Project: Investigate a tmdb movies Dataset

Table of Contents

- Introduction
- Data Wrangling
- Exploratory Data Analysis
- Conclusions

Introduction

> In the course of the report, I'll provide a brief introduction to the dataset we've selected for analysis. At the end of this section, we described the questions that we have plan on exploring over the course of the report.

```
In [1]: %matplotlib inline
        import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        import seaborn as sns
        import warnings
        warnings.filterwarnings('ignore')
```

TMDB Movies Dataset

This data set contains information about 10,000 movies collected from The Movie Database (TMDb), including user ratings and revenue. We are free to provide helpful questions and its answers.

Questions

- What genres are most popular from year to year?
- What kinds of properties are associated with movies that have high revenue?
- Who top 15 directors gross the most revenue?
- What is the relationship between genres?

Data Wrangling

> In the section of the report, we will load in the data, check for cleanliness, and then trim and clean our dataset for analysis. Then we will make sure that we document our steps and reasoning for cleaning the data in the report.

General Properties

A quick review of our data to see what we dealing with.

```
In [2]: # Load your data and print out a few lines. Perform any inspection
        df = pd.read_csv('tmdb-movies.csv')
        df.head(5)
```

Out[2]:

	id	imdb_id	popularity	budget	revenue	original_title	cast	homepage	director	tagline	keywords	overview	runtime	genres	production_companies	release_date	vote_count	vote_average	release_year	budget_adj	revenue_adj
0	135397	tt0369610	32.985763	150000000	1513528810	Jurassic World	Chris Pratt|Bryce Dallas Howard|Irrfan Khan|Vi...	http://www.jurassicworld.com/	Colin Trevorrow	The park is open.	monster dinosaur|dna|tyrannosaurus rex|genetic ...	A new genetically modified species of dinosaur ...	124	Action|Adventure|Science Fiction|Thriller	Universal Studios|Amblin Entertainment|Legend...	6/9/15	5562	6.5	2015	137999939.3	1392445892
1	76341	tt1392190	28.419936	150000000	378436354	Mad Max: Fury Road	Tom Hardy|Charlize Theron|Hugh Keays-Byrne|Nic...	http://www.madmaxmovie.com/	George Miller	What a lovely day.	post apocalypse|manhunt|chase|guy with gun|c...	Set in a post-apocalyptic wasteland, in a high-...	120	Action|Adventure|Science Fiction|Thriller	Village Roadshow Pictures|Kennedy Miller Produ...	5/13/15	6071	7.1	2015	137999939.3	348161292
2	262500	tt2908446	13.112507	110000000	295238201	Insurgent	Shailene Woodley|Ansel Elgort|Kate Winslet|Ke...	http://www.insurgentmovie.com/	Robert Schwentke	One choice can destroy everything	dystopia|based on young adult novel|sequel|ch...	Beatrice Prior must confront her inner demons ...	119	Adventure|Science Fiction|Thriller	Summit|Red Wagon Entertainment|Neal H. Moritz ...	3/18/15	2908	6.4	2015	107999999.7	271619025.4
3	140607	tt2488496	11.173104	200000000	2068178225	Star Wars: The Force Awakens	Harrison Ford|Mark Hamill|Carrie Fisher|Adam ...	http://www.starwarsmovie.com/	J. J. Abrams	Every generation has a story.	rescue|mystery|reunion|resistance|first order...	Episode VII - Thirty years after defeating the ...	136	Action|Adventure|Science Fiction|Fantasy	Lucamount|Truenorth Productions|Bad Robot	12/15/15	4636	7.5	2015	191999994.5	2068178225
4	168259	tt3659388	10.739009	135000000	1363529905	Furious 7	Vin Diesel|Paul Walker|Jason Statham|Michelle ...	http://www.furious7.com/	James Wan	Vengeance hits home	finance|vehicle|airplane|actress|flashback|...	Deckard Shaw seeks revenge against Dominic To...	137	Action|Crime|Thriller	Universal|Original Film|Media Rights Capital|...	4/1/15	2947	7.3	2015	124999994.4	1363529905

5 rows × 21 columns

```
In [3]: # Check for any missing data and how many
        df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10866 entries, 0 to 10865
Data columns (total 21 columns):
id                      10866 non-null  int64
imdb_id                 10856 non-null  object
popularity              10866 non-null  float64
budget                  10866 non-null  int64
revenue                 10866 non-null  int64
original_title          10866 non-null  object
cast                    10790 non-null  object
homepage                2936 non-null   object
director                10822 non-null  object
tagline                 8042 non-null   object
keywords                7973 non-null   object
overview                10862 non-null  object
runtime                 10866 non-null  int64
genres                  10843 non-null  object
production_companies    9836 non-null   object
release_date            10866 non-null  object
vote_count              10866 non-null  int64
vote_average            10866 non-null  float64
release_year            10866 non-null  int64
budget_adj              10866 non-null  float64
revenue_adj             10866 non-null  float64
dtypes: float64(6), int64(6), object(9)
memory usage: 1.7 MB
```

> Generally, imdb_id, cast, homepage, director, tagline, keywords, overview, genres and production_companies these are the columns with missing values.

```
In [4]: df.describe()
```

```
                 id    popularity        budget       revenue       runtime  vote_count  vote_average  release_year    budget_adj   revenue_adj
count  10866.000000  10866.000000  10866.000000  10866.000000  10866.000000  10866.000000  10866.000000  10866.000000  10866.000000  10866.000000
mean   66064.177434      0.646441  14625701.094789  39826885.760203    102.070863    217.389748      5.975012   2001.322658  17560799.988288  51364363.240020
std    92130.136561      1.000231  30913213.831336  117003486.582501     31.381405    575.619058      0.935142      12.812941  34307794.039619  144632428.394948
min        5.000000      0.000065      0.000000      0.000000      0.000000      10.000000      1.500000   1960.000000      0.000000      0.000000
25%    10596.250000      0.207583      0.000000      0.000000     90.000000      17.000000      5.400000   1995.000000      0.000000      0.000000
50%    20669.000000      0.383856      0.000000      0.000000    104.000000      38.000000      6.000000   2006.000000      0.000000      0.000000
75%    75610.000000      0.713817  15000000.000000  24000000.000000    112.000000     145.750000      6.600000   2011.000000  20853251.079250  33697095.720500
max   417859.000000     32.985763 425000000.000000 2781505847.000000    900.000000    9767.000000      8.400000   2015.000000 425000000.000000 2827123750.410000
```

> 10856 imdb_id missing, plus we only can correct them or we may drop them since we don't need them certainly.

```
In [5]: df.dtypes
```

```
id                      int64
imdb_id                 object
popularity              float64
budget                  int64
revenue                 int64
original_title          object
cast                    object
homepage                object
director                object
tagline                 object
keywords                object
overview                object
runtime                 int64
genres                  object
production_companies    object
release_date            object
vote_count              int64
vote_average            float64
release_year            int64
budget_adj              float64
revenue_adj             float64
dtypes: float64(6), int64(6), object(9)
memory usage: 1.7 MB
```

```
In [6]: df.nunique()
```

```
id                      10865
imdb_id                 10855
popularity              10724
budget                    563
revenue                   4702
original_title           9629
cast                     9773
homepage                 2896
director                 5067
tagline                  7805
keywords                 8415
overview                10847
runtime                   247
genres                   2039
production_companies     7445
release_date             5909
vote_count               1289
vote_average               72
release_year               56
budget_adj              2614
revenue_adj             4840
dtypes: int64    6
        object   9
        float    6
dtype: int64
```

Data Cleaning (Replace, Null Values, Wrong Types, unnecessary columns)

Dal will prompt unwanted columns, then check on new dataset.

```
In [7]: # drop unused columns, then check on new dataset
        df.drop('imdb_id', inplace=True)
        df.drop('cast', inplace=True)
        df.drop('homepage', inplace=True)
        df.drop('tagline', inplace=True)
        df.drop('keywords', inplace=True)
        df.drop('overview', inplace=True)
        df.drop('production_companies', inplace=True)
        df.drop('release_date', inplace=True)
        df.drop('budget_adj', inplace=True)
        df.drop('revenue_adj', inplace=True)
        df.head(5)
```

Out[7]:

	id	popularity	budget	revenue	original_title	director	runtime	genres	vote_count	vote_average	release_year
0	135397	32.985763	150000000	1513528810	Jurassic World	Colin Trevorrow	124	Action|Adventure|Science Fiction|Thriller	5562	6.5	2015
1	76341	28.419936	150000000	378436354	Mad Max: Fury Road	George Miller	120	Action|Adventure|Science Fiction|Thriller	6071	7.1	2015
2	262500	13.112507	110000000	295238201	Insurgent	Robert Schwentke	119	Adventure|Science Fiction|Thriller	2908	6.4	2015
3	140607	11.173104	200000000	2068178225	Star Wars: The Force Awakens	J. J. Abrams	136	Action|Adventure|Science Fiction|Fantasy	4636	7.5	2015
4	168259	10.739009	135000000	1363529905	Furious 7	James Wan	137	Action|Crime|Thriller	2947	7.3	2015

Number of duplicates

```
In [8]: # Number of duplicated rows
        sum(df.duplicated())
```

```
Out[8]: 1
```

Drop duplicated rows check.

```
In [9]: # drop duplicated rows
        df.drop_duplicates(inplace=True)
        sum(df.duplicated())
```

```
Out[9]: 0
```

Drop null values.

```
In [10]: # drop null values since they are not numeric & don't suggest filling them with the mean
         df.dropna(inplace=True)
         df.shape
```

```
Out[10]: (10834, 12)
```

Check on null values.

```
In [11]: df.isnull().sum()
```

```
Out[11]: id            0
         popularity    0
         budget        0
         revenue       0
         original_title 0
         director      0
         runtime       0
         genres        0
         vote_count    0
         vote_average  0
         release_year  0
         dtype: int64
```

Exploratory Data Analysis

Research Question 1 Which genres are most popular from year to year?

We can see genres count in every year before piping.

```
In [12]: # compute the genre frequency, split on '|'
         df['genres'] = df['genres'].str.split('|')
         df.df
```

```
genre_df = pd.DataFrame({'genre': genre,
                         'id': id,
                         'popularity': popularity,
                         'budget': budget,
                         'revenue': revenue,
                         'original_title': original_title,
                         'director': director,
                         'runtime': runtime,
                         'vote_count': vote_count,
                         'vote_average': vote_average,
                         'release_year': release_year})
genre_df.head(5)
```

Out[12]:

	genre	id	popularity	budget	revenue	original_title	director	runtime	vote_count	vote_average	release_year
0	Action	135397	32.985763	150000000	1513528810	Jurassic World	Colin Trevorrow	124	5562	6.5	2015
1	Adventure	135397	32.985763	150000000	1513528810	Jurassic World	Colin Trevorrow	124	5562	6.5	2015
2	Science Fiction	135397	32.985763	150000000	1513528810	Jurassic World	Colin Trevorrow	124	5562	6.5	2015
3	Thriller	135397	32.985763	150000000	1513528810	Jurassic World	Colin Trevorrow	124	5562	6.5	2015
4	Action	76341	28.419936	150000000	378436354	Mad Max: Fury Road	George Miller	120	6071	7.1	2015

```
In [13]: # Count the genre
         genre = genre_df['genre'].value_counts()
         genre.head(5)
```

```
Out[13]: Drama        4761
         Comedy       3793
         Thriller     2907
         Action       2384
         Adventure    1471
         Name: genre, dtype: int64
```

```
In [14]: # Let group genre by release_year
         genre = genre_df.groupby('release_year')['genre'].value_counts()
         # Let's see what genre was popular in year release_year
         genre[2010].head(5)
         genre[2011].head(5)
         genre[2012].head(5)
         genre[2013].head(5)
         genre[2014].head(5)
         genre[2015].head(5)
         genre[2015].tail(5)
         genre[2015].shape
```

```
release_year  genre
2015          Drama          1 0
              Comedy         1 0
              Thriller       1 0
              Action         1 0
              Adventure      1 0
              Documentary    1 0
              Science Fiction1 0
              Crime          1 0
              Family         1 0
              Fantasy        1 0
              Foreign        1 0
              History        1 0
              Music          1 0
              Mystery        1 0
              TV Movie       1 0
              Western        1 0
              dtype: int64
```

Top genres in each year

```
In [15]: # Let group genre by release_year
         genre = genre_df.groupby('release_year')['genre'].value_counts()
         # Let's see what genre was popular in year release_year
         genre[2010].head(1)
         genre[2011].head(1)
         genre[2012].head(1)
         genre[2013].head(1)
         genre[2014].head(1)
         genre[2015].head(1)
         genre[2015].tail(1)
         genre[2015].shape
```

```
release_year  genre
2015          Drama    1 0
              dtype: int64
```

Adventure has the most popularity in 15 year.

```
In [16]: # Let group genre by year to see change in score and popularity
         genre_pop = genre_df.groupby('genre')
         genre_pop.mean()
```

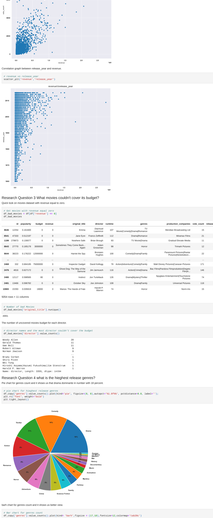
```
                 id  popularity        budget       revenue       runtime  vote_count  vote_average  release_year
genre
Action    101000.1    0.969398  29165159.148  72564351.108    108.399  1161.159732      5.732146   2000.932
Adventure 101000.1    1.035283  32779671.301  88389765.315    107.376  1253.311471      5.873556   2000.873
Animation 101000.1    0.877482  26955516.129  73649163.670    106.481  1099.862366      6.296774   2000.957
Comedy    101000.1    0.590222  15851650.439  40686817.650    103.351   387.975307      5.920623   2000.975
Crime     101000.1    0.742253  19789932.773  52616865.897    109.696   670.203843      5.976353   2000.796
Documentary101000.1   0.283987   1041987.263   2161420.419    100.962    75.138889      7.346154   2000.939
Drama     101000.1    0.560345  14068687.689  36687740.602    107.083   329.635835      6.153207   2000.887
Family    101000.1    0.699456  17926936.651  55578050.188    100.651   534.192565      5.945419   2000.961
Fantasy   101000.1    0.644487  14988100.000  40351509.231    105.791   292.357671      5.851852   2000.970
Foreign   101000.1    0.192494   364907.407    424212.074    119.980    14.111111      6.377778   2000.925
History   101000.1    0.603731  12375749.375  34772913.965    127.775   293.543758      6.436893   2000.961
Music     101000.1    0.490314  14369966.667  51369999.167    109.486   151.401515      6.070455   2000.909
Mystery   101000.1    0.644009  14331360.515  39272827.340    106.836   281.988823      6.071754   2000.950
TV Movie  101000.1    0.059232    159776.471    139000.000     81.099    30.142857      5.571429   2000.714
Western   101000.1    0.574821  17654912.500  34848301.333    116.465   219.333333      5.838462   2000.973
```

```
In [17]: genre_pop.mean().shape
```

```
Out[17]: (8, 8)
```

The graph shows as the most genres from year to year.

```
In [18]: # Let plot it
         genre_pop = genre_df.groupby('genre')['release_year', 'popularity', 'budget', 'revenue', 'vote_count', 'vote_average', 'runtime'].mean()
         # Plot the figure and legend
         plt.figure(figsize=(15,8))
         sns.barplot(genre_pop.index, genre_pop['popularity'], palette='Paired')
         plt.legend(genre_pop.index, loc='upper right')
         plt.title('Genre Popularity')
         plt.xlabel('Genre')
         plt.ylabel('Popularity')
         plt.show()
```



Research Question 2 What kinds of properties are associated with movies that have high revenues?

Correlation based on revenue.

```
In [19]: # Let's look at revenue
         genre_pop.corr()['revenue'].sort_values(ascending=False), indicator=True
```

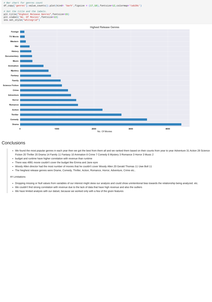
```
Out[19]:
                 id  popularity        budget       revenue       runtime  vote_count  vote_average  release_year
genre
Adventure 101000.1    1.035283  32779671.301  88389765.315    107.376  1253.311471      5.873556   2000.873
SCI-FI    101000.1    1.000231  30913213.831 117003486.583     31.381   575.619058      0.935142     12.813
Action    101000.1    0.969398  29165159.148  72564351.108    108.399  1161.159732      5.732146   2000.932
Thriller  101000.1    0.742253  19789932.773  52616865.897    109.696   670.203843      5.976353   2000.796
Fantasy   101000.1    0.644487  14988100.000  40351509.231    105.791   292.357671      5.851852   2000.970
Crime     101000.1    0.742253  19789932.773  52616865.897    109.696   670.203843      5.976353   2000.796
Western   101000.1    0.574821  17654912.500  34848301.333    116.465   219.333333      5.838462   2000.973
Drama     101000.1    0.560345  14068687.689  36687740.602    107.083   329.635835      6.153207   2000.887
Romance   101000.1    0.560345  14068687.689  36687740.602    107.083   329.635835      6.153207   2000.887
Family    101000.1    0.699456  17926936.651  55578050.188    100.651   534.192565      5.945419   2000.961
Comedy    101000.1    0.590222  15851650.439  40686817.650    103.351   387.975307      5.920623   2000.975
Mystery   101000.1    0.644009  14331360.515  39272827.340    106.836   281.988823      6.071754   2000.950
Music     101000.1    0.490314  14369966.667  51369999.167    109.486   151.401515      6.070455   2000.909
Horror    101000.1    0.414662  11232576.923  29232576.923    100.462   202.230769      5.569231   2000.962
War       101000.1    0.564383  18436954.667  39823319.667    134.667   238.333333      6.433333   2000.667
Animation 101000.1    0.877482  26955516.129  73649163.670    106.481  1099.862366      6.296774   2000.957
Documentary101000.1   0.283987   1041987.263   2161420.419    100.962    75.138889      7.346154   2000.939
TV Movie  101000.1    0.059232    159776.471    139000.000     81.099    30.142857      5.571429   2000.714
Foreign   101000.1    0.192494   364907.407    424212.074    119.980    14.111111      6.377778   2000.925
History   101000.1    0.603731  12375749.375  34772913.965    127.775   293.543758      6.436893   2000.961
Music     101000.1    0.490314  14369966.667  51369999.167    109.486   151.401515      6.070455   2000.909
Mystery   101000.1    0.644009  14331360.515  39272827.340    106.836   281.988823      6.071754   2000.950
TV Movie  101000.1    0.059232    159776.471    139000.000     81.099    30.142857      5.571429   2000.714
Western   101000.1    0.574821  17654912.500  34848301.333    116.465   219.333333      5.838462   2000.973
```

```
In [20]: # Let's look at correlation
         genre_pop.corr()
```

```
                 id  popularity        budget       revenue       runtime  vote_count  vote_average  release_year
genre
Adventure 101000.1    1.035283  32779671.301  88389765.315    107.376  1253.311471      5.873556   2000.873
SCI-FI    101000.1    1.000231  30913213.831 117003486.583     31.381   575.619058      0.935142     12.813
Action    101000.1    0.969398  29165159.148  72564351.108    108.399  1161.159732      5.732146   2000.932
Thriller  101000.1    0.742253  19789932.773  52616865.897    109.696   670.203843      5.976353   2000.796
Fantasy   101000.1    0.644487  14988100.000  40351509.231    105.791   292.357671      5.851852   2000.970
Crime     101000.1    0.742253  19789932.773  52616865.897    109.696   670.203843      5.976353   2000.796
Western   101000.1    0.574821  17654912.500  34848301.333    116.465   219.333333      5.838462   2000.973
Drama     101000.1    0.560345  14068687.689  36687740.602    107.083   329.635835      6.153207   2000.887
Romance   101000.1    0.560345  14068687.689  36687740.602    107.083   329.635835      6.153207   2000.887
Family    101000.1    0.699456  17926936.651  55578050.188    100.651   534.192565      5.945419   2000.961
Comedy    101000.1    0.590222  15851650.439  40686817.650    103.351   387.975307      5.920623   2000.975
Mystery   101000.1    0.644009  14331360.515  39272827.340    106.836   281.988823      6.071754   2000.950
Music     101000.1    0.490314  14369966.667  51369999.167    109.486   151.401515      6.070455   2000.909
Horror    101000.1    0.414662  11232576.923  29232576.923    100.462   202.230769      5.569231   2000.962
War       101000.1    0.564383  18436954.667  39823319.667    134.667   238.333333      6.433333   2000.667
Animation 101000.1    0.877482  26955516.129  73649163.670    106.481  1099.862366      6.296774   2000.957
Documentary101000.1   0.283987   1041987.263   2161420.419    100.962    75.138889      7.346154   2000.939
TV Movie  101000.1    0.059232    159776.471    139000.000     81.099    30.142857      5.571429   2000.714
Foreign   101000.1    0.192494   364907.407    424212.074    119.980    14.111111      6.377778   2000.925
History   101000.1    0.603731  12375749.375  34772913.965    127.775   293.543758      6.436893   2000.961
Music     101000.1    0.490314  14369966.667  51369999.167    109.486   151.401515      6.070455   2000.909
Mystery   101000.1    0.644009  14331360.515  39272827.340    106.836   281.988823      6.071754   2000.950
TV Movie  101000.1    0.059232    159776.471    139000.000     81.099    30.142857      5.571429   2000.714
Western   101000.1    0.574821  17654912.500  34848301.333    116.465   219.333333      5.838462   2000.973
```

Correlation graph between popularity and revenue.

```
In [21]: # Make a scatter plot
         sns.pairplot(genre_pop, x_vars=['popularity'], y_vars=['revenue'])
```



Correlation graph between vote_average and revenue.

```
In [22]: # Another scatter plot
         sns.pairplot(genre_pop, x_vars=['vote_average'], y_vars=['revenue'])
```


Correlation graph between release_year and revenue.

```
In [23]: # Another scatter plot
         sns.pairplot(genre_pop, x_vars=['release_year'], y_vars=['revenue'])
```


Research Question 3 What movies couldn't cover its budget?

Look at a random sample of 20 movies.

```
In [24]: # Let's look at a random sample of 20 movies
         df.sample(20)
         df.shape
         df[df['revenue'] < 0].shape
         df[df['revenue'] == 0].shape
```

```
Out[24]: (10834, 12)
         (0, 12)
         (5696, 12)
```

```
In [25]: # Number of movies budget exceeds revenue
         df.sample(20)
         df[df['revenue'] < df['budget']].shape
```

```
Out[25]: (5696, 12)
```

Research Question 4 Who is the highest release genres?

Find top genres count and it shows on bar chart.

```
In [26]: # Let's look at top genre
         df.sample(20)
         df['genres'].value_counts().head(10).plot(kind='pie', figsize=(8,8), autopct='%1.0f%%', pctdistance=0.8, labels=[''])
         # Let's add a legend
         plt.legend(['Drama', 'Comedy', 'Thriller', 'Action', 'Adventure', 'Horror', 'Documentary', 'Crime', 'Animation', 'Family'])
         # Let's add a title
         plt.title('Top 10 Genres')
         # Let's add a circle
         from matplotlib.patches import Circle
         my_circle = Circle((0,0), 0.70, fc='white')
         plt.gca().add_patch(my_circle)
         plt.show()
```



```
In [27]: # Let's look at highest release genres
         df.sample(20)
         df.groupby('genres')['release_year'].count().sort_values(ascending=False).head(10).plot(kind='barh')
         plt.xlabel('Top 10 Genres')
         plt.ylabel('Release Year')
         plt.title('Highest Release Genres')
         plt.show()
```


Conclusions

> In this section, you should continue to explore the data to address your additional questions. Hypothesis testing, multivariate exploration, and machine learning are all options for further exploration. Also, you should draw conclusions about your data, and discuss any limitations or shortcomings in your analysis.

- We found the most popular genres in each year then we get the best from them all and we ranked them based on their counts from year to year.Adventure is Action 24 Science Fiction 23 Thriller 22 Crime 21 Drama 20 Fantasy 19 Western 18 Animation 17 Comedy 16 Mystery 15 Romance 14 Horror 13 Music 12 Documentary 11 Foreign 10 Family 9 History 8 War 7 TV Movie 6 SCI-FI 5
- Budget and runtime have higher correlation with revenue than votes.
- Budget and runtime have higher correlation with popularity than votes.
- Woody Allen directed had the most number of movies that he couldn't cover Woody Allen 32 Clint Eastwood 31 Steven Soderbergh 27 Tim Burton 23 Martin Scorsese 20 Spike Lee 19 Ridley Scott 18 Robert Rodriguez 17 Terry Gilliam 16 Oliver Stone 15 Brian De Palma 14 Joel Coen 13 Joe Dante 12
- Universal Pictures 1 Universal Pictures 2 Universal Pictures 3 Universal Pictures 4 Universal Pictures 5 Universal Pictures 6 Universal Pictures 7 Universal Pictures 8 Universal Pictures 9 Universal Pictures 10 Universal Pictures 11 Universal Pictures 12 Universal Pictures 13 Universal Pictures 14 Universal Pictures 15 Universal Pictures 16 Universal Pictures 17 Universal Pictures 18 Universal Pictures 19 Universal Pictures 20 Universal Pictures 21 Universal Pictures 22 Universal Pictures 23 Universal Pictures 24 Universal Pictures 25 Universal Pictures 26 Universal Pictures 27 Universal Pictures 28 Universal Pictures 29 Universal Pictures 30 Universal Pictures 31 Universal Pictures 32
- Universal Pictures 1 Universal Pictures 2 Universal Pictures 3 Universal Pictures 4 Universal Pictures 5 Universal Pictures 6 Universal Pictures 7 Universal Pictures 8 Universal Pictures 9 Universal Pictures 10 Universal Pictures 11 Universal Pictures 12 Universal Pictures 13 Universal Pictures 14 Universal Pictures 15 Universal Pictures 16 Universal Pictures 17 Universal Pictures 18 Universal Pictures 19 Universal Pictures 20 Universal Pictures 21 Universal Pictures 22 Universal Pictures 23 Universal Pictures 24 Universal Pictures 25 Universal Pictures 26 Universal Pictures 27 Universal Pictures 28 Universal Pictures 29 Universal Pictures 30 Universal Pictures 31 Universal Pictures 32
- Universal Pictures 1 Universal Pictures 2 Universal Pictures 3 Universal Pictures 4 Universal Pictures 5 Universal Pictures 6 Universal Pictures 7 Universal Pictures 8 Universal Pictures 9 Universal Pictures 10 Universal Pictures 11 Universal Pictures 12 Universal Pictures 13 Universal Pictures 14 Universal Pictures 15 Universal Pictures 16 Universal Pictures 17 Universal Pictures 18 Universal Pictures 19 Universal Pictures 20 Universal Pictures 21 Universal Pictures 22 Universal Pictures 23 Universal Pictures 24 Universal Pictures 25 Universal Pictures 26 Universal Pictures 27 Universal Pictures 28 Universal Pictures 29 Universal Pictures 30 Universal Pictures 31 Universal Pictures 32
- Universal Pictures 1 Universal Pictures 2 Universal Pictures 3 Universal Pictures 4 Universal Pictures 5 Universal Pictures 6 Universal Pictures 7 Universal Pictures 8 Universal Pictures 9 Universal Pictures 10 Universal Pictures 11 Universal Pictures 12 Universal Pictures 13 Universal Pictures 14 Universal Pictures 15 Universal Pictures 16 Universal Pictures 17 Universal Pictures 18 Universal Pictures 19 Universal Pictures 20 Universal Pictures 21 Universal Pictures 22 Universal Pictures 23 Universal Pictures 24 Universal Pictures 25 Universal Pictures 26 Universal Pictures 27 Universal Pictures 28 Universal Pictures 29 Universal Pictures 30 Universal Pictures 31 Universal Pictures 32