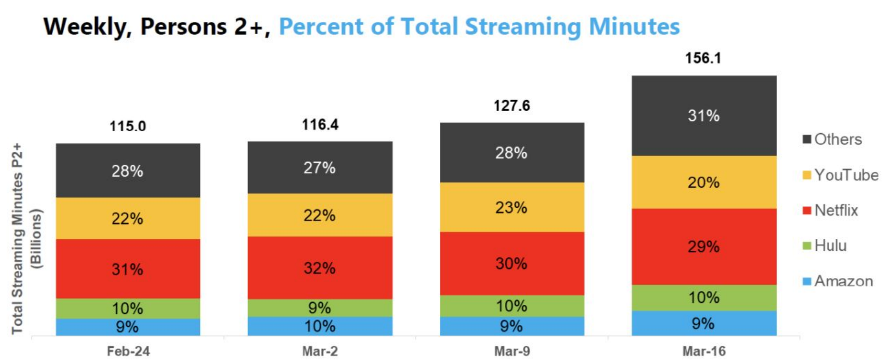
Streaming Platform Twitter Trends in the Covid-19 Pandemic: Final Report

Authors: Jennifer Han, Christina DaSilva
IST 652 - Scripting for Data Analysis
Section: Wednesday 7 pm Eastern Time

Overview

In early 2020, the Covid-19 pandemic drastically altered people's lives around the world. With many staying at home in efforts to slow the spread of the virus, daily habits and behaviors also changed. We are interested in learning about trends in movie or television show streaming and well-being which emerged during this time period, reflected by Twitter chatter. According to a Nielsen report, "American consumers spent an estimated 400 billion minutes streaming content to their televisions over the first three weeks of March, an 85 percent increase from the same period March 29."[1]

We are also interested in analyzing the volume of Twitter conversations about the 3 major streaming platforms. From the Nielsen report, "Netflix has the biggest share of video streaming on television at 29 percent, followed by YouTube with 20 percent, Hulu at 10 percent, and Amazon Prime Video snagging 9 percent."[2] We are curious to see if the volume of tweets that mention these 3 major streaming platforms reflect the varying degrees of engagement users have with the streaming platforms' content.



Source: Nielsen

There are other streaming platforms that have gained in prominence as well, namely Disney+ and Apple TV+, but we have deemed these out of scope for this set of analyses.

[1] https://www.indiewire.com/2020/03/tv-streaming-march-2020-increase-nielsen-report-1202221796/
[2] https://www.indiewire.com/2020/03/tv-streaming-march-2020-increase-nielsen-report-1202221796/

Data Sources

There are four data sources we plan to use:

1. 2020 World Happiness Report
    a. Dimensions: 20 columns, 154 rows
    b. Type: Structured
    c. Source: https://www.kaggle.com/mathurinache/world-happiness-report?select=2020.csv
2. World Cities Data
    a. Dimensions: 11 columns, 26,570 rows
    b. Type: Structured
    c. Source: https://simplemaps.com/data/world-cities
3. US Cities Data
    a. Dimensions: 17 columns, 28,339 rows
    b. Type: Structured
    c. Source: https://simplemaps.com/data/us-cities
4. Twitter:
    a. Scraping conditions:
        - Created between Jan 1, 2020 to Dec 31, 2020
        - Terms “netflix”, “prime video”, “primevideo”, or “hulu” are in the tweet text content
        - Within a 10km radius of a list of cities
    b. World Cities Tweets Dataset Dimensions: 37 columns, 45,325 rows
    c. US Cities Tweets Dataset Dimensions: 37 columns, 107,293 rows
    d. Type: Unstructured
    e. Source: Specifically the twint library https://github.com/twintproject/twint

We use unstructured data from Twitter scraping with the twint library and structured data from the 2020 World Happiness Report for this project. Twitter data can tell us patterns in what people are talking about and what these trends look like. The Happiness Report data may reveal how these trends may be loosely related to overall happiness scores of various populations.

Data Cleansing and Pre-Processing

Here is are the specific details and steps we used for collecting the data from the data sources mentioned above:

2020 Happiness Report Dataset:

- Read in the happiness 2020 report from a csv file format and stored it as a pandas dataframe

Twitter Scraping with Twint:

- In order to scrape Twitter, we first needed to narrow the scope down and set geographic boundaries for scraping Twitter data.
- For both the world capital cities' tweets scraping and the US cities tweets scraping, we used the following parameters for scraping:
    - Created between Jan 1, 2020 to Dec 31, 2020
    - Where the terms "netflix", "prime video", "primevideo", or "hulu" show up anywhere in the tweet text content
        - We include both "prime video" and "primevideo" because both Netflix's username and Hulu's username are 1 word, whereas Prime Video's username is "@primevideo". Thus, we didn't want to unfairly scrape less tweets that @ mention "@primevideo" and also "prime video" in the content of the tweet. Had we not done both "prime video" and "primevideo" as part of the search terms, Amazon Prime Video's tweet volume may have been unfairly represented as having lower volume.
    - Within a 10km radius of the list of cities (see how the cities were defined)
        - We decided on a 10km radius by researching several major cities and their average city size radius was somewhere around 10km. We recognize some cities are more dense than others. While this is not a perfect representation of a city's boundaries, it is indeed an estimation.
- For world tweet data from the top 50 largest world capital cities by population:
    - Read in a dataset for US cities data from a csv file. (Website source: https://simplemaps.com/data/world-cities)
    - Stored the data in a pandas dataframe

    - Pulled only the rows of the dataframe where the 'capital' column = 'primary', meaning that this city is the primary capital city for the country
    - Dropped duplicates in the dataframe, since some of the countries had more than 1 primary capital city listed. We used the first primary capital city listed as the one to represent the country in this case.
    - Sorted the dataset by the population column with largest capital cities listed first
    - We scraped tweets from only the world's top 50 largest capital cities by population size from this dataset
- For US tweet data from the top 50 largest US cities by population:
    - Read in a dataset for US cities data from a csv file. (Website source: https://simplemaps.com/data/us-cities)
    - Stored the data in a pandas dataframe
    - Sorted the dataset by the population column with largest cities listed first
    - We scraped tweets from only the top 50 largest US cities by population size from this dataset

Joining Datasets:

- Joining 2020 Happiness Report dataset with the World Capital Cities dataset:
    - We join these 2 datasets on the Country Name fields (called 'Country name' in the 2020 Happiness Report dataset, and called 'country' in the capital cities dataset)
    - Some data cleansing was required before performing this join because some of the country names are not a complete match. Example: "South Korea" in one dataset, and "Korea, South" in the other dataset.
    - We also did some data cleansing for countries that were recognized as a country entity in one dataset, but not the other. This is typically for politically disputed territories. Examples: Hong Kong, Palestinian Territories, etc.
- To perform a good portion of our analyses, we needed to join the scraped tweets dataset with the cities datasets (whether for world capital cities or US cities):
    - Because the scraped tweets are only tracking the Geo column, in the format "lat,lng,radius", we needed to first extract this column and find its matching city name.
    - Created a new column in the scraped tweets datasets for the city name.

- ○ Now, we are able to join the scraped tweets datasets with the cities datasets by joining on the ‘city_ascii’ column defined in both datasets.

Known Data Limitations

There are some known limitations that we have identified and want to explicitly outline.

1. Using a country’s capital as a representative for the country overall:

We do not scrape all tweets from a given country. In the analysis for correlating a country’s happiness index score to their Twitter behavior patterns for streaming platforms, we are using the country’s capital city as a representative for the country’s Twitter behavior. We made this choice in order to A) limit the sheer volume of tweets to scrape since it would be a huge task to scrape all tweets from an entire country, and B) to adhere to the twint Python library’s constraints, which can take in a city’s geographic coordinates and radius as a parameter. It cannot take in a country name as a parameter, and defining all the geographic coordinates within a city’s abnormally shaped surface area would be a much larger task, hence we scoped it out of this project.

2. Limited number of Twitter users turn on geolocation for tweets:

We recognize that the tweet data we are scraping based on the geographic coordinates parameter is actually only a small portion of all tweets that include the keywords “netflix”, “prime video”, “primevideo”, and “hulu”. According to a research paper published by the National Institutes of Health, “approximately 0.85% of tweets are geotagged, meaning that the exact position of where the tweeter was when the tweet was posted is recorded using longitude and latitude measurements.”[3] While this proportion may seem small, this subset of tweets are actually over 4 million tweets every 24 hours, using an estimate of 500 million tweets per day.[4] According to this study performed by Luke Sloan and Jeffrey Morgan, there does seem to be a difference in users who turn on geolocation and those who don’t. They claim that “the biggest differences in geolocation-based behavior are related to language–both of the user interface and the tweet,” while behavioral differences related to gender and age tend to be small.[5] Thus, we want to acknowledge the potential biases in the datasets, and ensure these are kept top-of-mind during the following analyses and conclusions sections.

---

[3] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4636345/
[4] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4636345/
[5] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4636345/

Methods of Analysis

Here are the 4 primary analyses we performed for this report:

ANALYSIS #1:

Research Question: What were the time series and demographic trends of tweets mentioning the 3 streaming platforms (Netflix, Amazon Prime, Hulu) during 2020 in the top 50 world capital cities?

- Type of analysis: Summary statistics to identify world tweeting trends (details to follow)
  - Fields of the data used (found below for each sub-analysis)
  - Output (found below for each sub-analysis)
- Number of world tweets in 2020 by each streaming platform, aggregated by:
  - World Tweets Data Set
    - Platform mentioned in tweet
    - Country
    - Tweet ID
  - Output: Histogram plot (utilizing seaborn and matplotlib python packages)
- Top five countries with the most tweets, by each platform, aggregated by:
  - World Tweets Data Set:
    - Platform mentioned in tweet
    - Country
    - Tweet ID
  - Output: Dataframe and strip plot (utilizing pandas, seaborn and matplotlib python packages, respectively)
- Total number of world tweets in 2020 by month, aggregated by:
  - World Tweets Data Set:
    - Platform mentioned in tweet
    - Country
    - Tweet ID
    - Date of tweet (used to derive month of tweet)
  - Output: Histogram plot (utilizing seaborn and matplotlib python packages)

- Number of world tweets in 2020 by month and platform, aggregated by:
    - World Tweets Data Set:
        - Date of tweet (used to derive month of tweet)
    - Output: Histogram plots for each streaming platform (utilizing seaborn and matplotlib python packages)
- Number of tweets in 2020 by country and by month, aggregated by:
    - World Tweets Data Set:
        - Platform mentioned in tweet
        - Country
        - Date of tweet (used to derive month of tweet)
    - Output: A grid of histogram plots, one for each country (utilizing seaborn and matplotlib python packages)
- World tweets by region and happiness rating, aggregated by:
    - World Tweets Data Set:
        - Country (used to merge the two data sets)
    - World Happiness Ratings Data Set:
        - Ladder score
        - Regional indicator
        - Country (used to merge the two data sets)
    - Output: Scatterplot plot (utilizing seaborn and matplotlib python packages)

ANALYSIS #2:

Research Question: What were the time series and demographic trends of tweets mentioning the 3 streaming platforms (Netflix, Amazon Prime, Hulu) during 2020 in the top 50 US cities?

- Type of analysis: Summary statistics to identify tweeting trends in US cities (details below)
    - Fields of the data used (found below for each sub-analysis)
    - Output (found below for each sub-analysis)

- Total number of US tweets in 2020 by month, aggregated by:
    - US Cities Tweets Data Set:
        - Platform mentioned in tweet
        - Tweet ID
        - Date of tweet (used to derive month of tweet)
    - Output: Histogram plot (utilizing seaborn and matplotlib python packages)

- Number of US tweets in 2020 by month and platform, aggregated by:
    - US Cities Tweets Data Set:
        - Date of tweet (used to derive month of tweet)
    - Output: Histogram plots for each streaming platform (utilizing seaborn and matplotlib python packages)

- Number of tweets in 2020 by US city and by month, aggregated by:
    - US Cities Tweets Data Set:
        - Platform mentioned in tweet
        - US city
        - Date of tweet (used to derive month of tweet)
    - Output: A grid of histogram plots, one for each country (utilizing seaborn and matplotlib python packages)

ANALYSIS #3:

Research Question: Is there a relationship between a country's capital city tweet data mentioning the 3 streaming platforms and their country's overall happiness index report?

Type of analysis: Correlation analysis

Fields of the data used:

- Ladder score
- population
- Logged GDP per capita
- Social support
- Healthy life expectancy

- Freedom to make life choices
- Generosity
- Perceptions of corruption
- likes_count
- retweets_count
- platform_Amazon Prime
- platform_ Hulu
- platform_Netflix

Output: Correlation analysis table where the table values are R correlation coefficient values, and the rows and columns are the variables (field names) in the dataset

ANALYSIS #4:

Research Question: What are the top keywords used in tweets mentioning the 3 streaming platforms from the top 50 US cities?

Type of analysis: Tokenization and frequency distribution
Fields of the data used:

- Tweets text field - which we use to calculate new fields in a dataframe: token (word) and frequency

Output: Table of the token (word) and frequency, and a word cloud visualization

Overview of the Program

Here is an overview of our program:

- Read in the 2020 World Happiness Report data and store as a dataframe
- Read in the World Cities data and store as a dataframe. Extract the capital cities and drop duplicates so there is only 1 capital city per country.
- Join the World Cities Capital data with the 2020 World Happiness Report data on the country name column. Clean up the country names to ensure clear matches are found for the country name data when performing the joins. Drop countries that are not recognized in both datasets. Store only the top 50 largest capital cities in the world by population size.
- Read in the US cities data and store in a dataframe. Store only the top 50 largest US cities by population size.

- Create a function to scrape Twitter based on a set of defined parameters
- Scrape tweets in the given parameters for the top 50 world capital cities and store as json files
- Scrape tweets in the given parameters for the top 50 US cities and store as json files
- Read in the scraped tweets from the world capitals by streaming platforms. Add a new column for each of the dataframes to denote the streaming platform. Aggregate the data frames into 1 dataframe for world capitals across all 3 streaming platforms.
- Repeat the same step outlined above, but for the tweets from the US cities.
- Extract the latitude and longitude data from the Geo column of the world cities tweets dataframe and the US cities tweets dataframe. Add a new column for the matching city name based on the geographic coordinates.
- Perform the 4 primary buckets of analyses mapped to our 4 research questions. Detailed analyses and conclusions outlined in the following section.

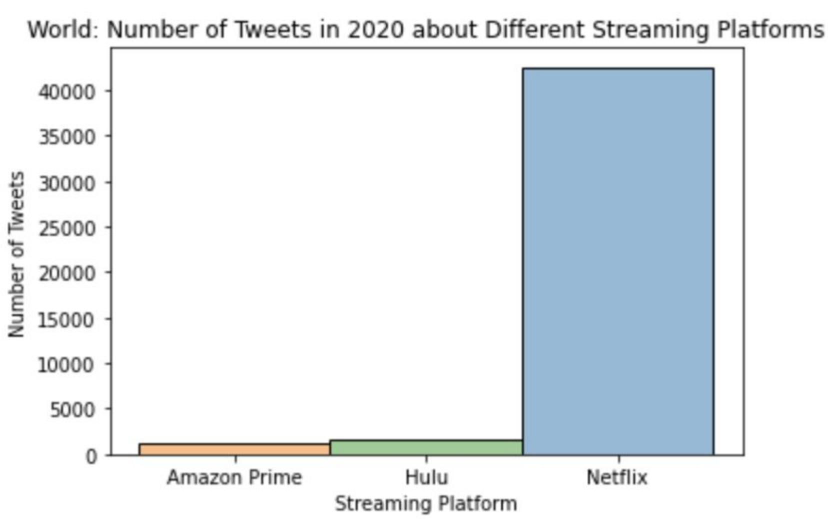
Results and Conclusions

ANALYSIS #1

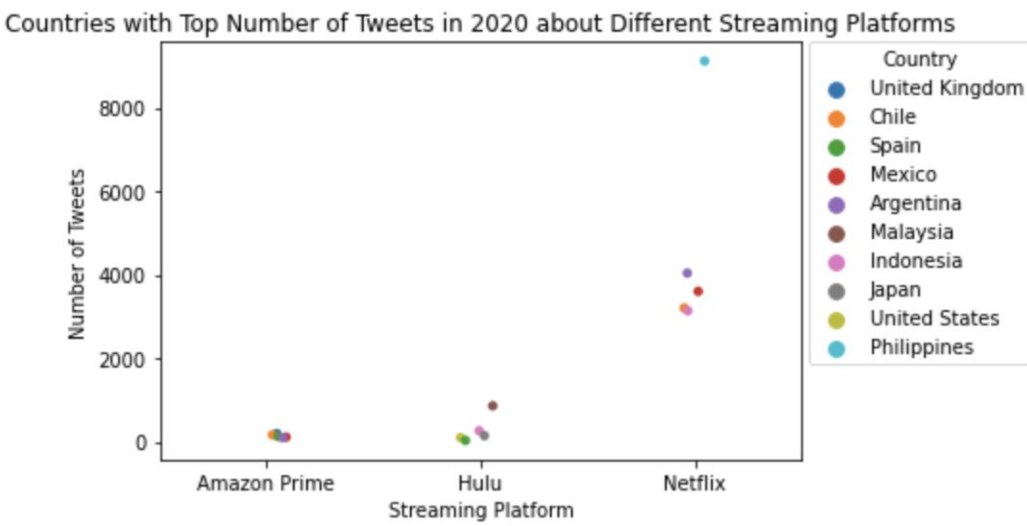
Research Question: What were the time series and demographic trends of tweets mentioning the 3 streaming platforms (Netflix, Amazon Prime, Hulu) during 2020 in the top 50 world capital cities?

Results:

Tweets from around the world about Netflix were more abundant than those about Amazon Prime and Hulu. These totals can be found in the figure, below:

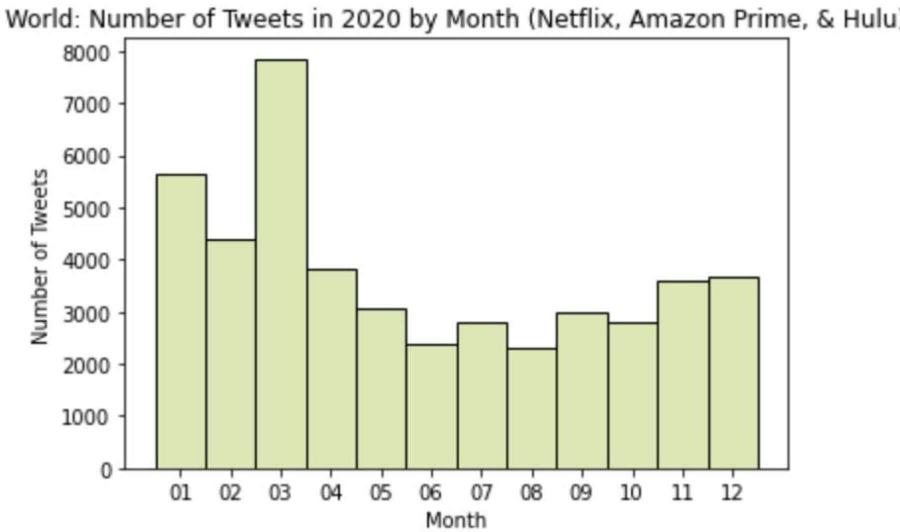


The countries that tweeted most about Netflix, Hulu, and Amazon Prime were Phillipines, Malaysia, and United Kingdom, respectively. The top five countries tweeting about these three streaming platforms can be found below:

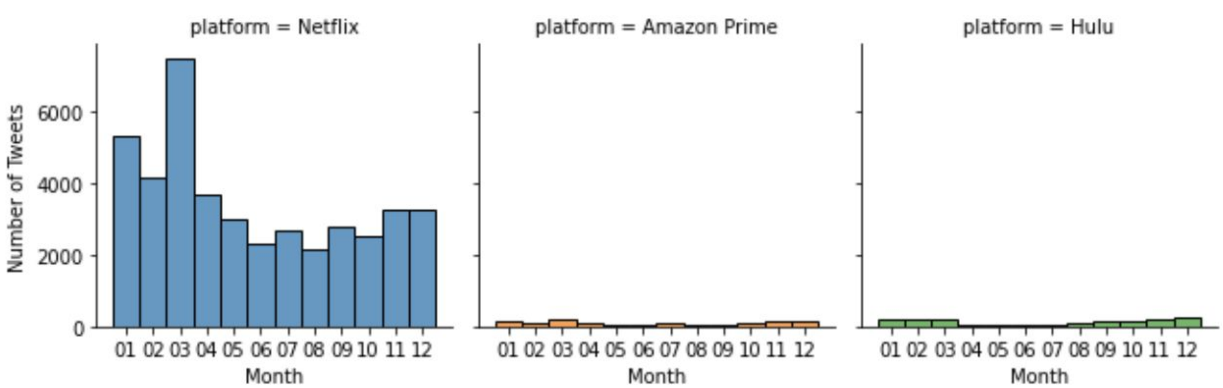


Platform	Country	Tweet Count
Amazon Prime	United Kingdom	197
Amazon Prime	Chile	164
Amazon Prime	Spain	123
Amazon Prime	Mexico	112
Amazon Prime	Argentina	101
Hulu	Malaysia	860
Hulu	Indonesia	263
Hulu	Japan	144
Hulu	United States	98
Hulu	Spain	31
Netflix	Philippines	9117
Netflix	Argentina	4038
Netflix	Mexico	3601
Netflix	Chile	3202
Netflix	Indonesia	3136

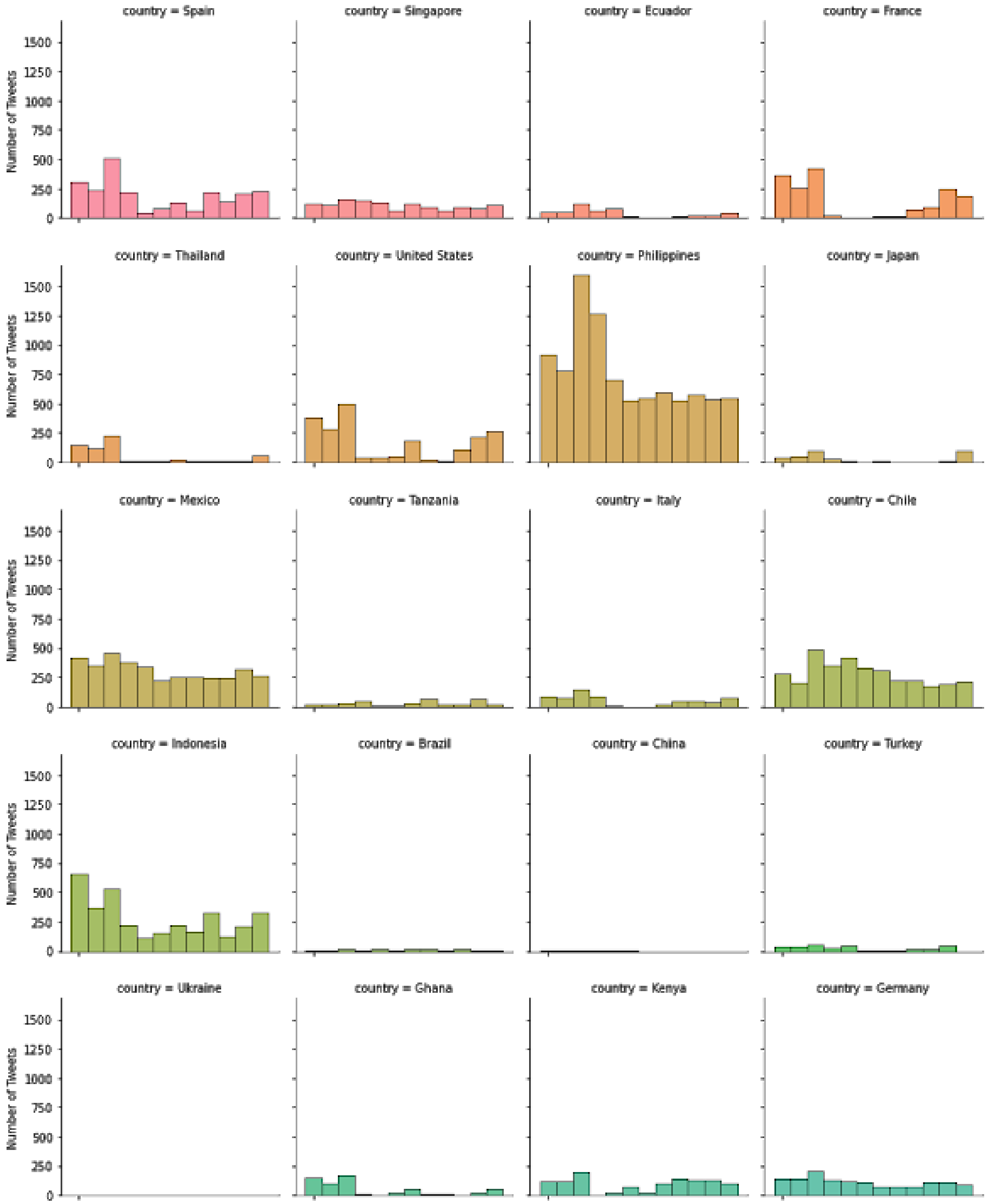
March 2020 contained the highest volume of tweets about streaming platforms. There is a notable increase from February 2020 to March 2020, as the pandemic began having a global impact. The number of tweets decrease into the summer months and begin to rise through the autumn and winter months.

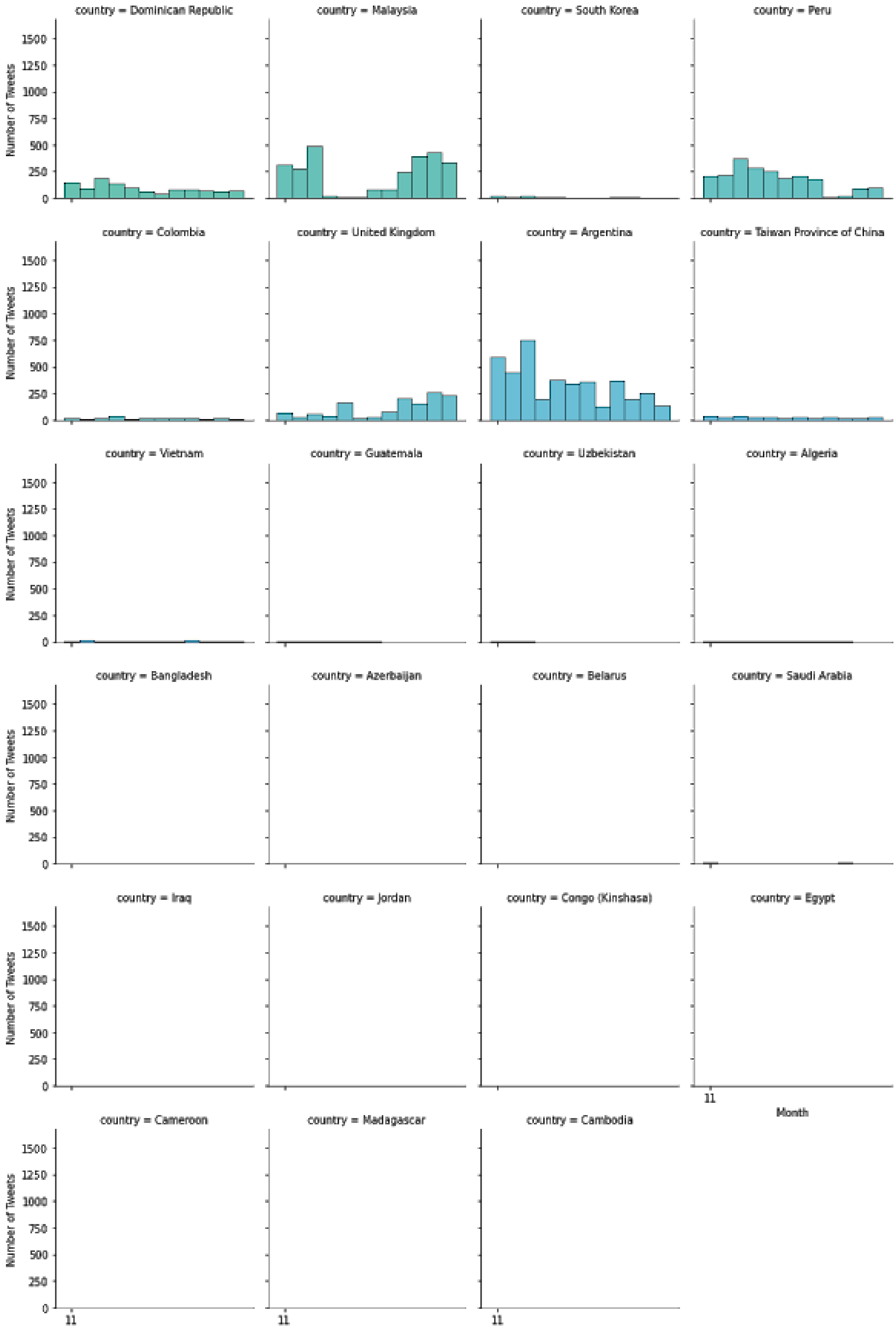


This trend is also seen when focusing on tweets specifically mentioning Netflix, as this was the platform most frequently tweeted about. Similar trends were found in tweets about Amazon Prime and Netflix, but at a much lower scale.

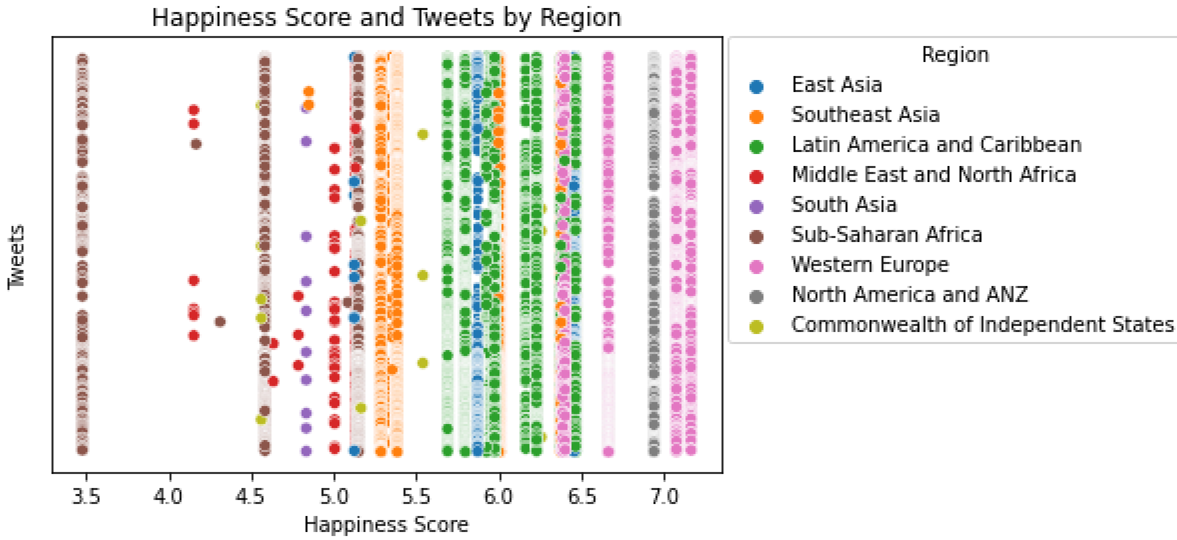


Similar trends are also seen when looking at the same set of tweets grouped by country by month. Most countries displayed a spike in the number of tweets about streaming platforms in March 2020, as found on the following two pages:


country = Spain
country = Singapore
country = Ecuador
country = France
country = Thailand
country = United States
country = Philippines
country = Japan
country = Mexico
country = Tanzania
country = Italy
country = Chile
country = Indonesia
country = Brazil
country = China
country = Turkey
country = Ukraine
country = Ghana
country = Kenya
country = Germany
Number of Tweets
0
250
500
750
1000
1250
1500


country = Dominican Republic
country = Malaysia
country = South Korea
country = Peru
country = Colombia
country = United Kingdom
country = Argentina
country = Taiwan Province of China
country = Vietnam
country = Guatemala
country = Uzbekistan
country = Algeria
country = Bangladesh
country = Azerbaijan
country = Belarus
country = Saudi Arabia
country = Iraq
country = Jordan
country = Congo (Kinshasa)
country = Egypt
country = Cameroon
country = Madagascar
country = Cambodia
Number of Tweets
0
250
500
750
1000
1250
1500
11
Month

People throughout the world tweeted about streaming platforms in 2020. The chart below shows tweets (by tweet ID) by the region, and happiness score from where the tweet originated. Tweet activity was fairly consistent across regions, despite happiness score.
The lowest happiness scores were seen in Sub-Sarahan Africa, and the highest in Western Europe.

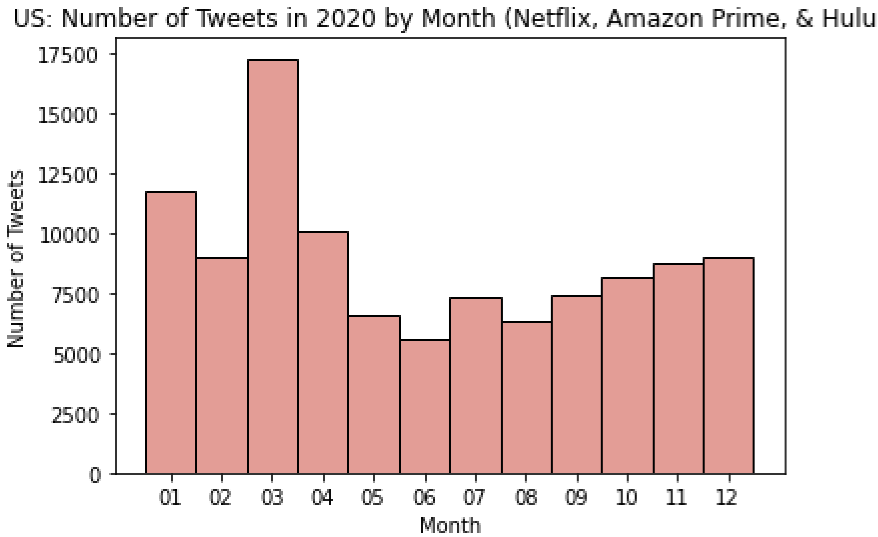


ANALYSIS #2

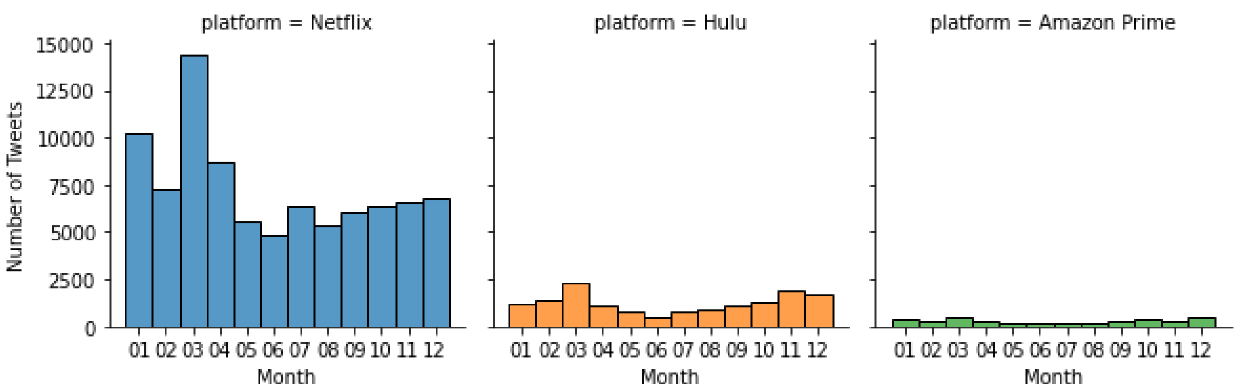
Research Question: What were the time series and demographic trends of tweets mentioning the 3 streaming platforms (Netflix, Amazon Prime, Hulu) during 2020 in the top 50 US cities?

Results:

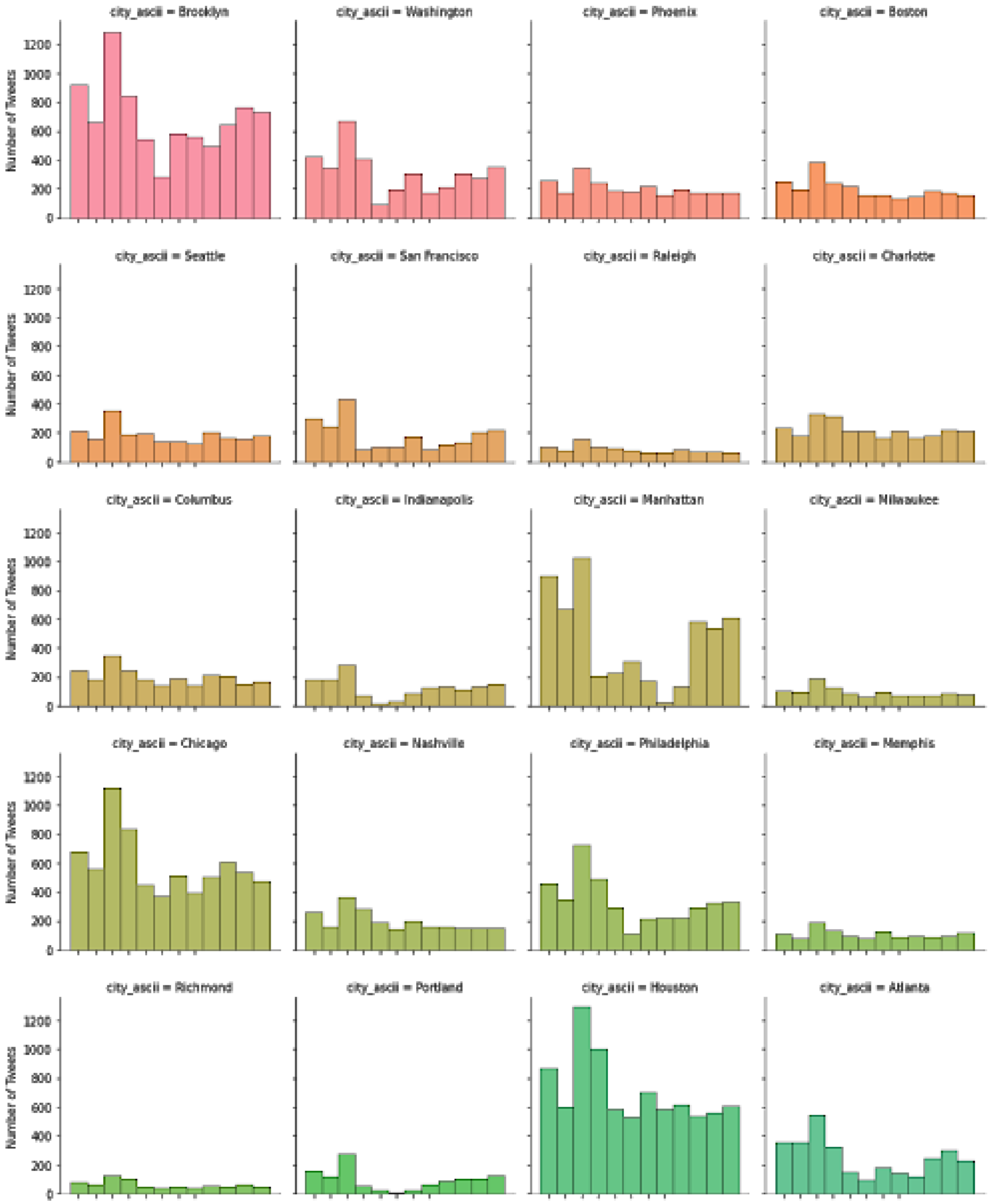
The trend seen in tweets from the US mentioning streaming platforms by month throughout 2020 mimics the trend seen in similar tweets around the world during the same time period; there was a drastic increase in these tweets in March 2020, which decreases during the summer months, and then begins to increase again with the fall and winter months.

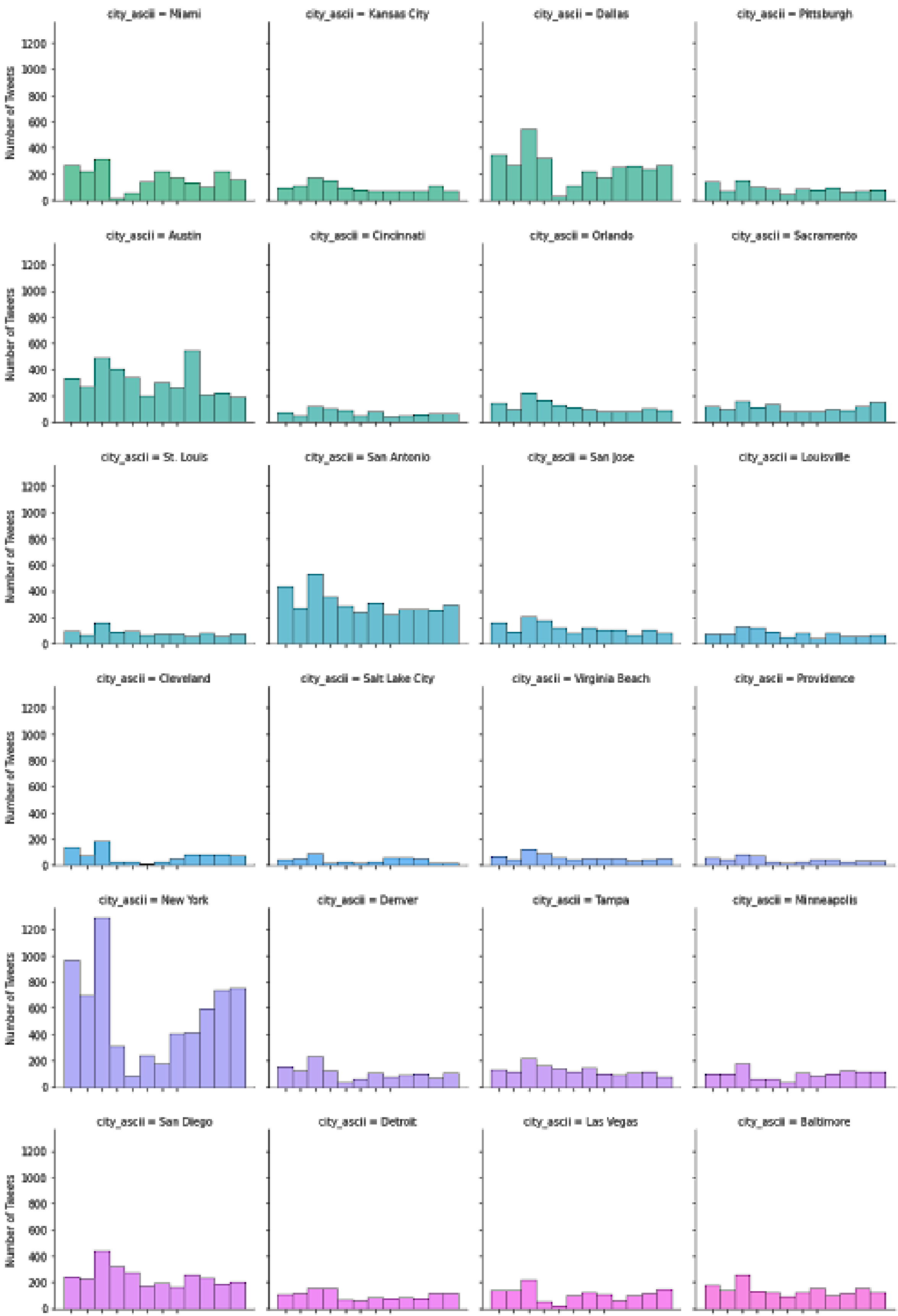


The number of US tweets by platform and by month also mimic the trends seen in tweets from around the world.



Again, the trends in tweets across US cities mimicked those from around the world by month, as well as the overall US trends, as found on the following two pages:


city_ascii = Brooklyn
city_ascii = Washington
city_ascii = Phoenix
city_ascii = Boston
city_ascii = Seattle
city_ascii = San Francisco
city_ascii = Raleigh
city_ascii = Charlotte
city_ascii = Columbus
city_ascii = Indianapolis
city_ascii = Manhattan
city_ascii = Milwaukee
city_ascii = Chicago
city_ascii = Nashville
city_ascii = Philadelphia
city_ascii = Memphis
city_ascii = Richmond
city_ascii = Portland
city_ascii = Houston
city_ascii = Atlanta
Number of Tweets
0
200
400
600
800
1000
1200


city_ascii = Miami
city_ascii = Kansas City
city_ascii = Dallas
city_ascii = Pittsburgh
city_ascii = Austin
city_ascii = Cincinnati
city_ascii = Orlando
city_ascii = Sacramento
city_ascii = St. Louis
city_ascii = San Antonio
city_ascii = San Jose
city_ascii = Louisville
city_ascii = Cleveland
city_ascii = Salt Lake City
city_ascii = Virginia Beach
city_ascii = Providence
city_ascii = New York
city_ascii = Denver
city_ascii = Tampa
city_ascii = Minneapolis
city_ascii = San Diego
city_ascii = Detroit
city_ascii = Las Vegas
city_ascii = Baltimore
Number of Tweets
0
200
400
600
800
1000
1200

ANALYSIS #3

Research Question: Is there a relationship between a country's capital city tweet data mentioning the 3 streaming platforms and their country's overall happiness index report?

Results:

There are a few variable comparisons that unveil some interesting insights from this correlation analysis.

Correlation of Overall Ladder score (Happiness index score) and # of tweets per streaming platform:

- Ladder score (Happiness index score) and # of Amazon Prime tweets are positively correlated: R = 0.54. Comparing this to Hulu and Netflix, this is a much higher correlation than the other 2 platforms.
- Ladder score (Happiness index score) and # of Hulu tweets not correlated: R = -0.01.
- Ladder score (Happiness index score) and # of Netflix tweets are positively correlated: R = 0.31, but a weaker correlation than the Ladder score and # of Amazon Prime tweets variables.

Correlation of Other Happiness Index Variables and # of tweets per streaming platform:

- # of Amazon Prime tweets has a consistently higher R correlation coefficient than the other streaming platforms for the following 3 happiness index breakdown variables: Logged GDP per capita, Social support, and Healthy life expectancy
- Logged GDP per capita:
    - # of Amazon Prime tweets and Logged GDP per capita: R = 0.43
    - # of Hulu tweets and Logged GDP per capita: R = 0.16
    - # of Netflix tweets and Logged GDP per capita: R = 0.15
- Social support:
    - # of Amazon Prime tweets and Social support: R = 0.45
    - # of Hulu tweets and Social support: R = 0.01
    - # of Netflix tweets and Social support: R = 0.23
- Healthy life expectancy:

- # of Amazon Prime tweets and Healthy life expectancy: R = 0.48
- # of Hulu tweets and Healthy life expectancy: R = 0.05
- # of Netflix tweets and Healthy life expectancy: R = 0.10

As we recall from the Nielsen report, “Netflix has the biggest share of video streaming on television at 29 percent, followed by YouTube with 20 percent, Hulu at 10 percent, and Amazon Prime Video snagging 9 percent.”[6] We hypothesize that given the much higher volume of Netflix tweets and Netflix’s largest share of video streaming, a large majority of people have Netflix as their priority streaming platform. And only a smaller portion of the population would be able to afford additional platforms like Hulu and Amazon Prime as additional streaming options on top of their existing Netflix membership. Given Amazon Prime’s high subscription price at $13, there may be a relationship between a population’s overall income and its ability to afford additional services like Amazon Prime, which has both streaming entertainment benefits and 2-day shipping benefits for Amazon use cases on the platform all-up. We hypothesize that the ability to afford Amazon Prime is potentially a “luxury” or a premium offering that is more common in populations with higher logged GDP per capita, social support, and healthy life expectancy. Hence, this may be why we are seeing much higher correlations between volume of Amazon Prime tweets, and the overall Ladder score, logged GDP per capita, social support, and healthy life expectancy variables.

	Ladder score	population	Logged GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices	Generosity	Perceptions of corruption	likes_count	retweets_count	replies_count	platform_Amazon Prime	platform_Hulu	platform_Netflix
Ladder score	1.000000	0.092946	0.754602	0.826611	0.745345	0.525630	0.029104	-0.350129	0.294066	0.200077	0.264444	0.536595	-0.008961	0.306943
population	0.092946	1.000000	0.212646	0.062889	0.219841	0.178426	0.071970	0.062203	0.355075	0.403031	0.408425	0.145704	0.194133	0.368849
Logged GDP per capita	0.754602	0.212646	1.000000	0.736044	0.865481	0.209105	-0.132002	-0.400931	0.119219	0.094971	0.128693	0.427974	0.162868	0.149955
Social support	0.826611	0.062889	0.736044	1.000000	0.749022	0.439606	-0.093064	-0.342934	0.190180	0.167036	0.180875	0.454370	0.005182	0.234979
Healthy life expectancy	0.745345	0.219841	0.865481	0.749022	1.000000	0.303234	-0.154759	-0.432570	0.067052	0.028924	0.058391	0.483570	0.051160	0.099840
Freedom to make life choices	0.525630	0.178426	0.209105	0.439606	0.303234	1.000000	0.248565	-0.293139	0.234147	0.318657	0.243545	0.082766	0.171368	0.265616
Generosity	0.029104	0.071970	-0.132002	-0.093064	-0.154759	0.248565	1.000000	-0.083421	0.101371	0.245632	0.297384	0.070037	0.273498	0.058503
Perceptions of corruption	-0.350129	0.062203	-0.400931	-0.342934	-0.432570	-0.293139	-0.083421	1.000000	0.010957	0.144513	0.142551	-0.193060	0.126406	0.051856
likes_count	0.294066	0.355075	0.119219	0.190180	0.067052	0.234147	0.101371	0.010957	1.000000	0.835555	0.855276	0.345354	0.112015	0.956166
retweets_count	0.200077	0.403031	0.094971	0.167036	0.028924	0.318657	0.245632	0.144513	0.835555	1.000000	0.707665	0.153304	0.343367	0.837978
replies_count	0.264444	0.408425	0.128693	0.180875	0.058391	0.243545	0.297384	0.142551	0.855276	0.707665	1.000000	0.477872	0.232312	0.869253
platform_Amazon Prime	0.536595	0.145704	0.427974	0.454370	0.483570	0.082766	0.070037	-0.193060	0.345354	0.153304	0.477872	1.000000	0.015983	0.400839
platform_Hulu	-0.008961	0.194133	0.162868	0.005182	0.051160	0.171368	0.273498	0.126406	0.112015	0.343367	0.232312	0.015983	1.000000	0.147641
platform_Netflix	0.306943	0.368849	0.149955	0.234979	0.099840	0.265616	0.058503	0.051856	0.956166	0.837978	0.869253	0.400839	0.147641	1.000000

[6] https://www.indiewire.com/2020/03/tv-streaming-march-2020-increase-nielsen-report-1202221796/

ANALYSIS #4

Research Question: What are the top keywords used in tweets mentioning the 3 streaming platforms from the top 50 US cities?

Results:

From these results, we can get a quick overview of the contents of the tweets content. Not surprisingly, “watch” and “watching” are the top 2 most common words. What’s noteworthy is that “show” is the 3rd most common term, and the terms “season”, “series”, and “shows” are all on this top 50 list of words. These words are all contextually related to television shows, whereas “movie” is only the 6th most common term, and there are no other words for movies anywhere in this top 50 list, like “movies” (the plural form), “film”, or “films”. This might suggest that a larger portion of all of the tweets content are related to television shows more than they are about movie content on the streaming platforms. However, this is purely an inference, since we aren’t looking at the actual titles of television shows or movies mentioned in the tweets content. Positive words like “good”, “like”, and “love” are mentioned on this list, whereas no negative standalone words are on this list. Since we didn’t perform a deeper natural language processing analysis on this to know if these “positive” words are used next to negation words, like “not good”, “didn’t like”, or “didn’t love”. However, it is noteworthy that there are not standalone negative words anywhere on this top 50 words list, like “hated”, “sucked”, or “terrible”.

We can form 2 hypotheses that would need further analyses to prove with greater confidence. First, we hypothesize that a larger portion of the tweets content mentioning the streaming platforms are about television shows, and a smaller portion of tweets are about movies. Second, perhaps people generally tweet about streaming platforms when they have positive comments or praise for the streaming platform in general or entertainment content offerings.

	token	frequency
0	watch	13496
1	watching	11847
2	show	8494
3	good	7576
4	like	6799
5	amp	5991
6	movie	5670
7	season	5427
8	new	5128
9	one	5110
10	need	4273
11	watched	4250
12	series	4215
13	get	4186
14	love	4013
15	time	3943
16	got	3890
17	really	3551
18	know	3158
19	shows	2940
20	see	2774

Conclusions

Overall, tweets in 2020 mentioning streaming platforms that were generated throughout the US and the world exhibited similar trends. There were far more tweets about Netflix than Hulu or Amazon Prime. The number of tweets mentioning these platforms drastically increased in March 2020, decreased through the summer months, and began to increase again into the fall and winter months. It is apparent that as people across the world spent more time streaming videos as quarantine for the Covid-19 pandemic began. The decrease in tweets about streaming in the summer months may be due to multiple factors. This may include a shift in activities due to both fatigue around activities people initially found solace in, as well as the warmer weather (at least in the Northern Hemisphere). Tweet activity was fairly consistent across regions, despite happiness score. Countries in Western Europe had the highest happiness score in 2020.

Higher correlations were found between happiness score and the Amazon Prime platform, as well as between happiness score and gross domestic product per capita, social support, and healthy life expectancy, respectively. This is likely due to the fact that Amazon Prime provides a more premium service than Hulu and Netflix.

In terms of tweet content, people generally tweeted positive sentiments when tweeting about streaming platforms throughout 2020. There is evidence to hypothesize those tweeting about streaming platforms were referring to shows over movies, however further analysis would be required to confirm this assumption.

The information gained throughout this analysis is not only of human interest, but could also be utilized as business insight for streaming platforms, both those included in this analysis and those excluded. It could also inform show and movie producers in terms of choosing a streaming platform to pitch their content to.

In general, throughout 2020 and the first year of the Covid-19 pandemic, people throughout the world turned to streaming services, likely as a form of entertainment, yet also solace through one of the most universally exigent times in recent history. Although most people throughout the world were physically apart, streaming services proved to be a unifying force.

Team Roles and Responsibilities

Tasks	Primary Owner(s)
Scope and define the final project research questions	Both
Identify the data sources	Both
Twitter scraping using Twint library	Both
Collecting and storing the data from data sources	Both
Data cleansing	Jennifer
Analysis #1 and #2	Christina
Analysis #3 and #4	Jennifer
Data visualizations	Christina
Writing the report	Both
PowerPoint presentation content	Both

Streaming_Happiness_Project

March 21, 2021

1 Streaming Platform Twitter Trends in the Covid-19 Pandemic

1.1 Data Load and Preparation

Filename: Streaming_Happiness_Project.ipynb Date Created: March 2, 2021 Last Updated: March 20, 2021 Created By: Jennifer Han and Christina DaSilva Purpose: Read in Twitter data on streaming platforms and World Happiness Report data and perform analyses

1.1.1 Package Install and Import Statements

[2]:
```
#!pip install twint #Only needed for retreiving tweets (already done)
```

[3]:
```
#!pip install WordCloud #Run, if necessary
```

[4]:
```
import pandas as pd
#import twint #Only needed for retreiving tweets (already done)
import json
import csv
import nltk
import re
import seaborn as sea
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
import matplotlib.pyplot as plt
```

1.1.2 File Access

[5]:
```
# THIS IS ONLY REQUIRED WHEN LOADING IN COLAB
# Mount Google Drive for file access
# from google.colab import drive
# drive.mount('/content/drive/')
```

```
Mounted at /content/drive/
```

[6]:
```
# # THIS IS ONLY REQUIRED WHEN LOADING IN COLAB
# Change directory to shared project folder
# %cd /content/drive/MyDrive/IST652_Final_Project
```

```
/content/drive/.shortcut-targets-by-
id/1uHFNLuM2WRelgZSY8a1DwH3DDjseXPAe/IST652_Final_Project
```

1.1.3 Load World Happiness Report Data

[7]:
```
################################
# WORLD HAPPINESS REPORT DATA #
################################
# Read in World Happiness Report data from csv
happiness_filename = '2020_world_happiness_report.csv'

happiness_df = pd.read_csv(happiness_filename)

# Print out some information about the data read in
print('Successfully read in data from:', happiness_filename, '\n')
print('Happiness Dataframe Shape: {row} rows and {col} columns'.format(row =␣
 ↪happiness_df.shape[0], col = happiness_df.shape[1]), '\n')
print('First 10 rows Happiness Dataframe:\n', happiness_df.head(10), '\n')
```

```
Successfully read in data from: 2020_world_happiness_report.csv

Happiness Dataframe Shape: 153 rows and 20 columns

First 10 rows Happiness Dataframe:
   Country name  … Dystopia + residual
0       Finland  …            2.762835
1       Denmark  …            2.432741
2   Switzerland  …            2.350267
3       Iceland  …            2.460688
4        Norway  …            2.168266
5   Netherlands  …            2.352117
6        Sweden  …            2.246299
7   New Zealand  …            2.128108
8       Austria  …            2.398446
9    Luxembourg  …            2.153700

[10 rows x 20 columns]
```

1.1.4 Load World Cities Data

[8]:
```
#####################
# WORLD CITIES DATA #
#####################
# Read in World Cities data and store in a dataframe
# The worldcities.csv file is sorted from largest population cities to smallest
# Website source: https://simplemaps.com/data/world-cities
```

```python
world_cities_file = 'worldcities.csv'
world_cities_df = pd.read_csv(world_cities_file)

print('First 10 rows World Cities Dataframe: ', world_cities_df.head(10), '\n')

# Get only the rows of the dataset that are the capital cities
# According to the data dictionary for the dataset (see website), where capital␣
↪column = "primary")
capitals_df = world_cities_df[world_cities_df['capital'] == 'primary']

# Drop duplicates to clean up the data. Some countries have more than 1 primary␣
↪capital listed in this dataset
capitals_df.drop_duplicates(subset = 'country', inplace = True)

print('First 10 rows Capitals Dataframe:\n', capitals_df.head(10), '\n')
print('Capitals Dataframe Shape: ', capitals_df.shape, '\n')
```

```
First 10 rows World Cities Dataframe:            city   city_ascii      lat  …
capital  population          id
0        Tokyo        Tokyo  35.6897  …  primary  37977000.0  1392685764
1      Jakarta      Jakarta  -6.2146  …  primary  34540000.0  1360771077
2        Delhi        Delhi  28.6600  …    admin  29617000.0  1356872604
3       Mumbai       Mumbai  18.9667  …    admin  23355000.0  1356226629
4       Manila       Manila  14.5958  …  primary  23088000.0  1608618140
5     Shanghai     Shanghai  31.1667  …    admin  22120000.0  1156073548
6    São Paulo    Sao Paulo -23.5504  …    admin  22046000.0  1076532519
7        Seoul        Seoul  37.5833  …  primary  21794000.0  1410836482
8  Mexico City  Mexico City  19.4333  …  primary  20996000.0  1484247881
9    Guangzhou    Guangzhou  23.1288  …    admin  20902000.0  1156237133

[10 rows x 11 columns]

First 10 rows Capitals Dataframe:
             city    city_ascii      lat  …  capital  population          id
0          Tokyo         Tokyo  35.6897  …  primary  37977000.0  1392685764
1        Jakarta       Jakarta  -6.2146  …  primary  34540000.0  1360771077
4         Manila        Manila  14.5958  …  primary  23088000.0  1608618140
7          Seoul         Seoul  37.5833  …  primary  21794000.0  1410836482
8    Mexico City   Mexico City  19.4333  …  primary  20996000.0  1484247881
10       Beijing       Beijing  39.9050  …  primary  19433000.0  1156228865
11         Cairo         Cairo  30.0561  …  primary  19372000.0  1818253931
14        Moscow        Moscow  55.7558  …  primary  17125000.0  1643318494
15       Bangkok       Bangkok  13.7500  …  primary  17066000.0  1764068610
16  Buenos Aires  Buenos Aires -34.5997  …  primary  16157000.0  1032717330

[10 rows x 11 columns]
```

```
Capitals Dataframe Shape:  (197, 11)


/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:18:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

1.1.5 Join Capitals and World Happiness Report Data

[9]:
```python
# Join together the happiness df and the capitals df on the country field

# Use a left join to check for spelling discrepancies
happiness_capitals_df = happiness_df.merge(capitals_df, how = 'left', left_on =␣
↪'Country name', right_on = 'country')
print('First 10 rows Happiness Capitals Dataframe:\n', happiness_capitals_df.
↪head(10), '\n')

print('Happiness Capitals Dataframe Shape: ', happiness_capitals_df.shape, '\n')

# Data cleaning for country names that are spelled differently
# Find the rows with NaN due to the left join and the country names not␣
↪matching precisely
cities_with_NAN_df = happiness_capitals_df[happiness_capitals_df['city_ascii'].
↪isna()]
print('Cities with NAN due to left join: \n', cities_with_NAN_df[['Country␣
↪name', 'city_ascii']], '\n')

# Create a dictionary with the happiness dataset country name mapped to the␣
↪capitals dataset country names
# Any country with 'None' denoted means that there are no cities in that␣
↪country where the capital field = primary
# Thus, we need to remove that row from the happiness_df before doing the left␣
↪join
country_name_mapping = {
        'Canada': 'None',
        'Czech Republic':'Czechia',
        'Taiwan Province of China': 'Taiwan',
        'Trinidad and Tobago': 'Trinidad And Tobago',
        'South Korea': 'Korea, South',
        'Bosnia and Herzegovina': 'Bosnia And Herzegovina',
        'North Cyprus': 'None',
        'Hong Kong S.A.R. of China': 'None',
        'Ivory Coast': 'None',
        'Gambia': 'Gambia, The',
```

```
        'Palestinian Territories': 'None',
        'Myanmar': 'None'
        }


# Clean up the happiness_df and the capitals_df for the left join on country␣
↪name to remove all NaN values
# Replace the country field in the capitals dataframe with the new values
capitals_clean_df = capitals_df
for key in country_name_mapping:
        capitals_clean_df = capitals_clean_df.
↪replace(country_name_mapping[key], key)

# Drop rows of the happiness dataframe for cities where there are no cities␣
↪with capital = primary according to the World Cities dataset
happiness_clean_df = happiness_df
indexes_to_drop = []
for key in country_name_mapping:
        if country_name_mapping[key] == 'None':
                happiness_clean_df = happiness_clean_df.
↪drop(happiness_clean_df[ happiness_clean_df['Country name'] == key].index)


# Create a new left join merge of the cleaned up happiness_clean_df and the␣
↪capitals_clean_df
happiness_capitals_clean_df = happiness_clean_df.merge(capitals_clean_df, how =␣
↪'left', left_on = 'Country name', right_on = 'country')

# Sort the dataframe with the largest population capitals first
happiness_capitals_clean_df = happiness_capitals_clean_df.
↪sort_values(by=['population'], ascending = False)
print('First 10 rows Happiness Capitals Clean Dataframe:\n',␣
↪happiness_capitals_clean_df.head(10), '\n')

# Check to ensure that the new left join with clean data is returning no NaN␣
↪values, which means every left join from the happiness dataset was successful
print('Post Data Cleansing -- Cities with NaN values due to left join:',␣
↪len(happiness_capitals_clean_df[happiness_capitals_clean_df['city_ascii'].
↪isna()]), '\n')

# Get the top 50 rows
top50_happiness_capitals_clean_df = happiness_capitals_clean_df[0:50]
```

```
First 10 rows Happiness Capitals Dataframe:
   Country name     Regional indicator  …  population            id
0      Finland         Western Europe  …    642045.0  1.246178e+09
1      Denmark         Western Europe  …   1085000.0  1.208764e+09
```

```
2  Switzerland         Western Europe  …    133798.0  1.756374e+09
3      Iceland         Western Europe  …    128793.0  1.352327e+09
4       Norway         Western Europe  …    693494.0  1.578325e+09
5  Netherlands         Western Europe  …   1406000.0  1.528800e+09
6       Sweden         Western Europe  …    972647.0  1.752426e+09
7  New Zealand  North America and ANZ  …    418500.0  1.554772e+09
8      Austria         Western Europe  …   1840573.0  1.040262e+09
9   Luxembourg         Western Europe  …    122273.0  1.442263e+09

[10 rows x 31 columns]

Happiness Capitals Dataframe Shape:  (153, 31)

Cities with NAN due to left join:
                   Country name city_ascii
10                       Canada        NaN
18               Czech Republic        NaN
24    Taiwan Province of China        NaN
41          Trinidad and Tobago        NaN
60                  South Korea        NaN
68       Bosnia and Herzegovina        NaN
75                 North Cyprus        NaN
77   Hong Kong S.A.R. of China        NaN
84                  Ivory Coast        NaN
112                      Gambia        NaN
124    Palestinian Territories        NaN
132                     Myanmar        NaN

First 10 rows Happiness Capitals Clean Dataframe:
     Country name                  Regional indicator  …  population
id
60         Japan                            East Asia  …  37977000.0
1392685764
80     Indonesia                       Southeast Asia  …  34540000.0
1360771077
50   Philippines                       Southeast Asia  …  23088000.0
1608618140
59   South Korea                            East Asia  …  21794000.0
1410836482
22        Mexico          Latin America and Caribbean  …  20996000.0
1484247881
89         China                            East Asia  …  19433000.0
1156228865
131        Egypt        Middle East and North Africa  …  19372000.0
1818253931
71        Russia  Commonwealth of Independent States  …  17125000.0
1643318494
52      Thailand                       Southeast Asia  …  17066000.0
```

```
1764068610
53      Argentina         Latin America and Caribbean  …  16157000.0
1032717330

[10 rows x 31 columns]

Post Data Cleansing -- Cities with NaN values due to left join: 0
```

1.1.6 Load US Cities Data

[10]:
```
##################
# US CITIES DATA #
##################
# Read in US Cities data and store in a dataframe
# The uscities.csv file is sorted from largest population cities to smallest
# Website source: https://simplemaps.com/data/us-cities

us_cities_file = 'uscities.csv'
us_cities_df = pd.read_csv(us_cities_file)

# Sort the dataframe with the largest population US cities first
us_cities_df = us_cities_df.sort_values(by=['population'], ascending = False)

# Get just the top 50 rows
top50_us_cities_df = us_cities_df[0:50]

print('First 10 rows US Cities Dataframe: ', top50_us_cities_df.head(10), '\n')
```

```
First 10 rows US Cities Dataframe:              city  …          id
0      New York  …  1840034016
1   Los Angeles  …  1840020491
2       Chicago  …  1840000494
3         Miami  …  1840015149
4        Dallas  …  1840019440
5  Philadelphia  …  1840000673
6       Houston  …  1840020925
7       Atlanta  …  1840013660
8    Washington  …  1840006060
9        Boston  …  1840000455

[10 rows x 17 columns]
```

Set Up Twitter Scraping Functions

[11]:
```
# Create a function that scrapes data per city using twint
```

```
def scrape_tweets_by_city(lat, long, radius, output_filename, search_term,␣
↪since_date, until_date):
        c = twint.Config()
        c.Search = search_term
        c.Output = output_filename
        c.Geo = '{lat},{long},{radius}km'.format(lat = lat, long = long, radius␣
↪= radius)
        c.Since = since_date
        c.Until = until_date
        c.Count = True
        c.Store_json = True
        twint.run.Search(c)


# Create csv output filenames
world_netflix_output_filename = 'world_tweets_netflix.json'
world_prime_output_filename = 'world_tweets_prime.json'
world_hulu_output_filename = 'world_tweets_hulu.json'

us_netflix_output_filename = 'us_tweets_netflix.json'
us_prime_output_filename = 'us_tweets_prime.json'
us_hulu_output_filename = 'us_tweets_hulu.json'


# Set date range for Jan - Dec 2020
since_date = '2020-01-01'
until_date = '2020-12-31'


# Set radius
city_radius = 10
```

1.1.7 Scrape Twitter Data

(Already run - do not need to run again)

[12]:
```
# Note: Due to the long time runtime for this code, it is commented out as it␣
↪has already been run

######################################
# SCRAPE WORLD CAPITALS TWITTER DATA #
######################################

# Scrape tweets that mention the 3 streaming platforms for top 50 world capital␣
↪cities by population
# Note: This will take some time to run and complete.
```

```
######### COMMENTED OUT SO THIS DOESN'T RE-RUN!!!!!!!
"""
print('Scraping tweets in 2020 for world capitals...')

for index, row in top50_happiness_capitals_clean_df.iterrows():
        print('Scraping Tweets from: ', row['city'], '\n')
        scrape_tweets_by_city(row['lat'], row['lng'], city_radius,␣
 ↪world_netflix_output_filename, 'netflix', since_date, until_date)
        scrape_tweets_by_city(row['lat'], row['lng'], city_radius,␣
 ↪world_prime_output_filename, '\"primevideo\"', since_date, until_date)
        scrape_tweets_by_city(row['lat'], row['lng'], city_radius,␣
 ↪world_prime_output_filename, '\"prime video\"', since_date, until_date)
        scrape_tweets_by_city(row['lat'], row['lng'], city_radius,␣
 ↪world_hulu_output_filename, 'hulu', since_date, until_date)
"""




##################################
# SCRAPE US CITIES TWITTER DATA #
##################################

# Scrape tweets that mention the 3 streaming platforms for top 50 US cities by␣
 ↪population
# Note: This will take some time to run and complete.


######### COMMENTED OUT SO THIS DOESN'T RE-RUN!!!!!!!
"""
print('Scraping tweets in 2020 for US capitals...')

for index, row in top50_us_cities_df.iterrows():
        print('Scraping Tweets from: ', row['city'], '\n')
        scrape_tweets_by_city(row['lat'], row['lng'], city_radius,␣
 ↪us_netflix_output_filename, 'netflix', since_date, until_date)
        scrape_tweets_by_city(row['lat'], row['lng'], city_radius,␣
 ↪us_prime_output_filename, '\"primevideo\"', since_date, until_date)
        scrape_tweets_by_city(row['lat'], row['lng'], city_radius,␣
 ↪us_prime_output_filename, '\"prime video\"', since_date, until_date)
        scrape_tweets_by_city(row['lat'], row['lng'], city_radius,␣
 ↪us_hulu_output_filename, 'hulu', since_date, until_date)
"""
```

[12]:
```
'\nprint(\'Scraping tweets in 2020 for US capitals…\')\n\nfor index, row in
top50_us_cities_df.iterrows():\n\tprint(\'Scraping Tweets from: \',
row[\'city\'], \'\n\')\n\tscrape_tweets_by_city(row[\'lat\'], row[\'lng\'],
```

```
city_radius, us_netflix_output_filename, \'netflix\', since_date,
until_date)\n\tscrape_tweets_by_city(row[\'lat\'], row[\'lng\'], city_radius,
us_prime_output_filename, \'"primevideo"\', since_date,
until_date)\n\tscrape_tweets_by_city(row[\'lat\'], row[\'lng\'], city_radius,
us_prime_output_filename, \'"prime video"\', since_date,
until_date)\n\tscrape_tweets_by_city(row[\'lat\'], row[\'lng\'], city_radius,
us_hulu_output_filename, \'hulu\', since_date, until_date)\n'
```

1.1.8 Read In, Store, Clean Twitter Data from JSON Files

World Capitals Twitter Data

[13]:
```
# WORLD CAPITALS TWITTER DATA
# Read in the 3 json files of tweet data and convert to pandas dataframes
# Insert a new column for the streaming platform name. This will be used when␣
↪we append the 3 dataframes into 1 dataframe.
world_netflix_df = pd.read_json(world_netflix_output_filename, lines = True)
world_netflix_df.insert(0, 'platform', 'Netflix')
print('World Netflix Dataframe Created: \n', world_netflix_df.head(10), '\n')
print('World Netflix Dataframe Shape: \n', world_netflix_df.shape, '\n')

world_prime_df = pd.read_json(world_prime_output_filename, lines = True)
world_prime_df.insert(0, 'platform', 'Amazon Prime')
print('World Amazon Prime Dataframe Created: \n', world_prime_df.head(10), '\n')
print('World Amazon Prime Dataframe Shape: \n', world_prime_df.shape, '\n')

world_hulu_df = pd.read_json(world_hulu_output_filename, lines = True)
world_hulu_df.insert(0, 'platform', 'Hulu')
print('World Hulu Dataframe Created: \n', world_hulu_df.head(10), '\n')
print('World Hulu Dataframe Shape: \n', world_hulu_df.shape, '\n')

# Append the dataframes to create 1 dataframe for world capitals tweets
world_tweets_all_platforms_df = world_netflix_df.append(world_prime_df).
↪append(world_hulu_df)
print('World All Platforms Dataframe Created: \n',␣
↪world_tweets_all_platforms_df.head(10), '\n')
print('World All Platforms Dataframe Shape: \n', world_tweets_all_platforms_df.
↪shape, '\n')

# Insert 2 new columns for the lat and lng columns extracted from the geo column
world_tweets_all_platforms_df.insert(1, 'lat', 'Not assigned')
world_tweets_all_platforms_df.insert(2, 'lng', 'Not assigned')

# Create 2 functions that parse the geo column and returns the lat, long values
def extract_lat(row):
        geolocation = row['geo'].split(',')
        return float(geolocation[0])
```

```
def extract_long(row):
        geolocation = row['geo'].split(',')
        return float(geolocation[1])

# Use .apply() function to assign the lat and lng columns from the geo column␣
 ↪for all rows of the dataframe
world_tweets_all_platforms_df['lat'] = world_tweets_all_platforms_df.
 ↪apply(lambda row: extract_lat(row), axis=1)
world_tweets_all_platforms_df['lng'] = world_tweets_all_platforms_df.
 ↪apply(lambda row: extract_long(row), axis=1)

print('World All Platforms Dataframe lat, lng columns added: \n',␣
 ↪world_tweets_all_platforms_df.head(10), '\n')

# Left join world_tweets_all_platforms_df with the city_ascii and country␣
 ↪columns from top50_happiness_capitals_clean_df on the lat and lng columns
print('Finding matching city and country for World Tweets Dataframe...')
world_tweets_all_platforms_df = world_tweets_all_platforms_df.
 ↪merge(top50_happiness_capitals_clean_df[['city_ascii', 'country', 'lat',␣
 ↪'lng']], how='left', left_on=['lat', 'lng'], right_on=['lat','lng'])
print('World All Platforms Dataframe matched city and country columns: \n',␣
 ↪world_tweets_all_platforms_df.head(10), '\n')
print('World All Platforms Dataframe Shape: \n', world_tweets_all_platforms_df.
 ↪shape, '\n')

# Drop the extraneous lat and lng columns now that we have the city and country␣
 ↪columns added. We only needed them to perform the left join
world_tweets_all_platforms_df = world_tweets_all_platforms_df.drop(['lat',␣
 ↪'lng'], axis = 1)
print('World All Platforms Dataframe dropped lat lng columns: \n',␣
 ↪world_tweets_all_platforms_df.head(10), '\n')
```

```
/usr/local/lib/python3.7/dist-packages/dateutil/parser/_parser.py:1218:
UnknownTimezoneWarning: tzname EST identified but not understood.  Pass
`tzinfos` argument in order to correctly return a timezone-aware datetime.  In a
future version, this will raise an exception.
  category=UnknownTimezoneWarning)
/usr/local/lib/python3.7/dist-packages/dateutil/parser/_parser.py:1218:
UnknownTimezoneWarning: tzname EDT identified but not understood.  Pass
`tzinfos` argument in order to correctly return a timezone-aware datetime.  In a
future version, this will raise an exception.
  category=UnknownTimezoneWarning)
```

```
World Netflix Dataframe Created:
   platform                   id  …  trans_src trans_dest
0  Netflix  1343948213116551170  …
```

```
1  Netflix  1343917356456636417  …
2  Netflix  1343901145345900545  …
3  Netflix  1343889948944945153  …
4  Netflix  1343888922447777792  …
5  Netflix  1343878569622028288  …
6  Netflix  1343872556391022595  …
7  Netflix  1343862001785663488  …
8  Netflix  1343831708936761344  …
9  Netflix  1343808973095366658  …

[10 rows x 37 columns]

World Netflix Dataframe Shape:
 (42530, 37)


/usr/local/lib/python3.7/dist-packages/dateutil/parser/_parser.py:1218:
UnknownTimezoneWarning: tzname EST identified but not understood.  Pass
`tzinfos` argument in order to correctly return a timezone-aware datetime.  In a
future version, this will raise an exception.
  category=UnknownTimezoneWarning)
/usr/local/lib/python3.7/dist-packages/dateutil/parser/_parser.py:1218:
UnknownTimezoneWarning: tzname EDT identified but not understood.  Pass
`tzinfos` argument in order to correctly return a timezone-aware datetime.  In a
future version, this will raise an exception.
  category=UnknownTimezoneWarning)

World Amazon Prime Dataframe Created:
        platform                   id  …  trans_src trans_dest
0  Amazon Prime  1343862001785663488  …
1  Amazon Prime  1342480833060356097  …
2  Amazon Prime  1342038474321317890  …
3  Amazon Prime  1336956907165081603  …
4  Amazon Prime  1251153879862964226  …
5  Amazon Prime  1249331709209796608  …
6  Amazon Prime  1247491816061952000  …
7  Amazon Prime  1247228547812323329  …
8  Amazon Prime  1246352987611484162  …
9  Amazon Prime  1343882324828594176  …

[10 rows x 37 columns]

World Amazon Prime Dataframe Shape:
 (1220, 37)


/usr/local/lib/python3.7/dist-packages/dateutil/parser/_parser.py:1218:
UnknownTimezoneWarning: tzname EST identified but not understood.  Pass
`tzinfos` argument in order to correctly return a timezone-aware datetime.  In a
```

```
future version, this will raise an exception.
  category=UnknownTimezoneWarning)
/usr/local/lib/python3.7/dist-packages/dateutil/parser/_parser.py:1218:
UnknownTimezoneWarning: tzname EDT identified but not understood.  Pass
`tzinfos` argument in order to correctly return a timezone-aware datetime.  In a
future version, this will raise an exception.
  category=UnknownTimezoneWarning)

World Hulu Dataframe Created:
   platform                   id  …  trans_src trans_dest
0      Hulu  1344069452988497920  …
1      Hulu  1343855550795149312  …
2      Hulu  1343586630339239943  …
3      Hulu  1343405622478290944  …
4      Hulu  1342779346080034816  …
5      Hulu  1342480520542883840  …
6      Hulu  1342383147850158080  …
7      Hulu  1341974560292233217  …
8      Hulu  1340570141906489345  …
9      Hulu  1340568764262170625  …

[10 rows x 37 columns]

World Hulu Dataframe Shape:
 (1575, 37)

World All Platforms Dataframe Created:
   platform                   id  …  trans_src trans_dest
0  Netflix  1343948213116551170  …
1  Netflix  1343917356456636417  …
2  Netflix  1343901145345900545  …
3  Netflix  1343889948944945153  …
4  Netflix  1343888922447777792  …
5  Netflix  1343878569622028288  …
6  Netflix  1343872556391022595  …
7  Netflix  1343862001785663488  …
8  Netflix  1343831708936761344  …
9  Netflix  1343808973095366658  …

[10 rows x 37 columns]

World All Platforms Dataframe Shape:
 (45325, 37)

World All Platforms Dataframe lat, lng columns added:
   platform      lat       lng  …  translate  trans_src trans_dest
0  Netflix  35.6897  139.6922  …
1  Netflix  35.6897  139.6922  …
```

```
2  Netflix  35.6897  139.6922  …
3  Netflix  35.6897  139.6922  …
4  Netflix  35.6897  139.6922  …
5  Netflix  35.6897  139.6922  …
6  Netflix  35.6897  139.6922  …
7  Netflix  35.6897  139.6922  …
8  Netflix  35.6897  139.6922  …
9  Netflix  35.6897  139.6922  …

[10 rows x 39 columns]

Finding matching city and country for World Tweets Dataframe…
World All Platforms Dataframe matched city and country columns:
   platform      lat       lng  …  trans_dest  city_ascii country
0  Netflix  35.6897  139.6922  …                   Tokyo   Japan
1  Netflix  35.6897  139.6922  …                   Tokyo   Japan
2  Netflix  35.6897  139.6922  …                   Tokyo   Japan
3  Netflix  35.6897  139.6922  …                   Tokyo   Japan
4  Netflix  35.6897  139.6922  …                   Tokyo   Japan
5  Netflix  35.6897  139.6922  …                   Tokyo   Japan
6  Netflix  35.6897  139.6922  …                   Tokyo   Japan
7  Netflix  35.6897  139.6922  …                   Tokyo   Japan
8  Netflix  35.6897  139.6922  …                   Tokyo   Japan
9  Netflix  35.6897  139.6922  …                   Tokyo   Japan

[10 rows x 41 columns]

World All Platforms Dataframe Shape:
 (45325, 41)

World All Platforms Dataframe dropped lat lng columns:
   platform                   id  …  city_ascii country
0  Netflix  1343948213116551170  …       Tokyo   Japan
1  Netflix  1343917356456636417  …       Tokyo   Japan
2  Netflix  1343901145345900545  …       Tokyo   Japan
3  Netflix  1343889948944945153  …       Tokyo   Japan
4  Netflix  1343888922447777792  …       Tokyo   Japan
5  Netflix  1343878569622028288  …       Tokyo   Japan
6  Netflix  1343872556391022595  …       Tokyo   Japan
7  Netflix  1343862001785663488  …       Tokyo   Japan
8  Netflix  1343831708936761344  …       Tokyo   Japan
9  Netflix  1343808973095366658  …       Tokyo   Japan

[10 rows x 39 columns]
```

US Cities Twitter Data

[14]:
```
# US CITIES TWITTER DATA
# Read in the 3 json files of tweet data and convert to pandas dataframes
# Insert a new column for the streaming platform name. This will be used when␣
↪we append the 3 dataframes into 1 dataframe.
us_netflix_df = pd.read_json(us_netflix_output_filename, lines = True)
us_netflix_df.insert(0, 'platform', 'Netflix')
print('US Netflix Dataframe Created: \n', us_netflix_df.head(10), '\n')
print('US Netflix Dataframe Shape: \n', us_netflix_df.shape, '\n')

us_prime_df = pd.read_json(us_prime_output_filename, lines = True)
us_prime_df.insert(0, 'platform', 'Amazon Prime')
print('US Amazon Prime Dataframe Created: \n', us_prime_df.head(10), '\n')
print('US Amazon Prime Dataframe Shape: \n', us_prime_df.shape, '\n')

us_hulu_df = pd.read_json(us_hulu_output_filename, lines = True)
us_hulu_df.insert(0, 'platform', 'Hulu')
print('US Hulu Dataframe Created: \n', us_hulu_df.head(10), '\n')
print('US Hulu Dataframe Shape: \n', us_hulu_df.shape, '\n')

# Append the dataframes to create 1 dataframe for US cities tweets
us_tweets_all_platforms_df = us_netflix_df.append(us_prime_df).
↪append(us_hulu_df)
print('US All Platforms Dataframe Created: \n', us_tweets_all_platforms_df.
↪head(10), '\n')
print('US All Platforms Dataframe Shape: \n', us_tweets_all_platforms_df.shape,␣
↪'\n')



# Insert 2 new columns for the lat and lng columns extracted from the geo column
us_tweets_all_platforms_df.insert(1, 'lat', 'Not assigned')
us_tweets_all_platforms_df.insert(2, 'lng', 'Not assigned')

# We will use the same 2 functions extract_lat and extract_long defined earlier␣
↪to parse the geo column and return the lat, long values
# Use .apply() function to assign the lat and lng columns from the geo column␣
↪for all rows of the dataframe
us_tweets_all_platforms_df['lat'] = us_tweets_all_platforms_df.apply(lambda row:
↪ extract_lat(row), axis=1)
us_tweets_all_platforms_df['lng'] = us_tweets_all_platforms_df.apply(lambda row:
↪ extract_long(row), axis=1)

print('US All Platforms Dataframe lat, lng columns added: \n',␣
↪us_tweets_all_platforms_df.head(10), '\n')

# Left join us_tweets_all_platforms_df with the city_ascii column from␣
↪top50_us_cities_df on the lat and lng columns
```

```
print('Finding matching city for US Tweets Dataframe...')
us_tweets_all_platforms_df = us_tweets_all_platforms_df.
 ↪merge(top50_us_cities_df[['city_ascii', 'lat', 'lng']], how='left',␣
 ↪left_on=['lat', 'lng'], right_on=['lat','lng'])
print('US All Platforms Dataframe matched city and country columns: \n',␣
 ↪us_tweets_all_platforms_df.head(10), '\n')
print('US All Platforms Dataframe Shape: \n', us_tweets_all_platforms_df.shape,␣
 ↪'\n')

# Drop the extraneous lat and lng columns now that we have the city columns␣
 ↪added. We only needed them to perform the left join
us_tweets_all_platforms_df = us_tweets_all_platforms_df.drop(['lat', 'lng'],␣
 ↪axis = 1)
print('US All Platforms Dataframe dropped lat lng columns: \n',␣
 ↪us_tweets_all_platforms_df.head(10), '\n')
```

```
/usr/local/lib/python3.7/dist-packages/dateutil/parser/_parser.py:1218:
UnknownTimezoneWarning: tzname EST identified but not understood.  Pass
`tzinfos` argument in order to correctly return a timezone-aware datetime.  In a
future version, this will raise an exception.
  category=UnknownTimezoneWarning)
/usr/local/lib/python3.7/dist-packages/dateutil/parser/_parser.py:1218:
UnknownTimezoneWarning: tzname EDT identified but not understood.  Pass
`tzinfos` argument in order to correctly return a timezone-aware datetime.  In a
future version, this will raise an exception.
  category=UnknownTimezoneWarning)
```

```
US Netflix Dataframe Created:
   platform                   id  …  trans_src trans_dest
0  Netflix  1344070429900681218  …
1  Netflix  1344068892763807744  …
2  Netflix  1344068472494575617  …
3  Netflix  1344053767814184962  …
4  Netflix  1344048704400072709  …
5  Netflix  1344048644660465664  …
6  Netflix  1344033012737060865  …
7  Netflix  1344029858993725440  …
8  Netflix  1343996838387601408  …
9  Netflix  1343989226413699074  …

[10 rows x 37 columns]

US Netflix Dataframe Shape:
 (88549, 37)


/usr/local/lib/python3.7/dist-packages/dateutil/parser/_parser.py:1218:
UnknownTimezoneWarning: tzname EST identified but not understood.  Pass
```

```
`tzinfos` argument in order to correctly return a timezone-aware datetime.  In a
future version, this will raise an exception.
  category=UnknownTimezoneWarning)
/usr/local/lib/python3.7/dist-packages/dateutil/parser/_parser.py:1218:
UnknownTimezoneWarning: tzname EDT identified but not understood.  Pass
`tzinfos` argument in order to correctly return a timezone-aware datetime.  In a
future version, this will raise an exception.
  category=UnknownTimezoneWarning)

US Amazon Prime Dataframe Created:
        platform                   id  …  trans_src trans_dest
0  Amazon Prime  1344059014989099010  …
1  Amazon Prime  1344030839877881856  …
2  Amazon Prime  1344019729388810241  …
3  Amazon Prime  1343975344563499010  …
4  Amazon Prime  1343973196614279168  …
5  Amazon Prime  1343968652178182147  …
6  Amazon Prime  1343559722432155648  …
7  Amazon Prime  1343421521742737408  …
8  Amazon Prime  1343200403207446528  …
9  Amazon Prime  1342983601093275651  …

[10 rows x 37 columns]

US Amazon Prime Dataframe Shape:
 (3701, 37)


/usr/local/lib/python3.7/dist-packages/dateutil/parser/_parser.py:1218:
UnknownTimezoneWarning: tzname EST identified but not understood.  Pass
`tzinfos` argument in order to correctly return a timezone-aware datetime.  In a
future version, this will raise an exception.
  category=UnknownTimezoneWarning)
/usr/local/lib/python3.7/dist-packages/dateutil/parser/_parser.py:1218:
UnknownTimezoneWarning: tzname EDT identified but not understood.  Pass
`tzinfos` argument in order to correctly return a timezone-aware datetime.  In a
future version, this will raise an exception.
  category=UnknownTimezoneWarning)

US Hulu Dataframe Created:
   platform                   id  …  trans_src trans_dest
0     Hulu  1344068892763807744  …
1     Hulu  1344048704400072709  …
2     Hulu  1343951793525575681  …
3     Hulu  1343797981166907393  …
4     Hulu  1343786834921476097  …
5     Hulu  1343760649273401344  …
6     Hulu  1343714312616095745  …
7     Hulu  1343659409206079489  …
```

```
8     Hulu  1343611405375844352  …
9     Hulu  1343559722432155648  …

[10 rows x 37 columns]

US Hulu Dataframe Shape:
 (15043, 37)

US All Platforms Dataframe Created:
   platform                   id  …  trans_src trans_dest
0  Netflix  1344070429900681218  …
1  Netflix  1344068892763807744  …
2  Netflix  1344068472494575617  …
3  Netflix  1344053767814184962  …
4  Netflix  1344048704400072709  …
5  Netflix  1344048644660465664  …
6  Netflix  1344033012737060865  …
7  Netflix  1344029858993725440  …
8  Netflix  1343996838387601408  …
9  Netflix  1343989226413699074  …

[10 rows x 37 columns]

US All Platforms Dataframe Shape:
 (107293, 37)

US All Platforms Dataframe lat, lng columns added:
   platform      lat      lng  …  translate  trans_src trans_dest
0  Netflix  40.6943 -73.9249  …
1  Netflix  40.6943 -73.9249  …
2  Netflix  40.6943 -73.9249  …
3  Netflix  40.6943 -73.9249  …
4  Netflix  40.6943 -73.9249  …
5  Netflix  40.6943 -73.9249  …
6  Netflix  40.6943 -73.9249  …
7  Netflix  40.6943 -73.9249  …
8  Netflix  40.6943 -73.9249  …
9  Netflix  40.6943 -73.9249  …

[10 rows x 39 columns]

Finding matching city for US Tweets Dataframe…
US All Platforms Dataframe matched city and country columns:
   platform      lat      lng  …  trans_src  trans_dest city_ascii
0  Netflix  40.6943 -73.9249  …                          New York
1  Netflix  40.6943 -73.9249  …                          New York
2  Netflix  40.6943 -73.9249  …                          New York
3  Netflix  40.6943 -73.9249  …                          New York
```

```
4   Netflix  40.6943 -73.9249  …                          New York
5   Netflix  40.6943 -73.9249  …                          New York
6   Netflix  40.6943 -73.9249  …                          New York
7   Netflix  40.6943 -73.9249  …                          New York
8   Netflix  40.6943 -73.9249  …                          New York
9   Netflix  40.6943 -73.9249  …                          New York

[10 rows x 40 columns]

US All Platforms Dataframe Shape:
 (107293, 40)

US All Platforms Dataframe dropped lat lng columns:
   platform                   id  …  trans_dest city_ascii
0  Netflix  1344070429900681218  …              New York
1  Netflix  1344068892763807744  …              New York
2  Netflix  1344068472494575617  …              New York
3  Netflix  1344053767814184962  …              New York
4  Netflix  1344048704400072709  …              New York
5  Netflix  1344048644660465664  …              New York
6  Netflix  1344033012737060865  …              New York
7  Netflix  1344029858993725440  …              New York
8  Netflix  1343996838387601408  …              New York
9  Netflix  1343989226413699074  …              New York

[10 rows x 38 columns]
```

These are the following dataframes to be used and manipulated for the analyses below

1. **top50_happiness_capitals_clean_df**: The top 50 world capitals (based on population size) dataset joined with the associated happiness report dataset for that country.
2. **top50_us_cities_df**: The top 50 US cities (based on population size) dataset
3. **world_tweets_all_platforms_df**: Tweets from Jan 1 - Dec 31 2020 that had a geolocation in the top 50 world capitals that mention "netflix", "prime video" or "primevideo", and "hulu"
4. **us_tweets_all_platforms_df**: Tweets from Jan 1 - Dec 31 2020 that had a geolocation in the top 50 US cities that mention "netflix", "prime video" or "primevideo", and "hulu"

[15]:
```
world_tweets_all_platforms_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 45325 entries, 0 to 45324
Data columns (total 39 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   platform         45325 non-null  object
```

```
 1   id               45325 non-null  int64
 2   conversation_id  45325 non-null  int64
 3   created_at       45325 non-null  datetime64[ns]
 4   date             45325 non-null  datetime64[ns]
 5   time             45325 non-null  object
 6   timezone         45325 non-null  int64
 7   user_id          45325 non-null  int64
 8   username         45325 non-null  object
 9   name             45325 non-null  object
 10  place            45325 non-null  object
 11  tweet            45325 non-null  object
 12  language         45325 non-null  object
 13  mentions         45325 non-null  object
 14  urls             45325 non-null  object
 15  photos           45325 non-null  object
 16  replies_count    45325 non-null  int64
 17  retweets_count   45325 non-null  int64
 18  likes_count      45325 non-null  int64
 19  hashtags         45325 non-null  object
 20  cashtags         45325 non-null  object
 21  link             45325 non-null  object
 22  retweet          45325 non-null  bool
 23  quote_url        45325 non-null  object
 24  video            45325 non-null  int64
 25  thumbnail        45325 non-null  object
 26  near             45325 non-null  object
 27  geo              45325 non-null  object
 28  source           45325 non-null  object
 29  user_rt_id       45325 non-null  object
 30  user_rt          45325 non-null  object
 31  retweet_id       45325 non-null  object
 32  reply_to         45325 non-null  object
 33  retweet_date     45325 non-null  object
 34  translate        45325 non-null  object
 35  trans_src        45325 non-null  object
 36  trans_dest       45325 non-null  object
 37  city_ascii       45325 non-null  object
 38  country          45325 non-null  object
dtypes: bool(1), datetime64[ns](2), int64(8), object(28)
memory usage: 13.5+ MB
```

[16]:
```
us_tweets_all_platforms_df.info()
```

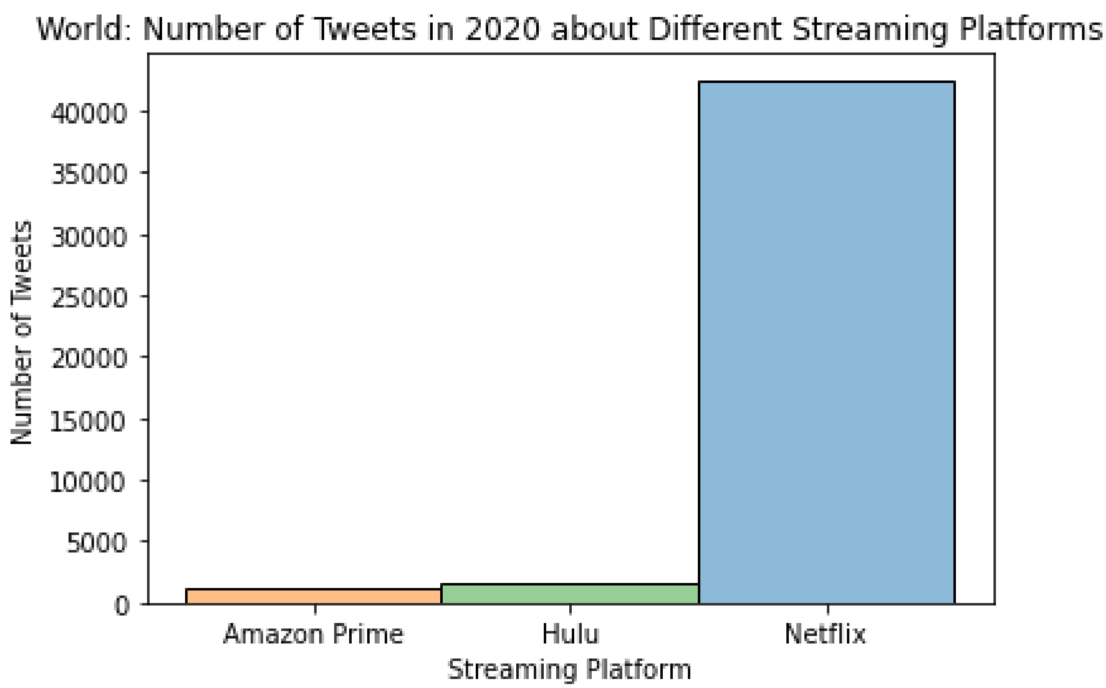
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 107293 entries, 0 to 107292
Data columns (total 38 columns):
 #   Column           Non-Null Count   Dtype
---  ------           --------------   -----
```

```
 0   platform         107293 non-null  object
 1   id               107293 non-null  int64
 2   conversation_id  107293 non-null  int64
 3   created_at       107293 non-null  datetime64[ns]
 4   date             107293 non-null  datetime64[ns]
 5   time             107293 non-null  object
 6   timezone         107293 non-null  int64
 7   user_id          107293 non-null  int64
 8   username         107293 non-null  object
 9   name             107293 non-null  object
 10  place            107293 non-null  object
 11  tweet            107293 non-null  object
 12  language         107293 non-null  object
 13  mentions         107293 non-null  object
 14  urls             107293 non-null  object
 15  photos           107293 non-null  object
 16  replies_count    107293 non-null  int64
 17  retweets_count   107293 non-null  int64
 18  likes_count      107293 non-null  int64
 19  hashtags         107293 non-null  object
 20  cashtags         107293 non-null  object
 21  link             107293 non-null  object
 22  retweet          107293 non-null  bool
 23  quote_url        107293 non-null  object
 24  video            107293 non-null  int64
 25  thumbnail        107293 non-null  object
 26  near             107293 non-null  object
 27  geo              107293 non-null  object
 28  source           107293 non-null  object
 29  user_rt_id       107293 non-null  object
 30  user_rt          107293 non-null  object
 31  retweet_id       107293 non-null  object
 32  reply_to         107293 non-null  object
 33  retweet_date     107293 non-null  object
 34  translate        107293 non-null  object
 35  trans_src        107293 non-null  object
 36  trans_dest       107293 non-null  object
 37  city_ascii       107293 non-null  object
dtypes: bool(1), datetime64[ns](2), int64(8), object(27)
memory usage: 31.2+ MB
```

1.2 Analysis

1.2.1 ANALYSIS #1: World Demographic Trends on Tweets of Streaming Platforms

[17]:
```
############################################################################
# ANALYSIS #1: World demographic trends on Tweets of streaming platforms #
############################################################################

# Hist plot of tweets by platform

# Aggregates world tweets by country, counting tweets, by platform, per country
w_platform_df = world_tweets_all_platforms_df.sort_values(by=['id'],␣
↪ascending=True).groupby(['platform','country']).id.count()
w_platform_df.head(45)

a = sea.histplot(data=w_platform_df,x='platform', hue='platform')
a = sea.histplot(data=world_tweets_all_platforms_df,x='platform',␣
↪hue='platform')
a.set(xlabel='Streaming Platform', ylabel='Number of Tweets', title='World:␣
↪Number of Tweets in 2020 about Different Streaming Platforms')
a.legend_.remove()
```



[36]:
```
# Top five countries with the most tweets, by each platform
w = world_tweets_all_platforms_df
w_agg = w.groupby(['platform', 'country']).count().reset_index()
```

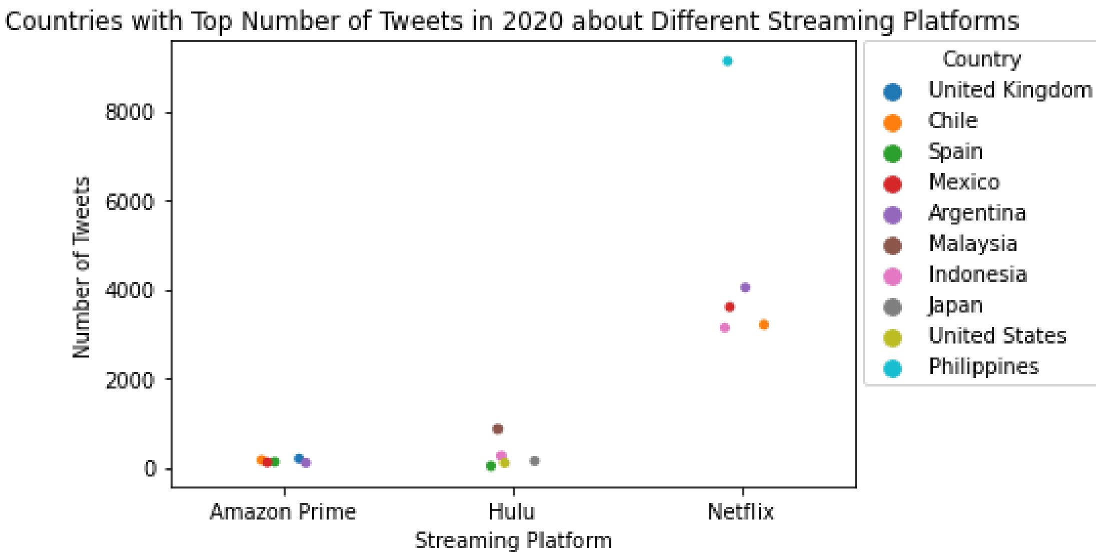
```python
#print(w_agg)

w_sorted = w_agg.groupby(['platform']).apply(lambda x: x.
↪sort_values(['id'],ascending = False)).reset_index(drop = True)
#print(w_sorted)

x = w_sorted.groupby(['platform']).head(5).reset_index()
x.rename(columns = {'platform':'Platform','country':'Country','id':'Tweet␣
↪Count'}, inplace = True)
x[['Platform','Country','Tweet Count']]
```

[36]:

	Platform	Country	Tweet Count
0	Amazon Prime	United Kingdom	197
1	Amazon Prime	Chile	164
2	Amazon Prime	Spain	123
3	Amazon Prime	Mexico	112
4	Amazon Prime	Argentina	101
5	Hulu	Malaysia	860
6	Hulu	Indonesia	263
7	Hulu	Japan	144
8	Hulu	United States	98
9	Hulu	Spain	31
10	Netflix	Philippines	9117
11	Netflix	Argentina	4038
12	Netflix	Mexico	3601
13	Netflix	Chile	3202
14	Netflix	Indonesia	3136

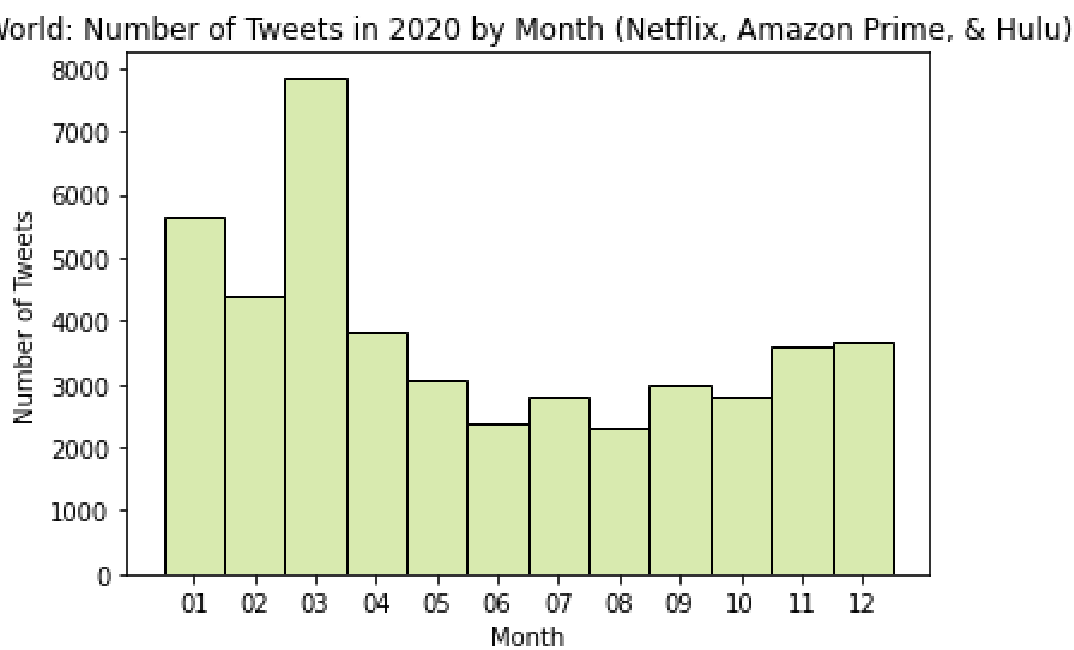
[19]:
```python
# Plot of Countries with Top Number of Tweets in 2020 about Different Streaming␣
↪Platforms
b=sea.stripplot(data=x, x='platform', y='id', hue='country')
b.legend(bbox_to_anchor=(1.01, 1),borderaxespad=0).set_title('Country') # move␣
↪legend outside plot
b.set(xlabel='Streaming Platform', ylabel='Number of Tweets', title='Countries␣
↪with Top Number of Tweets in 2020 about Different Streaming Platforms')
```

[19]:
```
[Text(0, 0.5, 'Number of Tweets'),
 Text(0.5, 0, 'Streaming Platform'),
 Text(0.5, 1.0, 'Countries with Top Number of Tweets in 2020 about Different
Streaming Platforms')]
```



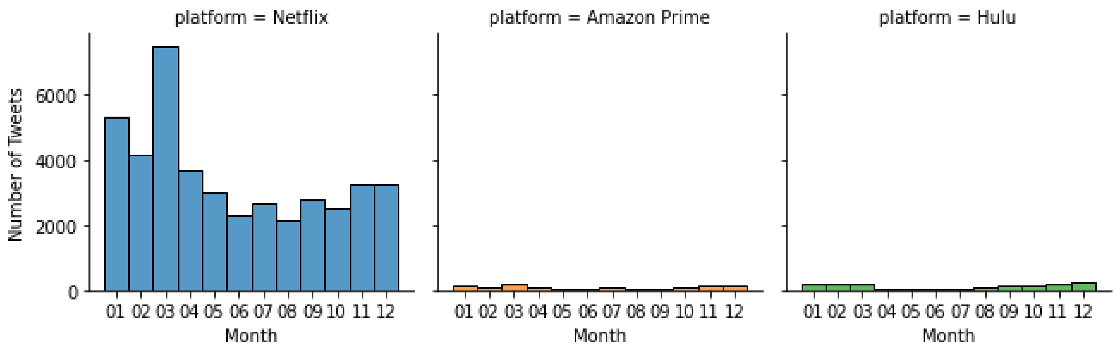
[20]:

```
# Number of World Tweets in 2020 by Month
w['month'] = w['date'].dt.strftime('%m')
y=w.sort_values('month', ascending=True)
c = sea.histplot(y,x='month', color='#cbe395').set(xlabel='Month',␣
↪ylabel='Number of Tweets', title='World: Number of Tweets in 2020 by Month␣
↪(Netflix, Amazon Prime, & Hulu)')
```



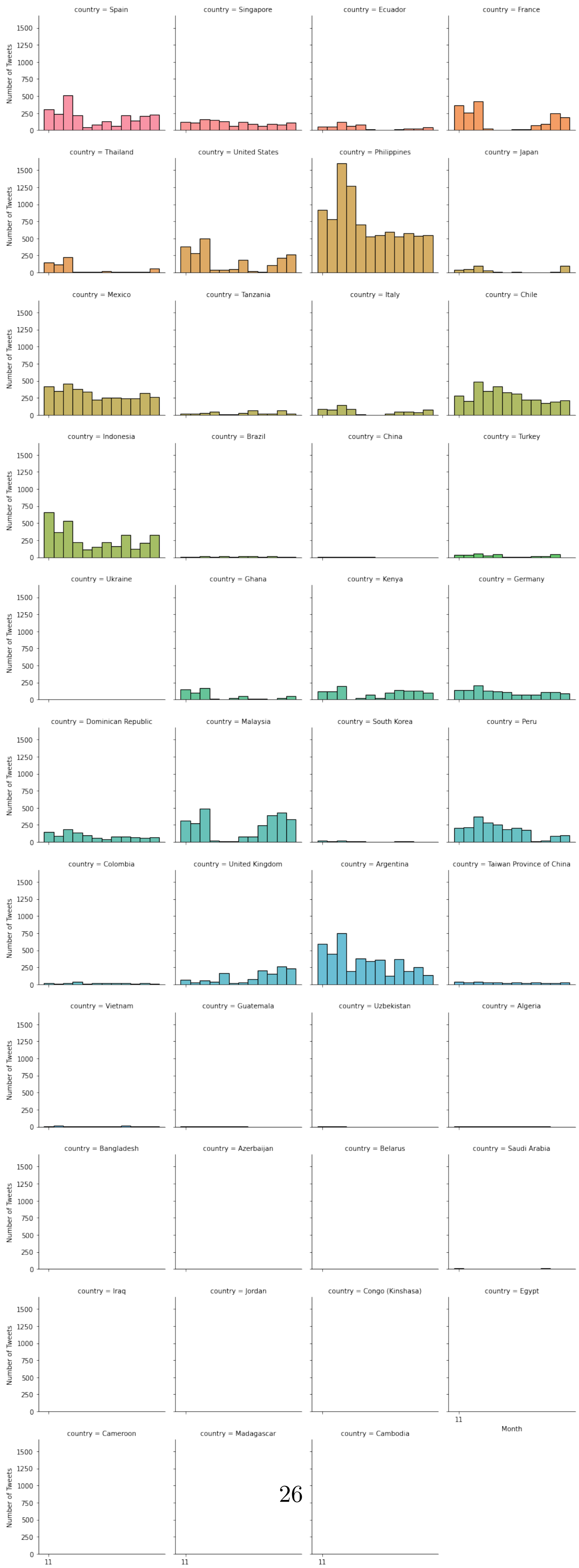
[21]:
```
# Number of World Tweets in 2020 by Month and Platform
y=w.sort_values('month', ascending=True)
d = sea.FacetGrid(y, col='platform', hue='platform')
d.map(sea.histplot, 'month').set(xlabel='Month', ylabel='Number of Tweets')
```

[21]:
```
<seaborn.axisgrid.FacetGrid at 0x7fa88ee7c8d0>
```



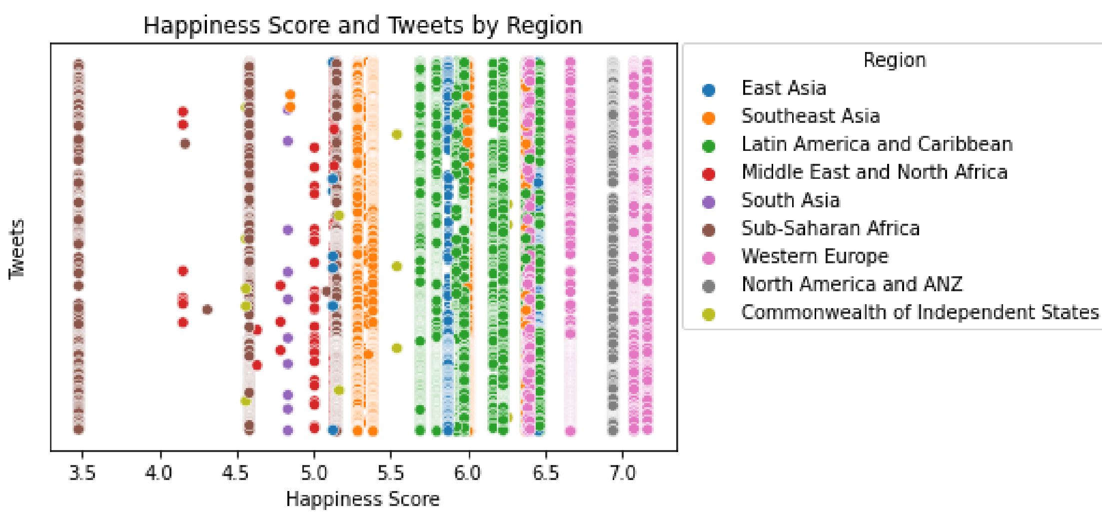
[22]:
```
# Number of Tweets in 2020 by Country by Month
y=w.sort_values('month', ascending=True)
e = sea.FacetGrid(y, col='country', col_wrap=4, hue='country')
e.map(sea.histplot, 'month').set(xlabel='Month', ylabel='Number of Tweets')
```

[22]:
```
<seaborn.axisgrid.FacetGrid at 0x7fa88de94690>
```


country = Spain
country = Singapore
country = Ecuador
country = France
country = Thailand
country = United States
country = Philippines
country = Japan
country = Mexico
country = Tanzania
country = Italy
country = Chile
country = Indonesia
country = Brazil
country = China
country = Turkey
country = Ukraine
country = Ghana
country = Kenya
country = Germany
country = Dominican Republic
country = Malaysia
country = South Korea
country = Peru
country = Colombia
country = United Kingdom
country = Argentina
country = Taiwan Province of China
country = Vietnam
country = Guatemala
country = Uzbekistan
country = Algeria
country = Bangladesh
country = Azerbaijan
country = Belarus
country = Saudi Arabia
country = Iraq
country = Jordan
country = Congo (Kinshasa)
country = Egypt
country = Cameroon
country = Madagascar
country = Cambodia
Number of Tweets
0
250
500
750
1000
1250
1500
11
Month

[50]:
```python
# Join world tweets (all platforms) and happiness rating by country - NEED TO␣
↪FIX Y AXIS
wtw = world_tweets_all_platforms_df
top50 = top50_happiness_capitals_clean_df

z=wtw.merge(top50, on='country', how='left')
z['All Platforms'] = 'All Platforms'
#z.head()

f = sea.scatterplot(data=z, x="Ladder score", y="id_x", hue='Regional␣
↪indicator')
f.legend(bbox_to_anchor=(1.01, 1),borderaxespad=0).set_title('Region') # move␣
↪legend outside plot
f.set(xlabel='Happiness Score', ylabel='Tweets', title='Happiness Score and␣
↪Tweets by Region')
f.set(yticklabels=[])
f.tick_params(left=False)
f
```

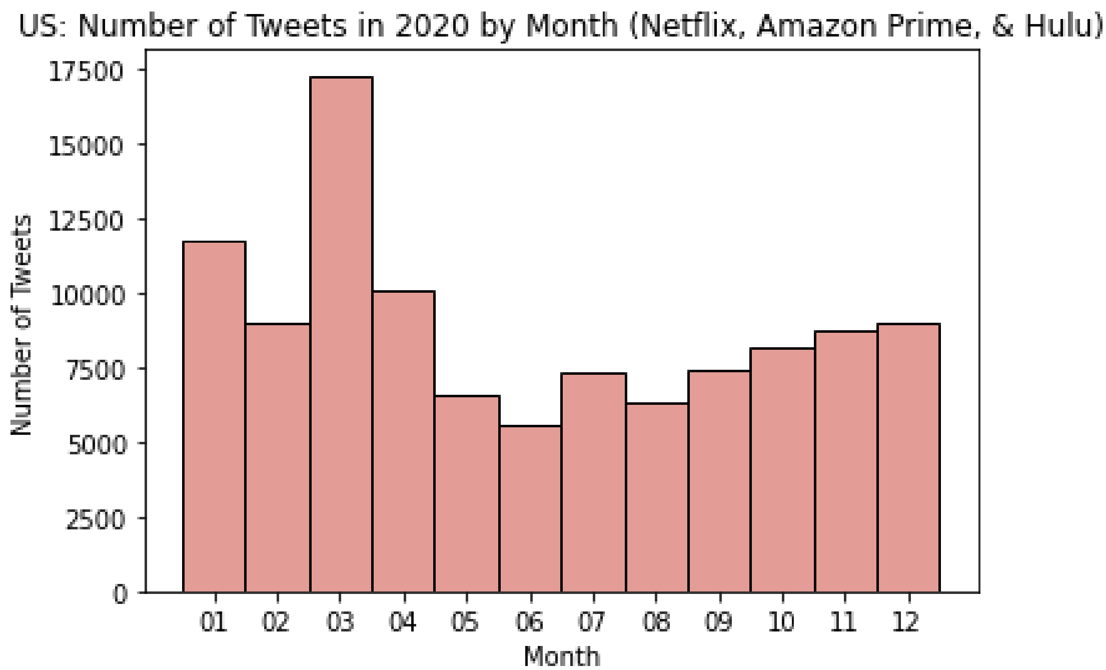
[50]: <matplotlib.axes._subplots.AxesSubplot at 0x7fa8782b4150>



1.2.2 ANALYSIS #2 US Demographic Trends on Tweets of Streaming Platforms

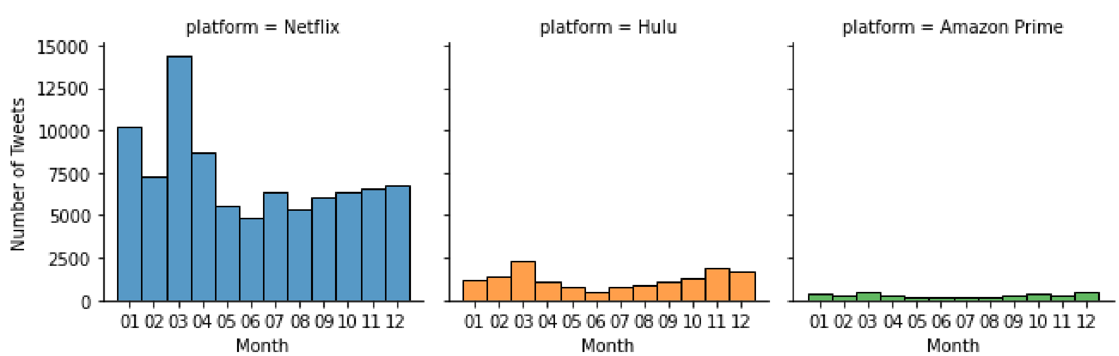
[25]:
```python
# Number of US Tweets in 2020 by Month
u = us_tweets_all_platforms_df
u['month'] = u['date'].dt.strftime('%m')
v=u.sort_values('month', ascending=True)
```

```
g = sea.histplot(v, x='month', color='#da7c73').set(xlabel='Month',␣
↪ylabel='Number of Tweets', title='US: Number of Tweets in 2020 by Month␣
↪(Netflix, Amazon Prime, & Hulu)')
```



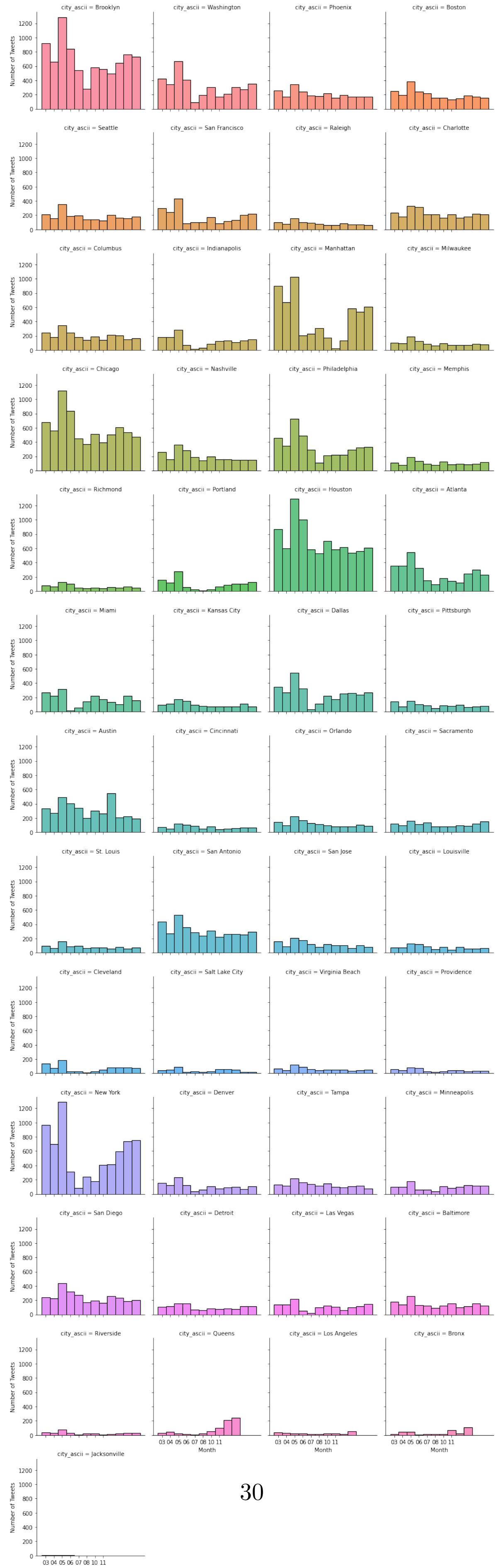
[26]:
```
# Number of US Tweets in 2020 by Month and Platform
u=u.sort_values('month', ascending=True)
h = sea.FacetGrid(u, col='platform', hue='platform')
h.map(sea.histplot, 'month').set(xlabel='Month', ylabel='Number of Tweets')
```

[26]: <seaborn.axisgrid.FacetGrid at 0x7fa88c0bf890>



[27]:
```
# Number of Tweets in 2020 by US Cities by Month
t=u.sort_values('month', ascending=True)
i = sea.FacetGrid(t, col='city_ascii', col_wrap=4, hue='city_ascii')
i.map(sea.histplot, 'month').set(xlabel='Month', ylabel='Number of Tweets')
```

[27]:
```
<seaborn.axisgrid.FacetGrid at 0x7fa88d4f0410>
```


city_ascii = Brooklyn
city_ascii = Washington
city_ascii = Phoenix
city_ascii = Boston
city_ascii = Seattle
city_ascii = San Francisco
city_ascii = Raleigh
city_ascii = Charlotte
city_ascii = Columbus
city_ascii = Indianapolis
city_ascii = Manhattan
city_ascii = Milwaukee
city_ascii = Chicago
city_ascii = Nashville
city_ascii = Philadelphia
city_ascii = Memphis
city_ascii = Richmond
city_ascii = Portland
city_ascii = Houston
city_ascii = Atlanta
city_ascii = Miami
city_ascii = Kansas City
city_ascii = Dallas
city_ascii = Pittsburgh
city_ascii = Austin
city_ascii = Cincinnati
city_ascii = Orlando
city_ascii = Sacramento
city_ascii = St. Louis
city_ascii = San Antonio
city_ascii = San Jose
city_ascii = Louisville
city_ascii = Cleveland
city_ascii = Salt Lake City
city_ascii = Virginia Beach
city_ascii = Providence
city_ascii = New York
city_ascii = Denver
city_ascii = Tampa
city_ascii = Minneapolis
city_ascii = San Diego
city_ascii = Detroit
city_ascii = Las Vegas
city_ascii = Baltimore
city_ascii = Riverside
city_ascii = Queens
city_ascii = Los Angeles
city_ascii = Bronx
city_ascii = Jacksonville
Number of Tweets
0
200
400
600
800
1000
1200
03 04 05 06 07 08 10 11
Month

1.2.3 ANALYSIS #3: World correlation analysis on Happiness Report and Twitter data

[28]:

```
##############################################################################
# ANALYSIS #3: World correlation analysis on Happiness Report and Twitter data #
##############################################################################

# Summarize the world_tweets_all_platforms_df dataframe
# Create a new dataframe for worlds tweets and select only the columns we care␣
↪about from the tweet data
print('Column Names of world_tweets_all_platforms_df: ',␣
↪world_tweets_all_platforms_df.columns)

world_tweets_select_cols_df = world_tweets_all_platforms_df[['platform',␣
↪'city_ascii', 'date', 'likes_count', 'retweets_count', 'replies_count',␣
↪'tweet']]
print('World Tweets (Selected Columns) Dataframe Created: \n',␣
↪world_tweets_select_cols_df.head(10), '\n')

# Expand the platform column into 3 columns that are either 0 or 1 for if the␣
↪tweet mentioned one of the 3 streaming platforms
world_tweets_select_cols_df = pd.get_dummies(world_tweets_select_cols_df,␣
↪prefix = 'platform', columns=['platform'])
print('Expanded platform Column into 3 with get_dummies(): \n',␣
↪world_tweets_select_cols_df.head(10), '\n')

# Perform a Group By on the dataframe by the city_ascii column
world_tweets_group_by_capital_df = world_tweets_select_cols_df.
↪groupby(by=['city_ascii']).sum().reset_index()
print('First 50 Rows World Tweets Dataframe Grouped By Capital Cities: \n',␣
↪world_tweets_group_by_capital_df.head(50), '\n')
print('World Tweets Dataframe Grouped By Capital Cities Shape: \n',␣
↪world_tweets_group_by_capital_df.shape, '\n')

# Scale down the top50_happiness_capitals_clean_df for only the columns we need␣
↪for this analysis
top50_happiness_capitals_select_cols_df =␣
↪top50_happiness_capitals_clean_df[['city_ascii', 'Country name', 'Ladder␣
↪score', 'population', 'Logged GDP per capita', 'Social support', 'Healthy␣
↪life expectancy', 'Freedom to make life choices', 'Generosity', 'Perceptions␣
↪of corruption']]
print('Happiness Report (Selected Columns) Dataframe Created: \n',␣
↪top50_happiness_capitals_select_cols_df.head(10), '\n')
```

```python
print('Happiness Report (Selected Columns) Dataframe Shape: \n',␣
→top50_happiness_capitals_select_cols_df.shape, '\n')

# Join top50_happiness_capitals_select_cols_df with␣
→world_tweets_group_by_capital_df on the 'city_ascii' column
# Use a left join because some capital cities had 0 tweets
world_tweets_and_happiness_joined_df = top50_happiness_capitals_select_cols_df.
→merge(world_tweets_group_by_capital_df, on='city_ascii', how = 'left')
print('Joined World Tweets and Happiness data together on city_ascii column:␣
→\n', world_tweets_and_happiness_joined_df.head(10), '\n')
print('Joined Dataframe Shape: \n', world_tweets_and_happiness_joined_df.shape,␣
→'\n')

# Run the correlation analysis
corr_analysis_df = world_tweets_and_happiness_joined_df.corr(method='pearson')
print('Correlation Analysis Results: \n', corr_analysis_df, '\n')
corr_analysis_df
```

```
Column Names of world_tweets_all_platforms_df:  Index(['platform', 'id',
'conversation_id', 'created_at', 'date', 'time',
       'timezone', 'user_id', 'username', 'name', 'place', 'tweet', 'language',
       'mentions', 'urls', 'photos', 'replies_count', 'retweets_count',
       'likes_count', 'hashtags', 'cashtags', 'link', 'retweet', 'quote_url',
       'video', 'thumbnail', 'near', 'geo', 'source', 'user_rt_id', 'user_rt',
       'retweet_id', 'reply_to', 'retweet_date', 'translate', 'trans_src',
       'trans_dest', 'city_ascii', 'country', 'month'],
      dtype='object')
World Tweets (Selected Columns) Dataframe Created:
   platform  …                                            tweet
0  Netflix  …                    bamba hotel …
1  Netflix  …       THE MOVIE Netflix             …
2  Netflix  …         … Netflix           htt…
3  Netflix  …                 Netflix     1
4  Netflix  …                THE MOVIE   Netflix   _(  _' ')_
5  Netflix  …             Netflix  CM
6  Netflix  …  @prestonoutatime @netflix Sweet! I'd add a boo…
7  Netflix  …  Netflix       PrimeVideo         …
8  Netflix  …       Netflix CM            …
9  Netflix  …    Ore ore    netflix

[10 rows x 7 columns]

Expanded platform Column into 3 with get_dummies():
   city_ascii       date  …  platform_Hulu  platform_Netflix
0      Tokyo 2020-12-29  …              0                 1
1      Tokyo 2020-12-29  …              0                 1
2      Tokyo 2020-12-29  …              0                 1
```

```
3      Tokyo 2020-12-29  …              0                 1
4      Tokyo 2020-12-29  …              0                 1
5      Tokyo 2020-12-29  …              0                 1
6      Tokyo 2020-12-29  …              0                 1
7      Tokyo 2020-12-29  …              0                 1
8      Tokyo 2020-12-29  …              0                 1
9      Tokyo 2020-12-29  …              0                 1

[10 rows x 9 columns]

First 50 Rows World Tweets Dataframe Grouped By Capital Cities:
         city_ascii  likes_count  …  platform_Hulu  platform_Netflix
0             Accra        16930  …            7.0             603.0
1           Algiers         7264  …            2.0              28.0
2             Amman            3  …            0.0               2.0
3            Ankara         1426  …            0.0             300.0
4      Antananarivo            0  …            0.0               1.0
5           Baghdad            0  …            0.0               3.0
6              Baku            0  …            0.0               2.0
7           Bangkok         5315  …            6.0             639.0
8           Beijing            0  …            8.0               2.0
9            Berlin        13471  …            4.0            1321.0
10           Bogota          593  …            1.0             184.0
11         Brasilia          196  …            2.0             105.0
12     Buenos Aires        21909  …           12.0            4038.0
13            Cairo           96  …            1.0               6.0
14    Dar es Salaam          722  …            6.0             297.0
15            Dhaka            0  …            0.0              10.0
16   Guatemala City            5  …            0.0              20.0
17            Hanoi          116  …            1.0              59.0
18          Jakarta        20086  …          263.0            3136.0
19         Kinshasa           17  …            0.0               1.0
20     Kuala Lumpur         9900  …          860.0            1799.0
21             Kyiv            1  …            0.0               7.0
22             Lima        10922  …            7.0            2063.0
23           London         8365  …           20.0            1103.0
24           Madrid        13038  …           31.0            2232.0
25           Manila        59993  …            0.0            9117.0
26      Mexico City        23469  …           25.0            3601.0
27            Minsk            0  …            0.0               3.0
28          Nairobi        12839  …           26.0            1119.0
29            Paris        20357  …            4.0            1642.0
30       Phnom Penh            0  …            0.0               2.0
31            Quito         2954  …            8.0             467.0
32           Riyadh           12  …            0.0              42.0
33             Rome         2368  …            3.0             554.0
34         Santiago        15900  …           28.0            3202.0
35    Santo Domingo         4295  …            0.0            1111.0
```

```
36          Seoul          652  …          1.0          104.0
37      Singapore         7892  …          6.0         1246.0
38         Taipei         1371  …          1.0          292.0
39       Tashkent            2  …          0.0            3.0
40          Tokyo         2965  …        144.0          136.0
41     Washington        10188  …         98.0         1927.0
42        Yaounde            0  …          0.0            1.0

[43 rows x 7 columns]

World Tweets Dataframe Grouped By Capital Cities Shape:
 (43, 7)

Happiness Report (Selected Columns) Dataframe Created:
        city_ascii Country name  …  Generosity  Perceptions of corruption
60          Tokyo        Japan  …   -0.246910                   0.654558
80        Jakarta    Indonesia  …    0.519587                   0.876296
50         Manila  Philippines  …   -0.105463                   0.733634
59          Seoul  South Korea  …   -0.043404                   0.789067
22    Mexico City       Mexico  …   -0.175267                   0.806822
89        Beijing        China  …   -0.181426                   0.753971
131         Cairo        Egypt  …   -0.196878                   0.787727
71         Moscow       Russia  …   -0.151154                   0.864803
52        Bangkok     Thailand  …    0.268685                   0.886272
53   Buenos Aires    Argentina  …   -0.194914                   0.842010

[10 rows x 10 columns]

Happiness Report (Selected Columns) Dataframe Shape:
 (50, 10)

Joined World Tweets and Happiness data together on city_ascii column:
      city_ascii Country name  …  platform_Hulu  platform_Netflix
0          Tokyo        Japan  …          144.0             136.0
1        Jakarta    Indonesia  …          263.0            3136.0
2         Manila  Philippines  …            0.0            9117.0
3          Seoul  South Korea  …            1.0             104.0
4    Mexico City       Mexico  …           25.0            3601.0
5        Beijing        China  …            8.0               2.0
6          Cairo        Egypt  …            1.0               6.0
7         Moscow       Russia  …            NaN               NaN
8        Bangkok     Thailand  …            6.0             639.0
9   Buenos Aires    Argentina  …           12.0            4038.0

[10 rows x 16 columns]

Joined Dataframe Shape:
 (50, 16)
```

```
Correlation Analysis Results:
                              Ladder score  …  platform_Netflix
Ladder score                      1.000000  …          0.306943
population                        0.092946  …          0.368849
Logged GDP per capita             0.754602  …          0.149955
Social support                    0.826611  …          0.234979
Healthy life expectancy           0.745345  …          0.099840
Freedom to make life choices      0.525630  …          0.265616
Generosity                        0.029104  …          0.058503
Perceptions of corruption        -0.350129  …          0.051856
likes_count                       0.294066  …          0.956166
retweets_count                    0.200077  …          0.837978
replies_count                     0.264444  …          0.869253
platform_Amazon Prime             0.536595  …          0.400839
platform_Hulu                    -0.008961  …          0.147641
platform_Netflix                  0.306943  …          1.000000

[14 rows x 14 columns]
```

[28]:


```
                              Ladder score  …  platform_Netflix
Ladder score                      1.000000  …          0.306943
population                        0.092946  …          0.368849
Logged GDP per capita             0.754602  …          0.149955
Social support                    0.826611  …          0.234979
Healthy life expectancy           0.745345  …          0.099840
Freedom to make life choices      0.525630  …          0.265616
Generosity                        0.029104  …          0.058503
Perceptions of corruption        -0.350129  …          0.051856
likes_count                       0.294066  …          0.956166
retweets_count                    0.200077  …          0.837978
replies_count                     0.264444  …          0.869253
platform_Amazon Prime             0.536595  …          0.400839
platform_Hulu                    -0.008961  …          0.147641
platform_Netflix                  0.306943  …          1.000000

[14 rows x 14 columns]
```


1.2.4 ANALYSIS #4: US text keyword analysis on Twitter data

[29]:

```python
###########################################################
# ANALYSIS #4: US text keyword analysis on Twitter data #
###########################################################

# Install punkt and stopwords
nltk.download('punkt')
```

```
nltk.download('stopwords')

# Convert the tweets from dataframe into a list of tweets
us_tweets_list = []
for index, row in us_tweets_all_platforms_df.iterrows():
        us_tweets_list.append(row['tweet'])

print('Length of us_tweets_list: ', len(us_tweets_list), '\n')

# Store all tokens from each tweet in us_tweets_list
tokens = [tok for tweet in us_tweets_list for tok in nltk.word_tokenize(tweet.
 ↪lower())]
print('Tokens Count: ', len(tokens), '\n')
print('First 30 Tokens: ', tokens[:30], '\n')

# Import in a list of english stopwords from nltk
stopwords = nltk.corpus.stopwords.words('english')

# Append netflix, primevideo, prime, and hulu to the stopwords list
# We didn't append "video" as a term to the stopwords list since it might be␣
 ↪used in other contexts outside of "prime video" as two words
stopwords.extend(['netflix', 'primevideo', 'prime',␣
 ↪'hulu','https',"n't",'shit'])
print('Length of Stopwords List: ', len(stopwords), '\n')

# Create a function to check if a word is all nonalpha characters OR in the␣
 ↪stopwords list
def filter_regex_and_stopwords(word, regex_string, stopwords_list):
        regex = re.compile(regex_string)
        if (regex.match(word)) and (word not in stopwords_list):
                return True
        else:
                return False

# Filter out stopwords and words with all nonalphabetic characters
tokens_filtered = []
for tok in tokens:
        if filter_regex_and_stopwords(tok, '[A-Za-z]', stopwords):
                tokens_filtered.append(tok)


# Create a freq distribution for the tokens
tokens_freq_dist = nltk.FreqDist(tokens_filtered)
top50_tokens = tokens_freq_dist.most_common(50)
print('Top 50 Tokens: \n', top50_tokens, '\n')

# Create a dataframe for the Top 50 Tokens
```

```
top50_tokens_df = pd.DataFrame(top50_tokens, columns = ['token', 'frequency'])
print('First 20 Rows of Top 50 Tokens Dataframe:\n', top50_tokens_df.head(20),␣
↪'\n')

top50_tokens_df
```

```
[nltk_data] Downloading package punkt to /root/nltk_data…
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package stopwords to /root/nltk_data…
[nltk_data]   Unzipping corpora/stopwords.zip.
Length of us_tweets_list:  107293

Tokens Count:  2486582

First 30 Tokens:  ['oh', 'at', 'dinner', ':', '@', 'netflix', 'should', 'have',
'a', 'dog', 'mode', 'that', 'auto', 'mutes', 'doorbells', ',', 'knocks', ',',
'and', 'barks', 'so', 'your', 'dog', 'doesn', ''', 't', 'go', 'crazy', 'while',
'you']

Length of Stopwords List:  186

Top 50 Tokens:
 [('watch', 13496), ('watching', 11847), ('show', 8494), ('good', 7576),
('like', 6799), ('amp', 5991), ('movie', 5670), ('season', 5427), ('new', 5128),
('one', 5110), ('need', 4273), ('watched', 4250), ('series', 4215), ('get',
4186), ('love', 4013), ('time', 3943), ('got', 3890), ('really', 3551), ('know',
3158), ('shows', 2940), ('see', 2774), ('episode', 2759), ('go', 2614),
('still', 2558), ('back', 2531), ('movies', 2529), ('documentary', 2505),
('great', 2448), ('would', 2399), ('tv', 2385), ('lol', 2353), ('people', 2347),
('think', 2340), ('seen', 2322), ('please', 2236), ('na', 2233), ('right',
2215), ('day', 2197), ('last', 2195), ('want', 2130), ('going', 2124), ('put',
2090), ('also', 1971), ('us', 1913), ('first', 1887), ('much', 1872), ('de',
1868), ('amazon', 1861), ('night', 1820), ('best', 1815)]

First 20 Rows of Top 50 Tokens Dataframe:
         token  frequency
0       watch      13496
1    watching      11847
2        show       8494
3        good       7576
4        like       6799
5         amp       5991
6       movie       5670
7      season       5427
8         new       5128
9         one       5110
10       need       4273
```

11	watched	4250
12	series	4215
13	get	4186
14	love	4013
15	time	3943
16	got	3890
17	really	3551
18	know	3158
19	shows	2940

[29]:

	token	frequency
0	watch	13496
1	watching	11847
2	show	8494
3	good	7576
4	like	6799
5	amp	5991
6	movie	5670
7	season	5427
8	new	5128
9	one	5110
10	need	4273
11	watched	4250
12	series	4215
13	get	4186
14	love	4013
15	time	3943
16	got	3890
17	really	3551
18	know	3158
19	shows	2940
20	see	2774
21	episode	2759
22	go	2614
23	still	2558
24	back	2531
25	movies	2529
26	documentary	2505
27	great	2448
28	would	2399
29	tv	2385
30	lol	2353
31	people	2347
32	think	2340
33	seen	2322
34	please	2236

35	na	2233
36	right	2215
37	day	2197
38	last	2195
39	want	2130
40	going	2124
41	put	2090
42	also	1971
43	us	1913
44	first	1887
45	much	1872
46	de	1868
47	amazon	1861
48	night	1820
49	best	1815

[30]:

```python
# Wordcloud!  What were people tweeting about...

# Convert the filtered tokens list to string using list comprehension
listToStr = ' '.join(map(str, tokens_filtered))
wordcloud = WordCloud(width = 1600, height = 1600,
                background_color ='white',
                stopwords = stopwords, max_words=200,
                min_font_size = 10).generate(listToStr)

plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```