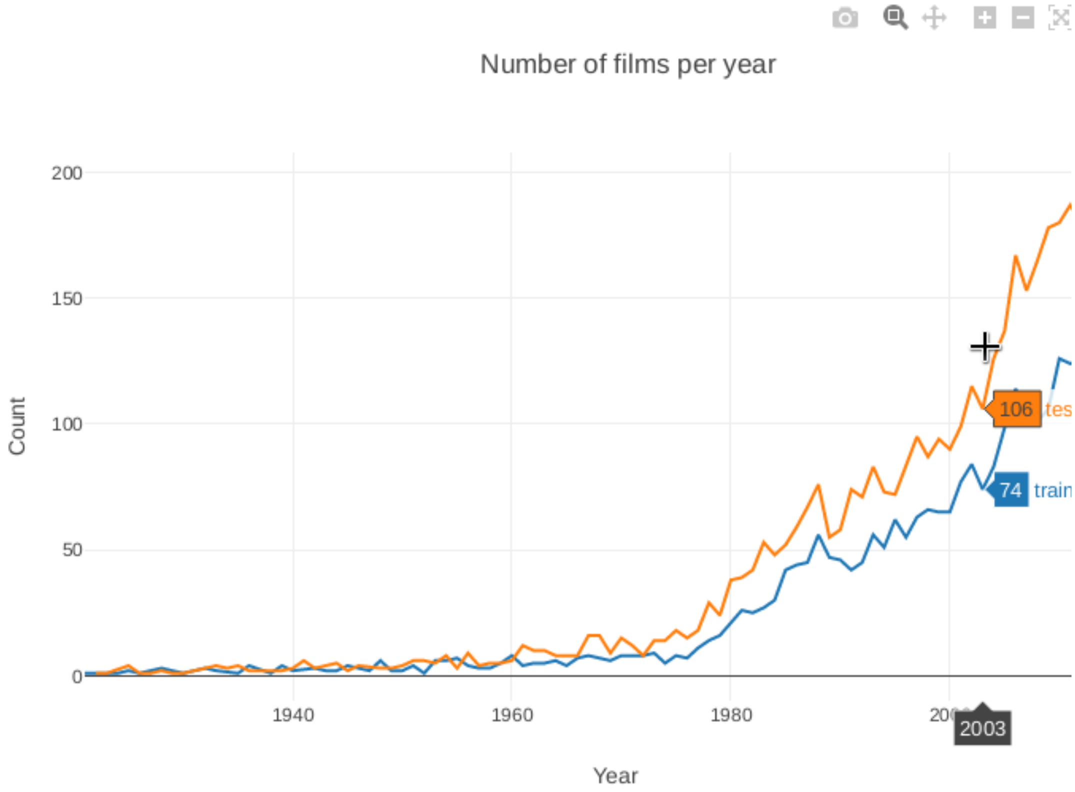
Analyze Worldwide Box Office Revenue with Python



This project is second in a series focused on data visualization with Plotly and S find the first [Project: Analyze Box Office Data with Seaborn and Python (https:// /learn/analyze-data-seaborn-python/)] on Coursera.

(Part 1) Libraries

In [1]:

```
import numpy as np
import pandas as pd
pd.set_option('max_columns', None)
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
plt.style.use('ggplot')
import datetime
import lightgbm as lgb
from scipy import stats
from scipy.sparse import hstack, csr_matrix
from sklearn.model_selection import train_test_split, KFold
from wordcloud import WordCloud
from collections import Counter
from nltk.corpus import stopwords
from nltk.util import ngrams
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer
from sklearn.preprocessing import StandardScaler
import nltk
nltk.download('stopwords')
stop = set(stopwords.words('english'))
import os
import plotly.offline as py
py.init_notebook_mode(connected=True)
import plotly.graph_objs as go
import plotly.tools as tls
import xgboost as xgb
import lightgbm as lgb
from sklearn import model_selection
from sklearn.metrics import accuracy_score
import json
import ast
from urllib.request import urlopen
from PIL import Image
from sklearn.preprocessing import LabelEncoder
import time
from sklearn.metrics import mean_squared_error
from sklearn.linear_model import LinearRegression
from sklearn import linear_model
```

```
[nltk_data] Downloading package stopwords to /home/rhyme/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

(Part 1) Data Loading and Exploration

In [28]:

```
train = pd.read_csv('data/train.csv')
test = pd.read_csv('data/test.csv')
```

In [3]:

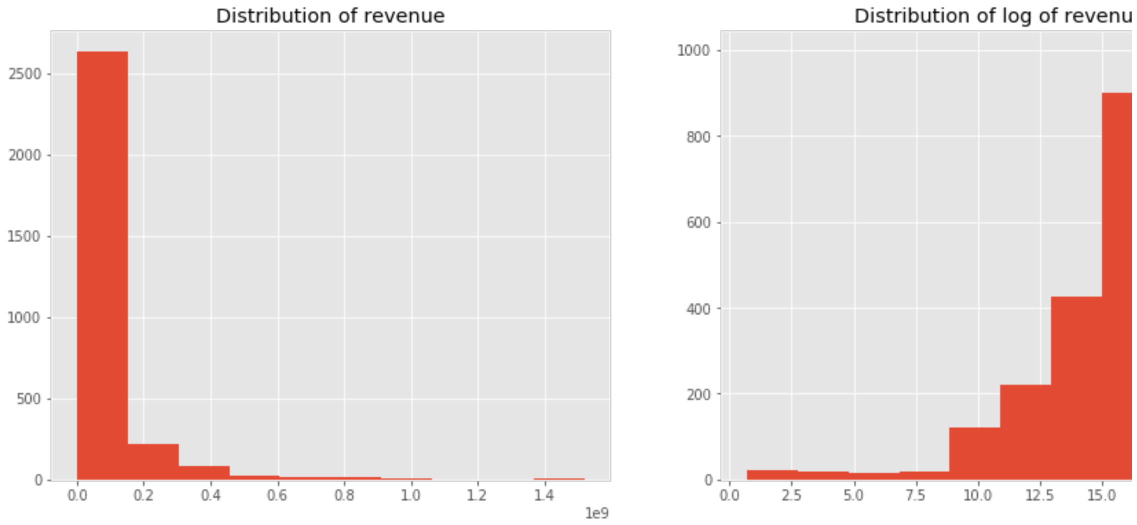
```
train.head()
```

	id	budget	homepage	imdb_id	original_language	original_title	overvie
0	1	14000000	NaN	tt2637294	en	Hot Tub Time Machine 2	When Lou who has become tl "father of the In...
1	2	40000000	NaN	tt0368933	en	The Princess Diaries 2: Royal Engagement	Mia Thermopo is now a college graduate and ...
2	3	3300000	http://sonyclassics.com/whiplash/	tt2582802	en	Whiplash	Under the direction o a ruthless instructor, ...
3	4	1200000	http://kahaanithefilm.com/	tt1821480	hi	Kahaani	Vidya Bagchi (Vidya Balan) arrives in Kolkata ...
4	5	0	NaN	tt1380152	ko	마린보이	Marine Bo is the stor of a forme national s

(Part 1) Visualizing the Target Distribution

In [4]:

```
fig, ax = plt.subplots(figsize = (16, 6))
plt.subplot(1, 2, 1)
plt.hist(train['revenue']);
plt.title('Distribution of revenue');
plt.subplot(1, 2, 2)
plt.hist(np.log1p(train['revenue']));
plt.title('Distribution of log of revenue');
```



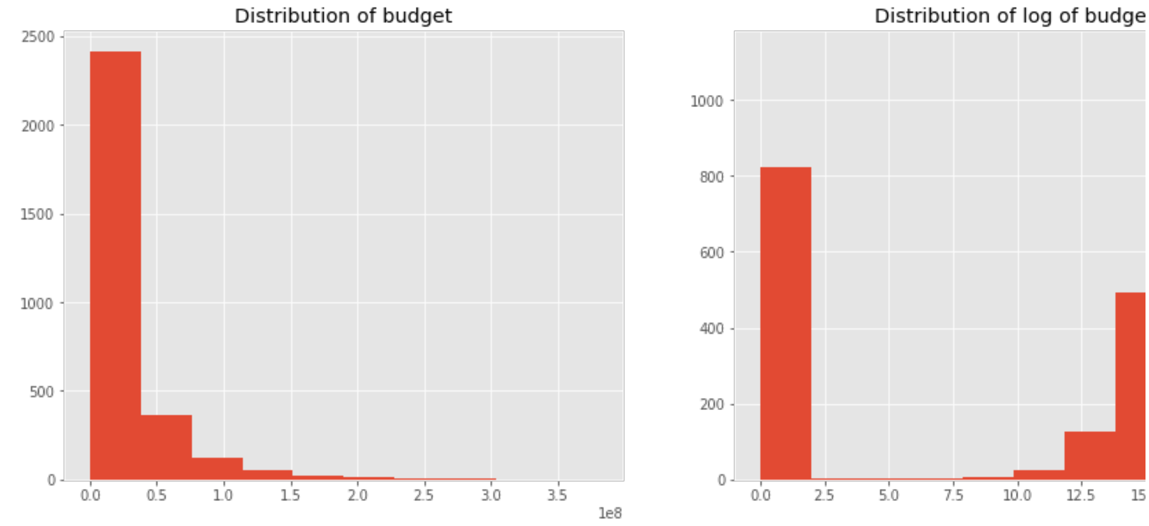
In [5]:

```
train['log_revenue'] = np.log1p(train['revenue'])
```

(Part 1) Relationship between Film Revenue and Budget

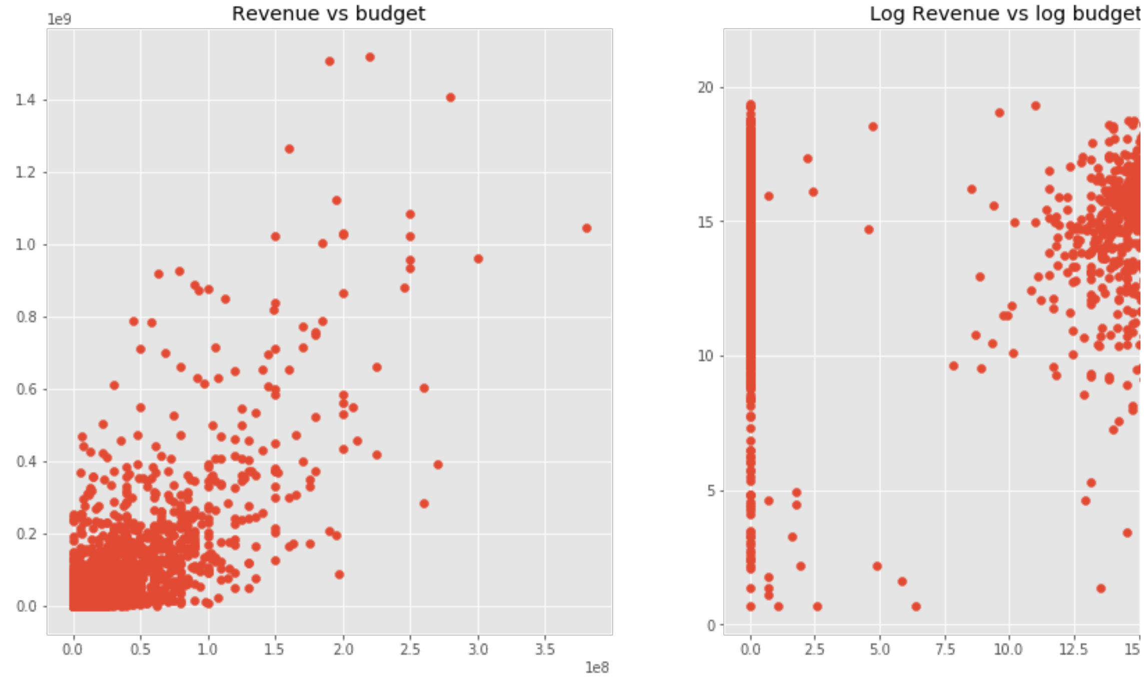
In [6]:

```
fig, ax = plt.subplots(figsize = (16, 6))
plt.subplot(1, 2, 1)
plt.hist(train['budget']);
plt.title('Distribution of budget');
plt.subplot(1, 2, 2)
plt.hist(np.log1p(train['budget']));
plt.title('Distribution of log of budget');
```



In [7]:

```
plt.figure(figsize=(16, 8))
plt.subplot(1, 2, 1)
plt.scatter(train['budget'], train['revenue'])
plt.title('Revenue vs budget');
plt.subplot(1, 2, 2)
plt.scatter(np.log1p(train['budget']), train['log_revenue'])
plt.title('Log Revenue vs log budget');
```



In [8]:

```
train['log_budget'] = np.log1p(train['budget'])
test['log_budget'] = np.log1p(test['budget'])
```

(Part 1) Does having an Official Homepage Affect Reven

In [9]:

```
train['homepage'].value_counts().head(10)
```

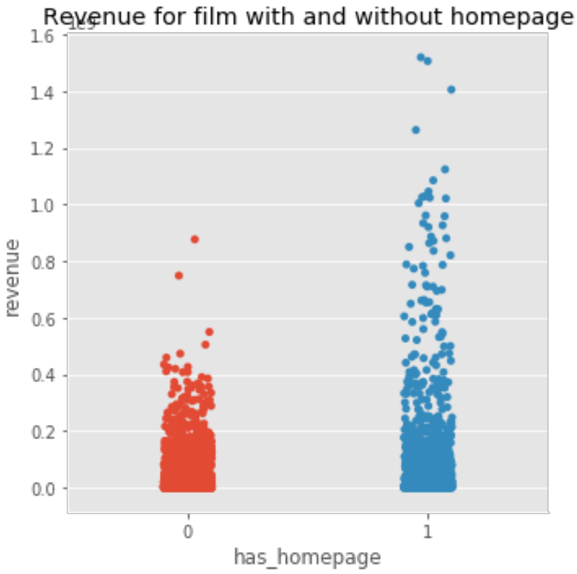
```
http://www.transformersmovie.com/ (http://www.transformersmovie.com/)                              4
http://www.thehobbit.com/ (http://www.thehobbit.com/)                                      2
http://www.lordoftherings.net/ (http://www.lordoftherings.net/)                            2
http://www.thefourthkind.net/ (http://www.thefourthkind.net/)                              1
http://www.miramax.com/movie/ready-to-wear/ (http://www.miramax.com/movie/ready-to-wear/)                1
http://www.foxmovies.com/movies/fight-club (http://www.foxmovies.com/movies/fight-club)             1
http://www.thx1138movie.com/ (http://www.thx1138movie.com/)                                 1
http://focusfeatures.com/its_kind_of_a_funny_story (http://focusfeatures.com/its_kind_of_a_funny_story)       1
http://cloudatlas.warnerbros.com/ (http://cloudatlas.warnerbros.com/)                       1
https://twitter.com/Stonewall_Movie (https://twitter.com/Stonewall_Movie)                     1
Name: homepage, dtype: int64
```

In [10]:

```
train['has_homepage'] = 0
train.loc[train['homepage'].isnull() == False, 'has_homepage'] = 1
test['has_homepage'] = 0
test.loc[test['homepage'].isnull() == False, 'has_homepage'] = 1
```

In [11]:

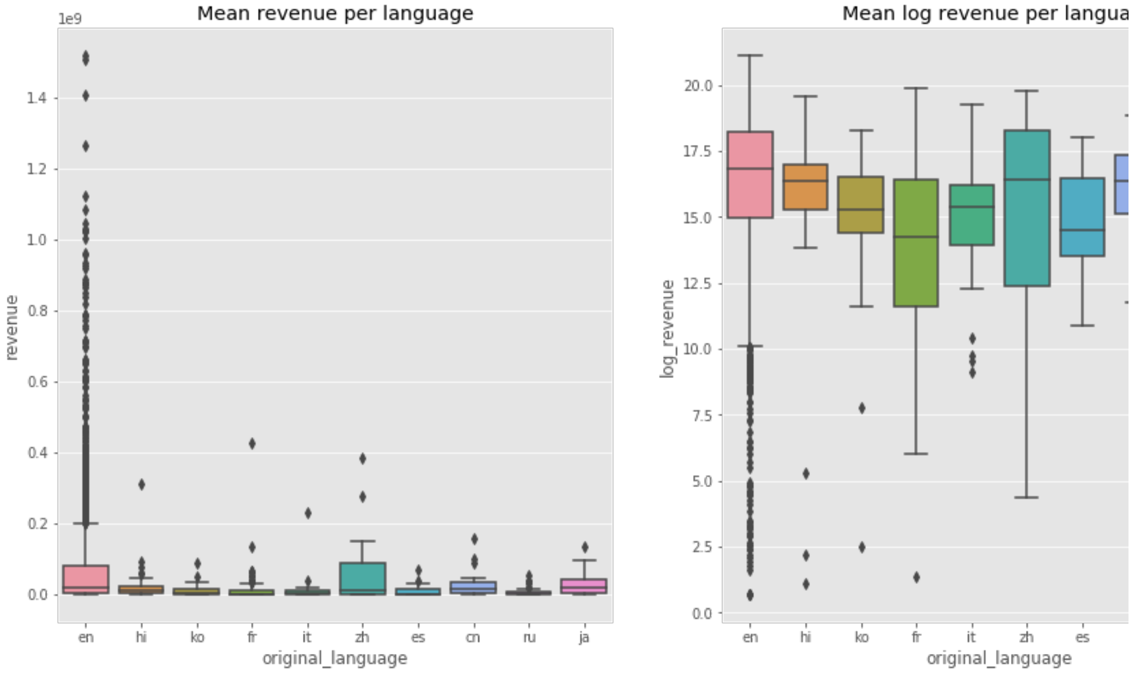
```
sns.catplot(x='has_homepage', y='revenue', data=train);
plt.title('Revenue for film with and without homepage');
```



(Part 1) Distribution of Languages in Film

In [12]:

```
plt.figure(figsize=(16, 8))
plt.subplot(1, 2, 1)
sns.boxplot(x='original_language', y='revenue', data=train.loc[train['original_
plt.title('Mean revenue per language');
plt.subplot(1, 2, 2)
sns.boxplot(x='original_language', y='log_revenue', data=train.loc[train['origi
plt.title('Mean log revenue per language');
```

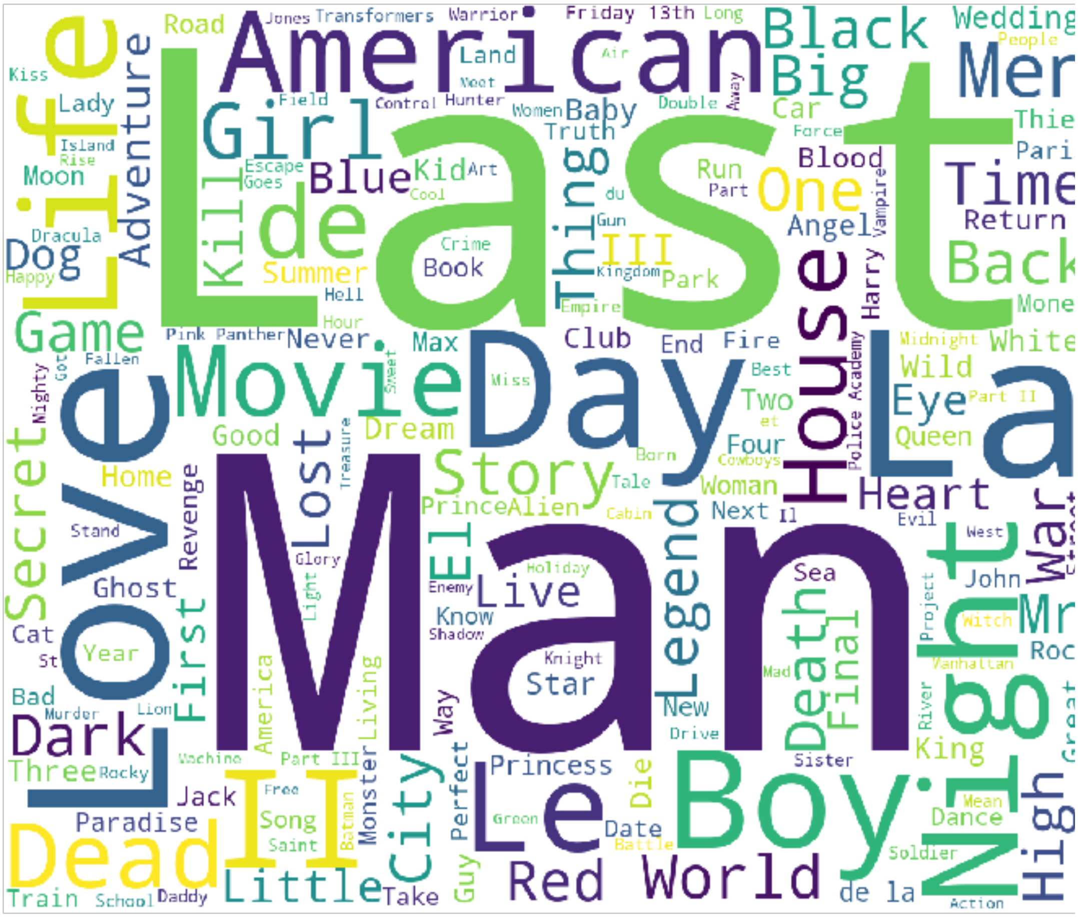


(Part 1) Frequent Words in Film Titles and Discriptions

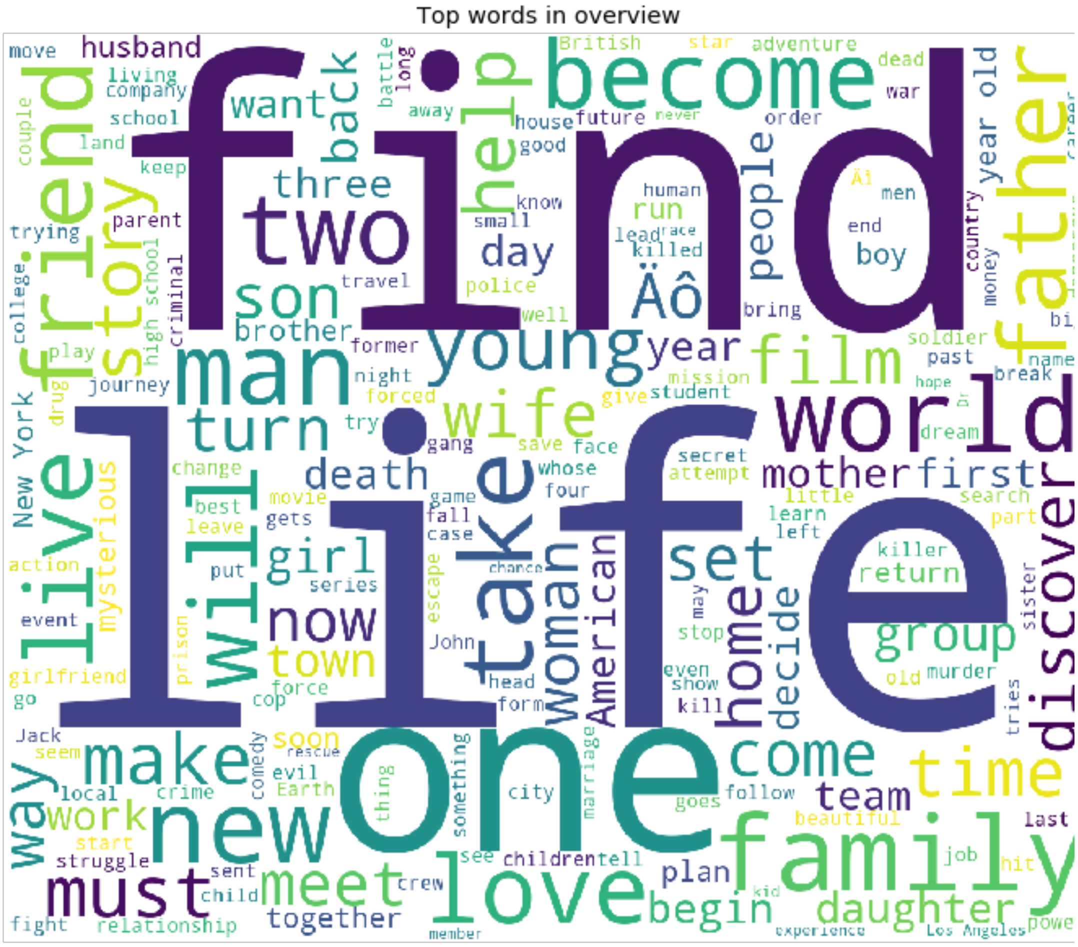
In [13]:

```
plt.figure(figsize = (12, 12))
text = ' '.join(train['original_title'].values)
wordcloud = WordCloud(max_font_size=None, background_color='white', width=1200,
plt.imshow(wordcloud)
plt.title('Top words in titles')
plt.axis("off")
plt.show()
```

Top words in titles



```
In [14]:  plt.figure(figsize = (12, 12))
          text = ' '.join(train['overview'].fillna('').values)
          wordcloud = WordCloud(max_font_size=None, background_color='white', width=1200,
          plt.imshow(wordcloud)
          plt.title('Top words in overview')
          plt.axis("off")
          plt.show()
```



(Part 1) Do Film Descriptions Impact Revenue?

In [15]:

```
import eli5

vectorizer = TfidfVectorizer(
            sublinear_tf=True,
            analyzer='word',
            token_pattern=r'\w{1,}',
            ngram_range=(1, 2),
            min_df=5)

overview_text = vectorizer.fit_transform(train['overview'].fillna(''))
linreg = LinearRegression()
linreg.fit(overview_text, train['log_revenue'])
eli5.show_weights(linreg, vec=vectorizer, top=20, feature_filter=lambda x: x !=
```

y top features

Weight?	Feature
+10.086	bombing
+9.263	complications
... 3742 more positive ...	
... 3431 more negative ...	
-9.170	critic
-9.195	status
-9.329	18
-9.356	politicians
-9.456	life they
-9.752	violence
-9.860	she will
-9.864	who also
-10.199	casino
-10.261	escape and
-10.427	receiving
-10.646	kept
-10.653	attracted to
-10.835	sally
-11.791	and be
-12.290	campaign
-13.815	mike
-15.395	woman from

In [16]:

```
print('Target value:', train['log_revenue'][1000])
eli5.show_prediction(linreg, doc=train['overview'].values[1000], vec=vectorizer
```

```
Target value: 16.44583954907521
```

y (score **16.446**) top features

Contribution?	Feature
+15.962	<BIAS>
+0.484	Highlighted in text (sum)

```
when elizabeth returns to her mother's home after her marriage breaks up, she recreates her imaginary cl
to escape from the trauma of losing her husband and her job. in between the chaos and mayhem that fred
attempts to win back her husband and return to normality.
```

Task 2: Analyzing Movie Release Dates

Note: If you are starting the notebook from this task, you can run cells from all in the kernel by going to the top menu and Kernel > Restart and Run All

In [29]:

```
test.loc[test['release_date'].isnull()== False, 'release_date'].head()
```

```
0    7/14/07
1    5/19/58
2    5/23/97
3     9/4/10
4    2/11/05
Name: release_date, dtype: object
```

Task 3: Preprocessing Features

In [30]:

```
def fix_date(x):
    year= x.split('/')[2]
    if int(year)<=19:
        return x[:-2] + '20' + year
    else:
        return x[:-2] +'19' + year
```

In [31]:

```
test.loc[test['release_date'].isnull()==True].head()
```

	id	budget	homepage	imdb_id	original_language	original_title	overview	popu
828	3829	0	NaN	tt0210130	en	Jails, Hospitals & Hip-Hop	Jails, Hospitals &amp; Hip-Hop is a cinematic ...	0.009

In [32]:

```
test.loc[test['release_date'].isnull()==True, 'release_date']= '05/01/00'
```

In [33]:

```
train['release_date']= train['release_date'].apply(lambda x: fix_date(x))
test['release_date']= test['release_date'].apply(lambda x: fix_date(x))
```

Task 4: Creating Features Based on Release Date

In [34]:

```
train['release_date']= pd.to_datetime(train['release_date'])
test['release_date']=pd.to_datetime(test['release_date'])
```

In [36]:

```
def process_date(df):
    date_parts= ['year','weekday','month','weekofyear','day','quarter']
    for part in date_parts:
        part_col = 'release_date' + '_' + part
        df[part_col] = getattr(df['release_date'].dt, part).astype(int)
        #from release date to date time format
    return df

train=process_date(train)
test= process_date(test)
```

Task 5: Using Plotly to Visualize the Number of Films Per

In [37]:

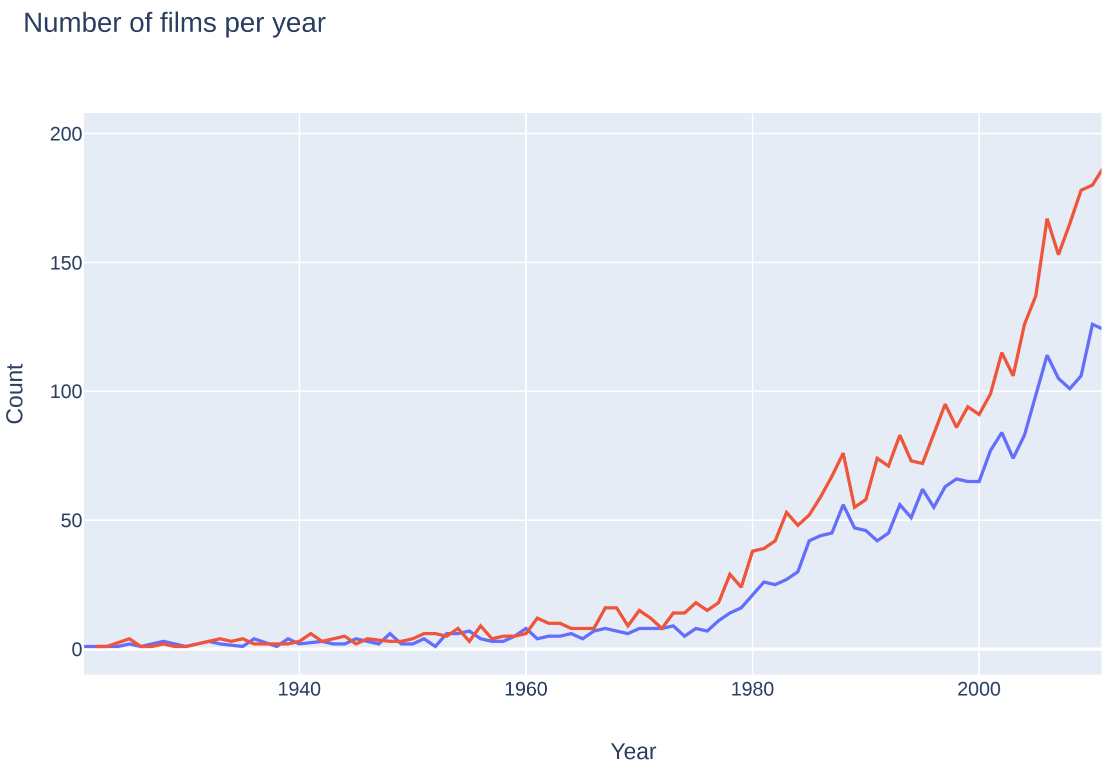
```
d1= train['release_date_year'].value_counts().sort_index()#asc
d2= test['release_date_year'].value_counts().sort_index()
```

In [38]:

```
import plotly.offline as py
py.init_notebook_mode(connected=True)
import plotly.graph_objs as go

data= [go.Scatter(x=d1.index, y=d1.values, name='train'),
      go.Scatter(x=d2.index, y=d2.values, name='test')]#index->year,counts->valu

layout= go.Layout(dict(title='Number of films per year',
                      xaxis=dict(title='Year'),
                      yaxis=dict(title='Count'),
                      ), legend=dict(orientation='v'))
py.iplot(dict(data=data, layout=layout))
```



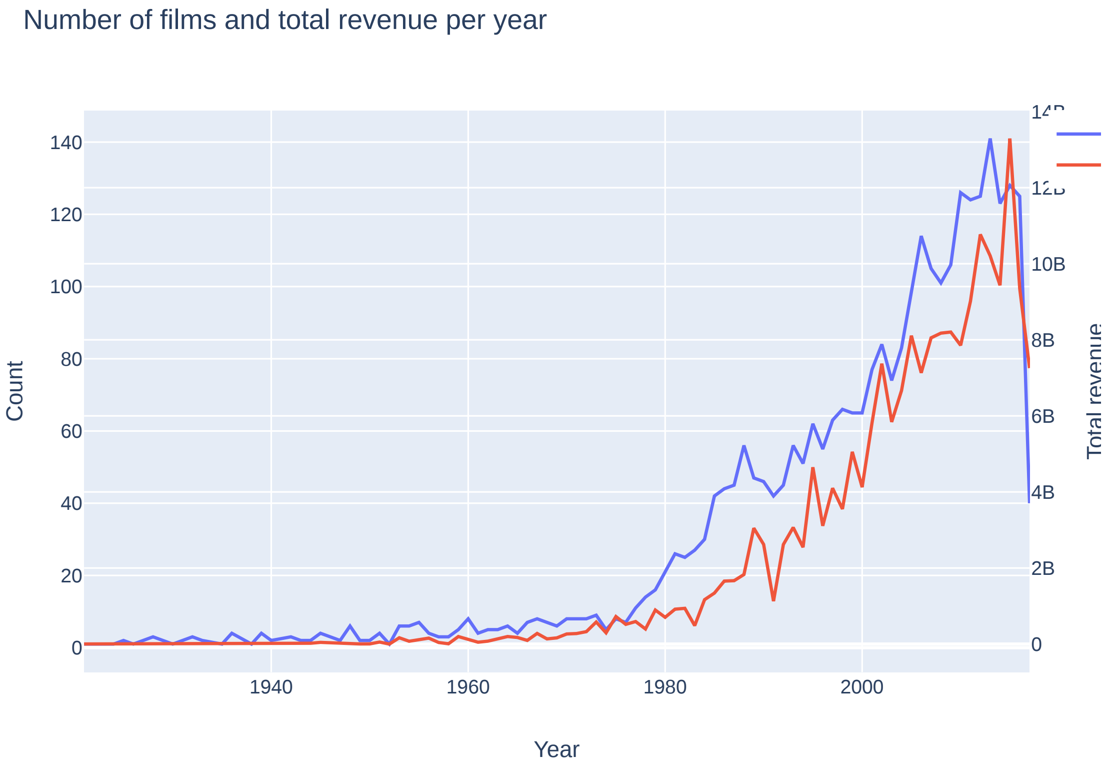
Task 6: Number of Films and Revenue Per Year

In [41]:

```
d1=  train['release_date_year'].value_counts().sort_index()
d2 = train.groupby(['release_date_year'])['revenue'].sum()

data= [go.Scatter(x=d1.index, y=d1.values, name='film count in a year'),
      go.Scatter(x=d2.index, y=d2.values, name='total revenue in a year', yaxis=

layout= go.Layout(dict(title='Number of films and total revenue per year',
                      xaxis=dict(title='Year'),
                      yaxis=dict(title='Count'),
                      yaxis2=dict(title='Total revenue', overlaying='y', side='
                  legend=dict(orientation='v'))
py.iplot(dict(data=data, layout=layout))
```

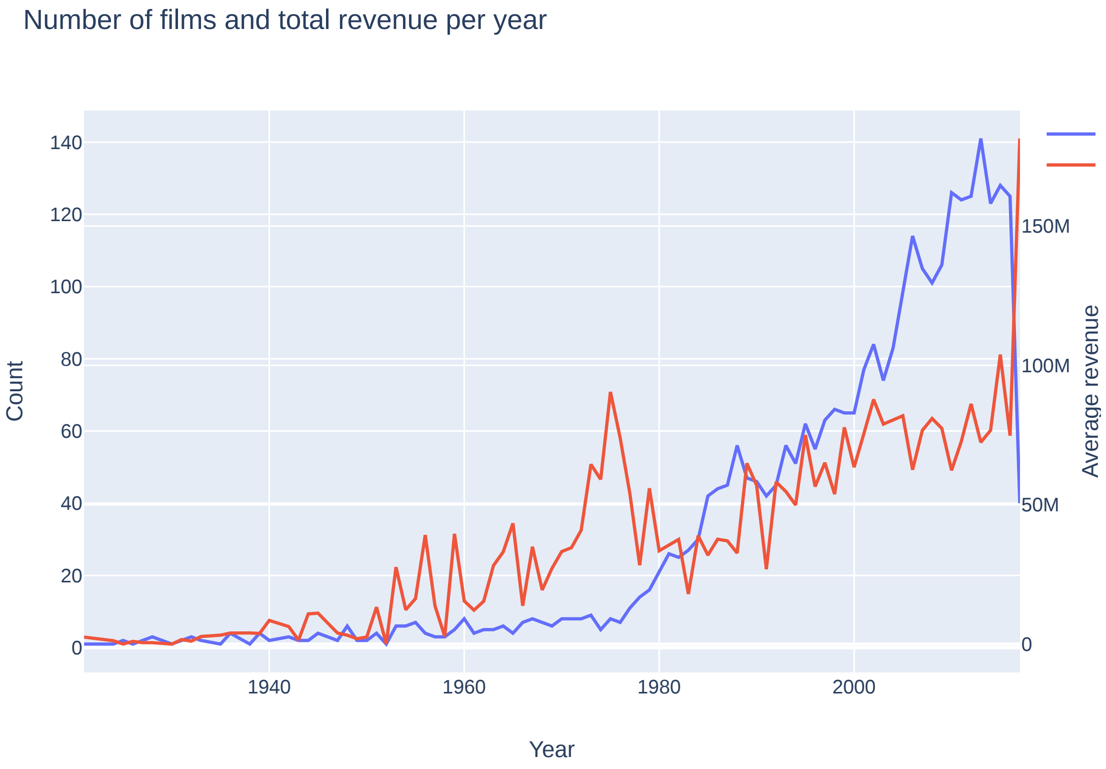


In [42]:

```
d1=  train['release_date_year'].value_counts().sort_index()
d2 = train.groupby(['release_date_year'])['revenue'].mean()

data= [go.Scatter(x=d1.index, y=d1.values, name='film count in a year'),
      go.Scatter(x=d2.index, y=d2.values, name='mean revenue in a year', yaxis=

layout= go.Layout(dict(title='Number of films and total revenue per year',
                      xaxis=dict(title='Year'),
                      yaxis=dict(title='Count'),
                      yaxis2=dict(title='Average revenue', overlaying='y', side=
                  legend=dict(orientation='v'))
py.iplot(dict(data=data, layout=layout))
```

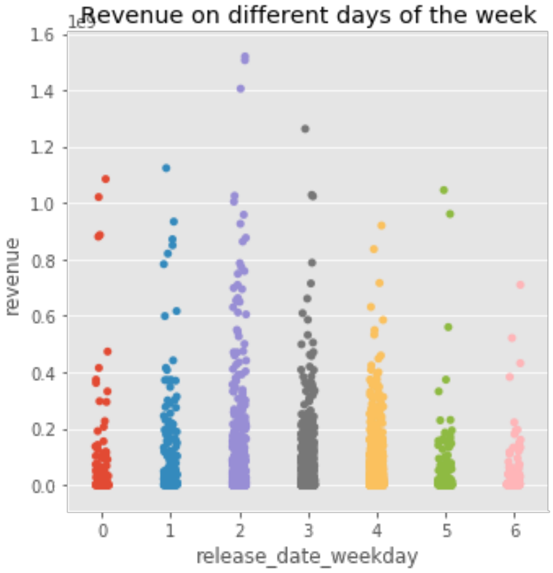


Task 7: Do Release Days Impact Revenue?

In [43]:

```
sns.catplot(x='release_date_weekday', y='revenue', data=train);
plt.title('Revenue on different days of the week')
```

```
Text(0.5, 1.0, 'Revenue on different days of the week')
```



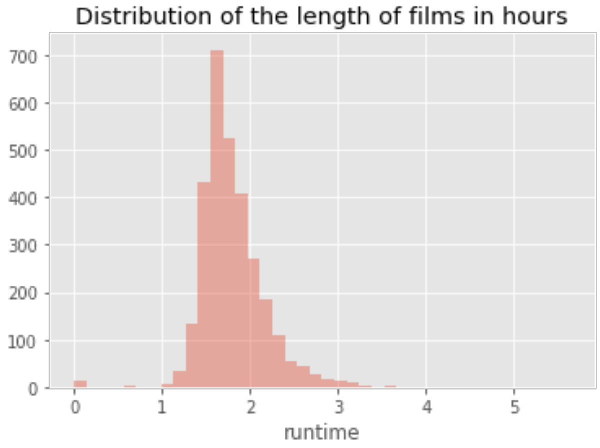
In [ ]:

Task 8: Relationship between Runtime and Revenue

In [45]:

```
sns.distplot(train['runtime'].fillna(0) / 60, bins=40, kde=False)
plt.title('Distribution of the length of films in hours')
```

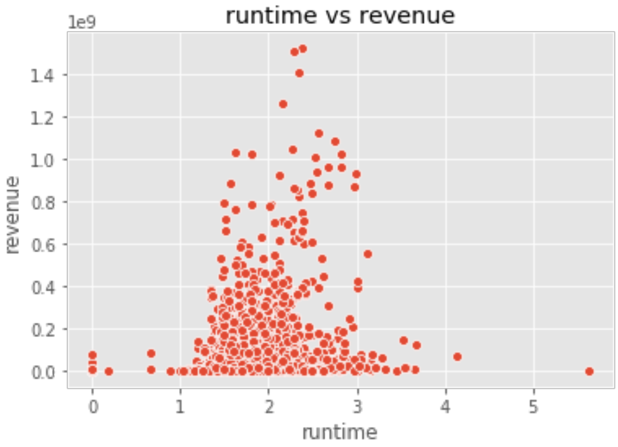
```
Text(0.5, 1.0, 'Distribution of the length of films in hours')
```



In [46]:

```
sns.scatterplot(train['runtime'].fillna(0) / 60, train['revenue'])
#or else write col names and specify data=train
plt.title('runtime vs revenue')
```

```
Text(0.5, 1.0, 'runtime vs revenue')
```


runtime vs revenue
1e9
revenue
runtime