Jared Yu

Data Visualization

Project 4

First Visualization (Ebola dataset)

The first visualization is based on a dataset from Kaggle that contains information about the spread of the Ebola virus. The Western African Ebola virus is an epidemic in the continent of Africa that spread from 2013 – 2016. The spread is mainly in the countries of Guinea, Liberia, and Sierra Leone. In these regions there has been great loss of life and damage to the socioeconomics. The fatality rate of the disease for those that were hospitalized was 57-59%. The first cases of this outbreak were in Guinea on December 2013. According to the page that hosts the data by Kaggle the user 'Devakumar kp,' the data seems to be sourced from the World Health Organization's website; however, a direct link isn't provided.

The range of the dates are from 2014 to 2016. The original dataset contains 10 variables and 2,485 rows. The variables include: Country, Date, No. of suspected cases, No. of probable cases, No. of confirmed cases, No. of probable and confirmed cases, No. of suspected deaths, No. of probable deaths, No. of confirmed deaths, and No. of confirmed, probable and suspected deaths. In most of the countries, there were a highly limited number of cases and the epidemic was primarily limited to three countries in Africa: Sierra Leone, Liberia, and Guinea. There are several other countries including European and the United States, however their numbers are comparatively so much fewer that they are noticeable only as small dots on the visualization. The visualization can be seen below in Figure 1.

The visualization is a Bubble Map from Python's Plot.ly package that includes an animation for timeseries data. The date starts from 2014 and extends to 2016, going on a roughly day-by-day basis (however there are gaps in dates). The changes aren't too noticeable since the focus is primarily on the north west tip of Africa where the epidemic seems to have originated. Over time, there are spreads of the cases to other parts of Africa, Europe, and the United States. However, it's only quite noticeable when you look for those countries specifically based on your knowledge of their respective locations. They appear mostly as dots, with counts of roughly between 1 and 20. These sizes are compared to the thousands that can be found in the three main countries.

Looking closely at the dataset, there are a set of possible variables that can be used for this visualization. The chosen variable is 'No. of probable and confirmed cases,' since out of them it seems to be the most interesting to use to understand the spread of the virus throughout the world. This variable describes the number of probable and confirmed cases of the Ebola virus for a country on a given date. Of course, it's also possible to use any of the others, for example, 'No. of confirmed deaths,' but the goal of this project is primarily to explore visualization tools and so there isn't a great deal of focus on the analysis of the data itself.

The variable 'Country' had to be altered through enough work to make the dataset conform to the parameters of the Plot.ly package. This variable gives the country for an observation that describes the number of cases of Ebola on a given date. Since the outbreak started in one area and spread elsewhere, not every country has the same number of records. To change things, all the countries were given an equal number of observations and if a country didn't have any cases prior to a certain date than it was given a 0 for those dates. In Liberia, there was also an issue of multiple values and so the logical one was used since it seemed that the second value was an error with the dataset. Some imputation necessarily had to be done since there were gaps in the dates for some countries. The solution was to let the previous date be the current number of cases up until it was updated with a new number of cases from a new date.

The variable 'Date' gives the exact day for an observation in the format of '2015-07-03', and makes this dataset fit nicely as a timeseries dataset. There are 259 unique days and so all 259 were used on the timeseries animation. This variable was important to take into consideration when imputing values and fixing the dataset so that it fit a certain format.

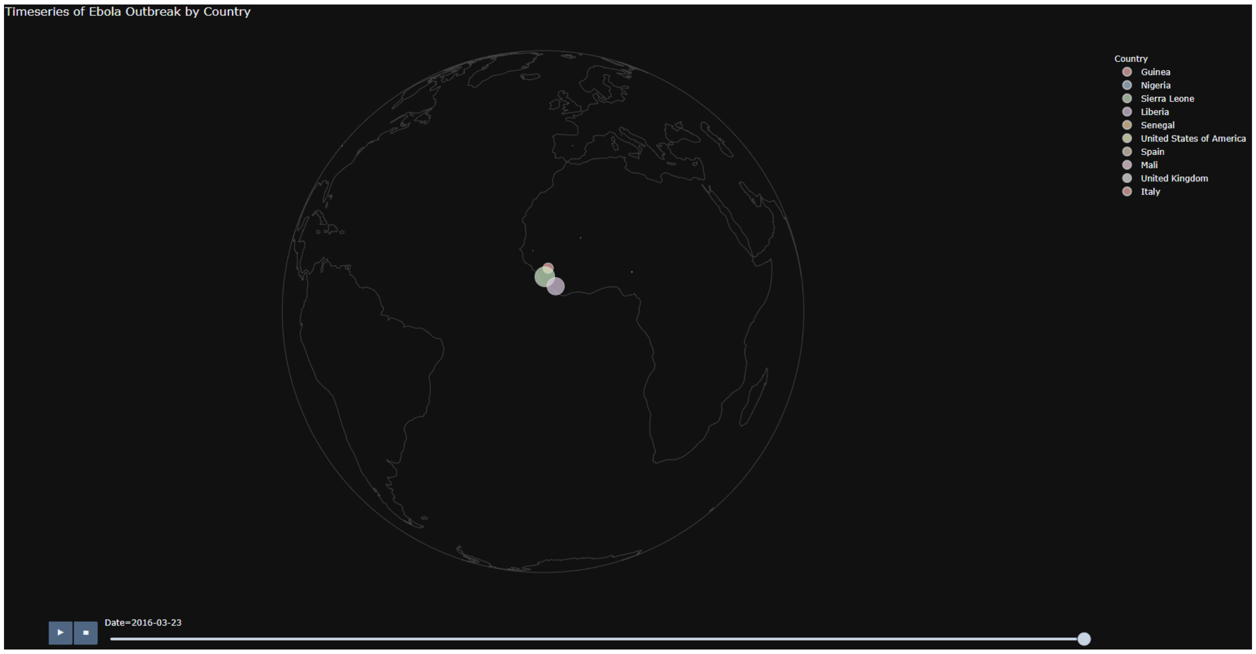
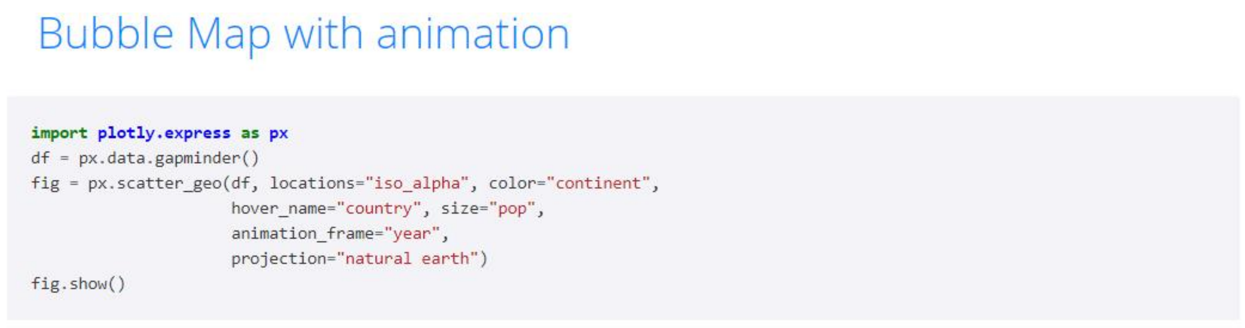


Figure 1

Some changes that were made from the original were the type of projection, template, size, title, and colors. Below in Figure 2 is a screenshot of the original based on source code. Some reasoning for the changes is that since the focus is on one small part of Africa, it doesn't make sense necessarily to focus on other areas. Also, since there are so few noticeable colors, having a strong contrast with a dark background helps to emphasize the important parts of the visualization. Furthermore, the visualization was increased to a height of 1,000 pixels so that it's possible to zoom-in and not lose context of the entire globe. The change of colors was more to experiment with the colors that Plot.ly has available to users. The default settings from the source code are fine and do well for the example, but there was an interest in trying out others that are possible.



Bubble Map with animation

```
import plotly.express as px
df = px.data.gapminder()
fig = px.scatter_geo(df, locations="iso_alpha", color="continent",
                     hover_name="country", size="pop",
                     animation_frame="year",
                     projection="natural earth")
fig.show()
```

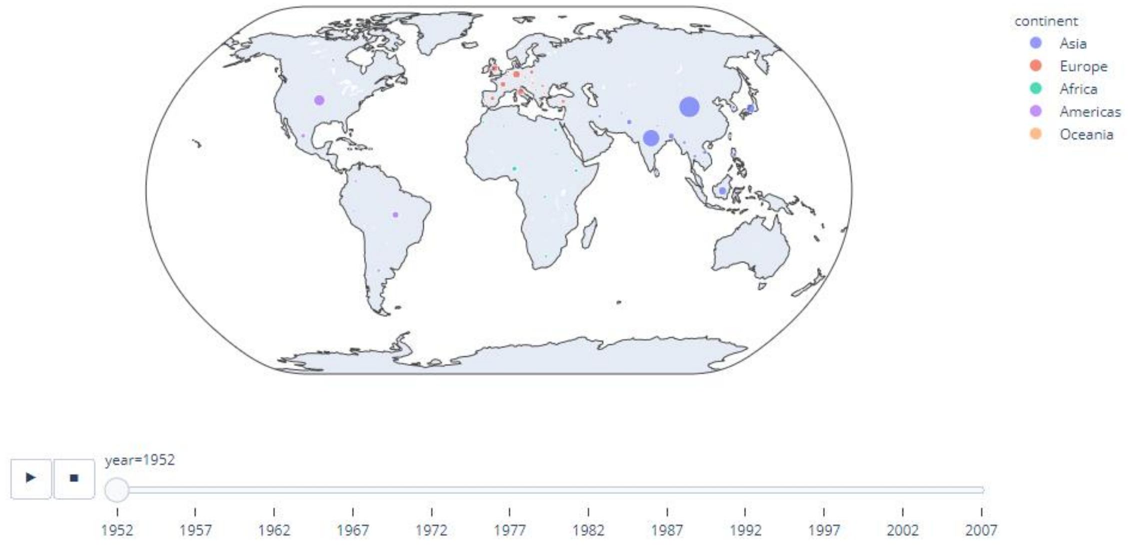


Figure 2

Second Visualization (European Football Goals by League)

The second visualization uses another dataset from Kaggle that contains information about Europe's football (soccer) games from 2008 to 2016. The dataset is contained within a database and so a SQL package was required in order to obtain the dataset that was used. The website Kaggle had a post in the forum by a user who showed the necessary SQL commands and they were used and adapted for this visualization.

There are over 25,000 matches, 10,000 players, 11 European countries, seasons 2008-2016, player and team attributes based on FIFA video games, squad formation for teams, betting odds, and detailed events for over 10,000 of the matches. There seems to be multiple sources for the entire database. The goal of the dataset largely is to identify winning odds for the purpose of possibly betting on games. This has all been provided by a user named 'Huge Mathien' 3 years ago on Kaggle. It's one of the most popular datasets with 2,480 'upvotes' and a ranking of gold. The usability is also rated at a 7.1/10. Many other users on Kaggle seem to also have benefited from this databased, with many kernels and discussions that explore the database further.

Within the database there are 8 different tables. The first table, ‘sqlite_sequence,’ is like a table of contents which shows the name of the other tables and the number of rows. In the table ‘Team’ there are 103,916 rows. In the tables ‘Country,’ ‘League,’ and ‘Team’ there are 51,958 rows. In the table ‘Player’ there are 11,075 rows. In the table ‘Player_Attributes’ there are 183,978 rows. Lastly, in the table ‘Team_Attributes’ there are 1,458 rows. In this visualization a dataset is created using SQL commands that utilizes information from the tables ‘Country,’ ‘League,’ and ‘Match’ to generate a dataset with 87 rows and 10 columns. This dataset is slightly modified from a Kaggle kernel by user, ‘Dima Rudov.’ It has been changed to include every European football league. However, ultimately only 3 of the columns are used. The variables are ‘league_name,’ ‘season,’ and ‘total_goals’ which are explained in the following paragraph.

The visualization technique used is that of a sunburst plot and it can be seen below in Figure 3. In the center are each of Europe’s main football leagues. Each of the countries: Spain, England, Italy, Netherlands, France, Germany, Portugal, Belgium, Scotland, Poland, and Switzerland are all represented with their corresponding top football leagues. These come from the variable called , ‘league_name.’ These are all parent nodes to the child nodes, where the child node is the total number of goals scored from all teams for each of the seasons ranging from the 2008/2009 season to the 2014/2015 season. The child node is called ‘season.’ The corresponding color and size of each node is based on the variable ‘total_goals’ that is detailed per season per league. The color is based on a continuous scale and shows that the most goals were in the Spanish league from 2008/2009 with 1,101 goals and the least goals were in Switzerland’s league with 425 goals from 2011/2012. It’s difficult however to draw any abstractions about skill since it makes sense that there are more teams/games in the Spanish league compared to the Swiss league. Also, simply having more goals doesn’t make a league better, perhaps their defense is also weaker too.

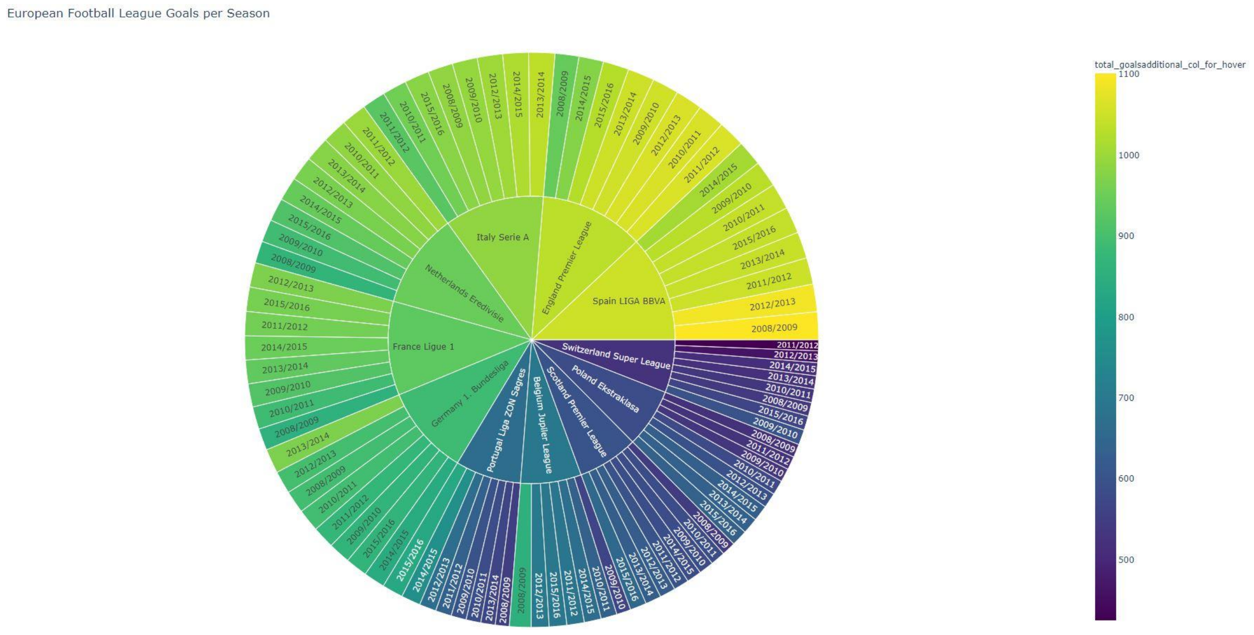
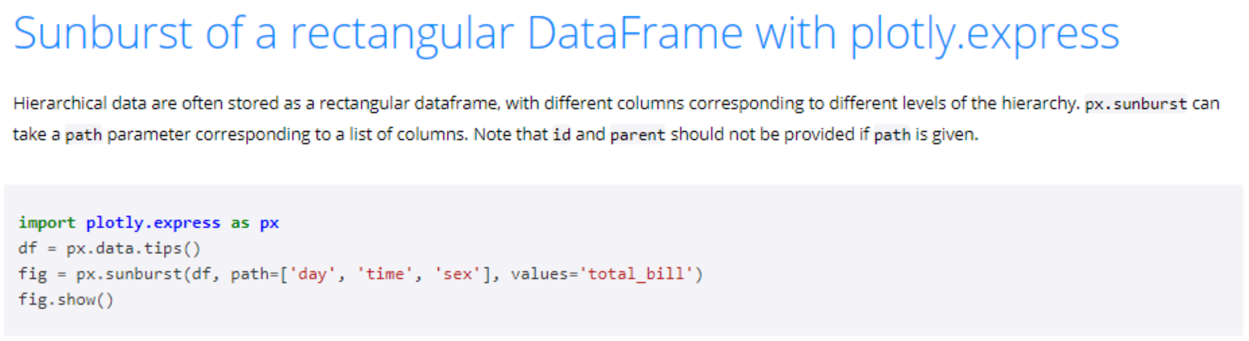


Figure 3

The original visualization from the source code can be seen below in Figure 4. In this situation, the colors are discrete and based on a value such as what day it is. For that given dataset it makes sense, since it’s much simpler to look at a relatively few slices and analyze their respective totals. However, in the European football league dataset, it’s not as simple since there are so many slices and they’re also all roughly the same size. To better visualize this type of data, a continuous color sequence is utilized to better help understand the data. On the right side is the legend that corresponds to the colors going from light to dark (high to low). An issue was that of the title for the plot and the legend where it was difficult to reformat them. An attempt was made to do so, but it wasn’t found out the proper method to go about completing the task. Another change was that the visualization was changed to a height of 1,000 pixels so that it’s more practical to analyze the many different slices of the sunburst visualization.



Sunburst of a rectangular DataFrame with plotly.express

Hierarchical data are often stored as a rectangular dataframe, with different columns corresponding to different levels of the hierarchy. `px.sunburst` can take a `path` parameter corresponding to a list of columns. Note that `id` and `parent` should not be provided if `path` is given.

```python
import plotly.express as px
df = px.data.tips()
fig = px.sunburst(df, path=['day', 'time', 'sex'], values='total_bill')
fig.show()
```

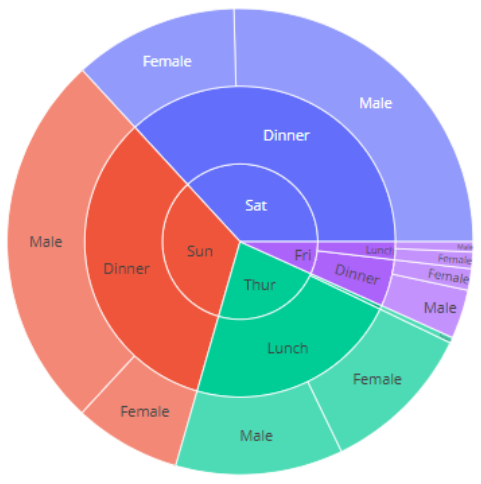


Figure 4

Third Visualization (Netflix movies)

The third and last visualization is based on another dataset from Kaggle that contains information about movies in the Netflix database. It has been posted by a user named, ‘Shivam Bansal’ and it was updated recently 2 months ago. The dataset has 648 ‘upvotes’ and a ranking of gold. The dataset gives information about TV shows and movies listed on Netflix as of 2019. The dataset has a usability score of 10/10 on the website Kaggle.

The dataset contains 6,234 rows and 12 variables. The variables include: show_id, type, title, director, cast, country, date_added, release_year, rating, duration, listed_in, and description. The variables used in the visualization are 'type', 'duration', 'country', and 'release_year'. The 'type' variable corresponds with whether the observation is a TV show or a movie. The 'duration' variable describes how many minutes a movie will last. The 'country' variable describes which country the movie originated in. The 'release_year' variable describes what year the movie originally was released in theaters.

Below in Figure 5 is the parallel categories diagram based off some source code from Plot.ly's website. From the screenshot it isn't as clear to see, but all the variables had been altered to some extent so that the visualization is simpler and less cluttered. If all the variables were left as they were, there would have been too many discrete objects in the visualization and therefore it would've been impossible to interact or interpret. On the left-hand side is the 'Continent' column which has been altered from the 'country' variable. Originally, one or more countries had been listed per observation. To overcome the obstacle of there being more than one possible country, only the first country was used in cases where there were multiple. Another change was to create a broader category of continent so that each of the countries would correspond only to one of the set of continents: North America, Europe, South America, Asia, Africa, or Oceania. In this way it's less cluttered and simpler to navigate the visualization.

The column in the middle is 'Release Date' and is altered from the variable 'release_year.' Originally, each of the observations had the exact date of release with month, day, and year. This was first changed by taking the year only from this column. Then, the years were then mapped to their corresponding decade. Therefore, the release date only mentions the decade that the movie came out. In this way, there are far fewer categories for this column and therefore the visualization again is simpler and cleaner to interact with. The other situation would involve thousands of discrete dates and the visualization is no longer possible to navigate.

The last column is 'Length (minutes).' Originally, I was interested in listing both movies and TV shows for the visualization. However, movies had a value in number of minutes while TV shows had a value for number of seasons. Since these two didn't merge well, I chose to use only movies. From the variable 'duration,' first the letters ' min' representing number of minutes were removed and then the remaining digits were converted to integer data types. In this case, there are still too many discrete values and so they were all reduced to their ten's place. For example, a 65-minute movie was changed to 60 minutes. In one case a duration was reduced to 0 minutes and so it had to be manually changed to 10 minutes.

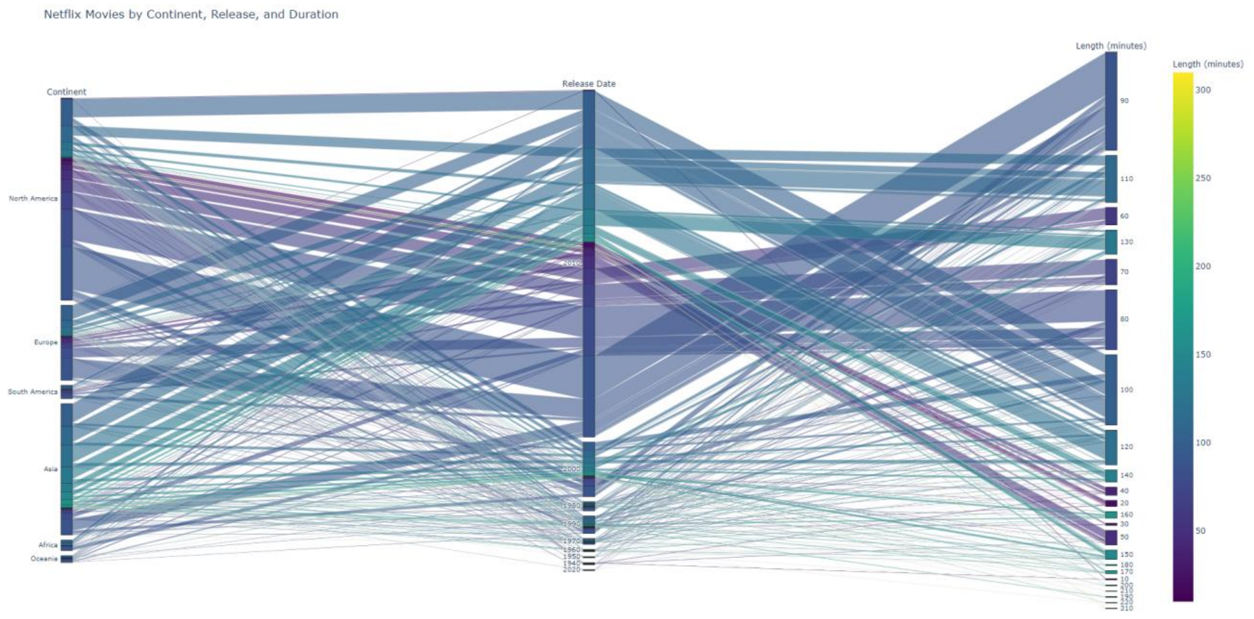


Figure 5

Below in Figure 6 is the original visualization from the source code based off the Plot.ly website. One of the changes is that the height had been increased to 1,000 pixels so that it's easier to navigate. With the large number of discrete objects, it's important that they're large enough to be interacted with using a mouse on a screen. A title was also added and the labeling for each of the categories were much simpler to complete in the visualization (a task that couldn't be done in the second visualization). Another change is that of the continuous color sequence was changed to be the same type that was used in the previous visualization with European football leagues. In this visualization, it seemed not too many changes could be done like was done with previous visualizations where more options were available.

Style Diagram

In this example `dimensions` represents a list of stings or the columns of data frame, and `labels` is a dictionary with string keys (column name) and string values ('desired label to be displayed'). See Plotly express reference page for more information.

```python
import plotly.express as px

df = px.data.tips()
fig = px.parallel_categories(df, dimensions=['sex', 'smoker', 'day'],
                color="size", color_continuous_scale=px.colors.sequential.Inferno,
                labels={'sex':'Payer sex', 'smoker':'Smokers at the table', 'day':'Day of week'})
fig.show()
```

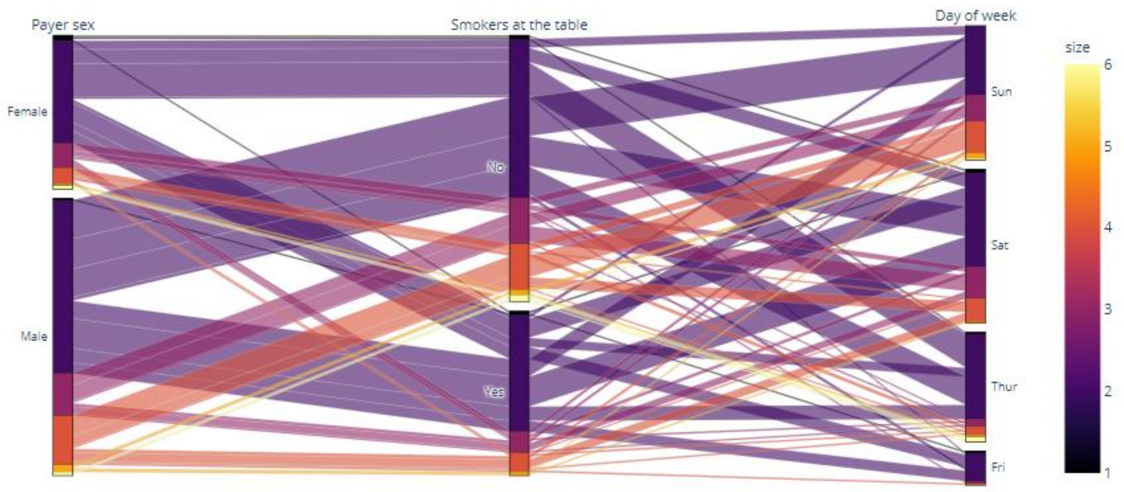


Figure 6

References:

https://plot.ly/python/bubble-maps/

https://www.kaggle.com/imdevskp/ebola-outbreak-20142016-complete-dataset

https://plot.ly/python/sunburst-charts/

https://www.kaggle.com/hugomathien/soccer

https://plot.ly/python/parallel-categories-diagram/

https://www.kaggle.com/shivamb/netflix-shows