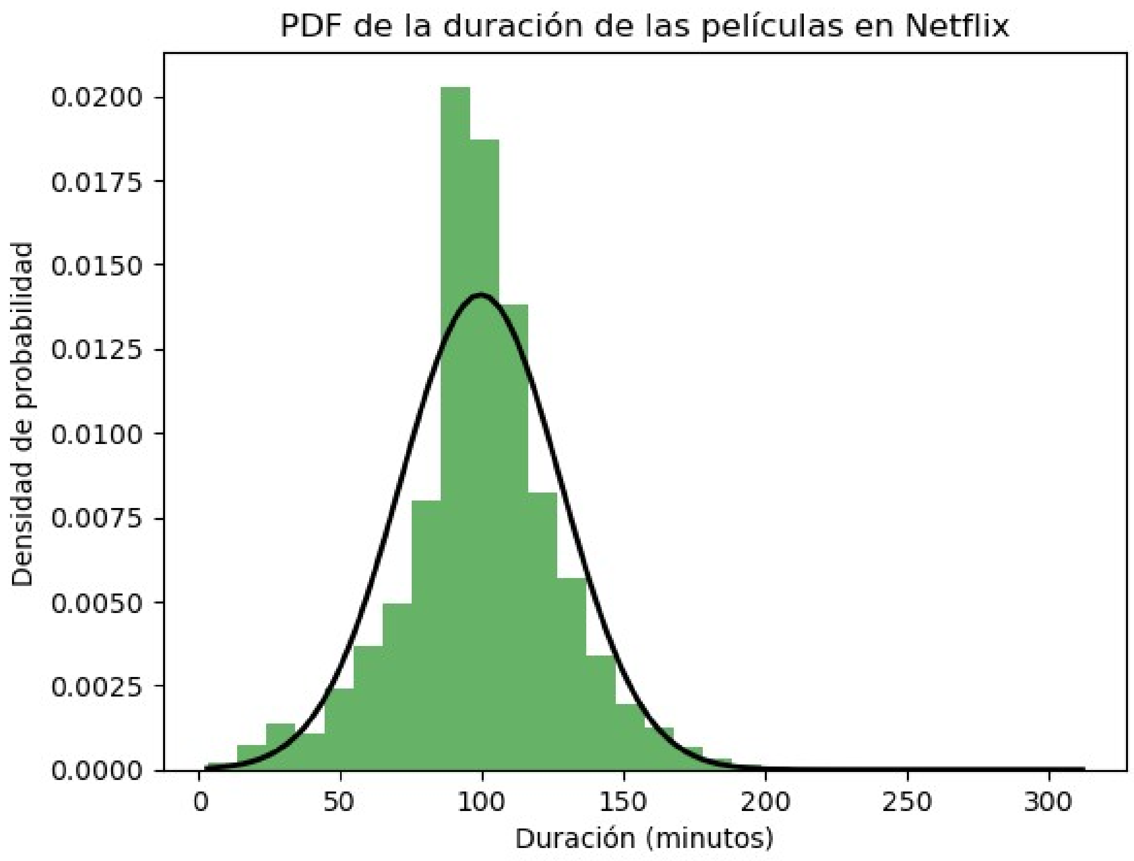
```python
import pandas as pd

df = pd.read_csv(r"C:\Users\Usuario\Desktop\DataScience\
netflix_titles.csv\netflix_titles.csv")
df
```

```
     show_id     type                  title         director  \
0         s1    Movie   Dick Johnson Is Dead  Kirsten Johnson
1         s2  TV Show          Blood & Water              NaN
2         s3  TV Show              Ganglands  Julien Leclercq
3         s4  TV Show  Jailbirds New Orleans              NaN
4         s5  TV Show           Kota Factory              NaN
...      ...      ...                    ...              ...
8802   s8803    Movie                 Zodiac    David Fincher
8803   s8804  TV Show            Zombie Dumb              NaN
8804   s8805    Movie             Zombieland  Ruben Fleischer
8805   s8806    Movie                   Zoom     Peter Hewitt
8806   s8807    Movie                 Zubaan      Mozez Singh

                                                   cast        country
\
0                                                   NaN  United States

1     Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban...   South Africa

2     Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nabi...            NaN

3                                                   NaN            NaN

4     Mayur More, Jitendra Kumar, Ranjan Raj, Alam K...          India

...                                                 ...            ...

8802  Mark Ruffalo, Jake Gyllenhaal, Robert Downey J...  United States

8803                                                NaN            NaN

8804  Jesse Eisenberg, Woody Harrelson, Emma Stone, ...  United States

8805  Tim Allen, Courteney Cox, Chevy Chase, Kate Ma...  United States

8806  Vicky Kaushal, Sarah-Jane Dias, Raaghav Chanan...          India


              date_added  release_year rating   duration  \
0     September 25, 2021          2020  PG-13     90 min
1     September 24, 2021          2021  TV-MA  2 Seasons
2     September 24, 2021          2021  TV-MA   1 Season
3     September 24, 2021          2021  TV-MA   1 Season
4     September 24, 2021          2021  TV-MA  2 Seasons
...                  ...           ...    ...        ...
```

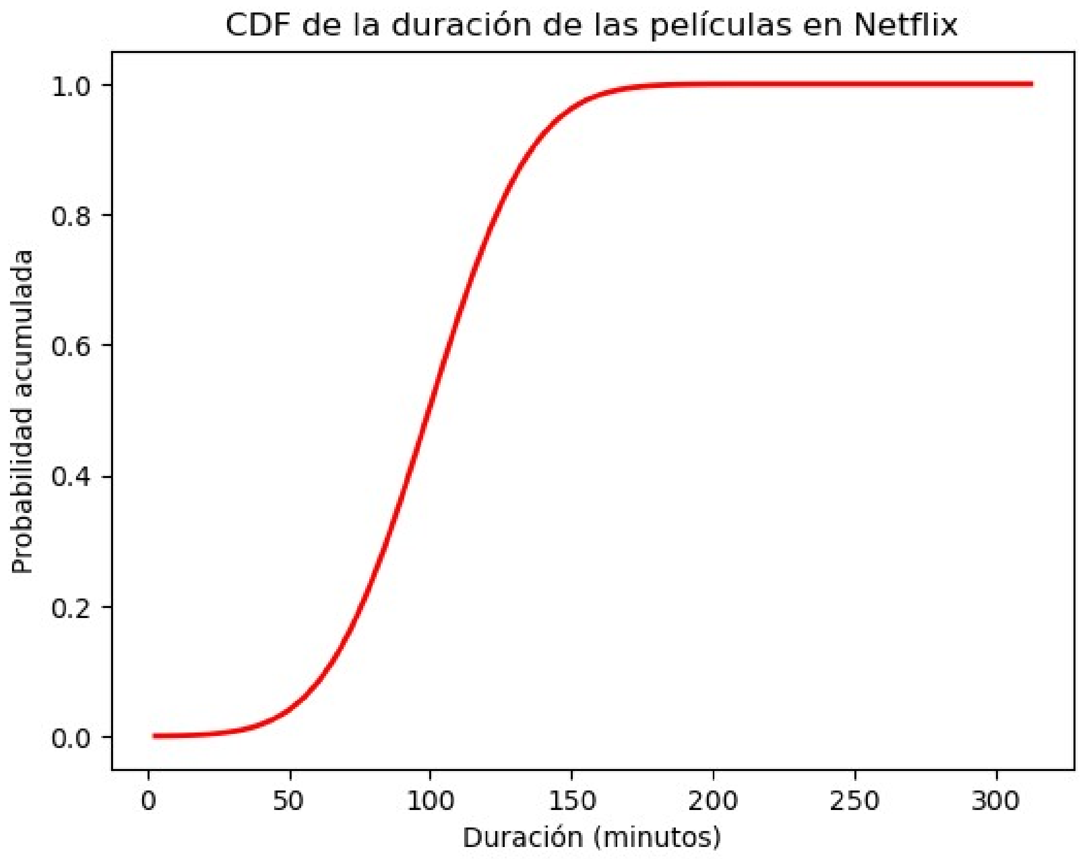
```
8802   November 20, 2019          2007      R    158 min
8803        July 1, 2019          2018  TV-Y7  2 Seasons
8804    November 1, 2019          2009      R     88 min
8805    January 11, 2020          2006     PG     88 min
8806       March 2, 2019          2015  TV-14    111 min

                                              listed_in  \
0                                         Documentaries
1       International TV Shows, TV Dramas, TV Mysteries
2     Crime TV Shows, International TV Shows, TV Act...
3                                Docuseries, Reality TV
4     International TV Shows, Romantic TV Shows, TV ...
...                                                 ...
8802                     Cult Movies, Dramas, Thrillers
8803             Kids' TV, Korean TV Shows, TV Comedies
8804                            Comedies, Horror Movies
8805                Children & Family Movies, Comedies
8806     Dramas, International Movies, Music & Musicals

                                            description
0     As her father nears the end of his life, filmm...
1     After crossing paths at a party, a Cape Town t...
2     To protect his family from a powerful drug lor...
3     Feuds, flirtations and toilet talk go down amo...
4     In a city of coaching centers known to train I...
...                                                 ...
8802  A political cartoonist, a crime reporter and a...
8803  While living alone in a spooky town, a young g...
8804  Looking to survive in a world taken over by zo...
8805  Dragged from civilian life, a former superhero...
8806  A scrappy but poor boy worms his way into a ty...

[8807 rows x 12 columns]

# Eliminar filas con valores nulos en las columnas seleccionadas
df = df.dropna(subset=['release_year', 'duration'])

# Filtrar solo las filas donde 'duration' es numérico
df = df[df['duration'].str.contains('min')]

# Convertir la columna 'duration' a un formato numérico
df['duration'] = df['duration'].str.replace(' min', '').astype(int)  # 
Ahora solo contiene valores numéricos

# Nota: Asegúrate de que 'duration' solo contenga valores que se 
puedan convertir a int

import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm
```

```
# Filtrar solo las películas
movies = df[df['type'] == 'Movie']

# Calcular la PDF
mu, std = norm.fit(movies['duration'])
pdf = norm.pdf(np.linspace(movies['duration'].min(),
movies['duration'].max(), 100), mu, std)

# Graficar la PDF
plt.hist(movies['duration'], bins=30, density=True, alpha=0.6,
color='g')
plt.plot(np.linspace(movies['duration'].min(),
movies['duration'].max(), 100), pdf, 'k', linewidth=2)
plt.title('PDF de la duración de las películas en Netflix')
plt.xlabel('Duración (minutos)')
plt.ylabel('Densidad de probabilidad')
plt.show()
```

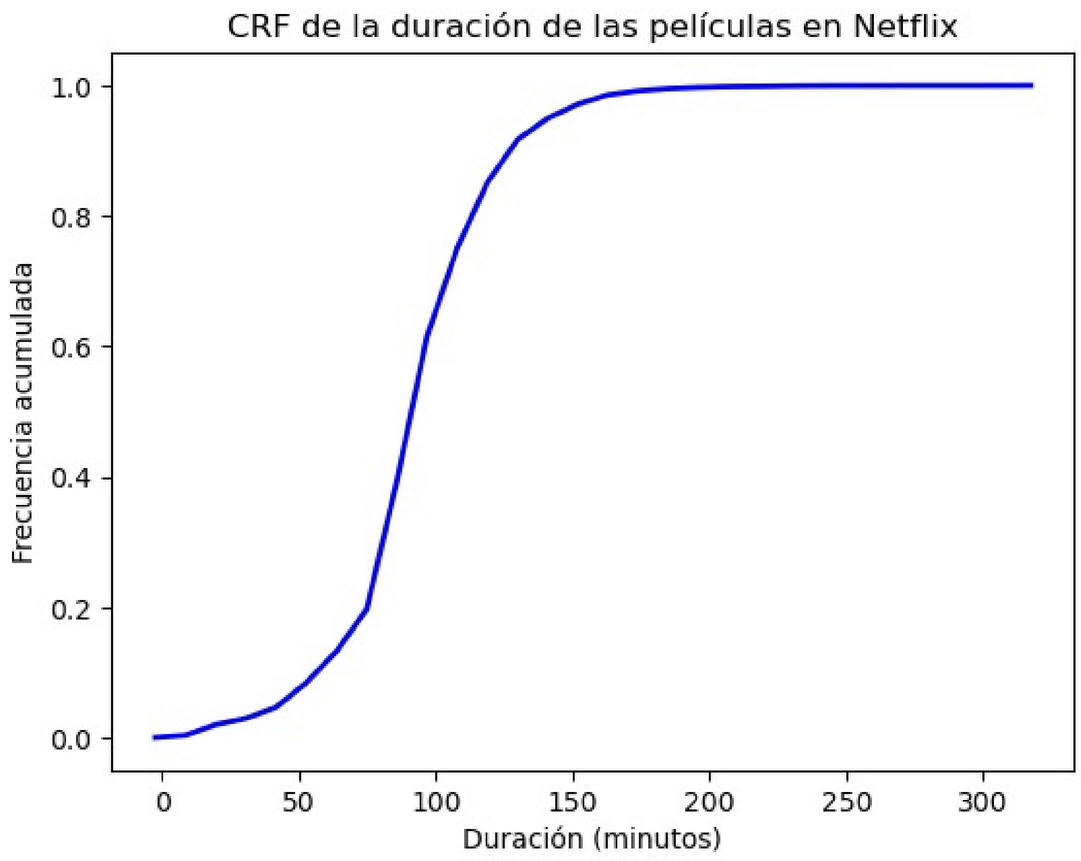


```
# Calcular la CDF
cdf = norm.cdf(np.linspace(movies['duration'].min(),
movies['duration'].max(), 100), mu, std)
```

```python
# Graficar la CDF
plt.plot(np.linspace(movies['duration'].min(),
movies['duration'].max(), 100), cdf, 'r', linewidth=2)
plt.title('CDF de la duración de las películas en Netflix')
plt.xlabel('Duración (minutos)')
plt.ylabel('Probabilidad acumulada')
plt.show()
```



```python
from scipy.stats import cumfreq

# Calcular la frecuencia acumulativa
res = cumfreq(movies['duration'], numbins=30)

# Graficar la CRF
x = res.lowerlimit + np.linspace(0, res.binsize*res.cumcount.size,
res.cumcount.size)
plt.plot(x, res.cumcount/len(movies['duration']), 'b', linewidth=2)
plt.title('CRF de la duración de las películas en Netflix')
plt.xlabel('Duración (minutos)')
plt.ylabel('Frecuencia acumulada')
plt.show()
```



```
df = pd.read_csv(r'C:\Users\Usuario\Desktop\DataScience\
netflix_titles.csv\netflix_titles.csv')


df['country'] = df['country'].fillna(df['country'].mode()[0])


df['cast'].replace(np.nan, 'No Data',inplace  = True)
df['director'].replace(np.nan, 'No Data',inplace  = True)

# Drops

df.dropna(inplace=True)

# Drop Duplicates

df.drop_duplicates(inplace= True)

df.isnull().sum()
```

```
show_id         0
type            0
title           0
director        0
```

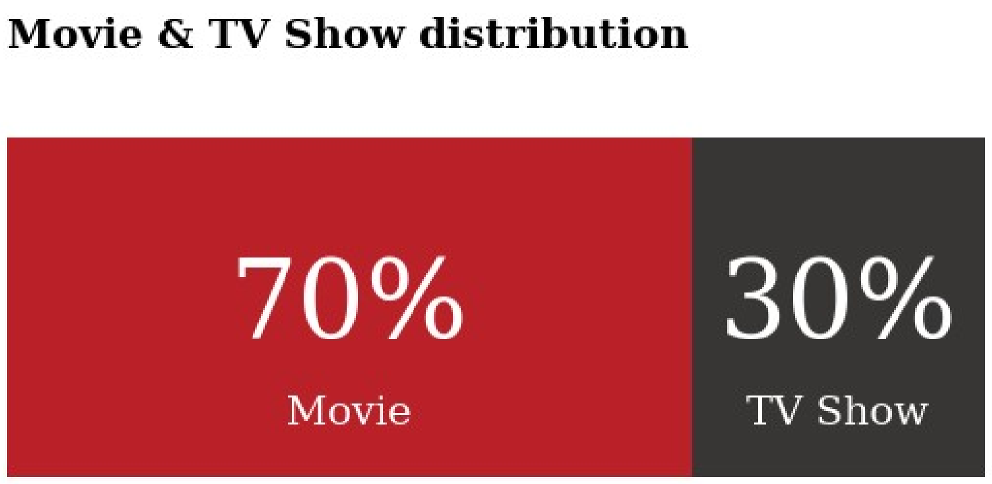
```
cast            0
country         0
date_added      0
release_year    0
rating          0
duration        0
listed_in       0
description     0
dtype: int64
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 8790 entries, 0 to 8806
Data columns (total 12 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   show_id       8790 non-null   object
 1   type          8790 non-null   object
 2   title         8790 non-null   object
 3   director      8790 non-null   object
 4   cast          8790 non-null   object
 5   country       8790 non-null   object
 6   date_added    8790 non-null   object
 7   release_year  8790 non-null   int64 
 8   rating        8790 non-null   object
 9   duration      8790 non-null   object
 10  listed_in     8790 non-null   object
 11  description   8790 non-null   object
dtypes: int64(1), object(11)
memory usage: 892.7+ KB
```

```
# For viz: Ratio of Movies & TV shows

x=df.groupby(['type'])['type'].count()
y=len(df)
r=((x/y)).round(2)

mf_ratio = pd.DataFrame(r).T

fig, ax = plt.subplots(1,1,figsize=(6.5, 2.5))

ax.barh(mf_ratio.index, mf_ratio['Movie'],
        color='#b20710', alpha=0.9, label='Male')
ax.barh(mf_ratio.index, mf_ratio['TV Show'], left=mf_ratio['Movie'],
        color='#221f1f', alpha=0.9, label='Female')

ax.set_xlim(0, 1)
ax.set_xticks([])
ax.set_yticks([])
```

```
#ax.set_yticklabels(mf_ratio.index, fontfamily='serif', fontsize=11)


# movie percentage
for i in mf_ratio.index:
    ax.annotate(f"{int(mf_ratio['Movie'][i]*100)}%",
                   xy=(mf_ratio['Movie'][i]/2, i),
                   va = 'center', ha='center',fontsize=40,
fontweight='light', fontfamily='serif',
                   color='white')

    ax.annotate("Movie",
                   xy=(mf_ratio['Movie'][i]/2, -0.25),
                   va = 'center', ha='center',fontsize=15,
fontweight='light', fontfamily='serif',
                   color='white')


for i in mf_ratio.index:
    ax.annotate(f"{int(mf_ratio['TV Show'][i]*100)}%",
                   xy=(mf_ratio['Movie'][i]+mf_ratio['TV Show'][i]/2,
i),
                   va = 'center', ha='center',fontsize=40,
fontweight='light', fontfamily='serif',
                   color='white')
    ax.annotate("TV Show",
                   xy=(mf_ratio['Movie'][i]+mf_ratio['TV Show'][i]/2,
-0.25),
                   va = 'center', ha='center',fontsize=15,
fontweight='light', fontfamily='serif',
                   color='white')






# Title & Subtitle
fig.text(0.125,1.03,'Movie & TV Show distribution',
fontfamily='serif',fontsize=15, fontweight='bold')

for s in ['top', 'left', 'right', 'bottom']:
    ax.spines[s].set_visible(False)


# Removing legend due to labelled plot
ax.legend().set_visible(False)
plt.show()
```

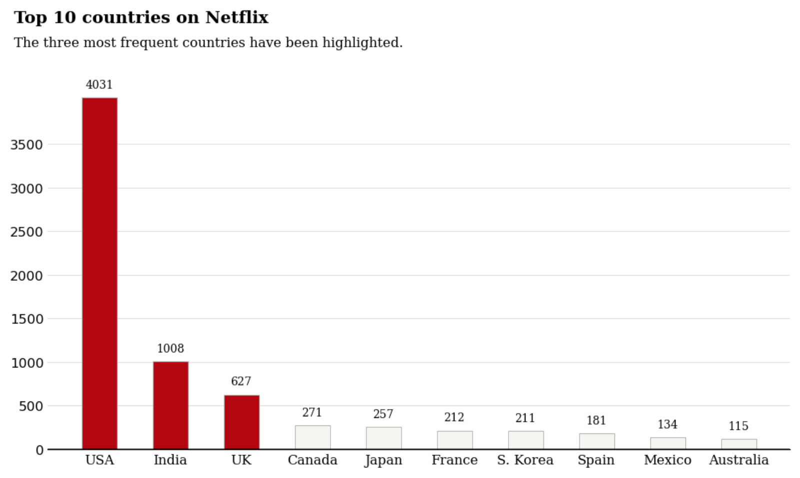


```
# Helper column for various plots
df['count'] = 1

# Many productions have several countries listed - this will skew our
results , we'll grab the first one mentioned

# Lets retrieve just the first country
df['first_country'] = df['country'].apply(lambda x: x.split(",")[0])
df['first_country'].head()

# Rating ages from this notebook: https://www.kaggle.com/andreshg/eda-
beginner-to-expert-plotly (thank you!)

ratings_ages = {
    'TV-PG': 'Older Kids',
    'TV-MA': 'Adults',
    'TV-Y7-FV': 'Older Kids',
    'TV-Y7': 'Older Kids',
    'TV-14': 'Teens',
    'R': 'Adults',
    'TV-Y': 'Kids',
    'NR': 'Adults',
    'PG-13': 'Teens',
    'TV-G': 'Kids',
    'PG': 'Older Kids',
    'G': 'Kids',
    'UR': 'Adults',
    'NC-17': 'Adults'
}

df['target_ages'] = df['rating'].replace(ratings_ages)
df['target_ages'].unique()

# Genre
```

```python
df['genre'] = df['listed_in'].apply(lambda x :
x.replace(' ,',',').replace(', ',',').split(','))

# Reducing name length

df['first_country'].replace('United States', 'USA', inplace=True)
df['first_country'].replace('United Kingdom', 'UK',inplace=True)
df['first_country'].replace('South Korea', 'S. Korea',inplace=True)


data = df.groupby('first_country')
['count'].sum().sort_values(ascending=False)[:10]

# Plot

color_map = ['#f5f5f1' for _ in range(10)]
color_map[0] = color_map[1] = color_map[2] =  '#b20710' # color
highlight

fig, ax = plt.subplots(1,1, figsize=(12, 6))
ax.bar(data.index, data, width=0.5,
       edgecolor='darkgray',
       linewidth=0.6,color=color_map)

#annotations
for i in data.index:
    ax.annotate(f"{data[i]}",
                   xy=(i, data[i] + 150), #i like to change this to
roughly 5% of the highest cat
                   va = 'center', ha='center',fontweight='light',
fontfamily='serif')



# Remove border from plot

for s in ['top', 'left', 'right']:
    ax.spines[s].set_visible(False)

# Tick labels

ax.set_xticklabels(data.index, fontfamily='serif', rotation=0)

# Title and sub-title

fig.text(0.09, 1, 'Top 10 countries on Netflix', fontsize=15,
fontweight='bold', fontfamily='serif')
fig.text(0.09, 0.95, 'The three most frequent countries have been
highlighted.', fontsize=12, fontweight='light', fontfamily='serif')
```

```python
fig.text(1.1, 1.01, 'Insight', fontsize=15, fontweight='bold',
fontfamily='serif')

fig.text(1.1, 0.67, '''
The most prolific producers of
content for Netflix are, primarily,
the USA, with India and the UK
a significant distance behind.

It makes sense that the USA produces
the most content as, afterall,
Netflix is a US company.
'''
         , fontsize=12, fontweight='light', fontfamily='serif')

ax.grid(axis='y', linestyle='-', alpha=0.4)

grid_y_ticks = np.arange(0, 4000, 500) # y ticks, min, max, then step
ax.set_yticks(grid_y_ticks)
ax.set_axisbelow(True)

#Axis labels

#plt.xlabel("Country", fontsize=12, fontweight='light',
fontfamily='serif',loc='left',y=-1.5)
#plt.ylabel("Count", fontsize=12, fontweight='light',
fontfamily='serif')
 #plt.legend(loc='upper right')

# thicken the bottom line if you want to
plt.axhline(y = 0, color = 'black', linewidth = 1.3, alpha = .7)

ax.tick_params(axis='both', which='major', labelsize=12)


import matplotlib.lines as lines
l1 = lines.Line2D([1, 1], [0, 1], transform=fig.transFigure,
figure=fig,color='black',lw=0.2)
fig.lines.extend([l1])

ax.tick_params(axis=u'both', which=u'both',length=0)

plt.show()
```

```
C:\Users\Usuario\AppData\Local\Temp\ipykernel_96516\960872908.py:70:
UserWarning: set_ticklabels() should only be used with a fixed number
of ticks, i.e. after set_ticks() or using a FixedLocator.
  ax.set_xticklabels(data.index, fontfamily='serif', rotation=0)
```

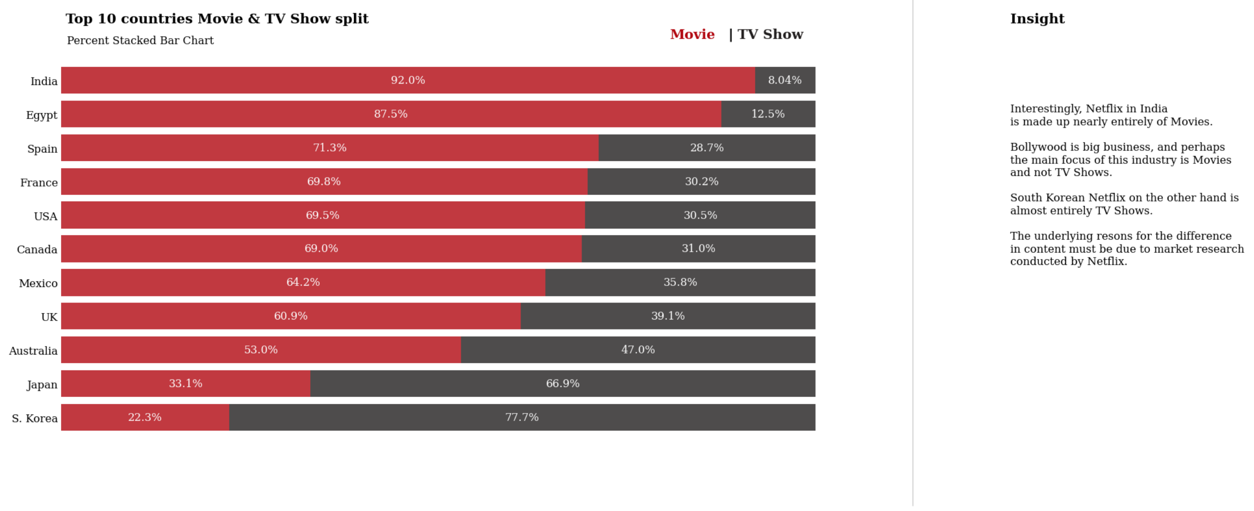


Insight

The most prolific producers of
content for Netflix are, primarily,
the USA, with India and the UK
a significant distance behind.

It makes sense that the USA produces
the most content as, afterall,
Netflix is a US company.

```
country_order = df['first_country'].value_counts()[:11].index
data_q2q3 = df[['type', 'first_country']].groupby('first_country')
['type'].value_counts().unstack().loc[country_order]
data_q2q3['sum'] = data_q2q3.sum(axis=1)
data_q2q3_ratio = (data_q2q3.T / data_q2q3['sum']).T[['Movie', 'TV
Show']].sort_values(by='Movie',ascending=False)[::-1]




###
fig, ax = plt.subplots(1,1,figsize=(15, 8),)

ax.barh(data_q2q3_ratio.index, data_q2q3_ratio['Movie'],
        color='#b20710', alpha=0.8, label='Movie')
ax.barh(data_q2q3_ratio.index, data_q2q3_ratio['TV Show'],
left=data_q2q3_ratio['Movie'],
        color='#221f1f', alpha=0.8, label='TV Show')


ax.set_xlim(0, 1)
ax.set_xticks([])
ax.set_yticklabels(data_q2q3_ratio.index, fontfamily='serif',
fontsize=11)

# male percentage
for i in data_q2q3_ratio.index:
    ax.annotate(f"{data_q2q3_ratio['Movie'][i]*100:.3}%",
                   xy=(data_q2q3_ratio['Movie'][i]/2, i),
                   va = 'center', ha='center',fontsize=12,
fontweight='light', fontfamily='serif',
                   color='white')

for i in data_q2q3_ratio.index:
    ax.annotate(f"{data_q2q3_ratio['TV Show'][i]*100:.3}%",
                   xy=(data_q2q3_ratio['Movie'][i]+data_q2q3_ratio['TV
Show'][i]/2, i),
                   va = 'center', ha='center',fontsize=12,
fontweight='light', fontfamily='serif',
                   color='white')


fig.text(0.13, 0.93, 'Top 10 countries Movie & TV Show split',
fontsize=15, fontweight='bold', fontfamily='serif')
fig.text(0.131, 0.89, 'Percent Stacked Bar Chart',
fontsize=12,fontfamily='serif')

for s in ['top', 'left', 'right', 'bottom']:
    ax.spines[s].set_visible(False)

#ax.legend(loc='lower center', ncol=3, bbox_to_anchor=(0.5, -0.06))

fig.text(0.75,0.9,"Movie", fontweight="bold", fontfamily='serif',
fontsize=15, color='#b20710')
fig.text(0.81,0.9,"|", fontweight="bold", fontfamily='serif',
fontsize=15, color='black')
fig.text(0.82,0.9,"TV Show", fontweight="bold", fontfamily='serif',
fontsize=15, color='#221f1f')


fig.text(1.1, 0.93, 'Insight', fontsize=15, fontweight='bold',
fontfamily='serif')

fig.text(1.1, 0.44, '''
Interestingly, Netflix in India
is made up nearly entirely of Movies.

Bollywood is big business, and perhaps
the main focus of this industry is Movies
and not TV Shows.

South Korean Netflix on the other hand is
almost entirely TV Shows.

The underlying resons for the difference
in content must be due to market research
conducted by Netflix.
'''
         , fontsize=12, fontweight='light', fontfamily='serif')



import matplotlib.lines as lines
l1 = lines.Line2D([1, 1], [0, 1], transform=fig.transFigure,
figure=fig,color='black',lw=0.2)
fig.lines.extend([l1])




ax.tick_params(axis='both', which='major', labelsize=12)
ax.tick_params(axis=u'both', which=u'both',length=0)

plt.show()

C:\Users\Usuario\AppData\Local\Temp\ipykernel_96516\532488442.py:20:
UserWarning: set_ticklabels() should only be used with a fixed number
of ticks, i.e. after set_ticks() or using a FixedLocator.
  ax.set_yticklabels(data_q2q3_ratio.index, fontfamily='serif',
fontsize=11)
```



Interestingly, Netflix in India
is made up nearly entirely of Movies.

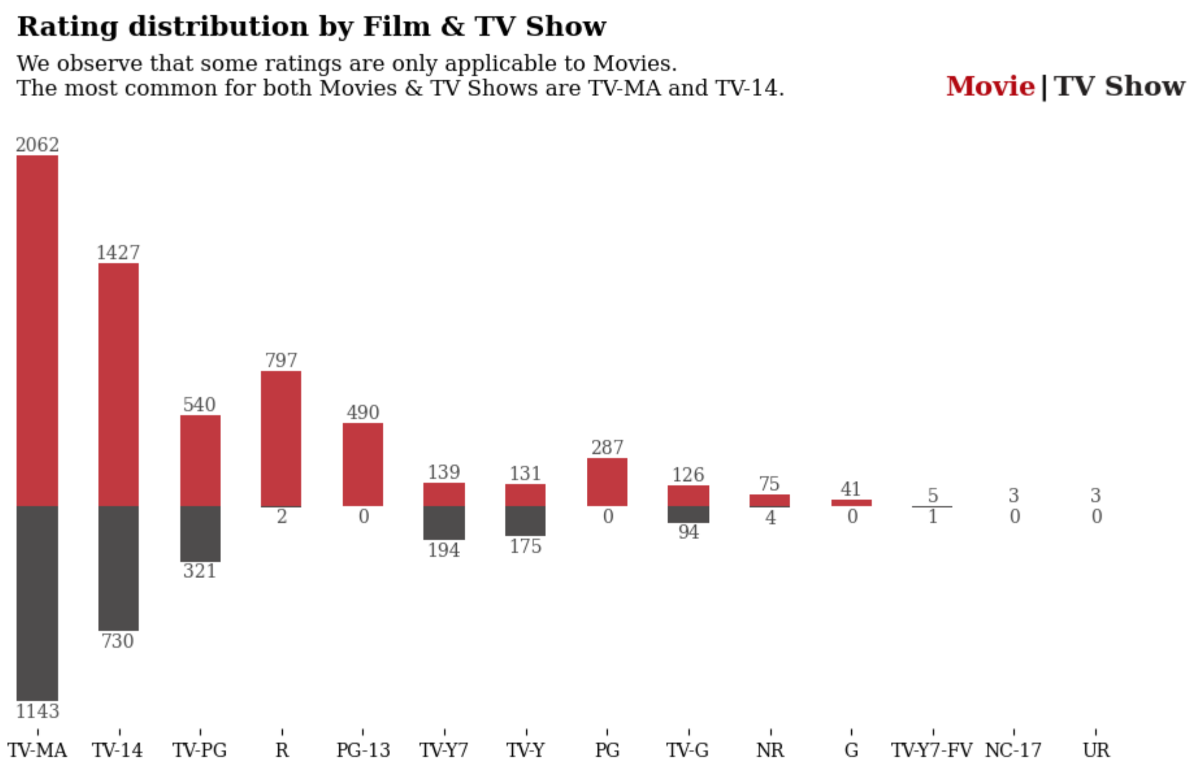
Bollywood is big business, and perhaps
the main focus of this industry is Movies
and not TV Shows.

South Korean Netflix on the other hand is
almost entirely TV Shows.

The underlying resons for the difference
in content must be due to market research
conducted by Netflix.

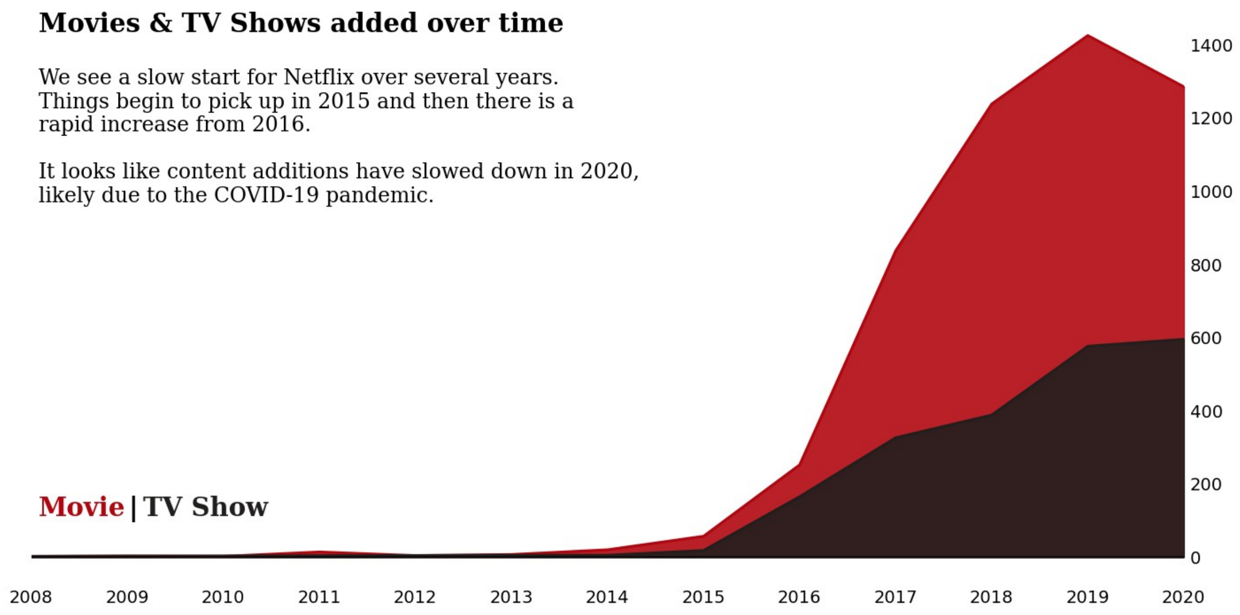
```
order = pd.DataFrame(df.groupby('rating')
['count'].sum().sort_values(ascending=False).reset_index())
rating_order = list(order['rating'])

mf = df.groupby('type')
['rating'].value_counts().unstack().sort_index().fillna(0).astype(int)
[rating_order]

movie = mf.loc['Movie']
tv = - mf.loc['TV Show']


fig, ax = plt.subplots(1,1, figsize=(12, 6))
ax.bar(movie.index, movie, width=0.5, color='#b20710', alpha=0.8,
label='Movie')
ax.bar(tv.index, tv, width=0.5, color='#221f1f', alpha=0.8, label='TV
Show')
#ax.set_ylim(-35, 50)

# Annotations
for i in tv.index:
    ax.annotate(f"{-tv[i]}",
                   xy=(i, tv[i] - 60),
                   va = 'center', ha='center',fontweight='light',
fontfamily='serif',
                   color='#4a4a4a')

for i in movie.index:
    ax.annotate(f"{movie[i]}",
                   xy=(i, movie[i] + 60),
                   va = 'center', ha='center',fontweight='light',
fontfamily='serif',
                   color='#4a4a4a')



for s in ['top', 'left', 'right', 'bottom']:
    ax.spines[s].set_visible(False)

ax.set_xticklabels(mf.columns, fontfamily='serif')
ax.set_yticks([])

ax.legend().set_visible(False)
fig.text(0.16, 1, 'Rating distribution by Film & TV Show',
fontsize=15, fontweight='bold', fontfamily='serif')
fig.text(0.16, 0.89,
'''We observe that some ratings are only applicable to Movies.
The most common for both Movies & TV Shows are TV-MA and TV-14.
'''

, fontsize=12, fontweight='light', fontfamily='serif')


fig.text(0.755,0.924,"Movie", fontweight="bold", fontfamily='serif',
fontsize=15, color='#b20710')
fig.text(0.815,0.924,"|", fontweight="bold", fontfamily='serif',
fontsize=15, color='black')
fig.text(0.825,0.924,"TV Show", fontweight="bold", fontfamily='serif',
fontsize=15, color='#221f1f')

plt.show()
```

```
C:\Users\Usuario\AppData\Local\Temp\ipykernel_96516\4814534.py:33:
UserWarning: set_ticklabels() should only be used with a fixed number
of ticks, i.e. after set_ticks() or using a FixedLocator.
  ax.set_xticklabels(mf.columns, fontfamily='serif')
```



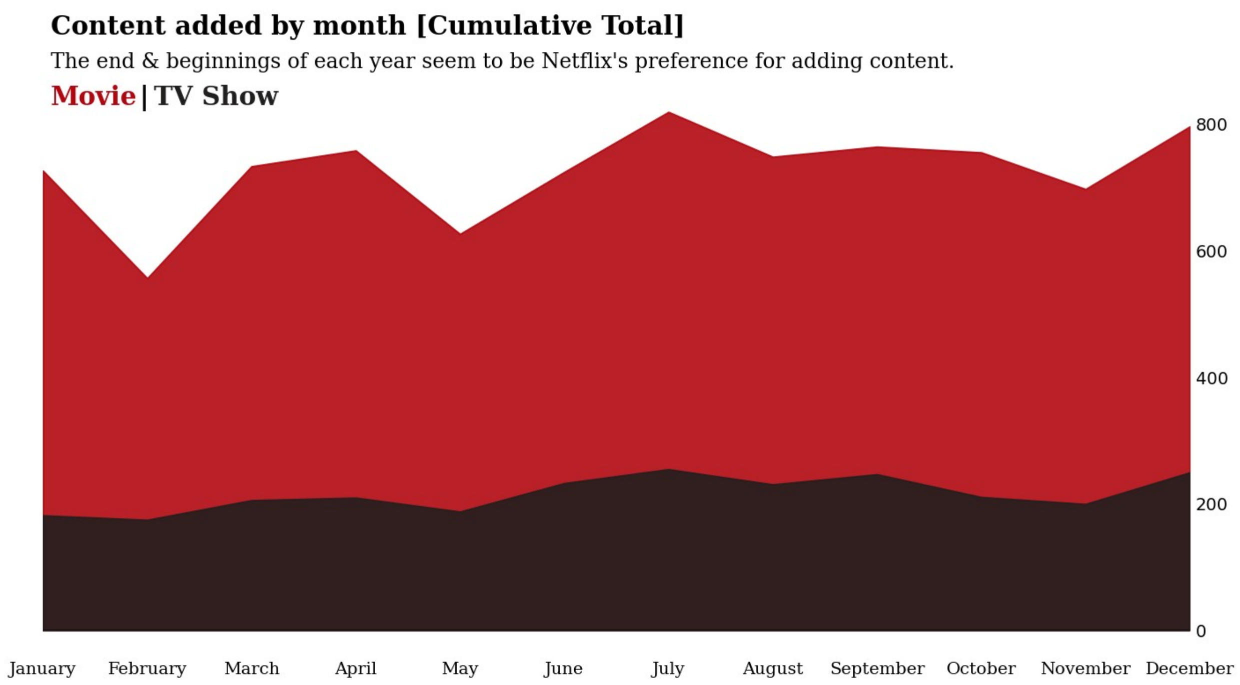
```
plt.rcParams['figure.dpi'] = 140

df["date_added"] = pd.to_datetime(df['date_added'], errors = 'coerce')

df['month_added']=df['date_added'].dt.month
df['month_name_added']=df['date_added'].dt.month_name()
df['year_added'] = df['date_added'].dt.year

df.head(3)


fig, ax = plt.subplots(1, 1, figsize=(12, 6))
color = ["#b20710", "#221f1f"]

for i, mtv in enumerate(df['type'].value_counts().index):
    mtv_rel = df[df['type']==mtv]
['year_added'].value_counts().sort_index()
    ax.plot(mtv_rel.index, mtv_rel, color=color[i], label=mtv)
    ax.fill_between(mtv_rel.index, 0, mtv_rel, color=color[i],
alpha=0.9)

ax.yaxis.tick_right()

ax.axhline(y = 0, color = 'black', linewidth = 1.3, alpha = .7)
```

```
#ax.set_ylim(0, 50)
#ax.legend(loc='upper left')
for s in ['top', 'right','bottom','left']:
    ax.spines[s].set_visible(False)

ax.grid(False)

ax.set_xlim(2008,2020)
plt.xticks(np.arange(2008, 2021, 1))

fig.text(0.13, 0.85, 'Movies & TV Shows added over time', fontsize=15,
fontweight='bold', fontfamily='serif')
fig.text(0.13, 0.59,
'''We see a slow start for Netflix over several years.
Things begin to pick up in 2015 and then there is a
rapid increase from 2016.

It looks like content additions have slowed down in 2020,
likely due to the COVID-19 pandemic.
'''

, fontsize=12, fontweight='light', fontfamily='serif')


fig.text(0.13,0.2,"Movie", fontweight="bold", fontfamily='serif',
fontsize=15, color='#b20710')
fig.text(0.19,0.2,"|", fontweight="bold", fontfamily='serif',
fontsize=15, color='black')
fig.text(0.2,0.2,"TV Show", fontweight="bold", fontfamily='serif',
fontsize=15, color='#221f1f')

ax.tick_params(axis=u'both', which=u'both',length=0)

plt.show()
```



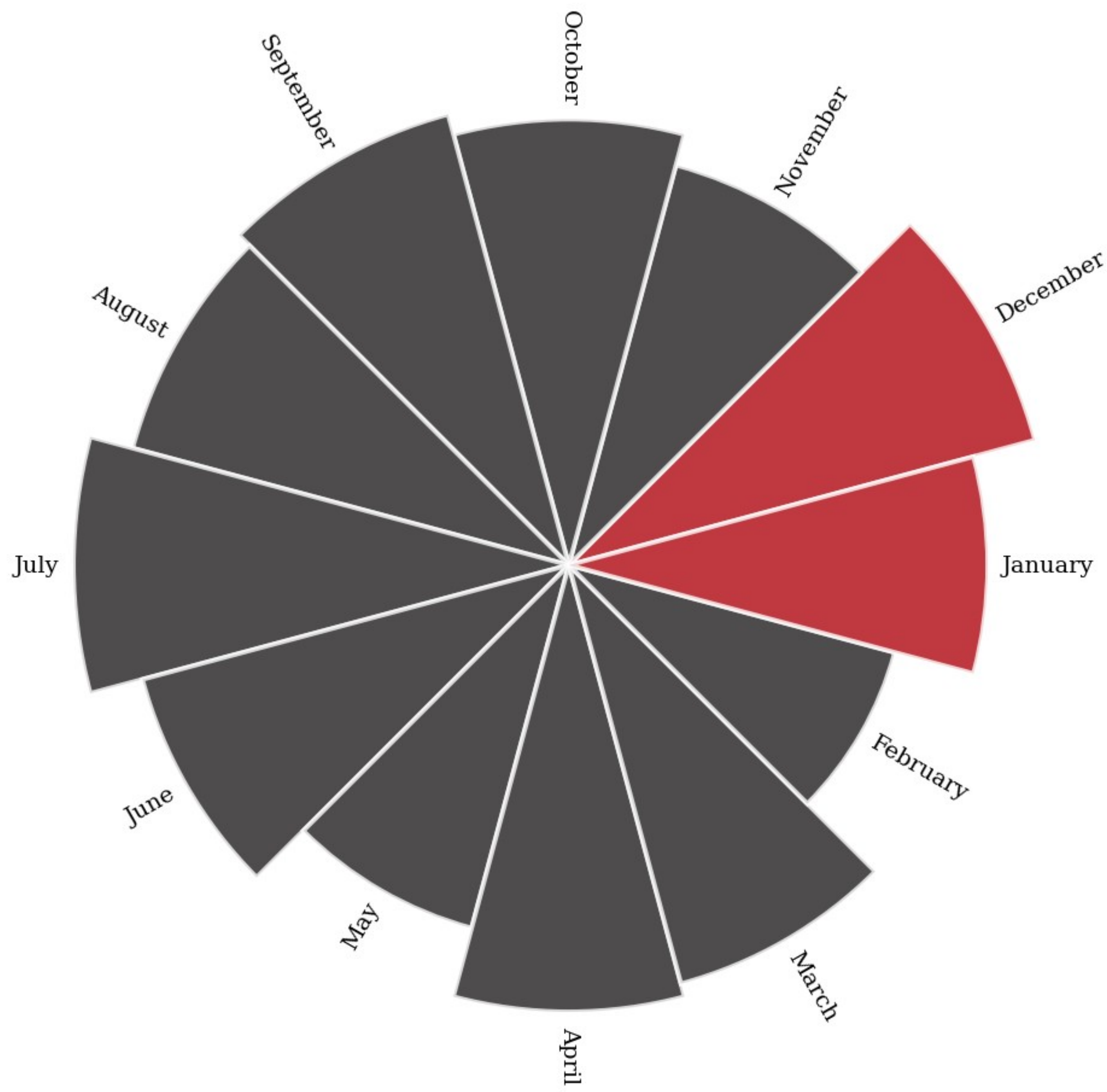
```
month_order = ['January',
 'February',
 'March',
 'April',
 'May',
 'June',
 'July',
 'August',
 'September',
 'October',
 'November',
 'December']

df['month_name_added'] = pd.Categorical(df['month_name_added'],
categories=month_order, ordered=True)

data_sub = df.groupby('type')
['month_name_added'].value_counts().unstack().fillna(0).loc[['TV
Show','Movie']].cumsum(axis=0).T

fig, ax = plt.subplots(1, 1, figsize=(12, 6))
color = ["#b20710", "#221f1f"]

for i, mtv in enumerate(df['type'].value_counts().index):
    mtv_rel = data_sub[mtv]
    ax.fill_between(mtv_rel.index, 0, mtv_rel, color=color[i],
label=mtv,alpha=0.9)



ax.yaxis.tick_right()
ax.axhline(y = 0, color = 'black', linewidth = 1.3, alpha = .4)

#ax.set_ylim(0, 50)
#ax.legend(loc='upper left')
for s in ['top', 'right','bottom','left']:
    ax.spines[s].set_visible(False)

ax.grid(False)
ax.set_xticklabels(data_sub.index, fontfamily='serif', rotation=0)
ax.margins(x=0) # remove white spaces next to margins

#ax.set_xlim(2008,2020)
#plt.xticks(np.arange(2008, 2021, 1))

fig.text(0.13, 0.95, 'Content added by month [Cumulative Total]',
fontsize=15, fontweight='bold', fontfamily='serif')
fig.text(0.13, 0.905,
"The end & beginnings of each year seem to be Netflix's preference for
adding content."

, fontsize=12, fontweight='light', fontfamily='serif')



fig.text(0.13,0.855,"Movie", fontweight="bold", fontfamily='serif',
fontsize=15, color='#b20710')
fig.text(0.19,0.855,"|", fontweight="bold", fontfamily='serif',
fontsize=15, color='black')
fig.text(0.2,0.855,"TV Show", fontweight="bold", fontfamily='serif',
fontsize=15, color='#221f1f')


ax.tick_params(axis=u'both', which=u'both',length=0)

plt.show()
```

```
C:\Users\Usuario\AppData\Local\Temp\ipykernel_96516\2403010786.py:37:
UserWarning: set_ticklabels() should only be used with a fixed number
of ticks, i.e. after set_ticks() or using a FixedLocator.
  ax.set_xticklabels(data_sub.index, fontfamily='serif', rotation=0)
```



```
data_sub2 = data_sub

data_sub2['Value'] = data_sub2['Movie'] + data_sub2['TV Show']
data_sub2 = data_sub2.reset_index()

df_polar = 
data_sub2.sort_values(by='month_name_added',ascending=False)


color_map = ['#221f1f' for _ in range(12)]
color_map[0] = color_map[11] =  '#b20710' # color highlight


# initialize the figure
plt.figure(figsize=(8,8))
ax = plt.subplot(111, polar=True)
plt.axis('off')

# Constants = parameters controling the plot layout:
upperLimit = 30
lowerLimit = 1
labelPadding = 30

# Compute max and min in the dataset
max = df_polar['Value'].max()

# Let's compute heights: they are a conversion of each item value in 
those new coordinates
# In our example, 0 in the dataset will be converted to the lowerLimit
```

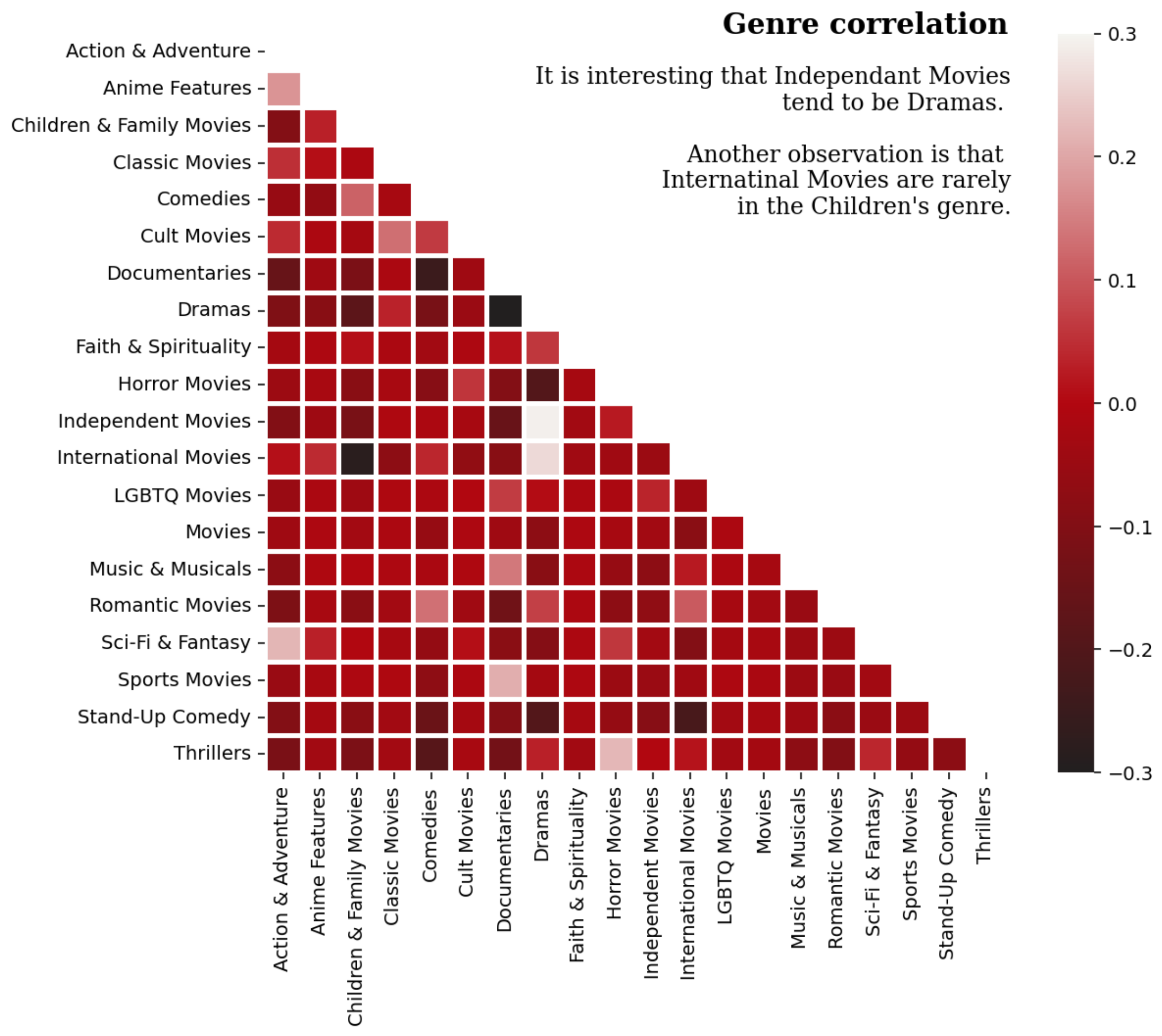
```
(10)
# The maximum will be converted to the upperLimit (100)
slope = (max - lowerLimit) / max
heights = slope * df_polar.Value + lowerLimit

# Compute the width of each bar. In total we have 2*Pi = 360°
width = 2*np.pi / len(df_polar.index)

# Compute the angle each bar is centered on:
indexes = list(range(1, len(df_polar.index)+1))
angles = [element * width for element in indexes]
angles

# Draw bars
bars = ax.bar(
    x=angles,
    height=heights,
    width=width,
    bottom=lowerLimit,
    linewidth=2,
    edgecolor="white",
    color=color_map,alpha=0.8
)

# Add labels
for bar, angle, height, label in zip(bars,angles, heights,
df_polar["month_name_added"]):

    # Labels are rotated. Rotation must be specified in degrees :(
    rotation = np.rad2deg(angle)

    # Flip some labels upside down
    alignment = ""
    if angle >= np.pi/2 and angle < 3*np.pi/2:
        alignment = "right"
        rotation = rotation + 180
    else:
        alignment = "left"


    # Finally add the labels
    ax.text(
        x=angle,
        y=lowerLimit + bar.get_height() + labelPadding,
        s=label,
        ha=alignment, fontsize=10,fontfamily='serif',
        va='center',
        rotation=rotation,
        rotation_mode="anchor")
```



```
import seaborn as sns

from sklearn.preprocessing import MultiLabelBinarizer

import matplotlib.colors


# Custom colour map based on Netflix palette
cmap = matplotlib.colors.LinearSegmentedColormap.from_list("",
['#221f1f', '#b20710','#f5f5f1'])
```

```
def genre_heatmap(df, title):
    df['genre'] = df['listed_in'].apply(lambda x :
x.replace(' ,',',').replace(', ',',').split(','))
    Types = []
    for i in df['genre']: Types += i
    Types = set(Types)
    print("There are {} types in the Netflix {}
Dataset".format(len(Types),title))
    test = df['genre']
    mlb = MultiLabelBinarizer()
    res = pd.DataFrame(mlb.fit_transform(test), columns=mlb.classes_,
index=test.index)
    corr = res.corr()
    mask = np.zeros_like(corr, dtype=np.bool_)
    mask[np.triu_indices_from(mask)] = True
    fig, ax = plt.subplots(figsize=(10, 7))
    fig.text(.54,.88,'Genre correlation',
fontfamily='serif',fontweight='bold',fontsize=15)
    fig.text(.75,.665,
            '''
             It is interesting that Independant Movies
             tend to be Dramas.

             Another observation is that
             Internatinal Movies are rarely
             in the Children's genre.
             ''', fontfamily='serif',fontsize=12,ha='right')
    pl = sns.heatmap(corr, mask=mask, cmap=cmap, vmax=.3, vmin=-.3,
center=0, square=True, linewidths=2.5)



df_tv = df[df["type"] == "TV Show"]
df_movies = df[df["type"] == "Movie"]

genre_heatmap(df_movies, 'Movie')
plt.show()
```

```
C:\Users\Usuario\AppData\Local\Temp\ipykernel_96516\2710882134.py:14:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  df['genre'] = df['listed_in'].apply(lambda x :
x.replace(' ,',',').replace(', ',',').split(','))

There are 20 types in the Netflix Movie Dataset
```


Genre correlation
It is interesting that Independant Movies tend to be Dramas.
Another observation is that Internatinal Movies are rarely in the Children's genre.
Action & Adventure
Anime Features
Children & Family Movies
Classic Movies
Comedies
Cult Movies
Documentaries
Dramas
Faith & Spirituality
Horror Movies
Independent Movies
International Movies
LGBTQ Movies
Movies
Music & Musicals
Romantic Movies
Sci-Fi & Fantasy
Sports Movies
Stand-Up Comedy
Thrillers
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3