Seaborn_and_Linear_Regression

July 16, 2023

1 Introduction

Do higher film budgets lead to more box office revenue? Let's find out if there's a relationship using the movie budgets and financial performance data that I've scraped from the-numbers.com on **May 1st, 2018**.

2 Import Statements

[1]:
```
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)

import pandas as pd
import matplotlib.pyplot as plt

import seaborn as sns
from sklearn.linear_model import LinearRegression
```

3 Notebook Presentation

[2]:
```
pd.options.display.float_format = '{:,.2f}'.format

from pandas.plotting import register_matplotlib_converters
register_matplotlib_converters()
```

4 Read the Data

[3]:
```
data = pd.read_csv('cost_revenue_dirty.csv')
```

5 Explore and Clean the Data

[4]:
```
data.shape
```

[4]:
```
(5391, 6)
```

[5]:
```
data.sample(5)
```

[5]:
```
      Rank Release_Date                                Movie_Title  \
639   3334     6/3/1992                          The Lawnmower Man
591   3576     1/3/1991                           Haakon Haakonsen
524   2886     1/3/1989                           New York Stories
3039  1752   12/21/2007                            P.S. I Love You
4179  3399    6/22/2012  Seeking a Friend for the End of the World

     USD_Production_Budget USD_Worldwide_Gross USD_Domestic_Gross
639             $10,000,000         $32,100,816        $32,100,816
591              $8,500,000         $15,024,232        $15,024,232
524             $15,000,000         $10,763,469        $10,763,469
3039            $30,000,000        $155,769,678        $53,695,808
4179            $10,000,000         $11,766,959         $7,078,738
```

[6]:
```
data.tail()
```

[6]:
```
      Rank Release_Date                    Movie_Title USD_Production_Budget  \
5386  2950    10/8/2018                            Meg           $15,000,000
5387   126   12/18/2018                        Aquaman          $160,000,000
5388    96   12/31/2020                     Singularity          $175,000,000
5389  1119   12/31/2020           Hannibal the Conqueror           $50,000,000
5390  2517   12/31/2020  Story of Bonnie and Clyde, The           $20,000,000

     USD_Worldwide_Gross USD_Domestic_Gross
5386                  $0                 $0
5387                  $0                 $0
5388                  $0                 $0
5389                  $0                 $0
5390                  $0                 $0
```

[7]:
```
print(f'Any NaN values among the data? {data.isna().values.any()}')
```

```
Any NaN values among the data? False
```

[8]:
```
print(f'Any duplicates? {data.duplicated().values.any()}')

duplicated_rows = data[data.duplicated()]
print(f'Number of duplicates: {len(duplicated_rows)}')
```

```
Any duplicates? False
Number of duplicates: 0
```

[9]:
```
# Show NaN values and data types per column
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5391 entries, 0 to 5390
Data columns (total 6 columns):
 #   Column                 Non-Null Count  Dtype
```

```
---  ------                  --------------  -----
 0   Rank                    5391 non-null   int64
 1   Release_Date            5391 non-null   object
 2   Movie_Title             5391 non-null   object
 3   USD_Production_Budget   5391 non-null   object
 4   USD_Worldwide_Gross     5391 non-null   object
 5   USD_Domestic_Gross      5391 non-null   object
dtypes: int64(1), object(5)
memory usage: 252.8+ KB
```

5.0.1 Data Type Conversions

Challenge: Convert the `USD_Production_Budget`, `USD_Worldwide_Gross`, and `USD_Domestic_Gross` columns to a numeric format by removing `$` signs and `,`. Note that *domestic* in this context refers to the United States.

[10]:
```
chars_to_remove = [',', '$']
columns_to_clean = ['USD_Production_Budget',
                    'USD_Worldwide_Gross',
                    'USD_Domestic_Gross']

for col in columns_to_clean:
    for char in chars_to_remove:
        # Replace each character with an empty string
        data[col] = data[col].astype(str).str.replace(char, "")
    # Convert column to a numeric data type
    data[col] = pd.to_numeric(data[col])
```

[11]:
```
data.head()
```

[11]:

	Rank	Release_Date	Movie_Title	USD_Production_Budget	USD_Worldwide_Gross	USD_Domestic_Gross
0	5293	8/2/1915	The Birth of a Nation	110000	11000000	10000000
1	5140	5/9/1916	Intolerance	385907	0	0
2	5230	12/24/1916	20,000 Leagues Under the Sea	200000	8000000	8000000
3	5299	9/17/1920	Over the Hill to the Poorhouse	100000	3000000	3000000
4	5222	1/1/1925	The Big Parade	245000	22000000	11000000

Challenge: Convert the `Release_Date` column to a Pandas Datetime type.

[12]:
```
data.Release_Date = pd.to_datetime(data.Release_Date)
data.head()
```

[12]:
```
   Rank Release_Date                    Movie_Title  USD_Production_Budget  \
0  5293   1915-08-02          The Birth of a Nation                 110000
1  5140   1916-05-09                    Intolerance                 385907
2  5230   1916-12-24   20,000 Leagues Under the Sea                 200000
3  5299   1920-09-17  Over the Hill to the Poorhouse                100000
4  5222   1925-01-01                The Big Parade                  245000

   USD_Worldwide_Gross  USD_Domestic_Gross
0             11000000            10000000
1                    0                   0
2              8000000             8000000
3              3000000             3000000
4             22000000            11000000
```

[13]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5391 entries, 0 to 5390
Data columns (total 6 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Rank                   5391 non-null   int64
 1   Release_Date           5391 non-null   datetime64[ns]
 2   Movie_Title            5391 non-null   object
 3   USD_Production_Budget  5391 non-null   int64
 4   USD_Worldwide_Gross    5391 non-null   int64
 5   USD_Domestic_Gross     5391 non-null   int64
dtypes: datetime64[ns](1), int64(4), object(1)
memory usage: 252.8+ KB
```

5.0.2 Descriptive Statistics

Challenge:

1. What is the average production budget of the films in the data set?
2. What is the average worldwide gross revenue of films?
3. What were the minimums for worldwide and domestic revenue?
4. Are the bottom 25% of films actually profitable or do they lose money?
5. What are the highest production budget and highest worldwide gross revenue of any film?
6. How much revenue did the lowest and highest budget films make?

[14]:
```
data.describe()
```

[14]:
```
          Rank  USD_Production_Budget  USD_Worldwide_Gross  USD_Domestic_Gross
count 5,391.00              5,391.00             5,391.00            5,391.00
mean  2,696.00         31,113,737.58        88,855,421.96       41,235,519.44
std   1,556.39         40,523,796.88       168,457,757.00       66,029,346.27
min       1.00              1,100.00                0.00                0.00
25%   1,348.50          5,000,000.00        3,865,206.00        1,330,901.50
```

```
50%    2,696.00          17,000,000.00        27,450,453.00        17,192,205.00
75%    4,043.50          40,000,000.00        96,454,455.00        52,343,687.00
max    5,391.00         425,000,000.00     2,783,918,982.00       936,662,225.00
```

[15]:
```
data[data.USD_Production_Budget == 1100.00]
```

[15]:
```
      Rank Release_Date        Movie_Title  USD_Production_Budget  \
2427  5391   2005-05-08  My Date With Drew                   1100

      USD_Worldwide_Gross  USD_Domestic_Gross
2427               181041              181041
```

[16]:
```
data[data.USD_Production_Budget == 425000000.00]
```

[16]:
```
      Rank Release_Date Movie_Title  USD_Production_Budget  \
3529     1   2009-12-18      Avatar              425000000

      USD_Worldwide_Gross  USD_Domestic_Gross
3529           2783918982           760507625
```

6 Investigating the Zero Revenue Films

Challenge How many films grossed $0 domestically (i.e., in the United States)? What were the highest budget films that grossed nothing?

[17]:
```
zero_domestic = data[data.USD_Domestic_Gross == 0]
print(f'Number of films that grossed $0 domestically {len(zero_domestic)}')
zero_domestic.sort_values('USD_Production_Budget', ascending=False)
```

```
Number of films that grossed $0 domestically 512
```

[17]:
```
      Rank Release_Date                          Movie_Title  \
5388    96   2020-12-31                          Singularity
5387   126   2018-12-18                              Aquaman
5384   321   2018-09-03                    A Wrinkle in Time
5385   366   2018-10-08                       Amusement Park
5090   556   2015-12-31  Don Gato, el inicio de la pandilla
…      …          …                                    …
4787  5371   2014-12-31                Stories of Our Lives
3056  5374   2007-12-31                          Tin Can Man
4907  5381   2015-05-19                     Family Motocross
5006  5389   2015-09-29              Signed Sealed Delivered
5007  5390   2015-09-29                 A Plague So Pleasant

      USD_Production_Budget  USD_Worldwide_Gross  USD_Domestic_Gross
5388              175000000                    0                   0
5387              160000000                    0                   0
5384              103000000                    0                   0
```

```
5385              100000000                    0                    0
5090               80000000              4547660                    0
…                         …                    …                    …
4787                  15000                    0                    0
3056                  12000                    0                    0
4907                  10000                    0                    0
5006                   5000                    0                    0
5007                   1400                    0                    0

[512 rows x 6 columns]
```

Challenge: How many films grossed $0 worldwide? What are the highest budget films that had no revenue internationally?

[18]:
```
zero_worldwide = data[data.USD_Worldwide_Gross == 0]
print(f'Number of films that grossed $0 worldwide {len(zero_worldwide)}')
zero_worldwide.sort_values('USD_Production_Budget', ascending=False)
```

```
Number of films that grossed $0 worldwide 357
```

[18]:
```
      Rank Release_Date             Movie_Title  USD_Production_Budget  \
5388    96   2020-12-31              Singularity              175000000
5387   126   2018-12-18                  Aquaman              160000000
5384   321   2018-09-03       A Wrinkle in Time              103000000
5385   366   2018-10-08          Amusement Park              100000000
5058   880   2015-11-12        The Ridiculous 6               60000000
…      …          …                       …                      …
4787  5371   2014-12-31    Stories of Our Lives                  15000
3056  5374   2007-12-31             Tin Can Man                  12000
4907  5381   2015-05-19        Family Motocross                  10000
5006  5389   2015-09-29  Signed Sealed Delivered                   5000
5007  5390   2015-09-29    A Plague So Pleasant                   1400

      USD_Worldwide_Gross  USD_Domestic_Gross
5388                    0                   0
5387                    0                   0
5384                    0                   0
5385                    0                   0
5058                    0                   0
…                     …                   …
4787                    0                   0
3056                    0                   0
4907                    0                   0
5006                    0                   0
5007                    0                   0

[357 rows x 6 columns]
```

6.0.1 International releases

[19]:
```
international_releases = data.loc[(data.USD_Domestic_Gross == 0) &
                                  (data.USD_Worldwide_Gross != 0)]
print(f'Number of international releases: {len(international_releases)}')
international_releases.head()
```

```
Number of international releases: 155
```

[19]:

	Rank	Release_Date	Movie_Title	USD_Production_Budget	USD_Worldwide_Gross	USD_Domestic_Gross
71	4310	1956-02-16	Carousel	3380000	3220	0
1579	5087	2001-02-11	Everything Put Together	500000	7890	0
1744	3695	2001-12-31	The Hole	7500000	10834406	0
2155	4236	2003-12-31	Nothing	4000000	63180	0
2203	2513	2004-03-31	The Touch	20000000	5918742	0

6.0.2 Unreleased Films

Challenge: * Identify which films were not released yet as of the time of data collection (May 1st, 2018). * How many films are included in the dataset that have not yet had a chance to be screened in the box office? * Create another DataFrame called data_clean that does not include these films.

[20]:
```
# Date of Data Collection
scrape_date = pd.Timestamp('2018-5-1')
```

[21]:
```
future_releases = data[data.Release_Date >= scrape_date]
print(f'Number of unreleased movies: {len(future_releases)}')
future_releases
```

```
Number of unreleased movies: 7
```

[21]:

	Rank	Release_Date	Movie_Title	USD_Production_Budget	USD_Worldwide_Gross	USD_Domestic_Gross
5384	321	2018-09-03	A Wrinkle in Time			
5385	366	2018-10-08	Amusement Park			
5386	2950	2018-10-08	Meg			
5387	126	2018-12-18	Aquaman			
5388	96	2020-12-31	Singularity			
5389	1119	2020-12-31	Hannibal the Conqueror			
5390	2517	2020-12-31	Story of Bonnie and Clyde, The			

5384	103000000	0	0
5385	100000000	0	0
5386	15000000	0	0
5387	160000000	0	0
5388	175000000	0	0
5389	50000000	0	0
5390	20000000	0	0

[22]:
```
# exclude future releases
data_clean = data.drop(future_releases.index)
```

[23]:
```
# difference is 7 rows
data.shape[0] - data_clean.shape[0]
```

[23]: 7

6.0.3 Films that Lost Money

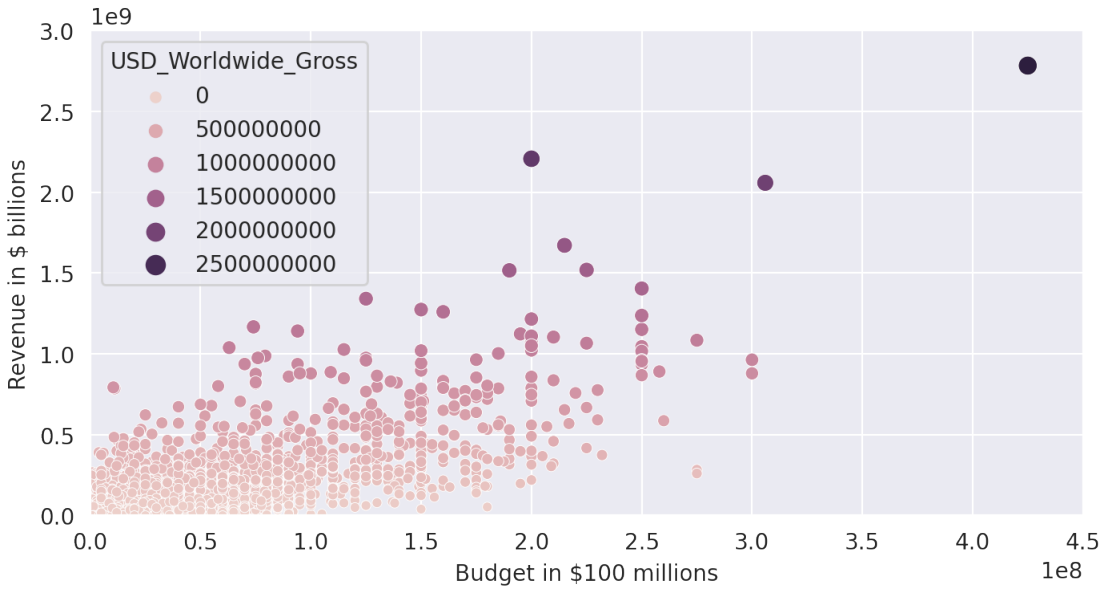
Challenge: What is the percentage of films where the production costs exceeded the worldwide gross revenue?

[24]:
```
money_losing = data_clean.query('USD_Production_Budget > USD_Worldwide_Gross')
money_losing.shape[0]/data_clean.shape[0]
```

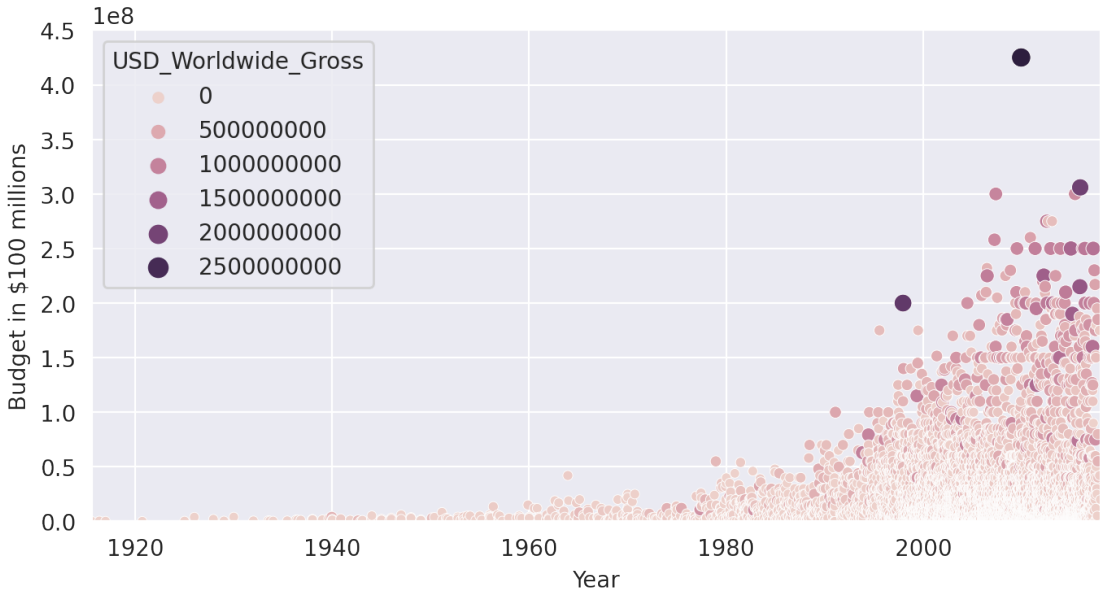
[24]: 0.37277117384843983

7 Seaborn for Data Viz: Bubble Charts

[25]:
```
plt.figure(figsize=(8,4), dpi=200)

# set styling on a single chart
with sns.axes_style('darkgrid'):
  ax = sns.scatterplot(data=data_clean,
                       x='USD_Production_Budget',
                       y='USD_Worldwide_Gross',
                       hue='USD_Worldwide_Gross',
                       size='USD_Worldwide_Gross')

  ax.set(ylim=(0, 3000000000),
         xlim=(0, 450000000),
         ylabel='Revenue in $ billions',
         xlabel='Budget in $100 millions')
```



[26]:
```
plt.figure(figsize=(8,4), dpi=200)

with sns.axes_style("darkgrid"):
    ax = sns.scatterplot(data=data_clean,
                         x='Release_Date',
                         y='USD_Production_Budget',
                         hue='USD_Worldwide_Gross',
                         size='USD_Worldwide_Gross',)

    ax.set(ylim=(0, 450000000),
           xlim=(data_clean.Release_Date.min(), data_clean.Release_Date.max()),
           xlabel='Year',
           ylabel='Budget in $100 millions')
```



8 Converting Years to Decades Trick

Challenge: Create a column in `data_clean` that has the decade of the release.

[27]:
```
dt_index = pd.DatetimeIndex(data_clean.Release_Date)
years = dt_index.year
```

[28]:
```
# How to convert the year 1999 to the 90s decade
1999//10
```

[28]: 199

[29]:
```
199*10
```

[29]: 1990

[30]:
```
decades = years//10*10
data_clean['Decade'] = decades
```

8.0.1 Separate the "old" (before 1969) and "New" (1970s onwards) Films

Challenge: Create two new DataFrames: `old_films` and `new_films` * `old_films` should include all the films before 1969 (up to and including 1969) * `new_films` should include all the films from 1970 onwards * How many films were released prior to 1970? * What was the most expensive film made prior to 1970?

[31]:
```
old_films = data_clean[data_clean.Decade <= 1960]
new_films = data_clean[data_clean.Decade > 1960]
```

[32]:
```
old_films.describe()
```

[32]:

	Rank	USD_Production_Budget	USD_Worldwide_Gross
count	153.00	153.00	153.00
mean	4,274.77	4,611,297.65	30,419,634.38
std	742.14	5,713,648.85	54,931,828.93
min	1,253.00	100,000.00	0.00
25%	3,973.00	1,250,000.00	5,273,000.00
50%	4,434.00	2,900,000.00	10,000,000.00
75%	4,785.00	5,000,000.00	33,208,099.00
max	5,299.00	42,000,000.00	390,525,192.00

	USD_Domestic_Gross	Decade
count	153.00	153.00
mean	22,389,473.87	1,949.15
std	32,641,752.41	12.72
min	0.00	1,910.00
25%	5,000,000.00	1,940.00
50%	10,000,000.00	1,950.00
75%	28,350,000.00	1,960.00
max	198,680,470.00	1,960.00

[33]:
```
old_films.sort_values('USD_Production_Budget', ascending=False).head()
```

[33]:

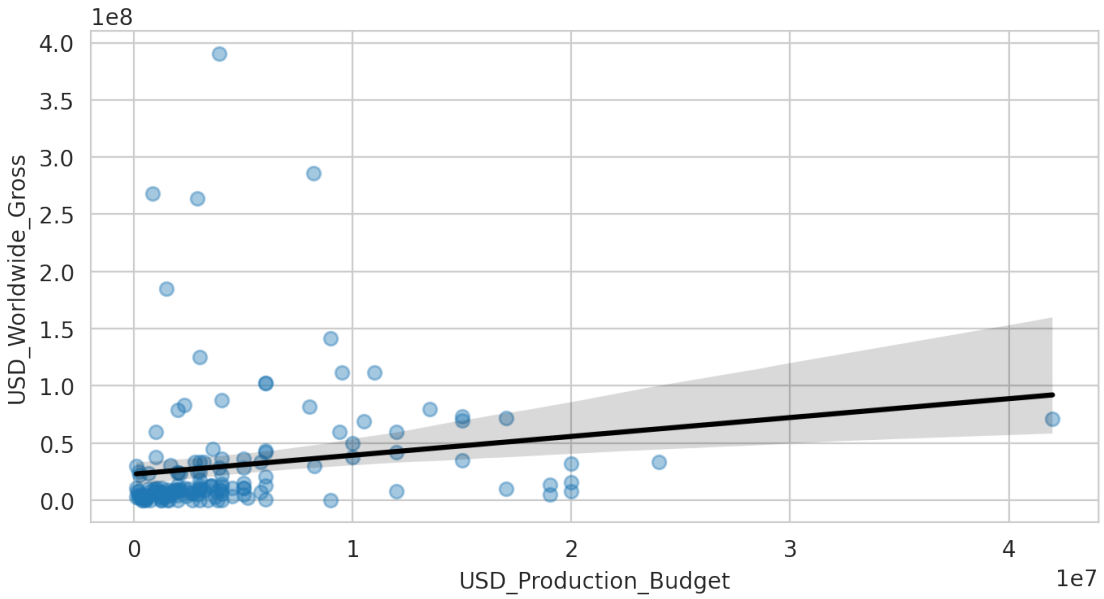
	Rank	Release_Date	Movie_Title	USD_Production_Budget
109	1253	1963-12-06	Cleopatra	42000000
150	2175	1969-12-16	Hello, Dolly	24000000
143	2465	1969-01-01	Sweet Charity	20000000
118	2425	1965-02-15	The Greatest Story Ever Told	20000000
148	2375	1969-10-15	Paint Your Wagon	20000000

	USD_Worldwide_Gross	USD_Domestic_Gross	Decade
109	71000000	57000000	1960
150	33208099	33208099	1960
143	8000000	8000000	1960
118	15473333	15473333	1960
148	31678778	31678778	1960

9 Seaborn Regression Plots

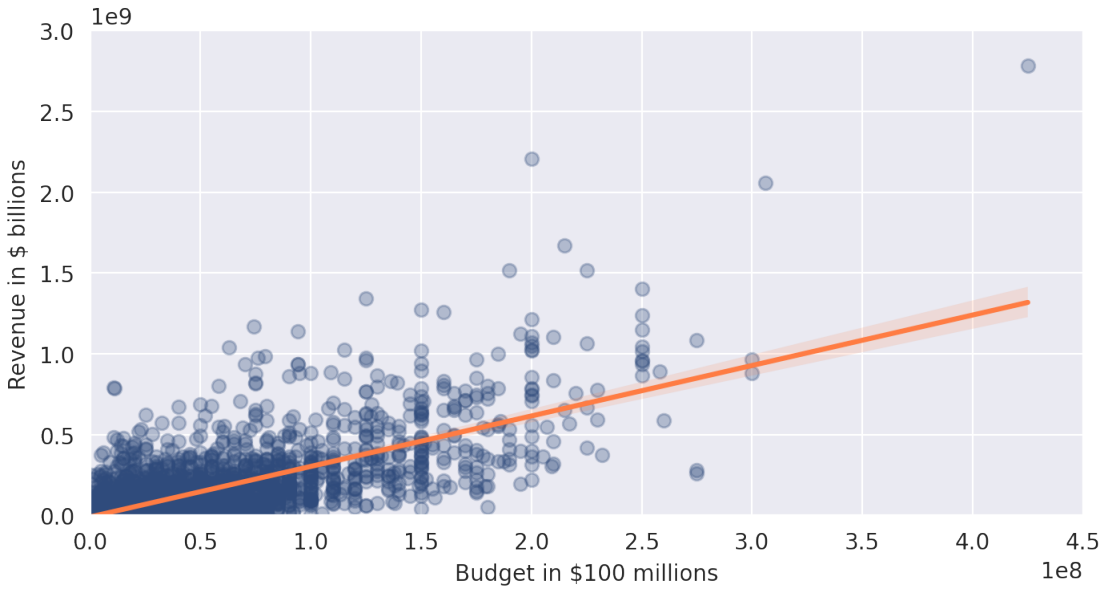
[34]:
```
plt.figure(figsize=(8,4), dpi=200)
with sns.axes_style("whitegrid"):
  sns.regplot(data=old_films,
              x='USD_Production_Budget',
```

```
            y='USD_Worldwide_Gross',
            scatter_kws = {'alpha': 0.4},
            line_kws = {'color': 'black'})
```



Challenge: Use Seaborn's `.regplot()` to show the scatter plot and linear regression line against the `new_films`.

[35]:
```
plt.figure(figsize=(8,4), dpi=200)
with sns.axes_style('darkgrid'):
  ax = sns.regplot(data=new_films,
                   x='USD_Production_Budget',
                   y='USD_Worldwide_Gross',
                   color='#2f4b7c',
                   scatter_kws = {'alpha': 0.3},
                   line_kws = {'color': '#ff7c43'})

  ax.set(ylim=(0, 3000000000),
         xlim=(0, 450000000),
         ylabel='Revenue in $ billions',
         xlabel='Budget in $100 millions')
```



10 Run our own regression with scikit-learn

Our Linear Model:

$$REV\hat{E}NUE = \theta_0 + \theta_1 BUDGET$$

[36]:
```
# Create regression object
regression = LinearRegression()
```

[37]:
```
# Explanatory Variable(s) or Feature(s)
X = pd.DataFrame(new_films, columns=['USD_Production_Budget'])

# Response Variable or Target
y = pd.DataFrame(new_films, columns=['USD_Worldwide_Gross'])
```

[38]:
```
# Find the best-fit line
regression.fit(X, y)
```

[38]:
```
LinearRegression()
```

[39]:
```
# Theta zero
regression.intercept_
```

[39]:
```
array([-8650768.00661024])
```

[40]:
```
# Theta one
regression.coef_
```

[40]:
```
array([[3.12259592]])
```

[41]:
```
# R-squared
regression.score(X, y)
```

[41]:
```
0.5577032617720403
```

Challenge: Run a linear regression for the `old_films`. Calculate the intercept, slope and r-squared. How much of the variance in movie revenue does the linear model explain in this case?

[42]:
```
X = pd.DataFrame(old_films, columns=['USD_Production_Budget'])
y = pd.DataFrame(old_films, columns=['USD_Worldwide_Gross'])
regression.fit(X, y)
print(f'The slope coefficient is: {regression.coef_[0]}')
print(f'The intercept is: {regression.intercept_[0]}')
print(f'The r-squared is: {regression.score(X, y)}')
```

```
The slope coefficient is: [1.64771314]
The intercept is: 22821538.635080382
The r-squared is: 0.02937258620576877
```

Only 3% this makes sense considering how poorly our data points aligned with our line earlier.

11 Let's use our model to make a prediction

We've just estimated the slope and the y-intercept! Remember that our linear model has the following form:

$$\hat{REVENUE} = \theta_0 + \theta_1 BUDGET$$

Challenge: How much global revenue does our model estimate for a film with a budget of $350 million?

[43]:
```
22821538 + 1.64771314 * 350000000
```

[43]:
```
599521137.0
```

[44]:
```
budget = 350000000
revenue_estimate = regression.intercept_[0] + regression.coef_[0,0]*budget
revenue_estimate = round(revenue_estimate, -6)
print(f'The estimated revenue for a $350 film is around ${revenue_estimate:.10}.
 ↪')
```

```
The estimated revenue for a $350 film is around $600000000.0.
```

[ ]:
```

```