AnalizaFilmova

March 28, 2025

Testiranje

[1]:
```
import pandas as pd
import numpy as np
import operator
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

Analiza filmova

Projekat iz Statističkih osnova inteligentne obrade podataka

Veljko Vračarević RN140

Opis:

Sadržaj:

Prikupljanje podataka

Sređivanje podataka

Analiza filmova po žanrovima

Analiza filmova po godinama

Skup podataka se sastoji od filmova i ponekih serija (pretežno filmovi) koje sam ranije gledao. Posmatraju se sledeća obeležja:

Budžet

Trajanje

Žanr

Ocena

Godina

Kratak opis posmatranih obeležja:

Budžet: Budžet za snimanje filma. Mnogi budžeti koje sam nalazio su dati u vidu neke procene od-do, u tom slučaju sam uzimao aritmetičku sredinu.

Trajanje: Dužina filma izražena u minutima.

Žanr: Žanrovi su prikupljeni u vidu liste oblika [‘žanr1’, ‘žanr2’, ...], neki imaju više neki manje žanrova kojima pripadaju.

Ocena: Od 0 do 5 na osnovu ličnog utiska, ocene mogu biti i oblika 0,5, 1,5 itd.

Godina: Godina koje je film izašao u bioskope.

Prikupljanje podataka

Za prikupljanje podataka sam koristio svoju već postojeću listu odgledanih filmova i serija na sajtu Letterboxd. (link do sajta) Eksportovane tabele sa podacima nisu posedovale sva obeležja potrebna za ovu analizu pa su žanrovi i dužine filmova naknadno unesene pomoću python web scrapera a budžeti ručno traženi za svaki element skupa. Za web scraper su korišćeni linkovi do svakog filma koji su eksportovani sa sajta. Delovi koda kao i njihov kratak opis nalaze se u nastavku:

```
from bs4 import BeautifulSoup
import requests
import pandas as pd
import numpy as np

data_frame = pd.read_csv(r'C:\Users\cho\Desktop\StatistikaProj\data\filmovi.csv')

links = np.asarray(data_frame["Letterboxd URI"])
length_rows = []
count = 0

for link in links:
    page_to_scrape = requests.get(link)

    soup = BeautifulSoup(page_to_scrape.text, "html.parser")
    p_tag = soup.find('p', class_='text-link text-footer')
    if p_tag:
        text = p_tag.get_text(strip=True)

        # filter for numeric characters
        minutes = ''.join(filter(str.isdigit, text.split('mins')[0].strip()))

        length_rows.append(minutes)

    count = count + 1
    print("Count: ", count)

df = pd.DataFrame(length_rows)

output_file = 'lengths.csv'
df.to_csv(output_file, index=False, header=False)

print(f"Lengths saved to {output_file}")
```

Sređivanje podataka

Obzirom na to da nisam uspeo da nađem budžete svih odgledanih filmova, filmovi bez budžeta (Budžet = 0) neće ući u analizu.

[2]:
```
data_frame = pd.read_csv('data/filmovi.csv')
data_frame
```

[2]:
```
                          Title     Budget  Year  \
0                        13 Sins    4000000  2014
1              A Monster Calls   43000000  2016
2    A Nightmare on Elm Street    1800000  1984
3                A Serbian Film          0  2010
4                       Aladdin   28000000  1992
..                          …        …   …
342   X-Men Origins: Wolverine  175000000  2009
343                     Yes Man   70000000  2008
344                   Yu-Gi-Oh!          0  1998
345                 Zombieland   23600000  2009
346                    Heretic   10000000  2024

                                        Genre  Rating        Letterboxd URI  \
0                             Thriller|Horror     3.5  https://boxd.it/4Ste
1      Animation|Fantasy|Adventure|Family|Drama     5.0  https://boxd.it/7DjE
2                                      Horror     3.5  https://boxd.it/2aw0
3                       Horror|Thriller|Crime     1.5  https://boxd.it/2wBc
4    Adventure|Romance|Family|Fantasy|Animation     4.0  https://boxd.it/29yE
..                                         …     …                     …
342            ScienceFiction|Adventure|Action     4.0  https://boxd.it/27ug
343                          Comedy|Romance     3.5  https://boxd.it/1WPW
344                  Family|Animation|Comedy     5.0  https://boxd.it/u3RQ
345                           Comedy|Horror     3.5  https://boxd.it/1En6
346                          Thriller|Horror     4.0  https://boxd.it/H1lY

     Length
0        93
1       108
2        91
3       104
4        95
..      …
342     107
343     104
344     648
345      88
346     111

[347 rows x 7 columns]
```

[3]:
```
data_frame = data_frame[data_frame.Budget != 0]
data_frame
```

[3]:

	Title	Budget	Year	\
0	13 Sins	4000000	2014	
1	A Monster Calls	43000000	2016	
2	A Nightmare on Elm Street	1800000	1984	
4	Aladdin	28000000	1992	
5	Alien	10700000	1979	
..	…	…	…	
341	Wrong Turn 3: Left for Dead	2000000	2009	
342	X-Men Origins: Wolverine	175000000	2009	
343	Yes Man	70000000	2008	
345	Zombieland	23600000	2009	
346	Heretic	10000000	2024	

	Genre	Rating	Letterboxd URI	\
0	Thriller|Horror	3.5	https://boxd.it/4Ste	
1	Animation|Fantasy|Adventure|Family|Drama	5.0	https://boxd.it/7DjE	
2	Horror	3.5	https://boxd.it/2aw0	
4	Adventure|Romance|Family|Fantasy|Animation	4.0	https://boxd.it/29yE	
5	Horror|ScienceFiction	4.5	https://boxd.it/2awY	
..	…	…	…	
341	Thriller|Horror	2.0	https://boxd.it/1yJ6	
342	ScienceFiction|Adventure|Action	4.0	https://boxd.it/27ug	
343	Comedy|Romance	3.5	https://boxd.it/1WPW	
345	Comedy|Horror	3.5	https://boxd.it/1En6	
346	Thriller|Horror	4.0	https://boxd.it/H1lY	

	Length
0	93
1	108
2	91
4	95
5	117
..	…
341	91
342	107
343	104
345	88
346	111

[273 rows x 7 columns]

[4]:
```
data_frame.describe()
```

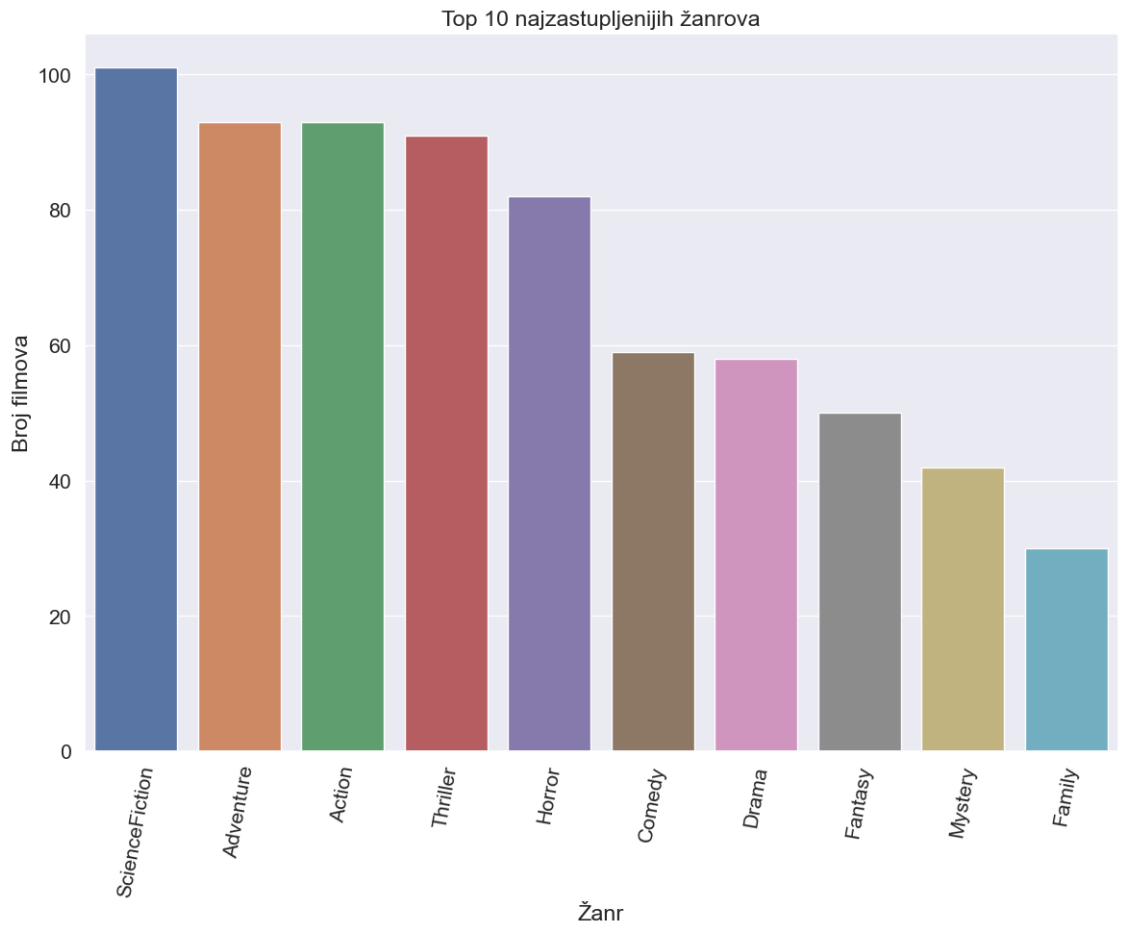
[4]:

	Budget	Year	Rating	Length
count	2.730000e+02	273.000000	262.000000	273.000000
mean	7.483736e+07	2007.520147	3.763359	115.029304
std	8.801633e+07	10.604199	0.848321	35.927918
min	7.000000e+03	1937.000000	0.000000	48.000000
25%	1.070000e+07	2003.000000	3.000000	93.000000
50%	4.000000e+07	2009.000000	4.000000	108.000000
75%	1.150000e+08	2014.000000	4.500000	128.000000
max	4.600000e+08	2024.000000	5.000000	495.000000

Analiza filmova

[14]:
```
genre_dict = {}

genres = data_frame["Genre"]
genres = genres.str.split("|")
genres = np.array(genres)
for genreList in genres:
    #check if there is a problematic list which is just a float
    for genre in genreList:
        genre = genre.lstrip() #trim the whitespaces
        if genre not in genre_dict:
            genre_dict[genre] = 1
        else:
            genre_dict[genre] += 1



sorted_genre_dict = sorted(genre_dict.items(), key = operator.itemgetter(1),␣
 ↪reverse = True)

# Prepare data for plotting
x_axis = []
y_axis = []
for item in sorted_genre_dict[:10]:
    x_axis.append(item[0])
    y_axis.append(item[1])

# Plot the bar chart
sns.set(rc={'figure.figsize':(12,10)}, font_scale=1.4)
ax = sns.barplot(x=x_axis, y=y_axis, hue=x_axis, legend=False)  # Fixed the␣
 ↪issue by using keyword arguments

# Rotate x-axis labels for better readability
for item in ax.get_xticklabels():
    item.set_rotation(80)
```

```
# Set plot labels and title
ax.set(xlabel='Žanr', ylabel='Broj filmova', title='Top 10 najzastupljenijih␣
 ↪žanrova')

# Show the plot
plt.tight_layout()
plt.show()
```



[6]:
```
year_set = set()
genre_set = set()
genres_and_year = data_frame[["Genre", "Year"]]

#########################
#create a set of unique years of movies
#########################
production_year = genres_and_year["Year"]
production_year = production_year.drop_duplicates()
for year in production_year:
```

```
    if year not in year_set:
        year_set.add(year)

#print(year_set)


#################################################################
#create a set of unique genres by parsing all the years
#################################################################
for year in year_set:
    genre_dict = {}
    genres_in_year = genres_and_year[genres_and_year.Year == year]
    genres_in_year = genres_in_year["Genre"].values
    for elem in genres_in_year:
        genres_row = elem.split("|")
        for genre in genres_row:
            if genre not in genre_set:
                genre_set.add(genre)


#print("year:", year, "\n", sorted(genre_dict.items(), key = operator.
 ↪itemgetter(1), reverse = True))


#################################################################################
#create a dataframe which contains the sum of movies' genre per year
#################################################################################
gerne_count_per_year_df = pd.DataFrame(index = sorted(year_set),␣
 ↪columns=sorted(genre_set))
gerne_count_per_year_df[:] = 0

for year in year_set:
    genre_dict = {}
    genres_in_year = genres_and_year[genres_and_year.Year == year]
    genres_in_year = genres_in_year["Genre"].values
    for elem in genres_in_year:
        genres_row = elem.split("|")
        for genre in genres_row:
            if genre not in genre_dict:
                genre_dict[genre] = 1
            else:
                genre_dict[genre] = genre_dict[genre] + 1

    aux_df = pd.DataFrame(genre_dict, index = [year])
    gerne_count_per_year_df.loc[year, aux_df.columns] = gerne_count_per_year_df.
 ↪loc[year, aux_df.columns] + aux_df.loc[year]
```

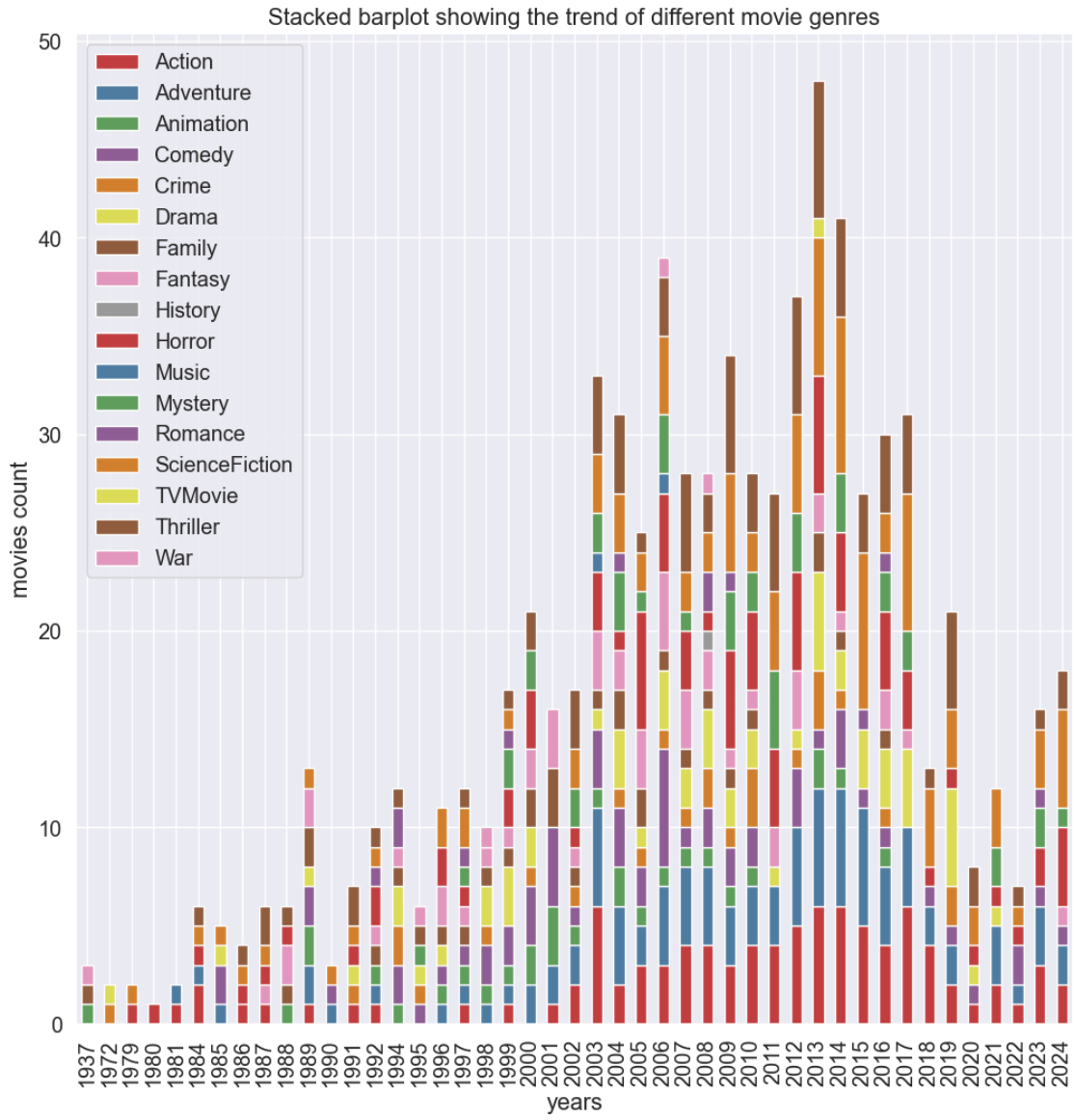
```
#########################################################
###most popular genre of movies from year to year
#########################################################
#print(gerne_count_per_year_df.apply( max, axis=1 ))
#print(gerne_count_per_year_df.idxmax(axis = 1))
most_popular_genre_by_year = pd.DataFrame([gerne_count_per_year_df.idxmax(axis␣
 ↪= 1).values,
                                           gerne_count_per_year_df.apply( max,␣
 ↪axis=1 ).values],
                                           columns = gerne_count_per_year_df.
 ↪index,
                                          index = ["genre", 'counts'])
```

[7]:
```
most_popular_genre_by_year
```

[7]:
```
             1937   1972    1979    1980    1981    1984    1985    1986  \
genre   Animation  Crime  Horror  Horror  Action  Action  Comedy  Action
counts          1      1       1       1       1       2       2       1

            1987     1988  …            2015    2016            2017  \
genre   Thriller  Fantasy  …  ScienceFiction  Action  ScienceFiction
counts         2        2  …               8       4               7

          2018   2019            2020       2021    2022    2023  \
genre   Action  Drama  ScienceFiction  Adventure  Comedy  Action
counts       4      5               2          3       2       3

                 2024
genre   ScienceFiction
counts               5

[2 rows x 45 columns]
```

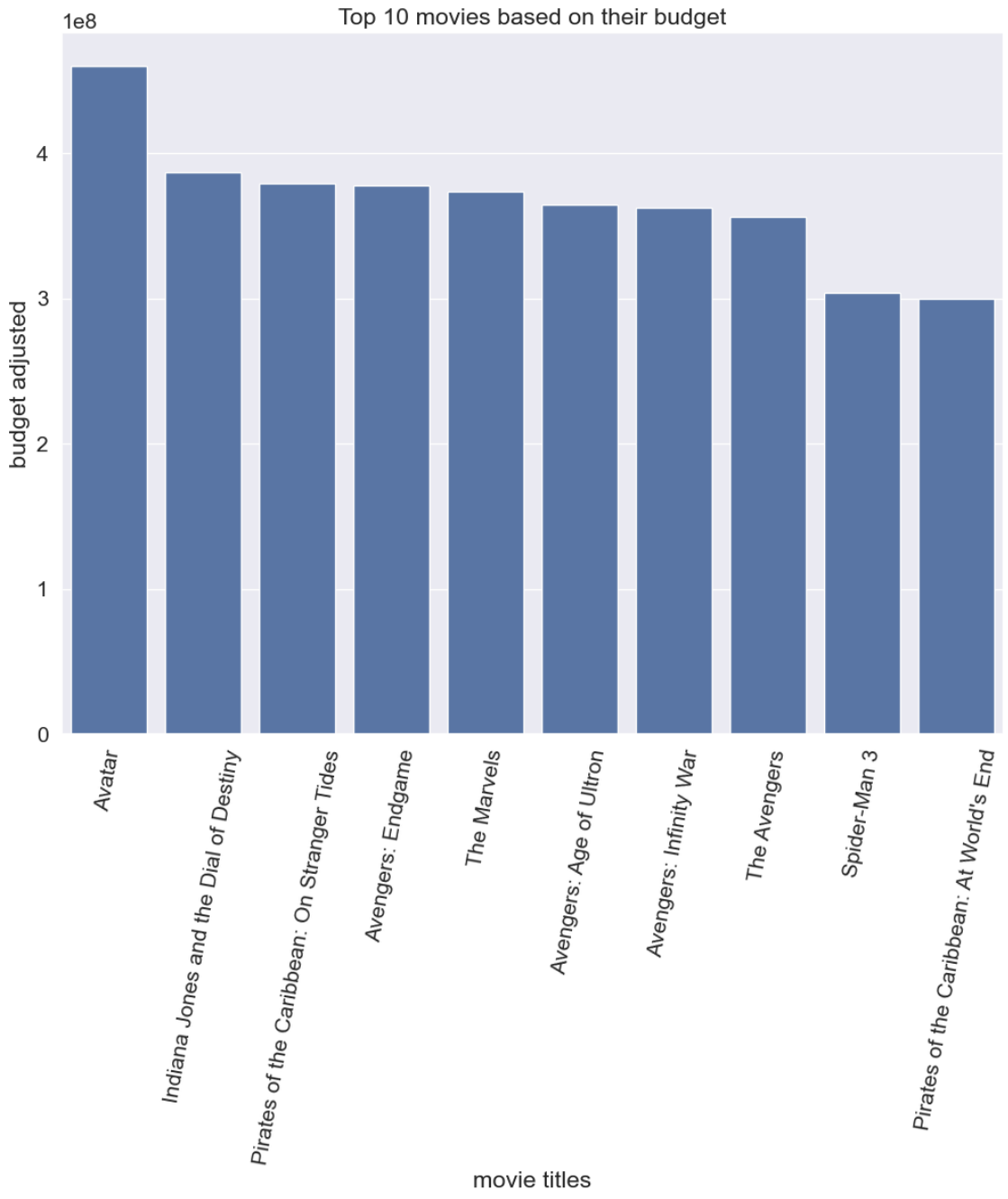
[8]:
```
sns.set(rc={'figure.figsize':(12,12)}, font_scale=1.3)
sns.set_palette("Set1", 20, .65)


ax = gerne_count_per_year_df.plot.bar(stacked=True);
ax.set(xlabel='years', ylabel='movies count', title = 'Stacked barplot showing␣
 ↪the trend of different movie genres')
plt.show()


#ax = gerne_count_per_year_df.plot.area(stacked=True);
```

```
#ax.set(xlabel='movie titles', ylabel='movies count', title = 'Stacked area␣
 ↪plot showing the trend of different movie genres')
#plt.show()
```



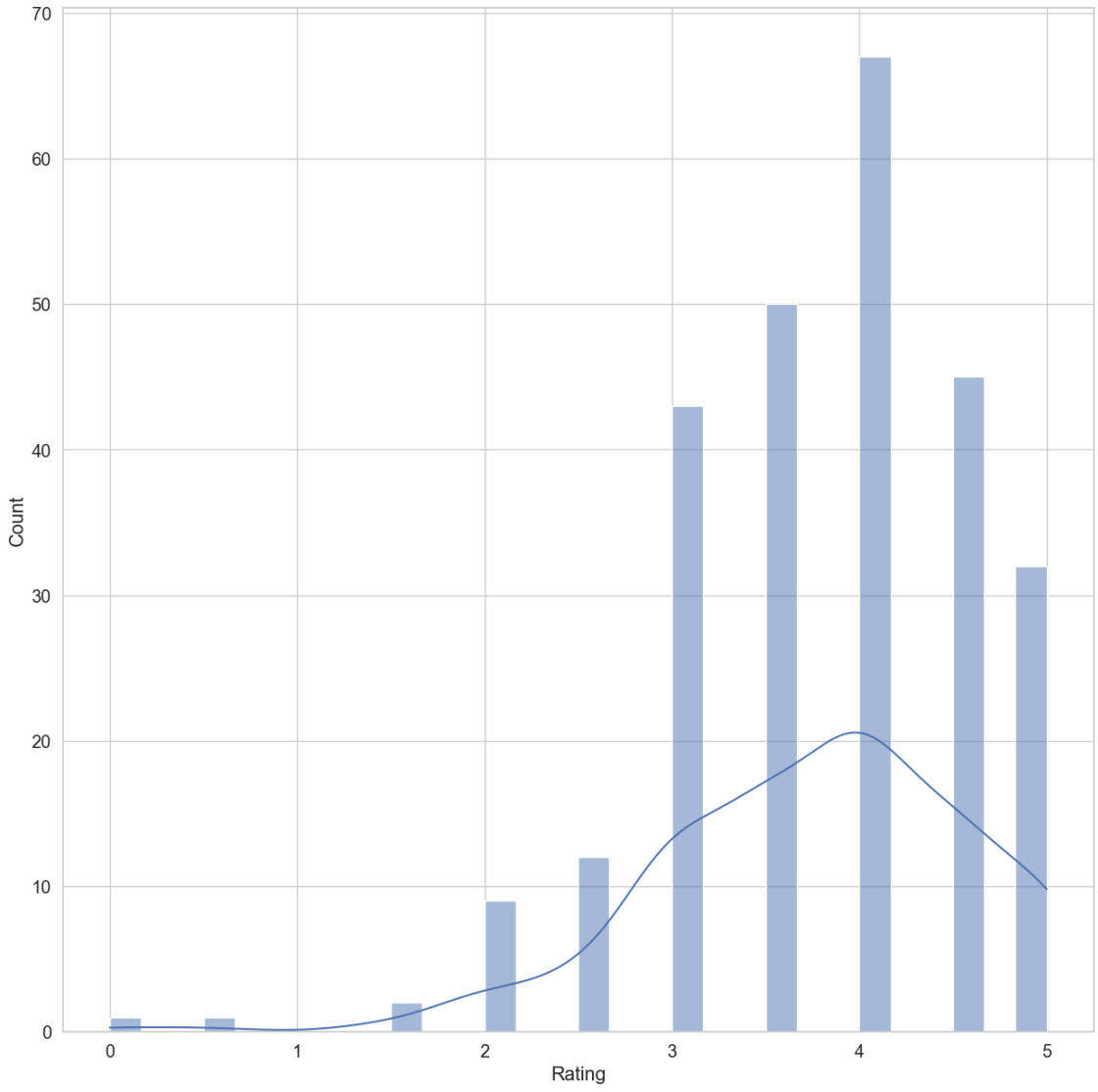
[13]:
```
#####
#Top 10 movie with the highest adjusted revenue
#####
movies_and_budget = data_frame[['Title', 'Budget']]

sns.set(rc={'figure.figsize':(12,9)}, font_scale=1.4)
```

```
top_movies = movies_and_budget.sort_values(by="Budget", ascending = False).
 ↪head(10)

ax = sns.barplot(x=top_movies["Title"], y=top_movies["Budget"])


#rotate x-axis' text
for item in ax.get_xticklabels():
    item.set_rotation(80)

ax.set(xlabel='movie titles', ylabel='budget adjusted', title = 'Top 10 movies␣
 ↪based on their budget')
plt.show()
```



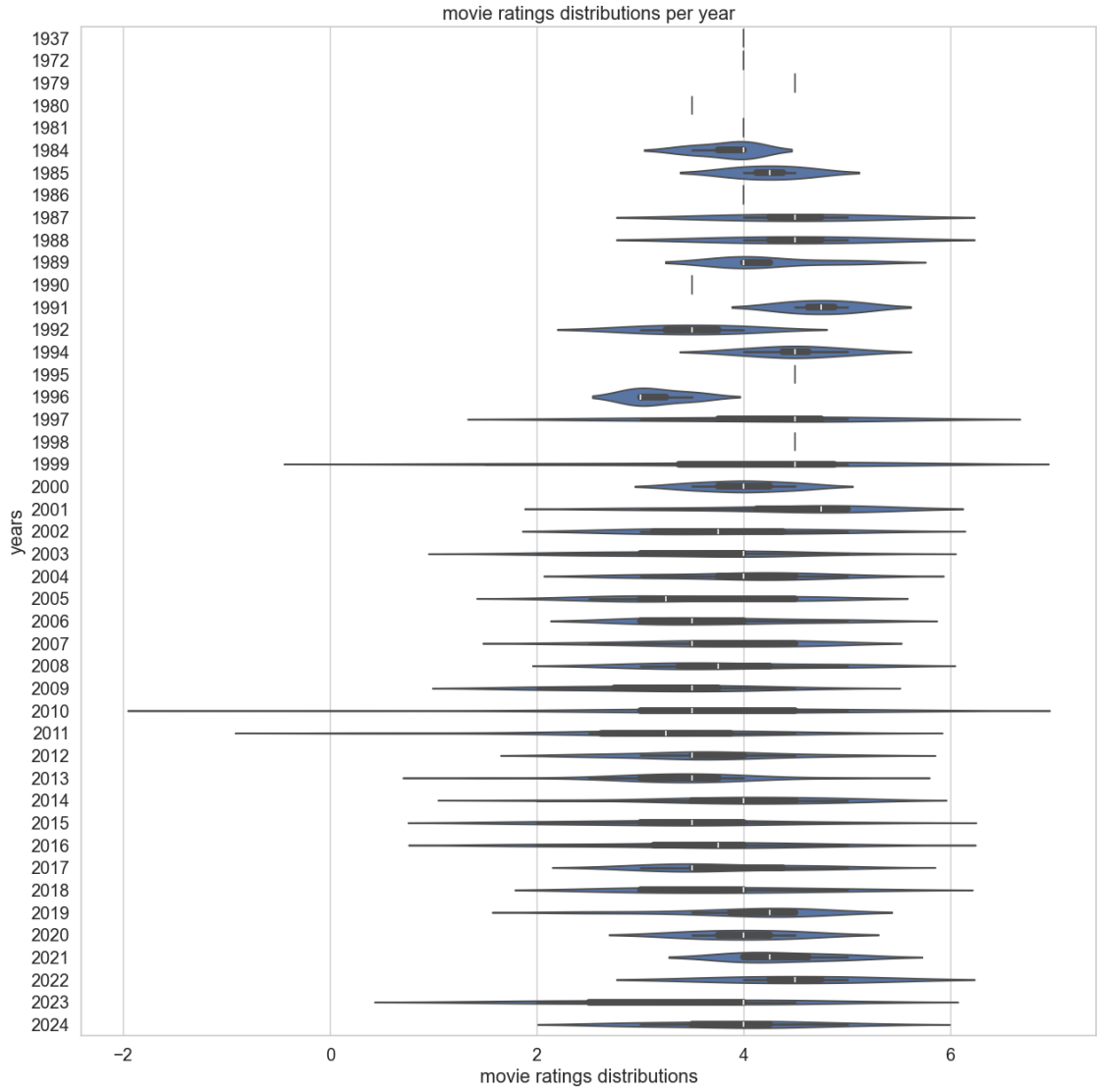
[19]:
```
#####
#movie ratings' distribution all over the years
#####

sns.set(rc={'figure.figsize': (15, 15)}, font_scale=1.3)

temp_df = data_frame[["Rating"]]
```

```
sns.set_style("whitegrid")

# Use histplot instead of distplot
ax = sns.histplot(temp_df["Rating"], bins=30, kde=True)

plt.show()
```



[20]:
```
#####
#movie ratings' distributions per year
#####

sns.set(rc={'figure.figsize':(15,15)}, font_scale=1.3)
```

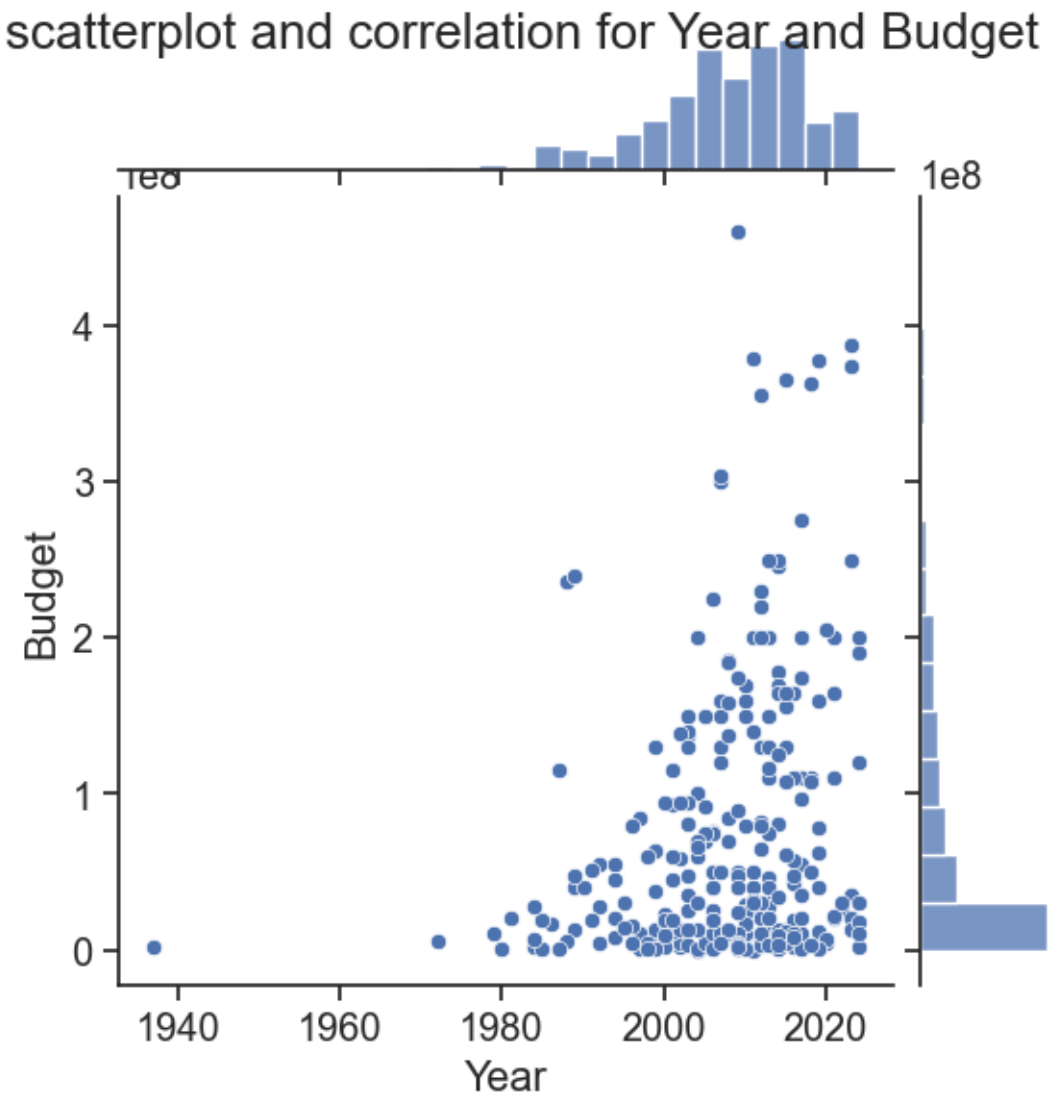
```
temp_df = data_frame[["Year", "Rating"]]


sns.set_style("whitegrid")
ax = sns.violinplot(x = temp_df.Rating, y = temp_df.Year, orient ="h")

ax.set(xlabel='movie ratings distributions', ylabel='years', title = 'movie␣
 ↪ratings distributions per year')
plt.show()
```

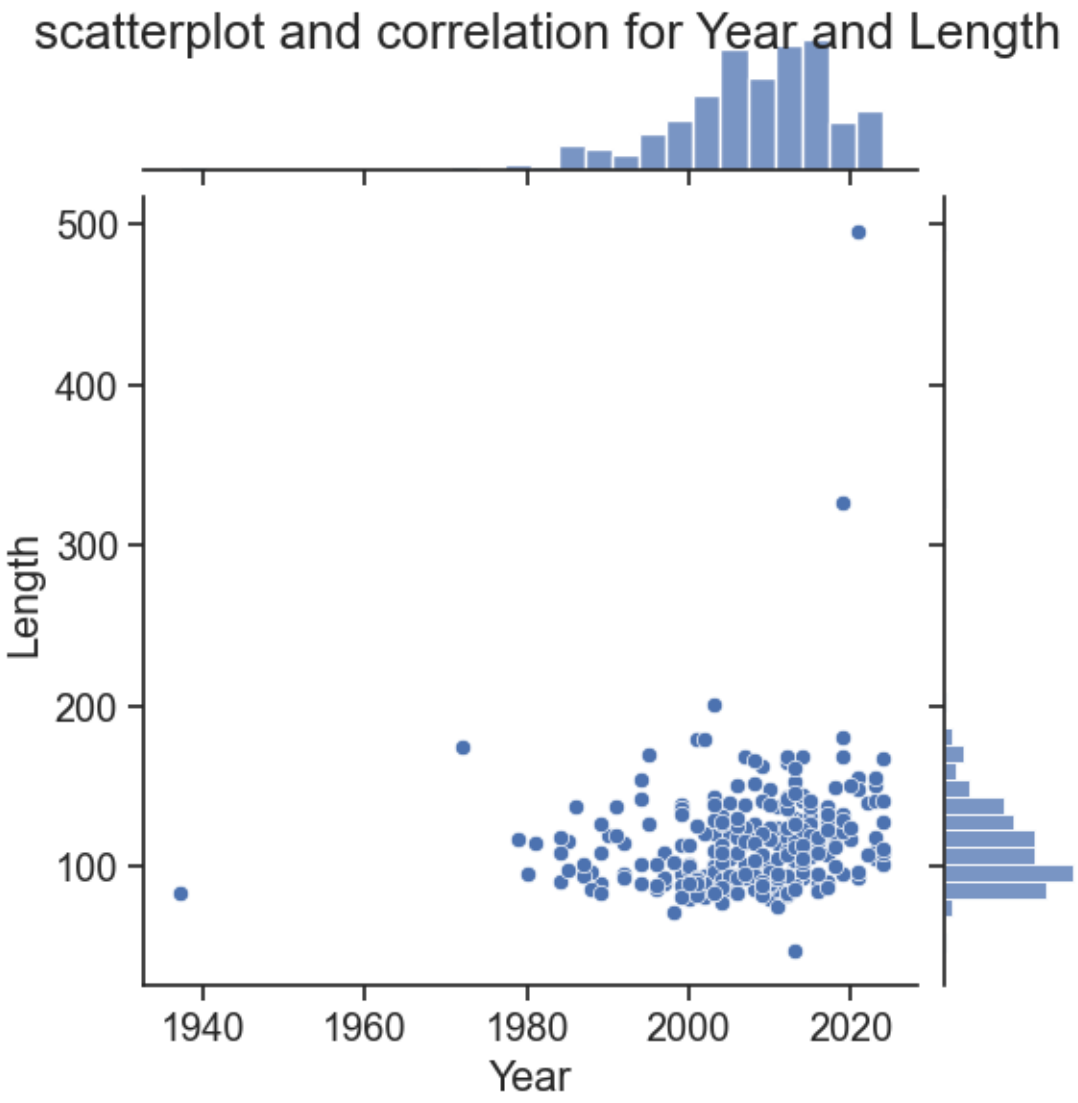


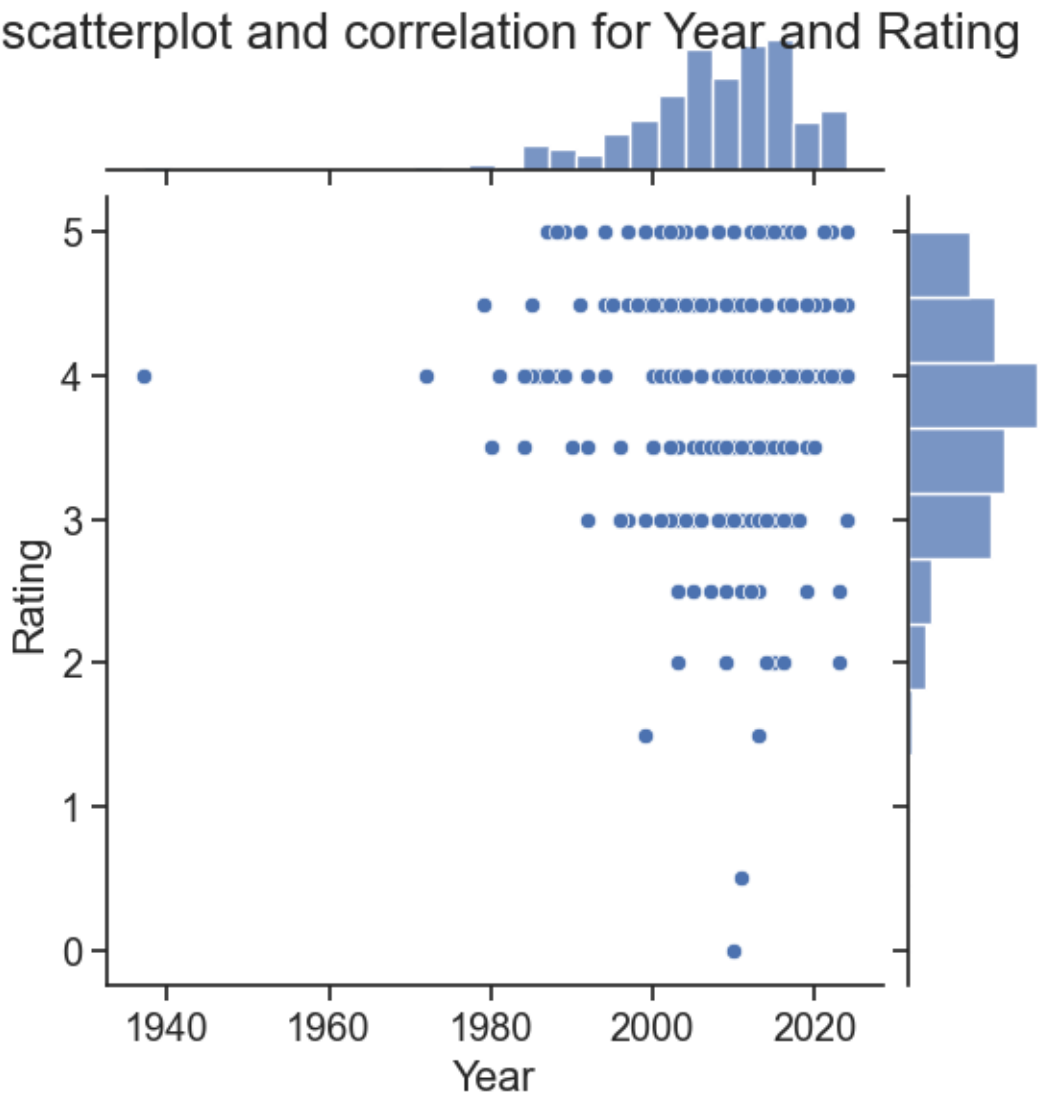
Korelacije

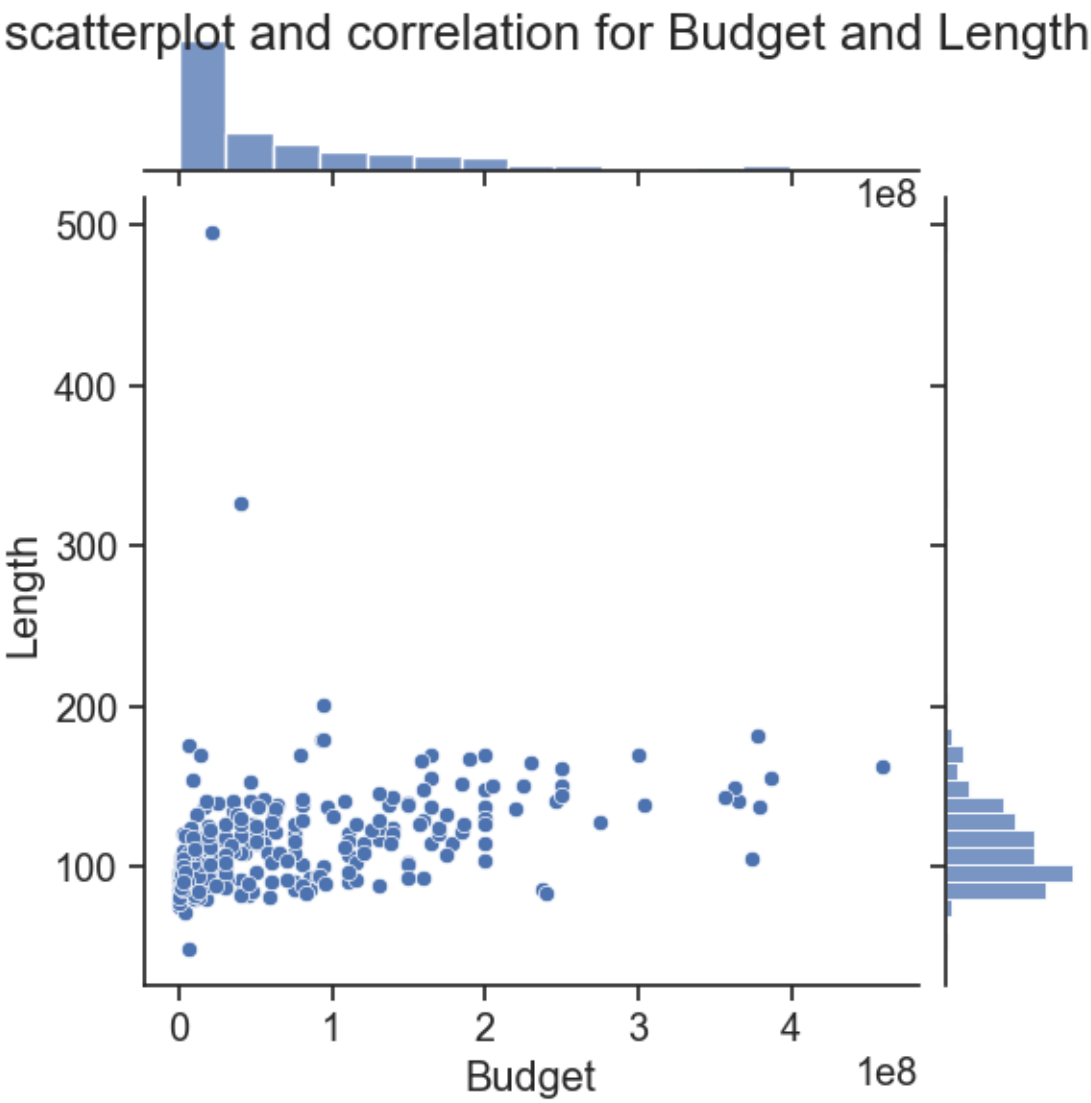
[23]:
```
#####
#correlation plots
#####

#get
aux_df = data_frame[['Budget', 'Year', 'Length', 'Rating']]

sns.set(rc={'figure.figsize':(15,15)}, font_scale=1.3, style="ticks")


f1 = sns.jointplot(x = "Year", y = "Budget", kind = "scatter", data = aux_df)
f1.fig.suptitle('scatterplot and correlation for Year and Budget')

f2 = sns.jointplot(x = "Year", y = "Length", kind = "scatter", data = aux_df)
f2.fig.suptitle('scatterplot and correlation for Year and Length')
f3 = sns.jointplot(x = "Year", y = "Rating", kind = "scatter", data = aux_df)
f3.fig.suptitle('scatterplot and correlation for Year and Rating')

f4 = sns.jointplot(x = "Budget", y = "Length", kind = "scatter", data = aux_df)
f4.fig.suptitle('scatterplot and correlation for Budget and Length')
f5 = sns.jointplot(x = "Budget", y = "Rating", kind = "scatter", data = aux_df)
f5.fig.suptitle('scatterplot and correlation for Budget and Rating')

f6 = sns.jointplot(x = "Length", y = "Rating", kind = "scatter", data = aux_df)
f6.fig.suptitle('scatterplot and correlation for Length and Rating')
```

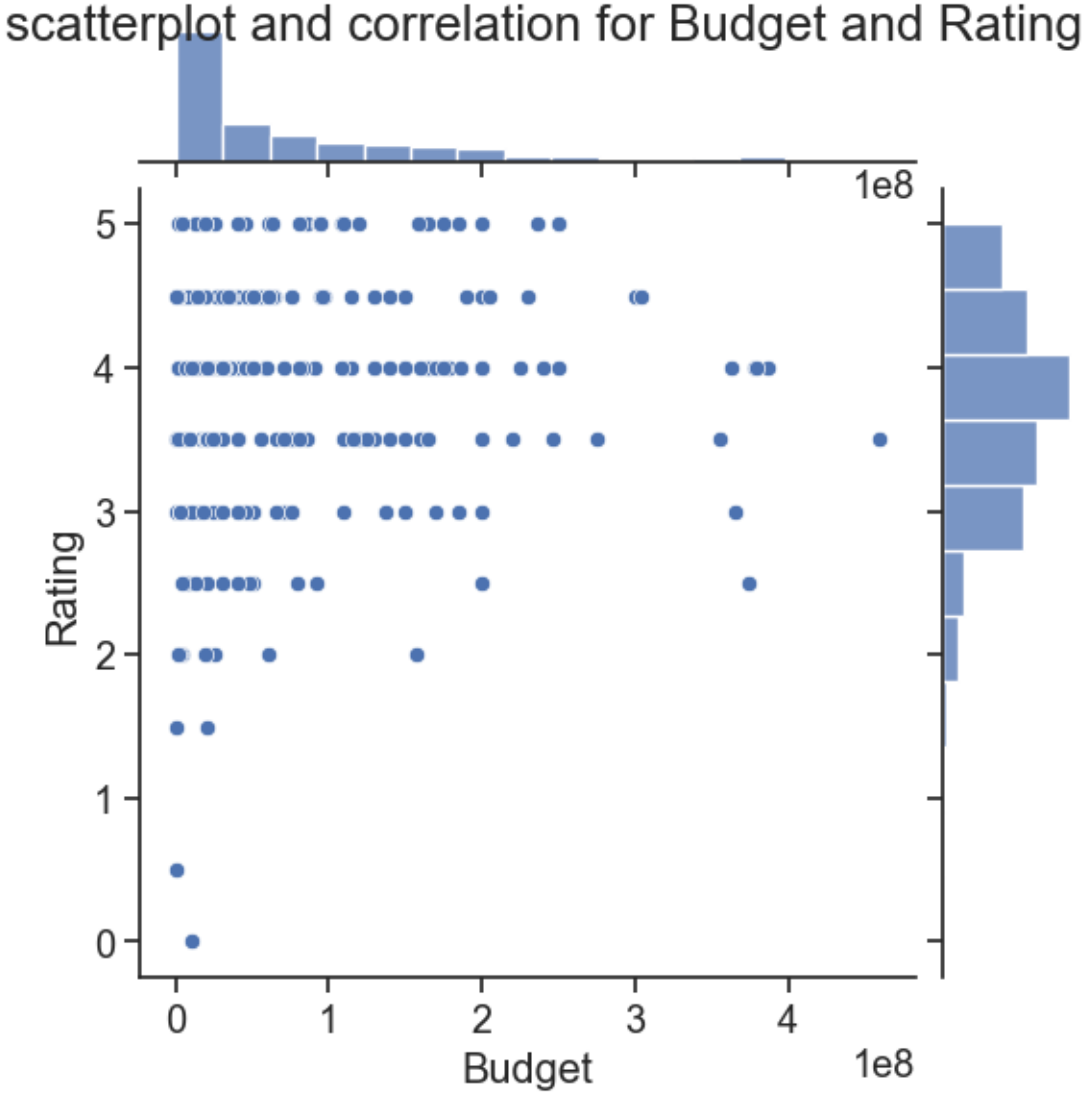
[23]:
```
Text(0.5, 0.98, 'scatterplot and correlation for Length and Rating')
```

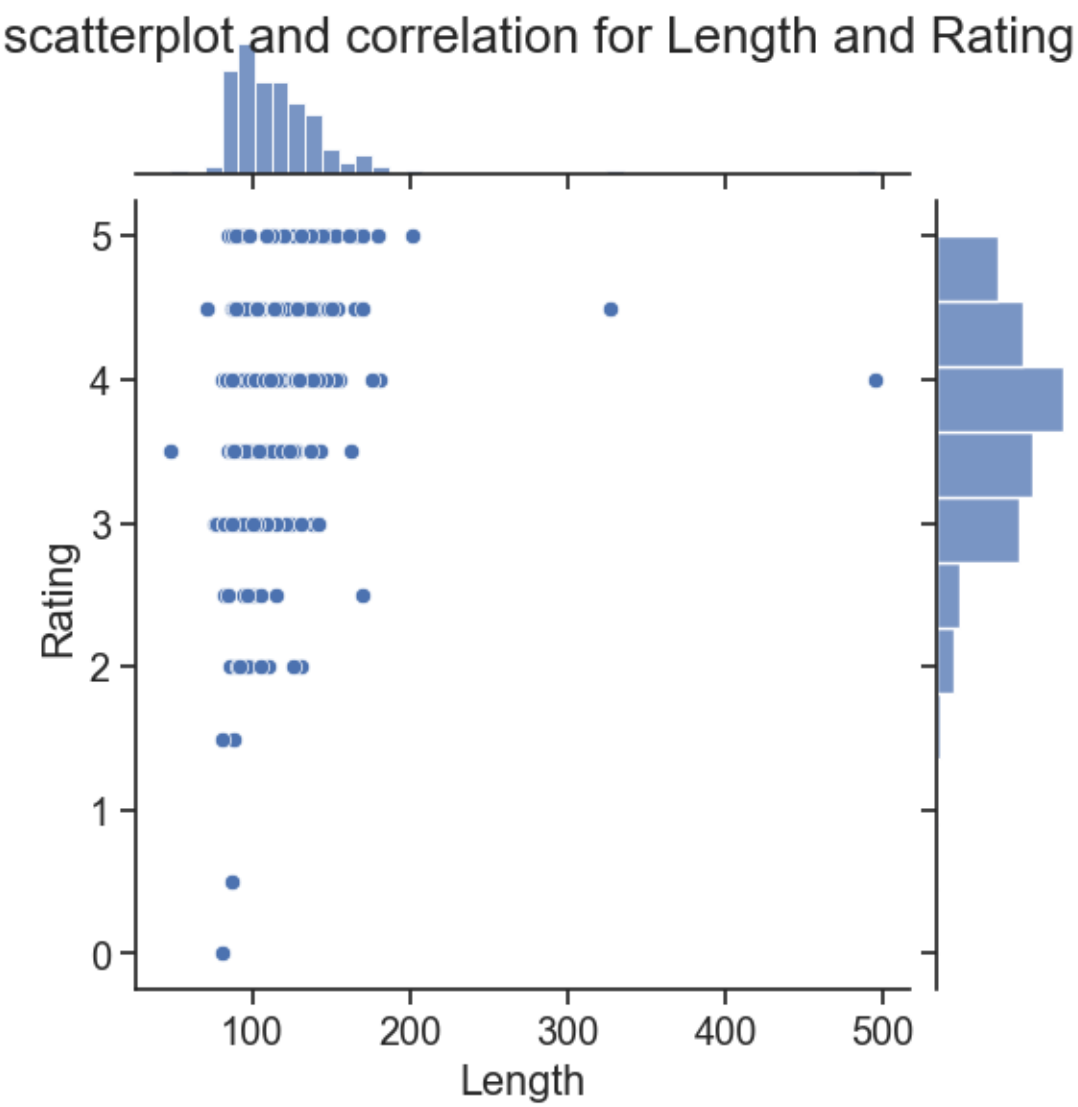

scatterplot and correlation for Year and Budget
1e8
1e8
4
3
2
1
0
Budget
1940
1960
1980
2000
2020
Year


scatterplot and correlation for Year and Length
500
400
300
200
100
Length
1940
1960
1980
2000
2020
Year

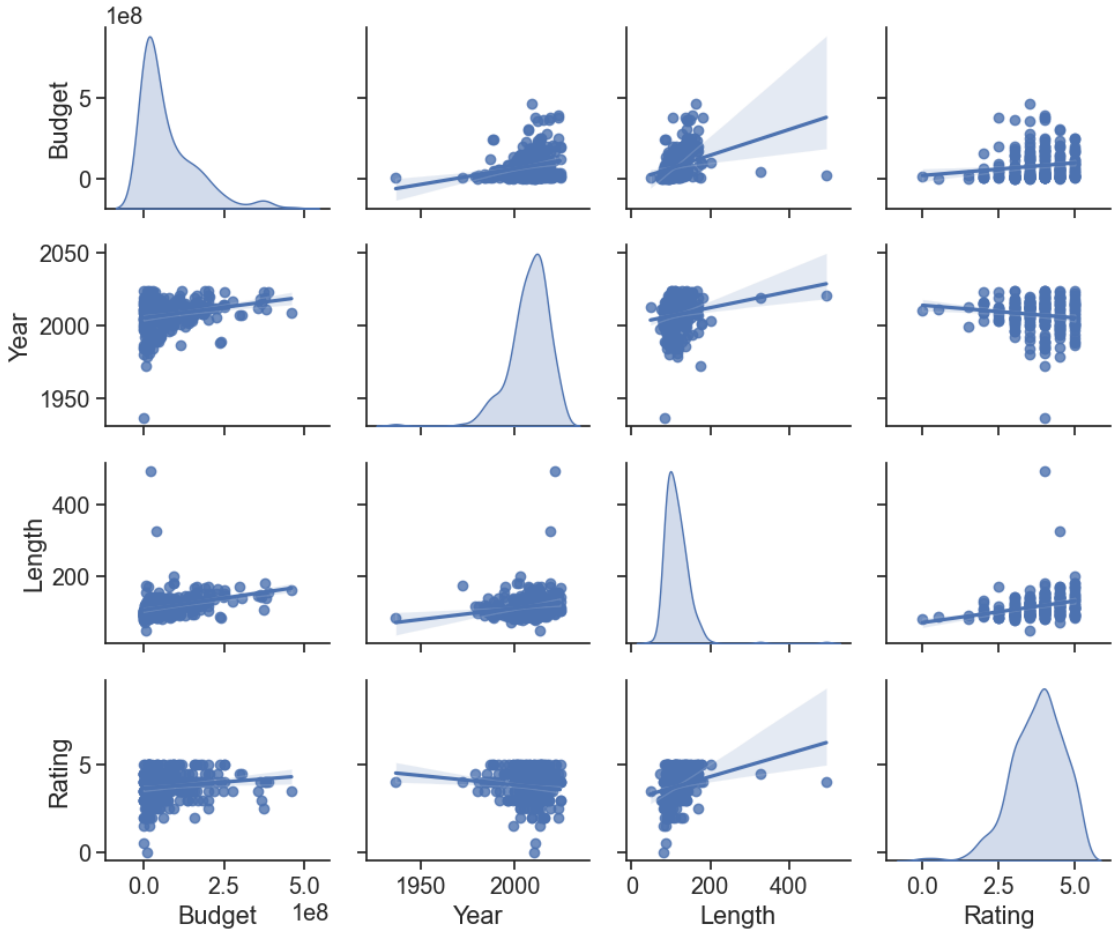

scatterplot and correlation for Year and Rating
Rating
5
4
3
2
1
0
1940
1960
1980
2000
2020
Year


scatterplot and correlation for Budget and Length
1e8
500
400
300
200
100
Length
0
1
2
3
4
Budget
1e8


scatterplot and correlation for Budget and Rating
1e8
5
4
3
2
1
0
Rating
0
1
2
3
4
Budget
1e8



[25]:
```
f1 = sns.pairplot(aux_df, kind="reg", diag_kind="kde",␣
↪diag_kws=dict(fill=True))  # Use fill=True instead of shade=True
f1.fig.suptitle('Scatterplots for Budget, Year, Length, and Rating\n')
f1.fig.tight_layout(rect=[0, 0.03, 1, 0.95])
```


Scatterplots for Budget, Year, Length, and Rating
1e8
Budget
5
0
2050
Year
2000
1950
400
Length
200
Rating
5
0
0.0
2.5
5.0
Budget
1e8
1950
2000
Year
0
200
400
Length
0.0
2.5
5.0
Rating

[ ]: