

**From Start to Finish Without Hating Your Life: Advice on Unguided Data Projects**

By Michael Hoang

If you’re like me, you’ve probably spent a fair amount of time on [Data Quest](https://www.dataquest.io/path/data-analyst-r/), progressing through the material, and on the cusp of finishing one of the various pathways (*if not already there*). Congrats! Good job! Mazel Tov! You’re probably on your way to start to apply for some data-related positions or whatever. Now is a good time to start to take the next step and start building out your portfolio with a few projects. As someone with a few under my belt already, I figured I could share how to go about this from selection 🡪 completion.

**“*But wait, couldn’t I just use what I have done so far on DQ for my portfolio?*”**

Sure, you coulddo that. Maybe do a bit of sprucing up along the way. However, those projects probably aren’t going to get you very far. While there are a few reasons for this, it largely has to do with the fact that it’s a **guided project**. This means that YOU really didn’t do any of the heavy lifting in terms of planning, designing the project or trouble-shooting throughout each step. All of this has already been figured out for you and it’s just a matter of you following some laid-out instruction to some relatively laid-out end goal. What people want to see is all of these other qualities and not just your data analysis + coding skills.



“Like you can’t say you’re a really good cook if the most you’ve ever done was follow those meal-kit subscriptions like Blue Apron or HelloFresh. All you did was follow the instructions and not burn your home to the ground “.

**“*OK, I get what you’re saying. So, how do I go about it then?*”**

Before figuring out what you want to do, it’s a good idea to set your expectations which depends on where you are in terms of this process. If this would be your first ever unguided project, this *PROBABLY* won’t be something that is going to be super technical or earth-shattering. It’s going to be fairly simplistic, which is fine! It’s all about getting your feet wet with a few easy wins and then progressing from there. So, hold off on those really complex ideas for now.



I mean is your first time doing anything going to be that amazing? No, right.

However, for those that are a few projects deep, it’s should be about filling out your portfolio to have at least one project showcasing (1) visualization, (2) descriptive and/or inferential analysis, (3) predictive modelling, (4) machine leaning application and (5) a combination of these. If you do have a particular area you would like to focus, then your portfolio will probably be skewed towards that specialization.

With that in mind, choosing your next unguided project also comes down to finding the right balance matching project complexity with your current skill set. This is a time to be honest with yourself as it can mean the difference between getting your project done or not. That’s not to say that you won’t be able to do it if you’re off the mark, but the key is that we want to make sure that this is as painless of a process as possible and ensure that the project *actually* gets completed.

Look, you will inevitably hit a snag somewhere down the road where the answer won’t be so clear at all. In fact, you’ll likely go off track in your search for the answer in order to learn to do that thing to solve your problem and fall into some rabbit hole learning something that is completely unrelated to your original goal. This will be a complete waste of time and effort that we will want to avoid at all cost if possible.



Now, you might think this won’t happen. But trust me, it will!

**“*Alright I know what sort of project to do, and it’s feasible for me. What’s next?*”**

Great. The next step is the fun part. It’s finding a topic that interest you. Now this can be something related to your field of work or something that is a personal interest to you. More often than not, this will be a point of contention for many since there are so much data available out there to work on that it can get overwhelming quickly. While it’s tempting to want to do many things at once, I would highly suggest to avoid this as you’ll be more likely to end up with a series of half-finished work because you’ll be pulled in to too many different directions. So, nail down a single choice.

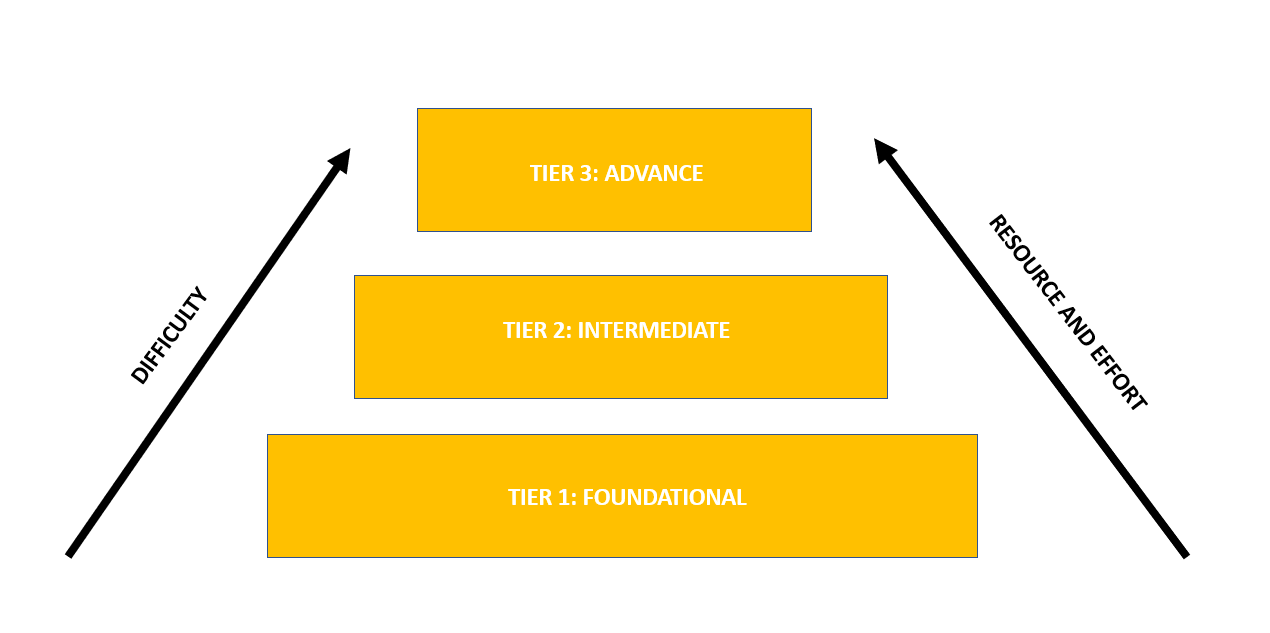
There are many ways to do this (i.e., use one of those [paper-fortune tellers](https://i.ytimg.com/vi/SAhiIlTxUYA/maxresdefault.jpg), spin a [virtual wheel](https://wheeldecide.com/), do a 10-hour gaming session fuelled by nothing but energy drinks and make a decision with a manic disposition that’s [akin to being on cocaine](https://www.researchgate.net/publication/265601410_Mania_induced_by_high_content_caffeinated_energy_drinks)), but I would suggest you choose a topic where you can tell a story with. This might sound weird, but keep up with me for a sec. As we’re using the projects in our portfolio to sell our skillsets, we want to provide one which are very engaging or salient to get a connection with the audience. I’ve always found an easy way to do so is through great story-telling. So, if you can find something that you’re passionate about and be able to spin an engaging story out of it, it’s definitely will be something that is worthwhile to pursue.



Think about all those times you had to make small talk or “get to know you” chats, people are usually receptive whenever there is a very engaging story being shared.

*“****Cool I’ve got my idea for a project. How do I go from idea to execution?****”*

Alright, it’s time to get to the brass tacks here. Knowing what your project will be, I would start with planning out what needs to be done, strategize the prospective protocols needed to get certain tasks completed and budget your time to get each step. Since most folks are more visual learners (*myself included*), it always helps to have a guide to visualize these thoughts based on the scope of the project. Here’s a good visual that helped me out with some of my projects:



Using a tier system like this one helps give you a good idea on how to strategize how to tackle a given project based on the scope of the project, specifically in terms of allocating resources and time. Obviously your million-dollar question would be “*What constitutes a tier-1, tier-2 or tier-3 project?*” Well, that sort of depends of a few factors, but mainly has to do with (1) your vision with the project and (2) your current skill set. Here’s how I would define it:

1. **TIER 1 – FOUNDATIONAL:** These are tasks that are pretty universal across any sort of data-related task (i.e., data wrangling, web scrapping, exploratory analysis, simplistic inferential analysis [like chi-square test or t-tests] and documentation). Something similar to those guided projects in the early stages of a DQ pathway. Usually reserve somewhere between a few hours or (*if you like to procrastinate like me*) 1 to 2 days to complete and is a great choice for those early along their data science path.
2. **TIER 2 – INTERMEDIATE**: We’re looking at a notable upgrade in terms of your task with things like more advanced visualization techniques, or more detailed reporting of data using inferential analyses that may involve some more classical machine learning techniques (like regression models). This would be something that would take a few days to a week of solid time to get done and would be something that would make up the bulk of your portfolio since it’s complex enough to show more advance skills + is something not too intensive to complete.
3. **TIER 3 – ADVANCE:**  This is where those cool and awesome looking project ideas that got you into data science lives. We’re talking about chat bots, automating certain tasks, predictive model implementations. Basically, this will be things that will take longer than a week to get done and would essentially serve as a highlight of your portfolio where you use it to show off those advance skills you’ve developed. This will be the most intensive thing you will probably do in your data science journey.

Now this example might not exactly match your current skill set as those that are probably further along may see this as very rudimentary, whilst those that just started out may be overwhelmed by this being the standard to judge projects. In either case, that’s totally fine! All this is meant to do is help you visualize and manage your expectations. So, if you know that certain things will just not be feasible due to how intensive a task will be, it may be worthwhile to try to reframe the scope of the project to something more manageable and in-line with prospective time line. Or, if you’re adamant about the project, just outsource some stuff **with having other collaborate with you** on the project.

OK, here’s the most dreaded part of all of this, setting deadlines. It’s always something that gave me a mini heart attack since it makes things “official”. Knowing this, it’s been my policy to keep a loose “deadline” with a ton of padded time in-between. While it’s important to keep an end-goal in mind, unexpected things always come up and the need to pivot will inevitably happen since things never go right the first time around. Plus, it’s always refreshing to be in a situation where you can “*under-sell and over-deliver*”, even if that’s only to yourself.

**“*OK, I’m ready to go. Any more tips you can give?*”**

Sure, there are a few that I can share that helped me get through a few of these:

* “**BE A PRODUCTIVE PROCRASTINATOR**”: I’ll have periods where I’m just not focused on the given task. So instead of just staring at a screen doing nothing for hours, just do something else that is beneficial in other facets of my life. This can be taking a break to regroup after a walk, do some outside reading that tangential to my learning, etc. As long as there is a benefit to you in some way, that’s a plus in my book.
* **GAMIFY THE BORING STUFF**: This is particularly notable during the data wrangling stage of any project where it’s basically just a matter of doing grunt work with reshaping the data or reformatting variables. I’ll set a personal quota to achieve and try to accomplish it for some kind of intrinsic or extrinsic reward for myself. However, DO NOT DO THIS FOR COMPLEX STUFF! Rushing it here = more time wasted to debug things.
* **“PROGRESS IS PROGRESS”**: It’s very easy to look at the lack of progress that you’ve made if you compare productivity on a per hour or per day basis. This will be a total bummer since there’s just too much fluctuation that can occur. However, with a long-game mindset, you can appreciate that any sort of progress (no matter how small) is meaningful and will help re-frame your thought process to be okay with some lull days.
* “**BE OKAY WITH CHANGE**”: Things rarely go right as originally thought in life, so we should probably expect the same with these projects. So instead of being super hung up about a particular aspect which is preventing you from progressing, I suggest that you be more open towards certain changes in order to have things completed instead of just having it sit there. Remember progress and action relies on momentum, if it stops, so does everything else.



Ok that’s all there is to it. If you’ve read this far and can follow some of the advice that I’ve written down, you’ll be in good shape to build out your portfolio. Now I know it’s easy to talk the talk, but what about walking the walk. Well, keep an eye out over the next coming week where I’ll be putting everything that I’ve outlined here in practice. So, make sure to check that out!

**Three Levels of Unguided Projects with Netflix Data Using R – Part 1**

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 most prolific streaming service, Netflix.

OK, let’s face it, Netflix is basically on the same tier as Amazon, Facebook and Google (i.e., they are taking over the world). They figured out understanding what we watch and how we 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 everything is more loosely restrictive, or is it? Perhaps the content is really skewed to favor a particular cohort over others and those that actually have a Netflix account (i.e., not borrowing a password) actually enjoy certain content. Well, since I have no idea what sort of content is on Netflix as a whole, I figured it might be a good idea to do some wrangling and exploration to see what it has to offer.

**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 use this [data set from Kaggle](https://www.kaggle.com/shivamb/netflix-shows) that contains about 7787 different titles. This has data of content ever since 2010 that contains both movie and TV show content. 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 PROCESS**

As the title suggests, the plan with this data set is to build THREE different projects out of it according to that 3-tier system that I mentioned previously. Since this is the first of three, I’ll be looking to build a Tier-1 project which means data wrangling and basic descriptive analysis. Considering that this would serve as the foundation for the other projects, it’s a good idea that I show this process.

Some of the libraries that I would be using over the course of these three projects will be:

* SKIMR – Used for a quick glance of the data set
* TIDYTEXT – I’ll be working with text data, so this makes the process easy to do when introducing stop words and filtering them out.
* TIDYVERSE – A collection of packages that makes the process of tidying data really easy
* SHINY – I’ll eventually be using this to make this interactive (will be using this in part 2 and 3)
* WORDCLOUD2 – We’ll be making word cloud at some point
* CLUSTER – Will be using this for cluster analysis

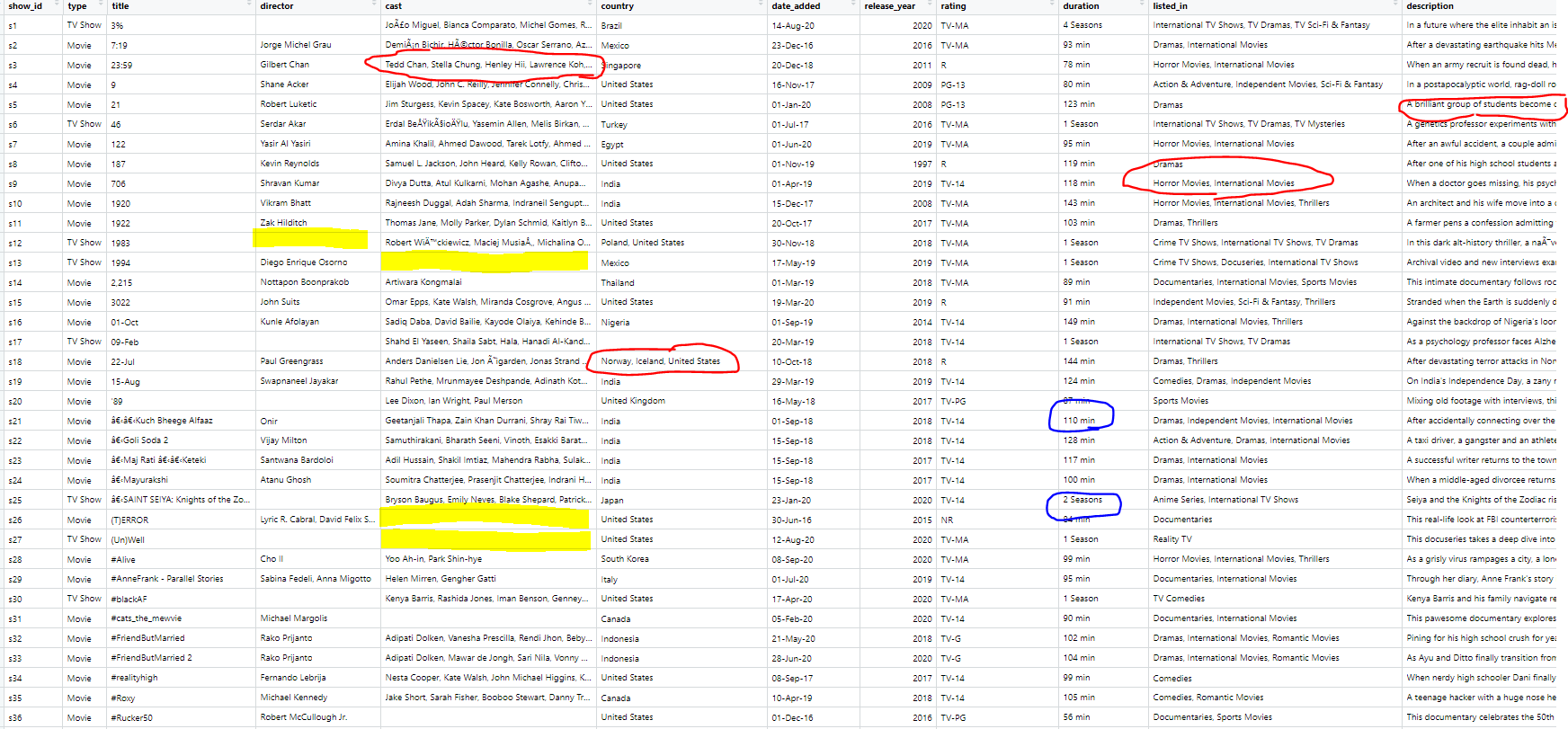
After reading in the data set and exploring it, you’ll notice that we come into a few issues:

```

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

View(netflix)

```



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 that

Overall, there’s actually quite a bit of wrangling that needs to be done before even going into the analysis. So, let’s go through this step-by-step.

**STEP 1: Dealing with the Blanks**

A quick way to deal with these blanks is to substitute this blank space with a NULL value which really is just good practice. Blanks aren’t the norm to handle missing values, we typically reserve this with either NA or NaN or some absurd value that has no real meaning (i.e., if scores are usually 1-100, using 999 is good to use to represent null values). In this case, I’ll replace these blanks with NAs.

```

netflix = netflix %>% mutate\_all(na\_if, "") # quick way to just replace all blanks in all columns with NAs

```



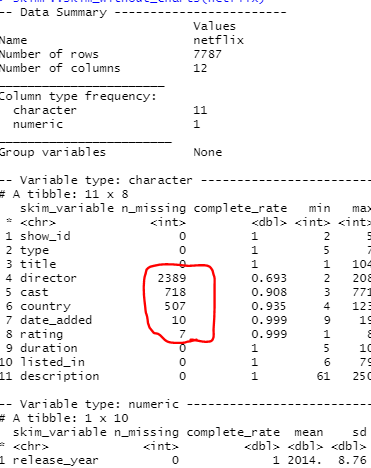
**STEP 2: Contending with Missing Values**

Using the skimr package, we can see a fair number of missing values now:

```

skimr::skim\_without\_charts(netflix)

```



We can see that there are a number of missing entries for different variables. Since I will be building two other projects from this data set, I would like to have as much data as possible so it I can reduce the degree of missingness here, that would be ideal. Considering that, it’s probably best to deal with this starting from lowest to highest.

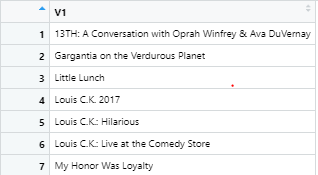
**RATING & DATE\_ADDED**

Looking here, it seems that there are 7 and 10 missing entries for rating and date\_added respectively. A quick Google Search for the answer along with imputation would really solve the trick here. It’s just a matter of which title has these missing values.

```

View(netflix$title[is.na(netflix$rating)])

```



The next process would be to just replace the missing content rating following a quick Google search:

```

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)))))))

)

```

You will just repeat the same process for the missing entries for date\_added.

**COUNTRY**

If we look at country, there’s over 500 missing values here. Doing a Google search is not a good use of my time in figuring this out. So how can we use some high IQ solutions for this? Well, just use existing data to help us out.   
  
One method I’ve used is to simply look at the “listed\_in” variable (*aka.* the genres). Here, I will look for language-specific listings which will help me piece out a country to put down. The two notable ones are: “British TV Shows” and “Korean TV Shows”. Using this I can just impute “*United Kingdom*” and “*South Korea*” respectively. To accomplish this, you’ll need to use the str\_detect() function.

```

netflix = netflix %>%

mutate(country =

ifelse(c(is.na(country) & (str\_detect(listed\_in, "[Kk]orea") == TRUE)), "South Korea",

ifelse(c(is.na(country) & (str\_detect(listed\_in, "[Bb]ritish") == TRUE)), "United Kingdom", country))

)

```

**NOTE**: There is one option that lists “Spanish-Language TV Shows”. Obviously, you can’t really use this as there’s several countries, including an entire continent, that speaks that language. So don’t use this!

To build on this, another solution we can use is to look at the titles and see notable franchise titles. An example of this is titles that contains “Monty Python” is notoriously a British film series or “Power Rangers” is the United States. Additionally, we can look for key words in the title that corresponds a given language to a country like Tamil + Hindi for India.

```

netflix = netflix %>%

mutate(

country =

ifelse(c(is.na(country) & str\_detect(title, "[Hh]indi") == T), "India",

ifelse(c(is.na(country) & str\_detect(title, "[Tt]amil") == T), "India",

ifelse(c(is.na(country) & str\_detect(title, "Power Rangers Super Megaforce") == T), "United States",

ifelse(c(is.na(country) & str\_detect(title, "Power Rangers Super Samurai") == T), "United States",

ifelse(c(is.na(country) & str\_detect(title, "Trailer Park Boys: Out of the Park: USA") == T), "United States",

ifelse(c(is.na(country) & str\_detect(title, "The Naked Gun 2 1/2: The Smell of Fear") == T), "United States",

ifelse(c(is.na(country) & str\_detect(title, "Power Rangers ") == T), "United States",

ifelse(c(is.na(country) & str\_detect(title, "Monty Python") == T), "United Kingdom",

ifelse(c(is.na(country) & str\_detect(title, "Calico Critters") == T), "United States",

ifelse(c(is.na(country) & str\_detect(title, "ChuChuTV") == T), "India",

ifelse(c(is.na(country) & str\_detect(title, "ChuChu TV") == T), "India",

ifelse(c(is.na(country) & str\_detect(title, "France") == T), "France",

ifelse(c(is.na(country) & str\_detect(title, "The Birth Reborn") == T), "Brazil", country)))))))))))))

)

```

Another solution is to simply utilize the common names of the cast to help figure out the missing countries. For instance, if you have the name “Aoi” = likely Japanese, “Arjun” or “Singh” = likely Indian, “Kevin Hart” = “American”. Additionally, we can look for notable actors and actresses that historically only ever work domestically to establish the country. For example, Beyoncé or Seth Rogen only ever done American content. On the other hand, Shah Rukh Khan has exclusively worked in Bollywood.

```

netflix = netflix %>%

mutate(

country =

ifelse(c(is.na(country) & str\_detect(cast, "[Aa]oi") == T), "Japan",

ifelse(c(is.na(country) & str\_detect(cast, "[Pp]atta.") == T), "Thailand",

ifelse(c(is.na(country) & str\_detect(cast, "Jackson") == T), "United States",

ifelse(c(is.na(country) & str\_detect(cast, "[Pp]rabh") == T), "India",

ifelse(c(is.na(country) & str\_detect(cast, "[Yy]ama") == T), "Japan",

ifelse(c(is.na(country) & str\_detect(cast, "Elyse Maloway") == T), "United States",

ifelse(c(is.na(country) & str\_detect(cast, "Michela Luci") == T), "Canada",

ifelse(c(is.na(country) & str\_detect(cast, "Singh") == T), "India",

ifelse(c(is.na(country) & str\_detect(cast, "Arjun") == T), "India",

ifelse(c(is.na(country) & str\_detect(cast, "[Hh]yun") == T), "South Korea",

ifelse(c(is.na(country) & str\_detect(cast, "Frank Grillo") == T), "United States",

ifelse(c(is.na(country) & str\_detect(cast, "Fred Armisen") == T), "United States",

ifelse(c(is.na(country) & str\_detect(cast, "Chris Rock") == T), "United States",

ifelse(c(is.na(country) & str\_detect(cast, "Kevin Hart") == T), "United States",

ifelse(c(is.na(country) & str\_detect(cast, "Oprah Winfrey") == T), "United States",

ifelse(c(is.na(country) & str\_detect(cast, "Quincy Jones") == T), "United States",

ifelse(c(is.na(country) & str\_detect(cast, "Iliza Shlesinger") == T), "United States",

ifelse(c(is.na(country) & str\_detect(cast, "Bert Kreischer") == T), "United States",

ifelse(c(is.na(country) & str\_detect(cast, "Killer Mike") == T), "United States",

ifelse(c(is.na(country) & str\_detect(cast, "Stephen Fry") == T), "United Kingdom",

ifelse(c(is.na(country) & str\_detect(cast, "[Rr]yoko") == T), "Japan", country)))))))))))))))))))))

)

```

Lastly, we can just rely on the description found in the missing cases and use mentioned countries or other key words to help out with placing a country.

```

netflix= netflix %>%

mutate(

country =

ifelse(c(is.na(country) & str\_detect(description, "Polish") == T), "Poland",

ifelse(c(is.na(country) & str\_detect(description, "Kuala") == T), "Malaysia",

ifelse(c(is.na(country) & str\_detect(description, "French") == T), "France",

ifelse(c(is.na(country) & str\_detect(description, "[Mm]exican") == T), "Mexico",

ifelse(c(is.na(country) & str\_detect(description, "Japan's top male") == T), "Japan",

ifelse(c(is.na(country) & str\_detect(description, "Brazilian") == T), "Brazil", country))))))

)

```

Everything else that’s totally unknown. I would just list it as “Unknown”. Again, that’s totally an option to begin with as I want to keep as much data as possible, but if I could piece together more usable data than just coping out with a generic answer, it reduces the capabilities for more advance projects.

```

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

```

**CAST & COUNTRY**

In the case of cast and country, you can’t really gain any insight from existing data to impute in these variables. Additionally, we also have to contend with the fact that some content legitimately doesn’t have credited cast nor directors since the content could be a documentary or reality TV show where such roles don’t matter. Since I really don’t want to waste hours of my life to do a Google check, along with manually enter these missing values, I’ll just impute this as having “Unknown/No Cast” or “Unknown/No Director(s)” for applicable variables.

```

netflix = netflix %>%

mutate(

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

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

)

```

**STEP 3: Addressing the different formatting in date\_added**

To address the issue of date formatting, I needed to find out all of the entries where the date entered differs from the majority (i.e., dd-mmm-yy). This turned out to all be in the following format: Month Date, Year. Taking those 70 or so dates, I just ended up doing some manual entry switching up the dates. The reason is that while the original plan was to use lubridate to help with the conversion, the issue with the month of May being similar on both sides may result in an incorrect identification and thus more work. So low-brain solution of manual entry was the best bet for me.

```

netflix = netflix %>%

mutate(date\_added =

ifelse(date\_added == " August 4, 2017", "04-Aug-17",

ifelse(date\_added == " December 23, 2018", "23-Dec-18",

ifelse(date\_added == " December 15, 2018", "15-Dec-18",

ifelse(date\_added == " July 1, 2017", "01-Jul-17",

ifelse(date\_added == " July 26, 2019", "26-Jul-19",

ifelse(date\_added == " May 26, 2016", "26-May-16",

ifelse(date\_added == " November 1, 2019", "01-Nov-19",

ifelse(date\_added == " December 2, 2017", "2-Dec-17",

ifelse(date\_added == " March 15, 2019", "15-Mar-19",

ifelse(date\_added == " October 1, 2019", "01-Oct-19",

ifelse(date\_added == " December 15, 2017", "15-Dec-17",

ifelse(date\_added == " July 1, 2017", "01-Jul-17",

ifelse(date\_added == " August 4, 2017", "04-Aug-17",

ifelse(date\_added == " April 4, 2017", "04-Apr-17",

ifelse(date\_added == " December 28, 2016", "28-Dec-16",

ifelse(date\_added == " March 31, 2018", "31-Mar-18",

ifelse(date\_added == " February 1, 2019", "01-Feb-19",

ifelse(date\_added == " January 1, 2018", "01-Jan-18",

ifelse(date\_added == " July 1, 2017", "01-Jul-17",

ifelse(date\_added == " February 24, 2018", "24-Feb-18",

ifelse(date\_added == " March 31, 2018", "31-Mar-18",

ifelse(date\_added == " July 20, 2018", "20-Jul-18",

ifelse(date\_added == " January 17, 2018", "17-Jan-18",

ifelse(date\_added == " September 7, 2016", "07-Sep-18", date\_added))))))))))))))))))))))))

```

**STEP 4: Separating out the text**

This step is actually not the worst this to handle as it’s just a matter of appropriately using the piping capabilities afforded from the tidyverse package, along with both separate() and pivot\_longer() functions that will perform the separation of text and stacking of each word for a given row, and the formation of new variables to indicate both the placement of each word from the original variable.

Looking at some of the entries of these text, it was found that the FIRST listed name corresponds to the lead in their respective role. So, the first listed actor/actress = headlining cast and the rest = supporting. Similarly, the first listed director = lead director whilst the rest are supporting or guest directors. Comparably, for genres and countries, the first listed one corresponds to the principal genre/country of origin.

**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() 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 5: 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.

**EXPLORATORY ANALYSIS**

A quick look at the data we see that the majority of the content on Netflix are movies, which includes documentaries.

```

chisq.test(netflix %>%

group\_by(type) %>%

summarise(count = n()) %>%

arrange(desc(count)) %>%

select(count))

netflix %>%

group\_by(type) %>%

summarise(count = n()) %>%

arrange(desc(count)) %>%

ggplot(

aes(x = "", y = count, fill = type)

) +

geom\_bar(stat = 'identity', color = 'black') +

scale\_fill\_manual(values = c("TV Show" = "firebrick2", "Movie" = "slateblue3")) +

geom\_text(

aes(label = paste0(round((count/sum(count))\*100, 2), "%")),

position = position\_stack(vjust = 0.5),

color = "white",

fontface = "bold",

size = 15

) +

coord\_polar(theta = 'y', start = 0) +

theme\_classic() +

theme(

axis.title = element\_blank(),

axis.line = element\_blank(),

axis.text = element\_blank(),

legend.text = element\_text(size = 24),

legend.title = element\_blank(),

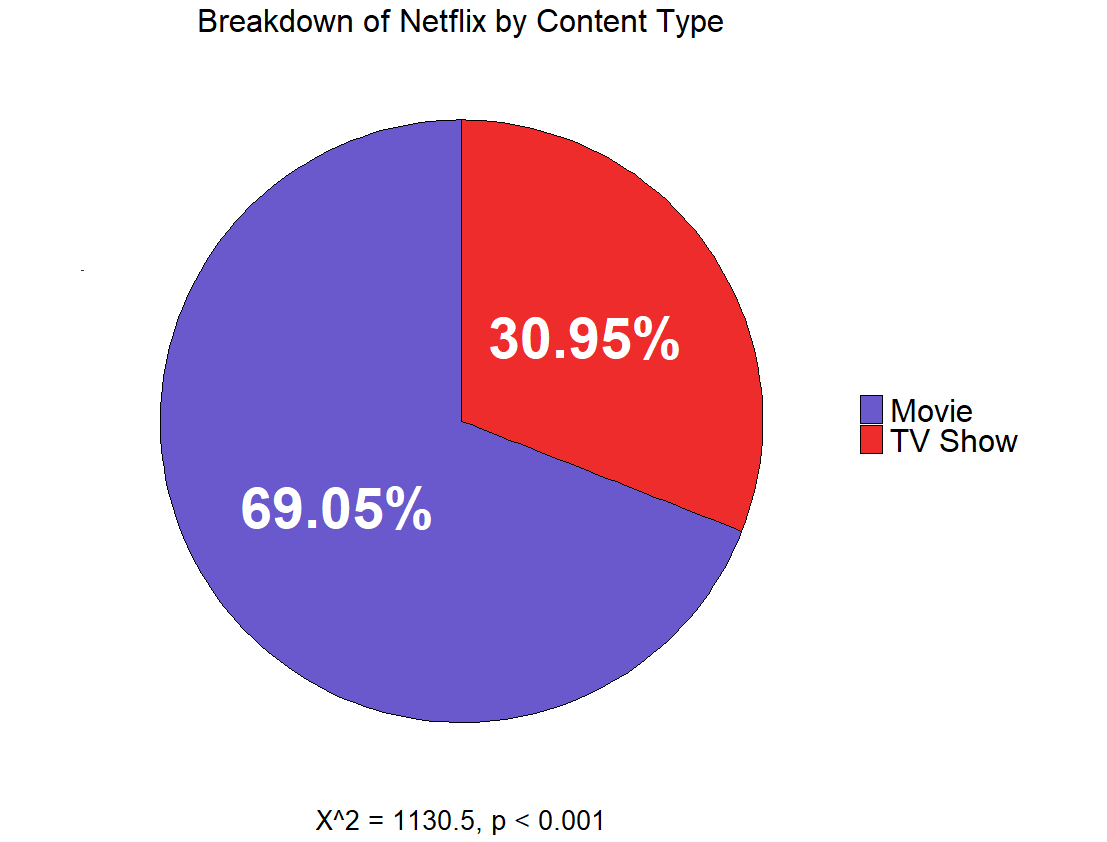
plot.title = element\_text(hjust = 0.5, size = 24),

plot.caption = element\_text(hjust = 0.5, size = 20)

) +

labs(title = "Breakdown of Netflix by Content Type", caption = "Χ^2 = 1130.5, p < 0.001")

```



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.

```

netflix\_genre\_split %>%

group\_by(genre, type, listing\_type) %>%

summarise(count = n()) %>%

arrange(desc(count)) %>%

ggplot(

aes(x = reorder(genre, count), y = count, fill = type)

) +

geom\_bar(stat = 'identity', color = 'black') +

theme\_classic() +

theme(

axis.text = element\_text(color = 'black'),

axis.title = element\_text(size = 20),

plot.title = element\_text(hjust = 0.5, size = 24),

legend.title = element\_blank(),

strip.text.x = element\_text(color = 'black', size = 15, face = 'bold'),

strip.text.y = element\_text(color = 'black', size = 15, face = 'bold'),

axis.text.x = element\_text(size = 15)

) +

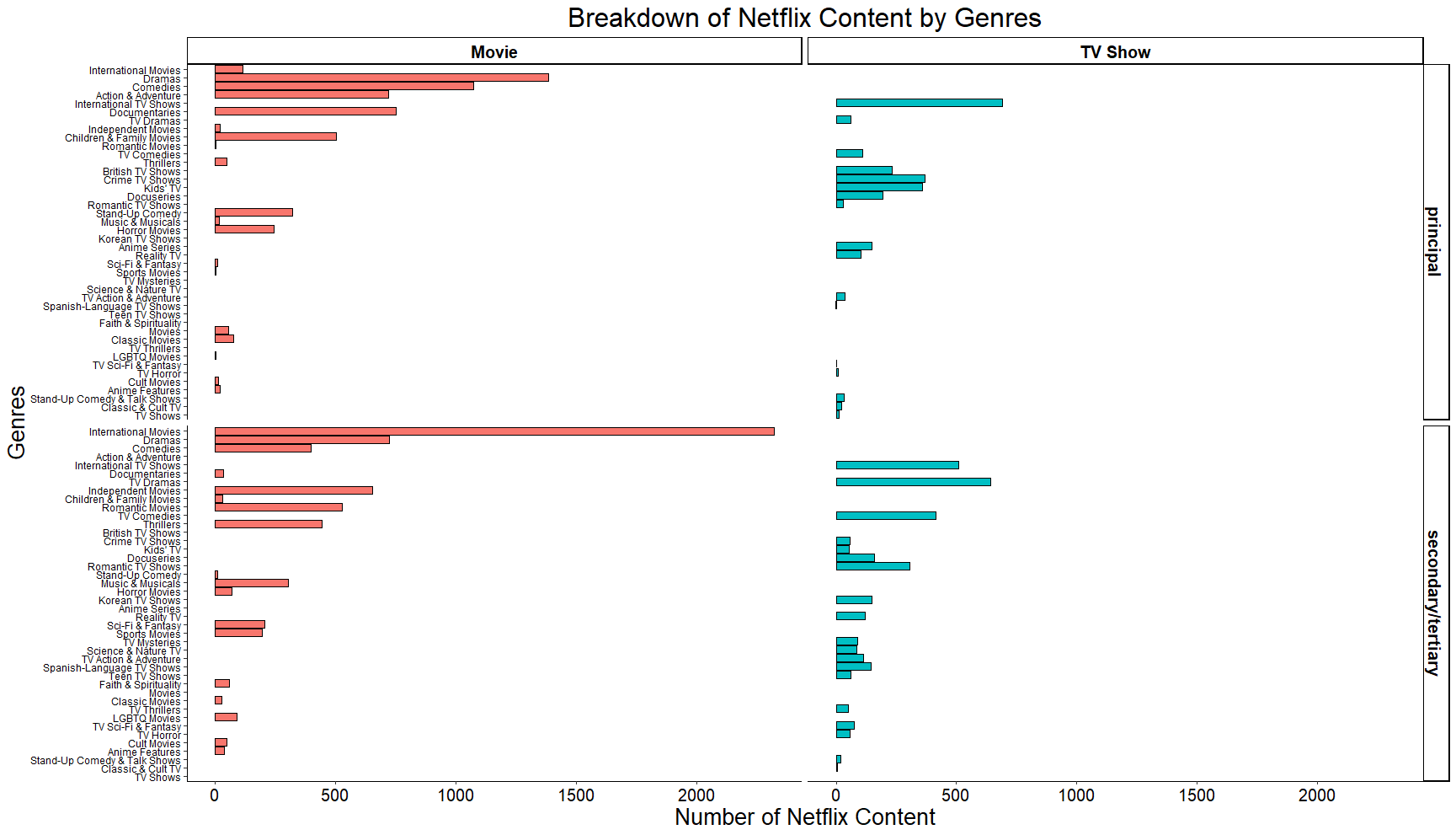
labs(x = "Genres", y = "Number of Netflix Content", title = "Breakdown of Netflix Content by Genres") +

coord\_flip() +

guides(fill = F) +

facet\_grid(cols = vars(type), rows = vars(listing\_type))

```



```

chisq.test(netflix\_country\_split %>%

mutate(

english\_or\_not =

ifelse(c(country\_type == "main country" & country\_name == "United States"), "English Speaking",

ifelse(c(country\_type == "main country" & country\_name == "United Kingdom"), "English Speaking",

ifelse(c(country\_type == "main country" & country\_name == "Canada"), "English Speaking",

ifelse(c(country\_type == "main country" & country\_name == "New Zealand"), "English Speaking",

ifelse(c(country\_type == "main country" & country\_name == "Australia"), "English Speaking",

ifelse(c(country\_type == "main country" & country\_name == "Ireland"), "English Speaking",

ifelse(c(country\_type == "main country" & country\_name == "Jamaica"), "English Speaking",

ifelse(c(country\_type == "main country" & country\_name == "Barbados"), "English Speaking",

ifelse(c(country\_type == "main country" & country\_name == "Belize"), "English Speaking",

ifelse(country\_type == "main country", "Non-English Speaking", NA))))))))))

) %>%

filter(!is.na(english\_or\_not)) %>%

group\_by(english\_or\_not) %>%

summarise(count = n()) %>%

arrange(desc(count)) %>%

select(count))

netflix\_country\_split %>%

mutate(

english\_or\_not =

ifelse(c(country\_type == "main country" & country\_name == "United States"), "English Speaking",

ifelse(c(country\_type == "main country" & country\_name == "United Kingdom"), "English Speaking",

ifelse(c(country\_type == "main country" & country\_name == "Canada"), "English Speaking",

ifelse(c(country\_type == "main country" & country\_name == "New Zealand"), "English Speaking",

ifelse(c(country\_type == "main country" & country\_name == "Australia"), "English Speaking",

ifelse(c(country\_type == "main country" & country\_name == "Ireland"), "English Speaking",

ifelse(c(country\_type == "main country" & country\_name == "Jamaica"), "English Speaking",

ifelse(c(country\_type == "main country" & country\_name == "Barbados"), "English Speaking",

ifelse(c(country\_type == "main country" & country\_name == "Belize"), "English Speaking",

ifelse(country\_type == "main country", "Non-English Speaking", NA))))))))))

) %>%

filter(!is.na(english\_or\_not)) %>%

group\_by(english\_or\_not) %>%

summarise(count = n()) %>%

arrange(desc(count)) %>%

ggplot(

aes(x = "", y = count, fill = english\_or\_not)

) +

geom\_bar(stat = 'identity', color = 'black') +

scale\_fill\_manual(values = c("English Speaking" = "firebrick2", "Non-English Speaking" = "slateblue3")) +

geom\_text(

aes(label = paste0(round((count/sum(count))\*100, 2), "%")),

position = position\_stack(vjust = 0.5),

color = "white",

fontface = "bold",

size = 15

) +

coord\_polar(theta = "y", start = 0) +

theme\_classic() +

theme(

axis.title = element\_blank(),

axis.line = element\_blank(),

axis.text = element\_blank(),

legend.text = element\_text(size = 24),

legend.title = element\_blank(),

plot.title = element\_text(hjust = 0.5, size = 24),

plot.caption = element\_text(hjust = 0.5, size = 20)

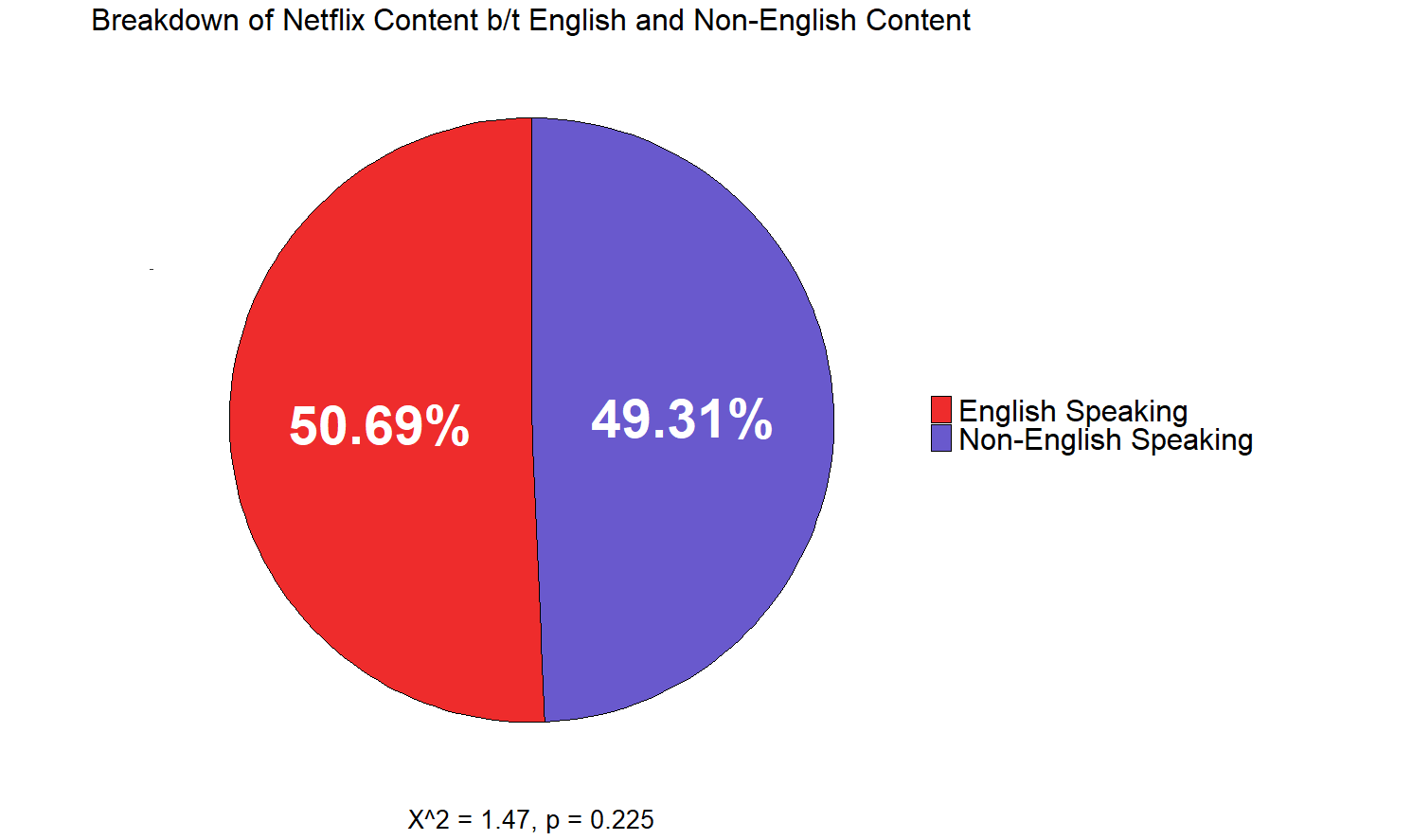
) +

labs(title = "Breakdown of Netflix Content b/t English and Non-English Content",

caption = "X^2 = 1.47, p = 0.225")

```

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



```

chisq.test(netflix %>%

group\_by(rating) %>%

summarise(count = n()) %>%

arrange(desc(count)) %>%

select(count))

netflix %>%

group\_by(rating) %>%

summarise(count = n()) %>%

arrange(desc(count)) %>%

ggplot(

aes(x = reorder(rating, count), y = count, fill = rating)

) +

geom\_bar(stat = 'identity', color = 'black') +

theme\_classic() +

labs(y = "Number of Netflix Content",

x = "Content Type",

title = "Breakdown of Netflix Content by Content Ratings",

caption = "X^2 = 16393, p < 0.001") +

guides(fill = F) +

theme(

axis.text = element\_text(color = 'black', size = 15),

axis.title = element\_text(color = 'black', size = 15),

strip.text = element\_text(color = 'black', size = 15),

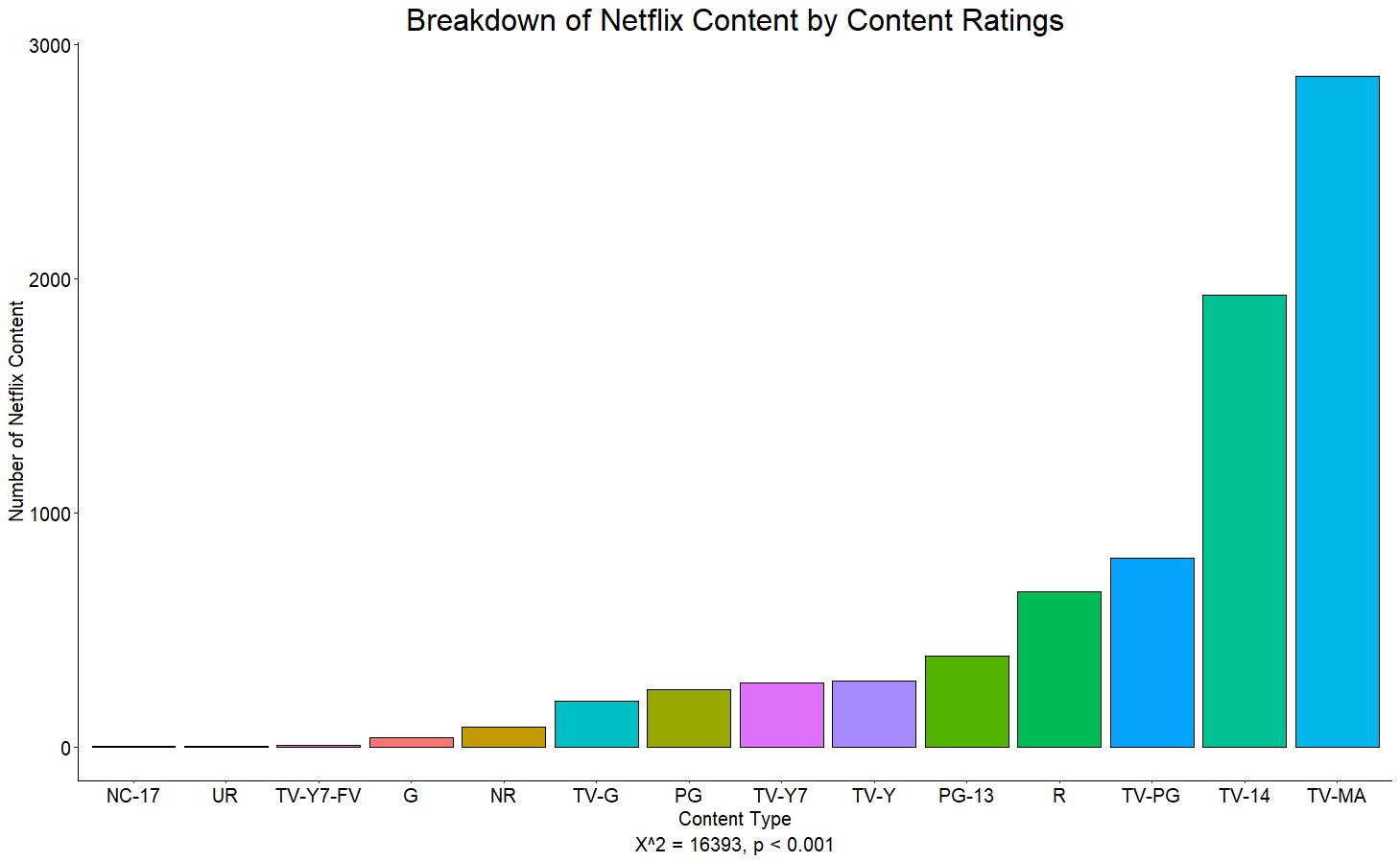
plot.title = element\_text(color = "black", size = 24, hjust = 0.5),

plot.caption = element\_text(color = 'black', size = 15, hjust = 0.5)

)

```

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



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.

```

netflix\_countryxcast\_df %>%

filter(cast\_type == "headliner") %>%

filter(cast != "Unknown/No Cast") %>%

group\_by(english\_or\_not, cast) %>%

summarise(count = n()) %>%

arrange(desc(count)) %>%

top\_n(3) %>%

ggplot(

aes(x = reorder(cast, count), y = count, fill = english\_or\_not)

) +

geom\_bar(

stat = 'identity', color = 'black'

) +

scale\_fill\_manual(values = c("English Speaking" = "firebrick2", "Non-English Speaking" = "slateblue3")) +

guides(fill = F) +

labs(x = "Headlining Cast",

y = "Number of Netflix Content",

title = "Top 3 Headlining Cast Member Based on Number of Netflix Credits",

caption = "\*Excluding non-credited content") +

theme\_classic() +

coord\_flip() +

facet\_wrap(vars(english\_or\_not)) +

theme(

axis.text = element\_text(color = 'black', size = 13),

axis.title = element\_text(color = 'black', size = 15),

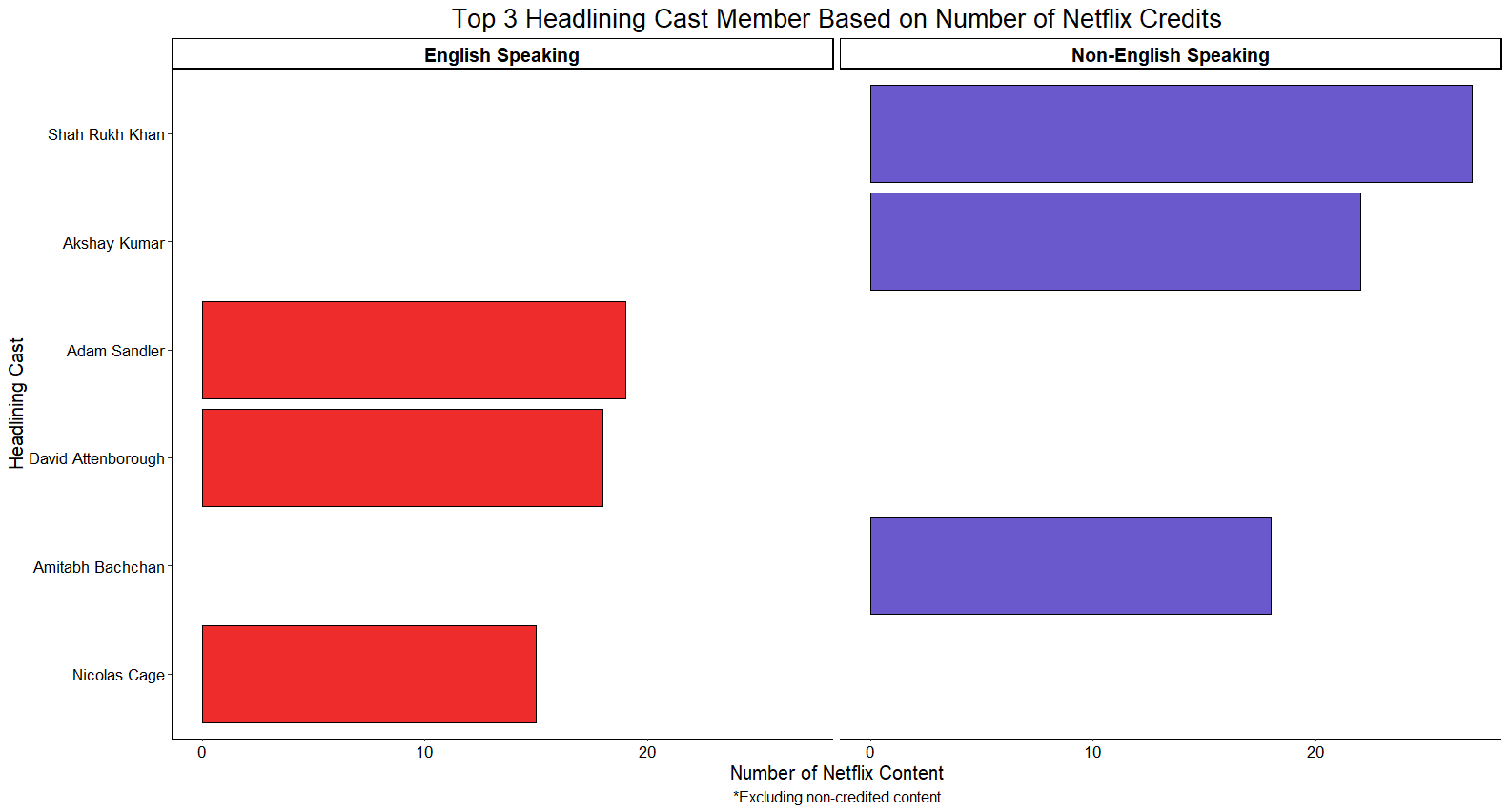
plot.title = element\_text(color = 'black', size = 20, hjust = 0.5),

plot.caption = element\_text(color = 'black', hjust = 0.5, size = 12),

strip.text = element\_text(color = 'black', face = 'bold', size = 15)

)

```



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.

```

netflix\_countryxdirector\_df %>%

filter(director\_type == "lead") %>%

filter(director\_name != "Unknown/No Director(s)") %>%

group\_by(english\_or\_not, director\_name) %>%

summarise(count = n()) %>%

arrange(desc(count)) %>%

top\_n(3) %>%

ggplot(

aes(x = reorder(director\_name, count), y = count, fill = english\_or\_not)

) +

geom\_bar(

stat = 'identity', color = 'black'

) +

scale\_fill\_manual(values = c("English Speaking" = "firebrick2", "Non-English Speaking" = "slateblue3")) +

guides(fill = F) +

labs(x = "Lead Director",

y = "Number of Netflix Content",

title = "Top 3 Lead Director Based on Number of Netflix Credits",

caption = "\*Excluding non-credited content") +

theme\_classic() +

coord\_flip() +

facet\_wrap(vars(english\_or\_not)) +

theme(

axis.text = element\_text(color = 'black', size = 13),

axis.title = element\_text(color = 'black', size = 15),

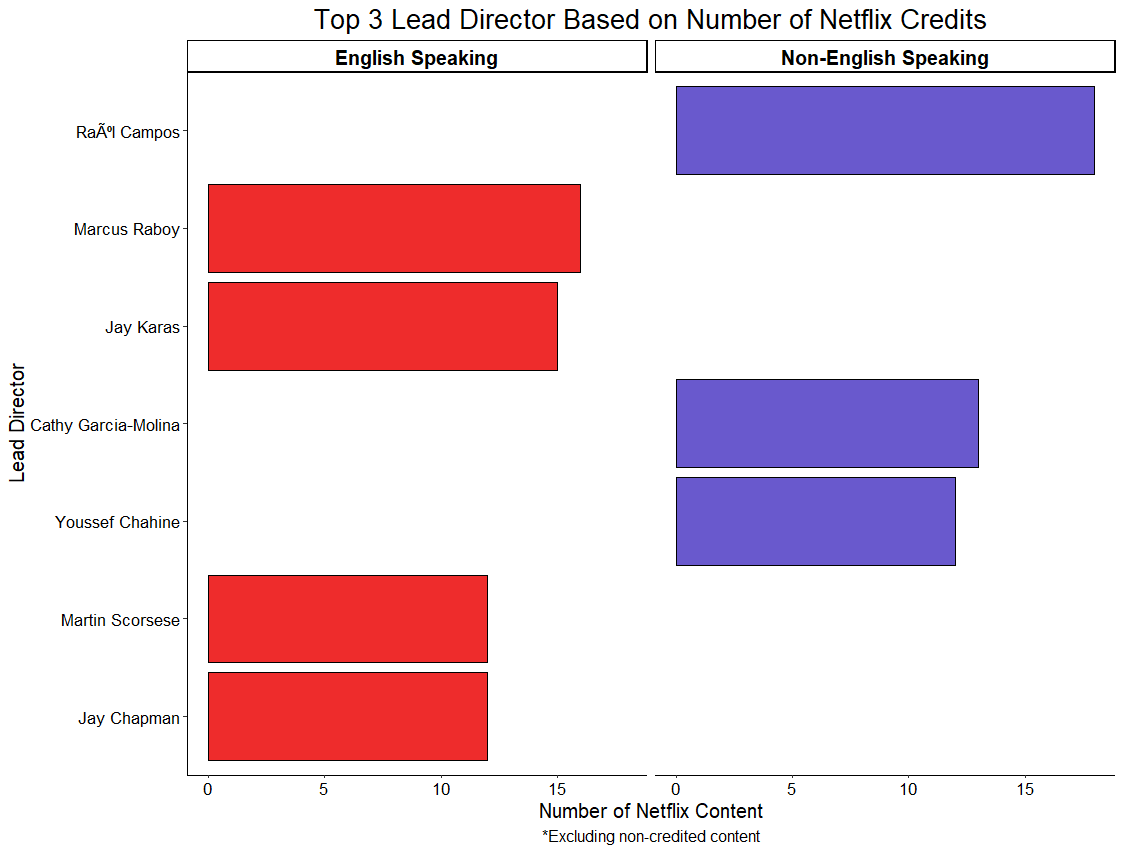
plot.title = element\_text(color = 'black', size = 20, hjust = 0.5),

plot.caption = element\_text(color = 'black', hjust = 0.5, size = 12),

strip.text = element\_text(color = 'black', face = 'bold', size = 15)

)

```



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.

```

wordcloud2(

data = netflix\_descriptionxlanguage %>%

filter(!is.na(english\_or\_not)) %>%

filter(english\_or\_not == "English Speaking") %>%

group\_by(keywords) %>%

summarise(count = n()) %>%

arrange(desc(count)) %>%

top\_n(100),

minSize = 0.6,

minRotation = 0,

backgroundColor = "orange")

```



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

```

wordcloud2(

data = netflix\_descriptionxlanguage %>%

filter(!is.na(english\_or\_not)) %>%

filter(english\_or\_not == "Non-English Speaking") %>%

group\_by(keywords) %>%

summarise(count = n()) %>%

arrange(desc(count)) %>%

top\_n(100),

minSize = 0.6,

minRotation = 0,

backgroundColor = "orange")

```



**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 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 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.

Thanks for the read.