jupyter movies-data-final (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3

Run Code

In [2]:
```
import numpy as np
import pandas as pd
import pandas_bokeh
import panel as pn
import pycountry
pn.extension()
import panel.widgets as pnw
```

In [3]:
```
import geopandas as gpd
import json
import pylab as plt
```

In [4]:
```
import hvplot.pandas
import holoviews as hv
```

In [5]:
```
hv.extension('bokeh')
```

In [6]:
```
from holoviews import opts
```

In [7]:
```
from bokeh.resources import INLINE
import bokeh.io
bokeh.io.output_notebook(INLINE)
```

BokehJS 2.3.2 successfully loaded.

In [8]:
```
from bokeh.io import output_file, show, output_notebook, export_png
from bokeh.models import ColumnDataSource, GeoJSONDataSource, LinearColorMapper, ColorBar
from bokeh.plotting import figure
from bokeh.palettes import brewer
```

In [9]:
```
# import internet
internet= pd.read_csv('World Development Indicators - internet adoption.csv')

#import content by country
content_by_country= pd.read_excel('Movies produced by country-NFLX.xlsx')

#import genre and year
year= pd.read_excel('Movies produced by country and year-NFLX.xlsx')
genre= pd.read_excel('Movies produced by country and Genre-NFLX-Updated.xlsx')

# all content
```

```
all_content= pd.read_excel('netflix_titles modified.xlsx')
all_content = all_content.rename(columns = {'duration (seasons or mins)':'duration'})
```

In [10]:

```
# dfs
movies_only= all_content[all_content['type']== 'Movie']
shows_only= all_content[all_content['type']== 'TV Show']
movies_with_rev= movies_only.copy()
movies_with_rev[movies_with_rev['Revenue_MM'].notnull()]

#Angela's df items

count_con=year.groupby(['country']).count()
count_con= count_con[count_con['NFLX_Key'] > 1]
country_list= count_con.index.to_list()
country_year = year[year['country'].isin(country_list)]
count_year = country_year.groupby(['country', 'Year']).count()
count_year=count_year.rename(columns={"NFLX_Key": "Count"})

heatmap_country_list=["United States","India", "United Kingdom",
                      "Canada", "France","Japan","Spain","South Korea",
                     "Germany","Australia","Mexico",
                     "China","Hong Kong","Turkey","Taiwan"]
heatmap_country_list=heatmap_country_list
heatmap_genre = genre[genre['country'].isin(heatmap_country_list)]
heatmap_genre= heatmap_genre.drop(['Genre','type','NFLX_Key'], axis=1)
heatmap_genre=heatmap_genre.dropna()
heatmap_genre=heatmap_genre.rename(columns={"Value": "Genre"})
heatmap_genre.loc[:,'Genre']=heatmap_genre.loc[:,'Genre'].str.lstrip()
heatmap_genre= heatmap_genre.groupby(['country', 'Genre']).count()
heatmap_genre=heatmap_genre.rename(columns={"release_year": "Count"})
#heatmap_genre=heatmap_genre.sort_values('country', ascending=False)
#heatmap_genre

#pivots
rating_pivot = pd.pivot_table(all_content, index= 'rating',  values='NFLX_Key', aggfunc= len)

country_content_pivot = pd.pivot_table(content_by_country, index= 'country',  values='NFLX_Key', aggfunc= len)
sorted_country_content_pivot = country_content_pivot.sort_values('NFLX_Key', ascending=False)
top_15_countries = sorted_country_content_pivot.head(15)
```

In [11]:

```
#Data cleaning for genre
movies_only['genre'] = movies_only.loc[:,'listed_in'].str.split(',')
#genre = movies_only['genre'].apply(pd.Series).stack().unique()
#max(movies_only['genre'].str.len())
#The maximum number of genres associated with a movie is 3, so we split the genre list into 3 columns
```

```
<ipython-input-11-b3b611444fc4>:2: 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
  movies_only['genre'] = movies_only.loc[:,'listed_in'].str.split(',')
```

In [12]:

```
new_columns = pd.DataFrame(movies_only.genre.to_list(), columns = ['genre1', 'genre2', 'genre3'])
movies_only = pd.concat([movies_only, new_columns], axis = 1)
```

In [13]:
```
seasonality = movies_only[['date_added', 'title', 'genre1', 'genre2', 'genre3']]
g1 = seasonality.groupby(['genre1', 'date_added'], as_index = False).count()
g2 = seasonality.groupby(['genre2', 'date_added'], as_index = False).count()
g3 = seasonality.groupby(['genre3', 'date_added'], as_index = False).count()
```

In [14]:
```
g2.loc[:,'genre2'] = g2.loc[:,'genre2'].str.lstrip()
g3.loc[:,'genre3'] = g3.loc[:,'genre3'].str.lstrip()
```

In [15]:
```
for item in (g1, g2, g3):
    item['month_added'] = pd.to_datetime(item['date_added']).dt.to_period('m')
```

In [16]:
```
g1 = g1.rename(columns = {'title': 'count', 'genre1': 'genre'})
g2 = g2.rename(columns = {'title': 'count', 'genre2': 'genre'})
g3 = g3.rename(columns = {'title': 'count', 'genre3': 'genre'})
```

In [17]:
```
genre_total = pd.concat([g1, g2, g3])
genre_final = genre_total.groupby(['genre', 'month_added'], as_index = False)['count'].sum()
genre_final['month'] = genre_final['month_added'].dt.month
genre_final['year'] = genre_final['month_added'].dt.year
```

In [18]:
```
genre_year1 = genre_final.sort_values(['year','month','genre'])
```

In [19]:
```
d = {1:'JAN', 2:'FEB', 3:'MAR', 4:'APR', 5:'MAY', 6:'JUN',
     7:'JUL', 8:'AUG', 9:'SEP', 10:'OCT', 11:'NOV', 12:'DEC'}
genre_year1['month'] = genre_year1.loc[:,'month'].map(d)
genre_year2 = genre_year1.rename(columns = {'genre': 'Genre', 'month': 'Month', 'year': 'Year', 'count': 'Count'})
```

In [20]:
```
#keep year 2015 ~ 2019 only
years = [2015,2016,2017,2018,2019]
genre_year3 = genre_year2[genre_year2.Year.isin(years)]
```

In [21]:
```
#drop genre with 0 movie count
genretodrop = ['LGBTQ Movies', 'Faith & Spirituality', 'Anime Features']
genre_year3 = genre_year3[~genre_year3.Genre.isin(genretodrop)]
```

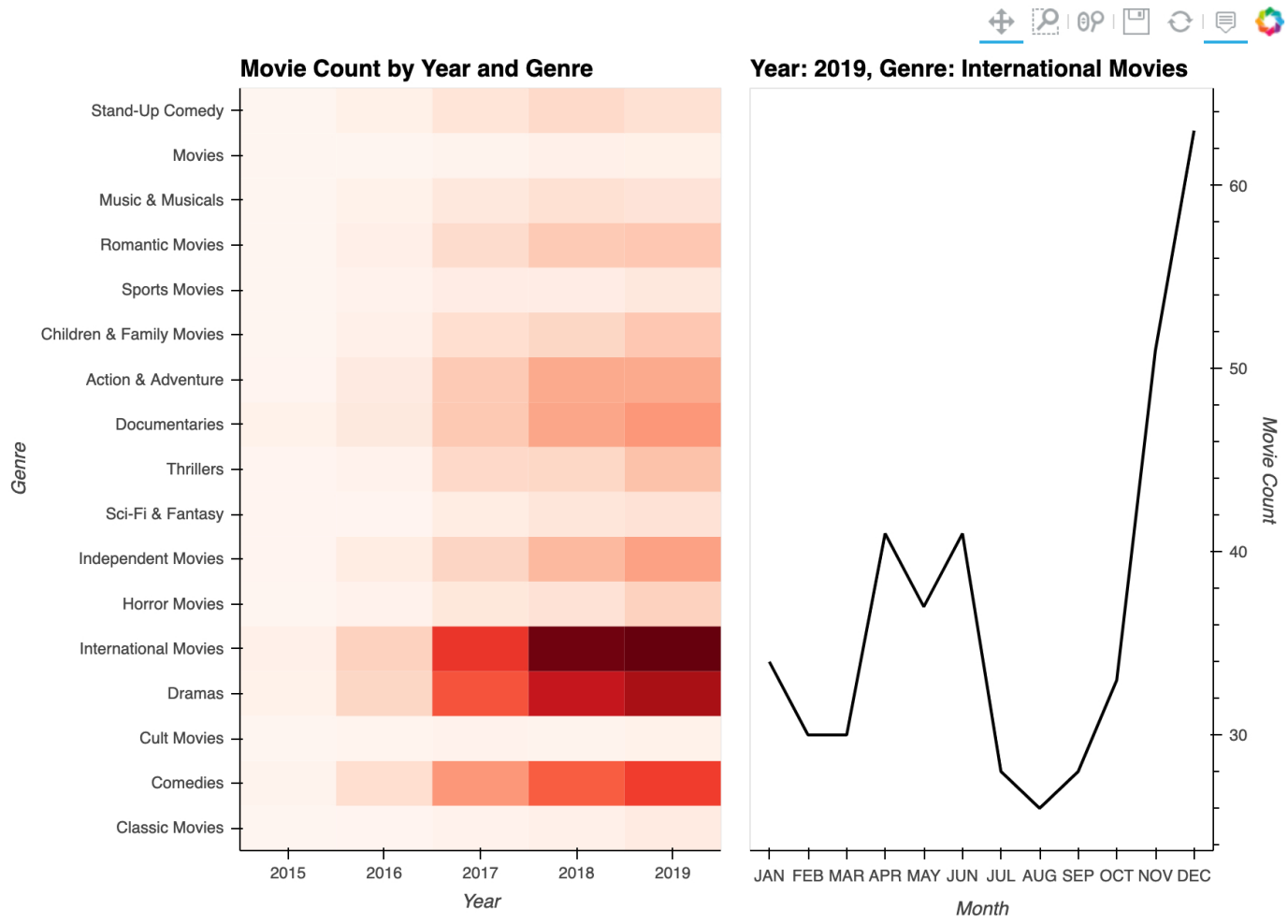
In [22]:
```
genre_year3['Year'] = genre_year3['Year'].astype(str)
```

In [23]:
```
#Tap Stream: http://holoviews.org/reference/streams/bokeh/Tap.html#streams-bokeh-gallery-tap
def genre_year_hmap():
    dataset = hv.Dataset(genre_year3, vdims=('Count','Movie Count'))
    genre_year_heatmap = hv.HeatMap(dataset.aggregate(['Year', 'Genre'], np.sum),
                     label='Movie Count by Year and Genre').opts(cmap = 'reds',
                                                                 height = 600,
                                                                 width = 500,
                                                                 tools = ['hover'],
                                                                 shared_axes = False)
    posxy = hv.streams.Tap(source=genre_year_heatmap, x='2019', y='International Movies')
    def tap_histogram(x, y):
        return hv.Curve(dataset.select(Genre=y, Year=x), kdims='Month',
                        label=f'Year: {x}, Genre: {y}').opts(height=600,
                                                             line_color='black',
```

```
                                                    width=375,
                                                    yaxis='right',
                                                    tools = ['hover'])
    tap_dmap = hv.DynamicMap(tap_histogram, streams=[posxy])
    return (genre_year_heatmap + tap_dmap)
```

In [24]:
```
genre_year_hmap()
```

Out[24]:



In [25]:
```
count_year.reset_index(inplace=True)
count_year1= count_year[count_year['country'].isin(heatmap_country_list)]
count_year1
```

Out[25]:

	country	Year	Count	title
11	Australia	1981	1	1
12	Australia	1986	1	1
13	Australia	1992	2	2

14	Australia	1998	2	2
15	Australia	2003	2	2
...	...	...	...	...
795	United States	2016	306	306
796	United States	2017	406	406
797	United States	2018	436	436
798	United States	2019	374	374
799	United States	2020	11	11

423 rows × 4 columns

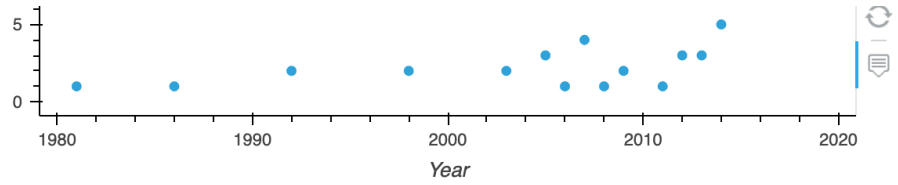
In [26]:
```
def bars_map():
    dataset = hv.Dataset(count_year, vdims=('Count','Movie Count'))
    top_movie_bars = hv.Bars(dataset.aggregate(['country', 'Count'], np.sum),
                     label='Movie Count by Year and Genre').opts(height = 600,
                                                                 width = 500,
                                                                 tools = ['hover'])

    posxy = hv.streams.Tap(source=top_movie_bars, x='India', y=836)
    def tap_histogram(x, y):
        return hv.Scatter(dataset.select(count=int(y), country=x), kdims='Year',
                    label=f'country:{x}').opts(height=600,width=375,
                                               yaxis='right',
                                               tools = ['hover']
                                               )
    tap_dmap = hv.DynamicMap(tap_histogram, streams=[posxy])
    return ( bars_map + tap_dmap)
```

In [27]:
```
uGroup=pn.widgets.Select(name='Country',
                         options=heatmap_country_list,
                         value='country', width=175)
```

In [28]:
```
def seasonality_by_year():
    seasonality= count_year1.hvplot(x = 'Year', y = 'Count', groupby = 'country',
                kind ='scatter', width = 600, widget_location='right_top',
                title = 'Seasonality')
    return seasonality
```

In [29]:
```
seasonality_by_year()
```

Out[29]:



In [30]:
```
def heatmap_genre_country():
    #heatmap_labels= dict(x="Genre", y="country", color="Count")

    heatmap=heatmap_genre.hvplot.heatmap(title='Top 15 Countries by Genre',
                                         x='Genre', y='country', C='Count',
                  height=500, width=1000, colorbar=False,rot=70)
    return heatmap
```

In [31]:
```
#all options at hvplot.help('hist')
def movie_length_distribution():
    mov_distr= movies_only.hvplot.hist(y= 'duration',
                            bins= 30,
                            height= 400,
                            title= 'Distribution by Movie Duration',
                            ylabel= 'Count',
                            xlabel= 'duration (minutes)',
                            color= 'red',
                            alpha= 0.4,
                            hover_color= 'blue',
                            xlim= (1, 200),
                           )
    return mov_distr
```

In [32]:
```
shows_only = shows_only.sort_values(by = 'duration')
tv_show_duration = shows_only.groupby(['duration']).count()
tv_show_duration = tv_show_duration.reset_index()
tv_show_duration = tv_show_duration[['duration', 'NFLX_Key']]
tv_show_duration = tv_show_duration.rename(columns = {'NFLX_Key': 'Count'})
tv_show_duration = tv_show_duration[tv_show_duration['duration'] <= 10]
```

In [33]:
```
def tv_show_season_dist():
    tv_dist = tv_show_duration.hvplot.bar(y = 'Count',
                                    x = 'duration',
                                    height= 400,
                                    title= 'Distribution by TV Show Seasons',
                                    ylabel= 'Count',
                                    xlabel= 'duration (seasons)',
                                    color= 'black',
                                    alpha= 0.4,
                                    hover_color= 'green',
                                    shared_axes = False)
    return tv_dist
```

In [34]:
```
def show_rating_distribution():
    rating_distr = rating_pivot.hvplot.bar(y= 'NFLX_Key',
                            height= 400,
                            title= 'Distribution by Rating',
```

```
                            ylabel= 'Count',
                            xlabel= 'Rating',
                            color= 'red',
                            alpha= 0.6,
                            hover_color= 'cyan'
                           )
    return rating_distr
```

In [35]:

```
def movies_by_country():
    movies_distr = top_15_countries.hvplot.bar(y= 'NFLX_Key',
                            height= 400,
                            title= 'Top 15 Countries Where Movies are Filmed',
                            rot= 45,
                            ylabel= 'Count',
                            color= 'blue',
                            alpha= 0.6,
                            hover_color= 'gray'
                           )
    return movies_distr
```

In [36]:

```
#hvplot.help('scatter')
def rating_to_release():
    rating_release_rev = movies_with_rev.hvplot(x='Revenue_MM',
                             y= 'IMDB_score',
                            kind= 'scatter',
                            c= 'release_year',
                            cmap= 'coolwarm',
                           size= 100,
                           height= 600,
                           alpha= 0.8,
                           title= 'IMDB_Score and Release Year by Revenue (MM)',
                           xlim=(20,400),
                           ylim= (3, 10))

    return rating_release_rev
```

In [37]:

```
#filter out countries that are not in the netflix dataset
all_content_country = all_content['country'].apply(pd.Series).stack().unique()
internet_adopt = internet[internet['Country'].isin(all_content_country)]
```

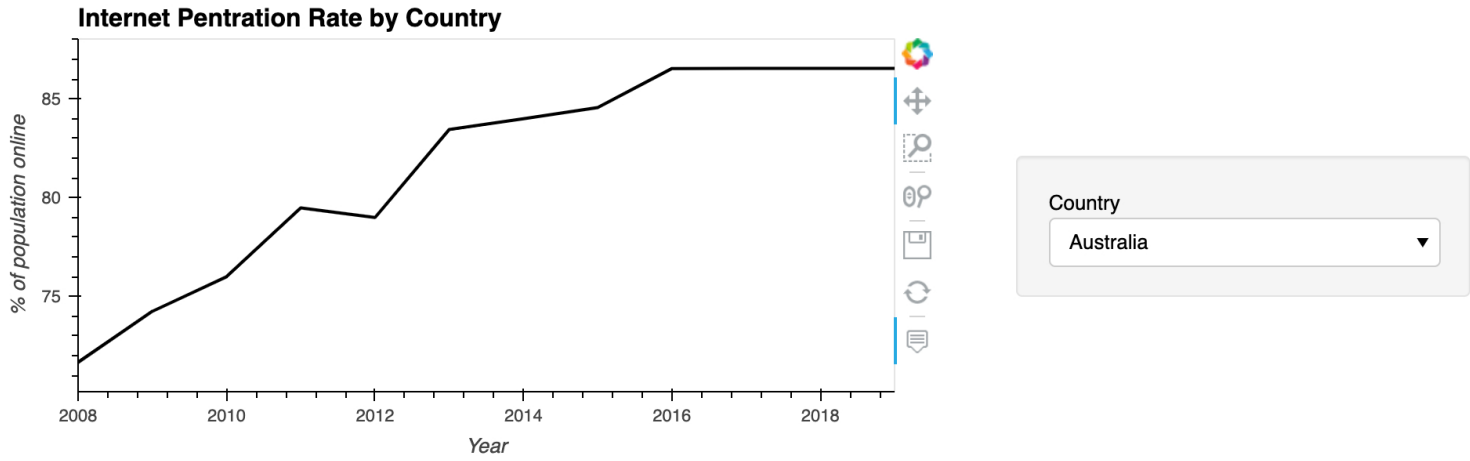
In [38]:

```
# hvplot.help('line')
internet_adopt.sort_values(by= ['Year'])


top_15_countries.columns.name = None
top_15_countries_internet = top_15_countries.reset_index()
filtered_countries =  list(top_15_countries_internet.country)

filtered_internet_top_countries = internet[ internet.Country.isin(filtered_countries)  == True]
filtered_internet_top_countries

def internet_adoption():
    int_adop = filtered_internet_top_countries.hvplot(x= 'Year',
                             y= '%_of_pop_using_internet',
                             groupby= 'Country',
```

```
                                kind= 'line',
                                width= 600,
                                ylabel= '% of population online',
                                title= 'Internet Pentration Rate by Country ',
                                color= 'black',
                                shared_axes = False
                               )
    return int_adop
```

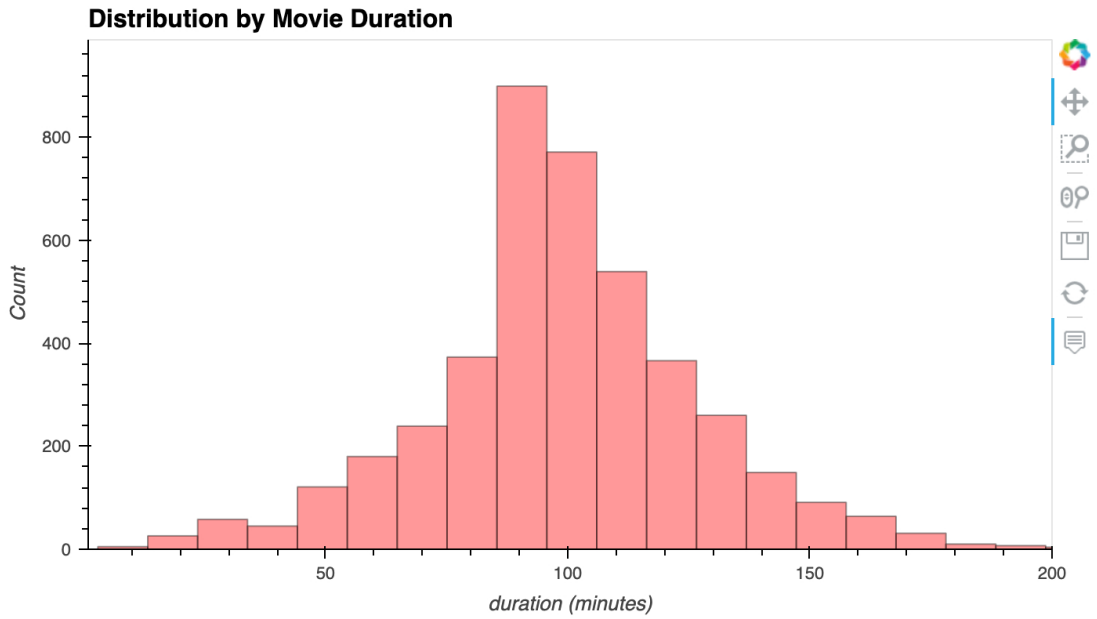
In [39]:
```
internet_adoption()
```

Out[39]:



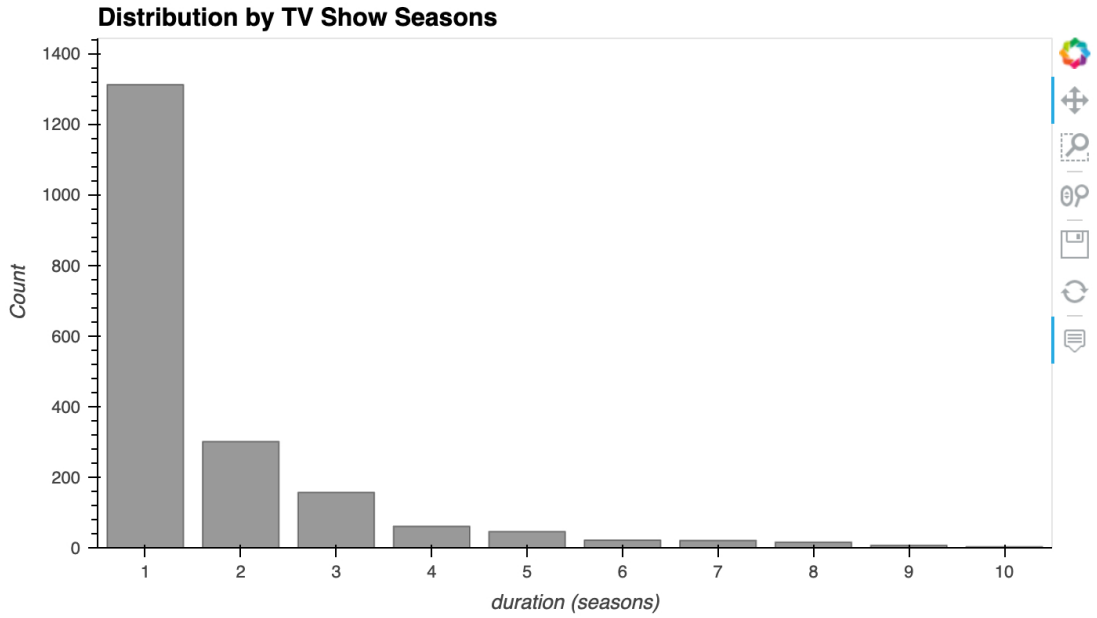
In [40]:
```
movie_length_distribution()
```

Out[40]:



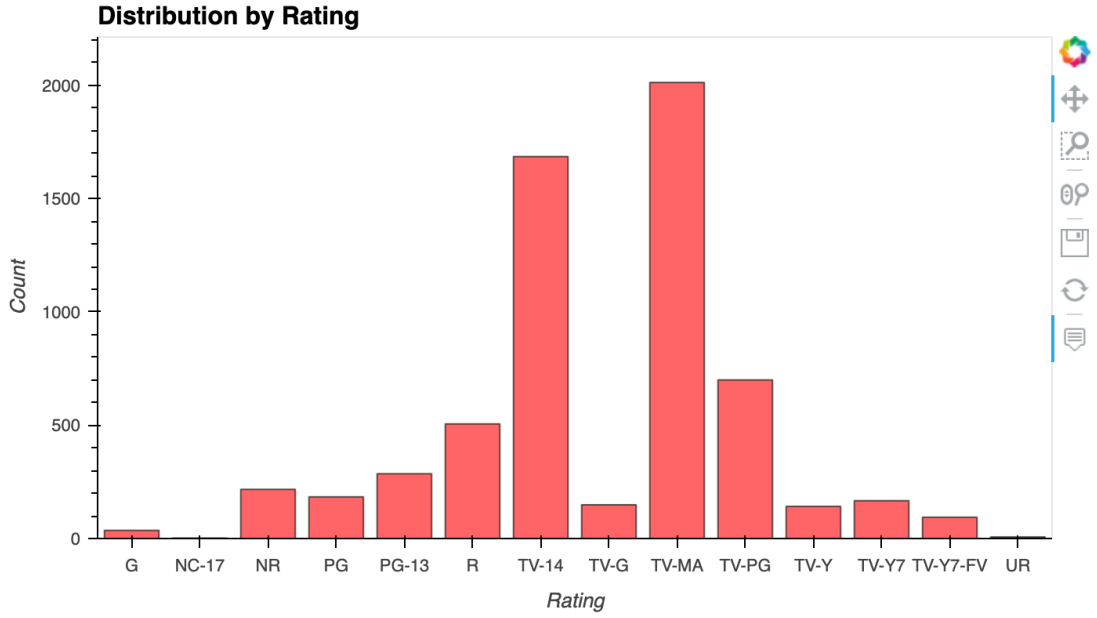
In [41]:
```
1 tv_show_season_dist()
```

Out[41]:



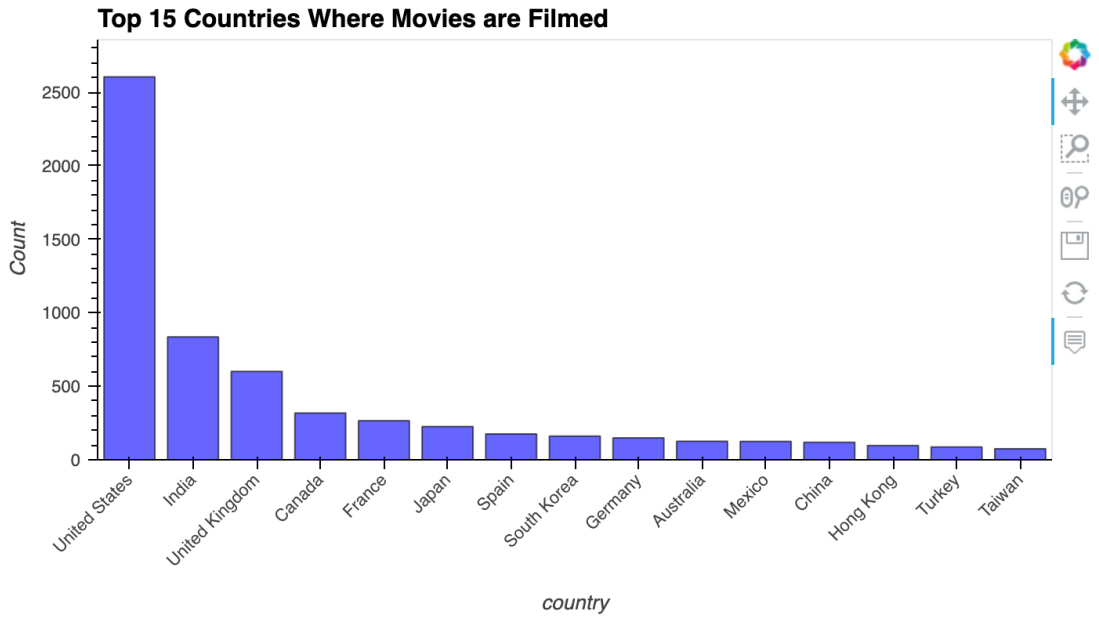
In [42]:
```
1 show_rating_distribution()
```

Out[42]:



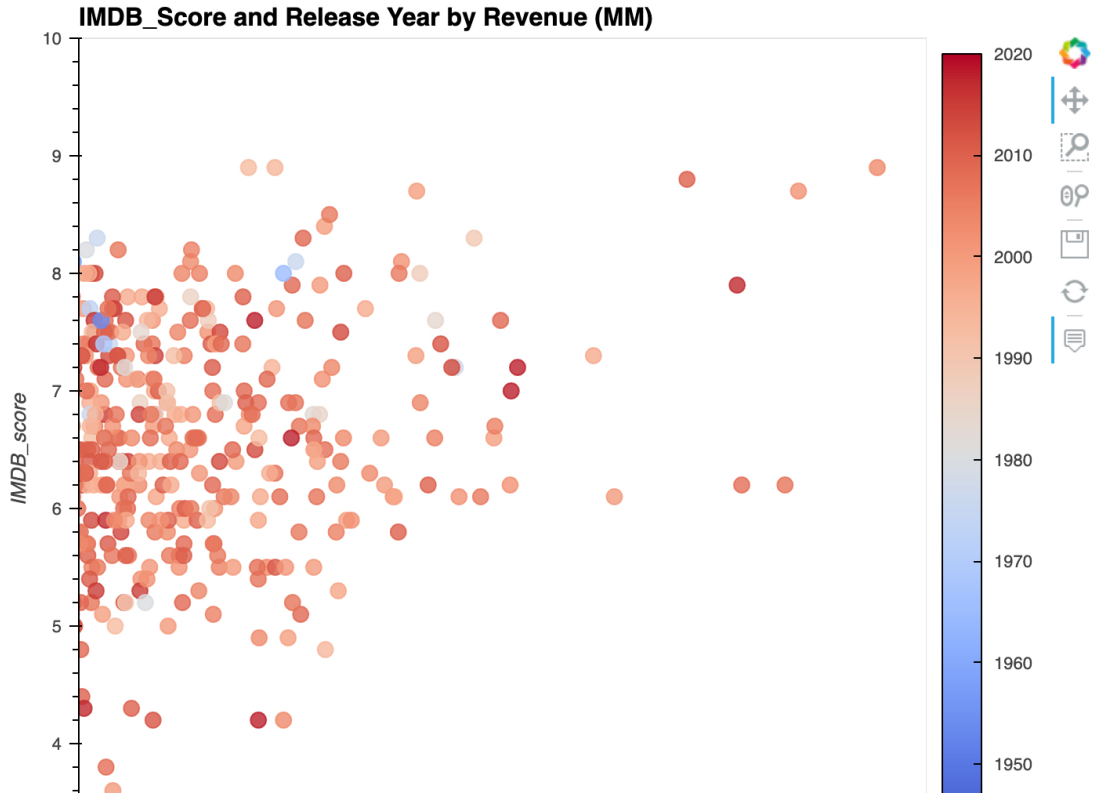
In [43]:
```
1 movies_by_country()
```

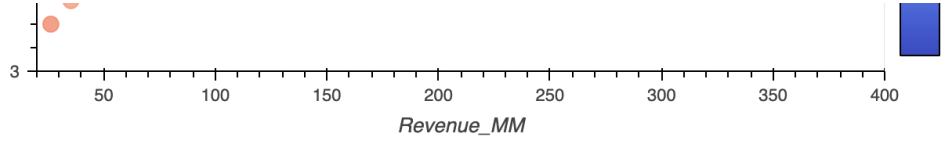
Out[43]:



In [44]:
```
1 rating_to_release()
```

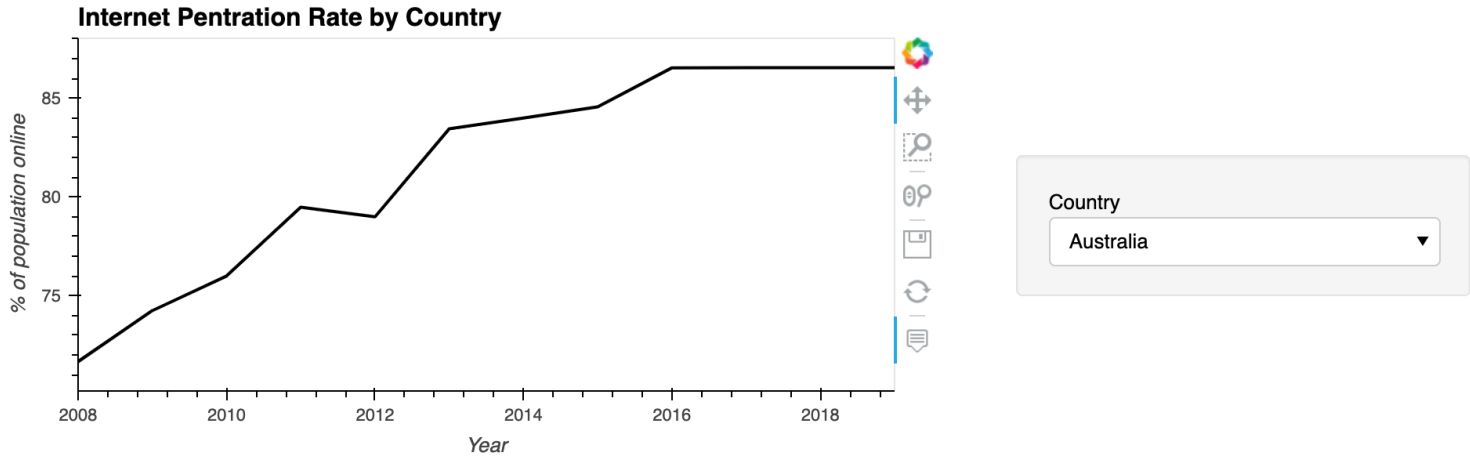
Out[44]:





In [45]:
```
internet_adoption()
```

Out[45]:

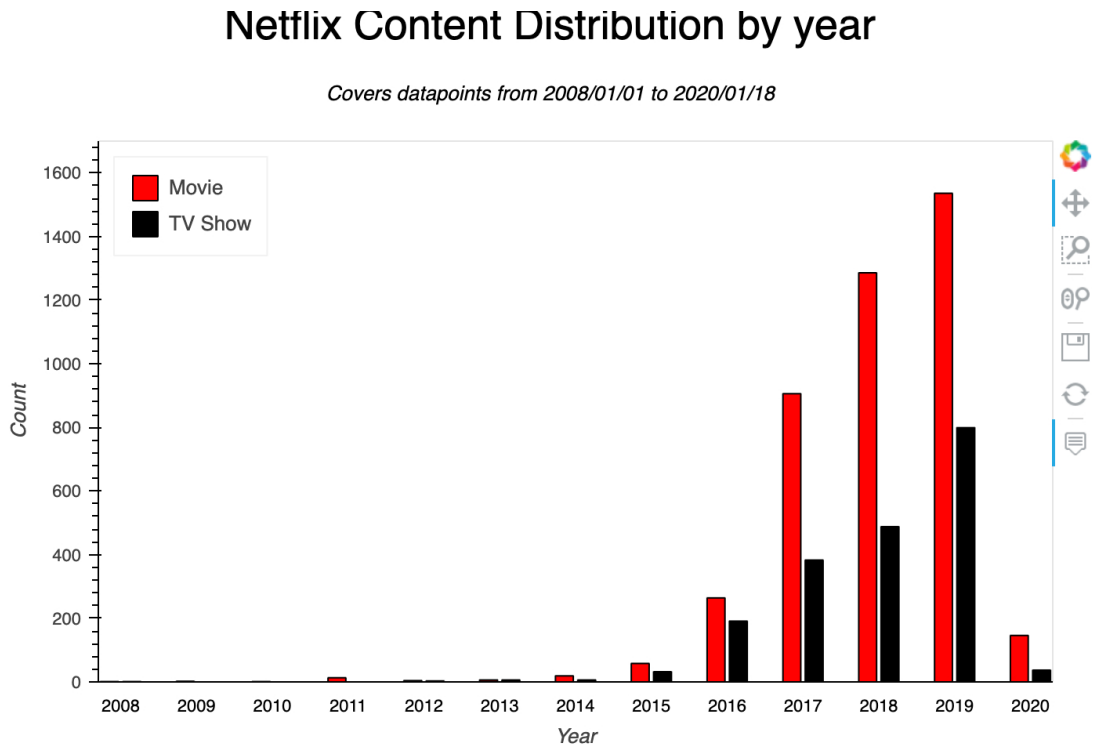


In [46]:
```
all_content['year_added'] = pd.to_datetime(all_content.loc[:,'date_added']).dt.to_period('y')
all_content = all_content.loc[all_content['year_added'].notnull()]
content = all_content.groupby(['type', 'year_added'], as_index = False).count()
new_content = content.iloc[:, 0: 3]
new_content = new_content.rename(columns = {'NFLX_Key': 'Count'})
```

In [47]:
```
def content_type ():
    content_dist = new_content.hvplot.bar(y = 'Count',
                                          x = 'year_added',
                                         height = 400,
                                         xlabel = 'Year',
                                         ylabel = 'Count',
                                         by = 'type',
                                         legend='top_left',
                                         color = ['red','black'],
                                         ylim = (0, 1700),
                                         hover_color= 'gray',
                                         shared_axes = False
                                         )
    content_dist = content_dist.opts(multi_level=False)
    content_layout = pn.Column(pn.pane.Markdown("##Netflix Content Distribution by year", align="center"),
                               pn.pane.Markdown("*Covers datapoints from 2008/01/01 to 2020/01/18*", align="center"
                               content_dist)
    return content_layout
```

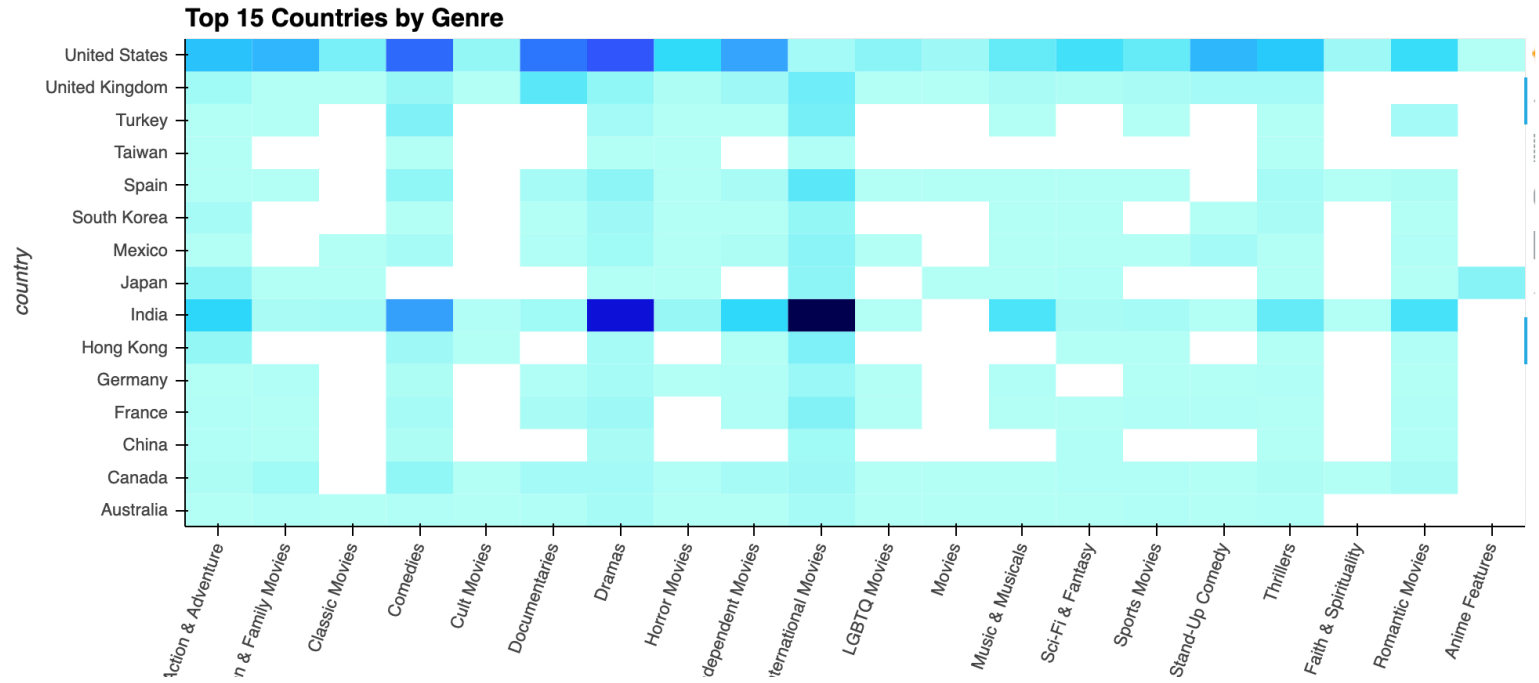
In [48]:
```
content_type()
```

Out[48]:



In [49]:
```
1 heatmap_genre_country()
```

Out[49]:



In [50]:
```
world = gpd.read_file(gpd.datasets.get_path('naturalearth_lowres'))
```

In [51]:
```
world = world.rename(columns = {'iso_a3':'Country_Code'})
```

In [52]:
```
world = world[['Country_Code', 'geometry']]
```

In [53]:
```
#add country code to count_year
def countrycode(col):
    Code = []
    for country in col:
        try:
            code = pycountry.countries.get(name = country).alpha_3
            Code.append(code)
        except:
            Code.append('None')
    return Code
```

In [54]:
```
count_year['Country_Code'] = countrycode(count_year.country)
count_year = count_year[count_year.Country_Code != 'None']
count_year = count_year.drop(columns = 'title')
```

In [55]:
```
count_year2 = count_year[count_year.Year >= 2008]
count_year2 = count_year2[count_year2.Year < 2020]
count_year2 = count_year2.sort_values(by = 'Year')
count_year2 = count_year2.rename(columns = {'Count': '# of Movies Produced'})
```

In [56]:
```
count_year2 = count_year2.drop(columns = 'country')
```

In [57]:
```
merged = count_year2.merge(internet, on = ['Country_Code','Year'], how = 'inner')
```

In [58]:
```
def get_dataset(year=None):
    df = merged
    if year is not None:
        df = df[df['Year'] == year]
    merged1 = world.merge(df, on = 'Country_Code', how = 'right')
    index_names = merged1[merged1['geometry'] == None ].index
    merged1.drop(index_names, inplace = True)
    return merged1

def get_geodatasource(world1):
    json_data = json.dumps(json.loads(world1.to_json()))
    return GeoJSONDataSource(geojson = json_data)
```

In [59]:
```
def bokeh_plot_map(world1):
    geosource = get_geodatasource(world1)
    palette = brewer['OrRd'][8]
    palette = palette[::-1]
```

```
    #internet penetration rate
    color_mapper = LinearColorMapper(palette = palette, low = 10, high = 100)
    color_bar = ColorBar(color_mapper=color_mapper, label_standoff=8, width=500, height=20,
                         location=(0,0), orientation='horizontal')

    tools = 'wheel_zoom,pan,reset'

    m = figure(title = 'World Internet Penetration Rate',
               plot_height=400 , plot_width=850, toolbar_location='right', tools=tools)
    m.xgrid.grid_line_color = None
    m.ygrid.grid_line_color = None
    m.patches('xs','ys', source=geosource, fill_alpha=1, line_width=0.5, line_color='black',
              fill_color={'field' :'%_of_pop_using_internet' , 'transform': color_mapper})
    m.add_layout(color_bar, 'below')

    return m
```

In [60]:

```
def bokeh_plot_map2(world1):
    geosource = get_geodatasource(world1)
    palette = brewer['OrRd'][8]
    palette = palette[::-1]

    color_mapper2 = LinearColorMapper(palette = palette, low = 0, high = 150)
    color_bar2 = ColorBar(color_mapper=color_mapper2, label_standoff=8, width=500, height=20,
                         location=(0,0), orientation='horizontal')
    tools = 'wheel_zoom,pan,reset'

    m2 = figure(title = 'World Movie Production Count',
               plot_height=400 , plot_width=850, toolbar_location='right', tools=tools)
    m2.xgrid.grid_line_color = None
    m2.ygrid.grid_line_color = None
    m2.patches('xs','ys', source=geosource, fill_alpha=1, line_width=0.5, line_color='black',
              fill_color={'field' :'# of Movies Produced' , 'transform': color_mapper2})
    m2.add_layout(color_bar2, 'below')

    return m2
```

In [61]:

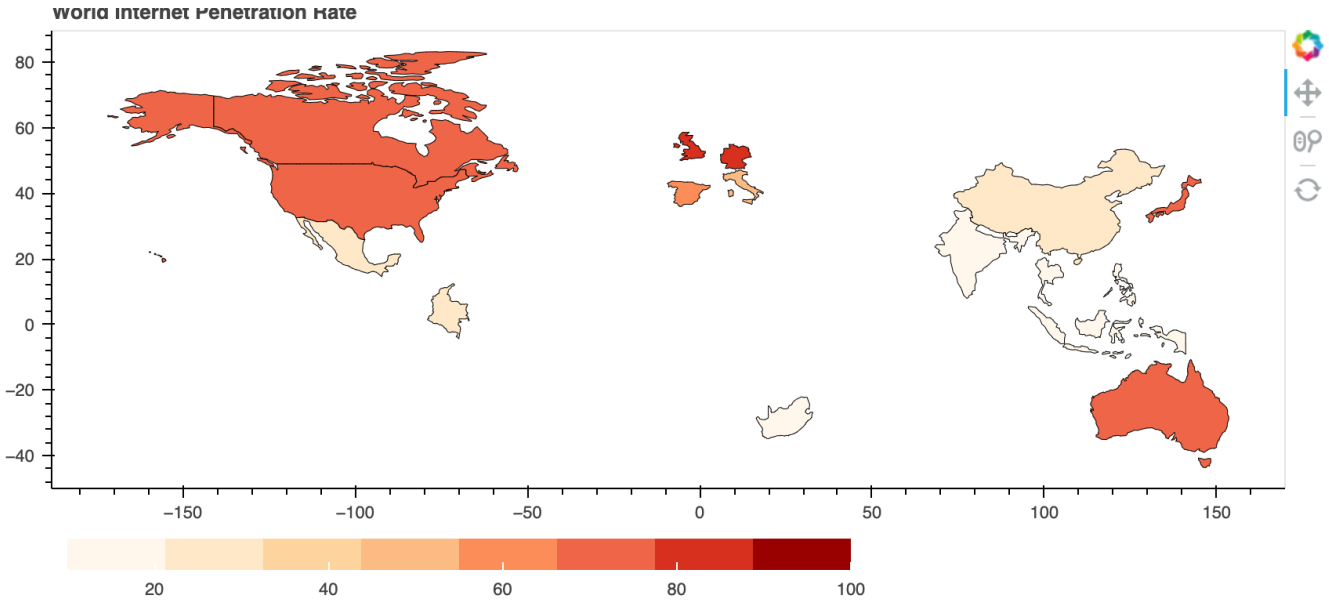
```
def map_dash1():
    from bokeh.models.widgets import DataTable
    map_pane = pn.pane.Bokeh(width=850)
    year_slider = pnw.IntSlider(start=2008,end=2019,value=2008)
    def update_map(event):
        world1 = get_dataset(year=year_slider.value)
        map_pane.object = bokeh_plot_map(world1)
        return
    year_slider.param.watch(update_map,'value')
    year_slider.param.trigger('value')
    app = pn.Column(year_slider,map_pane)
    return app
```

In [62]:

```
map_dash1()
```

Out[62]:

2008



In [63]:

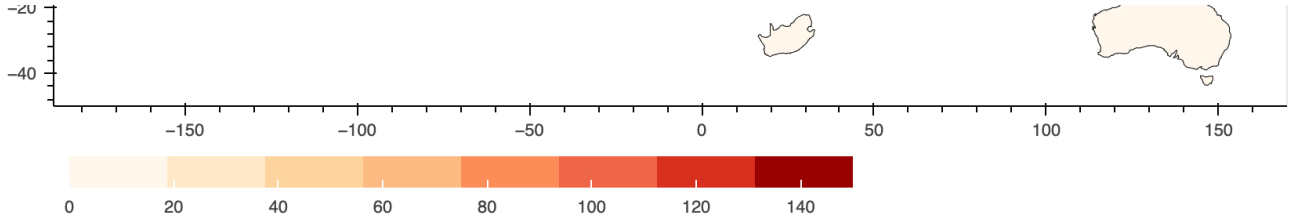
```
def map_dash2():
    from bokeh.models.widgets import DataTable
    map_pane2 = pn.pane.Bokeh(width=850)
    year_slider2 = pnw.IntSlider(start=2008,end=2019,value=2008)
    def update_map2(event):
        world1 = get_dataset(year=year_slider2.value)
        map_pane2.object = bokeh_plot_map2(world1)
        return
    year_slider2.param.watch(update_map2,'value')
    year_slider2.param.trigger('value')
    app2 = pn.Column(year_slider2,map_pane2)
    return app2
```

In [64]:

```
map_dash2()
```

Out[64]:

2008



In [65]:
```
png_pane = pn.pane.PNG('https://i.pinimg.com/originals/da/f3/0f/daf30fac5e16393d66a3684dd27e29af.png',
                       width=100, height=100)
text = pn.pane.Markdown("# Netflix Analysis", width = 300, margin = 30)
title = pn.Row(png_pane, text)
```

In [66]:
```
tab1 = pn.Column(content_type)
tab2 = pn.Column(movie_length_distribution,pn.Row(pn.Spacer(height = 15)), tv_show_season_dist)
tab3 = pn.Column(show_rating_distribution)
tab4=  pn.Column(movies_by_country,pn.Column(pn.Row(pn.Spacer(height=30),seasonality_by_year),
                          pn.Column(pn.Spacer(height = 15)),internet_adoption))
tab5 = pn.Column(map_dash1, pn.Row(pn.Spacer(height = 15)),map_dash2)
tab6 = pn.Column(rating_to_release)
tab7 = pn.Column(genre_year_hmap)
tab8 = pn.Column(heatmap_genre_country)
tabs = pn.Tabs(("Content Type Dist", tab1),
               ("Movie + TV Show Length Dist", tab2),
               ("Show Rating Dist", tab3),
               ("Movies by Country(Seasonality and Internet Adoption)", tab4),
               ("World Internet Penetration Rate and Movie", tab5),
               ("Rating to Release", tab6),
               ("Dist by Genre & Year", tab7),
               ("Countries and Genre Heatmap ", tab8)
              )
```

In [67]:
```
# Using .show() to start a Bokeh server instance from within your jupyter notebook for rapid prototyping
pn.Column(title, tabs).show(title="Netflix Analysis")
```

```
Launching server at http://localhost:50143
```

Out[67]:
```
<bokeh.server.server.Server at 0x7f8b61869040>
```

```
WARNING:bokeh.core.validation.check:W-1002 (EMPTY_LAYOUT): Layout has no children: Column(id='3209', ...)
WARNING:bokeh.core.validation.check:W-1002 (EMPTY_LAYOUT): Layout has no children: Column(id='3288', ...)
WARNING:bokeh.core.validation.check:W-1002 (EMPTY_LAYOUT): Layout has no children: Column(id='3352', ...)
WARNING:bokeh.core.validation.check:W-1002 (EMPTY_LAYOUT): Layout has no children: Column(id='3415', ...)
WARNING:bokeh.core.validation.check:W-1002 (EMPTY_LAYOUT): Layout has no children: Column(id='3478', ...)
WARNING:bokeh.core.validation.check:W-1002 (EMPTY_LAYOUT): Layout has no children: Column(id='3542', ...)
WARNING:bokeh.core.validation.check:W-1002 (EMPTY_LAYOUT): Layout has no children: Column(id='3611', ...)
WARNING:bokeh.core.validation.check:W-1002 (EMPTY_LAYOUT): Layout has no children: Column(id='3680', ...)
WARNING:bokeh.core.validation.check:W-1002 (EMPTY_LAYOUT): Layout has no children: Column(id='3697', ...)
WARNING:bokeh.core.validation.check:W-1002 (EMPTY_LAYOUT): Layout has no children: Column(id='3714', ...)
WARNING:bokeh.core.validation.check:W-1002 (EMPTY_LAYOUT): Layout has no children: Column(id='3782', ...)
WARNING:bokeh.core.validation.check:W-1002 (EMPTY_LAYOUT): Layout has no children: Column(id='4066', ...)
```

In [ ]:
```
#turn the notebook into a deployable app
pn.Column(title, tabs).servable(title="Netflix Analysis")
```

In [ ]:
```
1
```