Python初級數據分析員證書

（六）數據分析 及可視化專案

13. 數據分析專案

Demo 10 – Amazon Bestseller

Review

- Statistics
- Hypothesis testing
- Algebra
- Linear regression
- Propositional logic
- Python
- R
- SQL

- Pandas, NumPy, SciPy
- Data Visualization, Matplotlib, Seaborn, Plotly
- Dashboard Visualization, Business Intelligence
- Storytelling

Chapter Summary

- Scenario
- Wordcloud
- Missingno
- Data Import
- Ratings
- Reviews
- Author

Scenario

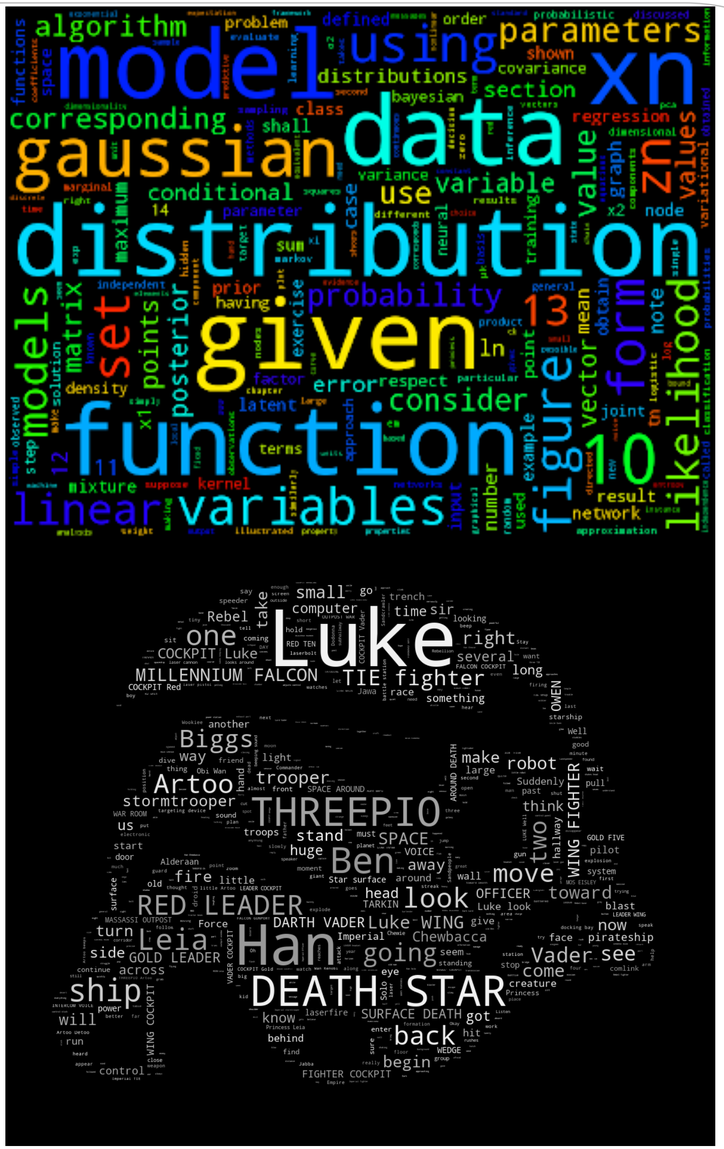
By analysing a database of Amazon Top 50 Bestselling Books 2009-2019, you are asked to get inspired for books trading for a local online book shop.

WordCloud

WordCloud are useful for visualizing common words in a text or data set.

```
1 pip install wordcloud
```

```
Collecting wordcloud
  Downloading wordcloud-1.9.1.1-cp39-cp39-mac
Requirement already satisfied: numpy>=1.6.1 i
om wordcloud) (1.23.1)
Requirement already satisfied: pillow in /Use
dcloud) (9.2.0)
Requirement already satisfied: matplotlib in
wordcloud) (3.5.2)
```

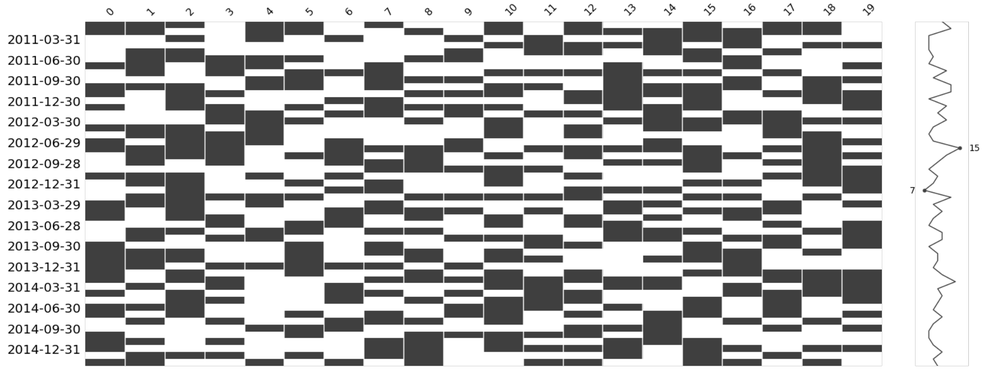


Missingno

Missingno is an excellent and simple to use Python library that provides a series of visualisations to understand the presence and distribution of missing data within a pandas DataFrame.

```
1 pip install missingno
```

```
Collecting missingno
  Downloading missingno-0.5.2-py3-none-any.whl
Requirement already satisfied: numpy in /Users/
ingno) (1.23.1)
Requirement already satisfied: matplotlib in /U
missingno) (3.5.2)
Requirement already satisfied: scipy in /Users/
ingno) (1.10.1)
```



Import needed library

```
import numpy as np
import pandas as pd

# Basic Visualization tools
import matplotlib
import matplotlib.pyplot as plt
plt.style.use('ggplot')
plt.rcParams['figure.dpi'] = 300
import seaborn as sns
sns.set_palette('husl')

plt.rc('figure',figsize=(17,13))
import plotly.express as px
import plotly.graph_objs as go
import plotly.offline as pyo
from plotly.subplots import make_subplots

# Special Visualization
import wordcloud, missingno
from wordcloud import WordCloud # wordcloud
import missingno as msno # check missing value
#import networkx as nx
```

```

# Plotly visualization
import plotly.offline as py
from plotly.offline import iplot, init_notebook_mode
import plotly.graph_objs as go
# Required to use plotly offline in jupyter notebook
py.init_notebook_mode(connected = True)

#importing plotly and cufflinks in offline mode
import cufflinks as cf
import plotly.offline
cf.go_offline()
cf.set_config_file(offline=False, world_readable=True)

# Display markdown formatted output like bold, italic bold etc
from IPython.display import Markdown
def bold(string):
    display(Markdown(string))
```

Data Import

```
1 data = pd.read_csv('bestsellers.csv')
2 data.head()
```

	Name	Author	User Rating	Reviews	Price	Year	Genre
0	10-Day Green Smoothie Cleanse	JJ Smith	4.7	17350	8	2016	Non Fiction
1	11/22/63: A Novel	Stephen King	4.6	2052	22	2011	Fiction
2	12 Rules for Life: An Antidote to Chaos	Jordan B. Peterson	4.7	18979	15	2018	Non Fiction
3	1984 (Signet Classics)	George Orwell	4.7	21424	6	2017	Fiction
4	5,000 Awesome Facts (About Everything!) (Natio...	National Geographic Kids	4.8	7665	12	2019	Non Fiction

```
1 data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 550 entries, 0 to 549
Data columns (total 7 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Name         550 non-null    object
 1   Author       550 non-null    object
 2   User Rating  550 non-null    float64
 3   Reviews      550 non-null    int64
 4   Price        550 non-null    int64
 5   Year         550 non-null    int64
 6   Genre        550 non-null    object
dtypes: float64(1), int64(3), object(3)
memory usage: 30.2+ KB
```

DF.describe(include='all')

Apart from using data.describe() for numerical data structure viewing, we may use data.describe(include=['O']) for string data viewing, and data.describe(include='all') for all

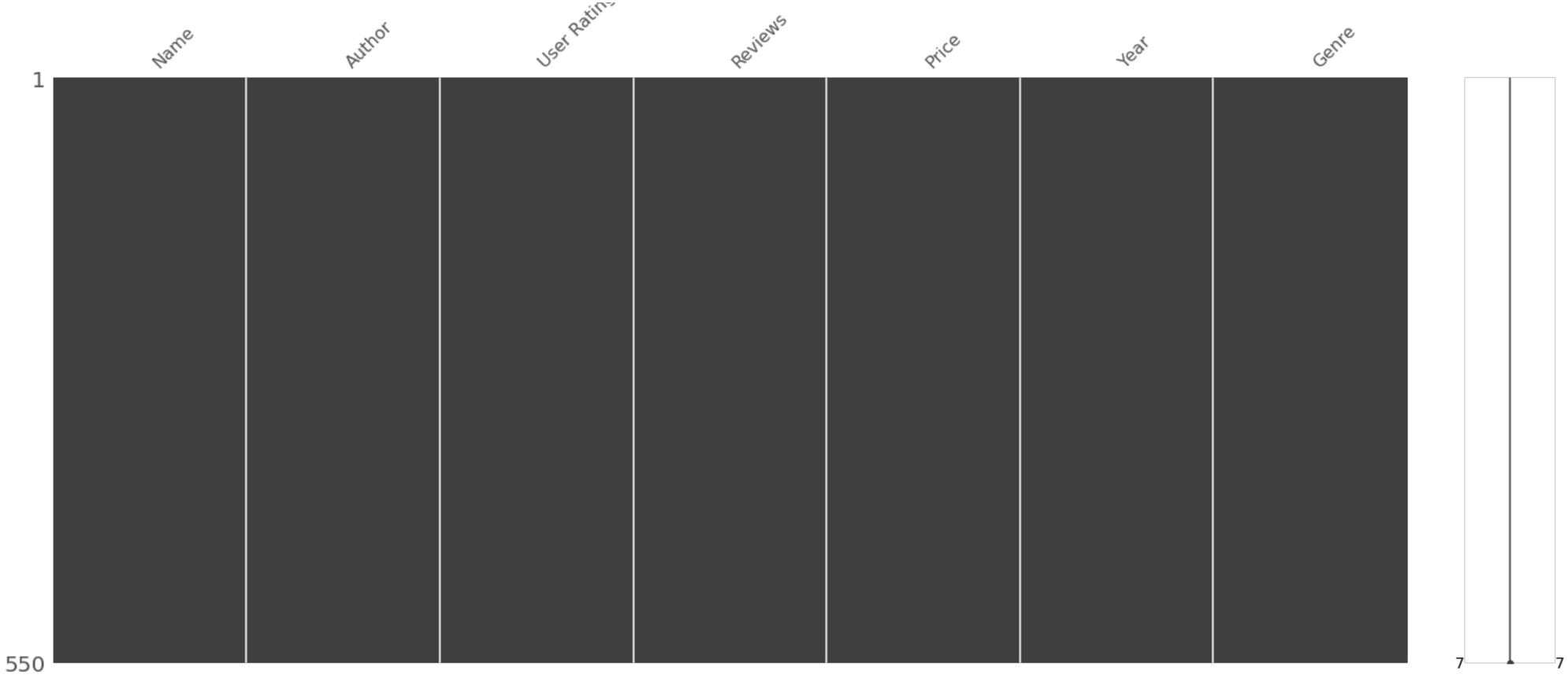
```
1 data.describe(include='all')
```

	Name	Author	User Rating	Reviews	Price	Year	Genre
count	550	550	550.000000	550.000000	550.000000	550.000000	550
unique	351	248	NaN	NaN	NaN	NaN	2
top	Publication Manual of the American Psychologic...	Jeff Kinney	NaN	NaN	NaN	NaN	Non Fiction
freq	10	12	NaN	NaN	NaN	NaN	310
mean	NaN	NaN	4.618364	11953.281818	13.100000	2014.000000	NaN
std	NaN	NaN	0.226980	11731.132017	10.842262	3.165156	NaN
min	NaN	NaN	3.300000	37.000000	0.000000	2009.000000	NaN
25%	NaN	NaN	4.500000	4058.000000	7.000000	2011.000000	NaN
50%	NaN	NaN	4.700000	8580.000000	11.000000	2014.000000	NaN
75%	NaN	NaN	4.800000	17253.250000	16.000000	2017.000000	NaN
max	NaN	NaN	4.900000	87841.000000	105.000000	2019.000000	NaN

Check missing data with missingno

```
1 msno.matrix(data)
```

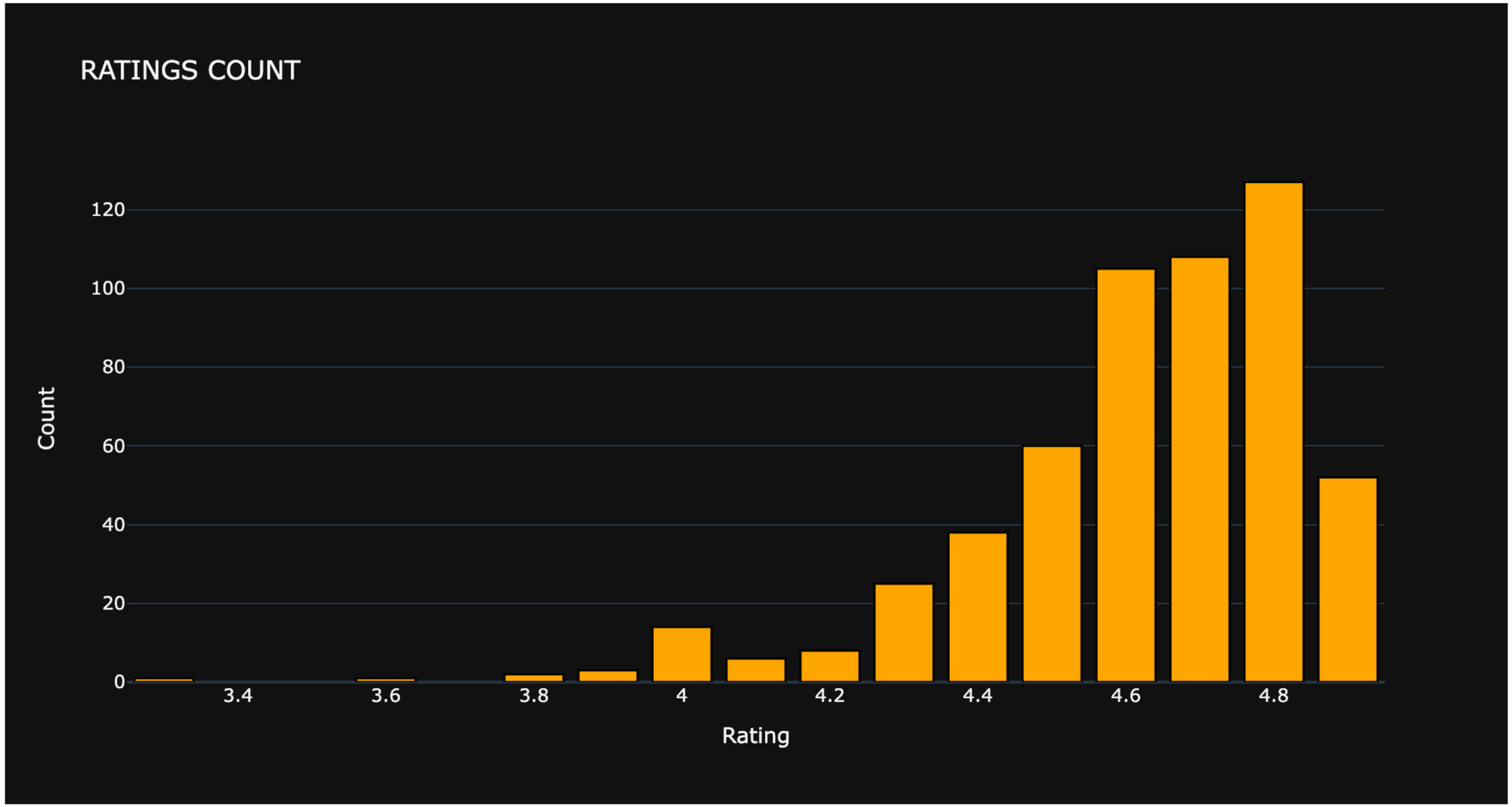
```
<AxesSubplot:>
```



Bestseller rating counts

```
temp_df = data['User Rating'].value_counts().reset_index()

# create trace1
trace1 = go.Bar(x = temp_df['index'], y = temp_df['User Rating'],
                marker = dict(color = 'rgb(255,165,0)',
                              line=dict(color='rgb(0,0,0)',width=1.5)))
layout = go.Layout(template= "plotly_dark",title = 'RATINGS COUNT' ,
                   xaxis = dict(title = 'Rating'), yaxis = dict(title = 'Count'))
fig = go.Figure(data = [trace1], layout = layout)
fig.show()
bold("**MOST OF THE RATINGS ARE IN THE RANGE OF 4.6 TO 4.8**")
```

Bestseller rating counts

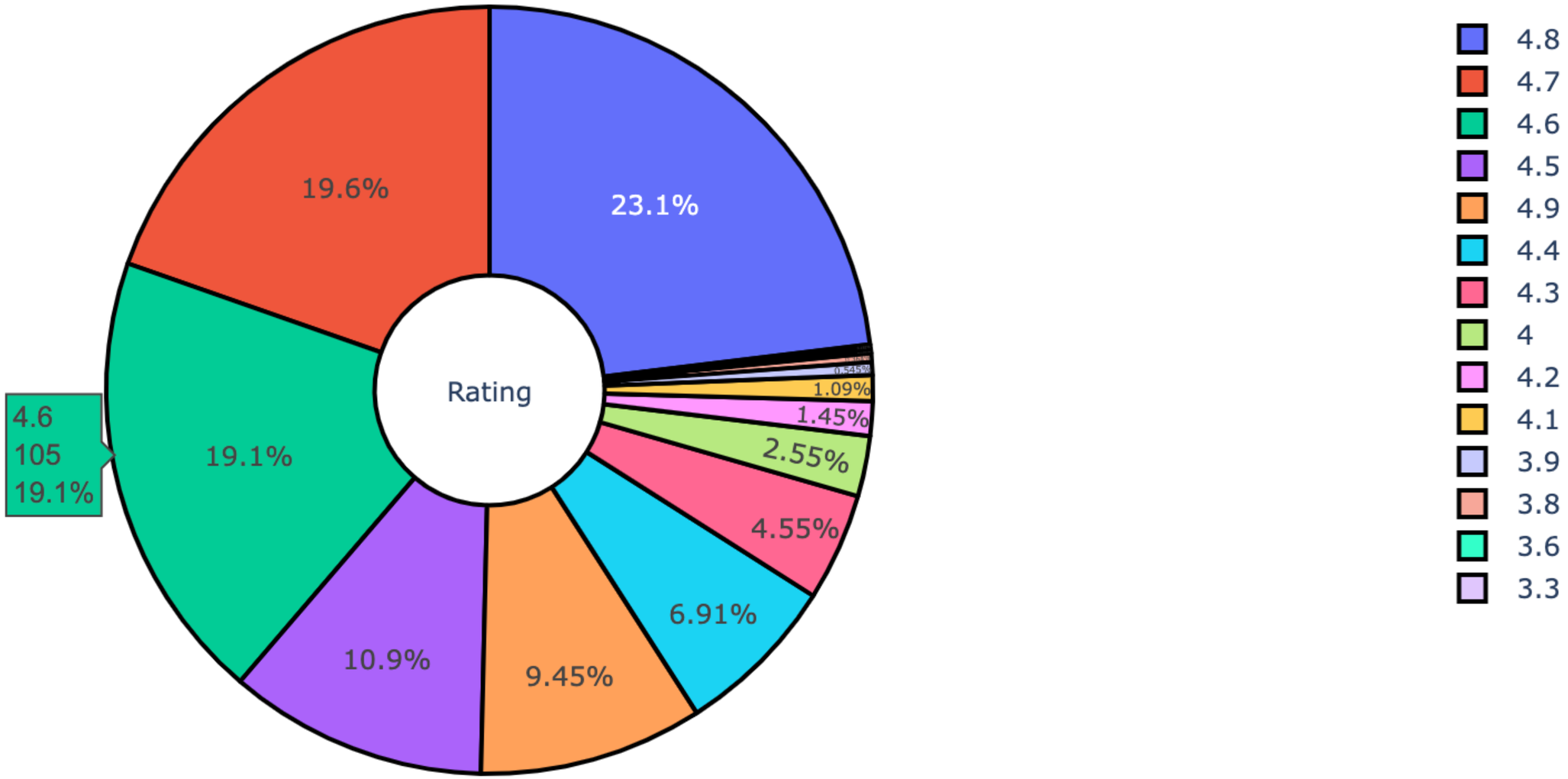


MOST OF THE RATINGS ARE IN THE RANGE OF 4.6 TO 4.8

Bestseller rating counts

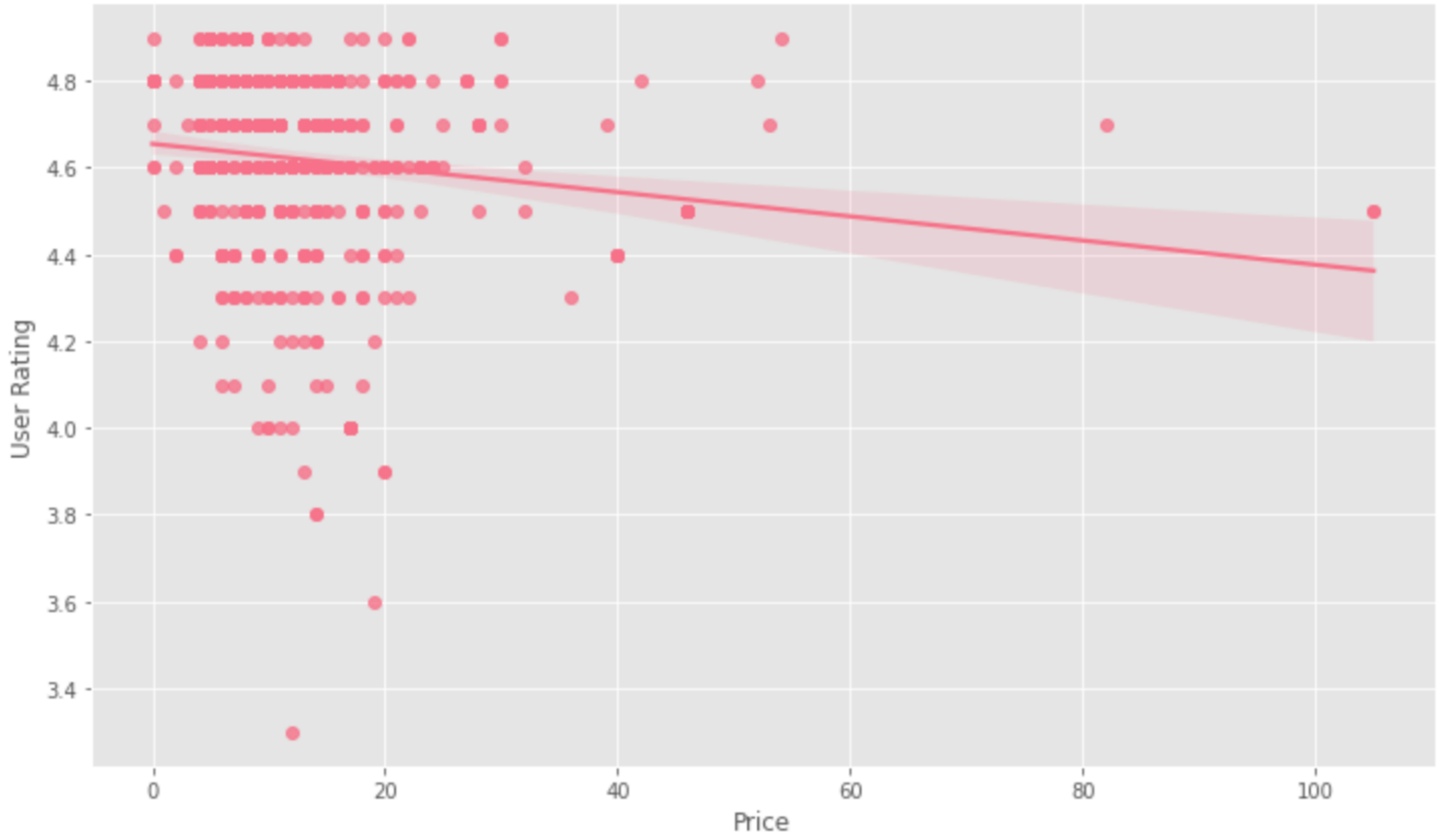
```
def pie_plot(cnt_srs, title):
    labels=cnt_srs.index
    values=cnt_srs.values
    trace = go.Pie(labels=labels,
                   values=values,
                   title=title,
                   hoverinfo=['percent+value'],
                   textinfo='percent',
                   textposition='inside',
                   hole=0.3,
                   showlegend=True,
                   marker=dict(colors=plt.cm.viridis_r(np.linspace(0, 1, 14)),
                               line=dict(color='#000000', width=2)  )
                  )
    return trace
py.iplot([pie_plot(data['User Rating'].value_counts(), 'Rating')])
```

Bestseller rating counts



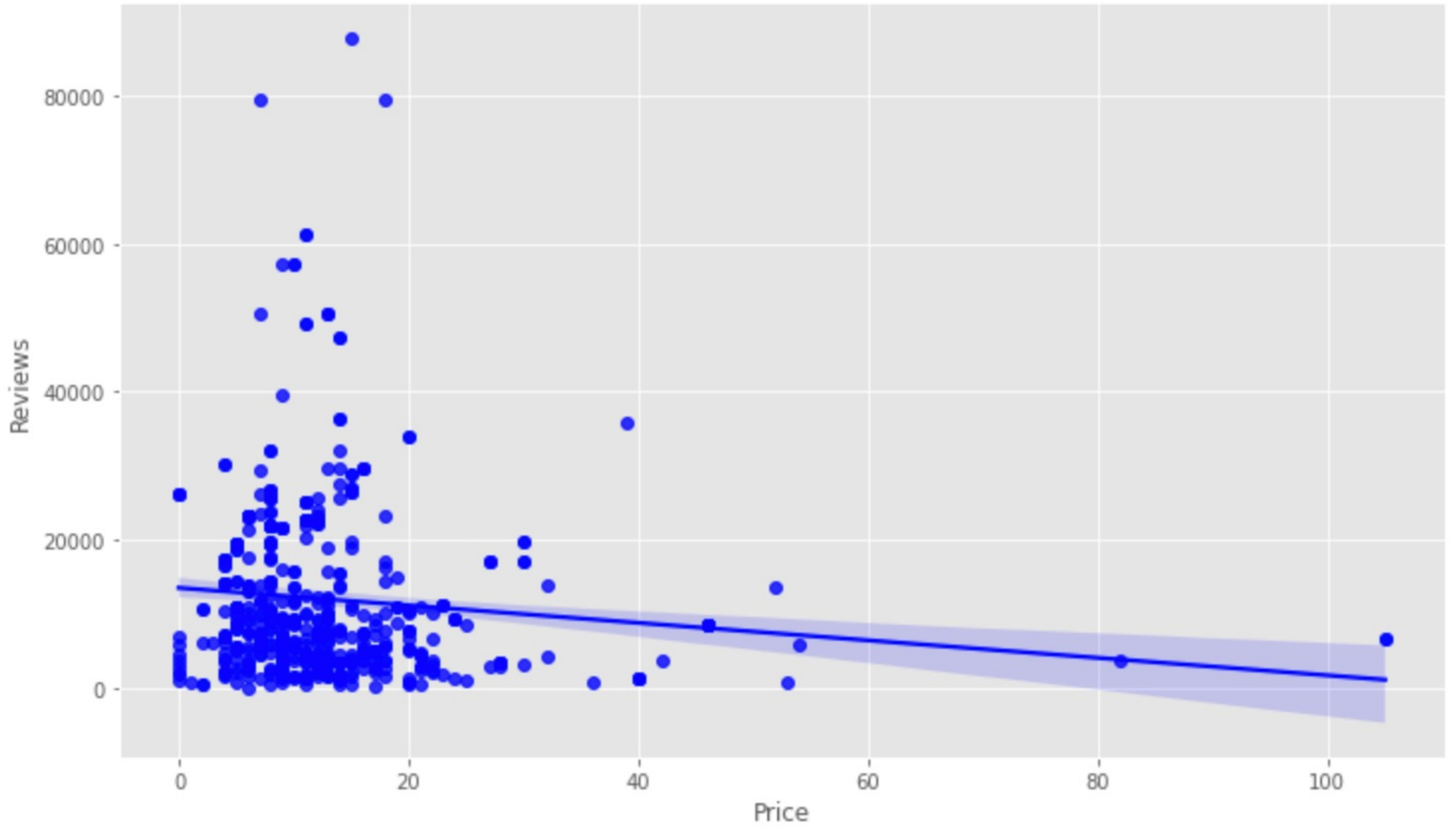
Relationship between Price and User Rating

```
1 fig, ax = plt.subplots(1,1, figsize=(12, 7), dpi=72)
2 sns.regplot(data=data, x='Price', y='User Rating', ax=ax)
3 plt.show()
```



Relationship between Price and Reviews

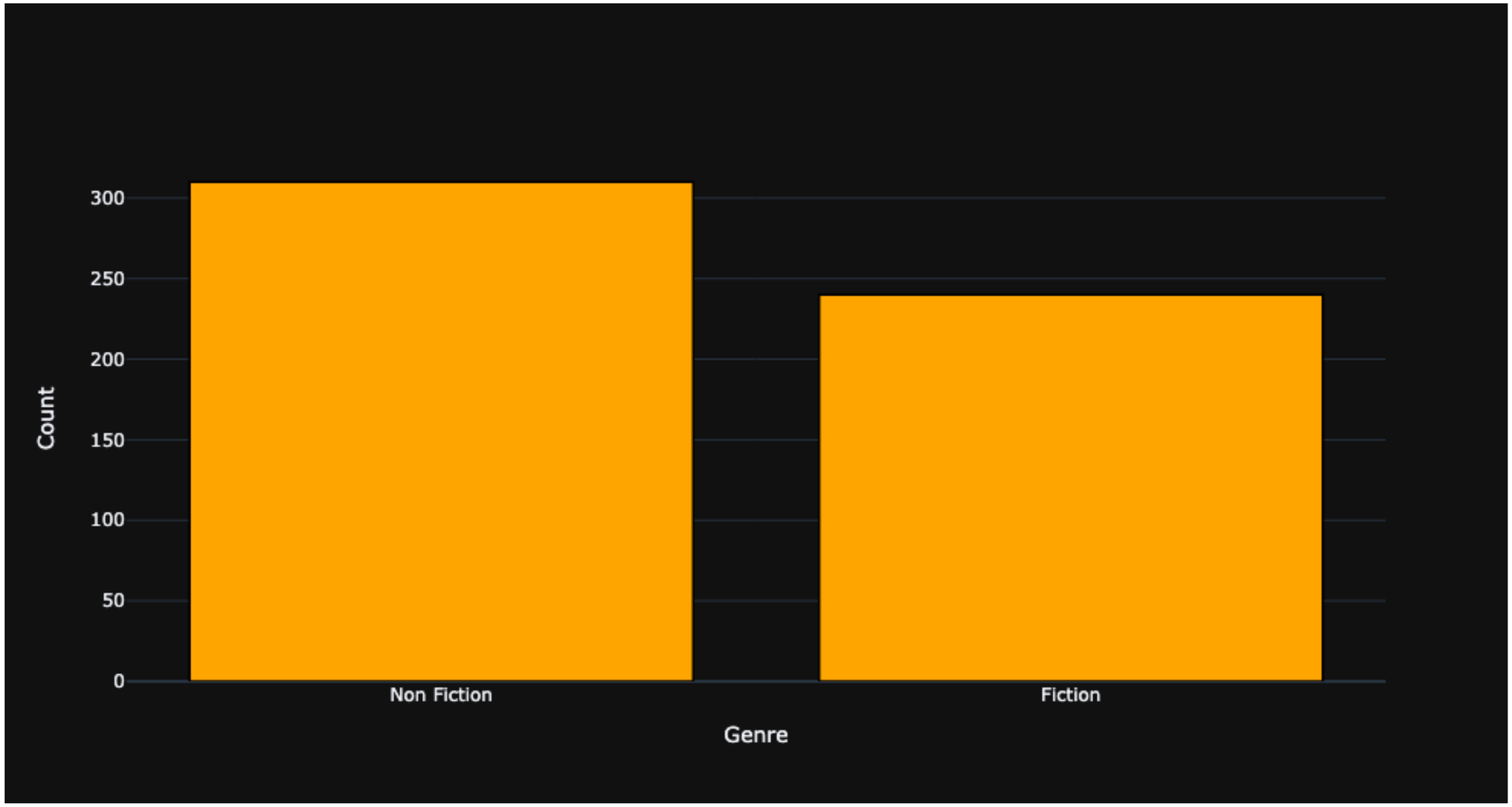
```
1 fig, ax = plt.subplots(1,1, figsize=(12, 7), dpi=72)
2 sns.regplot(data=data, x='Price', y='Reviews', color='b', ax=ax)
3 plt.show()
```



Fiction and Non-fiction Bestseller

```
temp_df = data['Genre'].value_counts().reset_index()

trace1 = go.Bar(
                x = temp_df['index'],
                y = temp_df['Genre'],
                marker = dict(color = 'rgb(255,165,0)',
                              line=dict(color='rgb(0,0,0)',width=1.5)))

layout = go.Layout(template= "plotly_dark", xaxis = dict(title = 'Genre'),
                   yaxis = dict(title = 'Count'))
fig = go.Figure(data = [trace1], layout = layout)
fig.show()
```

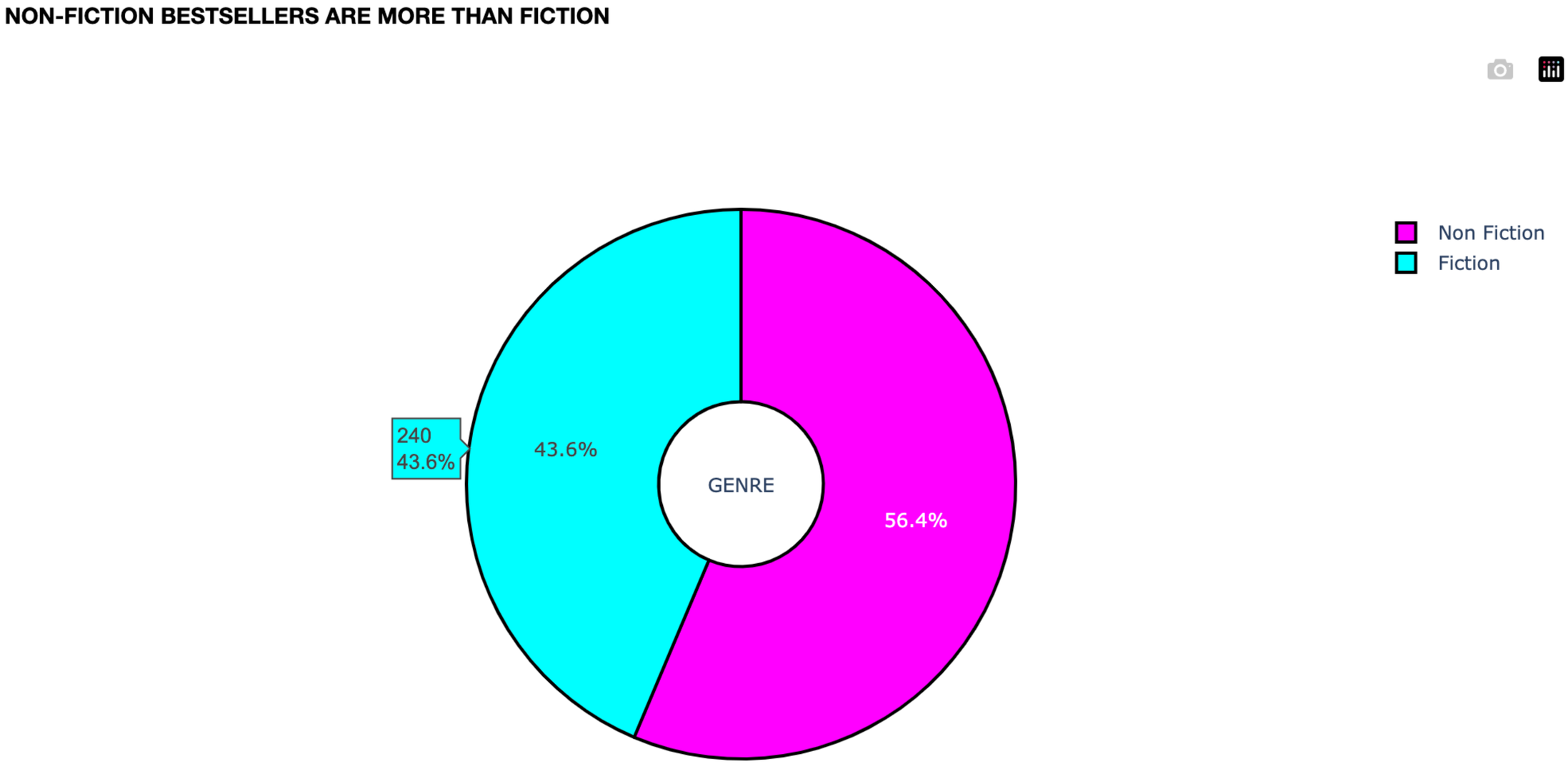
Fiction and Non-fiction Bestseller



Fiction and Non-fiction Bestseller

```
def pie_plot(cnt_srs, colors, title):
    '''A Function To Plot Pie Plot using Plotly'''
    labels=cnt_srs.index
    values=cnt_srs.values
    trace = go.Pie(labels=labels,
                   values=values,
                   title=title,
                   hoverinfo='percent+value',
                   textinfo='percent',
                   textposition='inside',
                   hole=0.3,
                   showlegend=True,
                   marker=dict(colors=colors,
                               line=dict(color='#000000',
                                         width=2),  )
                  )
    return trace

bold("**NON-FICTION BESTSELLERS ARE MORE THAN FICTION**")
py.iplot([pie_plot(data['Genre'].value_counts(), ['magenta', 'cyan'], 'GENRE')])
```

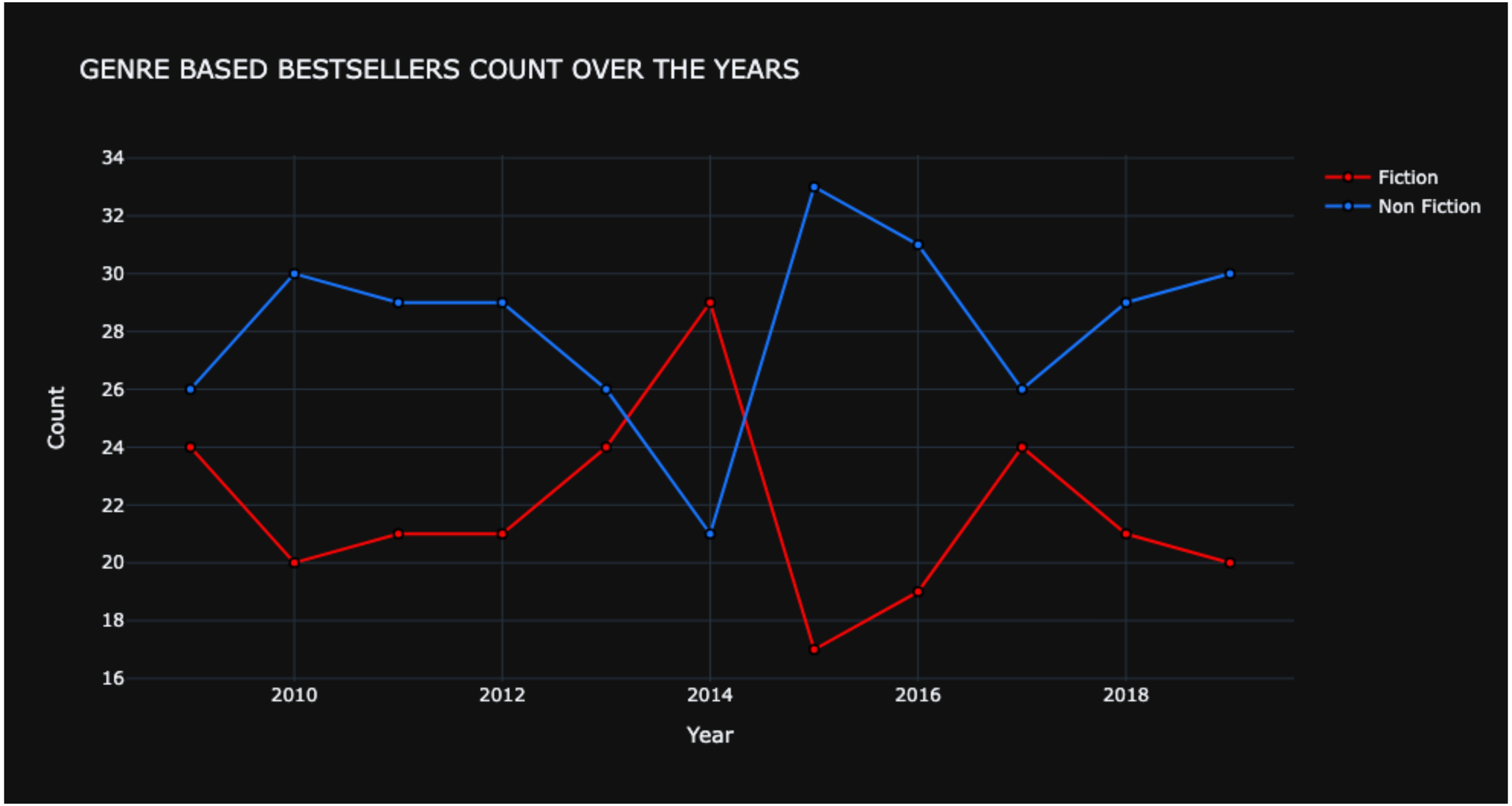
Fiction and Non-fiction Bestseller



Bestseller count based on genre over the years

```
d1 = data[data["Genre"] == "Fiction"]
d2 = data[data["Genre"] == "Non Fiction"]

# Fiction
vc1 = pd.DataFrame(data={'count' : d1.Year.value_counts() })
vc1['percent'] = vc1['count'].apply(lambda x : 100*x/sum(vc1['count']))
vc1 = vc1.sort_index()

# Non-Fiction
vc2 = pd.DataFrame(data={'count' : d2.Year.value_counts() })
vc2['percent'] = vc2['count'].apply(lambda x : 100*x/sum(vc2['count']))
vc2 = vc2.sort_index()

trace1 = go.Scatter(x=vc1.index, y=vc1["count"], name="Fiction",
                        marker=dict(color = 'rgb(249, 6, 6)',
                        line=dict(color='rgb(0,0,0)',width=1.5)))

trace2 = go.Scatter(x=vc2.index, y=vc2["count"], name="Non Fiction",
                        marker= dict(color = 'rgb(26, 118, 255)',
                        line=dict(color='rgb(0,0,0)',width=1.5)))
layout = go.Layout(hovermode= 'closest',
                    title = 'GENRE BASED BESTSELLERS COUNT OVER THE YEARS' ,
                    xaxis = dict(title = 'Year'), yaxis = dict(title = 'Count'),
                    template= "plotly_dark")
fig = go.Figure(data = [trace1, trace2], layout=layout)
fig.show()
```

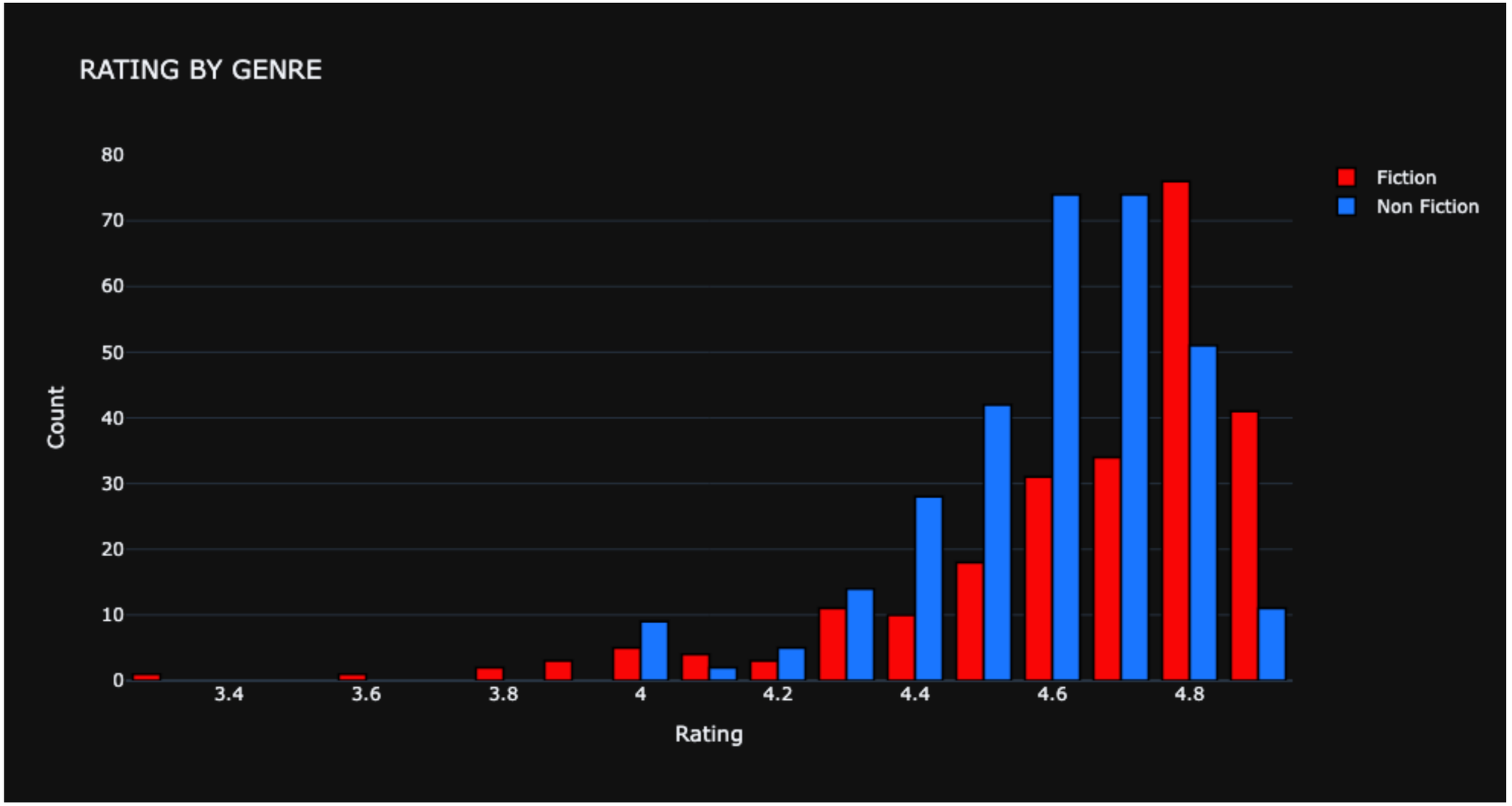
Bestseller count based on genre over the years



Rating on genre

```
df1 = data[data["Genre"] == "Fiction"]
df2 = data[data["Genre"] == "Non Fiction"]

temp_df1 = pd.DataFrame(data={'Rating':df1['User Rating'].value_counts()}).sort_index()
temp_df2 = pd.DataFrame(data={'Rating':df2['User Rating'].value_counts()}).sort_index()

trace1 = go.Bar(x = temp_df1.index, y = temp_df1['Rating'],
                name="Fiction", marker = dict(color = 'rgb(249, 6, 6)',
                                line=dict(color='rgb(0,0,0)',width=1.5)))
# create trace2
trace2 = go.Bar(x = temp_df2.index, y = temp_df2['Rating'],
            name = "Non Fiction", marker = dict(color = 'rgb(26, 118, 255)',
                            line=dict(color='rgb(0,0,0)',width=1.5)))

layout = go.Layout(template= "plotly_dark",title = 'RATING BY GENRE' ,
                   xaxis = dict(title = 'Rating'), yaxis = dict(title = 'Count'))
fig = go.Figure(data = [trace1, trace2], layout = layout)
fig.show()
```

Rating on genre



Top 9 multiple time bestseller

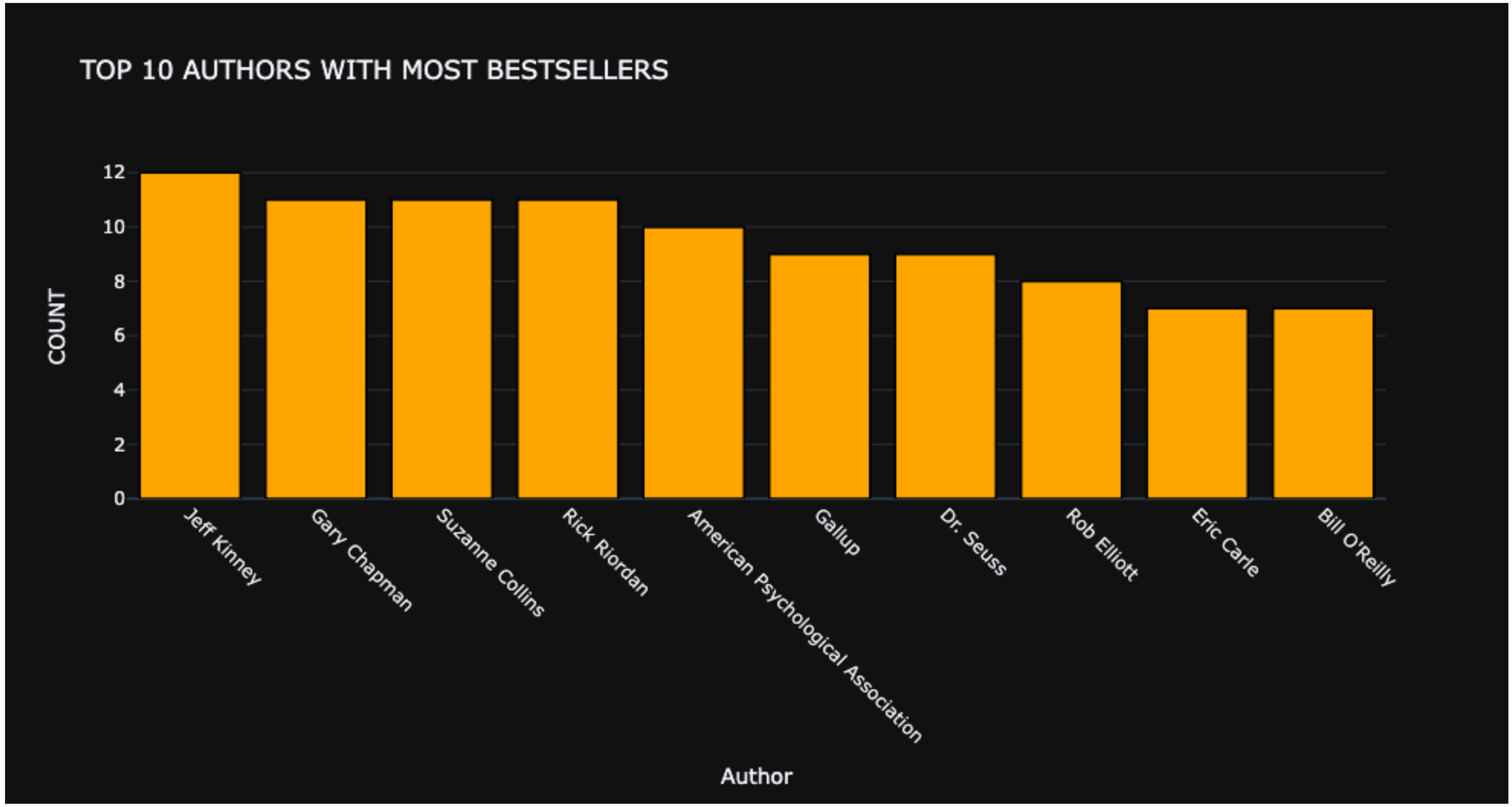
```
common_books = data['Name'].value_counts()[:9].rename_axis('Common Books')\
.reset_index(name='count')

fig = px.treemap(common_books, path=['Common Books'],
                 values='count',title='TOP 9 MULTIPLE TIMES BESTSELLERS')

fig.show()
```

Top 9 multiple time bestseller

TOP 10 AUTHORS WITH MOST BESTSELLERS

```
temp_df1 = data.groupby('Author').count().reset_index().sort_values('Name',ascending=False).head(10)

trace1 = go.Bar(
                x = temp_df1['Author'],
                y = temp_df1['Name'],
                marker = dict(color = 'rgb(255,165,0)',
                              line=dict(color='rgb(0,0,0)',width=1.5)))
layout = go.Layout(template= "plotly_dark",title = 'TOP 10 AUTHORS WITH MOST BESTSELLERS ' ,
                   xaxis = dict(title = 'Author',tickangle=45), yaxis = dict(title = 'COUNT'))
fig = go.Figure(data = [trace1], layout = layout)
fig.show()
```

TOP 10 AUTHORS WITH MOST BESTSELLERS



Authors with bestsellers

Authors with bestsellers in both the genre: fiction and non-fiction

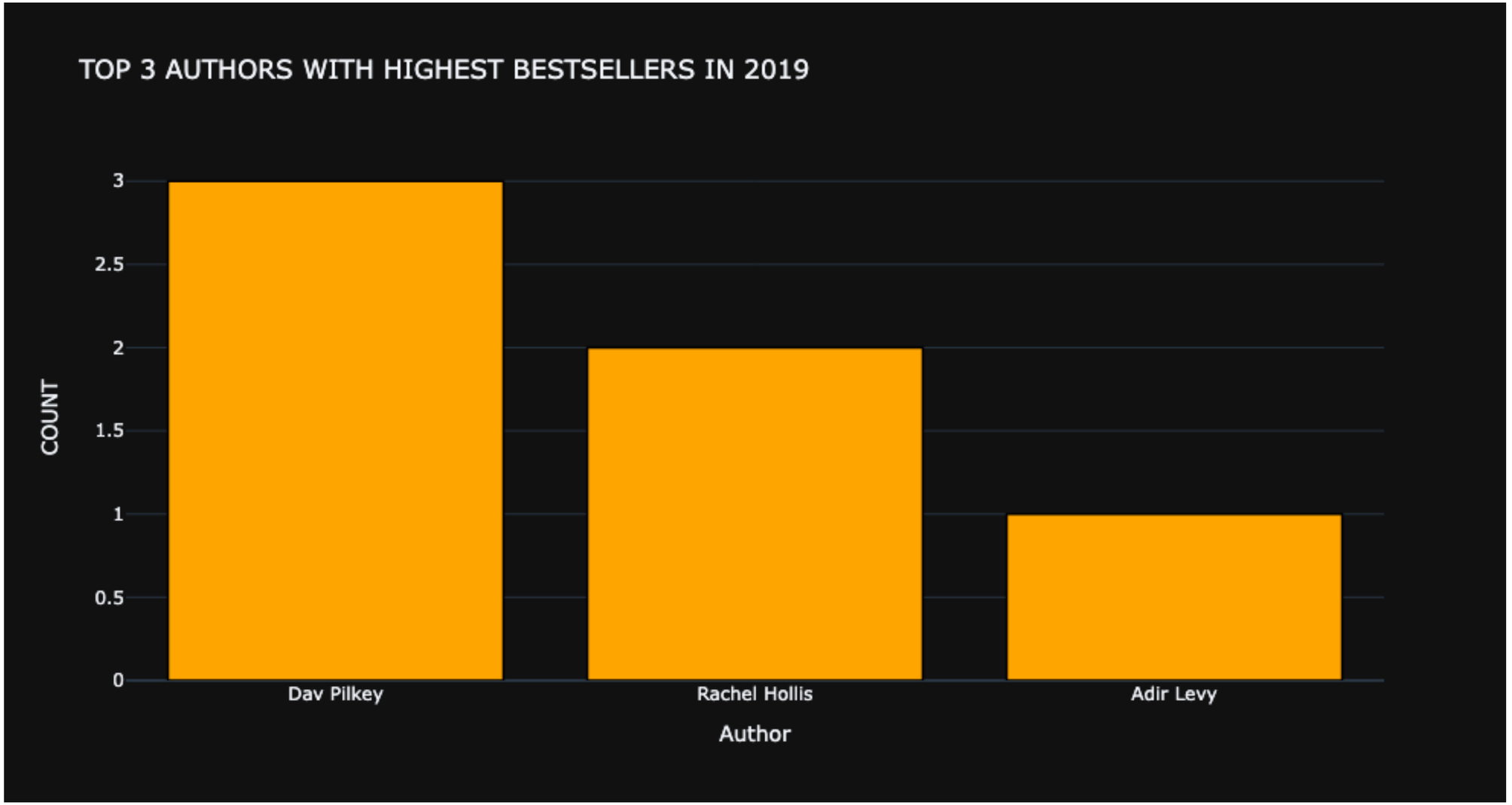
```
data[data['Author'].isin(list(data.groupby(['Author','Genre']).count().reset_index()\
                              ['Author'].value_counts()[:2].index))]
```

	Name	Author	User Rating	Reviews	Price	Year	Genre
28	Baby Touch and Feel: Animals	DK	4.6	5360	5	2015	Non Fiction
158	Harry Potter Coloring Book	Scholastic	4.7	3564	9	2015	Non Fiction
268	Pokémon Deluxe Essential Handbook: The Need-to...	Scholastic	4.7	3503	9	2016	Fiction
514	Ultimate Sticker Book: Frozen: More Than 60 Re...	DK	4.5	2586	5	2014	Fiction

Top 3 authors with highest bestsellers in 2019

```
temp_df1=data[data['Year']==2019].groupby('Author').count().reset_index()\
.sort_values('Name',ascending=False).head(3)

trace1 = go.Bar(
                x = temp_df1['Author'],
                y = temp_df1['Name'],
                marker = dict(color = 'rgb(255,165,0)',
                              line=dict(color='rgb(0,0,0)',width=1.5)))
layout = go.Layout(template= "plotly_dark",title = 'TOP 3 AUTHORS WITH HIGHEST BESTSELLERS IN 2019 ' ,
                   xaxis = dict(title = 'Author'), yaxis = dict(title = 'COUNT'))
fig = go.Figure(data = [trace1], layout = layout)
fig.show()
```

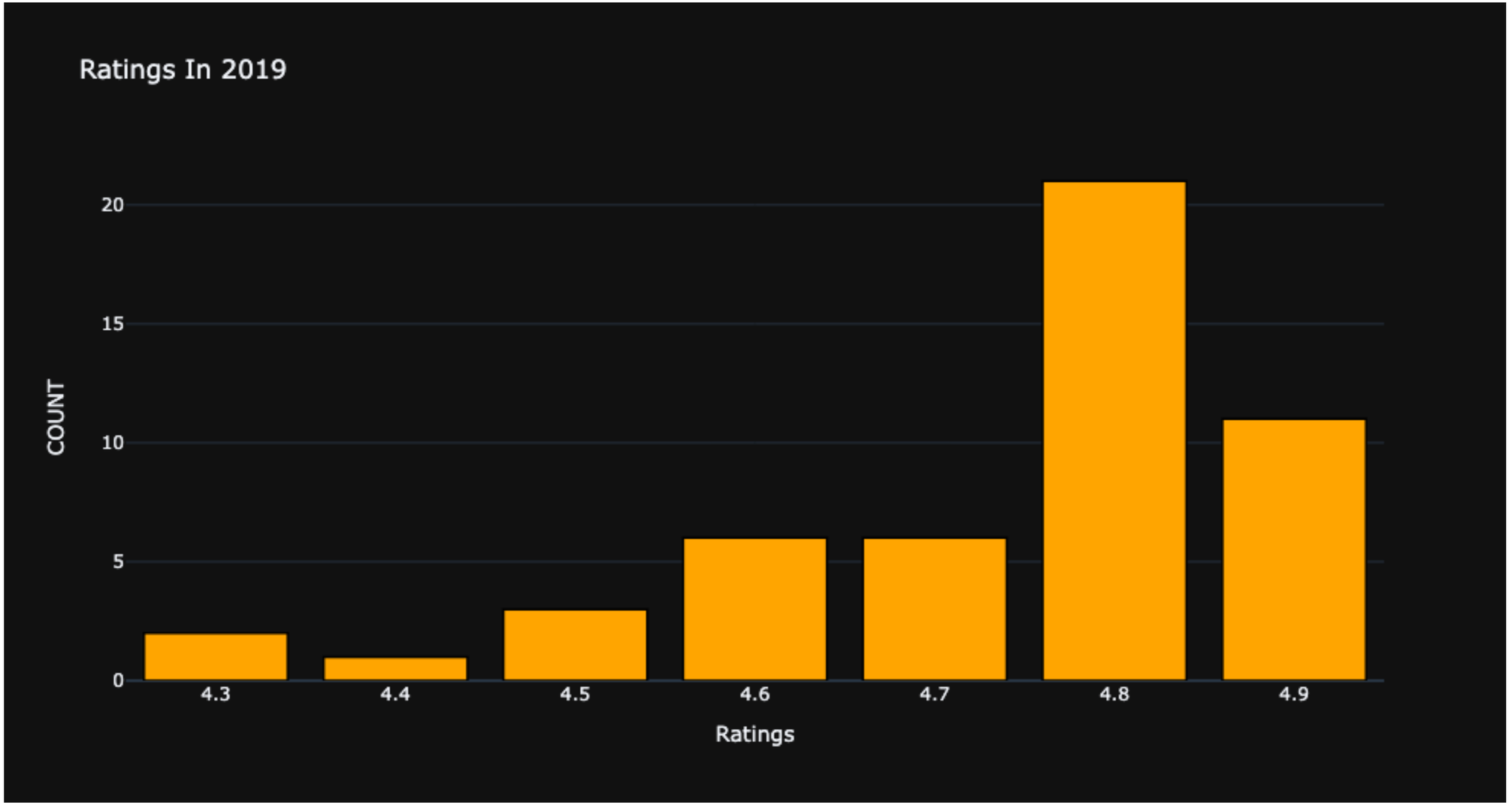
Top 3 authors with highest bestsellers in 2019



Rating in 2019

```
temp_df1=data[data['Year']==2019].groupby('User Rating').count().reset_index()\
.sort_values('Name',ascending=False)

trace1 = go.Bar(
                x = temp_df1['User Rating'],
                y = temp_df1['Name'],
                marker = dict(color = 'rgb(255,165,0)',
                              line=dict(color='rgb(0,0,0)',width=1.5)))
layout = go.Layout(template= "plotly_dark",title = 'Ratings In 2019 ' ,
                   xaxis = dict(title = 'Ratings'), yaxis = dict(title = 'COUNT'))
fig = go.Figure(data = [trace1], layout = layout)
fig.show()
```

Rating in 2019



Chapter Wrap Up

Open Discussion:

- What is the key of the bestseller
- What is the genre and content shall the customer like
- What ideas shall we get from rating and review
- What further data we shall get for better estimation
- What books would you suggest to sell

Reference & Resources

Official Website:

https://plotly.com/python/

Plotly Graph Objects:

https://plotly.com/python/graph-objects/

Kaggle dataset:

https://www.kaggle.com/datasets

WorldBank API:

- https://blogs.worldbank.org/opendata/introducing-wbgapi-new-python-package-accessing-world-bank-data
- https://nbviewer.org/github/tgherzog/wbgapi/blob/master/examples/wbgapi-cookbook.ipynb
- https://pypi.org/project/wbgapi/

GitHub Open Source Code:

https://github.com/plotly/plotly.py