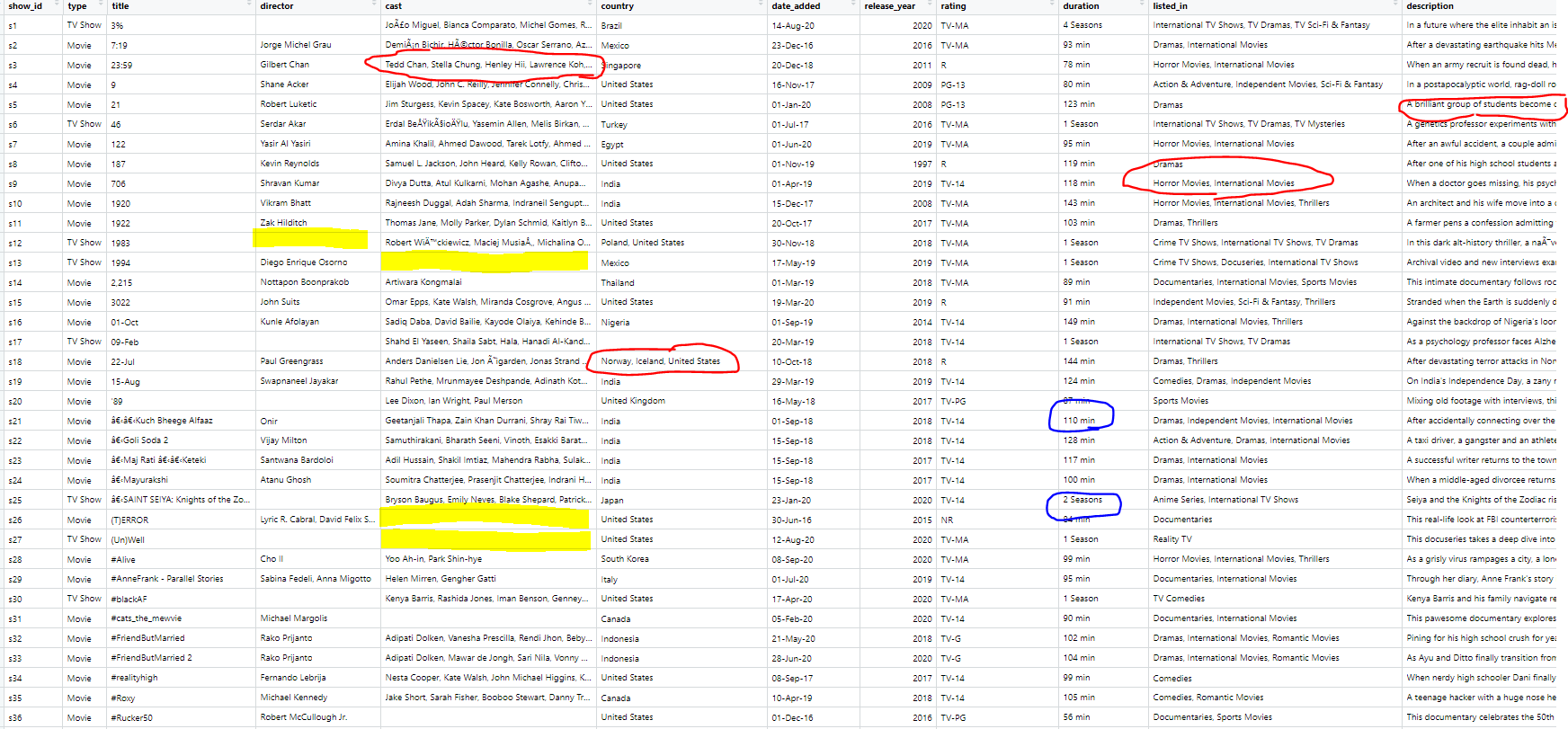
* [SKIMR](https://cran.r-project.org/web/packages/skimr/index.html) – Used for a quick glance of the data set
* [TIDYTEXT](https://cran.r-project.org/web/packages/tidytext/index.html) – I’ll be working with text data, so this makes the process easy to do when introducing stop words and filtering them out.
* [TIDYVERSE](https://cran.r-project.org/web/packages/tidyverse/index.html) – A collection of packages that makes the process of tidying data really easy
* [SHINY](https://cran.r-project.org/web/packages/shiny/index.html) – I’ll eventually be using this to make this interactive (*check out my next two articles*)
* [WORDCLOUD2](https://cran.r-project.org/web/packages/wordcloud2/wordcloud2.pdf) – We’ll be making word cloud at some point
* [CLUSTER](https://cran.r-project.org/web/packages/cluster/cluster.pdf) – Will be using this for the final project involving this data set

```

netflix = read.csv(“netflix\_title.csv”)

View(netflix)

```



After reading in the dataset and checking out the first few rows, you’ll notice that we’ll come across a few issues:

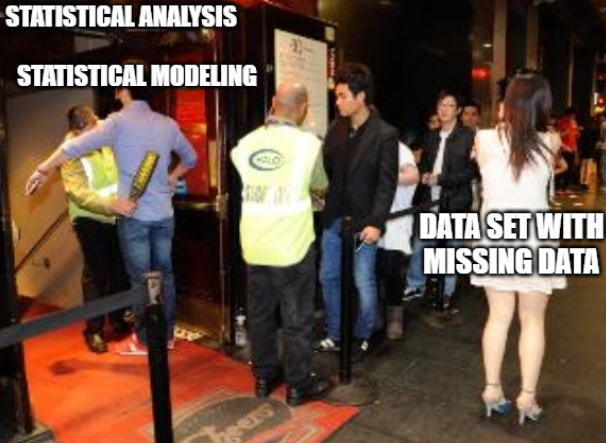
1. There are blanks in the data which resulted from the scrapping process
2. The text for cast, director, genre and description are all in one long string that needs to be separated out
3. Duration variable looks to have a distinction between TV series and movies whereby movies are recorded in minutes and series in terms of number of seasons
4. We have non-ASCII characters that we need to contend with
5. The date\_added section has two different formats used in inputting dates

Essentially, there’s a lot of stuff that needs to be done before even going into the analysis. But this is why reading in this data is so important since it gives us an idea of all of the little things that needs to be addressed so that we get the most accurate analysis as possible. So, let’s go through each of these step-by-step.

**STEP 1: DEALING WITH BLANKS REPRESENTING MISSING ENTRIES**

Blank entries, or more commonly known as missing data, are the most common thing that you’ll have to contend with in any data science/research project. While there are [many processes to handle them](https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3668100/), most of which are relatively easy to do, the actual manner in which you handle them is something that can make or break your analysis***.*** Let me explain:

In a perfect world, we would have a completely detailed data set that clearly describes each subject according to some chosen descriptor. However, in real life, this is rarely ever going to happen and we would have no idea as to the reasoning for these missing entries. Like could it be because it truly doesn’t exist or an error in the data entry or whatever? We’re basically speculating at this point. However, we do know that having these missing entries will prevent us from conducting key analysis test such as [statistical hypothesis tests](https://en.wikipedia.org/wiki/Statistical_hypothesis_testing) (i.e., *T-test, Chi-Square Test, ANOVA*) or most [statistical modeling techniques](https://en.wikipedia.org/wiki/Statistical_model) (i.e., regression analysis). So, there’s an obvious need to figure this out.



“*It pays to do those little extra steps…*”

Normally, the process will be to either remove segments of the data with missing entries or imputing them with some value. However, this can become quite problematic as it negatively impacts the integrity of the data with the introduction of [bias](https://en.wikipedia.org/wiki/Bias_(statistics)). This means that this presently constructed data set, which is the most accurate representation of the information on Netflix content, will essentially become less of an accurate representation with these modifications. While a small change here or there wouldn’t really affect much in the grand scheme of things as it pertains to the accuracy of your statistical analysis to be correct (*what we refer to as* [*statistical power*](https://en.wikipedia.org/wiki/Power_of_a_test)), but making large-scale changes does. Particularly in the case where we need to make predictions using the available data.

While the number of missing data that would significantly impact the integrity of the data and your analysis isn’t universally concrete as the values stated range from [5%](https://datascienceplus.com/imputing-missing-data-with-r-mice-package/) to [40%](https://www.jclinepi.com/article/S0895-4356(02)00539-5/fulltext), a good rule of thumb that I follow (*largely because of my background in Epidemiology*) is that missingness of [10%](https://link.springer.com/article/10.1186/2193-1801-2-222) will likely introduce bias into the data.

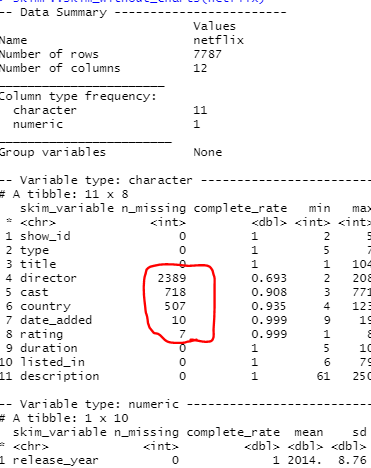


“*There’s a real art to finding the fight balance in things*”

Typically, the inclusion of blank entries to represent missing values is NOT THE NORM. In fact, when you access a count for the number of missing values, it’ll show up as being present instead of missing. There is typically another value used to reserve the fact that we have a missing entry like “NA” (used in R), “NaN” (used in Python), or the inclusion of some absurd value that has no real meaning. For example, if scores are usually in the range between 1 to 100, using a value like 999 would be a good way to represent null or missing values. In the case of R, I’ll replace these blanks with NAs using the [*mutate\_all()* function](https://rdrr.io/github/tidyverse/dplyr/man/mutate_all.html) found in the Tidyverse/dylyr package.



From assigning a symbol to represent missing values with NA, we can get a more accurate count of the missingness in the data using the [skim() function](https://docs.ropensci.org/skimr/reference/skim.html) from the skimr package.

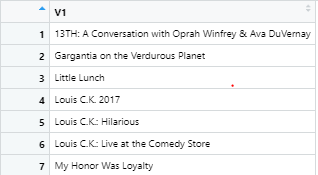


Here we can see that there are a number of variables of interest that has missing entries. Considering that we’ll be using this as the foundation for building additional projects which will benefit from having as accurate of a data to represent the population as possible, we’ll want to mitigate the degree of missingness here. With that in mind, it’ll probably be best to deal with this by starting from the variables with the lowest number of missing values and working our way up.

**STEP 2: IMPUTING MISSING VALUES**

Considering that the variables with the fewest number of missing entries (rating and date\_added) are something that is easily researched with help of the almighty Google search to find those missing entries, we can just manually impute the missing values in accordance to the formatting of the other entries. This would be possible with the use of the function [*mutate()*](https://dplyr.tidyverse.org/reference/mutate.html)*.*

For example: looking at those missing rating entries, which correspond to the content rating, we have the following:



```

netflix = netflix %>%

mutate(

rating = ifelse(c(is.na(rating) & title == "13TH: A Conversation with Oprah Winfrey & Ava DuVernay"), "TV-PG",

ifelse(c(is.na(rating) & title == "My Honor Was Loyalty"), 'PG-13',

ifelse(c(is.na(rating) & title == "Gargantia on the Verdurous Planet"), "TV-14",

ifelse(c(is.na(rating) & title == "Little Lunch"), "TV-Y7",

ifelse(c(is.na(rating) & title == "Louis C.K.: Live at the Comedy Store"), "TV-MA",

ifelse(c(is.na(rating) & title == "Louis C.K.: Hilarious"), "TV-MA",

ifelse(c(is.na(rating) & title == "Louis C.K. 2017"), "TV-MA", rating)))))))

)

```

However, for cases where the variable has a few hundred missing entries, like country, this approach will be highly inefficient. So, we’ll need to do some big brain strategy to make this work. Now this can vary in a number of ways depending on your approach, which is fine so long as this can be rationalized (i.e., after enough alcohol, I can lie to myself that it works).



So, this is my process:

**NOTE**: Working with the parameter that anything more than 10% of the data being missing or imputed equates to introducing bias into my future analyses, this would correspond to 778 rows that I can potentially drop and/or impute without running into a problem. Fortunately, with respect to the “country” variable having only 507 missing values, so we’re good.

Instead of having to individually search out each of the titles of movies or TV series that has a missing country of origin, a strategy that one can use is to isolate those missing rows.

```

View(netflix$country[is.na(netflix$country)])

```

Amongst the listed rows, you’ll realize that you can use the corresponding information from the other variables to figure out a potential country of origin. These include:

* GENRE:
  + “South Korea” for rows with “Korean TV Shows” listed as genre or
  + “United Kingdom” for rows with “British TV Shows” listed as genre
* TITLE:
  + imputing country for well-known film or TV franchises like “Monty Python” is quintessentially English and thus would likely have the “United Kingdom” listed as country of origin;
  + certain titles are duplicated because of language dubbing which we can use to select country of origin such as the case for titles containing “Tamil” or “Hindi” = “India”
* CAST and DIRECTOR:
  + While not always the case, certain actors are notorious for having a presence in a particular country’s cinema. For instance, I’m not likely going to find Solomon Khan in any sort of project outside of Bollywood, thus I can list that TV series or movie as having “India” as the country of origin
  + I’ve also used notably naming conventions used in certain countries to help with identifying country of origin such as the name “Aoi” or “Sasaki” for “Japan” or “Singh” for ‘India’
  + Similarly, certain director is notorious for existing in a particular country’s cinema like Quentin Tarantino being exclusively Hollywood and thus, his stuff should be listed as “United States”
* DESCRIPTION
  + Using keyword pertaining to notable locations, cohorts of people or language to define country of origin

As for anything that is leftover, I’ll just listed it as “Unknown”.

```

netflix = netflix %>% mutate(country = ifelse(is.na(country), "Unknown", country))

```

As for handling unknown director(s) and cast(s), the above process would not be a viable option since there would be multiple here. Also, considering that it’s entirely possible that are no listed cast or director due to the nature of the content (i.e., documentary or reality TV show), it’s absence would make sense. As such, we’ll just impute a term to indicate that this is unknown or non-existent. Considering that this really isn’t a matter of imputing data as much as it’s labelling missing entries, we’re basically good overall in terms of keeping the integrity of the data.

```

netflix = netflix %>%

mutate(

director = ifelse(is.na(director), "Unknown/No Director(s)", director),

cast = ifelse(is.na(cast), "Unknown/No Cast", cast)

)

```

**STEP 3: SEPARATING OUT THE TEXT**

This step actually isn’t too bad, it’s just a matter of appropriately using the piping capability afforded by the [magrittr package](https://cran.r-project.org/web/packages/magrittr/index.html) found in Tidyverse, along with the use of the [*separate()*](https://www.rdocumentation.org/packages/tidyr/versions/1.1.3/topics/separate)and the [*pivot\_longer()*](https://tidyr.tidyverse.org/reference/pivot_longer.html) function that is used to separate out the text and stack each of the separated words for a given row, respectively.

Now, as previously mentioned, the entries for this chained text variable correspond to the casting and directing hierarchy where the lead role or lead director is the first name listed. As such, we can separate each name out and have the initial cast or director name be stored as the “headlining actor/actress” and “lead director” respectively. Comparably, for the other text chained variables like genre and country, which has multiple entries as well, the same sort of hierarchy convention applies where the principal to tertiary listing correspond to the listing of each of these variables.

**NOTE**: In the case of genre and country, the splitting of text will result in some extra whitespaces which will need to be trimmed off. This is accomplished using the [*str\_trim()*](https://stringr.tidyverse.org/reference/str_trim.html) function.

```

netflix\_cast\_split = netflix %>%

separate(

cast, into = c("headliner", 'cast member 1', 'cast member 2', 'cast member 3', 'cast member 4', 'cast member 5', 'cast member 6', 'cast member 7', 'cast member 8', 'cast member 9', 'cast member 10', 'cast member 11', 'cast member 12', 'cast member 13', 'cast member 14', 'cast member 15', 'cast member 16', 'cast member 17', 'cast member 18', 'cast member 19', 'cast member 20', 'cast member 21', 'cast member 22', 'cast member 23', 'cast member 24', 'cast member 25', 'cast member 26', 'cast member 27', 'cast member 28', 'cast member 29', 'cast member 30', 'cast member 31', 'cast member 32', 'cast member 33', 'cast member 34', 'cast member 35', 'cast member 36', 'cast member 37', 'cast member 38', 'cast member 39', 'cast member 40', 'cast member 41', 'cast member 42', 'cast member 43', 'cast member 44', 'cast member 45', 'cast member 46', 'cast member 47', 'cast member 48', 'cast member 49'), sep = ", "

) %>%

pivot\_longer(headliner:`cast member 49`, names\_to = "cast\_type", values\_to = 'cast') %>%

filter(!is.na(cast)) %>%

mutate(cast\_type = ifelse(cast\_type == "headliner", "headliner", "supporting cast"))

netflix\_director\_split = netflix %>%

separate(

director, into = c("lead","assitant 1","assistant 2","assistant 3","assistant 4","assistant 5","assistant 6",

"assistant 7","assistant 8","assistant 9","assistant 10","assistant 11","assistant 12"), sep = ", ") %>%

pivot\_longer(lead:`assistant 12`, names\_to = "director\_type", values\_to = "director\_name") %>%

filter(!is.na(director\_name)) %>%

mutate(director\_type = ifelse(director\_type != "lead", "assistant/guest", "lead"))

netflix\_genre\_split = netflix %>%

separate(listed\_in, into = c('principal', 'secondary', 'tertiary'), sep = ",") %>%

pivot\_longer(principal:tertiary, names\_to = "listing\_type", values\_to = "genre") %>%

filter(!is.na(genre)) %>%

mutate(listing\_type = ifelse(listing\_type == "principal", "principal", 'secondary/tertiary'))

netflix\_genre\_split$genre = str\_trim(netflix\_genre\_split$genre, side = 'both')

netflix\_country\_split = netflix %>%

separate(

country, c("main country", "secondary country", 'tertiary country', 'fourth country', "fifth country", 'sixth country', "seventh country", 'eighth country', 'nineth country', 'tenth country', 'eleventh country', 'twelfth country'), sep = ",") %>%

pivot\_longer(`main country`:`twelfth country`, names\_to = "country\_type", values\_to = "country\_name") %>%

filter(!is.na(country\_name)) %>%

mutate(

country\_type = ifelse(country\_type == "main country", 'main country', 'other country'),

country\_name = ifelse(country\_name == "", "Unknown Country", country\_name)

)

netflix\_country\_split$country\_name = str\_trim(netflix\_country\_split$country\_name, side = 'both')

```

A special scenario comes with dealing with description section. The process here is a bit more nuanced as we’ll have to deal with a bunch of words that hold no real value in terms of information. These are the stop words. Additionally, I will also need to contend with punctuation marks, symbols and non-ASCII characters here. While this does seem like a lot, it’s actually pretty easy to do if we use a few key functions as shown below:

```

netflix\_description = netflix %>%

mutate(

description = gsub('[\\,.;:!?"]', "", description) # Substitute symbols and marks for blanks

)

netflix\_description = netflix\_description %>%

mutate(

description = stringi::stri\_trans\_general(description, “latin-ascii”) # converting non-ASCII

)

netflix\_description$original\_description = netflix\_description$description

# I just want to have the original to compare to

netflix\_description = netflix\_description %>%

unnest\_tokens(

output = word,

input = description # Enables splitting up the description by each word for an individual row

) %>%

anti\_join(

stop\_words,

by = “word” # allows for return all rows that aren’t included in the filter list (this case stop\_words)

)

colnames(netflix\_description) = c("show\_id", 'type', 'title', 'director', 'cast', 'country', 'date\_added', 'release\_year', 'rating', 'duration', 'listed\_in', 'original\_description', "keywords")

# renaming the columns

netflix\_description$keywords = str\_trim(netflix\_description$keywords, side = 'both')

# Removes whitespace from splitting description words

**STEP 4: Creating the necessary data sets**

This process may seem a bit strange since you might just create a very wide data set with all of the individual components. However, as I mentioned previously, we’re going to build off of this with two other projects whereby things are going to be WAY easier if you make a few data sets that separate certain variable text out. Plus, if I’m doing some exploratory analysis, something like this can make certain comparisons easy to do when looking at potential interaction effects. In my case, I’ll create data sets with the following separations:

1. Keywords in description & Genre
2. Keywords in description & Language
3. Director & Cast
4. Director & Genre
5. Director & Language
6. Cast & Genre
7. Cast & Language
8. Language & Genre
9. Director, Cast & Language
10. Director, Cast & Genre
11. Director, Language & Genre
12. Cast, Language & Genre
13. Cast, Language, Director & Genre

I’m not going to show the entire process here, but essentially you will need to use the same process above with separating long-text data with *separate() and pivot\_longer() function* with existing data set.

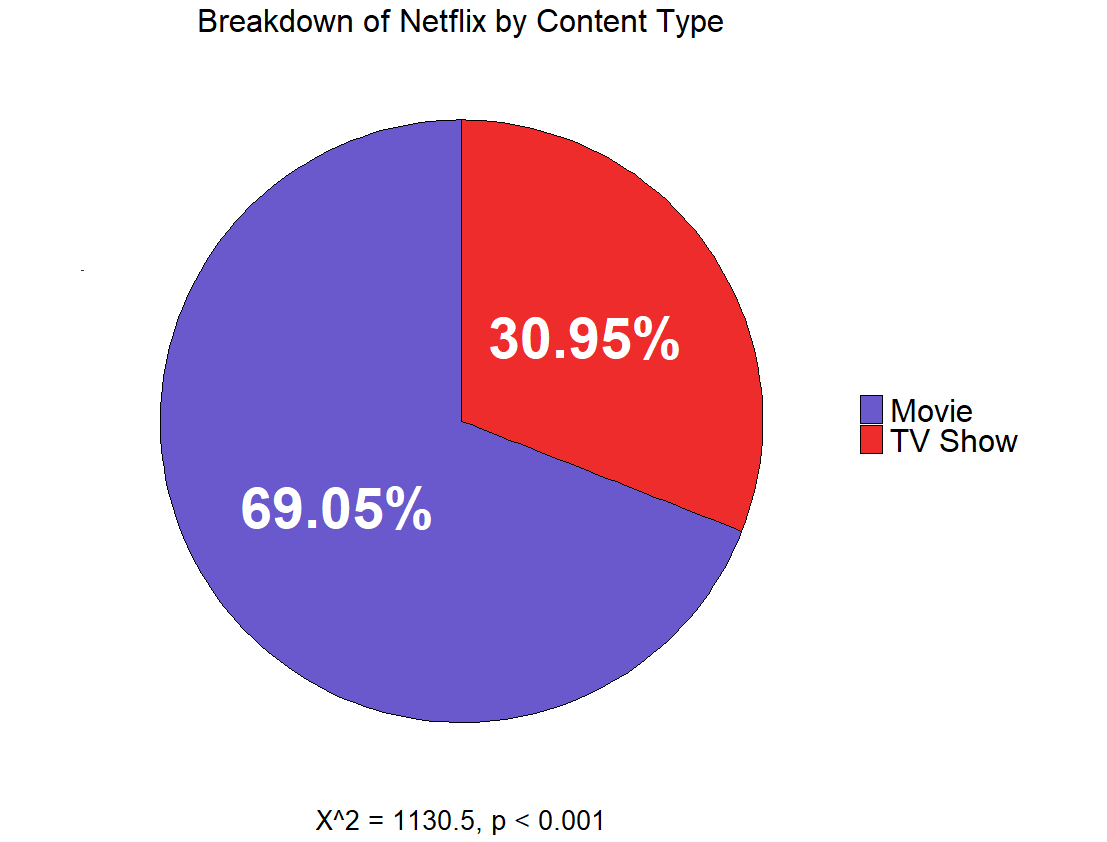
Boom, we’re now ready to do some exploration and answer those questions. If you want to check out the entire clean up sequence, [you can look it up here](https://github.com/Vibe1990/Shiny-Project-Ideas/blob/main/Netflix%20Project/Cleaning%20%26%20Pre-Processing).

**EXPLORATORY ANALYSIS**

We’re ready to do some exploration and answer some of those questions. Since the code can get pretty wild here, you can check out the code used to make these visualizations [here](https://github.com/Vibe1990/Shiny-Project-Ideas/blob/main/Netflix%20Project/Tier%201A:%20Visualization%20Stuff%20for%20Analysis).

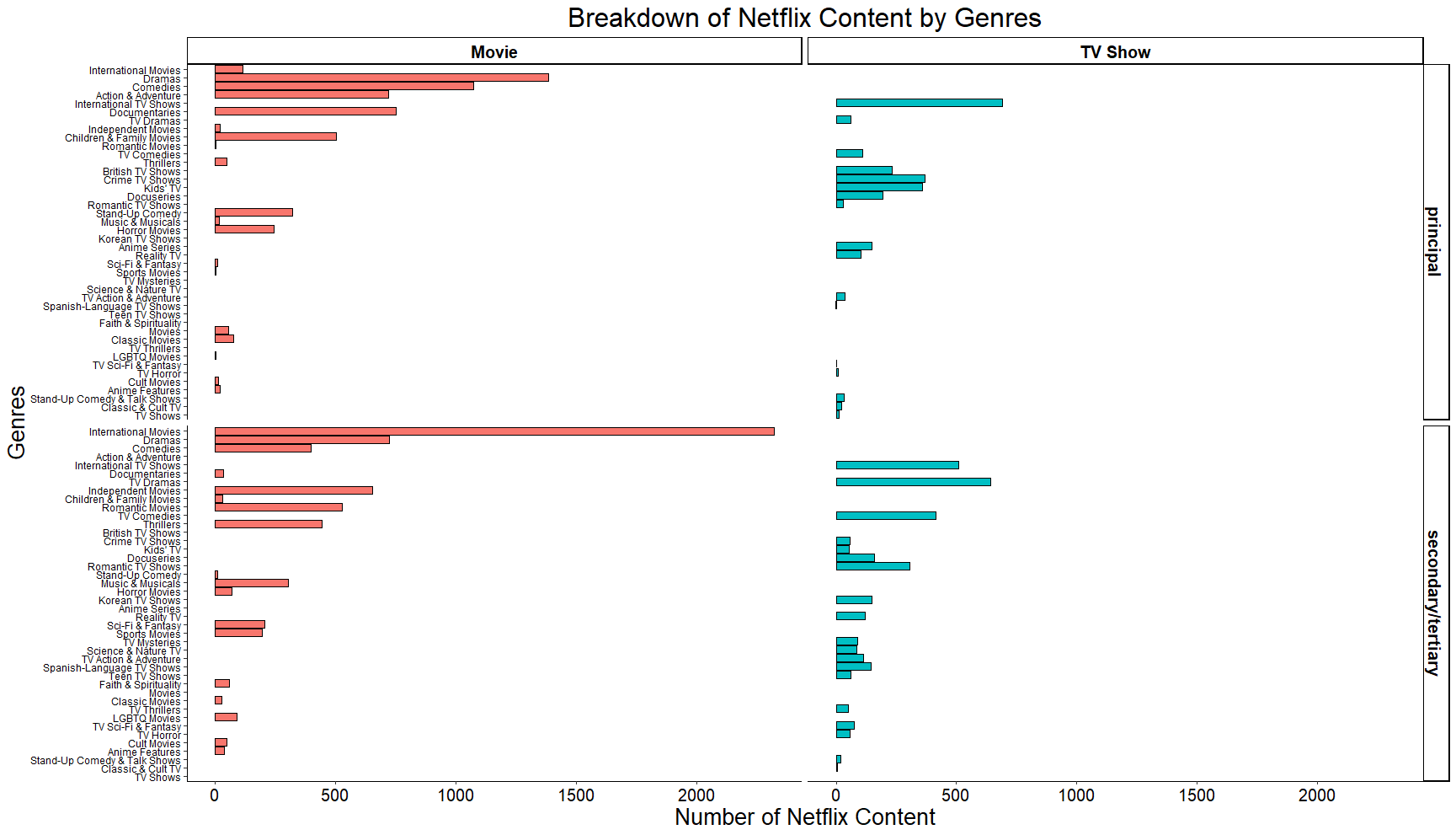
**1A) What’s the breakdown between TV series and movies?**

We can see that the majority of the content on Netflix are movies (~ 69%) as compared to TV series.



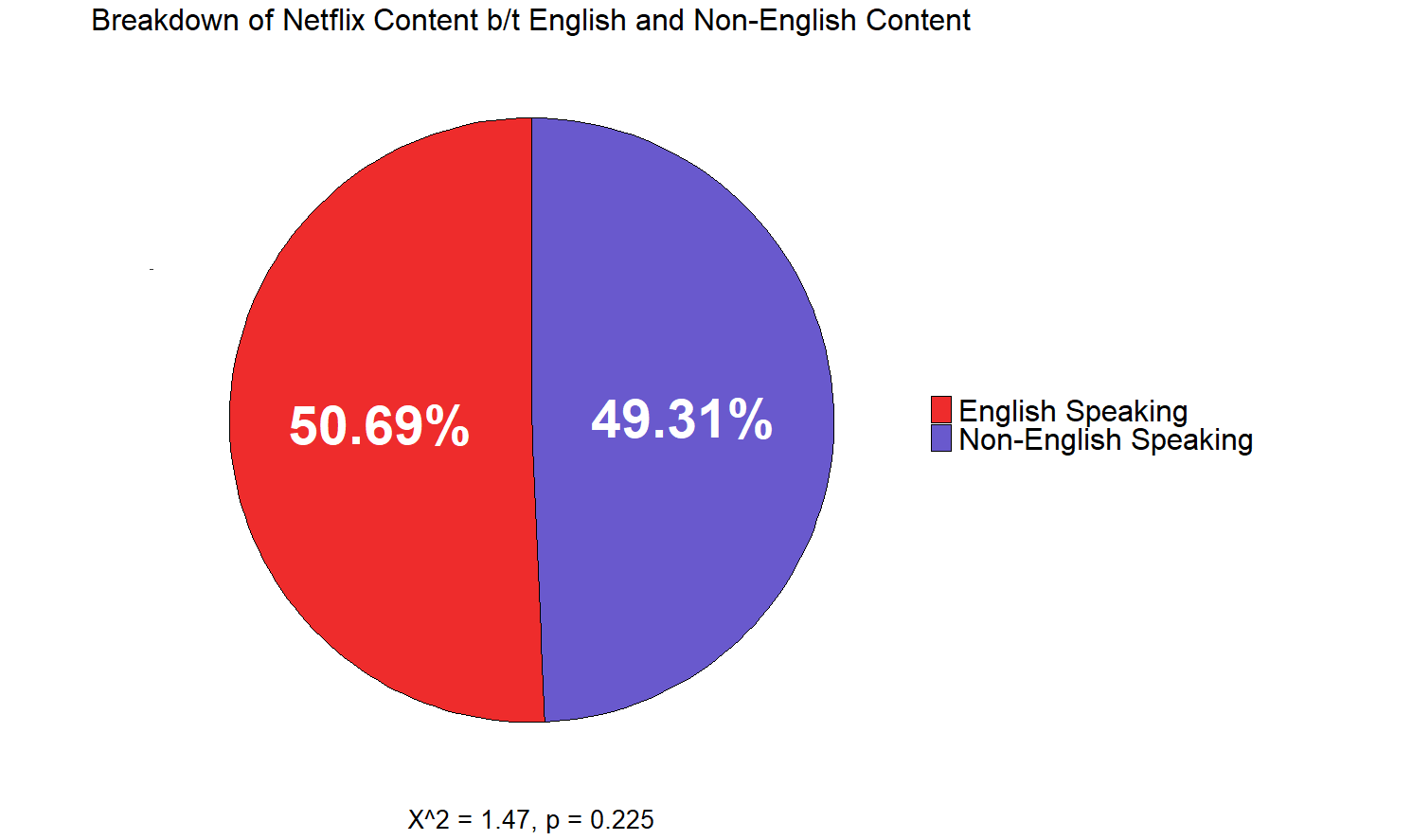
**1B)** **How does the breakdown differ in regards to genres?**

Examining a bit deeper in relation to genre, the majority of the content appears to be international content, however much of that is actually a secondary listing. In terms of primary categorization, most of the movie-related content on Netflix are principally dramas followed by comedies, whilst for TV series, it namely action and adventure.



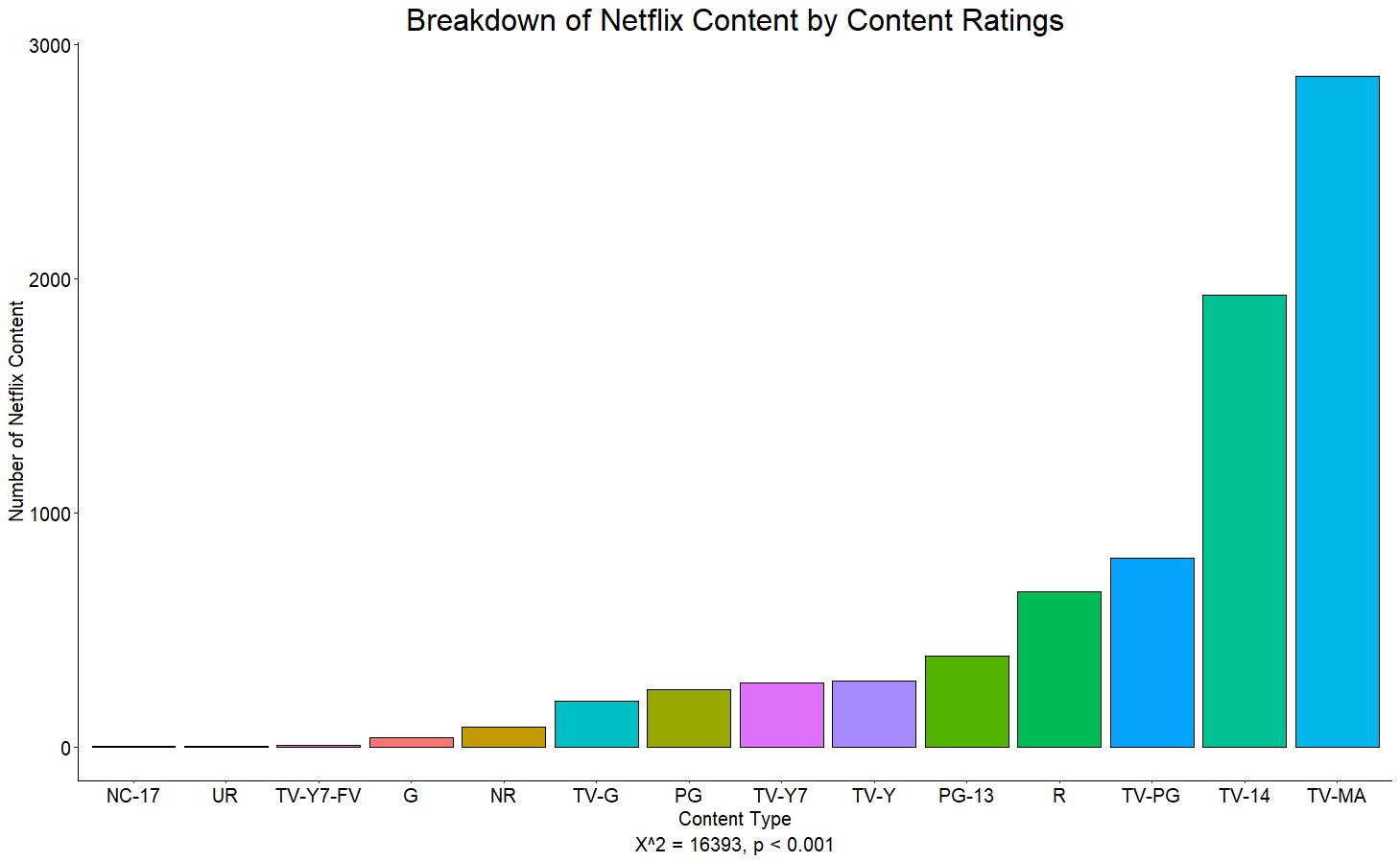
**2) What’s the breakdown between English-Speaking and non-English-speaking content?**

Interestingly enough, there is about a 50:50 split in terms of English and non-English content.



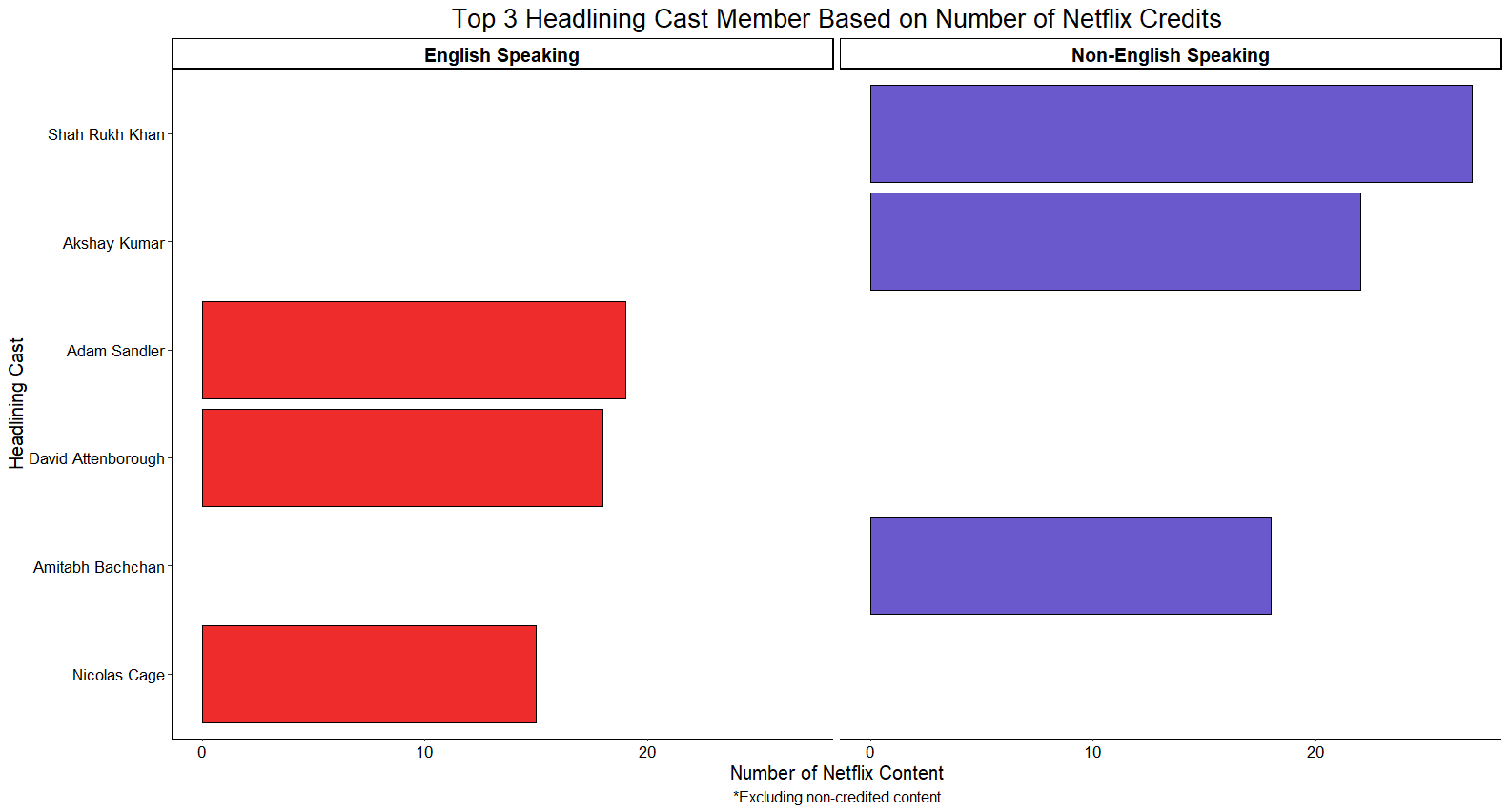
**3) What’s the distribution of Netflix content in terms of content ratings?**

In terms of content rating, the majority have a Mature rating, which is 17+.

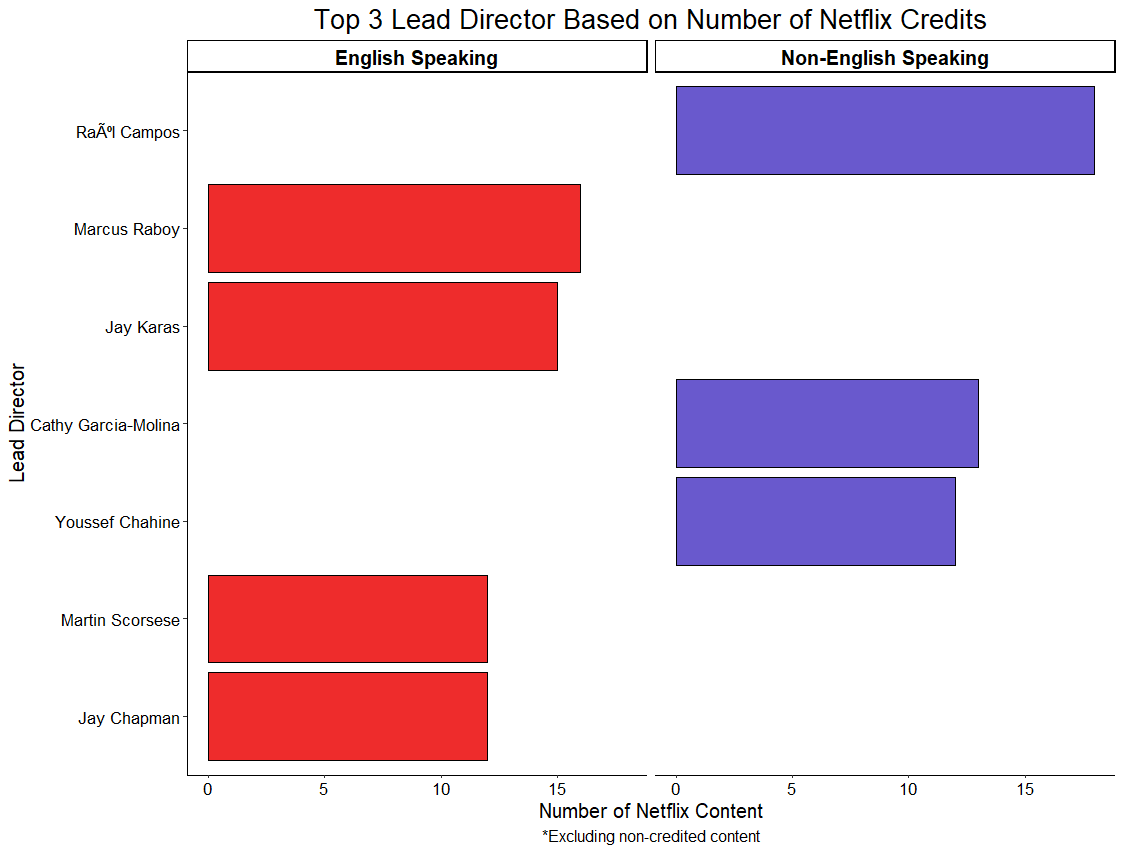


**4) Which actors/actresses/directors are credited with the most headlining English-speaking or Non-English-speaking roles on Netflix?**

Amongst the cast, we see that in terms of overall Netflix content that Shah Rukh Khan had the most credits as the lead. This was also the case for non-English-speaking content. However, in terms of English-speaking content, Adam Sandler had the most leading credits.



In a comparative analysis with lead directors, it was found that Raul Campos had the most directing credits for Netflix content overall as well as for non-English-speaking content. However, for English-speaking contents, Marcus Raboy had the most lead directing credits.



**5) What are some of the most common terms used to describe English-speaking and Non-English-Speaking Netflix content?**

Lastly, here is a quick look at some of the top keywords found in the description of English-speaking content with the use of word clouds. It appears that “life”, “family”, “world” and “documentary” are the most common words to appear.



As for non-English-speaking content on Netflix, the most common terms found in the description are “life”, “woman”, “family” and “love”.



**CONCLUSION**

Overall, this was a fairly straightforward task looking into Netflix data that would be appropriate for a Tier-1 data project as it really only took a few hours to a day to complete (most of which stemmed from the cleaning process). Going from data wrangling to exploratory analysis, every aspect of the process relied on those same foundational skills that we’ve built upon early on the data science path.

Sure, there may be some things that you’ve may have been unfamiliar with, particular with text data, but this can easily be reviewed. Everything else should remain fairly familiar to you in one way or another. Obviously, we can dive a lot deeper in this exploratory analysis by examining the interrelationship of the above comparisons with an interaction factor, say breakdown of Netflix content by content type and content rating.

So, what’s the next step? Well, I’m going to step this up a bit by introducing some more advance techniques to make better use of this data. How exactly will I be doing this? You’re just going to have to wait and see. So, keep an eye out for the next article.

If you’re interested in check out some of my other projects, you can head over to my [GitHub](https://github.com/Vibe1990) to check some of them out. Alternatively, if you’re got some idea on a collaborative project or just want to connect, hit me up on my [LinkedIn](https://www.linkedin.com/in/michael-hoang-3222a220/).

Thanks for the read.