ndgigliotti / dsc-phase-1-project

forked from learn-co-curriculum/dsc-phase-1-project

☆ 0 stars   61 forks

☆ Star   Watch ▾

<> Code   Pull requests   Actions   Projects   Wiki   Security   Insights

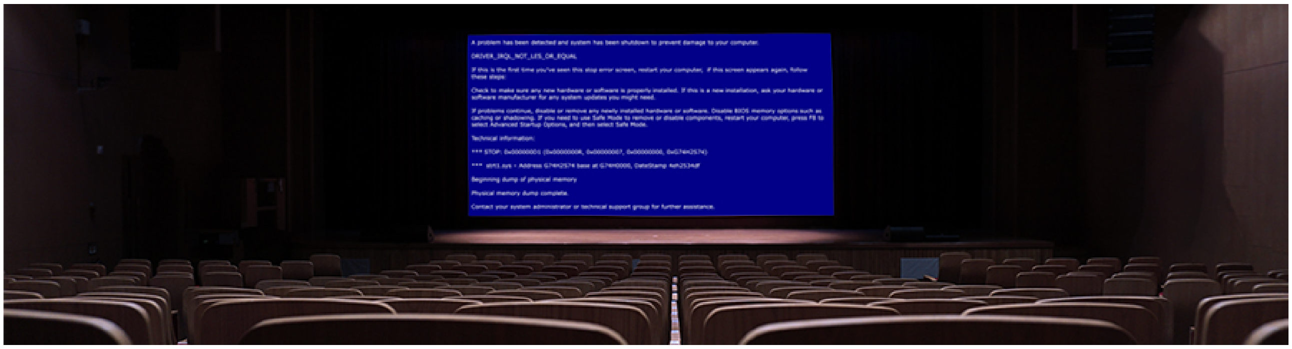
master ▾   ···

This branch is 53 commits ahead of learn-co-curriculum:master.   Pull request   Compare

ndgigliotti add README …   39 minutes ago   60

View code

README.md



Movie Genre Profitability for Microsoft

Author: Nicholas Gigliotti

Overview

I conduct an analysis of the profitability of different movie genres in relation to production budget for Microsoft. Microsoft wants to enter the movie business and develop original content. They will have to decide which genres they wish to invest in early on, since different genres have different production requirements. I conclude that Microsoft should invest in horror for low-budget productions and animation for high-budget productions. Horror has the strongest correlation with return on investment (ROI) of any genre, overall. I further conclude that Microsoft should stay away from drama, action, and crime movies because these are negatively correlated with ROI.

Business Problem

Microsoft has decided to enter the movie business and create original material. They want to know what kinds of movies are currently profitable, and they want concrete, actionable, insights.

In my analysis, I attempt to answer the following questions for Microsoft:

1. What genres have the strongest correlation with return on investment?
2. How does budget affect these correlations?
3. Are high or low-budget films more profitable?

Why Genre?

Different film genres have different markets, and need to be created by different groups of artists. Choosing which genres to invest in is one of the most fundamental early decisions Microsoft will have to make.

Data Understanding

I use data from two sources in my analysis: The Numbers and the Internet Movie Database (IMDb). IMDb is an expansive and easily accessible source of movie data which, most importantly, includes genre labels for thousands of films. IMDb lacks financial data, however, so I am forced to rely on The Numbers.

```python
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import ticker
import seaborn as sns

import utils
import cleaning
import plotting

%matplotlib inline
sns.set(font_scale=1.25)
pd.options.display.float_format = '{:,.2f}'.format
```

The Numbers

My financial data comes from a website called "The Numbers" which has a healthy collection of production budget and revenue data. The Numbers is owned by Nash Information Services, a movie industry research and consulting firm. The most important columns for my analysis are `production_budget` , `domestic_gross` , and `worldwide_gross` . I use these columns later to calculate profit and return on investment (ROI).

The table includes a little under 6,000 observations.

```
tn = pd.read_csv(os.path.join('zippedData', 'tn.movie_budgets.csv.gz'),
                 parse_dates=['release_date'])
tn
```

<style scoped> .dataframe tbody tr th:only-of-type { vertical-align: middle; }

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

</style>

	id	release_date	movie	production_budget	domestic_gross
0	1	2009-12-18	Avatar	$425,000,000	$760,507,625
1	2	2011-05-20	Pirates of the Caribbean: On Stranger Tides	$410,600,000	$241,063,875

	id	release_date	movie	production_budget	domestic_gross
2	3	2019-06-07	Dark Phoenix	$350,000,000	$42,762,350
3	4	2015-05-01	Avengers: Age of Ultron	$330,600,000	$459,005,868
4	5	2017-12-15	Star Wars Ep. VIII: The Last Jedi	$317,000,000	$620,181,382
...	...	...	...	...	...
5777	78	2018-12-31	Red 11	$7,000	$0
5778	79	1999-04-02	Following	$6,000	$48,482
5779	80	2005-07-13	Return to the Land of Wonders	$5,000	$1,338
5780	81	2015-09-29	A Plague So Pleasant	$1,400	$0
5781	82	2005-08-05	My Date With Drew	$1,100	$181,041

5782 rows × 6 columns

Internet Movie Database

My genre data comes from IMDb, a subsidiary of Amazon which is a well known source of movie information. Naturally, the most important column for my analysis will be `genres`. I later use this column to compute Pearson correlations between genres and different financial statistics.

This table is much larger than `tn`, with a little over 146,000 observations.

```
imdb = pd.read_csv(os.path.join('zippedData', 'imdb.title.basics.csv.gz'))
imdb
```

<style scoped> .dataframe tbody tr th:only-of-type { vertical-align: middle; }

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

</style>

	tconst	primary_title	original_title	start_year	runtime_minute
0	tt0063540	Sunghursh	Sunghursh	2013	175.00
1	tt0066787	One Day Before the Rainy Season	Ashad Ka Ek Din	2019	114.00
2	tt0069049	The Other Side of the Wind	The Other Side of the Wind	2018	122.00
3	tt0069204	Sabse Bada Sukh	Sabse Bada Sukh	2018	nan
4	tt0100275	The Wandering Soap Opera	La Telenovela Errante	2017	80.00
...	...	...	...	...	...
146139	tt9916538	Kuambil Lagi Hatiku	Kuambil Lagi Hatiku	2019	123.00
146140	tt9916622	Rodolpho Teóphilo - O Legado de um Pioneiro	Rodolpho Teóphilo - O Legado de um Pioneiro	2015	nan
146141	tt9916706	Dankyavar Danka	Dankyavar Danka	2013	nan

	tconst	primary_title	original_title	start_year	runtime_minute
146142	tt9916730	6 Gunn	6 Gunn	2017	116.00
146143	tt9916754	Chico Albuquerque - Revelações	Chico Albuquerque - Revelações	2013	nan

146144 rows × 6 columns

Data Preparation

Describe and justify the process for preparing the data for analysis.

Questions to consider:

- Were there variables you dropped or created?
- How did you address missing values or outliers?
- Why are these choices appropriate given the data and the business problem?

The Numbers

I start by replacing the incorrect `id` column with a column of genuinely unique ID numbers. I also create a `release_year` column, because it will come in handy later when merging tables.

```
del tn['id']
tn.insert(0, 'tn_id', np.arange(tn.shape[0]) + 1)
tn.insert(2, 'release_year', tn['release_date'].dt.year)
tn.head()
```

<style scoped> .dataframe tbody tr th:only-of-type { vertical-align: middle; }

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

</style>

	tn_id	release_date	release_year	movie	production_budget	dor
0	1	2009-12-18	2009	Avatar	$425,000,000	$76
1	2	2011-05-20	2011	Pirates of the Caribbean: On Stranger Tides	$410,600,000	$24
2	3	2019-06-07	2019	Dark Phoenix	$350,000,000	$42
3	4	2015-05-01	2015	Avengers: Age of Ultron	$330,600,000	$45
4	5	2017-12-15	2017	Star Wars Ep. VIII: The Last Jedi	$317,000,000	$62

The columns `production_budget` , `domestic gross` , and `worldwide gross` are in string format, so I remove the extraneous symbols and convert them to `np.float64` .

```
money_cols = ['production_budget', 'domestic_gross', 'worldwide_gross']
tn[money_cols] = (tn.loc[:, money_cols]
                    .apply(cleaning.process_strings)
                    .apply(lambda x: x.astype('float64')))
tn.sort_values('worldwide_gross').head()
```

<style scoped> .dataframe tbody tr th:only-of-type { vertical-align: middle; }

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

</style>

	tn_id	release_date	release_year	movie	production_budget	do
5037	5038	2019-04-23	2019	Living Dark: The Story of Ted the Caver	1,750,000.00	0.0
3975	3976	2015-05-15	2015	Pound of Flesh	7,500,000.00	0.0
4627	4628	2011-06-28	2011	2:13	3,500,000.00	0.0
4628	4629	2013-01-29	2013	Batman: The Dark Knight Returns, Part 2	3,500,000.00	0.0
3947	3948	2019-06-21	2019	Burn Your Maps	8,000,000.00	0.0

These 0 values for `domestic_gross` and `worldwide_gross` look very suspicious. Some of these 0s are for Netflix original productions such as *Bright* and *The Ridiculous 6*. Obviously those should not be counted as massive commercial failures simply because they were not released in theaters. Other 0s are for movies like *PLAYMOBIL*, which other sources report as generating revenue. Still other 0s are for movies which were released only domestically or only abroad.

```
tn.query('(domestic_gross == 0) & (worldwide_gross == 0)').head()
```

<style scoped> .dataframe tbody tr th:only-of-type { vertical-align: middle; }

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

</style>

	tn_id	release_date	release_year	movie	production_budget
194	195	2020-12-31	2020	Moonfall	150,000,000.00
479	480	2017-12-13	2017	Bright	90,000,000.00
480	481	2019-12-31	2019	Army of the Dead	90,000,000.00
535	536	2020-02-21	2020	Call of the Wild	82,000,000.00
670	671	2019-08-30	2019	PLAYMOBIL	75,000,000.00

I remove any rows where the domestic or worldwide gross is 0, since nearly every 0 is a null value or error.

```
tn = tn.loc[tn.query('(domestic_gross > 0) & (worldwide_gross > 0)').index]
tn.sort_values('worldwide_gross').head()
```

<style scoped> .dataframe tbody tr th:only-of-type { vertical-align: middle; }

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

</style>

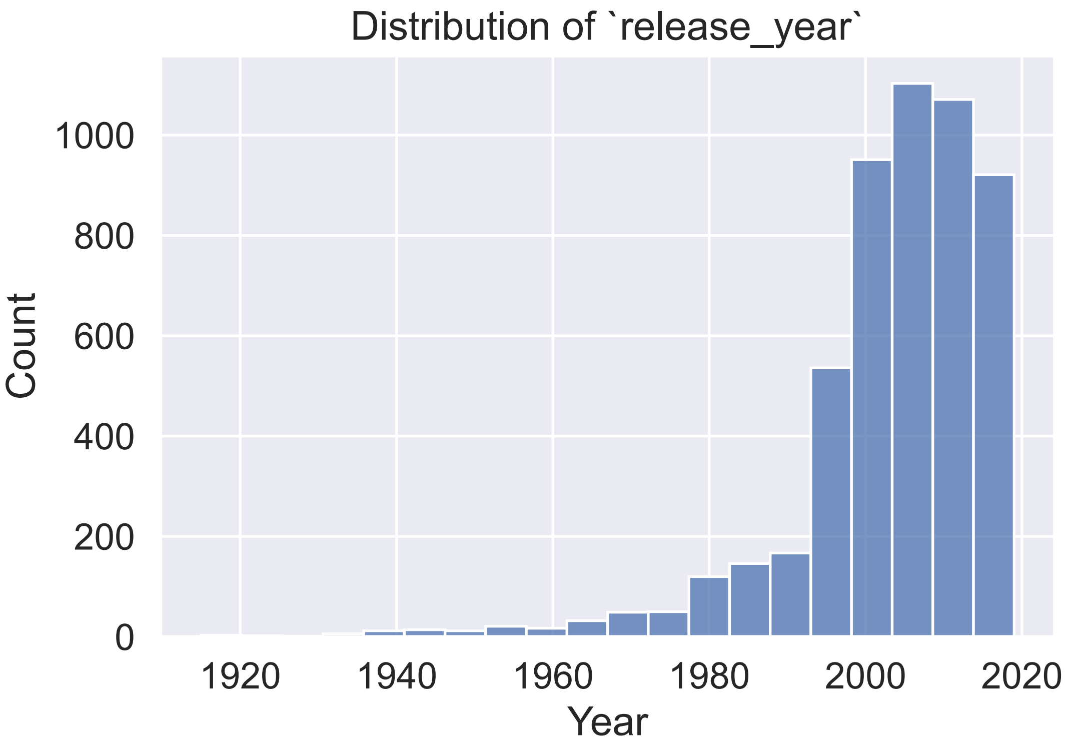
	tn_id	release_date	release_year	movie	production_budget	do
5770	5771	2008-08-14	2008	The Rise and Fall of Miss Thang	10,000.00	40

	tn_id	release_date	release_year	movie	production_budget	do
5518	5519	2005-10-13	2005	The Dark Hours	400,000.00	42
5769	5770	1996-04-01	1996	Bang	10,000.00	52
5466	5467	2018-05-11	2018	Higher Power	500,000.00	52
5027	5028	1993-01-01	1993	Ed and his Dead Mother	1,800,000.00	67

Looks like the data extends back in time much farther than I want.

```
ax = sns.histplot(data=tn, x='release_year', bins=20, palette='deep')
ax.set_title('Distribution of `release_year`')
ax.set_xlabel('Year')
```

```
Text(0.5, 0, 'Year')
```



I drop everything earlier than 2009 because I'm only interested in data that's relevant to current box office performance. 2020 was a particularly bad year because of the COVID-19 pandemic, so I leave that out as well.

```
tn = tn.loc[tn.query('(release_year <= 2019) & (release_year >= 2009)').index]
tn.sort_values('release_date').head()
```

<style scoped> .dataframe tbody tr th:only-of-type { vertical-align: middle; }

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

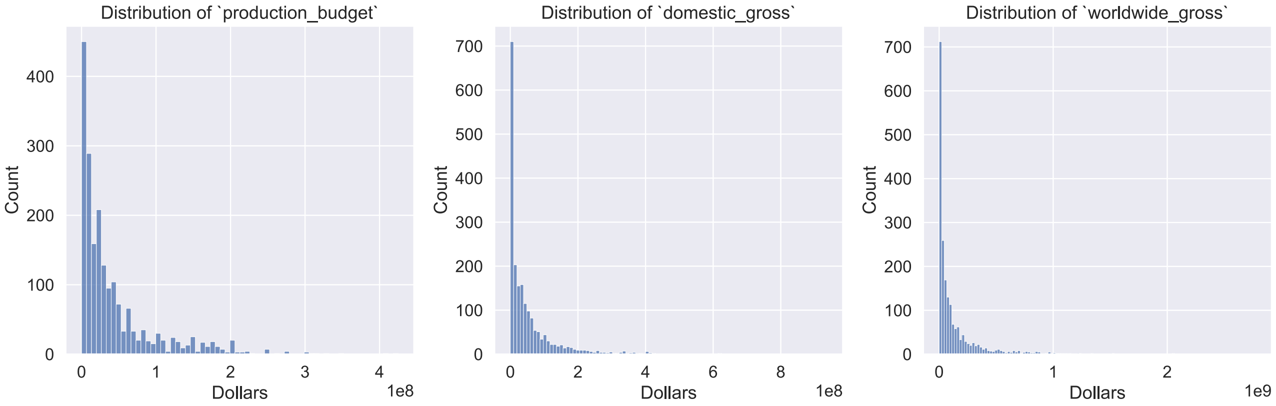
</style>

	tn_id	release_date	release_year	movie	production_budget
2934	2935	2009-01-09	2009	The Unborn	16,000,000.00
4318	4319	2009-01-09	2009	Not Easily Broken	5,000,000.00
1880	1881	2009-01-09	2009	Bride Wars	30,000,000.00
1164	1165	2009-01-16	2009	Defiance	50,000,000.00
2736	2737	2009-01-16	2009	Notorious	19,000,000.00

Looks like all of the basic money distributions are very right-skewed, which is not surprising. I expect there to be many more small films than big films, financially-speaking.

```
plotting.multi_hist(tn, include=money_cols, xlabel='Dollars', palette='deep')
```

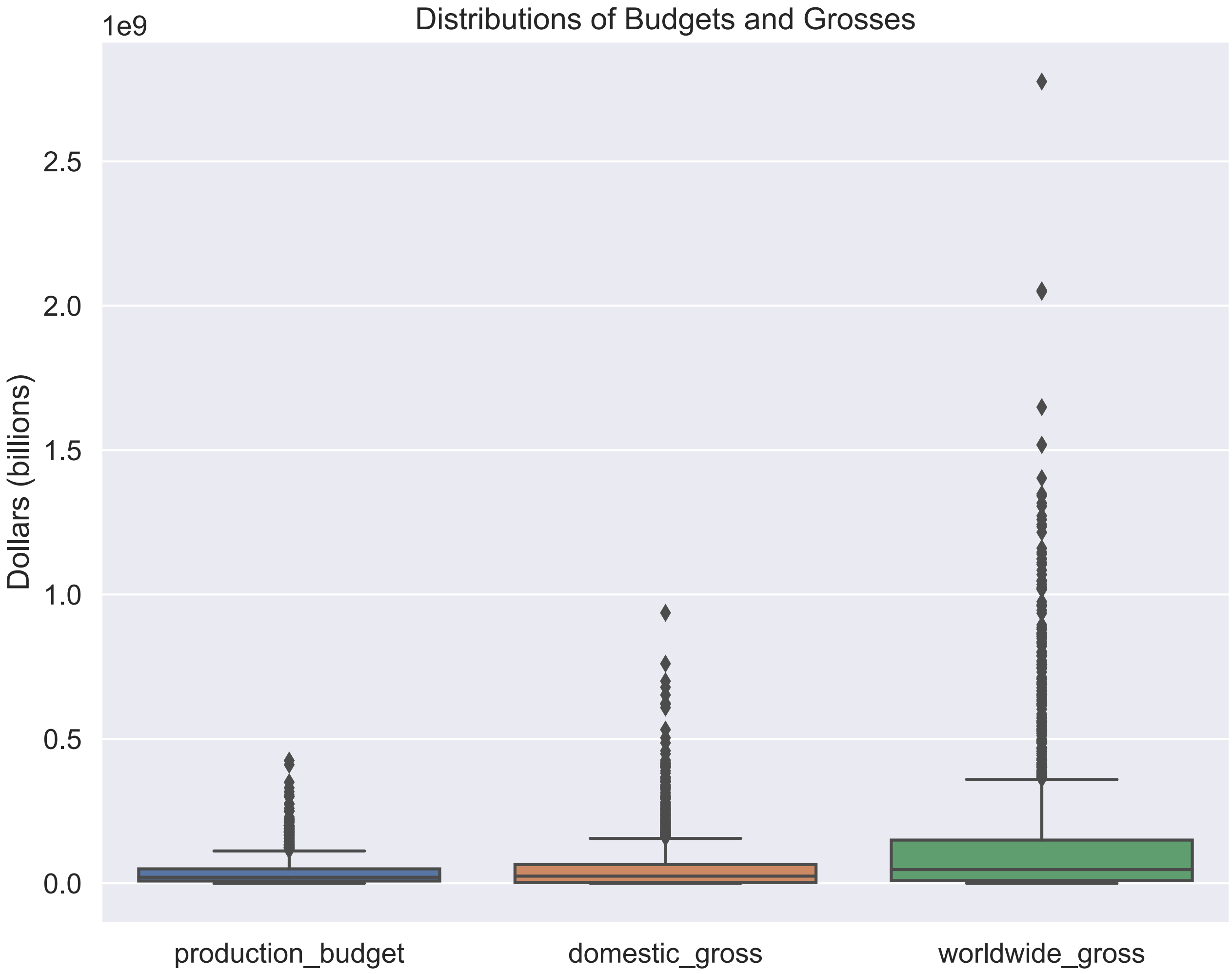
```
array([<AxesSubplot:title={'center':'Distribution of `production_budget`'},
xlabel='Dollars', ylabel='Count'>,
       <AxesSubplot:title={'center':'Distribution of `domestic_gross`'},
xlabel='Dollars', ylabel='Count'>,
       <AxesSubplot:title={'center':'Distribution of `worldwide_gross`'},
xlabel='Dollars', ylabel='Count'>],
      dtype=object)
```



These box plots indicate that there are many extreme values in the dataset. The data points beyond the upper whiskers are not truly outliers in this case. *Avatar* really does have a worldwide gross of 2.8 billion dollars. There is not a good scientific reason to altar or remove these values.

```
fix, ax = plt.subplots(figsize=(10, 8))
ax = sns.boxplot(data=tn[money_cols],
                  ax=ax,
                 palette='deep')
ax.set_title('Distributions of Budgets and Grosses')
ax.set_ylabel('Dollars (billions)')
```

```
Text(0, 0.5, 'Dollars (billions)')
```



Financial Calculations

I calculate domestic and worldwide profit by subtracting `production_budget` from each respective gross column.

```
tn['worldwide_profit'] = tn.eval('worldwide_gross - production_budget')
tn['domestic_profit'] = tn.eval('domestic_gross - production_budget')
tn.sort_values('worldwide_profit', ascending=False).head()
```

<style scoped> .dataframe tbody tr th:only-of-type { vertical-align: middle; }

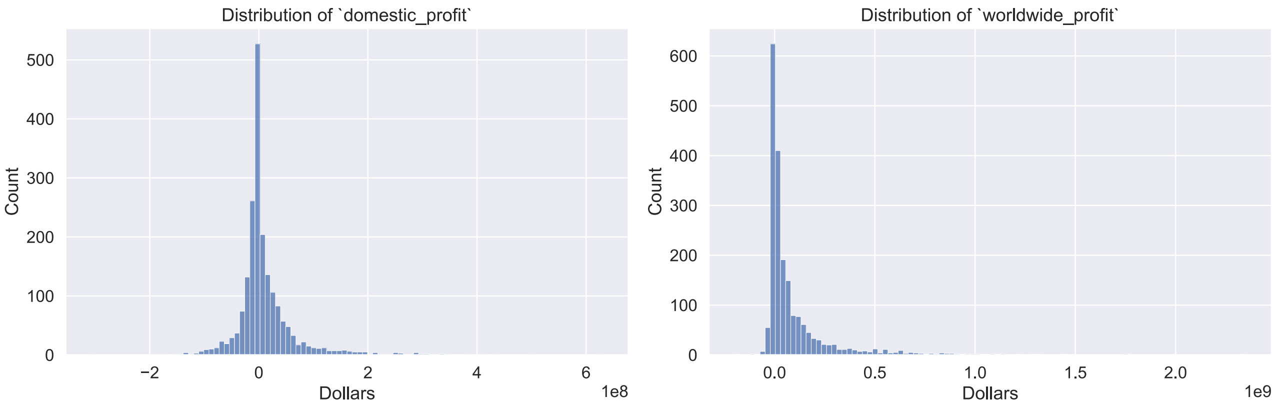
```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

</style>

	tn_id	release_date	release_year	movie	production_budget	do
0	1	2009-12-18	2009	Avatar	425,000,000.00	760
6	7	2018-04-27	2018	Avengers: Infinity War	300,000,000.00	678
5	6	2015-12-18	2015	Star Wars Ep. VII: The Force Awakens	306,000,000.00	936
33	34	2015-06-12	2015	Jurassic World	215,000,000.00	652
66	67	2015-04-03	2015	Furious 7	190,000,000.00	353

The distribution of `domestic_profit` is almost symmetrical around 0, although it is still right-skewed overall. The distribution of `worldwide_profit` is even more right-skewed. In both distributions the positive skew indicates that there are more winners than losers. This is unsurprising, since production companies strive to generate profit.

```
ax = plotting.multi_hist(tn,
                          include=['domestic_profit', 'worldwide_profit'],
                          xlabel='Dollars',
                          bins=100)
```



I calculate the percent return on investment (ROI) by dividing profit by budget and multiplying by 100. The sorted result is... ominous...

```
tn['worldwide_roi'] = tn.eval('(worldwide_profit / production_budget) * 100')
tn['domestic_roi'] = tn.eval('(domestic_profit / production_budget) * 100')
tn.sort_values('worldwide_roi', ascending=False).head()
```

<style scoped> .dataframe tbody tr th:only-of-type { vertical-align: middle; }

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

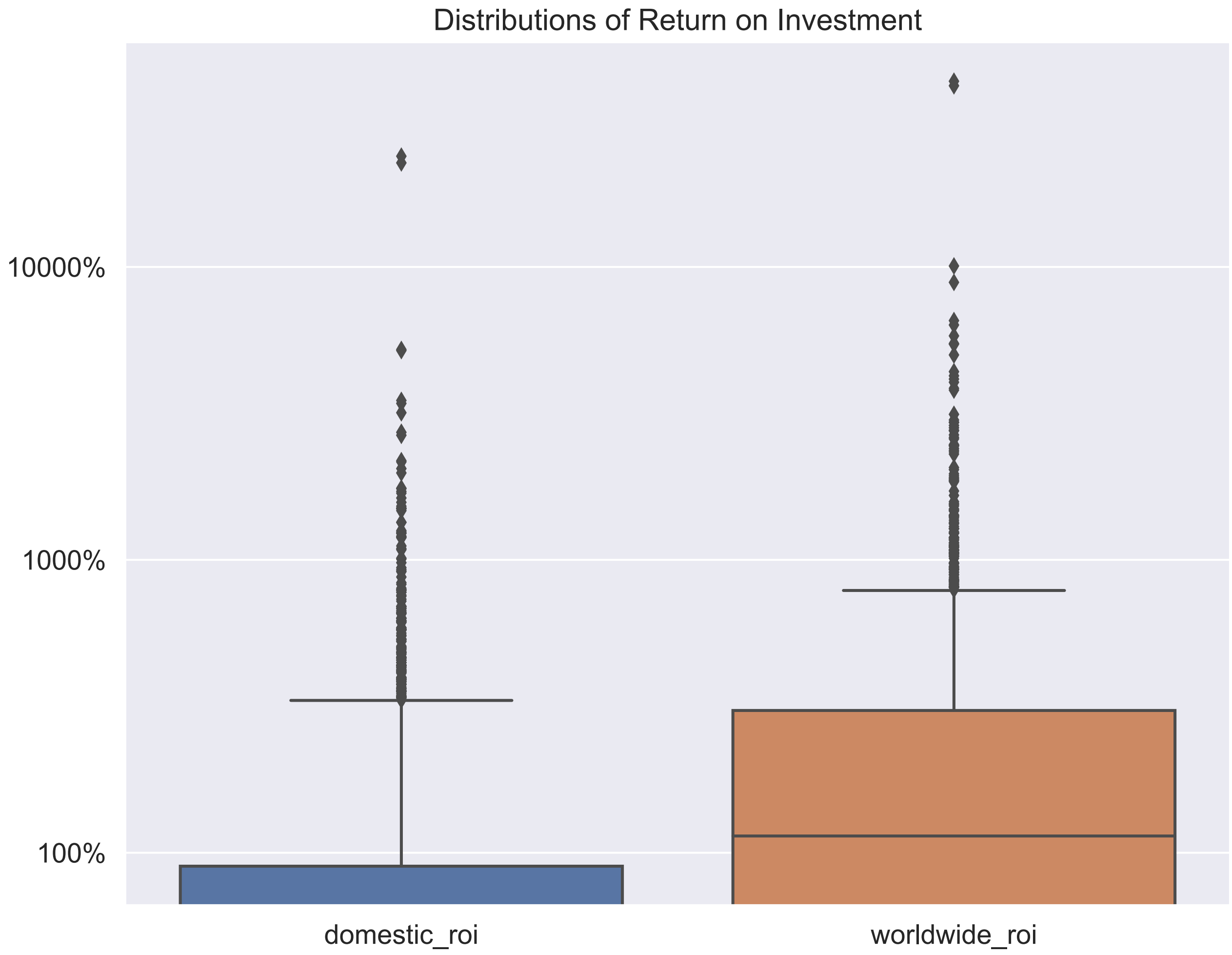
</style>

	tn_id	release_date	release_year	movie	production_budget
5492	5493	2009-09-25	2009	Paranormal Activity	450,000.00
5679	5680	2015-07-10	2015	The Gallows	100,000.00
5211	5212	2012-01-06	2012	The Devil Inside	1,000,000.00
5459	5460	2009-04-23	2009	Home	500,000.00
5062	5063	2011-04-01	2011	Insidious	1,500,000.00

The following is a box plot of `domestic_roi` and `worldwide_roi` plotted on a logarithmic scale. Interestingly, `domestic_roi` is heavily clustered under 100%, whereas the upper quartile of `worldwide_roi` is much higher. This is probably because production companies focus on the worldwide market nowadays.

There are a number of extreme values beyond the upper whiskers, but as you can see in the previous cell, these are just extremely successful horror movies. There is not a good scientific reason to altar or remove these data points.

```
fix, ax = plt.subplots(figsize=(10, 8))
ax = sns.boxplot(data=tn[['domestic_roi', 'worldwide_roi']],
                 ax=ax,
                 palette='deep')
ax.set_title('Distributions of Return on Investment')
ax.set_ylabel(None)
ax.set_yscale('log')
ax.yaxis.set_major_formatter(ticker.PercentFormatter())
```



Looks like there are some duplicate titles under `movie` , but those rows turn out to be acceptable.

```
cleaning.info(tn)
```

<style scoped> .dataframe tbody tr th:only-of-type { vertical-align: middle; }

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

</style>

	dup	dup_%	nan	nan_%
tn_id	0	0.00	0	0.00
release_date	1329	66.72	0	0.00
release_year	1981	99.45	0	0.00
movie	4	0.20	0	0.00
production_budget	1687	84.69	0	0.00
domestic_gross	0	0.00	0	0.00
worldwide_gross	0	0.00	0	0.00
worldwide_profit	0	0.00	0	0.00
domestic_profit	0	0.00	0	0.00
worldwide_roi	1	0.05	0	0.00
domestic_roi	1	0.05	0	0.00

```
tn[tn[['movie']].duplicated(keep=False)].sort_values('movie')
```

<style scoped> .dataframe tbody tr th:only-of-type { vertical-align: middle; }

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

</style>

	tn_id	release_date	release_year	movie	production_budget	d
2140	2141	2009-12-04	2009	Brothers	26,000,000.00	28
3307	3308	2015-08-14	2015	Brothers	13,000,000.00	65
243	244	2015-03-27	2015	Home	130,000,000.00	17
5459	5460	2009-04-23	2009	Home	500,000.00	15

	tn_id	release_date	release_year	movie	production_budget	d
38	39	2010-05-14	2010	Robin Hood	210,000,000.00	10
408	409	2018-11-21	2018	Robin Hood	99,000,000.00	30
5009	5010	2010-04-09	2010	The Square	1,900,000.00	40
5099	5100	2013-10-25	2013	The Square	1,500,000.00	12

Time to save the data and move on.

```
tn.to_json(os.path.join('cleanData', 'tn.profit.json'))
```

Internet Movie Database

After taking a look at my cleaning report, I can see that there are a number of duplicates under `primary_title` and many null values under `runtime_minutes`. I deal with the duplicates first, and later drop the `runtime_minutes` column altogether.

```
cleaning.info(imdb)
```

<style scoped> .dataframe tbody tr th:only-of-type { vertical-align: middle; }

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

</style>

	dup	dup_%	nan	nan_%
runtime_minutes	145776	99.75	31739	21.72
genres	145058	99.26	5408	3.70

	dup	dup_%	nan	nan_%
original_title	8370	5.73	21	0.01
tconst	0	0.00	0	0.00
primary_title	10073	6.89	0	0.00
start_year	146125	99.99	0	0.00

These duplicates are indeed going to be a problem.

```
imdb[imdb[['primary_title', 'original_title', 'start_year']
          ].duplicated(keep=False)].sort_values('primary_title')
```

<style scoped> .dataframe tbody tr th:only-of-type { vertical-align: middle; }

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

</style>

	tconst	primary_title	original_title	start_year	runtime_minute
103890	tt6085916	(aguirre)	(aguirre)	2016	97.00
106201	tt6214664	(aguirre)	(aguirre)	2016	98.00
129962	tt8032828	100 Milioni di bracciate	100 Milioni di bracciate	2017	nan
129979	tt8034014	100 Milioni di bracciate	100 Milioni di bracciate	2017	nan
20394	tt1855110	180	180	2011	121.00
...	...	...	...	...	...
66990	tt3815124	Ângelo de Sousa - Tudo o Que Sou Capaz	Ângelo de Sousa - Tudo o Que Sou Capaz	2010	60.00

	tconst	primary_title	original_title	start_year	runtime_minute
66992	tt3815128	Ângelo de Sousa - Tudo o Que Sou Capaz	Ângelo de Sousa - Tudo o Que Sou Capaz	2010	60.00
66995	tt3815134	Ângelo de Sousa - Tudo o Que Sou Capaz	Ângelo de Sousa - Tudo o Que Sou Capaz	2010	60.00
92592	tt5352034	Çagrilan	Çagrilan	2016	85.00
109103	tt6412726	Çagrilan	Çagrilan	2016	nan

3031 rows × 6 columns

I drop rows with duplicates across `primary_title`, `original_title`, and `start_year`.

```
imdb.drop_duplicates(
    subset=['primary_title', 'original_title', 'start_year'], inplace=True)
```

Next I preprocess the titles of both `imdb` and `tn` in preparation for the merge. Since these tables do not share a unique identifier, I have to merge them using the year and title fields.

My string processing function makes all characters lowercase, removes punctuation, and translates Unicode characters to ASCII.

```
imdb['clean_title'] = cleaning.process_strings(imdb.loc[:, 'primary_title'])
tn = tn.assign(clean_title=cleaning.process_strings(tn['movie']))
```

I merge the tables crudely along the year and title fields. While this merge is sufficient for my analysis, it is inefficient. Some movies are lost in translation because their titles do not match character-for-character between tables.

```
imdb = pd.merge(imdb,
                tn,
                how='inner',
                left_on=['start_year', 'clean_title'],
                right_on=['release_year', 'clean_title'])
```

```
display(imdb.shape)
imdb.head()
```

```
(1387, 18)
```

<style scoped> .dataframe tbody tr th:only-of-type { vertical-align: middle; }

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

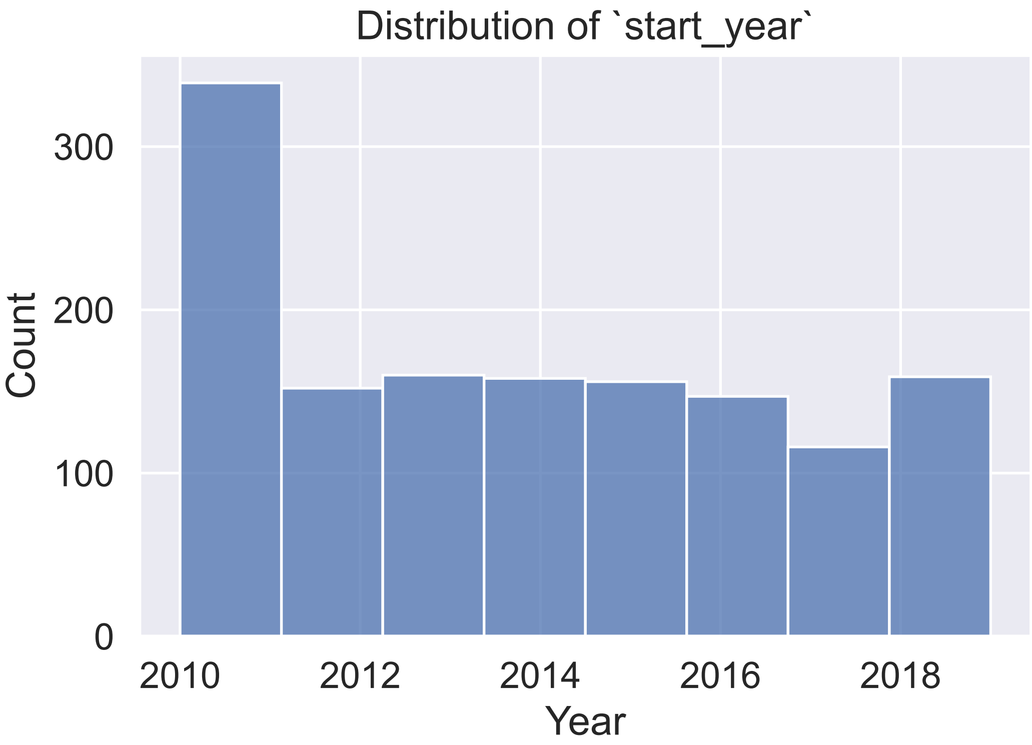
</style>

	tconst	primary_title	original_title	start_year	runtime_minutes
0	tt0359950	The Secret Life of Walter Mitty	The Secret Life of Walter Mitty	2013	114.00
1	tt0365907	A Walk Among the Tombstones	A Walk Among the Tombstones	2014	114.00
2	tt0369610	Jurassic World	Jurassic World	2015	124.00
3	tt0376136	The Rum Diary	The Rum Diary	2011	119.00
4	tt0383010	The Three Stooges	The Three Stooges	2012	92.00

Looks like the `start_year` range is appropriate. There is a large spike around 2010, which is not ideal. Unfortunately, I am working with a pretty small dataset at this point (~1400 observations), so I am reluctant to discard these early years.

```
ax = sns.histplot(imdb, x='start_year', bins=8, palette='deep')
ax.set_title('Distribution of `start_year`')
ax.set_xlabel('Year')
```

```
Text(0.5, 0, 'Year')
```



Next I drop all irrelevant or extraneous columns and check again for nulls and duplicates.

```
imdb.drop(columns=['start_year', 'release_year', 'clean_title',
          'movie', 'original_title', 'runtime_minutes'], inplace=True)
```

```
cleaning.info(imdb)
```

<style scoped> .dataframe tbody tr th:only-of-type { vertical-align: middle; }

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

</style>

	dup	dup_%	nan	nan_%

	dup	dup_%	nan	nan_%
tconst	0	0.00	0	0.00
primary_title	14	1.01	0	0.00
genres	1171	84.43	0	0.00
tn_id	14	1.01	0	0.00
release_date	820	59.12	0	0.00
production_budget	1146	82.62	0	0.00
domestic_gross	14	1.01	0	0.00
worldwide_gross	14	1.01	0	0.00
worldwide_profit	14	1.01	0	0.00
domestic_profit	14	1.01	0	0.00
worldwide_roi	15	1.08	0	0.00
domestic_roi	15	1.08	0	0.00

Everything looks to be in order, but I need to convert the `genres` column from `string` to `list` in order to pull apart the individual genre labels.

```
imdb['genres'] = imdb.loc[:, 'genres'].str.split(',')
imdb[['genres']]
```

<style scoped> .dataframe tbody tr th:only-of-type { vertical-align: middle; }

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

</style>

	genres
0	[Adventure, Comedy, Drama]

	genres
1	[Action, Crime, Drama]
2	[Action, Adventure, Sci-Fi]
3	[Comedy, Drama]
4	[Comedy, Family]
...	...
1382	[Horror, Thriller]
1383	[Crime, Drama, Thriller]
1384	[Drama, Horror, Mystery]
1385	[Documentary]
1386	[Biography, Drama]

1387 rows × 1 columns

Time to inspect the distribution of genres.

```
imdb.explode('genres')['genres'].value_counts()
```

```
Drama          671
Comedy         488
Action         421
Adventure      345
Thriller       235
Crime          214
Romance        183
Horror         150
Biography      129
Sci-Fi         129
Fantasy        120
Mystery        115
Animation      100
Family          87
Music           50
History         39
Documentary     35
Sport           32
War             17
Western         10
```

```
Musical          9
Name: genres, dtype: int64
```

Inspecting the movies in the "Music" genre reveals that they are in fact musicals. I collapse these two labels into "Musical".

```
imdb.explode('genres').query('genres == "Music"').head()
```

<style scoped> .dataframe tbody tr th:only-of-type { vertical-align: middle; }

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

</style>

	tconst	primary_title	genres	tn_id	release_date	production_bu
30	tt0475290	Hail, Caesar!	Music	2422	2016-02-05	22,000,000.00
128	tt1017451	The Runaways	Music	3757	2010-03-19	9,500,000.00
152	tt1068242	Footloose	Music	2339	2011-10-14	24,000,000.00
170	tt1126591	Burlesque	Music	1024	2010-11-24	55,000,000.00
195	tt1193631	Step Up 3D	Music	1909	2010-08-06	30,000,000.00

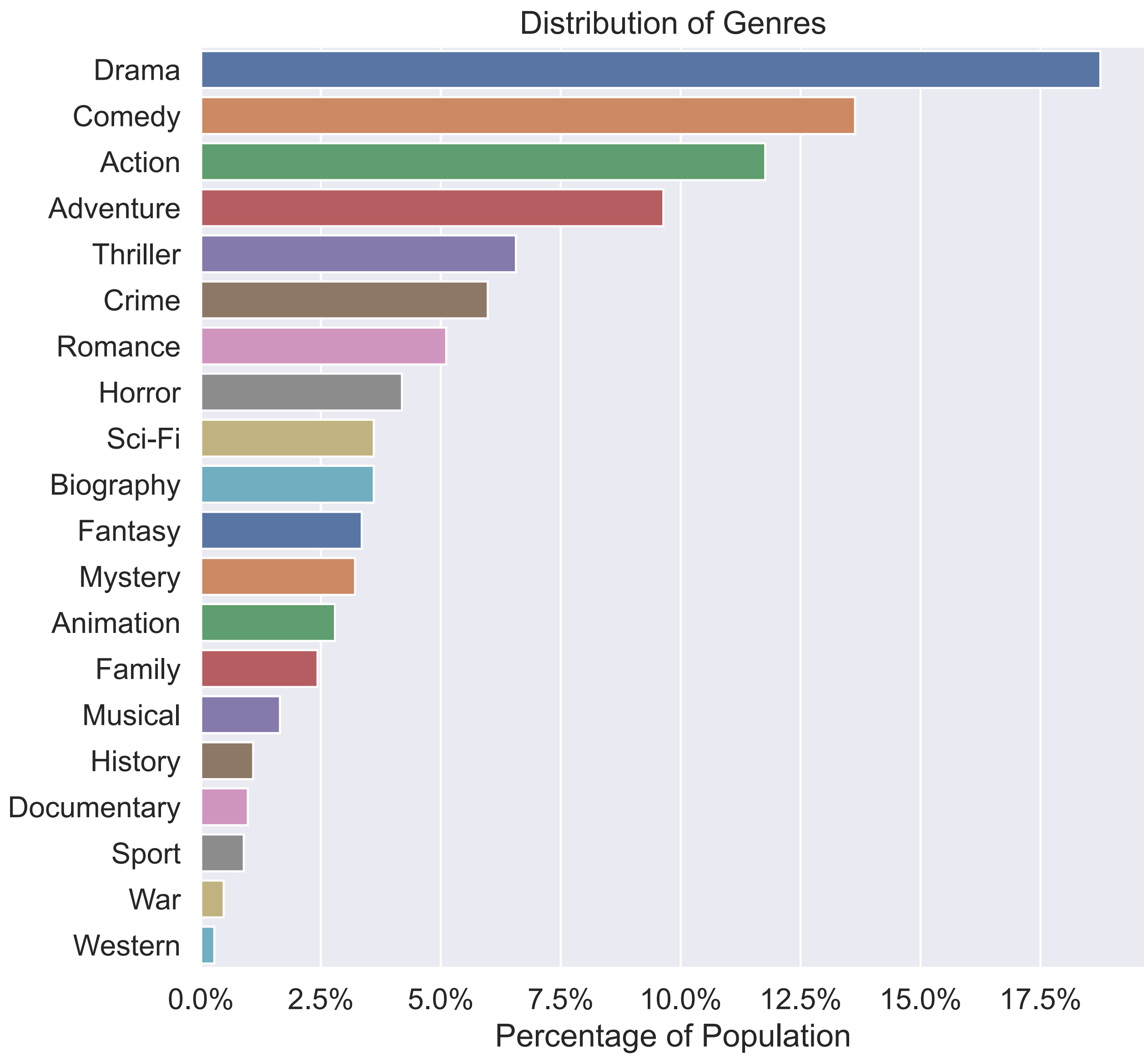
```
imdb['genres'] = utils.map_list_likes(
    imdb['genres'], lambda x: 'Musical' if x == 'Music' else x)
imdb.explode('genres')['genres'].value_counts()
```

```
Drama          671
Comedy         488
Action         421
Adventure      345
Thriller       235
Crime          214
Romance        183
```

```
Horror          150
Sci-Fi          129
Biography       129
Fantasy         120
Mystery         115
Animation       100
Family           87
Musical          59
History          39
Documentary      35
Sport            32
War              17
Western          10
Name: genres, dtype: int64
```

Here is the final genre distribution chart. I choose to keep low-frequency genres like "War" and "Western" unless they prove disruptive.

```
genre_counts = imdb.explode(
    'genres')['genres'].value_counts(normalize=True) * 100
fig, ax = plt.subplots(figsize=(8, 8))
ax = sns.barplot(x=genre_counts.values,
                 y=genre_counts.index, ax=ax, palette='deep')
ax.set_title('Distribution of Genres')
ax.set_xlabel('Percentage of Population')
ax.xaxis.set_major_formatter(ticker.PercentFormatter())
```



Distribution of Genres

Time to save the data and move on.

```
imdb.to_json(os.path.join('cleanData', 'imdb.tn.basics.json'))
```

Data Modeling

Cross-Tabulation

I begin by creating a movie-per-movie genre frequency table using cross-tabulation. Since no movie can have more than one of each genre (or less than zero), the frequencies can be interpreted as binary truth values. Now I can compute correlations between genres and financial outcomes.

```
combos = pd.crosstab(imdb.explode('genres')[
                     'tconst'], imdb.explode('genres')['genres'])
combos = combos.astype(np.bool_)
combos = combos.sort_index(axis=1).sort_index(axis=0)
combos.to_json(os.path.join('precomputed', 'genre_combos.json'))
combos.head()
```

<style scoped> .dataframe tbody tr th:only-of-type { vertical-align: middle; }

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

</style>

genres	Action	Adventure	Animation	Biography	Comedy	Crime
tconst						
tt0359950	False	True	False	False	True	False
tt0365907	True	False	False	False	False	True
tt0369610	True	True	False	False	False	False
tt0376136	False	False	False	False	True	False
tt0383010	False	False	False	False	True	False

I set the index of `imdb` to `tconst` for the upcoming computations. I need to use these unique IDs to relate the rows of `imdb` to the rows of `combos` .

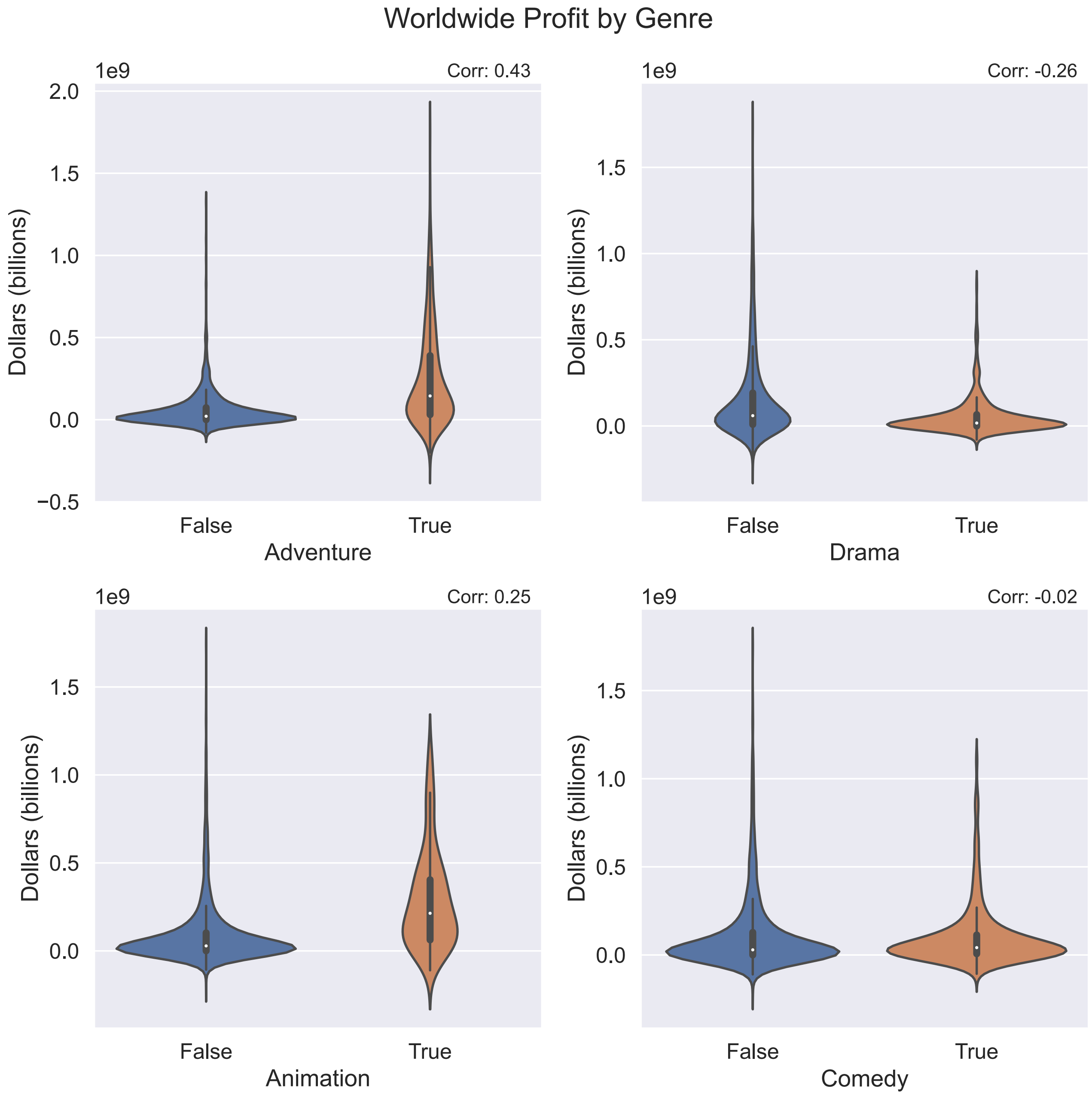
```
imdb.set_index('tconst', inplace=True)
```

Calculating Correlation

The **Pearson correlation coefficient** is a measure of the degree to which the relationship between two variables resembles a linear relationship. But it's hard to understand intuitively how genre could have anything approaching a linear relationship with, say, profit. What does that even mean?

It all makes good sense if you consider the following violin plots. The blobs indicate the location and density of the points in the distribution. Notice that genres which are positively correlated with profit have a fat violin on `False` and a narrow violin on `True`. Notice that genres which are negatively correlated with profit have a fat violin on `True` and a narrow violin on `False`. And finally, notice that genres with no correlation with profit have two fat violins.

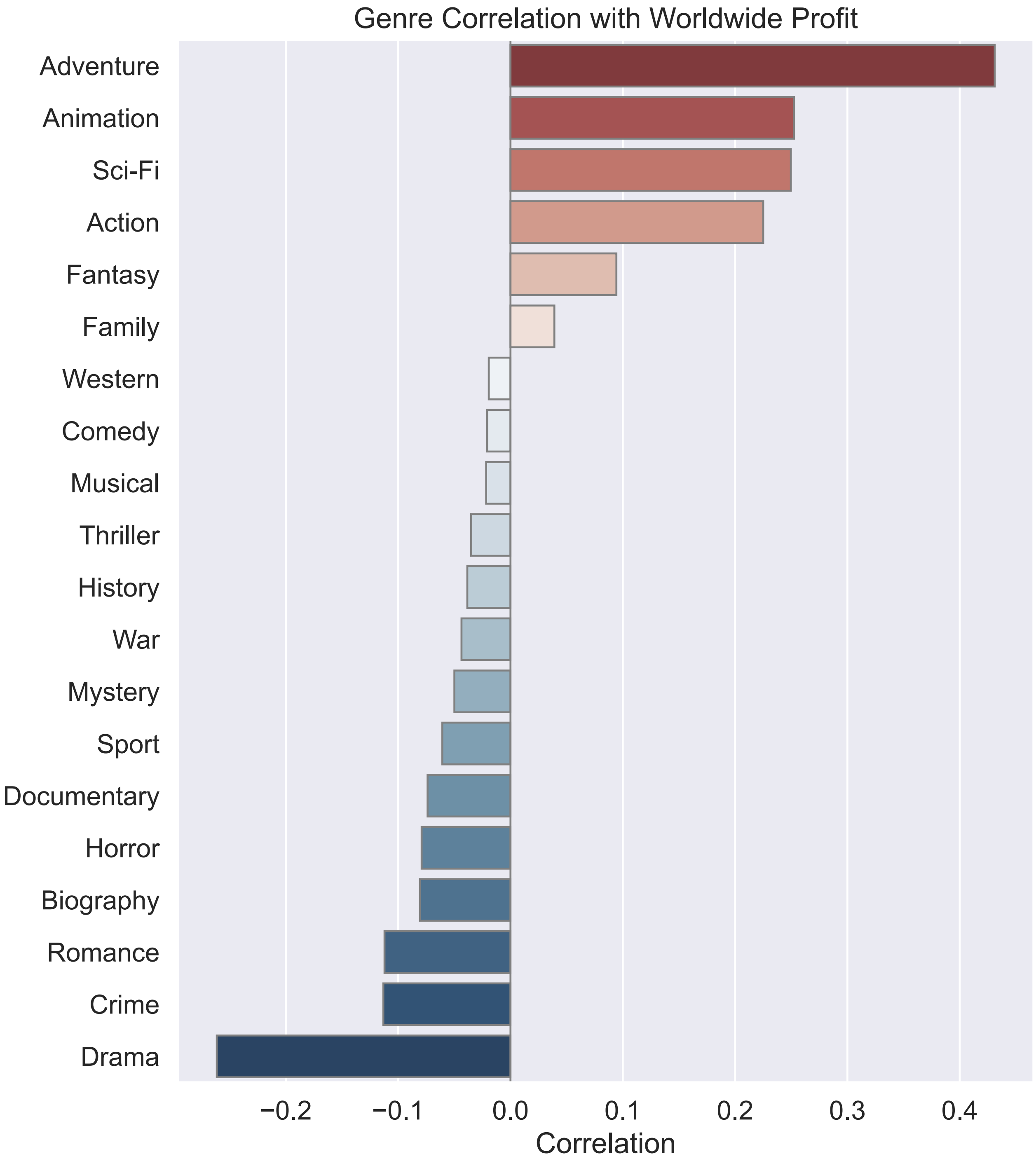
```
axes = plotting.boolean_violinplots(
        combos,
        imdb['worldwide_profit'],
        suptitle='Worldwide Profit by Genre',
        include=['Adventure', 'Drama', 'Animation', 'Comedy'],
        ylabel='Dollars (billions)',
        size=3,
        figsize=(10, 10),
        palette='deep')
plt.savefig(os.path.join('images', 'violinplots.jpg'),
        dpi=300,
        format='JPG')
```



Correlation with Profit

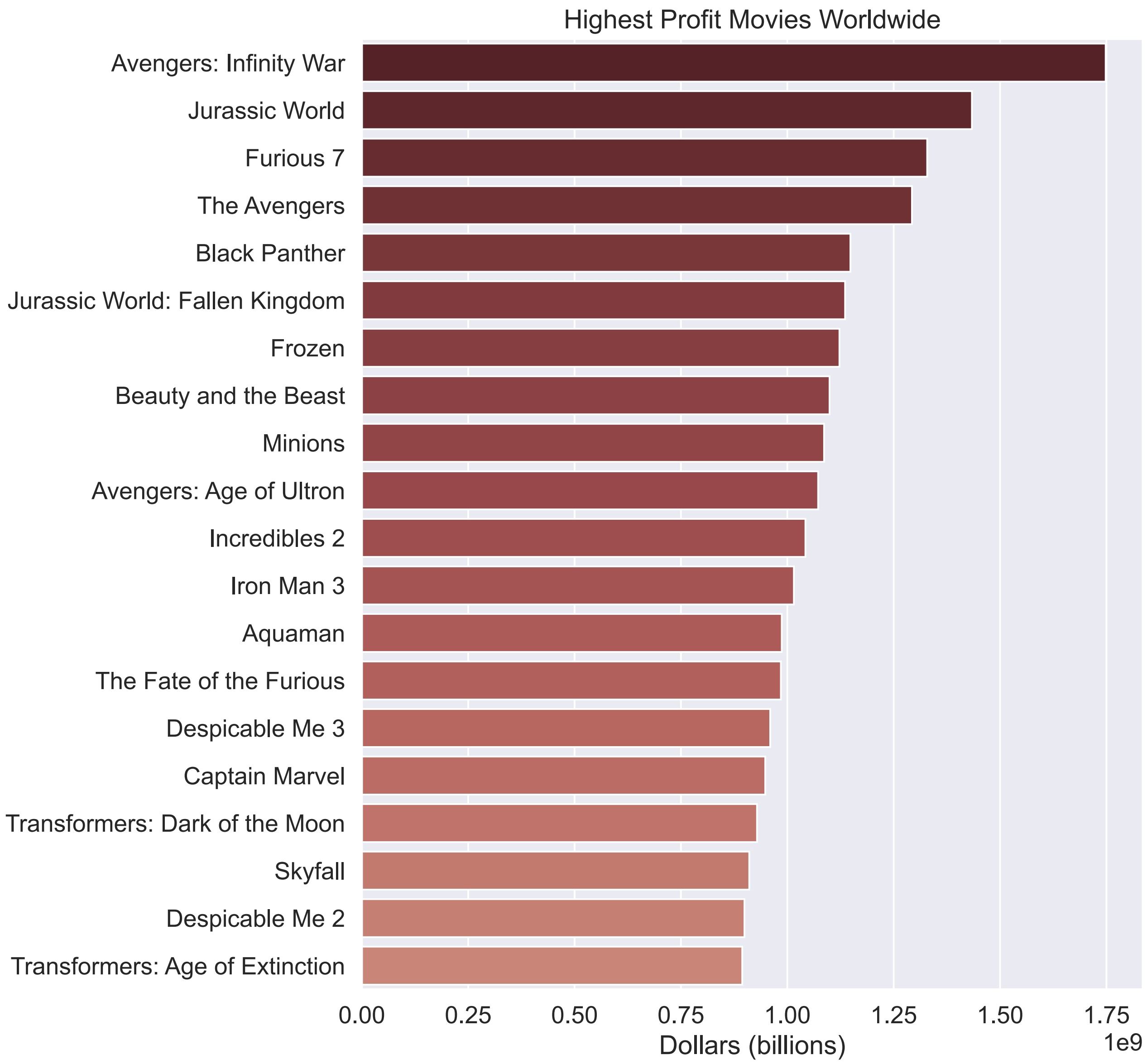
Here are the correlations between each genre and worldwide profit. Notice that the frontrunners are "Adventure", "Animation", "Sci-Fi", and "Action". Also notice that "Drama" has the strongest negative correlation. This is an interesting result.

```
ax = plotting.cat_correlation(combos, imdb['worldwide_profit'])
ax.set_title('Genre Correlation with Worldwide Profit')
ax.set_ylabel(None)
plt.savefig(os.path.join('images', 'corr_world_profit.jpg'),
            dpi=300,
            format='JPG')
```



As a sanity check, I plot the movies with the highest worldwide profit. Many adventure, sci-fi, and action titles show up: *The Avengers*, *Jurassic World*, *Black Panther*, *The Dark Knight Rises*. There are also several animated films: *Frozen*, *Beauty and the Beast*, *Incredibles 2*. Looks like the correlation numbers make sense.

```
reds = sns.color_palette('Reds_r', 40, desat=0.6)
ax = plotting.topn_ranking(imdb,
                           'primary_title',
                           'worldwide_profit',
                           20,
                           figsize=(8, 10),
                                        palette=reds)
ax.set_title('Highest Profit Movies Worldwide')
ax.set_ylabel(None)
ax.set_xlabel('Dollars (billions)')
plt.savefig(os.path.join('images', 'top_world_profit.jpg'),
            dpi=300,
            format='JPG')
```



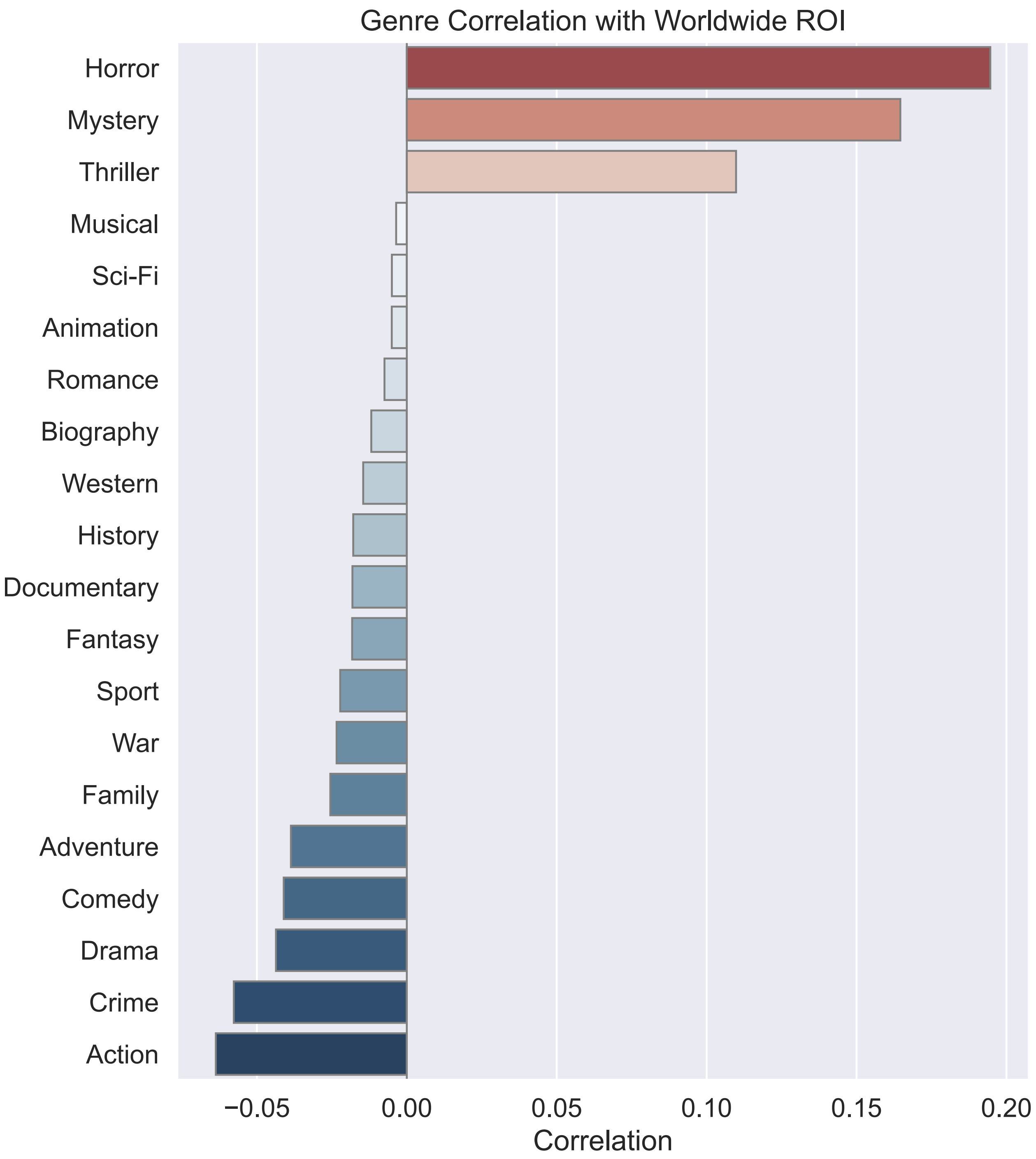
Correlation with ROI

The correlations with worldwide ROI are strikingly different from those with profit. Horror? Mystery? Thriller? These all had a weak negative correlation with worldwide profit. Why are they suddenly the only positive values?

Here's my conjecture: it's because ROI places heavy weight on budget, and top-earning horror films are often very low-budget. A low-budget film can generate revenue which is exponentially higher than its budget. A high-budget film will have a hard time doing that.

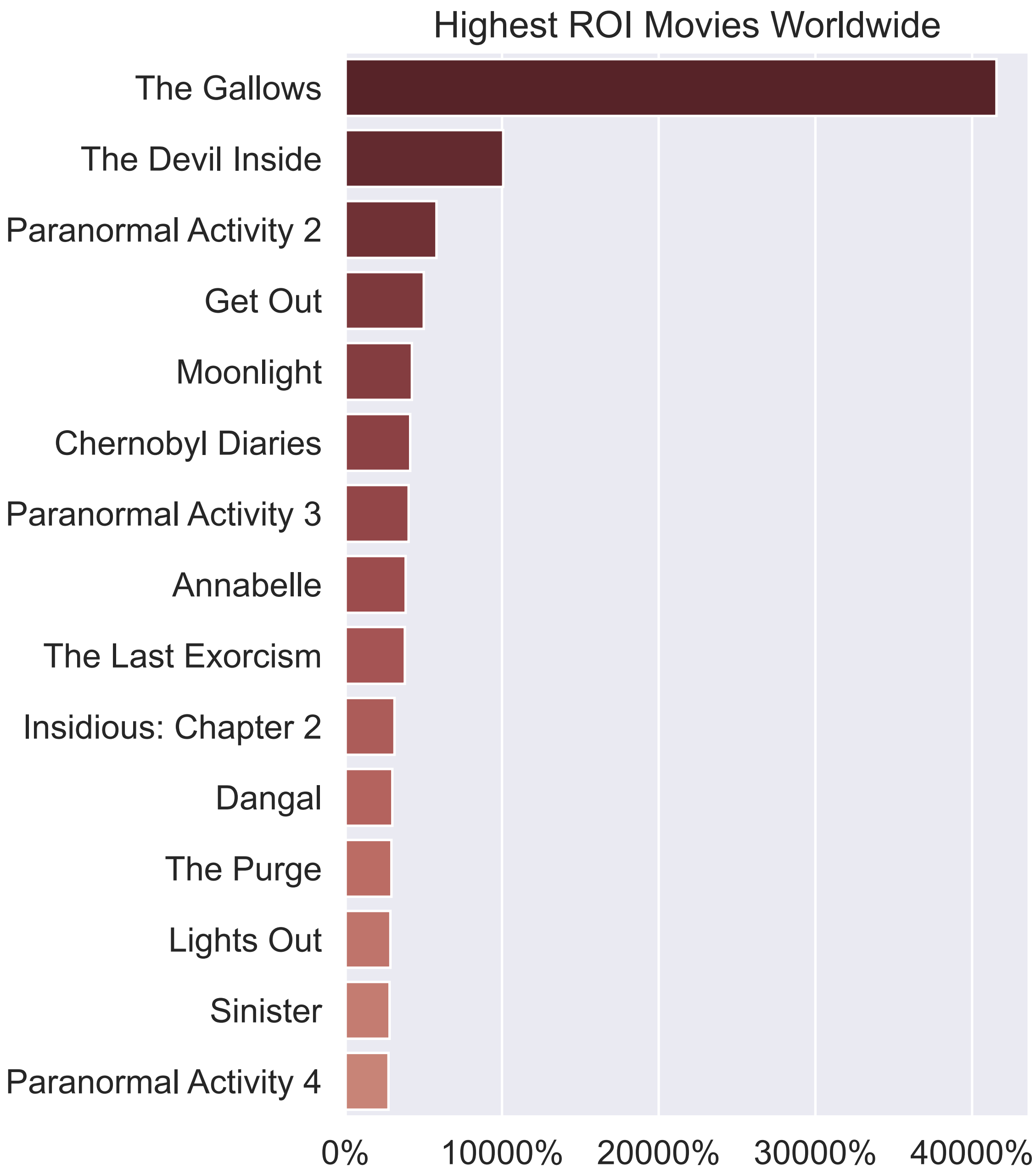
Horror movies have a reputation for being low-budget. *Paranormal Activity*, for example, is well-known for its low budget. *The Blair Witch Project* is another obvious example, since it's just a shaky-cam movie with a bunch of kids in the woods. Nonetheless, both of these movies were highly successful at the box office.

```
ax = plotting.cat_correlation(combos, imdb['worldwide_roi'])
ax.set_title('Genre Correlation with Worldwide ROI')
ax.set_ylabel(None)
plt.savefig(os.path.join('images', 'corr_world_roi.jpg'),
            dpi=300,
            format='JPG')
```



Here's another sanity check: the highest ROI movies worldwide. Nearly all of them are horror titles.

```
reds = sns.color_palette('Reds_r', 30, desat=0.6)
ax = plotting.topn_ranking(imdb,
                           'primary_title',
                           'worldwide_roi',
                           15,
                           figsize=(5, 8),
                           palette=reds)
ax.xaxis.set_major_formatter(ticker.PercentFormatter())
ax.set_title('Highest ROI Movies Worldwide')
ax.set_xlabel(None)
ax.set_ylabel(None)
plt.savefig(os.path.join('images', 'top_world_roi.jpg'),
            dpi=300,
            format='JPG')
```



Effects of Budget

Next, I partition the movies by budget quartile. "Low Budget" refers to the lower quartile (25th percentile) and below. "High Budget" refers to the upper quartile (75th percentile) and above. I want to plot the genre-ROI-correlations for low-budget films alongside those for high-budget films.

```
quartile_labels = ['Low Budget', 'Mid-Low Budget',
                   'Mid-High Budget', 'High Budget']
imdb['budget_quartile'] = pd.qcut(
    imdb['production_budget'], 4, quartile_labels)
quartile_intervals = pd.qcut(imdb['production_budget'], 4).dtype.categories
world_roi_by_budget = combos.groupby(
    imdb['budget_quartile']).corrwith(imdb['worldwide_roi'])
world_roi_by_budget
```

<style scoped> .dataframe tbody tr th:only-of-type { vertical-align: middle; }

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

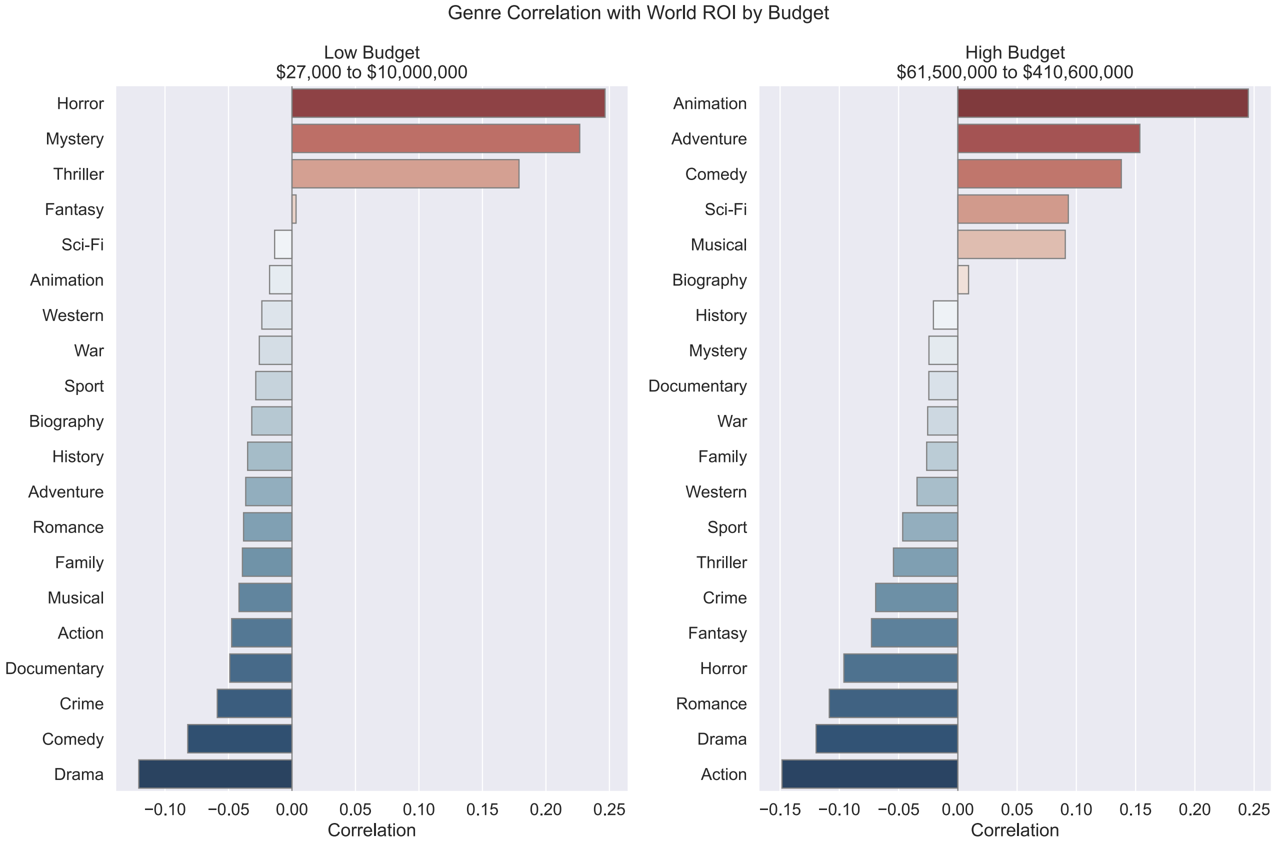
</style>

genres	Action	Adventure	Animation	Biography	Comedy	Cr
budget_quartile						
Low Budget	-0.05	-0.04	-0.02	-0.03	-0.08	-0
Mid-Low Budget	-0.11	-0.04	-0.01	0.05	0.04	-0
Mid-High Budget	-0.02	-0.04	-0.02	-0.02	-0.03	-0
High Budget	-0.15	0.15	0.25	0.01	0.14	-0

Here's a plot of worldwide ROI computed separately for low-budget films and high-budget films. Looks like evidence supporting my conjecture that top-earning horror films are often very low-budget, and that low-budget movies are capable of achieving very high ROI.

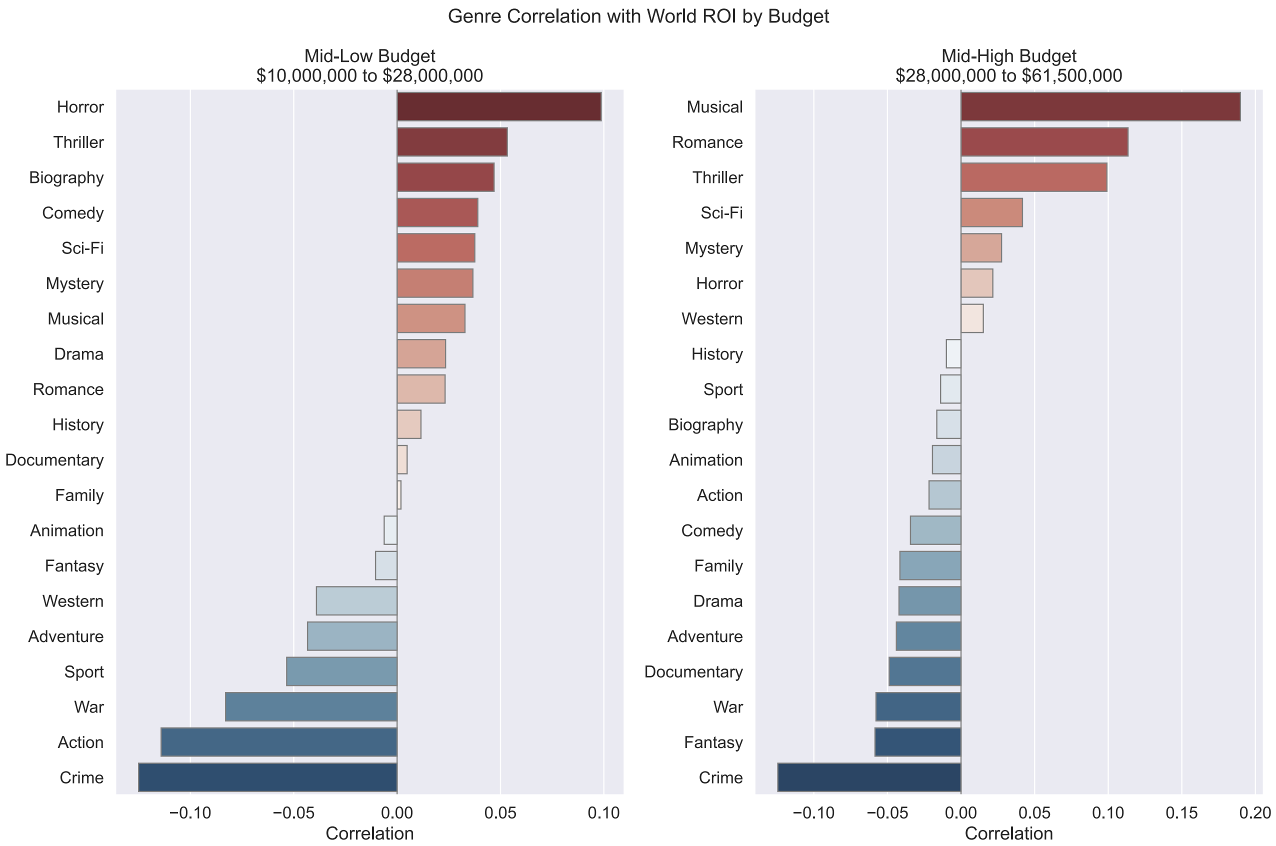
Interestingly animation, and not adventure, is the frontrunner for high-budget films. Adventure, which led in correlation with worldwide profit, is now in second place.

```
plotting.cat_corr_by_bins(world_roi_by_budget,
                          'Low Budget',
                          'High Budget',
                          quartile_intervals[0],
                          quartile_intervals[3],
                          'Genre Correlation with World ROI by Budget')
plt.savefig(os.path.join('images', 'corr_world_roi_by_budget.jpg'),
                            dpi=300,
                            format='JPG')
```



Next is the analogous plot for midrange budgets. The correlation scores here are lower, but you can see that Horror and Thriller are still at the top of the mix for mid-low budget films. It's notable that Musical and Romance movies lead the way for mid-high budget films. These genres definitely go together.

```
plotting.cat_corr_by_bins(world_roi_by_budget,
                          'Mid-Low Budget',
                          'Mid-High Budget',
                          quartile_intervals[1],
                          quartile_intervals[2],
                          'Genre Correlation with World ROI by Budget')
plt.savefig(os.path.join('images', 'corr_world_roi_by_budget_mid.jpg'),
            dpi=300,
            format='JPG')
```



I perform the same calculations for domestic ROI. The results are similar, with some small differences. Notably, Comedy has risen up in ranking considerably for everything but low-budget films.

```
domestic_roi_by_budget = combos.groupby(
    imdb['budget_quartile']).corrwith(imdb['domestic_roi'])
domestic_roi_by_budget
```

<style scoped> .dataframe tbody tr th:only-of-type { vertical-align: middle; }

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

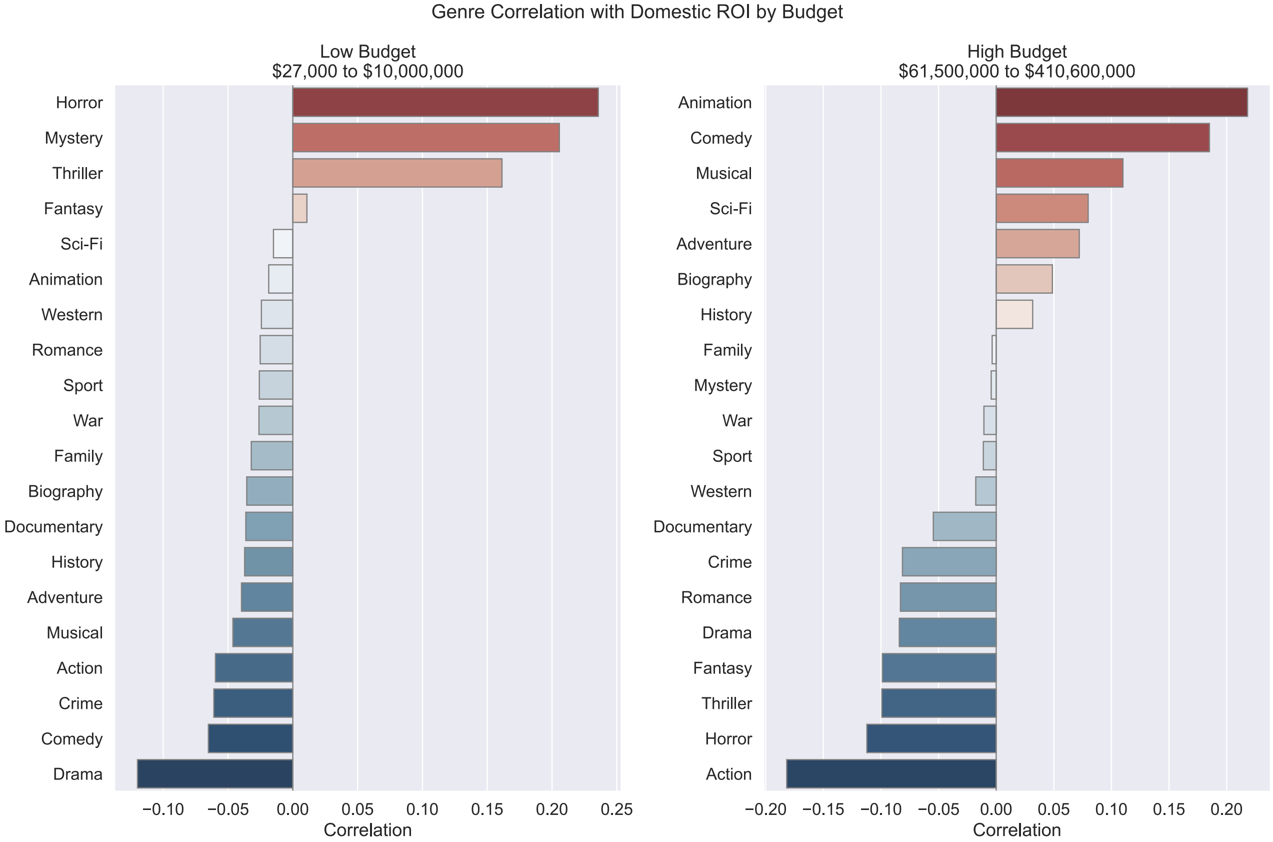
</style>

genres	Action	Adventure	Animation	Biography	Comedy	Cr
budget_quartile						

genres	Action	Adventure	Animation	Biography	Comedy	Cr
budget_quartile						
Low Budget	-0.06	-0.04	-0.02	-0.04	-0.07	-0
Mid-Low Budget	-0.12	-0.01	0.01	0.04	0.09	-0
Mid-High Budget	-0.09	-0.07	-0.03	0.02	0.08	-0
High Budget	-0.18	0.07	0.22	0.05	0.19	-0

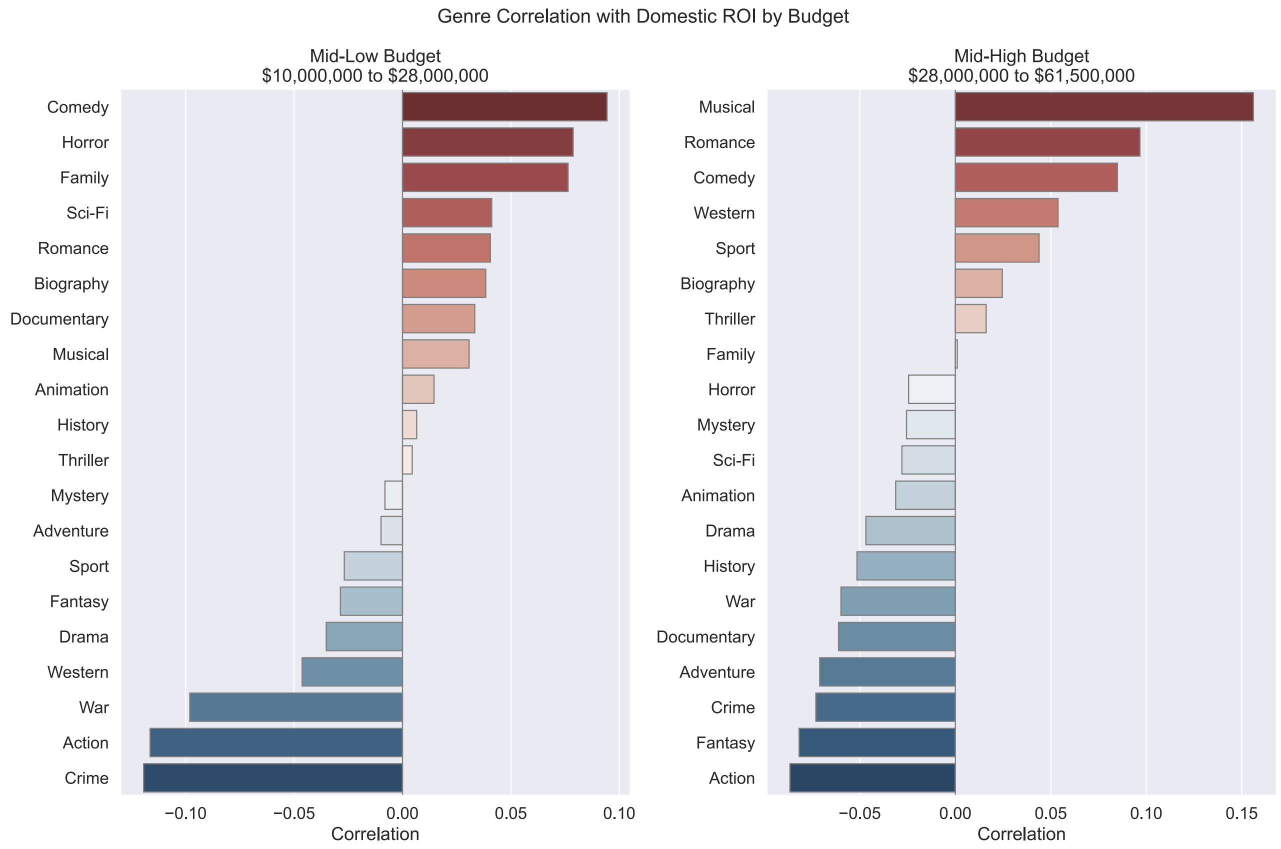
```
plotting.cat_corr_by_bins(domestic_roi_by_budget,
                          'Low Budget',
                          'High Budget',
                          quartile_intervals[0],
                          quartile_intervals[3],
                          'Genre Correlation with Domestic ROI by Budget')
```

```
array([<AxesSubplot:title={'center':'Low Budget\n\\$27,000 to \\$10,000,000'},
xlabel='Correlation'>,
       <AxesSubplot:title={'center':'High Budget\n\\$61,500,000 to
\\$410,600,000'}, xlabel='Correlation'>],
      dtype=object)
```



```
plotting.cat_corr_by_bins(domestic_roi_by_budget,
                          'Mid-Low Budget',
                          'Mid-High Budget',
                          quartile_intervals[1],
                          quartile_intervals[2],
                          'Genre Correlation with Domestic ROI by Budget')
```

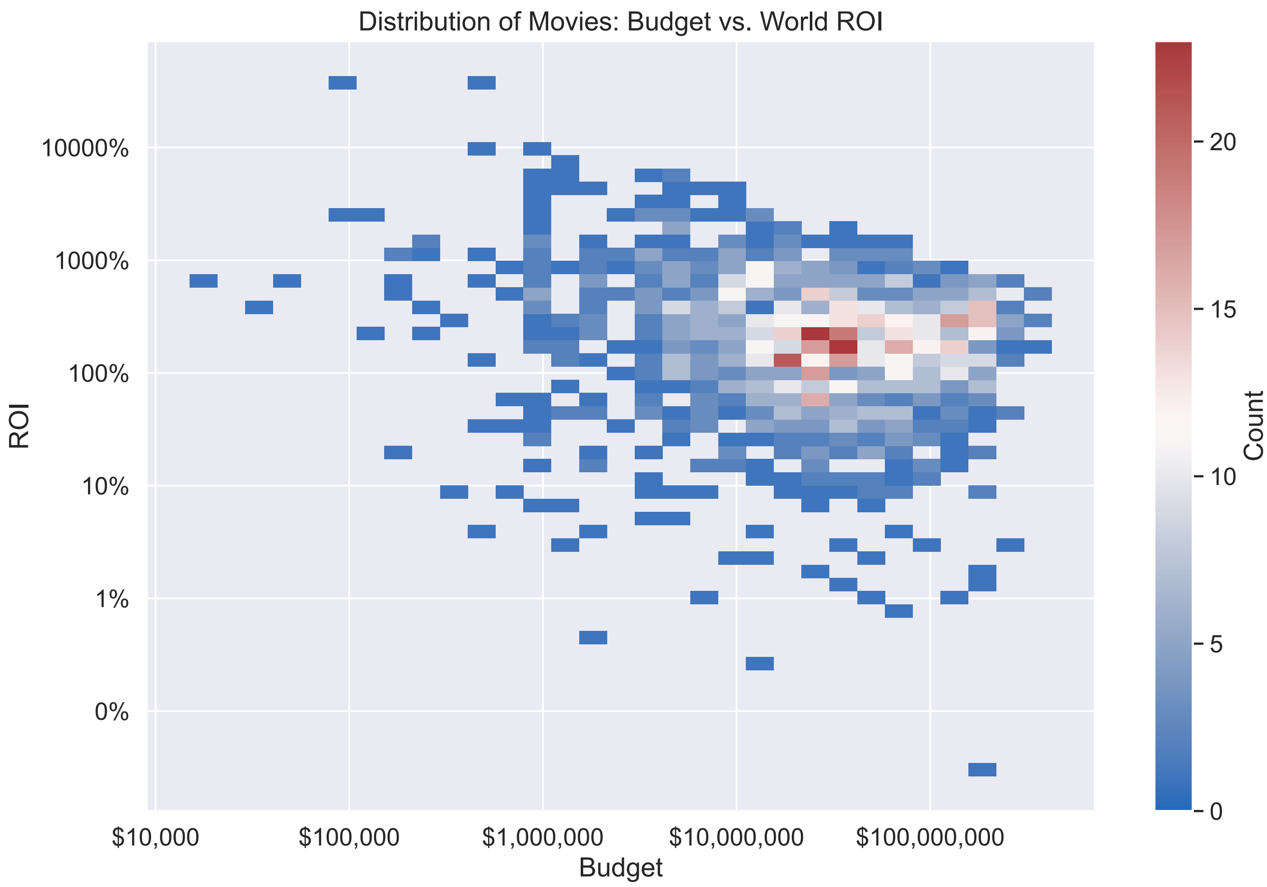
```
array([<AxesSubplot:title={'center':'Mid-Low Budget\n\\$10,000,000 to
\\$28,000,000'}, xlabel='Correlation'>,
       <AxesSubplot:title={'center':'Mid-High Budget\n\\$28,000,000 to
\\$61,500,000'}, xlabel='Correlation'>],
      dtype=object)
```



Budget Independently?

Could it be that low-budget movies simply have higher ROI in general than high-budget movies? No. Having a low-budget makes it possible to achieve an extremely high ROI percentage, but is not generally conducive to having high ROI. The following bivariate histogram shows that the highest concentration of movies is located between \$10M and \$100M with an ROI in the 100s.

```
fig, ax = plt.subplots(figsize=(12, 8))
cmap = sns.color_palette('vlag', as_cmap=True)
pos_world_rois = tn.query('worldwide_roi > 0')
ax = sns.histplot(data=pos_world_rois,
                  x='production_budget',
                  y='worldwide_roi',
                  ax=ax,
                  bins='auto',
                  stat='count',
                  cbar=True,
                  log_scale=True,
                  cmap=cmap,
                  cbar_kws={'label': 'Count'})
ax.yaxis.set_major_formatter(ticker.PercentFormatter())
ax.xaxis.set_major_formatter(ticker.StrMethodFormatter('${x:,.0f}'))
ax.set_xlabel('Budget')
ax.set_ylabel('ROI')
ax.set_title('Distribution of Movies: Budget vs. World ROI')
plt.savefig(os.path.join('images', 'budget_vs_world_roi.jpg'),
            dpi=300,
            format='JPG')
```



Production budget has almost no correlation with world ROI, though it is weakly negative.

```
tn[['production_budget']].corrwith(tn['worldwide_roi'])
```

```
production_budget   -0.04
dtype: float64
```

Conclusions

For high-budget productions, go with animation.

Animation has by far the strongest correlation (nearly 0.25) with ROI for high-budget films. The next best score is adventure, which is nearly 0.1 lower.

For low-budget productions, go with horror.

Nothing beats horror movies in terms of ROI, both overall and for low-budget films. The only other options are mystery and thriller, which both go along with horror anyway.

Stay away from drama, action, and crime.

Drama, action, and crime consistently show up in the negative on correlation with ROI. This means that movies achieve higher ROI when they are not drama, action, or crime. While it's possible to have success with these genres, they are the worst choices from an investment standpoint.

Evaluation

My analysis provides some useful insights for Microsoft, but there is much more work to be done. For one, making a successful movie is much more complicated than choosing a genre. There are numerous other factors to consider, such as cast and crew.

Furthermore, I conducted my analysis with a very limited dataset of around 1,400 observations. Many movies were lost in the merge between `imdb` and `tn` because these tables had no unique identifiers in common. The merge could be improved by using fuzzy string matching or another sophisticated process for dirty merging. The ideal situation would be to find a source of data which provides both genre labels and finances.

Nonetheless, I am very confident in the finding that horror movies have the highest ROI. That was a very robust and striking pattern in the data. I am fairly confident in my other findings relating to the business recommendations, but I would like to conduct further research.

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Languages

● **Jupyter Notebook** 99.7% ● **Python** 0.3%